(19) Deutsches Patent- und Markenamt

(10) **DE 11 2022 002 414 T5** 2024.03.21

(12) # Veröffentlichung

der internationalen Anmeldung mit der
(87) Veröffentlichungs-Nr.: **WO 2022/264034**
in der deutschen Übersetzung (Art. III § 8 Abs. 2 IntPatÜbkG)
(21) Deutsches Aktenzeichen: **11 2022 002 414.3**
(86) PCT-Aktenzeichen: **PCT/IB2022/055505**
(86) PCT-Anmeldetag: **14.06.2022**
(87) PCT-Veröffentlichungstag: **22.12.2022**
(43) Veröffentlichungstag der PCT Anmeldung in deutscher Übersetzung: **21.03.2024**

(51) Int Cl.: ***G06F 9/44*** (2018.01)
G06N 3/02 (2006.01)

(30) Unionspriorität:
17/350,467 **17.06.2021** **US**

(71) Anmelder:
International Business Machines Corporation, Armonk, NY, US

(74) Vertreter:
Richardt Patentanwälte PartG mbB, 65185 Wiesbaden, DE

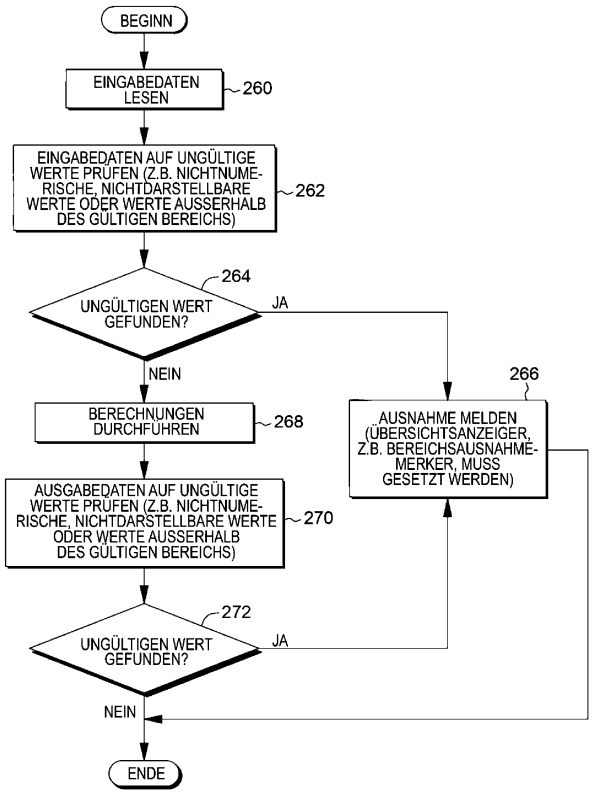
(72) Erfinder:
Albarakat, Laith, Poughkeepsie, NY, US; Bradbury, Jonathan, Poughkeepsie, NY, US; Slegel, Timothy, Poughkeepsie, NY, US; Lichtenau, Cedric, 71032 Böblingen, DE; von Buttlar, Joachim, 71034 Böblingen, DE

Prüfungsantrag gemäß § 44 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **ÜBERSICHT ÜBER DIE AUSNAHMEN FÜR UNGÜLTIGE WERTE, DIE WÄHREND DER ANWEISUNGSAUSFÜHRUNG ERKANNT WERDEN**

(57) Zusammenfassung: Eine Übersicht über die Ausnahmen für ungültige Werte wird bereitgestellt, die während der Anweisungsausführung erkannt werden. Eine Meldung, dass ein als ungültig ermittelter Wert in den Eingabedaten für eine Berechnung einer oder mehrerer Berechnungen oder in den Ausgabedaten enthalten ist, die sich aus der einen oder mehreren Berechnungen ergeben haben, wird abgerufen. Der Wert wird aufgrund einer Ausnahme einer Mehrzahl von Ausnahmen als ungültig ermittelt. Auf der Grundlage des Abrufens der Meldung, dass der Wert als ungültig ermittelt wurde, wird ein Übersichtsanzeiger gesetzt. Der Übersichtsanzeiger stellt die Mehrzahl der Ausnahmen insgesamt dar.



Beschreibung

HINTERGRUND

[0001] Ein oder mehrere Aspekte beziehen sich allgemein auf ein Ermöglichen der Verarbeitung in einer Datenverarbeitungsumgebung und insbesondere auf eine Verbesserung dieser Verarbeitung.

[0002] Um die Verarbeitung in Datenverarbeitungsumgebungen zu verbessern, die daten- und/oder rechenintensiv sind, werden Co-Prozessoren verwendet, z.B. Beschleuniger für künstliche Intelligenz (auch als Prozessoren für neuronale Netzwerke oder Beschleuniger für neuronale Netzwerke bezeichnet). Diese Beschleuniger stellen eine hohe Rechenleistung bereit, die zum Beispiel zum Durchführen entsprechender Berechnungen wie Berechnungen bei Matrizen oder Tensoren verwendet werden.

[0003] Tensor-Berechnungen werden beispielsweise bei komplexen Verarbeitungen verwendet, darunter Deep Learning (tiefes Lernen), bei dem es sich um einen Teilbereich des maschinellen Lernens handelt. Deep Learning oder maschinelles Lernen, ein Aspekt der künstlichen Intelligenz, wird bei verschiedenen Techniken verwendet, darunter im Ingenieurwesen, in der Fertigung, der Medizintechnik, der Automobiltechnik, der Computerverarbeitung usw., ohne auf diese beschränkt zu sein.

[0004] Das Durchführen von Tensor-Berechnungen nimmt viel Zeit und Rechenleistung in Anspruch. Aus diesem Grund werden Verbesserungen in Bezug auf diese Leistung angestrebt, um die Ausführung und Systemleistung zu verbessern.

KURZDARSTELLUNG

[0005] Die Mängel des Standes der Technik werden beseitigt, und durch die Bereitstellung eines Computerprogrammprodukts zum Ermöglichen der Verarbeitung in einer Datenverarbeitungsumgebung ergeben sich zusätzliche Vorteile. Das Computerprogrammprodukt umfasst ein oder mehrere durch einen Computer lesbare Speichermedien und Programmanweisungen, die auf dem einen oder mehreren durch einen Computer lesbaren Speichermedien gespeichert sind, um ein Verfahren durchzuführen. Das Verfahren umfasst Abrufen einer Meldung, dass ein als ungültig ermittelter Wert in den Eingabedaten für eine Berechnung einer oder mehrerer Berechnungen oder in den Ausgabedaten enthalten ist, die sich aus der einen oder mehreren Berechnungen ergeben haben. Der Wert wird aufgrund einer Ausnahme einer Mehrzahl von Ausnahmen als ungültig ermittelt. Auf der Grundlage des Abrufens der Meldung, dass der Wert als ungültig ermittelt wurde, wird ein Übersichtsanzeiger gesetzt. Der Übersichtsanzeiger stellt die Mehrzahl der Ausnahmen insgesamt dar.

[0006] Ein Bereitstellen einer Meldung, dass ein ungültiger Wert in den Eingabe- oder Ausgabedaten erkannt wurde, ermöglicht die Fehlerbehebung zum Beispiel bei Modellen der künstlichen Intelligenz. Eine höhere Leistung von arithmetischen Operationen wird bereitgestellt, indem nicht spezifiziert wird, welches bestimmte Datenelement den ungültigen Wert aufweist, oder nicht unterschieden wird, welche der Mehrzahl von Ausnahmen erkannt wurde.

[0007] In einem Beispiel stellt der Übersichtsanzeiger die Mehrzahl von Ausnahmen dar, ohne dass zwischen der Mehrzahl von Ausnahmen unterschieden wird. In einem Beispiel wird der Übersichtsanzeiger unabhängig davon gesetzt, welche Ausnahme der Mehrzahl von Ausnahmen verwendet wird, um zu ermitteln, ob der Wert ungültig ist.

[0008] Die Verwendung eines Anzeigers zum Anzeigen eines ungültigen Wertes unabhängig von der Art des ungültigen Wertes (z.B. nichtnumerisch, nichtdarstellbar, außerhalb des gültigen Bereichs) ermöglicht Codieren und Verarbeiten, verringert die Komplexität und erhöht die Systemleistung.

[0009] In einem Beispiel handelt es sich bei dem Übersichtsanzeiger um einen Bereichsverletzungsanzeiger eines Ausnahme-Merkers, der an einem Ort bereitgestellt wird, der durch eine Anweisung spezifiziert wird, die zum Durchführen der einen oder mehrerer Berechnungen ausgegeben wird.

[0010] Zu der Mehrzahl von Ausnahmen gehören beispielsweise ein nichtnumerischer Wert, ein nichtdarstellbarer numerischer Wert und ein Wert außerhalb des gültigen Bereichs.

[0011] In einem Beispiel beruht das Abrufen der Meldung, dass der Wert als ungültig ermittelt wurde, auf Ausführen einer Anweisung, die die eine oder mehrere Berechnungen durchführt. Dies ermöglicht die Verarbeitung der Anweisung und das Melden eines ungültigen Wertes.

[0012] Die Anweisung ist beispielsweise so konfiguriert, dass sie eine Mehrzahl von Funktionen durchführt, die Berechnungen durchführen, und die Mehrzahl von Funktionen verwendet den Übersichtsanzeiger. Durch Vorliegen eines Anzeigers, der von der Mehrzahl von Funktionen genutzt wird, werden Komplexität, Codierungs- und Überprüfungsaufwand verringert.

[0013] In einem Beispiel handelt es sich bei der Anweisung um eine Anweisung in einem neuronalen Netzwerk, das Berechnungen bei Eingabe-Tensoren durchführt, um Ausgabe-Tensoren bereitzustellen, die für die Verarbeitung durch künstliche Intelligenz verwendet werden. Die Anweisung ist beispielsweise so konfiguriert, dass sie eine Mehrzahl von Funktionen durchführt, die Berechnungen durchführen, und die Mehrzahl von Funktionen verwendet den Übersichtsanzeiger.

[0014] Der Übersichtsanzeiger wird beispielsweise für eine bestimmte Anweisung definiert, und eine andere Anweisung verwendet einen anderen Übersichtsanzeiger.

[0015] In einem Beispiel wird ein Wert eines Bedingungscodes ermittelt, der auf der Grundlage des Ausführens der bestimmten Anweisung gesetzt wurde, und die Gültigkeit des Übersichtsanzeigers beruht auf dem Feststellen, dass der Wert des Bedingungscodes ein ausgewählter Wert ist.

[0016] Durch einen Computer implementierte Verfahren und Systeme in Bezug auf einen oder mehrere Aspekte werden hier ebenfalls beschrieben und beansprucht. Ferner werden auch Dienste in Bezug auf einen oder mehrere Aspekte hier beschrieben und beansprucht.

[0017] Zusätzliche Merkmale und Vorteile werden durch die hier beschriebenen Techniken umgesetzt. Weitere Ausführungsformen und Aspekte werden hier im Einzelnen beschrieben und als Teil der beanspruchten Aspekte angesehen.

KURZBESCHREIBUNG DER ZEICHNUNGEN

[0018] Ein oder mehrere Aspekte werden besonders hervorgehoben und in den Ansprüchen am Ende der Beschreibung eindeutig als Beispiele beansprucht. Das Vorstehende sowie Aufgaben, Merkmale und Vorteile eines oder mehrerer Aspekte ergeben sich aus der nachfolgenden ausführlichen Beschreibung in Verbindung mit den beigefügten Zeichnungen, in denen:

Fig. 1A ein Beispiel einer Datenverarbeitungsumgebung zeigt, um einen oder mehrere Aspekte der vorliegenden Erfindung zu integrieren und zu verwenden;

Fig. 1B weitere Einzelheiten eines Prozessors von **Fig.** 1A gemäß einem oder mehreren Aspekten der vorliegenden Erfindung zeigt;

Fig. 2A ein Beispiel einer Verarbeitung im Zusammenhang mit dem Ausführen einer Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk gemäß einem oder mehreren Aspekten der vorliegenden Erfindung zeigt;

Fig. 2B ein Beispiel einer Ausnahmeverarbeitung zeigt, die der Ausführung einer Anweisung, z.B. einer Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk, gemäß einem oder mehreren Aspekten der vorliegenden Erfindung zugehörig ist;

Fig. 3A ein Beispiel eines Formats einer Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk gemäß einem oder mehreren Aspekten der vorliegenden Erfindung zeigt;

Fig. 3B ein Beispiel eines von der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk verwendeten allgemeinen Registers gemäß einem oder mehreren Aspekten der vorliegenden Erfindung zeigt;

Fig. 3C Beispiele von Funktionscodes zeigt, die gemäß einem oder mehreren Aspekten der vorliegenden Erfindung von der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk unterstützt werden;

Fig. 3D ein Beispiel eines anderen, von der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk verwendeten allgemeinen Registers gemäß einem oder mehreren Aspekten der vorliegenden Erfindung zeigt;

Fig. 3E ein Beispiel eines Parameterblocks zeigt, der von einer Abfragefunktion der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk gemäß einem oder mehreren Aspekten der vorliegenden Erfindung verwendet wird;

Fig. 3F ein Beispiel eines Parameterblocks zeigt, der von einer oder mehreren Nichtabfragefunktionen der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk gemäß einem oder mehreren Aspekten der vorliegenden Erfindung verwendet wird;

Fig. 3G ein Beispiel eines Tensor-Deskriptors zeigt, der gemäß einem oder mehreren Aspekten der vorliegenden Erfindung von der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk verwendet wird;

Fig. 4 ein Beispiel eines Datentypformats eines Datentyps-1 einer Verarbeitung in einem neuronalen Netzwerk (Neural Network Processing, NNP) gemäß einem oder mehreren Aspekten der vorliegenden Erfindung zeigt;

die **Fig.** 5A bis 5C Beispiele eines Eingabe-Datenlayouts zeigen, das gemäß einem oder

mehreren Aspekten der vorliegenden Erfindung von der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk verwendet wird;

die **Fig.** 6A bis 6C eine beispielhafte Ausgabe zeigen, die dem Eingabe-Datenlayout der **Fig.** 5A bis 5C gemäß einem oder mehreren Aspekten der vorliegenden Erfindung entspricht;

die **Fig.** 7A und 7B ein Beispiel zum Ermöglichen der Verarbeitung in einer Datenverarbeitungsumgebung gemäß einem oder mehreren Aspekten der vorliegenden Erfindung zeigen;

Fig. 8A ein weiteres Beispiel einer Datenverarbeitungsumgebung zeigt, um einen oder mehrere Aspekte der vorliegenden Erfindung zu integrieren und zu verwenden;

Fig. 8B ein Beispiel weiterer Einzelheiten eines Speichers von **Fig.** 8A gemäß einem oder mehreren Aspekten der vorliegenden Erfindung zeigt;

Fig. 8C ein anderes Beispiel weiterer Einzelheiten eines Speichers von **Fig.** 8A gemäß einem oder mehreren Aspekten der vorliegenden Erfindung zeigt;

Fig. 9A noch ein weiteres Beispiel einer Datenverarbeitungsumgebung zeigt, um einen oder mehrere Aspekte der vorliegenden Erfindung zu integrieren und zu verwenden;

Fig. 9B weitere Einzelheiten des Speichers von **Fig.** 9A gemäß einem oder mehreren Aspekten der vorliegenden Erfindung zeigt;

Fig. 10 eine Ausführungsform einer Cloud-Computing-Umgebung gemäß einer oder mehreren Ausführungsformen der vorliegenden Erfindung zeigt; und

Fig. 11 ein Beispiel von Abstraktionsmodellschichten gemäß einem oder mehreren Aspekten der vorliegenden Erfindung zeigt.

AUSFÜHRLICHE BESCHREIBUNG

[0019] Gemäß einem oder mehreren Aspekten der vorliegenden Erfindung wird eine Funktionalität bereitgestellt, um die Verarbeitung in einer Datenverarbeitungsumgebung zu ermöglichen. Ein Übersichtsanzeiger wird beispielsweise bereitgestellt, der anzeigt, dass ein ungültiger Wert in Eingabedaten für eine Berechnung oder in Ausgabedaten, die sich aus der Berechnung ergeben, erkannt wurde. Der Wert kann auf der Grundlage einer Ausnahme von einer Mehrzahl von Ausnahmen (z.B. nichtnumerischer Wert, nichtdarstellbarer Wert, Wert außerhalb des gültigen Bereichs usw.) ungültig sein, und der eine Übersichtsanzeiger stellt die Mehrzahl von Ausnahmen dar, ohne zwischen ihnen zu unterscheiden. In einem Beispiel dient der Übersichtsanzeiger zur Verwendung für eine Auswahlanweisung, wobei andere Anweisungen andere Anzeiger verwenden würden. Da die Auswahlanweisung weiterhin so konfiguriert ist, dass sie eine Mehrzahl von Funktionen implementiert, die Berechnungen durchführen, ist der Übersichtsanzeiger für die Verwendung durch die Mehrzahl von Funktionen vorgesehen, sodass eine Mehrzahl von Funktionen denselben Übersichtsanzeiger verwendet.

[0020] In einem Beispiel handelt es sich bei der Auswahlanweisung um eine Anweisung in einem neuronalen Netzwerk, z.B. eine Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk, die so konfiguriert ist, dass sie mehrere Funktionen implementiert, darunter beispielsweise eine Abfragefunktion und eine Mehrzahl von Nichtabfragefunktionen. Zu den Nichtabfragefunktionen gehören zum Beispiel Funktionen im Zusammenhang mit Tensor-Berechnungen. Bei der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk handelt es sich zum Beispiel um eine einzelne Anweisung (z.B. eine einzelne entworfene Hardware-Maschinenanweisung an der Hardware-/Softwareschnittstelle), die Teil einer Befehlssatzarchitektur (instruction set architecture, ISA) ist, die auf einem universellen Prozessor verarbeitet wird (z.B. zumindest teilweise decodiert und/oder ausgeführt). Die Anweisung wird zum Beispiel von einem Programm auf dem universellen Prozessor zugeteilt, der die Anweisung decodiert und initiiert. Die durch die Anweisung spezifizierten Funktionen werden von dem universellen Prozessor und/oder einem speziellen Prozessor durchgeführt, z.B. einem Co-Prozessor, der für bestimmte Funktionen konfiguriert ist, der mit dem universellen Prozessor verbunden oder Teil davon ist. Die Anweisung wird anschließend auf dem universellen Prozessor beendet.

[0021] Gemäß einem oder mehreren Aspekten der vorliegenden Erfindung werden während der Ausführung der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk (oder einer anderen Anweisung) ungültige Werte (z.B. nichtnumerische, nichtdarstellbare Werte und/oder beispielsweise Werte außerhalb des gültigen Bereichs) erkannt und in einer Übersicht angezeigt, sodass die Leistung der Anweisung und/oder des Systems nicht beeinträchtigt werden. Es wird zum Beispiel gemeldet, dass ein ungültiger Wert, z.B. ein nichtnumerischer, nichtdarstellbarer Wert oder ein Wert außerhalb des gültigen Bereichs, entweder in eine Berechnung eingegeben wurde oder durch eine Berechnung erzeugt wurde. Auf diese Weise wird die Fehlerbehebung z.B. bei Modellen der künstlichen Intelligenz ermöglicht. Der ungültige Wert wird in einem Beispiel nicht genau identifiziert; in einem Beispiel kann jedoch eine Meldung bereitgestellt werden oder implizieren, durch welche Berechnung der ungültige Wert erzeugt werden konnte.

[0022] Eine Ausführungsform einer Datenverarbeitungsumgebung zum Integrieren und Verwenden eines oder mehrerer Aspekte der vorliegenden Erfindung wird mit Bezug auf **Fig.** 1A beschrieben. Die Datenverarbeitungsumgebung beruht zum Beispiel auf der Befehlssatzarchitektur z/Architecture®, die von der International Business Machines Corporation, Armonk, New York, angeboten wird. Eine Ausführungsform der Befehlssatzarchitektur z/Architecture wird in einer Veröffentlichung mit dem Titel „z/Architecture Principles of Operation“, IBM Publication Nr. SA22-7832-12, 13. Auflage, September 2019 beschrieben, die hiermit in ihrer Gesamtheit durch Bezugnahme hierin aufgenommen ist. Bei der Befehlssatzarchitektur z/Architecture handelt es sich jedoch nur um eine beispielhafte Architektur; andere Architekturen und/oder andere Arten von Datenverarbeitungsumgebungen der International Business Machines Corporation und/oder anderer Unternehmen können einen oder mehrere Aspekte der vorliegenden Erfindung enthalten und/oder verwenden. z/Architecture und IBM sind Marken oder eingetragene Marken der International Business Machines Corporation in mindestens einem Rechtsraum.

[0023] Mit Bezug auf **Fig.** 1A umfasst eine Datenverarbeitungsumgebung 100 zum Beispiel ein Computersystem 102, das z.B. in Form einer universellen Datenverarbeitungseinheit dargestellt ist. Das Computersystem 102 kann einen oder mehrere universelle Prozessoren oder Verarbeitungseinheiten 104 (z.B. Zentraleinheiten (CPUs)), mindestens einen speziellen Prozessor wie einen Prozessor 105 eines neuronalen Netzwerks, einen Speicher 106 (zum Beispiel auch als Systemspeicher, Hauptarbeitsspeicher, Hauptspeicher, Zentralspeicher oder Speicher bekannt) und eine oder mehrere Eingabe-/Ausgabeschnittstellen (E/A-Schnittstellen) 108, die über einen oder mehrere Busse und/oder andere Verbindungen miteinander verbunden sind, umfassen, ohne auf diese beschränkt zu sein. Die Prozessoren 104, 105 und der Speicher 106 sind zum Beispiel über einen oder mehrere Busse 110 mit den E/A-Schnittstellen 108 verbunden, und die Prozessoren 104, 105 sind über einen oder mehrere Busse 111 miteinander verbunden.

[0024] Bei dem Bus 111 handelt es sich zum Beispiel um einen Speicher- oder Cache-Kohärenzbus, und der Bus 110 stellt z.B. eine oder mehrere von beliebigen mehreren Arten von Busstrukturen dar, darunter einen Speicherbus oder eine Speichersteuereinheit, einen Peripheriebus, eine AGP-Schnittstelle (Accelerated Graphics Port) und einen Prozessor oder lokalen Bus, der eine beliebige aus einer Vielfalt von Busarchitekturen nutzt. Beispielsweise und nicht einschränkend enthalten solche Architekturen den Industry-Standard-Architecture(ISA)-Bus, den Micro-Channel-Architecture(MCA)-Bus, den Enhanced-ISA(EISA)-Bus, den lokalen Video-Electronics-Standards-Association(VESA)-Bus und den Peripheral-Component-Interconnects(PCI)-Bus.

[0025] Ein oder mehrere spezielle Prozessoren (z.B. Prozessoren eines neuronalen Netzwerks) können zum Beispiel von einem oder mehreren universellen Prozessoren getrennt, jedoch mit diesen verbunden sein und/oder in einen oder mehrere universelle Prozessoren integriert sein. Zahlreiche Varianten sind möglich.

[0026] Der Speicher 106 kann zum Beispiel einen Cache 112, z.B. einen gemeinsam genutzten Cache, enthalten, der mit den lokalen Caches 114 der Prozessoren 104 und/oder mit dem Prozessor 105 eines neuronalen Netzwerks z.B. über einen oder mehrere Busse 111 verbunden sein kann. Der Speicher 106 kann weiterhin ein oder mehrere Programme oder Anwendungen 116 und mindestens ein Betriebssystem 118 enthalten. Ein beispielhaftes Betriebssystem ist ein Betriebssystem z/OS®, das von der International Business Machines Corporation, Armonk, New York, angeboten wird. z/OS ist eine Marke oder eingetragene Marke der International Business Machines Corporation in mindestens einem Rechtsraum. Es können auch andere von der International Business Machines Corporation und/oder anderen Unternehmen angebotene Betriebssysteme verwendet werden. Der Speicher 106 kann auch eine oder mehrere durch einen Computer lesbare Programmanweisungen 120 enthalten, die so konfiguriert sein können, dass sie Funktionen von Ausführungsformen von Aspekten der Erfindung ausführen.

[0027] In einer oder mehreren Ausführungsformen enthält der Speicher 106 darüber hinaus die Prozessor-Firmware 122. Die Prozessor-Firmware enthält z.B. den Mikrocode oder Millicode eines Prozessors. Sie enthält zum Beispiel die Anweisungen auf Hardware-Ebene und/oder Datenstrukturen, die für die Implementierung von Maschinencode höherer Ebene verwendet werden. In einer Ausführungsform enthält sie zum Beispiel proprietären Code, in der Regel in Form von Mikrocode oder Millicode, der vertrauenswürdige Software oder für die zugrunde liegende Hardware spezifischen Mikrocode oder Millicode enthält und der den Zugriff des Betriebssystems auf die System-Hardware steuert.

[0028] Das Computersystem 102 kann z.B. über die E/A-Schnittstellen 108 mit einer oder mehreren externen Einheiten 130, z.B. einem Benutzerterminal, einem Bandlaufwerk, einer Zeigeeinheit, einer Anzeige und einer oder mehreren Datenspeichereinheiten 134 usw., Daten austauschen. Eine Datenspeichereinheit 134 kann ein oder mehrere Programme 136, eine oder mehrere durch einen Computer lesbare Programmanweisungen 138 und/oder Daten usw. speichern. Die durch einen Computer lesbaren Programmanweisungen können so konfiguriert sein, dass sie Funktionen von Ausführungsformen von Aspekten der Erfindung ausführen.

[0029] Das Computersystem 102 kann auch z.B. über die E/A-Schnittstellen 108 mit der Netzwerk-Schnittstelle 132 Daten austauschen, die es dem Computersystem 102 ermöglichen, mit einem oder mehreren Netzwerken wie einem lokalen Netzwerk (LAN), einem allgemeinen Weitverkehrsnetzwerk (WAN) und/oder einem öffentlichen Netzwerk (z.B. dem Internet) Daten auszutauschen, die die Datenübertragung mit anderen Datenverarbeitungssystemen oder Systemen bereitstellen.

[0030] Das Computersystem 102 kann wechselbare/nichtwechselbare, flüchtige/nichtflüchtige Computersystem-Speichermedien enthalten und/oder mit diesen verbunden sein. Es kann nichtwechselbare, nichtflüchtige magnetische Medien (in der Regel als „Festplatte“ bezeichnet), ein Laufwerk für magnetische Speicherplatten zum Auslesen und Beschreiben einer wechselbaren, nichtflüchtigen magnetischen Speicherplatte (z.B. „Diskette“) und/oder ein Laufwerk für optische Speicherplatten zum Auslesen oder Beschreiben einer wechselbaren, nichtflüchtigen optischen Speicherplatte wie einer CD-ROM, DVD-ROM und andere optische Medien enthalten und/oder mit diesen verbunden sein. Es versteht sich, dass andere Hardware- und/oder Softwarekomponenten in Verbindung mit dem Computersystem 102 verwendet werden können. Zu Beispielen gehören, ohne auf diese beschränkt zu sein: Mikrocode oder Millicode, Einheitentreiber, redundante Verarbeitungseinheiten, Anordnungen externer Festplattenlaufwerke, RAID-Systeme, Bandlaufwerke und Speichersysteme für die Datenarchivierung usw.

[0031] Das Computersystem 102 kann mit zahlreichen anderen universellen oder speziellen Datenverarbeitungssystem-Umgebungen oder -Konfigurationen funktionsfähig sein. Beispiele für bekannte Datenverarbeitungssysteme, Umgebungen und/oder Konfigurationen, die für die Nutzung mit dem Computersystem 102 geeignet sein können, sind unter anderem, ohne auf diese beschränkt zu sein, Personal-Computer(PC)-Systeme, Server-Computer-Systeme, schlanke Clients, leistungsintensive Clients, Hand- oder Laptop-Einheiten, Mehrprozessorsysteme, Systeme auf der Grundlage von Mikroprozessoren, Beistellgeräte, programmierbare Unterhaltungselektronik, Netzwerk-PCs, Minicomputersysteme, Großrechnersysteme und verteilte Cloud-Computing-Umgebungen, die jedes beliebige der oben genannten Systeme oder Einheiten und Ähnliches enthalten.

[0032] In einem Beispiel enthält ein Prozessor (z.B. Prozessor 104 und/oder Prozessor 105) eine Mehrzahl von funktionalen Komponenten (oder einen Teilsatz davon), die zum Ausführen von Anweisungen verwendet werden. Wie in **Fig.** 1B gezeigt, enthalten diese funktionalen Komponenten zum Beispiel eine Anweisungsabrufkomponente 150, um auszuführende Anweisungen abzurufen; eine Anweisungsdecodiereinheit 152, um die abgerufenen Anweisungen zu decodieren und Operanden der decodierten Anweisungen abzurufen; eine oder mehrere Anweisungsausführungskomponenten 154, um die decodierten Anweisungen auszuführen; eine Speicherzugriffskomponente 156, um zum Ausführen von Anweisungen gegebenenfalls auf den Speicher zuzugreifen; und eine Rückschreibkomponente 158, um die Ergebnisse der ausgeführten Anweisungen bereitzustellen. Eine oder mehrere der Komponenten können bei der Anweisungsverarbeitung auf ein oder mehrere Register 160 zugreifen und/oder diese verwenden. Eine oder mehrere der Komponenten können darüber hinaus gemäß einem oder mehreren Aspekten der vorliegenden Erfindung mindestens einen Teil einer oder mehrerer anderer Komponenten umfassen oder Zugriff darauf haben, die beim Durchführen der Ausnahmeprüfung/-meldung und/oder beim Verarbeiten zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk von z.B. einer Anweisung zum Unterstützen einer Verarbeitung in einem neuronalen Netzwerk (oder einer anderen Verarbeitung, die einen oder mehrere Aspekte der vorliegenden Erfindung verwenden kann) wie hier beschrieben verwendet werden. Die eine oder mehrere andere Komponenten können zum Beispiel eine Ausnahmeprüfungs-/-meldungskomponente 170 und/oder eine Komponente 172 zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk (und/oder eine oder mehrere andere Komponenten) umfassen.

[0033] Gemäß einem oder mehreren Aspekten der vorliegenden Erfindung werden die Ausnahmeprüfung und -meldung während des Ausführens einer Anweisung, z.B. einer Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk, durchgeführt, ohne auf diese beschränkt zu sein. In einem Beispiel wird eine Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk auf einem universellen Prozessor (z.B. Prozessor 104) initiiert, und eine durch die Anweisung spezifizierte Funktion wird je nach Funktion entweder auf dem universellen Prozessor und/oder einem speziellen Prozessor (z.B. Prozessor 105 des neuronalen Netzwerks) ausgeführt. Die Anweisung wird anschließend auf dem universellen Prozessor beendet. In anderen Beispielen wird die Anweisung auf einem oder mehreren universellen Prozessoren oder einem oder mehreren speziellen Prozessoren initiiert, ausgeführt und beendet. Andere Varianten sind möglich.

[0034] Weitere Einzelheiten im Zusammenhang mit dem Ausführen einer Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk werden mit Bezug auf **Fig.** 2A beschrieben. Mit Bezug auf **Fig.** 2A wird in einem Beispiel eine Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk von einem Prozessor, z.B. einem universellen Prozessor (z.B. Prozessor 104) abgerufen und decodiert (200). Die decodierte Anweisung wird anschließend z.B. auf dem universellen Prozessor ausgegeben (210). Es wird ermittelt, welche Funktion durchgeführt wird (220). In einem Beispiel wird dies dadurch ermittelt, dass ein Funktionscodefeld der Anweisung geprüft wird, ein Beispiel hierfür wird nachstehend beschrieben. Die Funktion wird durchgeführt (230).

[0035] Um die Funktion durchzuführen, wird in einer Ausführungsform ermittelt, ob die Funktion auf einem speziellen Prozessor (z.B. Prozessor 105 des neuronalen Netzwerks) durchgeführt wird. In einem Beispiel wird beispielsweise eine Abfragefunktion der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk auf dem universellen Prozessor durchgeführt, und Nichtabfragefunktionen werden auf dem speziellen Prozessor durchgeführt. Andere Varianten sind jedoch ebenfalls möglich. Wenn die Funktion nicht auf dem speziellen Prozessor durchgeführt wird (es handelt sich z.B. um eine Abfragefunktion oder in einem anderen Beispiel um eine oder mehrere ausgewählte Funktionen), wird sie in einem Beispiel auf dem universellen Prozessor durchgeführt (234). Wenn die Funktion jedoch auf dem speziellen Prozessor durchgeführt wird (es handelt sich z.B. um eine Nichtabfragefunktion oder in einem anderen Beispiel um eine oder mehrere ausgewählte Funktionen), werden dem speziellen Prozessor Informationen z.B. von dem universellen Prozessor zum Ausführen der Funktion bereitgestellt, z.B. Speicheradressinformationen, die sich auf Tensor-Daten beziehen, die bei Berechnungen im neuronalen Netzwerk verwendet werden (236). Der spezielle Prozessor ruft die Informationen ab und führt die Funktion durch (238). Nachdem die Ausführung der Funktion beendet ist, kehrt die Verarbeitung zum universellen Prozessor zurück (240), der die Anweisung beendet (250). (In anderen Beispielen kann die Anweisung auf einem oder mehreren universellen Prozessoren oder einem oder mehreren speziellen Prozessoren initiiert, ausgeführt und beendet werden. Andere Varianten sind möglich.)

[0036] Gemäß einem oder mehreren Aspekten der vorliegenden Erfindung werden während der Ausführung jeder der Nichtabfragefunktionen (oder eines Teilsatzes davon) eine oder mehrere Prüfungen auf ungültige

Werte durchgeführt, z.B. auf nichtnumerische, nichtdarstellbare Werte und/oder Werte außerhalb des gültigen Bereichs, und wie mit Bezug auf **Fig.** 2B beschrieben wird auf der Grundlage des Erkennens solcher Werte eine Meldung über solch einen ungültigen Wert gemacht. Die Verarbeitung von **Fig.** 2B wird beispielsweise von einem speziellen Prozessor, z.B. Prozessor 105 des neuronalen Netzwerks, durchgeführt. In anderen Beispielen kann jedoch ein universeller Prozessor oder anderer Prozessor die Verarbeitung durchführen.

[0037] Mit Bezug auf **Fig.** 2B werden in einem Beispiel Eingabedaten (z.B. Eingabe-Tensor-Daten) für die von der Anweisung durchgeführte Funktion gelesen (260). Die Eingabedaten werden auf einen oder mehrere ungültige Werte geprüft, z.B. auf einen oder mehrere nichtnumerische Werte, einen oder mehrere nichtdarstellbare Werte, einen oder mehrere Werte außerhalb des gültigen Bereichs und/oder einen oder mehrere Ausnahmewerte (262). Es können zusätzliche, weniger und/oder andere Ausnahmen geprüft werden. Zu beispielhaften nichtdarstellbaren Werten gehören zum Beispiel Unendlich, Teilen durch null, eine Nichtzahl, Unendlich und Nichtzahl kombiniert (NINF) usw. Wenn ein ungültiger Wert erkannt wird (264), wird eine Meldung über den ungültigen Wert bereitgestellt (266).

[0038] In einem Beispiel wird die Meldung über den ungültigen Wert dem universellen Prozessor wieder bereitgestellt, der die Anweisung initiiert hat, und der universelle Prozessor setzt einen Übersichtsanzeiger auf z.B. eins, um den ungültigen Wert anzuzeigen. Im Übersichtsanzeiger wird nicht unterschieden, welche Art von Ausnahme dazu geführt hat, dass der ungültige Wert erkannt wurde. Das Setzen des Übersichtsanzeigers ist unabhängig von der Art der Ausnahme (z.B. nichtnumerischer Wert, einer der nichtdarstellbaren Werte, Wert außerhalb des gültigen Bereichs usw.). In einer Ausführungsform ist die Berechnung, die zum Erkennen des ungültigen Wertes führt, jedoch bekannt oder leicht zu ermitteln, da der Übersichtsanzeiger gesetzt ist und die Verarbeitung auf der Grundlage des Erkennens des ungültigen Wertes zum universellen Prozessor zurückkehrt.

[0039] In einem besonderen Beispiel handelt es sich bei dem Übersichtsanzeiger um einen Bereichsanzeiger (z.B. Bit) in einem Ausnahme-Merker, der an einem ausgewählten Ort bereitgestellt wird, der für die Anweisung zugänglich ist, die die Funktion initiiert, die die Berechnung(en) durchführt, bei denen der ungültige Wert erkannt wurde. Bei dem ausgewählten Ort handelt es sich beispielsweise um ein allgemeines Register, z.B. das allgemeine Register 0, oder ein von der Anweisung verwendeter Speicherort. Verschiedene Orte sind möglich. Ein Beispiel für den Ausnahme-Merker und insbesondere den Bereichsanzeiger wird nachstehend bereitgestellt.

[0040] Zurück zur Abfrage 264: Wenn kein ungültiger Wert in den Eingabedaten gefunden wird, werden eine oder mehrere Berechnungen durchgeführt, die auf der von der Anweisung spezifizierten Funktion beruhen, und Ausgabedaten werden erzeugt. In einem Beispiel wird geprüft, ob ein oder mehrere ungültige Werte in den Ausgabedaten enthalten sind (270). Wenn ein ungültiger Wert erkannt wird (272), wird eine Meldung über den ungültigen Wert bereitgestellt (266). In einem Beispiel wird diese Meldung dem universellen Prozessor bereitgestellt, der den Übersichtsanzeiger wie hier beschrieben setzt. In einem anderen Beispiel kann es sich jedoch um einen anderen Anzeiger handeln, der gleich oder anders als der Anzeiger sein kann, der für ungültige Eingabewerte verwendet wird.

[0041] Zurück zur ABFRAGE 272: Wenn kein ungültiger Wert gefunden wird, wird diese Ausnahmeverarbeitung beendet.

[0042] Wie hier beschrieben, wird beim Durchführen der Berechnungsverarbeitung erkannt, ob ein Wert in den Eingabe- oder Ausgabedaten ein ungültiger Wert ist, z.B. ein nichtnumerischer, nichtdarstellbarer numerischer Wert und/oder ein Wert außerhalb des gültigen Bereichs. Aufgrund des Erkennens des ungültigen Wertes wird ein Übersichtsanzeiger (z.B. ein Bereichsausnahme-Merker) gesetzt. In einem Beispiel wird nicht angegeben, welches Datenelement den ungültigen Wert bereitgestellt hat. Dies ermöglicht eine höhere Leistung bei arithmetischen Operationen.

[0043] Ein Beispiel einer Anweisung zum Verwenden eines Übersichtsanzeigers ist eine Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk. In einem Beispiel ist der Übersichtsanzeiger ausschließlich zur Verwendung durch die Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk und ihren Funktionen/Operationen vorgesehen. Andere Anweisungen können andere Anzeiger verwenden. Weitere Einzelheiten zu einer Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk und Funktionen, die von der Anweisung unterstützt werden, werden hier mit Bezug auf die **Fig.** 3A bis 3G beschrieben. In der vorliegenden Beschreibung der Anweisung und/oder Funktionen der Anweisung werden spezifische Speicherorte, spezifische Felder und/oder spezifische Größen der Felder

angegeben (z.B. spezifische Bytes und/oder Bits). Es können jedoch auch andere Speicherorte, Felder und/oder Größen bereitgestellt werden. Weiterhin kann das Setzen eines Bits auf einen bestimmten Wert, z.B. eins oder null, zwar angegeben werden, dies ist jedoch nur ein Beispiel. In anderen Beispielen kann für das Bit, sofern gesetzt, ein anderer Wert, z.B. der entgegengesetzte Wert oder ein anderer Wert, gesetzt werden. Zahlreiche Varianten sind möglich.

[0044] Mit Bezug auf **Fig.** 3A weist die Anweisung 300 zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk in einem Beispiel ein RRE-Format auf, das ein Register und eine Registeroperation mit einem erweiterten Operationscode (Opcode) bezeichnet. Wie in **Fig.** 3A gezeigt, enthält die Anweisung 300 zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk in einem Beispiel ein Operationscodefeld 302 (Opcode-Feld) (z.B. Bits 0 bis 15), das eine Operation zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk anzeigt. In einem Beispiel sind die Bits 16 bis 31 der Anweisung reserviert und müssen Nullen enthalten.

[0045] In einem Beispiel verwendet die Anweisung eine Mehrzahl von allgemeinen Registern, die implizit durch die Anweisung spezifiziert werden. Die Anweisung 300 zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk verwendet zum Beispiel die impliziten Register allgemeines Register 0 und allgemeines Register 1, von denen Beispiele mit Bezug auf die **Fig.** 3B bzw. 3D beschrieben werden.

[0046] Mit Bezug auf **Fig.** 3B enthält das allgemeine Register 0 in einem Beispiel ein Funktionscodefeld und Statusfelder, die nach Beenden der Anweisung aktualisiert werden können. Das allgemeine Register 0 enthält beispielsweise ein Antwortcodefeld 310 (z.B. Bits 0 bis 15), ein Ausnahme-Merkerfeld 312 (z.B. Bits 24 bis 31) und ein Funktionscodefeld 314 (z.B. Bits 56 bis 63). In einem Beispiel sind die Bits 16 bis 23 und 32 bis 55 des allgemeinen Registers 0 weiterhin reserviert und müssen Nullen enthalten. Ein oder mehrere Felder werden von einer bestimmten Funktion verwendet, die von der Anweisung durchgeführt wird. In einem Beispiel werden nicht alle Felder von allen Funktionen verwendet. Jedes der Felder wird nachstehend beschrieben:

[0047] Antwortcode (RC) 310: Dieses Feld (z.B. Bitpositionen 0 bis 15) enthält den Antwortcode. Wenn die Ausführung der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk mit einem Bedingungscode von z.B. eins beendet wird, wird ein Antwortcode gespeichert. Wenn eine ungültige Eingabebedingung auftritt, wird ein Nichtnullwert im Antwortcodefeld gespeichert, der den Grund der während der Ausführung erkannten ungültigen Eingabebedingung angibt, und ein ausgewählter Bedingungscode, z.B. 1, wird gesetzt. In einem Beispiel werden die im Antwortcodefeld gespeicherten Codes wie folgt definiert:

Antwortcode	Bedeutung
0001	Das Format des Parameterblocks wie durch die Versionsnummer des Parameterblocks angegeben wird vom Modell nicht unterstützt.
0002	Die spezifizierte Funktion ist in der Maschine nicht definiert oder installiert.
0010	Ein spezifiziertes Tensor-Datenlayout-Format wird nicht unterstützt.
0011	Ein spezifizierter Tensor-Datentyp wird nicht unterstützt.
0012	Eine spezifizierte einzelne Tensor-Dimension ist größer als die maximale Dimensionsindexgröße.
0013	Die Größe eines spezifizierten Tensors ist größer als die maximale Tensor-Größe.
0014	Die spezifizierte Tensor-Adresse ist nicht an einer 4-KB-Grenze ausgerichtet.
0015	Die funktionsspezifische Sicherungsbereichsadresse ist nicht an einer 4-KB-Grenze ausgerichtet.
F000-FFFF	Funktionsspezifische Antwortcodes. Diese Antwortcodes werden für bestimmte Funktionen definiert.

[0048] Ausnahme-Merker (EF) 312: Dieses Feld (z.B. Bitpositionen 24 bis 31) enthält die Ausnahme-Merker. Wenn während der Ausführung der Anweisung eine Ausnahmebedingung erkannt wird, wird die entsprechende Ausnahme-Merkersteuerung (z.B. Bit) z.B. auf eins gesetzt; andernfalls bleibt die Steuerung unverändert. Das Ausnahme-Merkerfeld muss vor dem ersten Aufrufen der Anweisung auf null initialisiert werden.

Reservierte Merker bleiben bei der Ausführung der Anweisung unverändert. In einem Beispiel sind die im Ausnahme-Merkerfeld gespeicherten Merker wie folgt definiert:

EF (Bit)	Bedeutung
0	Bereichsüberschreitung. Dieser Merker wird gesetzt, wenn ein ungültiger Wert entweder in einem Eingabe-Tensor erkannt oder im Ausgabe-Tensor gespeichert wurde. Dieser Merker ist z.B. nur gültig, wenn die Anweisung mit einem Bedingungscode, z.B. 0, beendet wird. In einem Beispiel handelt es sich bei diesem Merker um den hier beschriebenen Übersichtsanzeiger.
1 bis 7	Reserviert.

[0049] Funktionscode (FC) 314: Dieses Feld (z.B. Bitpositionen 56 bis 63) enthält den Funktionscode. Beispiele für zugewiesene Funktionscodes für die Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk sind in **Fig.** 3C dargestellt. Alle anderen Funktionen sind nicht zugewiesen. Wenn ein nichtzugewiesener oder nichtinstallierter Funktionscode spezifiziert ist, werden ein Antwortcode von z.B. 0002 hex und ein ausgewählter Bedingungscode, z.B. 1, gesetzt. Dieses Feld wird während der Ausführung nicht geändert.

[0050] Wie angegeben, verwendet die Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk neben dem allgemeinen Register 0 auch das allgemeine Register 1, ein Beispiel dafür ist in **Fig.** 3D dargestellt. Die Bits 40 bis 63 im 24-Bit-Adressierungsmodus, die Bits 33 bis 63 im 31-Bit-Adressierungsmodus oder die Bits 0 bis 63 im 64-Bit-Adressierungsmodus enthalten beispielsweise eine Adresse eines Parameterblocks 320. Der Inhalt des allgemeinen Registers 1 gibt zum Beispiel eine logische Adresse eines ganz links befindlichen Bytes des Parameterblocks im Speicher an. Der Parameterblock muss auf einer Doppelwortgrenze angegeben werden; andernfalls wird eine Spezifikationsausnahme erkannt. Der Inhalt des allgemeinen Registers 1 wird nicht bei allen Funktionen geändert.

[0051] Im Zugriffsregistermodus spezifiziert das Zugriffsregister 1 einen Adressraum, der beispielsweise den Parameterblock, Eingabe-Tensoren, Ausgabe-Tensoren und den funktionsspezifischen Sicherungsbereich enthält.

[0052] In einem Beispiel kann der Parameterblock je nach der Funktion, die die auszuführende Anweisung spezifiziert, unterschiedliche Formate haben. Die Abfragefunktion hat zum Beispiel einen Parameterblock eines Formats, und andere Funktionen der Anweisung haben einen Parameterblock eines anderen Formats. In einem anderen Beispiel verwenden alle Funktionen das gleiche Parameterblockformat. Andere Varianten sind ebenfalls möglich.

[0053] Ein Parameterblock und/oder die Informationen im Parameterblock werden beispielsweise im Speicher, in Hardware-Registern und/oder einer Kombination aus Speichern und/oder Registern gespeichert. Andere Beispiele sind ebenfalls möglich.

[0054] Ein Beispiel für einen von einer Abfragefunktion wie der Operation NNPA-Query Available Functions (Abfrage verfügbarer Funktionen = QAF) verwendeten Parameterblock wird mit Bezug auf **Fig.** 3E beschrieben. Wie in einem Beispiel gezeigt, enthält der Parameterblock 330 für die NNPA-Query Available Functions zum Beispiel:

[0055] Vektor 332 der installierten Funktionen: Dieses Feld (z.B. Bytes 0 bis 31) des Parameterblocks enthält den Vektor der installierten Funktionen. In einem Beispiel entsprechen die Bits 0 bis 255 des Vektors der installierten Funktionen den jeweiligen Funktionscodes 0 bis 255 der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk. Wenn ein Bit z.B. eins ist, ist die entsprechende Funktion installiert; andernfalls ist die Funktion nicht installiert.

[0056] Vektor 334 der installierten Parameterblockformate: Dieses Feld (z.B. Bytes 32 bis 47) des Parameterblocks enthält den Vektor der installierten Parameterblockformate. In einem Beispiel entsprechen die Bits 0 bis 127 des Vektors der installierten Parameterblockformate den Parameterblockformaten 0 bis 127 für die Nichtabfragefunktionen der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk. Wenn ein Bit z.B. eins ist, ist das entsprechende Parameterblockformat installiert; andernfalls ist das Parameterblockformat nicht installiert.

[0057] Installierte Datentypen 336: Dieses Feld (z.B. Bytes 48 und 49) des Parameterblocks enthält den Vektor der installierten Datentypen. In einem Beispiel entsprechen die Bits 0 bis 15 des Vektors der installierten Datentypen den installierten Datentypen. Wenn ein Bit z.B. eins ist, ist der entsprechende Datentyp installiert; andernfalls ist der Datentyp nicht installiert. Zu beispielhaften Datentypen gehören (zusätzliche, weniger und/oder andere Datentypen sind möglich):

Bit	Datentyp
0	NNP-Datentyp-1
1 bis 15	Reserviert

[0058] Installierte Datenlayout-Formate 338: Dieses Feld (z.B. Bytes 52 bis 55) des Parameterblocks enthält den Vektor der installierten Datenlayout-Formate. In einem Beispiel entsprechen die Bits 0 bis 31 des Vektors der installierten Datenlayout-Formate den installierten Datenlayout-Formaten. Wenn ein Bit z.B. eins ist, ist das entsprechende Datenlayout-Format installiert; andernfalls ist das Datenlayout-Format nicht installiert. Zu beispielhaften Datenlayout-Formaten gehören (zusätzliche, weniger und/oder andere Datenlayout-Formate sind möglich):

Bit	Datenlayout-Format
0	4D-Merkmalstensor
1	4D-Kernel-Tensor
2 bis 31	Reserviert

[0059] Maximale Dimensionsindexgröße 340: Dieses Feld (z.B. Bytes 60 bis 63) des Parameterblocks enthält z.B. eine binäre 32-Bit-Ganzzahl ohne Vorzeichen, die eine maximale Anzahl von Elementen in einer spezifizierten Dimensionsindexgröße für einen beliebigen spezifizierten Tensor angibt. In einem anderen Beispiel gibt die maximale Dimensionsindexgröße eine maximale Anzahl von Bytes in einer spezifizierten Dimensionsindexgröße für einen beliebigen spezifizierten Tensor an. Andere Beispiele sind ebenfalls möglich.

[0060] Maximale Tensor-Größe 342: Dieses Feld (z.B. Bytes 64 bis 71) des Parameterblocks enthält z.B. eine binäre 32-Bit-Ganzzahl ohne Vorzeichen, die eine maximale Anzahl von Bytes in einem beliebigen spezifizierten Tensor mit Füllbytes angibt, die von dem Tensor-Format benötigt werden. In einem anderen Beispiel gibt die maximale Tensor-Größe eine maximale Anzahl von Gesamtelementen in einem beliebigen spezifizierten Tensor an, einschließlich dem für das Tensor-Format erforderlichen Auffüllen. Andere Beispiele sind ebenfalls möglich.

[0061] Vektor 344 der installierten NNP-Datentyp-1-Konvertierungen: Dieses Feld (z.B. Bytes 72 und 73) des Parameterblocks enthält den Vektor der installierten NNP-Datentyp-1-Konvertierungen. In einem Beispiel entsprechen die Bits 0 bis 15 des Vektors der installierten NNP-Datentyp-1-Konvertierungen der installierten Datentyp-Konvertierung vom/zum NNP-Datentyp-1-Format. Wenn ein Bit eins ist, ist die entsprechende Konvertierung installiert; andernfalls ist die Konvertierung nicht installiert. Es können zusätzliche, weniger und/oder andere Konvertierungen spezifiziert werden.

Bit	Datentyp
0	Reserviert
1	BFP-Kleinformat
2	BFP-Kurzformat
3 bis 15	Reserviert

[0062] Ein Beispiel eines Parameterblocks für eine Abfragefunktion wird zwar mit Bezug auf **Fig.** 3E beschrieben, andere Formate eines Parameterblocks für eine Abfragefunktion einschließlich der Operation NNPA-Query Available Functions können jedoch ebenfalls verwendet werden. Das Format kann in einem Beispiel von der Art der Abfragefunktion abhängen, die durchgeführt werden soll. Der Parameterblock und/oder jedes Feld des Parameterblocks können weiterhin zusätzliche, weniger und/oder andere Informationen enthalten.

[0063] Zusätzlich zum Parameterblock für eine Abfragefunktion gibt es in einem Beispiel ein Parameterblockformat für Nichtabfragefunktionen wie die Nichtabfragefunktionen der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk. Ein Beispiel für einen von einer Nichtabfragefunktion wie einer Nichtabfragefunktion der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk verwendeten Parameterblock wird mit Bezug auf **Fig.** 3F beschrieben.

[0064] Wie in einem Beispiel gezeigt, enthält ein Parameterblock 350, der z.B. von den Nichtabfragefunktionen der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk verwendet wird, zum Beispiel:

[0065] Versionsnummer des Parameterblocks 352: Dieses Feld (z.B. Bytes 0 und 1) des Parameterblocks spezifiziert die Version und die Größe des Parameterblocks. In einem Beispiel sind die Bits 0 bis 8 der Versionsnummer des Parameterblocks reserviert und müssen Nullen enthalten, und die Bits 9 bis 15 der Versionsnummer des Parameterblocks enthalten eine binäre Ganzzahl ohne Vorzeichen, die das Format des Parameterblocks spezifiziert. Die Abfragefunktion stellt einen Mechanismus bereit, der die verfügbaren Parameterblockformate angibt. Wenn die spezifizierte Größe oder das spezifizierte Format des Parameterblocks vom Modell nicht unterstützt wird, wird ein Antwortcode von z.B. 0001 hex im allgemeinen Register 0 gespeichert, und die Anweisung wird durch Setzen eines Bedingungscodes, z.B. Bedingungscode 1, beendet. Die Versionsnummer des Parameterblocks wird vom Programm spezifiziert und wird während der Ausführung der Anweisung nicht geändert.

[0066] Versionsnummer des Modells 354: Dieses Feld (z.B. Byte 2) des Parameterblocks ist eine binäre Ganzzahl ohne Vorzeichen, die das Modell angibt, das die Anweisung ausgeführt hat (z.B. die bestimmte Nichtabfragefunktion). Wenn ein Fortsetzungsmerker (nachstehend beschrieben) eins ist, kann es sich bei der Versionsnummer des Modells um eine Eingabe für die Operation handeln, mit dem Zweck, den Inhalt eines Fortsetzungszustand-Pufferfeldes (nachstehend beschrieben) des Parameterblocks zu interpretieren, um die Operation wiederaufzunehmen.

[0067] Fortsetzungsmerker 356: Wenn dieses Feld (z.B. Bit 63) des Parameterblocks z.B. eins ist, bedeutet dies, dass die Operation teilweise beendet ist und der Inhalt des Fortsetzungszustandpuffers verwendet werden kann, um die Operation wiederaufzunehmen. Das Programm muss den Fortsetzungsmerker auf null initialisieren und darf den Fortsetzungsmerker nicht ändern, wenn die Anweisung erneut ausgeführt werden soll, um die Operation wiederaufzunehmen; andernfalls sind die Ergebnisse nicht vorhersehbar.

[0068] Wenn der Fortsetzungsmerker zu Beginn der Operation gesetzt wird und sich der Inhalt des Parameterblocks seit dem ersten Aufrufen geändert hat, sind die Ergebnisse nicht vorhersehbar.

[0069] Funktionsspezifische Sicherungsbereichsadresse 358: Dieses Feld (z.B. Bytes 56 bis 63) des Parameterblocks enthält die logische Adresse des funktionsspezifischen Sicherungsbereichs. In einem Beispiel muss die funktionsspezifische Sicherungsbereichsadresse an einer 4-KB-Grenze ausgerichtet werden; andernfalls wird ein Antwortcode von z.B. 0015 hex im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode von z.B. 1 beendet. Die Adresse hängt vom aktuellen Adressierungsmodus ab. Die Größe des funktionsspezifischen Sicherungsbereichs hängt vom Funktionscode ab.

[0070] Wenn sich der gesamte funktionsspezifische Sicherungsbereich mit der Zuordnung des Speicherbereichs für die Programmereignisaufzeichnung (program event recording, PER) überschneidet, wird gegebenenfalls ein PER-Speicheränderungsereignis für den funktionsspezifischen Sicherungsbereich erkannt. Wenn sich nur ein Teil des funktionsspezifischen Sicherungsbereichs mit der Zuordnung des PER-Speicherbereichs überschneidet, ist dies modellabhängig, wonach Folgendes eintritt:

* Ein PER-Speicheränderungsereignis wird gegebenenfalls für den gesamten funktionsspezifischen Sicherungsbereich erkannt.

* Ein PER-Speicheränderungsereignis wird gegebenenfalls für den Teil des funktionsspezifischen Sicherungsbereichs erkannt, der gespeichert wird.

[0071] Wenn sich der gesamte Parameterblock mit der Zuordnung des PER-Speicherbereichs überschneidet, wird gegebenenfalls ein PER-Speicheränderungsereignis für den Parameterblock erkannt. Wenn sich nur ein Teil des Parameterblocks mit der Zuordnung des PER-Speicherbereichs überschneidet, ist dies modellabhängig, wonach Folgendes eintritt:

* Ein PER-Speicheränderungsereignis wird gegebenenfalls für den gesamten Parameterblock erkannt.

* Ein PER-Speicheränderungsereignis wird gegebenenfalls für den Teil des Parameterblocks erkannt, der gespeichert wird.

[0072] Ein PER-Null-Adressenerkennungsereignis wird gegebenenfalls für den Parameterblock erkannt. Die Null-Adressenerkennung findet in einem Beispiel nicht auf die Tensor-Adressen oder die funktionsspezifische Sicherungsbereichsadresse Anwendung.

[0073] Ausgabe-Tensor-Deskriptoren (z.B. 1 und 2) 360/Eingabe-Tensor-Deskriptoren (z.B. 1 bis 3) 365: Ein Beispiel eines Tensor-Deskriptors wird mit Bezug auf **Fig.** 3G beschrieben. In einem Beispiel enthält ein Tensor-Deskriptor 360, 365:

[0074] Datenlayout-Format 382: Dieses Feld (z.B. Byte 0) des Tensor-Deskriptors spezifiziert das Datenlayout-Format. Zu gültigen Datenlayout-Formaten gehören zum Beispiel (zusätzliche, weniger und/oder andere Datenlayout-Formate sind möglich):

Format	Beschreibung	Ausrichtung (Bytes)
0		4D-Merkmalstensor 4096
1		4D-Kernel-Tensor 4096
2 bis 255	Reserviert	-

[0075] Wenn ein nichtunterstütztes oder reserviertes Datenlayout-Format spezifiziert ist, wird der Antwortcode von z.B. 0010 hex im allgemeinen Register 0 gespeichert, und die Anweisung wird durch Setzen eines Bedingungscodes, z.B. 1, beendet.

[0076] Datentyp 384: Dieses Feld (z.B. Byte 1) spezifiziert den Datentyp des Tensors. Beispiele von unterstützten Datentypen werden nachstehend beschrieben (zusätzliche, weniger und/oder andere Datentypen sind möglich):

Wert	Datentyp	Datengröße (Bits)
0	NNP-Datentyp-1	16
1 bis 255	Reserviert	-

[0077] Wenn ein nichtunterstützter oder reservierter Datentyp spezifiziert ist, wird ein Antwortcode von z.B. 0011 hex im allgemeinen Register 0 gespeichert, und die Anweisung wird durch Setzen eines Bedingungscodes, z.B. 1, beendet.

[0078] Indexgröße 386 der Dimensionen 1 bis 4: Die Dimensionsindexgrößen eins bis vier spezifizieren zusammen die Form eines 4D-Tensors. Jede Dimensionsindexgröße muss größer als null und kleiner oder gleich wie die maximale Dimensionsindexgröße sein (340, **Fig.** 3E); andernfalls wird ein Antwortcode von z.B. 0012 hex im allgemeinen Register 0 gespeichert, und die Anweisung wird durch Setzen eines Bedingungscodes, z.B. 1, beendet. Die Gesamtgröße des Tensors muss kleiner oder gleich wie die maximale Tensor-Größe sein (342, **Fig.** 3E); andernfalls wird ein Antwortcode von z.B. 0013 hex im allgemeinen Register 0 gespeichert, und die Anweisung wird durch Setzen eines Bedingungscodes, z.B. 1, beendet.

[0079] Um die Anzahl von Bytes in einem 4D-Merkmalstensor mit Elementen des NNPA-Datentyps-1 zu ermitteln (d.h. die Gesamtgröße des Tensors), wird in einem Beispiel Folgendes verwendet: dimension-index.4 * dimension-index-3 * ceil (dimension-index-2/32) * 32 ceil (dimension-index-1/64) * 64 *2.

[0080] Tensor-Adresse 388: Dieses Feld (z.B. Bytes 24 bis 31) des Tensor-Deskriptors enthält eine logische Adresse des ganz links befindlichen Bytes des Tensors. Die Adresse hängt vom aktuellen Adressierungsmodus ab.

[0081] Wenn die Adresse nicht auf die Grenze des zugehörigen Datenlayout-Formats ausgerichtet ist, wird ein Antwortcode von z.B. 0014 hex im allgemeinen Register 0 gespeichert, und die Anweisung wird durch Setzen eines Bedingungscodes, z.B. 1, beendet.

[0082] Im Zugriffsregistermodus spezifiziert das Zugriffsregister 1 den Adressraum, der alle aktiven Eingabe- und Ausgabe-Tensoren im Speicher enthält.

[0083] Zurück zu **Fig.** 3F: Der Parameterblock 350 enthält weiterhin in einem Beispiel die funktionsspezifischen Parameter 1 bis 5 (370), die wie hier beschrieben von spezifischen Funktionen verwendet werden können.

[0084] Der Parameterblock 350 enthält weiterhin in einem Beispiel ein Fortsetzungszustand-Pufferfeld 375, das Daten (oder einen Speicherort von Daten) enthält, die verwendet werden, wenn die Ausführung dieser Anweisung wiederaufgenommen werden soll.

[0085] Als Eingabe für die Operation müssen die reservierten Felder des Parameterblocks Nullen enthalten. Wenn die Operation endet, können reservierte Felder als Nullen gespeichert werden oder unverändert bleiben.

[0086] Ein Beispiel eines Parameterblocks für eine Nichtabfragefunktion wird zwar mit Bezug auf **Fig.** 3F beschrieben, andere Formate eines Parameterblocks für eine Nichtabfragefunktion, darunter eine Nichtabfragefunktion der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk, können jedoch ebenfalls verwendet werden. Das Format kann in einem Beispiel von der Art der Funktion abhängen, die durchgeführt werden soll. Ein Beispiel eines Tensor-Deskriptors wird zwar mit Bezug auf **Fig.** 3G beschrieben, andere Formate können jedoch ebenfalls verwendet werden. Darüber hinaus können unterschiedliche Formate für Eingabe- und Ausgabe-Tensoren verwendet werden. Andere Varianten sind möglich.

[0087] Weitere Einzelheiten zu verschiedenen Funktionen, die von einer Ausführungsform der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk unterstützt werden, werden nachstehend beschrieben:

Funktionscode 0: NNPA-QAF (Query Available Functions)

[0088] Die Abfragefunktion zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk (NNPA) stellt einen Mechanismus bereit, um ausgewählte Informationen anzugeben, zum Beispiel die Verfügbarkeit installierter Funktionen, installierte Parameterblockformate, installierte Datentypen, installierte Datenlayout-Formate, maximale Dimensionsindexgröße und maximale Tensor-Größe. Die Informationen werden abgerufen und an einem ausgewählten Ort wie einem Parameterblock (z.B. Parameterblock 330) abgelegt. Wenn die Operation endet, können reservierte Felder des Parameterblocks als Nullen gespeichert werden oder unverändert bleiben.

[0089] Beim Ausführen einer Ausführungsform der Abfragefunktion ruft ein Prozessor wie der universelle Prozessor 104 Informationen über einen spezifischen Prozessor, z.B. ein spezifisches Modell eines Prozessors eines neuronalen Netzwerks, z.B. der Prozessor 105 des neuronalen Netzwerks, ab. Ein spezifisches Modell eines Prozessors oder einer Maschine verfügt über bestimmte Funktionalitäten. Ein anderes Modell des Prozessors oder der Maschine kann zusätzliche, weniger und/oder andere Funktionalitäten haben und/oder einer anderen Generation (z.B. einer aktuellen oder künftigen Generation) angehören, die zusätzliche, weniger und/oder andere Funktionalitäten hat. Die abgerufenen Informationen werden in einem Parameterblock (z.B. Parameterblock 330) oder einer anderen Struktur abgelegt, die für eine oder mehrere Anwendungen zugänglich ist und/oder die diese Informationen bei der weiteren Verarbeitung verwenden kann. In einem Beispiel werden der Parameterblock und/oder die Informationen des Parameterblocks im Speicher gespeichert. In anderen Ausführungsformen können der Parameterblock und/oder die Informationen in einem oder mehreren Hardware-Registern gespeichert werden. In einem weiteren Beispiel kann es sich bei der Abfragefunktion um eine bevorzugte Operation handeln, die von dem Betriebssystem ausgeführt wird, das eine Anwendungsprogrammierschnittstelle zur Verfügung stellt, um diese Informationen der Anwendung oder einem nichtbevorzugten Programm zur Verfügung zu stellen. In noch einem anderen Beispiel wird die Abfragefunktion von einem speziellen Prozessor wie dem Prozessor 105 des neuronalen Netzwerks durchgeführt. Andere Varianten sind möglich.

[0090] Die Informationen werden z.B. von der Firmware des Prozessors abgerufen, der die Abfragefunktion ausführt. Die Firmware kennt die Attribute des spezifischen Modells des spezifischen Prozessors (z.B. Prozessor eines neuronalen Netzwerks). Diese Informationen können z.B. in einem Steuerblock, Register und/oder Speicher gespeichert sein und/oder dem Prozessor, der die Abfragefunktion ausführt, auf sonstige Weise zugänglich sein.

[0091] Die abgerufenen Informationen enthalten zum Beispiel modellabhängige detaillierte Informationen über mindestens ein oder mehrere Datenattribute des spezifischen Prozessors, darunter zum Beispiel ein oder mehrere installierte oder unterstützte Datentypen, ein oder mehrere installierte oder unterstützte Datenlayout-Formate und/oder ein oder mehrere installierte oder unterstützte Datengrößen des ausgewählten Modells des spezifischen Prozessors. Diese Informationen sind insofern modellabhängig, als andere Modelle (z.B. frühere Modelle und/oder künftige Modelle) möglicherweise nicht dieselben Datenattribute, z.B. dieselben Datentypen, Datengrößen und/oder Datenlayout-Formate, unterstützen. Wenn die Ausführung der Abfragefunktion (z.B. NNPA-QAF-Funktion) beendet ist, wird zum Beispiel ein Bedingungscode 0 gesetzt. Die Bedingungscodes 1, 2 und 3 sind in einem Beispiel nicht auf die Abfragefunktion anwendbar. Weitere Informationen zu den abgerufenen Informationen werden nachstehend beschrieben.

[0092] Wie bereits erwähnt, enthalten die abgerufenen Informationen in einem Beispiel modellabhängige Informationen über ein oder mehrere Datenattribute von z.B. einem bestimmten Modell eines Prozessors eines neuronalen Netzwerks. Ein Beispiel für ein Datenattribut sind die installierten Datentypen des Prozessors eines neuronalen Netzwerks. Ein bestimmtes Modell eines Prozessors eines neuronalen Netzwerks (oder eines anderen Prozessors) kann zum Beispiel einen oder mehrere Datentypen unterstützen, z.B. einen Datentyp NNP-Datentyp-1 (auch als Datentyp-1 für die Verarbeitung in einem neuronalen Netzwerk bezeichnet), und/oder andere Datentypen. Bei dem Datentyp NNP-Datentyp-1 handelt es sich um ein 16-Bit-Gleitkommaformat, das eine Reihe von Vorteilen für das Deep-Learning-Training und Inferenzberechnungen bereitstellt, darunter zum Beispiel: Beibehalten der Genauigkeit von Deep-Learning-Netzwerken; Beseitigen des Subnormalformats, das Rundungsmodi vereinfacht, und Behandeln von Grenzfällen; automatisches Runden auf den nächsten Wert für arithmetische Operationen; und spezielle Entitäten von Unendlich und Nichtzahl (NaN) werden zu einem Wert (NINF) zusammengefasst, der von arithmetischen Operationen akzeptiert und bearbeitet wird. NINF stellt bessere Standardwerte für Exponentenüberlauf und ungültige Operationen (z.B. Teilen durch null) bereit. Dadurch können viele Programme fortgesetzt werden, ohne solche Fehler zu verstecken und ohne spezielle Behandlung von Ausnahmebedingungen. Andere modellabhängige Datentypen sind ebenfalls möglich.

[0093] Ein Beispiel eines Formats des Datentyps NNP-Datentyp-1 ist in **Fig.** 4 dargestellt. Wie gezeigt, können die Daten von NNP-Datentyp-1 in einem Beispiel in einem Format 400 dargestellt werden, das zum Beispiel ein Vorzeichen 402 (z.B. Bit 0), einen Exponenten + 31 404 (z.B. Bits 1 bis 6) und eine Bruchzahl 406 (z.B. Bits 7 bis 15) enthält.

[0094] Beispielhafte Eigenschaften des NNP-Datentyp-1-Formats sind nachstehend dargestellt:

Eigenschaft	NNP-Datentyp-1
Formatlänge (Bits)	16 Bits
Länge des verzerrten Exponenten (Bits)	6 Bits
Länge der Bruchzahl (Bits)	9 Bits
Genauigkeit (p)	10 Bits
Maximaler Exponent in der linken Einheitsansicht (Emax)	32
Minimaler Exponent in der linken Einheitsansicht (Emin)	-31
Verzerrung in der linken Einheitsansicht (LUV)	31
Nmax	$(1\text{-}2^{-9}) \times 2^{33} \approx 8{,}6 \times 10^{9}$
Nmin	$(1+2^{-9}) \times 2^{-31} \approx 4{,}6 \times 10^{-10}$
Dmin	—

[0095] Dabei gibt ≈ an, dass es sich um einen Näherungswert handelt, Nmax ist die größte (nach Größenordnung) darstellbare endliche Zahl und Nmin ist die kleinste (nach Größenordnung) darstellbare Zahl.

[0096] Weitere Einzelheiten zum Datentyp NNP-Datentyp-1 werden nachstehend beschrieben:

[0097] Verzerrter Exponent: Die Verzerrung, die verwendet wird, damit Exponenten als Zahlen ohne Vorzeichen ausgedrückt werden können, ist vorstehend dargestellt. Verzerrte Exponenten ähneln den Eigenschaften des binären Gleitkommaformats mit der Ausnahme, dass wie nachstehend unter Bezugnahme auf die

Klassen des Datentyps NNP-Datentyp-1 verzerrten Exponenten von nur Nullen und nur Einsen keine besondere Bedeutung beigemessen wird.

[0098] Mantisse: Es wird angenommen, dass sich das Binärkomma einer Zahl des NNP-Datentyps-1 links vom äußersten linken Bruchbit befindet. Links vom Binärkomma befindet sich ein implizites Einheitsbit, von dem angenommen wird, dass es sich bei normalen Zahlen um eins und bei Nullen um null handelt. Die Bruchzahl mit dem impliziten Einheitsbit auf der linken Seite ist die Mantisse der Zahl.

[0099] Der Wert eines normalen NNP-Datentyps-1 ist die mit der Basis 2 multiplizierte Mantisse, die mit dem unverzerrten Exponenten potenziert wird.

[0100] Werte von Nichtnullzahlen: Die Werte von Nichtnullzahlen sind nachstehend aufgeführt:

Zahlenklasse	Wert
Normale Zahlen	± 2e-31 x (1.f)

[0101] Dabei ist e ein verzerrter Exponent in dezimaler Darstellung und f eine Bruchzahl in binärer Darstellung.

[0102] In einer Ausführungsform gibt es drei Klassen von Daten des NNP-Datentyps-1, darunter numerische und verwandte nichtnumerische Einheiten. Jedes Datenelement enthält ein Vorzeichen, einen Exponenten und eine Mantisse. Der Exponent ist so verzerrt, dass alle verzerrten Exponenten nichtnegative Zahlen ohne Vorzeichen sind und der kleinste verzerrte Exponent null ist. Die Mantisse enthält eine explizite Bruchzahl und ein implizites Einheitsbit links vom Binärkomma. Das Vorzeichenbit ist null für Plus und eins für Minus.

[0103] Alle erlaubten endlichen Nichtnullzahlen weisen eine eindeutige NNP-Datentyp-1-Darstellung auf. Es gibt keine subnormalen Zahlen, die für dieselben Werte mehrere Darstellungen zulassen, und es gibt keine subnormalen arithmetischen Operationen. Zu den drei Klassen gehören zum Beispiel:

Datenklasse	Vorzeichen		Verzerrter Exponent	
Einheitsbit*	Bruchzahl			
Null	±	0	0	0
Normale Zahlen	±	0	1	Nicht 0
Normale Zahlen	±	Nicht 0, nicht nurEinsen	1	Beliebig
Normale Zahlen	±	Nur Einsen	-	Nicht nur Einsen
NINF	±	Nur Einsen	-	Nur Einsen

[0104] Wobei: - „nicht zutreffend“ bedeutet, * bedeutet, dass das Einheitsbit implizit ist und NINF keine Zahl oder unendlich ist.

[0105] Weitere Einzelheiten zu den einzelnen Klassen werden nachstehend beschrieben:

Nullen: Nullen haben einen verzerrten Exponenten von null und eine Bruchzahl null. Das implizite Einheitsbit ist null.

[0106] Normale Zahlen: Normale Zahlen können einen verzerrten Exponenten von beliebigem Wert haben. Wenn der verzerrte Exponent 0 ist, muss die Bruchzahl ungleich null sein. Wenn der verzerrte Exponent nur aus Einsen besteht, darf die Bruchzahl nicht nur aus Einsen bestehen. Andere verzerrte Exponentenwerte können eine beliebige Bruchzahl aufweisen. Das implizite Einheitsbit ist bei allen normalen Zahlen eins.

[0107] NINF: Ein NINF wird durch einen verzerrten Exponenten von nur Einsen und eine Bruchzahl von nur Einsen dargestellt. Ein NINF stellt einen Wert dar, der nicht im Bereich der darstellbaren Werte des NNP-Datentyps-1 liegt (d.h. ein 16-Bit-Gleitkomma, das für Deep Learning konzipiert wurde und 6 Exponentenbits und 9 Bruchbits aufweist). In der Regel werden NINFs nur bei Berechnungen weitergeleitet, damit sie am Ende sichtbar bleiben.

[0108] Der Datentyp NNP-Datentyp-1 wird zwar in einem Beispiel unterstützt, andere spezialisierte oder nichtstandardisierte Datentypen können jedoch auch unterstützt werden, sowie ein oder mehrere Standarddatentypen, darunter, ohne auf diese beschränkt zu sein: IEEE 754 kurze Genauigkeit, binäres 16-Bit-Gleitkomma, IEEE Gleitkommazahlen mit halber Genauigkeit, 8-Bit-Gleitkommazahlen, 4-Bit-Ganzzahlenformat und/oder 8-Bit-Ganzzahlenformat, um nur einige zu nennen. Diese Datenformate haben unterschiedliche Qualitäten für die Verarbeitung in neuronalen Netzwerken. Kleinere Datentypen (z.B. weniger Bits) können zum Beispiel schneller verarbeitet werden und weniger Cache/Speicher in Anspruch nehmen, und größere Datentypen stellen eine höhere Ergebnisgenauigkeit im neuronalen Netzwerk bereit. Einem zu unterstützenden Datentyp sind ein oder mehrere Bits im Abfrageparameterblock zugewiesen (z.B. im Feld 336 für installierte Datentypen des Parameterblocks 330). Spezialisierte oder nichtstandardisierte Datentypen, die von einem bestimmten Prozessor unterstützt werden, werden zum Beispiel im Feld für die installierten Datentypen angegeben, Standarddatentypen werden jedoch nicht angegeben. In anderen Ausführungsformen werden auch ein oder mehrere Standarddatentypen angegeben. Andere Varianten sind möglich.

[0109] In einem bestimmten Beispiel ist Bit 0 des Feldes 336 für installierte Datentypen für den Datentyp NNP-Datentyp-1 reserviert, und wenn es auf z.B. 1 gesetzt ist, zeigt dies an, dass der Prozessor den NNP-Datentyp-1 unterstützt. In einem Beispiel ist der Bitvektor der installierten Datentypen so konfiguriert, dass er bis zu 16 Datentypen darstellt, wobei jedem Datentyp ein Bit zugewiesen ist. Ein Bitvektor in anderen Ausführungsformen kann jedoch mehr oder weniger Datentypen unterstützen. Ein Vektor kann weiterhin so konfiguriert sein, dass einem Datentyp ein oder mehrere Bits zugewiesen sind. Viele Beispiele sind möglich und/oder zusätzliche, weniger und/oder andere Datentypen können unterstützt und/oder im Vektor angegeben werden.

[0110] In einem Beispiel ruft die Abfragefunktion eine Angabe über die in dem modellabhängigen Prozessor installierten Datentypen ab und legt die Angabe im Parameterblock ab, indem z.B. ein oder mehrere Bits in dem Feld 336 für die installierten Datentypen des Parameterblocks 330 gesetzt werden. In einem Beispiel ruft die Abfragefunktion weiterhin eine Angabe über die installierten Datenlayout-Formate (ein anderes Datenattribut) ab und legt die Informationen im Parameterblock ab, indem z.B. ein oder mehrere Bits in dem Feld 338 für die installierten Datenlayout-Formate gesetzt werden. Zu beispielhaften Datenlayout-Formaten gehören zum Beispiel ein 4D-Merkmalstensor-Layout und ein 4D-Kernel-Tensor-Layout. Das 4D-Merkmalstensor-Layout wird in einem Beispiel von den hier angegebenen Funktionen verwendet, und in einem Beispiel verwendet die Faltungsfunktion das 4D-Kernel-Tensor-Layout. Diese Datenlayout-Formate ordnen die Daten im Speicher für einen Tensor so an, dass die Verarbeitungseffizienz beim Ausführen der Funktionen der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk erhöht wird. Um effizient zu arbeiten, verwendet die Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk zum Beispiel Eingabe-Tensoren, die in bestimmten Datenlayout-Formaten bereitgestellt werden. Es werden zwar beispielhafte Layouts bereitgestellt, für die hier beschriebenen Funktionen und/oder andere Funktionen können jedoch auch zusätzliche, weniger und/oder andere Layouts bereitgestellt werden.

[0111] Die Verwendung oder Verfügbarkeit von Layouts für ein bestimmtes Prozessormodell wird durch den Vektor der installierten Datenlayout-Formate bereitgestellt (z.B. Feld 338 des Parameterblocks 330). Bei dem Vektor handelt es sich zum Beispiel um einen Bitvektor der installierten Datenlayout-Formate, der es der CPU ermöglicht, den Anwendungen zu übermitteln, welche Layouts unterstützt werden. Bit 0 ist zum Beispiel für das 4D-Merkmalstensor-Layout reserviert, und wenn es z.B. auf 1 gesetzt wird, zeigt dies an, dass der Prozessor ein 4D-Merkmalstensor-Layout unterstützt; und Bit 1 ist für das 4D-Kernel-Tensor-Layout reserviert, und wenn es z.B. auf 1 gesetzt wird, zeigt dies an, dass der Prozessor ein 4D-Kernel-Tensor-Layout unterstützt. In einem Beispiel ist der Bitvektor der installierten Datenlayout-Formate so konfiguriert, dass er bis zu 16 Datenlayouts darstellt, wobei jedem Datenlayout ein Bit zugewiesen ist. Ein Bitvektor in anderen Ausführungsformen kann jedoch mehr oder weniger Datenlayouts unterstützen. Ein Vektor kann weiterhin so konfiguriert sein, dass einem Datenlayout ein oder mehrere Bits zugewiesen sind. Viele Beispiele sind möglich. Weitere Einzelheiten zum 4D-Merkmalstensor-Layout und 4D-Kernel-Tensor-Layout werden nachstehend beschrieben. Jetzt oder in der Zukunft können auch andere Layouts verwendet werden, um die Leistung zu optimieren.

[0112] In einem Beispiel arbeitet die Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk mit 4D-Tensoren, d.h. Tensoren mit 4 Dimensionen. Diese 4D-Tensoren ergeben sich aus den hier beschriebenen generischen Eingabe-Tensoren z.B. zeilenweise, d.h., beim Aufzählen der Tensor-Elemente in aufsteigender Reihenfolge der Speicheradressen wird zunächst die innere Dimension E1 durch die Werte der E1-Indexgröße, beginnend mit 0 bis zu E1-Indexgröße-1, erhöht, bevor der Index der E2-Dimension erhöht wird und das Durchlaufen der E1-Dimension wiederholt wird. Der Index der äußeren Dimension, der sogenannten E4-Dimension, wird zuletzt erhöht.

[0113] Tensoren mit einer geringeren Anzahl von Dimensionen (z.B. 3D- oder 1D-Tensoren) werden als 4D-Tensoren dargestellt, wobei eine oder mehrere Dimensionen des 4D-Tensors die ursprünglichen Tensor-Dimensionen, die auf 1 gesetzt wurden, überschreiten.

[0114] Die Umwandlung eines generischen zeilenweisen 4D-Tensors mit den Dimensionen E4, E3, E2, E1 in ein 4D-Merkmalstensor-Layout (hier auch als 4D-Merkmalstensor eines NNPA-Datenlayout-Formats 0 bezeichnet) wird hier beschrieben:

[0115] Ein sich ergebender Tensor kann zum Beispiel als ein 4D-Tensor von z.B. 64-Elementvektoren oder als 5D-Tensor mit folgenden Dimensionen dargestellt werden:

E4, [E1/64], E3, [E2/32] * 32, 64, wobei sich [] auf eine Obergrenze-Funktion (ceil function) bezieht. (Anders ausgedrückt: E4 * E3 * ceil (E2/32) * 32 * ceil (E1/64) * 64 Elemente.)

[0116] Ein Element [e4][e3][e2][e1] des generischen Tensors kann auf das folgende Element des sich ergebenden 5D-Tensors abgebildet werden:

$[e4][\lfloor e1/64 \rfloor][e3][e2][e1 \text{ MOD } 64]$, wobei $\lfloor \rfloor$ eine Untergrenze-Funktion (floor function) ist und mod für Modulo steht. (Anders ausgedrückt:
Element $(E3 * e2\_limit *$

$e1\_limit * e4x) + (e2\_limit * e3x * 64) + (e2x * 64) + (\lfloor e1x / 64 \rfloor * e2\_limit * E3 * 64) + (e1x$

$\text{mod } 64)$, wobei $e2\_limit = \lceil E2 / 32 \rceil * 32$ und $e1\_limit = \lceil E1 / 64 \rceil * 64$.)

[0117] Der sich ergebende Tensor kann größer sein als der generische Tensor. Elemente des sich ergebenden Tensors, die keine entsprechenden Elemente im generischen Tensor haben, werden als Füllelemente bezeichnet.

[0118] Man betrachte das Element [fe4][fe1][fe3][fe2][fe0] eines 4D-Merkmalstensors mit dem NNPA-Datenlayout-Format 0 eines 64-Elementvektors oder seine entsprechende Darstellung als ein 5D-Tensor von Elementen. Bei diesem Element handelt es sich entweder um ein Füllelement, oder sein entsprechendes Element im generischen 4D-Tensor mit den Dimensionen E4, E3, E2, E1 kann mit der folgenden Formel bestimmt werden:

- wenn fe2 ≥ E2, ist dies ein E2- (oder Seiten-)Füllelement
- sonst, wenn e1*64+fe0 ≥ E1, ist dies ein E1- (oder Zeilen-)Füllelement
- sonst ist das entsprechende Element im generischen 4D-Tensor:

$[fe4][fe3][fe2][fe1 * 64 + fe0]$

[0119] Für Modelle der künstlichen Intelligenz, die auf faltenden neuronalen Netzwerken beruhen, kann die Bedeutung der 4 Dimensionen eines Merkmalstensors allgemein wie folgt abgebildet werden:

- E4: N - Größe des Mini-Batch
- E3: H - Höhe des 3D-Tensors/Bildes
- E2: W - Breite des 3D-Tensors/Bildes
- E1: C - Kanäle oder Klassen des 3D-Tensors

[0120] Für Modelle der künstlichen Intelligenz, die auf maschinellem Lernen oder rekurrenten neuronalen Netzwerken beruhen, kann die Bedeutung der 4 Dimensionen eines 4D-Merkmalstensors allgemein wie folgt abgebildet werden:

- E4: T - Anzahl der Zeitschritte oder Modelle
- E3: Reserviert, allgemein auf 1 gesetzt
- E2: N_{mb} - Größe des Mini-Batch
- E1: L- Merkmale

[0121] Das NNPA-Datenlayout-Format 0 stellt z.B. eine zweidimensionale Datenlokalität mit 4-KB-Datenblöcken (Seiten) sowie eine 4-KB-Blockdatenausrichtung für die äußeren Dimensionen des erzeugten Tensors bereit.

[0122] Füllelement-Bytes werden für die Eingabe-Tensoren ignoriert und sind für Ausgabe-Tensoren nicht vorhersehbar. Die PER-Speicheränderung auf Füllbytes ist nicht vorhersehbar.

[0123] Ein Beispiel eines Eingabedatenlayouts für ein 4D-Merkmalstensor-Layout mit den Dimensionen E1, E2, E3 und E4 ist in den **Fig.** 5A bis 5C dargestellt, und eine beispielhafte Ausgabe für das 4D-Merkmalstensor-Layout ist in den **Fig.** 6A bis 6C dargestellt. Mit Bezug auf **Fig.** 5A ist ein 3D-Tensor 500 mit den Dimensionen E1, E2 und E3 dargestellt. In einem Beispiel enthält jeder 3D-Tensor eine Mehrzahl von 2D-Tensoren 502. Die Zahlen in jedem 2D-Tensor 502 beschreiben Speicher-Offsets in Bezug darauf, wo sich jedes der Elemente im Speicher befinden würde. Die Eingaben werden verwendet, um die Daten des ursprünglichen Tensors (z.B. ursprünglicher 4D-Tensor der **Fig.** 5A bis 5C) im Speicher anzuordnen, wie in den **Fig.** 6A bis 6C gezeigt, die den **Fig.** 5A bis 5C entsprechen.

[0124] In **Fig.** 6A enthält eine Speichereinheit 600 (z.B. eine Speicherseite) zum Beispiel eine vorausgewählte Anzahl (z.B. 32) von Zeilen 602, von denen jede durch z.B. e2_page_idx identifiziert wird; und jede Zeile hat eine vorausgewählte Anzahl (z.B. 64) von Elementen 604, die jeweils z.B. durch e1_page_idx identifiziert werden. Wenn eine Zeile nicht die vorausgewählte Anzahl von Elementen enthält, wird sie aufgefüllt (606), was als Zeilen- oder E1-Auffüllung bezeichnet wird; und wenn die Speichereinheit keine vorausgewählte Anzahl von Zeilen hat, wird sie aufgefüllt (608), was als Seiten- oder E2-Auffüllung bezeichnet wird. Die Zeilenauffüllung beispielsweise besteht z.B. aus Nullen oder anderen Werten, und die Seitenauffüllung besteht z.B. aus vorhandenen Werten, Nullen oder anderen Werten.

[0125] In einem Beispiel werden die Ausgabeelemente einer Zeile im Speicher (z.B. auf einer Seite) auf der Grundlage der Elementpositionen in der E1-Richtung der entsprechenden Eingabe bereitgestellt. Mit Bezug auf **Fig.** 5A werden zum Beispiel die Elementpositionen 0, 1 und 2 der drei dargestellten Matrizen (z.B. Elementpositionen an derselben Stelle in jeder Matrix) in Zeile 0 der Seite 0 von **Fig.** 6A dargestellt usw. In diesem Beispiel ist der 4D-Tensor klein, und alle Elemente jedes 2D-Tensors, der den 4D-Tensor darstellt, passen auf eine Seite. Dies ist jedoch nur ein Beispiel. Ein 2D-Tensor kann eine oder mehrere Seiten enthalten. Wenn ein 2D-Tensor auf der Grundlage einer Umformatierung eines 4D-Tensors erzeugt wird, beruht die Anzahl der Seiten des 2D-Tensors auf der Größe des 4D-Tensors. In einem Beispiel werden eine oder mehrere Obergrenze-Funktionen verwendet, um die Anzahl von Zeilen in einem 2D-Tensor und die Anzahl von Elementen in jeder Zeile zu ermitteln, dadurch wird angegeben, wie viele Seiten verwenden werden müssen. Andere Varianten sind möglich.

[0126] Zusätzlich zu dem 4D-Merkmalstensor-Layout kann ein Prozessor eines neuronalen Netzwerks in einem Beispiel einen 4D-Kernel-Tensor unterstützen, der die Elemente eines 4D-Tensors neu anordnet, um die Anzahl der Speicherzugriffe und Datenerfassungsschritte beim Ausführen bestimmter Operationen der künstlichen Intelligenz (z.B. der Unterstützung für die Verarbeitung in einem neuronalen Netzwerk), beispielsweise eine Faltung, zu verringern. Ein zeilenweiser, generischer 4D-Tensor mit den Dimensionen E4, E3, E2, E1 wird zum Beispiel in einen hier beschriebenen 4D-Kernel-Tensor mit einem NNPA-Datenlayout-Format 1 (4D-Kernel-Tensor) umgewandelt:

Ein sich ergebender Tensor kann als ein 4D-Tensor von z.B. 64-Elementvektoren oder als 5D-Tensor mit folgenden Dimensionen dargestellt werden:

$\lceil E1/64 \rceil, E4, E3, \lceil E2/32 \rceil * 32, 64$ wobei sich [] auf eine Obergrenze-Funktion bezieht. (Anders ausgedrückt: E4 * E3 * ceil (E2/32) * 32 * ceil (E1/64) * 64 Elemente.)

Ein Element [e4][e3][e2][e1] des generischen Tensors kann auf das folgende Element des sich ergebenden 5D-Tensors abgebildet werden:

$\lfloor e1/64 \rfloor [e4][e3][e2][e1 \bmod 64]$, wobei [] eine Untergrenze-Funktion ist und mod für Modulo steht. Anders ausgedrückt: Element

$(\lfloor e1x/64 \rfloor * E4 * E3 * e2\_limit * 64) +$

$(e4x * E3 * e2\_limit * 64) + (e3x * e2\_limit * 64) + (e2x * 64) + (e1x \bmod 64)$, wobei e2_limit

$= \lceil E2/32 \rceil * 32$ and $e1\_limit = \lceil E1/64 \rceil * 64$.

[0127] Der sich ergebende Tensor kann größer sein als der generische Tensor. Elemente des sich ergebenden Tensors, die keine entsprechenden Elemente im generischen Tensor haben, werden als Füllelemente bezeichnet.

[0128] Man betrachte das Element [fe1][fe4][fe3][fe2][fe0] eines 4D-Merkmalstensors mit dem NNPA-Datenlayout-Format 1 eines 64-Elementvektors oder seine entsprechende Darstellung als ein 5D-Tensor von Elementen. Bei diesem Element handelt es sich entweder um ein Füllelement, oder sein entsprechendes Element im generischen 4D-Tensor mit den Dimensionen E4, E3, E2, E1 kann mit der folgenden Formel bestimmt werden:

- wenn fe2 ≥ E2, ist dies ein E2- (oder Seiten-)Füllelement
- sonst, wenn e1*64+fe0 ≥ E1, ist dies ein E1- (oder Zeilen-)Füllelement
- sonst ist das entsprechende Element im generischen 4D-Tensor

$$[fe4][fe3][fe2][fe1 * 64 + fe0]$$

[0129] Für Modelle der künstlichen Intelligenz, die auf faltenden neuronalen Netzwerken beruhen, kann die Bedeutung der 4 Dimensionen eines Kernel-Tensors allgemein wie folgt abgebildet werden:

- E4: H - Höhe des 3D-Tensors/Bildes
- E3: W - Breite des 3D-Tensors/Bildes
- E2: C - Anzahl der Kanäle des 3D-Tensors
- E1: K - Anzahl der Kernels

[0130] Das NNPA-Datenlayout-Format 1 stellt z.B. einen zweidimensionalen Kernel-Parallelismus in 4-KB-Datenblöcken (Seiten) sowie eine 4-KB-Blockdatenausrichtung für die äußeren Dimensionen des erzeugten Tensors zur effizienten Verarbeitung bereit.

[0131] Füllbytes werden für die Eingabe-Tensoren ignoriert. Die PER-Speicheränderung auf Füllbytes ist nicht vorhersehbar.

[0132] Auch hier gilt, dass beispielhafte Datenlayout-Formate zwar ein 4D-Merkmalstensor-Layout und ein 4D-Kernel-Tensor-Layout enthalten, der Prozessor (z.B. der Prozessor 105 des neuronalen Netzwerks) auch andere Datenlayout-Formate unterstützen kann. Ein Hinweis auf unterstützte Datenlayouts wird abgerufen und in dem Abfrage-Parameterblock abgelegt, indem ein oder mehrere Bits z.B. im Feld 338 gesetzt werden.

[0133] Der Abfrage-Parameterblock enthält gemäß einem oder mehreren Aspekten der vorliegenden Erfindung auch andere Datenattributinformationen, die z.B. Informationen über die unterstützte Größe der Daten umfassen. Ein Prozessor wie ein Prozessor eines neuronalen Netzwerks weist auf der Grundlage von Größen des internen Puffers, Verarbeitungseinheiten, Datenbusstrukturen, Firmware-Beschränkungen usw. in der Regel Einschränkungen auf, die die maximale Größe der Tensor-Dimensionen und/oder die Gesamtgröße eines Tensors begrenzen können. Die Abfragefunktion stellt daher Felder bereit, um den Anwendungen diese Begrenzungen zu übermitteln. Der Prozessor ruft auf der Grundlage der Ausführung der Abfragefunktion zum Beispiel verschiedene Datengrößen wie eine maximale Dimensionsindexgröße (z.B. 65.536 Elemente) und eine maximale Tensor-Größe (z.B. 8 GB) ab und nimmt diese Informationen in die Felder 340 bzw. 342 des Parameterblocks (z.B. Parameterblock 330) auf. Der Prozessor (z.B. der Prozessor 105 des neuronalen Netzwerks) kann auch zusätzliche, weniger und/oder andere Größeninformationen unterstützen, die somit abgerufen und in dem Parameterblock, z.B. den Feldern 340, 342 und/oder anderen Feldern, abgelegt werden. In anderen Ausführungsformen können die Einschränkungen kleiner oder größer sein, und/oder die Größen können in anderen Einheiten angegeben werden, z.B. Bytes anstelle von Elementen, Elemente anstelle von Bytes usw. In anderen Ausführungsformen sind weiterhin unterschiedliche maximale Größen für jede einzelne Dimension möglich, anstatt dieselbe maximale Größe für alle Dimensionen. Zahlreiche Varianten sind möglich.

[0134] Gemäß einem oder mehreren Aspekten der vorliegenden Erfindung wird eine Abfragefunktion bereitgestellt, die detaillierte Informationen über ein spezifisches Modell eines ausgewählten Prozessors (z.B. Prozessor 105 des neuronalen Netzwerks) übermittelt. Zu den detaillierten Informationen gehören zum Beispiel modellabhängige Informationen über einen spezifischen Prozessor. (Ein Prozessor kann auch Standarddate-

nattribute wie Standarddatentypen, Standarddatenlayouts usw. unterstützen, die implizit sind und nicht unbedingt von der Abfragefunktion vorgestellt werden; obwohl die Abfragefunktion in anderen Ausführungsformen alle oder verschiedene ausgewählte Teilsätze von Datenattributen usw. anzeigen kann.) Beispielhafte Ausführungsformen werden zwar bereitgestellt, in anderen Ausführungsformen können jedoch andere Informationen bereitgestellt werden. Die abgerufenen Informationen, die für verschiedene Modelle eines Prozessors und/oder verschiedene Prozessoren unterschiedlich sein können, werden für das Durchführen von künstlicher Intelligenz und/oder einer anderen Verarbeitung verwendet. Die künstliche Intelligenz und/oder andere Verarbeitung können eine oder mehrere Nichtabfragefunktionen von z.B. der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk verwenden. Eine spezifische Nichtabfragefunktion, die bei der Verarbeitung verwendet wird, wird durchgeführt, indem die Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk ein oder mehrere Male ausgeführt wird und die spezifische Nichtabfragefunktion angegeben wird.

[0135] Weitere Einzelheiten für beispielhafte Nichtabfragefunktionen, die von der Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk unterstützt werden, werden nachstehend beschrieben (in anderen Ausführungsformen können zusätzliche, weniger und/oder andere Funktionen unterstützt werden):

Funktionscode 16: NNPA-ADD (Addition)

[0136] Wenn die NNPA-ADD-Funktion spezifiziert ist, wird jedes Element des durch den Tensor-Deskriptor 1 beschriebenen Eingabe-Tensors 1 zum entsprechenden Element des durch den Tensor-Deskriptor 2 beschriebenen Eingabe-Tensors 2 addiert, und die sich ergebende Summe wird in dem entsprechenden Element des durch den Ausgabe-Tensor-Deskriptor beschriebenen Ausgabe-Tensors abgelegt.

[0137] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode, z.B. 0010 hex bzw. 0011 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0138] Form, Datenlayout und Datentyp von Eingabe-Tensor 1, Eingabe-Tensor 2 und Ausgabe-Tensor müssen in einem Beispiel gleich sein; andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt.

[0139] Der Ausgabe-Tensor-Deskriptor 2, der Eingabe-Tensor-Deskriptor 3, die funktionsspezifischen Parameter 1 bis 5 und die funktionsspezifischen Sicherungsbereichs-Adressfelder werden in einem Beispiel ignoriert.

Funktionscode 17: NNPA-SUB (Subtraktion)

[0140] Wenn die NNPA-SUB-Funktion spezifiziert ist, wird jedes Element des durch den Tensor-Deskriptor 2 beschriebenen Eingabe-Tensors 2 von dem entsprechenden Element des durch den Tensor-Deskriptor 1 beschriebenen Eingabe-Tensors 1 subtrahiert, und die sich ergebende Differenz wird in dem entsprechenden Element des Ausgabe-Tensors abgelegt.

[0141] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode, z.B. 0010 hex bzw. 0011 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0142] Form, Datenlayout und Datentyp von Eingabe-Tensor 1, Eingabe-Tensor 2 und Ausgabe-Tensor müssen in einem Beispiel gleich sein; andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt.

[0143] Der Ausgabe-Tensor-Deskriptor 2, der Eingabe-Tensor-Deskriptor 3, die funktionsspezifischen Parameter 1 bis 5 und die funktionsspezifischen Sicherungsbereichs-Adressfelder werden in einem Beispiel ignoriert.

Funktionscode 18: NNPA-MUL (Multiplikation)

[0144] Wenn die NNPA-MUL-Funktion spezifiziert ist, wird das Produkt jedes Elements des durch den Tensor-Deskriptor 1 beschriebenen Eingabe-Tensors 1 (Multiplikator) und des entsprechenden Elements des durch den Tensor-Deskriptor 2 beschriebenen Eingabe-Tensors 2 (Multiplikand) in dem entsprechenden Element des Ausgabe-Tensors abgelegt.

[0145] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode, z.B. 0010 hex bzw. 0011 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0146] Form, Datenlayout und Datentyp von Eingabe-Tensor 1, Eingabe-Tensor 2 und Ausgabe-Tensor müssen in einem Beispiel gleich sein; andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt.

[0147] Der Ausgabe-Tensor-Deskriptor 2, der Eingabe-Tensor-Deskriptor 3, die funktionsspezifischen Parameter 1 bis 5 und die funktionsspezifischen Sicherungsbereichs-Adressfelder werden in einem Beispiel ignoriert.

Funktionscode 19: NNPA-DIV (Division)

[0148] Wenn die NNPA-DIV-Funktion spezifiziert ist, wird jedes Element des durch den Tensor-Deskriptor 1 beschriebenen Eingabe-Tensors 1 (Dividend) durch das entsprechende Element des durch den Tensor-Deskriptor 2 beschriebenen Eingabe-Tensors 2 (Divisor) dividiert, und der Quotient wird in dem entsprechenden Element des Ausgabe-Tensors abgelegt.

[0149] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode, z.B. 0010 hex bzw. 0011 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0150] Form, Datenlayout und Datentyp von Eingabe-Tensor 1, Eingabe-Tensor 2 und Ausgabe-Tensor müssen in einem Beispiel gleich sein; andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt.

[0151] Der Ausgabe-Tensor-Deskriptor 2, der Eingabe-Tensor-Deskriptor 3, die funktionsspezifischen Parameter 1 bis 5 und die funktionsspezifischen Sicherungsbereichs-Adressfelder werden in einem Beispiel ignoriert.

Funktionscode 20: NNPA-MIN (Minimum)

[0152] Wenn die NNPA-MIN-Funktion spezifiziert ist, wird jedes Element des durch den Tensor-Deskriptor 1 beschriebenen Eingabe-Tensors 1 mit dem entsprechenden Element des durch den Tensor-Deskriptor 2 beschriebenen Eingabe-Tensors 2 verglichen. Der kleinere der beiden Werte wird in dem entsprechenden Element des Ausgabe-Tensor-Deskriptors abgelegt. Wenn beide Werte gleich sind, wird der Wert in dem entsprechenden Element des Ausgabe-Tensors abgelegt.

[0153] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode, z.B. 0010 hex bzw. 0011 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0154] Form, Datenlayout und Datentyp von Eingabe-Tensor 1, Eingabe-Tensor 2 und Ausgabe-Tensor müssen in einem Beispiel gleich sein; andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt.

[0155] Der Ausgabe-Tensor-Deskriptor 2, der Eingabe-Tensor-Deskriptor 3, die funktionsspezifischen Parameter 1 bis 5 und die funktionsspezifischen Sicherungsbereichs-Adressfelder werden in einem Beispiel ignoriert.

Funktionscode 21: NNPA-MAX (Maximum)

[0156] Wenn die NNPA-MAX-Funktion spezifiziert ist, wird jedes Element des durch den Tensor-Deskriptor 1 beschriebenen Eingabe-Tensors 1 mit dem entsprechenden Element des durch den Tensor-Deskriptor 2 beschriebenen Eingabe-Tensors 2 verglichen. Der größere der beiden Werte wird in dem entsprechenden Element des Ausgabe-Tensor-Deskriptors abgelegt. Wenn beide Werte gleich sind, wird der Wert in dem entsprechenden Element des Ausgabe-Tensors abgelegt.

[0157] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode, z.B. 0010 hex bzw. 0011 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0158] Form, Datenlayout und Datentyp von Eingabe-Tensor 1, Eingabe-Tensor 2 und Ausgabe-Tensor müssen in einem Beispiel gleich sein; andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt.

[0159] Der Ausgabe-Tensor-Deskriptor 2, der Eingabe-Tensor-Deskriptor 3, die funktionsspezifischen Parameter 1 bis 5 und die funktionsspezifischen Sicherungsbereichs-Adressfelder werden in einem Beispiel ignoriert.

Funktionscode 32: NNPA-LOG (Natürlicher Logarithmus)

[0160] Wenn die NNPA-LOG-Funktion spezifiziert ist, ist für jedes Element des durch den Tensor-Deskriptor 1 beschriebenen Eingabe-Tensors, sofern dieses Element größer als null ist, das entsprechende Element des durch den Ausgabe-Tensor-Deskriptor beschriebenen Ausgabe-Tensors der natürliche Logarithmus dieses Elements. Andernfalls ist das entsprechende Element im Ausgabe-Tensor nicht numerisch darstellbar, und der der negativen Unendlichkeit zugehörige Wert im Zieldatentyp wird gespeichert.

[0161] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode, z.B. 0010 hex bzw. 0011 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0162] Form, Datenlayout und Datentyp von Eingabe-Tensor 1 und Ausgabe-Tensor müssen in einem Beispiel gleich sein; andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt.

[0163] Der Ausgabe-Tensor-Deskriptor 2, der Eingabe-Tensor-Deskriptor 2, der Eingabe-Tensor-Deskriptor 3, die funktionsspezifischen Parameter 1 bis 5 und die funktionsspezifischen Sicherungsbereichs-Adressfelder werden in einem Beispiel ignoriert.

Funktionscode 33: NNPA-EXP (Exponential)

[0164] Wenn die NNPA-EXP-Funktion spezifiziert ist, ist für jedes Element des durch den Tensor-Deskriptor 1 beschriebenen Eingabe-Tensors das entsprechende Element des durch den Ausgabe-Tensor-Deskriptor beschriebenen Ausgabe-Tensors das Exponential dieses Elements.

[0165] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode, z.B. 0010 hex bzw. 0011 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0166] Form, Datenlayout und Datentyp von Eingabe-Tensor 1 und Ausgabe-Tensor müssen in einem Beispiel gleich sein; andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt.

[0167] Der Ausgabe-Tensor-Deskriptor 2, der Eingabe-Tensor-Deskriptor 2, der Eingabe-Tensor-Deskriptor 3, die funktionsspezifischen Parameter 1 bis 5 und die funktionsspezifischen Sicherungsbereichs-Adressfelder werden in einem Beispiel ignoriert.

Funktionscode 49: NNPA-RELU (Rectified Linear Unit = Gleichgerichtete Lineareinheit)

[0168] Wenn die NNPA-RELU-Funktion spezifiziert ist, ist für jedes Element des durch den Tensor-Deskriptor 1 beschriebenen Eingabe-Tensors, sofern dieses Element kleiner oder gleich null ist, das entsprechende Element des durch den Ausgabe-Tensor-Deskriptor beschriebenen Ausgabe-Tensors null. Ansonsten ist das entsprechende Element im Ausgabe-Tensor das Minimum des Elements im Eingabe-Tensor und des im funktionsspezifischen Parameter 2 spezifizierten Begrenzungswerts (clipping value).

[0169] Der funktionsspezifische Parameter 1 definiert zum Beispiel den Begrenzungswert für die RELU-Operation. Der Begrenzungswert befindet sich zum Beispiel in den Bits 16 bis 31 des funktionsspezifischen Parameters 1. Der Begrenzungswert wird z.B. im NNPA-Datentyp-1-Format spezifiziert. Ein Begrenzungswert von null bedeutet, dass der maximale positive Wert verwendet wird; mit anderen Worten: es wird keine Begrenzung vorgenommen. Wenn ein negativer Wert spezifiziert ist, wird eine allgemeine Operandendaten-Ausnahme erkannt.

[0170] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode, z.B. 0010 hex bzw. 0011 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0171] Form, Datenlayout und Datentyp von Eingabe-Tensor 1 und Ausgabe-Tensor müssen in einem Beispiel gleich sein; andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt.

[0172] Der Ausgabe-Tensor-Deskriptor 2, der Eingabe-Tensor-Deskriptor 2, der Eingabe-Tensor-Deskriptor 3 und die funktionsspezifischen Sicherungsbereichs-Adressfelder werden in einem Beispiel ignoriert. Die funktionsspezifischen Parameter 2 bis 5 müssen in einem Beispiel Nullen enthalten.

Funktionscode 50: NNPA-TANH

[0173] Wenn die NNPA-TANH-Funktion spezifiziert ist, ist für jedes Element des durch den Tensor-Deskriptor 1 beschriebenen Eingabe-Tensors der entsprechende Elementwert des durch den Ausgabe-Tensor-Deskriptor beschriebenen Ausgabe-Tensors der hyperbolische Tangens dieses Elements.

[0174] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode, z.B. 0010 hex bzw. 0011 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0175] Form, Datenlayout und Datentyp von Eingabe-Tensor 1 und Ausgabe-Tensor müssen in einem Beispiel gleich sein; andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt.

[0176] Der Ausgabe-Tensor-Deskriptor 2, der Eingabe-Tensor-Deskriptor 2, der Eingabe-Tensor-Deskriptor 3, die funktionsspezifischen Parameter 1 bis 5 und die funktionsspezifischen Sicherungsbereichs-Adressfelder werden in einem Beispiel ignoriert.

Funktionscode 51: NNPA-SIGMOID

[0177] Wenn die NNPA-SIGMOID-Funktion spezifiziert ist, ist für jedes Element des durch den Tensor-Deskriptor 1 beschriebenen Eingabe-Tensors das entsprechende Element des durch den Ausgabe-Tensor-Deskriptor beschriebenen Ausgabe-Tensors das Sigmoid dieses Elements.

[0178] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode, z.B. 0010 hex bzw. 0011 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0179] Form, Datenlayout und Datentyp von Eingabe-Tensor 1 und Ausgabe-Tensor müssen in einem Beispiel gleich sein; andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt.

[0180] Der Ausgabe-Tensor-Deskriptor 2, der Eingabe-Tensor-Deskriptor 2, der Eingabe-Tensor-Deskriptor 3, die funktionsspezifischen Parameter 1 bis 5 und die funktionsspezifischen Sicherungsbereichs-Adressfelder werden in einem Beispiel ignoriert.

Funktionscode 52: NNPA-SOFTMAX

[0181] Wenn die NNPA-SOFTMAX-Funktion spezifiziert ist, wird für jeden Vektor in Dimension-1 des Eingabe-Tensors 1 der entsprechende Vektor im Ausgabe-Tensor wie folgt berechnet:

* Der Maximalwert des Vektors wird berechnet.

* Die Summe der Exponentiale der Differenz zwischen jedem Element in Dimension-1 des Vektors und dem vorstehend berechneten Maximalwert wird berechnet. Wenn sowohl das Element in Dimension-1 des Eingabevektors als auch der vorstehend berechnete Maximalwert numerische Werte sind und die Differenz nichtnumerisch ist, wird erzwungen, dass das Ergebnis dieses Exponentials für dieses Element auf null gesetzt wird.

* Für jedes Element im Vektor wird ein Zwischenquotient aus dem Exponential der Differenz zwischen dem Element und dem vorstehend berechneten Maximalwert, geteilt durch die vorstehend berechnete Summe, gebildet. Auf diesen Zwischenquotienten wird eine optionale Aktivierungsfunktion angewendet, um das entsprechende Element im Ausgabevektor zu bilden.

[0182] Dieser Prozess wird z.B. für alle Vektoren der Dimension-4-Indexgröße x, Dimension-3-Indexgröße x, Dimension-2-Indexgröße in Dimension-1 wiederholt.

[0183] In einem Beispiel steuert ein funktionsspezifischer NNPA-SOFTMAX-Parameter 1 die Aktivierungsfunktion. Ein ACT-Feld (z.B. Bits 28 bis 31) des funktionsspezifischen Parameters 1 spezifiziert zum Beispiel die Aktivierungsfunktion. Zu Beispielen für Aktivierungsfunktionen gehören:

ACT	Aktivierungsfunktion
0	Keine Aktivierungsfunktion durchgeführt
1	LOG
2 bis 15	Reserviert

[0184] Wenn für das ACT-Feld ein reservierter Wert spezifiziert ist, wird ein Antwortcode von z.B. F001 hex gemeldet, und die Operation wird mit einem Bedingungscode, z.B. 1, beendet.

[0185] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode, z.B. 0010 hex bzw. 0011 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0186] Wenn die Dimension-3-Indexgröße des Eingabe-Tensors in einem Beispiel ungleich eins ist, wird ein Antwortcode von z.B. F000 hex gespeichert, und die Anweisung wird mit Bedingungscodes, z.B. 1, beendet.

[0187] Form, Datenlayout und Datentyp von Eingabe-Tensor 1 und Ausgabe-Tensor müssen in einem Beispiel gleich sein; andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt.

[0188] Der Ausgabe-Tensor-Deskriptor 2, der Eingabe-Tensor-Deskriptor 2 und der Eingabe-Tensor-Deskriptor 3 werden in einem Beispiel ignoriert. Die funktionsspezifischen Parameter 2 bis 5 müssen in einem Beispiel Nullen enthalten.

[0189] Ein 8 KB großer funktionsspezifischer Sicherungsbereich kann von dieser Funktion verwendet werden.

[0190] Wenn der Vektor in Dimension-1 abgerufen wird, kann es in einer Ausführungsform vorkommen, dass die Elemente im Speicher je nach dem spezifizierten Datenlayout-Format nicht zusammenhängend sind. Wenn alle Elemente eines Vektors der Dimension-1 des Eingabe-Tensors 1 die größte negative Zahl enthalten, die im spezifizierten Datentyp darstellbar ist, können die Ergebnisse weniger genau sein.

Funktionscode 64: NNPA-BATCHNORM (Stapelnormalisierung)

[0191] Wenn die NNPA-BATCHNORM-Funktion spezifiziert ist, wird für jeden Vektor in Dimension-1 des Eingabe-Tensors 1 der entsprechende Vektor in Dimension-1 des Ausgabe-Tensors berechnet, indem jedes Element im Vektor mit dem entsprechenden Element im Vektor in Dimension-1 multipliziert wird, der den Eingabe-Tensor 2 bildet. Das Produkt mit vollständiger Genauigkeit wird anschließend zu dem entsprechenden Element im Dimension-1-Vektor addiert, der den Eingabe-Tensor 3 bildet, und anschließend wird auf die Genauigkeit des spezifizierten Datentyps des Ausgabe-Tensors gerundet. Dieser Prozess wird z.B. für alle Vektoren der Dimension-4-Indexgröße x, Dimension-3-Indexgröße x, Dimension-2-Indexgröße in Dimension-1 wiederholt.

[0192] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode, z.B. 0010 hex bzw. 0011 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0193] In einem Beispiel müssen die folgenden Bedingungen erfüllt sein, andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt:

* Form und Datenlayout des Eingabe-Tensors 1 und des Ausgabe-Tensors müssen gleich sein.

* Der Datentyp der Eingabe-Tensoren und des Ausgabe-Tensors muss gleich sein.

* Die Dimension-1-Indexgröße der Eingabe-Tensoren 1, 2, 3 und des Ausgabe-Tensors muss gleich sein.

* Die Indexgröße der Dimensionen 2, 3 und 4 der Eingabe-Tensoren 2 und 3 muss eins sein.

[0194] Der Ausgabe-Tensor-Deskriptor 2 und die funktionsspezifischen Sicherungsbereichs-Adressfelder werden in einem Beispiel ignoriert. Die funktionsspezifischen Parameter 2 bis 5 müssen in einem Beispiel Nullen enthalten.

Funktionscode 80: NNPA-MAXPOOL2D

Funktionscode 81: NNPA-AVGPOOL2D

[0195] Wenn entweder die NNPA-MAXPOOL2D- oder die NNPA-AVGPOOL2D-Funktion spezifiziert ist, wird der von dem Eingabe-Tensor-Deskriptor 1 beschriebene Eingabe-Tensor 1 durch die spezifizierte Operation verringert, um Fenster der Eingabe zusammenzufassen. Die Fenster der Eingabe werden ausgewählt, indem ein 2D-Gleitfenster über die Dimensionsindizes 2 und 3 bewegt wird. Die Zusammenfassung des Fensters ist ein Element des Ausgabe-Tensors. Die Dimensionen des Gleitfensters werden z.B. durch den funktionsspezifischen Parameter 4 und den funktionsspezifischen Parameter 5 beschrieben. Der Abstand, um den sich das Gleitfenster beim Berechnen benachbarter Elemente des Ausgabe-Tensors über den Eingabe-Tensor 1 bewegt, wird als Stride bezeichnet. Der Stride des Gleitfensters wird z.B. durch den funktionsspezifischen Parameter 2 und den funktionsspezifischen Parameter 3 spezifiziert. Wenn die NNPA-MAXPOOL2D-Operation spezifiziert ist, wird die nachstehend definierte Max-Operation bei dem Fenster durchgeführt. Wenn die NNPA-AVGPOOL2D-Operation spezifiziert ist, wird die nachstehend definierte AVG-Operation bei dem Fenster durchgeführt. Wenn der spezifizierte Auffüllungstyp „Gültig“ ist, werden alle Elemente im Fenster zu der Sammlung hinzugefügt, die zum Berechnen des sich ergebenden Ausgabeelements verwendet wird. Wenn der spezifizierte Auffüllungstyp „Gleich“ ist, kann je nach Lage des Fensters nur ein Teilsatz von Elementen des Fensters zu der Sammlung hinzugefügt werden, die zum Berechnen des sich ergebenden Ausgabeelements verwendet wird.

[0196] In einem Beispiel wird bei einer CollectElements-Operation ein Element zu der Sammlung von Elementen hinzugefügt, und die Anzahl der Elemente in der Sammlung wird erhöht. Jedes Mal, wenn sich die Anfangsposition des Fensters ändert, wird die Sammlung geleert. Es ist nicht vorhersehbar, ob auf Elemente zugegriffen wird, die zum Durchführen der Operation nicht erforderlich sind.

[0197] Max-Operation: In einem Beispiel wird der maximale Wert der Sammlung von Elementen im Fenster berechnet, indem alle Elemente in der Sammlung miteinander verglichen werden und der größte Wert zurückgegeben wird.

[0198] Avg-Operation (Durchschnitt): In einem Beispiel wird der Durchschnittswert der Sammlung von Elementen im Fenster als die Summe aller Elemente in der Sammlung, geteilt durch die Anzahl der Elemente in der Sammlung, berechnet.

[0199] In einem Beispiel werden die Felder wie folgt zugeordnet:

* Ein funktionsspezifischer Pooling-Parameter 1 steuert den Auffüllungstyp. Die Bits 29 bis 31 des funktionsspezifischen Parameters 1 enthalten zum Beispiel ein PAD-Feld (Auffüllungsfeld), das den Auffüllungstyp spezifiziert. Zu beispielhaften Typen gehören zum Beispiel:

PAD	Auffüllungstyp
0	Gültig
1	Gleich
2 bis 7	Reserviert

Wenn für das PAD-Feld ein reservierter Wert spezifiziert ist, wird ein Antwortcode von z.B. F000 hex gemeldet, und die Operation wird mit einem Bedingungscode, z.B. 1, beendet.
In einem Beispiel sind die Bitpositionen 0 bis 28 des funktionsspezifischen Parameters 1 reserviert und müssen Nullen enthalten.

* Der funktionsspezifische Parameter 2 enthält z.B. eine binäre 32-Bit-Ganzzahl ohne Vorzeichen, die den Dimension-2-Stride (D2S) spezifiziert, der die Anzahl von Elementen angibt, um die sich das Gleitfenster in Dimension 2 bewegt.

* Der funktionsspezifische Parameter 3 enthält z.B. eine binäre 32-Bit-Ganzzahl ohne Vorzeichen, die den Dimension-3-Stride (D3S) spezifiziert, der die Anzahl von Elementen angibt, um die sich das Gleitfenster in Dimension 3 bewegt.

* Der funktionsspezifische Parameter 4 enthält z.B. eine binäre 32-Bit-Ganzzahl ohne Vorzeichen, die die Dimension-2-Fenstergröße (D2WS) spezifiziert, die die Anzahl von Elementen in Dimension 2 angibt, die das Gleitfenster enthält.

* Der funktionsspezifische Parameter 5 enthält z.B. eine binäre 32-Bit-Ganzzahl ohne Vorzeichen, die die Dimension-3-Fenstergröße (D3WS) spezifiziert, die die Anzahl von Elementen in Dimension 3 angibt, die das Gleitfenster enthält.

[0200] In einem Beispiel müssen die in den funktionsspezifischen Parametern 2 bis 5 spezifizierten Werte kleiner oder gleich wie die maximale Dimensionsindexgröße sein, und die in den funktionsspezifischen Parametern 4 und 5 spezifizierten Werte müssen größer als null sein; andernfalls wird ein Antwortcode, z.B. 0012 hex, gemeldet, und die Operation wird mit einem Bedingungscode, z.B. 1, beendet.

[0201] Wenn der Dimension-2-Stride und der Dimension-3-Stride beide null sind und entweder die Dimension-2-Fenstergröße oder die Dimension-3-Fenstergröße größer als z.B. 1024 ist, wird ein Antwortcode, z.B. F001 hex, gespeichert. Wenn der Dimension-2-Stride und der Dimension-3-Stride beide größer als z.B. null sind und entweder die Dimension-2-Fenstergröße oder die Dimension-3-Fenstergröße größer als z.B. 64 ist, wird ein Antwortcode, z.B. F002 hex, gespeichert. Wenn der Dimension-2-Stride und der Dimension-3-Stride beide größer als z.B. null sind und entweder der Dimension-2-Stride oder der Dimension-3-Stride größer als z.B. 30 ist, wird ein Antwortcode, z.B. F003 hex, gespeichert. Wenn der Dimension-2-Stride und der Dimension-3-Stride beide größer als z.B. null sind und entweder die Dimension-2-Indexgröße des Eingabe-Tensors oder die Dimension-3-Indexgröße des Eingabe-Tensors größer als z.B. 1024 ist, wird ein Antwortcode, z.B. F004 hex, gespeichert. Bei allen vorstehend genannten Bedingungen wird die Anweisung mit einem Bedingungscode, z.B. 1, beendet.

[0202] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode,

z.B. 0010 hex bzw. 0011 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0203] In einem Beispiel müssen die folgenden Bedingungen erfüllt sein, andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt:

* Die Dimension-4-Indexgrößen und die Dimension-1-Indexgrößen des Eingabe-Tensors und des Ausgabe-Tensors müssen gleich sein.

* Das Datenlayout und der Datentyp des Eingabe-Tensors und des Ausgabe-Tensors müssen gleich sein.

* Wenn der Dimension-2-Stride und der Dimension-3-Stride beide null sind, müssen in einem Beispiel die folgenden zusätzlichen Bedingungen erfüllt sein:

* Die Dimension-2-Indexgröße des Eingabe-Tensors muss gleich wie die Dimension-2-Fenstergröße sein.

* Die Dimension-3-Indexgröße des Eingabe-Tensors muss gleich wie die Dimension-3-Fenstergröße sein.

* Die Dimension-2-Indexgröße und die Dimension-3-Indexgröße des Ausgabe-Tensors müssen eins sein.

* Die spezifizierte Auffüllung muss gültig sein.

* Wenn entweder der Dimension-2-Stride oder der Dimension-3-Stride ungleich null ist, müssen in einem Beispiel beide Strides ungleich null sein.

* Wenn der Dimension-2-Stride und der Dimension-3-Stride beide größer als null sind, müssen in einem Beispiel die folgenden zusätzlichen Bedingungen erfüllt sein:

* Wenn die spezifizierte Auffüllung „Gültig“ ist, muss die Dimension-2-Fenstergröße kleiner oder gleich wie die Dimension-2-Indexgröße des Eingabe-Tensors sein.

* Wenn die spezifizierte Auffüllung „Gültig“ ist, muss die Dimension-3-Fenstergröße kleiner oder gleich wie die Dimension-3-Indexgröße des Eingabe-Tensors sein.

* Wenn die spezifizierte Auffüllung „Gleich“ ist, müssen die folgenden Beziehungen zwischen der Dimension-2-Indexgröße und der Dimension-3-Indexgröße der Eingabe- und Ausgabe-Tensoren erfüllt sein (Pooling Same Padding):

$$O1D2IS = \left\lceil \frac{I1D2IS}{D2D} \right\rceil$$

$$O1D3IS = \left\lceil \frac{I1D3IS}{D3S} \right\rceil$$

wobei gilt:

IxDyIS	Dimension-y-Indexgröße des im Tensor-Deskriptor x definierten Ausgabe-Tensors x.
OxDyIS	Dimension-y-Indexgröße des im Tensor-Deskriptor x definierten Ausgabe-Tensors x.
D2S	Dimension-2-Stride.
D3S	Dimension-3-Stride.

* Wenn die spezifizierte Auffüllung „Gültig“ ist, müssen die folgenden Beziehungen zwischen der Dimension-2-Indexgröße und der Dimension-3-Indexgröße der Eingabe- und Ausgabe-Tensoren erfüllt sein (Pooling Valid Padding):

$$O1D2IS = \left\lceil \frac{(I1D2IS - D2WS + 1)}{D2S} \right\rceil$$

$$O1D3IS = \left\lceil \frac{(I1D2IS - D3WS + 1)}{D3S} \right\rceil$$

wobei D2WS die Dimension-2-Fenstergröße und D3WS die Dimension-3-Fenstergröße ist.

[0204] Der Ausgabe-Tensor-Deskriptor 2, die Eingabe-Tensor-Deskriptoren 2 und 3 und die funktionsspezifischen Sicherungsbereichs-Adressfelder werden ignoriert.

Funktionscode 96: NNPA-LSTMACT (Aktivierung langes Kurzzeitgedächtnis)

[0205] Wenn die NNPA-LSTMACT-Funktion spezifiziert ist, stellen der durch den Eingabe-Tensor-Deskriptor 1 beschriebene Eingabe-Tensor 1, der für jeden Dimension-4-Indexwert in vier Untertensoren aufgeteilt ist, zusammen mit dem durch den Eingabe-Tensor-Deskriptor 2 beschriebenen Eingabe-Tensor 2, der für jeden Dimension-4-Indexwert in vier Untertensoren aufgeteilt ist, und dem durch den Eingabe-Tensor-Deskriptor 3 beschriebenen Eingabe-Tensor 3 die Eingaben für eine LSTMACT-Operation dar. Am Ende der LSTMACT-Operation werden die Ergebnisse in den durch den Ausgabe-Tensor-Deskriptor 1 beschriebenen Ausgabe-Tensor 1 und den durch den Ausgabe-Tensor-Deskriptor 2 beschriebenen Ausgabe-Tensor 2 geschrieben.

[0206] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode 0010 hex bzw. 0011 hex im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0207] In einer Ausführungsform müssen die folgenden Bedingungen erfüllt sein, andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt:

* Die Dimension-4-Indexgröße für den Eingabe-Tensor 3 und die Ausgabe-Tensoren 1 und 2 muss gleich, z.B. 1, sein.

* Die Dimension-4-Indexgröße für den Eingabe-Tensor 1 und den Eingabe-Tensor 2 muss gleich, z.B. vier, sein.

* Die Dimension-3-Indexgröße für z.B. alle Eingabe-Tensoren und die beiden Ausgabe-Tensoren muss gleich, z.B. eins, sein.

* Das Datenlayout und der Datentyp von z.B. allen Eingabe-Tensoren und den beiden Ausgabe-Tensoren müssen gleich sein.

* Die Dimension-1-Indexgröße von z.B. allen Eingabe-Tensoren und den beiden Ausgabe-Tensoren muss gleich sein.

* Die Dimension-2-Indexgröße von z.B. allen Eingabe-Tensoren und den beiden Ausgabe-Tensoren muss gleich sein.

[0208] Die funktionsspezifischen Sicherungsbereichs-Adressfelder werden in einem Beispiel ignoriert. Die funktionsspezifischen Parameter 1 bis 5 müssen in einem Beispiel Nullen enthalten.

Funktionscode 97: NNPA-GRUACT (Aktivierung gegatterte rekurrente Einheit)

[0209] Wenn die NNPA-GRUACT-Funktion spezifiziert ist, stellen der durch den Eingabe-Tensor-Deskriptor 1 beschriebene Eingabe-Tensor 1, der für jeden Dimension-4-Indexwert in drei Untertensoren aufgeteilt ist, zusammen mit dem durch den Eingabe-Tensor-Deskriptor 2 beschriebenen Eingabe-Tensor 2, der für jeden Dimension-4-Indexwert in drei Untertensoren aufgeteilt ist, und dem durch den Eingabe-Tensor-Deskriptor 3 beschriebenen Eingabe-Tensor 3 die Eingaben für eine GRUACT-Operation dar. Am Ende der GRUACT-Operation wird der durch den Ausgabe-Tensor-Deskriptor beschriebene Ausgabe-Tensor gespeichert.

[0210] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode, z.B. 0010 hex bzw. 0011 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0211] In einer Ausführungsform müssen die folgenden Bedingungen erfüllt sein, andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt:

* Die Dimension-4-Indexgröße des Ausgabe-Tensors und des Eingabe-Tensors 3 muss gleich, z.B. eins, sein.

* Die Dimension-4-Indexgröße für den Eingabe-Tensor 1 und den Eingabe-Tensor 2 muss gleich, z.B. drei, sein.

* Die Dimension-3-Indexgröße für z.B. alle Eingabe-Tensoren und den Ausgabe-Tensor muss gleich, z.B. eins, sein.

* Die Dimension-1-Indexgröße von z.B. allen Eingabe-Tensoren und dem Ausgabe-Tensor muss gleich sein.

* Die Dimension-2-Indexgröße von z.B. allen Eingabe-Tensoren und dem Ausgabe-Tensor muss gleich sein.

* Das Datenlayout und der Datentyp von z.B. allen Eingabe-Tensoren und dem Ausgabe-Tensor müssen gleich sein.

[0212] Der Ausgabe-Tensor-Deskriptor 2 und die funktionsspezifischen Sicherungsbereichs-Adressfelder werden in einem Beispiel ignoriert. Die funktionsspezifischen Parameter 2 bis 5 müssen in einem Beispiel Nullen enthalten.

Funktionscode 112: NNPA-CONVOLUTION

[0213] Wenn die NNPA-CONVOLUTION-Funktion spezifiziert ist, wird für jedes Ausgabeelement in dem durch den Ausgabe-Tensor-Deskriptor 1 beschriebenen Ausgabe-Tensor ein dreidimensionales Eingabefenster 1, bestehend aus den Dimensionsindizes 3, 2 und 1, aus dem durch den Eingabe-Tensor-Deskriptor 1 beschriebenen Eingabe-Tensor 1 ausgewählt. Ein dreidimensionales Eingabefenster 2 der gleichen Größe, bestehend aus den Dimensionsindizes 4, 3 und 2, wird aus dem durch den Eingabe-Tensor-Deskriptor 2 beschriebenen Tensor 2 ausgewählt. Die Elemente im Eingabefenster 1 werden mit den entsprechenden Elementen im Eingabefenster 2 multipliziert, und alle Produkte werden addiert, um eine erste Summe zu bilden. Diese erste Summe wird zu dem entsprechenden Element des Eingabe-Tensors 3 addiert, um einen Zwischensummenwert zu berechnen. Das Element des Ausgabe-Tensors ist das Ergebnis der spezifizierten Aktivierungsfunktion, die bei der Zwischensumme durchgeführt wird. Wenn keine Aktivierungsfunktion spezifiziert ist, entspricht das Ausgabeelement der Zwischensumme.

[0214] Wenn der spezifizierte Auffüllungstyp „Gültig“ ist, werden alle Elemente im Fenster verwendet, um die sich ergebende erste Summe zu berechnen. Wenn der spezifizierte Auffüllungstyp „Gleich“ ist, können je nach Lage des Fensters einige Elemente des Eingabefensters 1 beim Berechnen der sich ergebenden ersten Summe als null angenommen werden.

[0215] Es ist nicht vorhersehbar, ob auf Elemente zugegriffen wird, die zum Durchführen der Operation nicht erforderlich sind.

[0216] In einem Beispiel werden die Felder eines von der Faltungsfunktion verwendeten funktionsspezifischen Parameters wie folgt zugeordnet:

* Ein funktionsspezifischer NNPA-CONVOLUTION-Parameter 1 steuert den Auffüllungstyp und die Aktivierungsfunktion. Die Bits 29 bis 31 des funktionsspezifischen Parameters 1 enthalten zum Beispiel ein PAD-Feld, das den Auffüllungstyp spezifiziert. Nachstehend sind beispielhafte Typen aufgeführt:

PAD	Auffüllungstyp
0	Gültig
1	Gleich
2 bis 7	Reserviert

[0217] Wenn für das PAD-Feld ein reservierter Wert spezifiziert ist, wird ein Antwortcode von z.B. F000 hex gemeldet, und die Operation wird mit einem Bedingungscode, z.B. 1, beendet.

[0218] Die Bits 24 bis 27 des funktionsspezifischen NNPA-CONVOLUTION-Parameters 1 enthalten in einem Beispiel weiterhin ein Aktivierungsfeld, das Aktivierungsfunktionen spezifiziert. Nachstehend sind beispielhafte Funktionen aufgeführt:

ACT	Aktivierungsfunktion
0	Keine Aktivierungsfunktion durchgeführt
1	RELU
2 bis 15	Reserviert

[0219] Wenn eine Aktivierungsfunktion RELU spezifiziert ist, wird der sich ergebende Wert des Ausgabeelements wie folgt ermittelt: Wenn der Zwischensummenwert kleiner oder gleich null ist, ist das entsprechende Element im Ausgabe-Tensor null; andernfalls ist das entsprechende Element im Ausgabe-Tensor das Minimum aus dem Zwischensummenwert und dem im funktionsspezifischen Parameter 4 spezifizierten Begrenzungswert.

[0220] Wenn für das ACT-Feld ein reservierter Wert spezifiziert ist, wird ein Antwortcode von z.B. F001 hex gemeldet, und die Operation wird mit einem Bedingungscode, z.B. 1, beendet.

* Der funktionsspezifische Parameter 2 enthält z.B. eine binäre 32-Bit-Ganzzahl ohne Vorzeichen, die den Dimension-2-Stride (D2S) spezifiziert, der die Anzahl von Elementen angibt, um die sich das Gleitfenster in Dimension 2 bewegt.

* Der funktionsspezifische Parameter 3 enthält z.B. eine binäre 32-Bit-Ganzzahl ohne Vorzeichen, die den Dimension-3-Stride (D3S) spezifiziert, der die Anzahl von Elementen angibt, um die sich das Gleitfenster in Dimension 3 bewegt.
Die in den funktionsspezifischen Parametern 2 und 3 spezifizierten Werte müssen kleiner als die maximale Dimensionsindexgröße sein; andernfalls wird ein Antwortcode, z.B. 0012 hex, gemeldet, und die Operation wird mit einem Bedingungscode, z.B. 1, beendet.

* Der funktionsspezifische Parameter 4 definiert den Begrenzungswert für die optionale RELU-Operation. In einem Beispiel befindet sich der Begrenzungswert in den Bits 16 bis 31 des funktionsspezifischen Parameters 4.

[0221] In einem Beispiel wird dieses Feld ignoriert, wenn das ACT-Feld null ist. Wenn das ACT-Feld RELU spezifiziert, wird der Begrenzungswert in dem NNP-Datentyp-1-Format spezifiziert. Ein Begrenzungswert von null bedeutet, dass der maximale positive Wert verwendet wird; mit anderen Worten: es wird keine Begrenzung vorgenommen. Wenn ein Nichtnull-Wert spezifiziert ist, wird eine allgemeine Operandendaten-Ausnahme erkannt.

[0222] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren mit Ausnahme von Eingabe-Tensor 2 keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp im Eingabe-Tensor 2 keinen 4D-Kernel-Tensor (z.B. Datenlayout = 1) spezifiziert, wird ein Antwortcode, z.B. 0010 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet. Wenn der Datentyp in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode, z.B. 0010 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0223] Wenn der Dimension-2-Stride und der Dimension-3-Stride beide null sind und die Dimension-3-Indexgröße oder die Dimension-4-Indexgröße des Eingabe-Tensors 2 größer als z.B. 448 ist, wird ein Antwortcode, z.B. F002 hex, gespeichert. Wenn der Dimension-2-Stride und der Dimension-3-Stride beide größer als null sind und entweder die Dimension-3-Indexgröße oder die Dimension-4-Indexgröße des Eingabe-Tensors 2 größer als z.B. 64 ist, wird ein Antwortcode, z.B. F003 hex, gespeichert, und die Operation wird mit einem Bedingungscode, z.B. 1, beendet. Wenn entweder der Dimension-2-Stride oder der Dimension-3-Stride größer als z.B. 13 ist, wird ein Antwortcode, z.B. F004 hex, gespeichert, und die Operation wird mit einem Bedingungscode, z.B. 1, beendet.

[0224] In einem Beispiel müssen die folgenden Bedingungen erfüllt sein, andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt:

* Das Datenlayout des Eingabe-Tensors 1, des Eingabe-Tensors 3 und des Ausgabe-Tensors muss gleich sein.

* Der Datentyp aller Eingabe-Tensoren und des Ausgabe-Tensors muss gleich sein.

* Die Indexgrößen von Dimension-2, Dimension-3 und Dimension-4 des Eingabe-Tensors 3 müssen 1 sein.

* Die Dimension-4-Indexgröße des Ausgabe-Tensors muss gleich wie die Dimension-4-Fenstergröße des Eingabe-Tensors 1 sein.

* Die Dimension-1-Indexgröße des Ausgabe-Tensors muss gleich wie die Dimension-1-Indexgröße des Eingabe-Tensors 2 und die Dimension-1-Indexgröße des Eingabe-Tensors 3 sein.

* Die Dimension-1-Indexgröße des Eingabe-Tensors 1 muss gleich wie die Dimension-2-Indexgröße des Eingabe-Tensors 2 sein.

* Wenn der Dimension-2-Stride und der Dimension-3-Stride beide null sind, müssen in einem Beispiel die folgenden zusätzlichen Bedingungen erfüllt sein:

* Die Dimension-2-Indexgröße des Eingabe-Tensors 1 muss gleich wie die Dimension-3-Indexgröße des Eingabe-Tensors 2 sein.

* Die Dimension-3-Indexgröße des Eingabe-Tensors 1 muss gleich wie die Dimension-4-Indexgröße des Eingabe-Tensors 2 sein.

* Die Dimension-2-Indexgröße und die Dimension-3-Indexgröße des Ausgabe-Tensors müssen eins sein.

* Die spezifizierte Auffüllung muss „Gültig“ sein.

* Wenn entweder der Dimension-2-Stride oder der Dimension-3-Stride ungleich null ist, müssen beide Strides ungleich null sein.

* Wenn der Dimension-2-Stride und der Dimension-3-Stride beide größer als null sind, müssen in einem Beispiel die folgenden zusätzlichen Bedingungen erfüllt sein:

* Wenn die spezifizierte Auffüllung „Gültig“ ist, muss die Dimension-2-Indexgröße des Eingabe-Tensors 1 größer oder gleich wie die Dimension-3-Indexgröße des Eingabe-Tensors 2 sein.

* Wenn die spezifizierte Auffüllung „Gültig“ ist, muss die Dimension-3-Indexgröße des Eingabe-Tensors 1 größer oder gleich wie die Dimension-4-Indexgröße des Eingabe-Tensors 2 sein.

* Wenn die spezifizierte Auffüllung „Gleich“ ist, müssen die folgenden Beziehungen zwischen der Dimension-2-Indexgröße und Dimension-3-Indexgröße des Eingabe-Tensors 1 und des Ausgabe-Tensors in einem Beispiel erfüllt sein (Convolution Same Padding):

$$O1D2IS = \left\lceil \frac{I1D2IS}{D2S} \right\rceil$$

$$O1D3IS = \left\lceil \frac{I1D3IS}{D3S} \right\rceil$$

wobei gilt:

O1D2IS	Dimension-2-Indexgröße des Ausgabe-Tensors.
O1D3IS	Dimension-3-Indexgröße des Ausgabe-Tensors.
I1D2IS	Dimension-2-Indexgröße des Eingabe-Tensors 1.
I1D3IS	Dimension-3-Indexgröße des Eingabe-Tensors 1.
D2S	Dimension-2-Stride.
D3S	Dimension-3-Stride.

* Wenn die spezifizierte Auffüllung „Gültig“ ist, müssen die folgenden Beziehungen zwischen der Dimension-2-Indexgröße und der Dimension-3-Indexgröße des Eingabe-Tensors 1, der Dimension-3-Indexgröße und der Dimension-4-Indexgröße des Eingabe-Tensors 2 und des Ausgabe-Tensors in einem Beispiel erfüllt sein (Convolution Valid Padding):

$$O1D2IS=\left\lceil \frac{(I1D2IS - I2D3IS + 1)}{D2S} \right\rceil$$

$$O1D3IS=\left\lceil \frac{(I1D3IS - I2D4IS + 1)}{D3S} \right\rceil$$

wobei gilt:

O1D2IS	Dimension-2-Indexgröße des Ausgabe- Tensors.
O1D3IS	Dimension-3-Indexgröße des Ausgabe-Tensors.
I1D2IS	Dimension-2-Indexgröße des Eingabe-Tensors 1.
I1D3IS	Dimension-3-Indexgröße des Eingabe-Tensors 1.
I2D3IS	Dimension-3-Indexgröße des Eingabe-Tensors 2.
I2D4IS	Dimension-4-Indexgröße des Eingabe-Tensors 2.
D2S	Dimension-2-Stride.
D3S	Dimension-3-Stride.

[0225] Der Ausgabe-Tensor-Deskriptor 2 und die funktionsspezifischen Sicherungsbereichs-Adressfelder werden in einem Beispiel ignoriert. Der funktionsspezifische Parameter 5 muss in einem Beispiel Nullen enthalten.

Funktionscode 113: NNPA-MATMUL-OP (Matrix-Multiplikationsoperation)

[0226] Wenn die NNPA-MATMUL-OP-Funktion spezifiziert ist, wird jedes Element in dem durch den Ausgabe-Tensor-Deskriptor beschriebenen Ausgabe-Tensor wie nachstehend in einem Beispiel beschrieben berechnet:

* Ein Dimension-1-Vektor wird aus dem durch den Eingabe-Tensor-Deskriptor 1 beschriebenen Eingabe-Tensor 1 unter Verwendung der nachstehend beschriebenen Operation „get-dimension-1-vector“ ausgewählt.

* Ein Dimension-2-Vektor wird aus dem durch den Eingabe-Tensor-Deskriptor 2 beschriebenen Eingabe-Tensor 2 unter Verwendung der nachstehend beschriebenen Operation „get-dimension-2-vector“ ausgewählt.

* Ein Zwischenpunktprodukt des Dimension-1-Vektors und des Dimension-2-Vektors wird unter Verwendung der nachstehend beschriebenen Punktproduktoperation berechnet.

* Eine Operation wird beim Zwischenpunktprodukt und dem Element des durch den Eingabe-Tensor-Deskriptor 3 beschriebenen Eingabe-Tensors 3 mit denselben Dimension-4-Index- und Dimension-1-Indexwerten wie beim Element des Ausgabe-Tensors durchgeführt. Das sich ergebende Element wird im Ausgabe-Tensor gespeichert. Eine fusionierte Operation wird vom funktionsspezifischen Parameter 1 festgelegt und nachstehend beschrieben.

[0227] Operation „Get-dimension-1-vector“: Für ein spezifiziertes Ausgabeelement wird ein Dimension-1-Vektor aus dem Eingabe-Tensor 1 ausgewählt, wobei der Eingabe-Dimension-4-Index dem Ausgabe-Dimension-4-Index, der Eingabe-Dimension-3-Index dem Ausgabe-Dimension-3-Index und der Eingabe-Dimension-2-Index dem Ausgabe-Dimension-2-Index entspricht.

[0228] Operation „Get-dimension-2-vector“: Für ein spezifiziertes Ausgabeelement wird ein Dimension-2-Vektor aus dem Eingabe-Tensor 2 ausgewählt, wobei der Eingabe-Dimension-4-Index dem Ausgabe-Dimension-4-Index, der Eingabe-Dimension-3-Index dem Ausgabe-Dimension-3-Index und der Eingabe-Dimension-1-Index dem Ausgabe-Dimension-1-Index entspricht.

[0229] Punktproduktoperation: Das Zwischenpunktprodukt von zwei Vektoren derselben Größe und desselben Datentyps wird als Summe der Produkte der einzelnen Elemente im Eingabevektor 1 und des entsprechenden Elements des Eingabevektors 2 berechnet.

[0230] Fusionierte Operation: Der funktionsspezifische Parameter 1 steuert die Operation, die beim Zwischenpunktprodukt und dem entsprechenden Element des Eingabe-Tensors 3 durchgeführt wird. In einem Beispiel enthält ein funktionsspezifischer NNPA-MATMUL-Parameter 1 ein Operationsfeld z.B. in den Bits 24 bis 31. Das Operationsfeld spezifiziert die durchgeführte Operation. Beispielhafte Operationen sind nachstehend aufgeführt:

OPERATION	Operationstyp
0	Addition
1	Vergleichen, ob das Punktprodukt hoch ist
2	Vergleichen, ob das Punktprodukt niedrig ist
3	Vergleichen, ob das Punktprodukt und das Element gleich sind
4	Vergleichen, ob das Punktprodukt und das Element nicht gleich sind
5	Vergleichen, ob das Punktprodukt nicht hoch ist
6	Vergleichen, ob das Punktprodukt niedrig ist

[0231] In einem Beispiel wird bei einer Operation des Typs Addition das Element des Eingabe-Tensors 3 zum Zwischenpunktprodukt addiert. Bei Operationen des Typs Vergleich wird das Zwischenpunktprodukt mit dem Element des Eingabe-Tensors 3 verglichen, und wenn der Vergleich zutreffend ist, wird das Ergebnis auf einen Wert von z.B. +1 gesetzt; andernfalls wird es in dem für den Ausgabe-Tensor spezifizierten Datentyp auf einen Wert von z.B. +0 gesetzt.

[0232] In einem Beispiel sind alle anderen Werte des Feldes OPERATION reserviert. Wenn für das Feld OPERATION ein reservierter Wert spezifiziert ist, wird ein Antwortcode von z.B. F000 hex gemeldet, und die Operation wird mit einem Bedingungscode, z.B. 1, beendet.

[0233] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode, z.B. 0010 hex bzw. 0011 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0234] In einer Ausführungsform müssen die folgenden Bedingungen erfüllt sein, andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt:

* Die Dimension-4-Indexgröße von allen Eingabe-Tensoren und dem Ausgabe-Tensor muss gleich sein.

* Die Dimension-3-Indexgröße von allen Eingabe-Tensoren und dem Ausgabe-Tensor muss gleich eins sein.

* Die Dimension-2-Indexgröße des Eingabe-Tensors 3 muss gleich eins sein.

* Die Dimension-2-Indexgröße des Eingabe-Tensors 1 und des Ausgabe-Tensors muss gleich sein.

* Die Dimension-1-Indexgröße des Eingabe-Tensors 1 und die Dimension-2-Indexgröße des Eingabe-Tensors 2 müssen gleich sein.

* Die Dimension-1-Indexgröße von Eingabe-Tensor 2, Eingabe-Tensor 3 und Ausgabe-Tensor muss gleich sein.

* Das Datenlayout und der Datentyp von allen Eingabe-Tensoren und dem Ausgabe-Tensor müssen gleich sein.

[0235] In einer Ausführungsform werden der Ausgabe-Tensor-Deskriptor 2 und die funktionsspezifischen Sicherungsbereichs-Adressfelder ignoriert. Die funktionsspezifischen Parameter 2 bis 5 müssen in einem Beispiel Nullen enthalten.

Funktionscode 114: NNPA-MATMUL-OP-BCAST23 (Matrix-Multiplikationsoperation - Broadcast 23)

[0236] Wenn die NNPA-MATMUL-OP-BCAST23-Funktion spezifiziert ist, wird jedes Element in dem durch den Ausgabe-Tensor-Deskriptor beschriebenen Ausgabe-Tensor wie nachstehend in einem Beispiel beschrieben berechnet:

* Ein Dimension-1-Vektor wird aus dem durch den Eingabe-Tensor-Deskriptor 1 beschriebenen Eingabe-Tensor 1 unter Verwendung der nachstehend beschriebenen Operation „get-dimension-1-vector“ ausgewählt.

* Ein Dimension-2-Vektor wird aus dem durch den Eingabe-Tensor-Deskriptor 2 beschriebenen Eingabe-Tensor 2 unter Verwendung der nachstehend beschriebenen Operation „get-dimension-2-vector“ ausgewählt.

* Ein Zwischenpunktprodukt eines Dimension-1-Vektors und eines Dimension-2-Vektors wird unter Verwendung der nachstehend beschriebenen Punktproduktoperation berechnet.

* Das Element des durch den Eingabe-Tensor-Deskriptor 3 beschriebenen Eingabe-Tensors 3 mit demselben Dimensionsindex-1-Wert wie das Ausgabe-Tensor-Element wird zu dem zuvor berechneten Punktprodukt addiert und im Ausgabe-Tensor gespeichert.

[0237] Operation „Get-dimension-1-vector“: Für ein spezifiziertes Ausgabeelement wird ein Dimension-1-Vektor aus dem Eingabe-Tensor 1 ausgewählt, wobei der Eingabe-Dimension-4-Index dem Ausgabe-Dimension-4-Index, der Eingabe-Dimension-3-Index dem Ausgabe-Dimension-3-Index und der Eingabe-Dimension-2-Index dem Ausgabe-Dimension-2-Index entspricht.

[0238] Operation „Get-dimension-2-vector“: Für ein spezifiziertes Ausgabeelement wird ein Dimension-2-Vektor aus dem Eingabe-Tensor 2 ausgewählt, wobei der Eingabe-Dimension-4-Index eins ist, der Eingabe-Dimension-3-Index dem Ausgabe-Dimension-3-Index und der Eingabe-Dimension-1-Index dem Ausgabe-Dimension-1-Index entspricht.

[0239] Punktproduktoperation: Das Zwischenprodukt von zwei Vektoren derselben Größe und desselben Datentyps wird als Summe der Produkte der einzelnen Elemente im Eingabevektor 1 und des entsprechenden Elements des Eingabevektors 2 berechnet.

[0240] Wenn das spezifizierte Datenlayout in einem Beispiel in einem der spezifizierten Tensor-Deskriptoren keinen 4D-Merkmalstensor spezifiziert (z.B. Datenlayout = 0) oder wenn der Datentyp in einem der spezifizierten Tensor-Deskriptoren keinen NNP-Datentyp-1 spezifiziert (z.B. Datentyp = 0), wird ein Antwortcode, z.B. 0010 hex bzw. 0011 hex, im allgemeinen Register 0 gesetzt, und die Anweisung wird mit einem Bedingungscode, z.B. 1, beendet.

[0241] In einer Ausführungsform müssen die folgenden Bedingungen erfüllt sein, andernfalls wird eine allgemeine Operandendaten-Ausnahme erkannt:

* Die Dimension-4-Indexgröße des Eingabe-Tensors 1 und des Ausgabe-Tensors muss gleich sein.

* Die Dimension-4-Indexgröße des Eingabe-Tensors 2 und des Eingabe-Tensors 3 muss gleich eins sein.

* Die Dimension-3-Indexgröße von allen Eingabe-Tensoren und dem Ausgabe-Tensor muss gleich eins sein.

* Die Dimension-2-Indexgröße des Eingabe-Tensors 3 muss gleich eins sein.

* Die Dimension-2-Indexgröße des Eingabe-Tensors 1 und des Ausgabe-Tensors muss gleich sein.

* Die Dimension-1-Indexgröße von Eingabe-Tensor 1 und die Dimension-2-Indexgröße von Eingabe-Tensor 2 müssen gleich sein.

* Die Dimension-1-Indexgröße von Eingabe-Tensor 2, Eingabe-Tensor 3 und Ausgabe-Tensor muss gleich sein.

* Das Datenlayout und der Datentyp von allen Eingabe-Tensoren und dem Ausgabe-Tensor müssen gleich sein.

[0242] In einer Ausführungsform werden der Ausgabe-Tensor-Deskriptor 2 und die funktionsspezifischen Sicherungsbereichs-Adressfelder ignoriert. Die funktionsspezifischen Parameter 1 bis 5 müssen in einem Beispiel Nullen enthalten.

[0243] Wenn sich der Ausgabe-Tensor in einer Ausführungsform mit einem Eingabe-Tensor oder dem Parameterblock für die Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk überschneidet, sind Ergebnisse nicht vorhersehbar.

[0244] Eine Spezifikationsausnahme wird zum Beispiel erkannt, wenn versucht wird, die Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk auszuführen, und der Parameterblock z.B. nicht auf einer Doppelwortgrenze angegeben ist.

[0245] Eine allgemeine Operandendaten-Ausnahme wird erkannt, wenn versucht wird, die Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk auszuführen, und zum Beispiel Tensor-Deskriptor-Widersprüche vorliegen.

[0246] Zu den sich ergebenden Bedingungscodes für die Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk gehören zum Beispiel: 0 - Normaler Abschluss; 1 - Antwortcode ist gesetzt; 2 –; 3 - Von der CPU festgelegte Menge an verarbeiteten Daten.

[0247] In einer Ausführungsform umfasst die Ausführungspriorität für die Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk zum Beispiel:

1. bis 7.	Ausnahmen mit der gleichen Priorität wie die Priorität der Programmunterbrechungsbedingungen für den allgemeinen Fall.
8.A	Bedingungscode 1, da ein nicht zugewiesener oder nichtinstallierter Funktionscode spezifiziert ist.
8.B	Spezifikationsausnahme, da der Parameterblock nicht auf der Doppelwortgrenze angegeben ist.
9.	Zugriffsausnahmen für einen Zugriff auf den Parameterblock.
10.	Bedingungscode 1, da das spezifizierte Format des Parameterblocks vom Modell nicht unterstützt wird.
11.A	Bedingungscode 1, da die spezifizierten Tensor-Datenlayouts nicht unterstützt werden.
11.B	Allgemeine Operandendaten-Ausnahme, da es unterschiedliche Datenlayouts zwischen Tensor-Deskriptoren gibt.
12.A	Bedingungscode 1 aufgrund anderer als der in den Punkten 8.A, 10 und 11.A vorstehend und 12.B.1 nachstehend aufgeführten Bedingungen.
12.B.1	Bedingungscode 1 aufgrund eines ungültigen Ausgabe-Tensor-Datentyps für NNPA-RELU und NNPA-CONVOLUTION.
12.B.2	Allgemeine Operandendaten-Ausnahme für ungültigen Wert für den funktionsspezifischen NNPA-RELU-Parameter 1 und den funktionsspezifischen NNPA-CONVOLUTION-Parameter 4.
13.A	Zugriffsausnahmen für einen Zugriff auf den Ausgabe-Tensor.
13.B	Zugriffsausnahmen für einen Zugriff auf die Eingabe-Tensoren.
13.C	Zugriffsausnahmen für einen Zugriff auf den funktionsspezifischen Sicherungsbereich.
14.	Bedingungscode 0:

[0248] Wie hier beschrieben, ist eine einzelne Anweisung (z.B. die Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk) so konfiguriert, dass sie eine Mehrzahl von Funktionen einschließlich einer Abfragefunktion und eine Mehrzahl von Nichtabfragefunktionen durchführt. Jede Nichtabfragefunktion kann mit großen Datenmengen arbeiten, und eine Ausnahmeprüfung wird bereitgestellt. Während der Berechnungen können zum Beispiel nichtnumerische, nichtdarstellbare numerische Werte und/oder Werte außerhalb des gültigen Bereichs in den Berechnungen weitergegeben werden. Es ist nützlich zu wissen, wann ein ungültiger Wert, z.B. ein nichtnumerischer, nichtdarstellbarer Wert und/oder ein Wert außerhalb

des gültigen Bereichs, erzeugt wird, da entschieden werden kann, keine weiteren Berechnungen durchzuführen (z.B. die Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk nicht auszuführen, um eine andere Funktion aufzurufen). Das Erkennen ungültiger Werte sollte jedoch nicht zu Leistungseinbußen führen. Gemäß einem oder mehreren Aspekten der vorliegenden Erfindung wird somit beim Durchführen einer Berechnungsverarbeitung erkannt, ob ein Eingabe- oder Ausgabewert ein ungültiger Wert ist, z.B. ein nichtnumerischer, nichtdarstellbarer numerischer Wert und/oder ein Wert außerhalb des gültigen Bereichs. Am Ende der Berechnungsverarbeitung (z.B. nach dem Erkennen einer ungültigen Eingabe vor dem Durchführen einer Berechnung oder nach dem Durchführen einer oder mehrerer Berechnungen der aufgerufenen Funktion) wird ein Übersichtsanzeiger (z.B. ein Bereichsausnahme-Merker) gesetzt (z.B. auf eins). In einem Beispiel wird nicht angegeben, welches Datenelement den ungültigen Wert bereitgestellt hat. Dies ermöglicht eine höhere Leistung bei arithmetischen Operationen. Die Meldung, dass der ungültige Wert entweder eine Eingabe war oder durch eine Berechnung erzeugt wurde, ermöglicht eine einfachere Fehlerbehebung bei Modellen der künstlichen Intelligenz. Eine genaue Identifizierung des ungültigen Wertes ist nicht erforderlich, es ist jedoch nützlich zu wissen, welche Berechnungsfunktion einen dieser Werte erzeugt haben könnte.

[0249] Ein oder mehrere Aspekte der vorliegenden Erfindung sind untrennbar mit der Computertechnik verbunden und ermöglichen die Verarbeitung in einem Computer, wodurch dessen Leistung verbessert wird. Die Verwendung einer einzelnen entworfenen Maschinenanweisung, die so konfiguriert ist, dass sie verschiedene Funktionen durchführt, verbessert die Leistung in der Datenverarbeitungsumgebung, indem die Komplexität verringert wird, die Inanspruchnahme von Ressourcen verringert wird und die Verarbeitungsgeschwindigkeit erhöht wird. Durch das Durchführen einer Ausnahmeprüfung und -meldung, die eine Übersicht über die Ausnahme bereitstellen, wird die Leistung darüber hinaus nicht beeinträchtigt, und das Erzeugen von und/oder die Fehlerbehebung bei Modellen der künstlichen Intelligenz werden ermöglicht. Die Anweisung und/oder die Optimierungen in der Verarbeitung können auf vielen technischen Gebieten verwendet werden, z. B. in der Computerverarbeitung, der medizinischen Verarbeitung, dem Maschinenbau, der Automobiltechnik, der Fertigung usw. Durch das Bereitstellen von Optimierungen werden diese technischen Gebiete verbessert, indem z. B. Fehler und/oder die Ausführungszeit verringert und/oder die Leistung verbessert werden.

[0250] Weitere Einzelheiten einer Ausführungsform zum Ermöglichen der Verarbeitung in einer Datenverarbeitungsumgebung in Bezug auf einen oder mehrere Aspekte der vorliegenden Erfindung werden mit Bezug auf die **Fig.** 7A und 7B beschrieben.

[0251] Mit Bezug auf **Fig.** 7A wird in einem Beispiel eine Meldung, dass ein als ungültig ermittelter Wert in Eingabedaten für eine Berechnung einer oder mehrerer Berechnungen oder in Ausgabedaten enthalten ist, die sich aus der einen oder mehreren Berechnungen ergeben haben, abgerufen (700). Der Wert wird aufgrund einer Ausnahme einer Mehrzahl von Ausnahmen als ungültig ermittelt (702). Auf der Grundlage des Abrufens der Meldung, dass der Wert als ungültig ermittelt wurde, wird ein Übersichtsanzeiger gesetzt (704). Der Übersichtsanzeiger stellt die Mehrzahl der Ausnahmen insgesamt dar (706).

[0252] Ein Bereitstellen einer Meldung, dass ein ungültiger Wert in den Eingabe- oder Ausgabedaten erkannt wurde, ermöglicht die Fehlerbehebung z.B. bei Modellen der künstlichen Intelligenz. Eine höhere Leistung von arithmetischen Operationen wird bereitgestellt, indem nicht spezifiziert wird, welches bestimmte Datenelement den ungültigen Wert aufweist, oder nicht unterschieden wird, welche der Mehrzahl von Ausnahmen erkannt wurde.

[0253] In einem Beispiel stellt der Übersichtsanzeiger die Mehrzahl von Ausnahmen dar, ohne dass zwischen der Mehrzahl von Ausnahmen unterschieden wird (708). In einem Beispiel wird der Übersichtsanzeiger unabhängig davon gesetzt, welche Ausnahme der Mehrzahl von Ausnahmen verwendet wird, um zu ermitteln, ob der Wert ungültig ist (710).

[0254] Die Verwendung eines Anzeigers zum Anzeigen eines ungültigen Wertes unabhängig von der Art des ungültigen Wertes (z.B. nichtnumerisch, nichtdarstellbar, außerhalb des gültigen Bereichs) ermöglicht Codieren und Verarbeiten, verringert die Komplexität und erhöht die Systemleistung.

[0255] In einem Beispiel handelt es sich bei dem Übersichtsanzeiger um einen Bereichsverletzungsanzeiger eines Ausnahme-Merkers, der an einem Ort bereitgestellt wird, der durch eine Anweisung spezifiziert wird, die zum Durchführen der einen oder mehreren Berechnungen ausgegeben wird (712).

[0256] Zu der Mehrzahl von Ausnahmen gehören beispielsweise ein nichtnumerischer Wert, ein nichtdarstellbarer numerischer Wert und ein Wert außerhalb des gültigen Bereichs (714).

[0257] Mit Bezug auf **Fig.** 7B beruht das Abrufen der Meldung, dass der Wert als ungültig ermittelt wurde, auf dem Ausführen einer Anweisung, die die eine oder mehrere Berechnungen durchführt (720). Dies ermöglicht die Verarbeitung der Anweisung und das Bereitstellen eines ungültigen Wertes.

[0258] Die Anweisung ist beispielsweise so konfiguriert, dass sie eine Mehrzahl von Funktionen durchführt, die Berechnungen durchführen, und die Mehrzahl von Funktionen verwendet den Übersichtsanzeiger (722). Durch Vorliegen eines Anzeigers, der von der Mehrzahl von Funktionen genutzt wird, werden Komplexität, Codierungs- und Überprüfungsaufwand verringert.

[0259] In einem Beispiel handelt es sich bei der Anweisung um eine Anweisung in einem neuronalen Netzwerk, die Berechnungen bei Eingabe-Tensoren durchführt, um Ausgabe-Tensoren bereitzustellen, die für die Verarbeitung durch künstliche Intelligenz verwendet werden (724). Die Anweisung ist beispielsweise so konfiguriert, dass sie eine Mehrzahl von Funktionen durchführt, die Berechnungen durchführen, und die Mehrzahl von Funktionen verwendet den Übersichtsanzeiger (726).

[0260] Der Übersichtsanzeiger wird beispielsweise für eine bestimmte Anweisung definiert, und eine andere Anweisung verwendet einen anderen Übersichtsanzeiger (728).

[0261] In einem Beispiel wird ein Wert eines Bedingungscodes ermittelt (730), der auf der Grundlage des Ausführens der bestimmten Anweisung gesetzt wurde, und die Gültigkeit des Übersichtsanzeigers beruht auf dem Feststellen, dass der Wert des Bedingungscodes ein ausgewählter Wert ist (732).

Andere Varianten und Ausführungsformen sind möglich.

[0262] Aspekte der vorliegenden Erfindung können von vielen Arten von Datenverarbeitungsumgebungen verwendet werden. Ein weiteres Beispiel einer Datenverarbeitungsumgebung zum Integrieren und Verwenden eines oder mehrerer Aspekte der vorliegenden Erfindung wird mit Bezug auf **Fig.** 8A beschrieben. Die Datenverarbeitungsumgebung von **Fig.** 8A beruht zum Beispiel auf der Befehlssatzarchitektur z/Architecture®, die von der International Business Machines Corporation, Armonk, New York, angeboten wird. Die Befehlssatzarchitektur z/Architecture ist jedoch nur eine beispielhafte Architektur. Auch hier kann die Datenverarbeitungsumgebung auf anderen Architekturen beruhen, darunter auf den x86-Architekturen von Intel®, anderen Architekturen der International Business Machines Corporation und/oder Architekturen anderer Unternehmen, ohne auf diese beschränkt zu sein. Intel ist eine Marke oder eingetragene Marke der Intel Corporation oder ihrer Tochtergesellschaften in den Vereinigten Staaten und anderen Ländern.

[0263] In einem Beispiel umfasst eine Datenverarbeitungsumgebung 10 einen zentralen Elektronikkomplex (central electronics complex, CEC) 11. Der zentrale Elektronikkomplex 11 umfasst eine Mehrzahl von Komponenten, z.B. einen Speicher 12 (auch als Systemspeicher, Hauptarbeitsspeicher, Hauptspeicher, Zentralspeicher, Speicher bekannt), der mit einem oder mehreren Prozessoren (auch als Zentraleinheiten (CPUs) 13 bekannt) verbunden ist, und einen oder mehrere spezielle Prozessoren (z.B. den Prozessor 31 des neuronalen Netzwerks) und ein Eingabe/Ausgabe(E/A)-Teilsystem 14.

[0264] Der eine oder mehrere spezielle Prozessoren können beispielsweise von dem einen oder mehreren universellen Prozessoren getrennt sein, und/oder mindestens ein spezieller Prozessor kann in mindestens einen universellen Prozessor integriert sein. Andere Varianten sind ebenfalls möglich.

[0265] Das E/A-Teilsystem 14 kann Teil des zentralen Elektronikkomplexes oder davon getrennt sein. Es steuert den Informationsfluss zwischen dem Hauptspeicher 12 und den Eingabe-/Ausgabesteuereinheiten 15 und den Eingabe/Ausgabe(E/A)-Einheiten 16, die mit dem zentralen Elektronikkomplex verbunden sind.

[0266] Es können viele Arten von E/A-Einheiten verwendet werden. Bei einer bestimmten Art handelt es sich um eine Datenspeichereinheit 17. Die Datenspeichereinheit 17 kann ein oder mehrere Programme 18, eine oder mehrere durch einen Computer lesbare Programmanweisungen 19 und/oder Daten usw. speichern. Die durch einen Computer lesbaren Programmanweisungen können so konfiguriert sein, dass sie Funktionen von Ausführungsformen von Aspekten der Erfindung ausführen.

[0267] Der zentrale Elektronikkomplex 11 kann wechselbare/nichtwechselbare, flüchtige/nichtflüchtige Computersystem-Speichermedien enthalten und/oder mit diesen verbunden sein. Er kann nichtwechselbare, nichtflüchtige magnetische Medien (in der Regel als „Festplatte“ bezeichnet), ein Laufwerk für magnetische Speicherplatten zum Auslesen und Beschreiben einer wechselbaren, nichtflüchtigen magnetischen Speicherplatte (z.B. „Diskette“) und/oder ein Laufwerk für optische Speicherplatten zum Auslesen oder Beschreiben einer wechselbaren, nichtflüchtigen optischen Speicherplatte wie einer CD-ROM, DVD-ROM und andere optische Medien enthalten und/oder mit diesen verbunden sein. Es versteht sich, dass andere Hardware- und/oder Softwarekomponenten in Verbindung mit dem zentralen Elektronikkomplex 11 verwendet werden können. Zu Beispielen gehören, ohne auf diese beschränkt zu sein: Mikrocode oder Millicode, Einheitentreiber, redundante Verarbeitungseinheiten, Anordnungen externer Festplattenlaufwerke, RAID-Systeme, Bandlaufwerke und Speichersysteme für die Datenarchivierung usw.

[0268] Der zentrale Elektronikkomplex 11 kann darüber hinaus mit zahlreichen anderen universellen oder speziellen Datenverarbeitungssystem-Umgebungen oder -Konfigurationen funktionsfähig sein. Beispiele für bekannte Datenverarbeitungssysteme, -Umgebungen und/oder -Konfigurationen, die für die Nutzung mit dem zentralen Elektronikkomplex 11 geeignet sein können, sind unter anderem, ohne auf diese beschränkt zu sein, PersonalComputer-Systeme (PC-Systeme), Server-Computer-Systeme, schlanke Clients, leistungsintensive Clients, Hand- oder Laptop-Einheiten, Mehrprozessorsysteme, Systeme auf Grundlage von Mikroprozessoren, Beistellgeräte, programmierbare Unterhaltungselektronik, Netzwerk-PCs, Minicomputersysteme, Großrechnersysteme und verteilte Cloud-Computing-Umgebungen, die jedes beliebige der oben genannten Systeme oder Einheiten und Ähnliches enthalten.

[0269] Der zentrale Elektronikkomplex 11 stellt in einer oder mehreren Ausführungsformen eine logische Partitionierung und/oder eine Virtualisierungsunterstützung bereit. In einer Ausführungsform umfasst der Speicher 12 zum Beispiel wie in **Fig.** 8B gezeigt eine oder mehrere logische Partitionen 20, einen Hypervisor 21, der die logischen Partitionen verwaltet, und eine Prozessor-Firmware 22. Ein Beispiel für den Hypervisor 21 ist der Processor Resource/System Manager(PR/SM™)-Hypervisor, der von der International Business Machines Corporation, Armonk, New York, angeboten wird. PR/SM ist eine Marke oder eingetragene Marke der International Business Machines Corporation in mindestens einem Rechtsraum.

[0270] Jede logische Partition 20 kann als separates System funktionieren. Das heißt, jede logische Partition kann unabhängig zurückgesetzt werden, ein Gastbetriebssystem 23 wie das Betriebssystem z/OS® der International Business Machines Corporation, Armonk, New York, oder anderen Steuercode 24 wie den Coupling-Facility-Steuercode (CFCC) ausführen und mit verschiedenen Programmen 25 arbeiten. Ein Betriebssystem oder ein Anwendungsprogramm, das auf einer logischen Partition ausgeführt wird, hat scheinbar Zugriff auf ein vollständiges und komplettes System, wobei tatsächlich nur ein Teil davon verfügbar ist. Das Betriebssystem z/OS wird zwar nur als Beispiel vorgeschlagen, andere von der International Business Machines Corporation und/oder anderen Unternehmen angebotene Betriebssysteme können jedoch ebenfalls gemäß einem oder mehreren Aspekten der vorliegenden Erfindung verwendet werden.

[0271] Der Speicher 12 ist z.B. mit den CPUs 13 (**Fig.** 8A) verbunden, bei denen es sich um physische Prozessorressourcen handelt, die den logischen Partitionen zugeordnet werden können. Eine logische Partition 20 kann zum Beispiel einen oder mehrere logische Prozessoren enthalten, von denen jeder alle oder einen Teil einer physischen Prozessorressource 13 darstellt, die der logischen Partition dynamisch zugeordnet werden kann.

[0272] In noch einer weiteren Ausführungsform stellt der zentrale Elektronikkomplex Unterstützung für virtuelle Maschinen (entweder mit oder ohne Unterstützung für eine logische Partitionierung) bereit. Wie in **Fig.** 8C dargestellt, enthält der Speicher 12 des zentralen Elektronikkomplexes 11 zum Beispiel eine oder mehrere virtuelle Maschinen 26, einen Manager für virtuelle Maschinen wie einen Hypervisor 27, der die virtuellen Maschinen verwaltet, und Prozessor-Firmware 28. Ein Beispiel für den Hypervisor 27 ist der z/VM®-Hypervisor, der von der International Business Machines Corporation, Armonk, New York, angeboten wird. Der Hypervisor wird manchmal auch als Host bezeichnet. z/VM ist eine Marke oder eingetragene Marke der International Business Machines Corporation in mindestens einem Rechtsraum.

[0273] Die Unterstützung für virtuelle Maschinen des zentralen Elektronikkomplexes stellt die Möglichkeit bereit, eine große Anzahl von virtuellen Maschinen 26 zu betreiben, die jeweils mit unterschiedlichen Programmen 29 arbeiten und ein Gastbetriebssystem wie das Betriebssystem Linux® ausführen können. Jede virtuelle Maschine 26 kann als separates System funktionieren. Das heißt, jede virtuelle Maschine kann unabhängig zurückgesetzt werden, ein Gastbetriebssystem ausführen und mit unterschiedlichen Program-

men arbeiten. Ein Betriebssystem oder ein Anwendungsprogramm, das auf einer virtuellen Maschine ausgeführt wird, hat scheinbar Zugriff auf ein vollständiges und komplettes System, wobei tatsächlich nur ein Teil davon verfügbar ist. z/VM und Linux werden zwar nur als Beispiel vorgeschlagen, andere Manager von virtuellen Maschinen und/oder Betriebssysteme können jedoch ebenfalls gemäß einem oder mehreren Aspekten der vorliegenden Erfindung verwendet werden. Die eingetragene Marke Linux® wird im Rahmen einer Unterlizenz der Linux Foundation verwendet, der exklusiven Lizenznehmerin von Linus Torvalds, dem Eigentümer der Marke weltweit.

[0274] Eine weitere Ausführungsform einer Datenverarbeitungsumgebung zum Integrieren und Verwenden eines oder mehrerer Aspekte der vorliegenden Erfindung wird mit Bezug auf **Fig.** 9A beschrieben. In diesem Beispiel umfasst eine Datenverarbeitungsumgebung 36 zum Beispiel eine native Zentraleinheit (CPU) 37, einen Speicher 38 und eine oder mehrere Eingabe-/Ausgabeeinheiten und/oder Schnittstellen 39, die zum Beispiel über einen oder mehrere Busse 40 und/oder andere Verbindungen miteinander verbunden sind. Die Datenverarbeitungsumgebung 36 kann beispielsweise einen PowerPC®-Prozessor enthalten, der von der International Business Machines Corporation, Armonk, New York, angeboten wird; einen HP Superdome mit Itanium®-II-Prozessoren von Intel®, der von Hewlett Packard Co., Palo Alto, Kalifornien, angeboten wird; und/oder andere Maschinen auf der Grundlage von Architekturen, die von der International Business Machines Corporation, Hewlett Packard, Intel, Oracle und/oder anderen angeboten werden. PowerPC ist eine Marke oder eingetragene Marke der International Business Machines Corporation in mindestens einem Rechtsraum. Itanium ist eine Marke oder eingetragene Marke der Intel Corporation oder ihrer Tochtergesellschaften in den Vereinigten Staaten und anderen Ländern.

[0275] Die native Zentraleinheit 37 enthält ein oder mehrere native Register 41 wie beispielsweise ein oder mehrere universelle Register und/oder ein oder mehrere spezielle Register, die während der Verarbeitung in der Umgebung verwendet werden. Diese Register enthalten Informationen über den jeweiligen Zustand der Umgebung zu jedem beliebigen Zeitpunkt.

[0276] Die native Zentraleinheit 37 führt darüber hinaus Anweisungen und Code aus, die im Speicher 38 gespeichert sind. In einem bestimmten Beispiel führt die Zentraleinheit den Emulatorcode 42 aus, der im Speicher 38 gespeichert ist. Mit diesem Code kann die in einer Architektur konfigurierte Datenverarbeitungseinheit eine andere Architektur emulieren. Mit dem Emulatorcode 42 können beispielsweise Maschinen, die auf anderen Architekturen wie der Befehlssatzarchitektur z/Architecture, z.B. PowerPC-Prozessoren, HP-Superdome-Server oder andere, die Befehlssatzarchitektur z/Architecture emulieren und Software und Anweisungen ausführen, die auf der Grundlage der Befehlssatzarchitektur z/Architecture entwickelt wurden.

[0277] Weitere Einzelheiten im Zusammenhang mit dem Emulatorcode 42 sind mit Bezug auf **Fig.** 9B beschrieben. Die im Speicher 38 gespeicherten Gastanweisungen 43 weisen Software-Anweisungen auf (z.B. in Verbindung mit Maschinenanweisungen), die für die Ausführung in einer anderen Architektur als derjenigen der nativen CPU 37 entwickelt wurden. Die Gastanweisungen 43 können zum Beispiel entwickelt worden sein, um auf einem Prozessor auf der Grundlage der Befehlssatzarchitektur z/Architecture ausgeführt zu werden, jedoch werden sie stattdessen auf der nativen CPU 37 emuliert, bei der es sich beispielsweise um einen Itanium-II-Prozessor von Intel handeln kann. In einem Beispiel enthält der Emulatorcode 42 eine Anweisungsabrufroutine 44, um eine oder mehrere Gastanweisungen 43 aus dem Speicher 38 abzurufen und für die abgerufenen Anweisungen optional eine lokale Pufferung bereitzustellen. Der Code enthält auch eine Anweisungsübersetzungsroutine 45, um die Art der abgerufenen Gastanweisung zu ermitteln und die Gastanweisung in eine oder mehrere entsprechende native Anweisungen 46 zu übersetzen. Diese Übersetzung umfasst zum Beispiel, dass die von der Gastanweisung durchzuführende Funktion identifiziert wird und die native(n) Anweisung(en), die diese Funktion durchführt bzw. durchführen, ausgewählt wird bzw. werden.

[0278] Der Emulatorcode 42 umfasst des Weiteren eine Emulationssteuerungsroutine 47, um zu bewirken, dass die nativen Anweisungen ausgeführt werden. Die Emulationssteuerungsroutine 47 kann bewirken, dass die native CPU 37 eine Routine mit nativen Anweisungen ausführt, die eine oder mehrere zuvor abgerufene Gastanweisungen emulieren, und dass die Steuerung nach der erfolgten Ausführung an die Anweisungsabrufroutine zurückgegeben wird, um das Abrufen der nächsten Gastanweisung oder einer Gruppe von Gastanweisungen zu emulieren. Das Ausführen der nativen Anweisungen 46 kann ein Laden von Daten aus dem Speicher 38 in ein Register; ein Zurückspeichern von Daten aus einem Register in einen Speicher; oder ein Durchführen eines Typs einer arithmetischen oder logischen Operation wie durch die Übersetzungsroutine festgelegt umfassen.

[0279] Jede Routine ist zum Beispiel in Software implementiert, die im Speicher gespeichert ist und von der nativen Zentraleinheit 37 ausgeführt wird. In anderen Beispielen können eine oder mehrere Routinen oder Operationen in Firmware, Hardware, Software oder Kombinationen davon implementiert sein. Die Register des emulierten Prozessors können unter Verwendung der Register 41 der nativen CPU oder unter Verwendung von Speicherorten im Speicher 38 emuliert werden. In Ausführungsformen können sich die Gastanweisungen 43, die nativen Anweisungen 46 und der Emulatorcode 42 im selben Speicher befinden oder auf verschiedene Speichereinheiten verteilt sein.

[0280] Eine Anweisung, die emuliert werden kann, enthält die hier beschriebene Anweisung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk gemäß einem oder mehreren Aspekten der vorliegenden Erfindung. Darüber hinaus können andere Anweisungen, ein oder mehrere Aspekte der Verarbeitung in einem neuronalen Netzwerk und/oder die Ausnahmeprüfung und -meldung gemäß einem oder mehreren Aspekten der vorliegenden Erfindung emuliert werden.

[0281] Die vorstehend beschriebenen Datenverarbeitungsumgebungen sind lediglich Beispiele für Datenverarbeitungsumgebungen, die verwendet werden können. Andere Umgebungen, darunter nichtpartitionierte Umgebungen, partitionierte Umgebungen, Cloud-Umgebungen und/oder emulierte Umgebungen, können verwendet werden, ohne auf diese beschränkt zu sein; Ausführungsformen sind nicht auf eine bestimmte Ausführungsform beschränkt. Hier sind zwar verschiedene Beispiele von Datenverarbeitungsumgebungen beschrieben, jedoch können ein oder mehrere Aspekte der vorliegenden Erfindung mit vielen Arten von Umgebungen verwendet werden. Die hier bereitgestellten Datenverarbeitungsumgebungen sind lediglich Beispiele.

[0282] Jede Datenverarbeitungsumgebung kann so konfiguriert sein, dass sie einen oder mehrere Aspekte der vorliegenden Erfindung enthält.

[0283] Ein oder mehrere Aspekte können sich auf Cloud-Computing beziehen.

[0284] Die vorliegende Offenbarung enthält zwar eine ausführliche Beschreibung von Cloud-Computing, es versteht sich jedoch, dass die Umsetzung der hier dargelegten Lehren nicht auf eine Cloud-Computing-Umgebung beschränkt ist. Stattdessen können Ausführungsformen der vorliegenden Erfindung gemeinsam mit beliebigen Arten von jetzt bekannter oder später entwickelter Datenverarbeitungsumgebung umgesetzt werden.

[0285] Cloud-Computing ist ein Modell zum Liefern eines Dienstes, der einen problemlosen, bedarfsorientierten Netzwerkzugang zu einem gemeinsamen Pool an konfigurierbaren Datenverarbeitungsressourcen (z.B. Netzwerke, Netzwerkbandbreite, Server, Verarbeitung, Speicher, Anwendungen, virtuelle Maschinen und Dienste) ermöglicht, die rasch bereitgestellt und mit minimalem Verwaltungsaufwand bzw. minimaler Interaktion mit einem Anbieter eines Dienstes freigegeben werden können. Dieses Cloud-Modell kann mindestens fünf Eigenschaften, mindestens drei Dienstmodelle und mindestens vier Implementierungsmodelle enthalten.

[0286] Bei den Eigenschaften handelt es sich um die Folgenden:

> On-Demand Self-Service (bedarfsorientierte Selbstbedienung): Ein Cloud-Nutzer kann einseitig automatisch nach Bedarf Datenverarbeitungsfunktionen wie Serverzeit und Netzwerkspeicher bereitstellen, ohne dass eine menschliche Interaktion mit dem Anbieter der Dienste erforderlich ist.

[0287] Broad Network Access (breiter Netzzugriff): Über ein Netzwerk sind Funktionen verfügbar, auf die durch Standardmechanismen zugegriffen wird, die die Verwendung durch heterogene schlanke oder leistungsintensive Client-Plattformen unterstützen (z.B. Mobiltelefone, Laptops und PDAs).

[0288] Ressource Pooling (Ressourcen-Bündelung): Die Datenverarbeitungsressourcen des Anbieters werden gebündelt, um mehreren Nutzern unter Verwendung eines Mehrmietermodells zu dienen, wobei verschiedene physische und virtuelle Ressourcen dynamisch nach Bedarf zugewiesen und neu zugewiesen werden. Es gibt eine gefühlte Standortunabhängigkeit, da der Nutzer allgemein keine Kontrolle bzw. Kenntnis über den genauen Standort der bereitgestellten Ressourcen hat, aber in der Lage sein kann, einen Standort auf einer höheren Abstraktionsebene festzulegen (z.B. Land, Staat oder Rechenzentrum).

[0289] Rapid Elasticity (schnelle Anpassungsfähigkeit): Funktionen können für eine schnelle horizontale Skalierung (scale out) schnell und elastisch bereitgestellt werden, in einigen Fällen auch automatisch, und

für ein schnelles Scale-in schnell freigegeben werden. Für den Nutzer erscheinen die für das Bereitstellen verfügbaren Funktionen häufig unbegrenzt und sie können jederzeit in jeder beliebigen Menge gekauft werden.

[0290] Measured Service (messbarer Dienst): Cloud-Systeme steuern und optimieren die Verwendung von Ressourcen automatisch, indem sie eine Messfunktion auf einer gewissen Abstraktionsebene nutzen, die für die Art von Dienst geeignet ist (z.B. Speicher, Verarbeitung, Bandbreite sowie aktive Benutzerkonten). Die Inanspruchnahme von Ressourcen kann überwacht, gesteuert und gemeldet werden, wodurch sowohl für den Anbieter als auch für den Nutzer des verwendeten Dienstes Transparenz bereitgestellt wird.

[0291] Es gibt folgende Dienstmodelle:

Software as a Service (Saas) (Software als Dienst): Die dem Nutzer bereitgestellte Funktion besteht darin, die in einer Cloud-Infrastruktur laufenden Anwendungen des Anbieters zu verwenden. Die Anwendungen sind über eine schlanke Client-Schnittstelle wie einen Web-Browser (z.B. auf dem Web beruhende eMail) von verschiedenen Client-Einheiten aus zugänglich. Der Nutzer verwaltet bzw. steuert die zugrunde liegende Cloud-Infrastruktur nicht, darunter das Netzwerk, Server, Betriebssysteme, Speicher bzw. sogar einzelne Anwendungsfunktionen, mit der möglichen Ausnahme von eingeschränkten benutzerspezifischen Einstellungen der Anwendungskonfiguration.

[0292] Platform as a Service (Paas) (Plattform als Dienst): Die dem Nutzer bereitgestellte Funktion besteht darin, durch einen Nutzer erstellte bzw. erhaltene Anwendungen, die unter Verwendung von durch den Anbieter unterstützten Programmiersprachen und Werkzeugen erstellt wurden, in der Cloud-Infrastruktur einzusetzen. Der Nutzer verwaltet bzw. steuert die zugrunde liegende Cloud-Infrastruktur nicht, darunter Netzwerke, Server, Betriebssysteme bzw. Speicher, hat aber die Kontrolle über die eingesetzten Anwendungen und möglicherweise über Konfigurationen der Hosting-Umgebung der Anwendung.

[0293] Infrastructure as a Service (laas) (Infrastruktur als Dienst): Die dem Nutzer bereitgestellte Funktion besteht darin, Verarbeiten, Speicher, Netzwerke und andere grundlegende Datenverarbeitungsressourcen bereitzustellen, wobei der Nutzer in der Lage ist, beliebige Software einzusetzen und auszuführen, zu der Betriebssysteme und Anwendungen gehören können. Der Nutzer verwaltet bzw. steuert die zugrunde liegende Cloud-Infrastruktur nicht, hat aber die Kontrolle über Betriebssysteme, Speicher, eingesetzte Anwendungen und möglicherweise eine eingeschränkte Kontrolle über ausgewählte Netzwerkkomponenten (z.B. Host-Firewalls).

[0294] Es gibt folgende Einsatzmodelle:

Private Cloud: Die Cloud-Infrastruktur wird einzig und allein für eine Organisation betrieben. Sie kann durch die Organisation oder einen Dritten verwaltet werden und kann sich in den eigenen Räumen oder in fremden Räumen befinden.

[0295] Community Cloud (Benutzergemeinschafts-Cloud): Die Cloud-Infrastruktur wird von mehreren Organisationen gemeinsam genutzt und unterstützt eine spezielle Benutzergemeinschaft, die gemeinsame Anliegen hat (z.B. Aufgabe, Sicherheitsanforderungen, Richtlinien sowie Überlegungen bezüglich der Einhaltung von Vorschriften). Sie kann durch die Organisationen oder einen Dritten verwaltet werden und kann sich in den eigenen Räumen oder fremden Räumen befinden.

[0296] Public Cloud (öffentliche Cloud): Die Cloud-Infrastruktur wird der allgemeinen Öffentlichkeit oder einer großen Branchengruppe zur Verfügung gestellt und gehört einer Organisation, die Cloud-Dienste verkauft.

[0297] Hybrid Cloud (hybride Cloud): Die Cloud-Infrastruktur besteht aus zwei oder mehr Clouds (privat, Benutzergemeinschaft oder öffentlich), die zwar einzelne Entitäten bleiben, aber durch eine standardisierte oder herstellereigene Technologie miteinander verbunden sind, die eine Übertragbarkeit von Daten und Anwendungen ermöglicht (z.B. Cloud-Zielgruppenverteilung für den Lastenausgleich zwischen Clouds).

[0298] Eine Cloud-Computing-Umgebung ist dienstorientiert und schwerpunktmäßig auf Statusunabhängigkeit, geringe Kopplung, Modularität und semantische Interoperabilität ausgerichtet. Der Kern der Cloud-Computing ist eine Infrastruktur, die ein Netzwerk aus miteinander verbundenen Knoten enthält.

[0299] Mit Bezug nunmehr auf **Fig.** 10 ist eine veranschaulichende Cloud-Computing-Umgebung 50 dargestellt. Wie gezeigt, enthält die Cloud-Computing-Umgebung 50 einen oder mehrere Cloud-Computing-Knoten

52, mit denen von Cloud-Nutzern verwendete lokale Datenverarbeitungseinheiten wie der persönliche digitale Assistent (PDA) oder das Mobiltelefon 54A, der Desktop-Computer 54B, der Laptop-Computer 54C und/oder das Kraftfahrzeug-Computersystem 54N Daten austauschen können. Die Knoten 52 können miteinander Daten austauschen. Sie können physisch oder virtuell in einem oder mehreren Netzwerken wie private, benutzergemeinschaftliche, öffentliche oder hybride Clouds wie oben beschrieben oder in einer Kombination davon in Gruppen angeordnet sein (nicht dargestellt). Dies ermöglicht es der Cloud-Computing-Umgebung 50, Infrastruktur, Plattformen und/oder Software als Dienste anzubieten, für die ein Cloud-Nutzer keine Ressourcen auf einer lokalen Datenverarbeitungseinheit vorhalten muss. Es versteht sich, dass die in **Fig.** 10 gezeigten Arten von Datenverarbeitungseinheiten 54A bis N nur veranschaulichend sein sollen und die Datenverarbeitungsknoten 52 und die Cloud-Computing-Umgebung 50 mit jeder Art von computergestützter Einheit über jede Art von Netzwerk und/oder netzwerkadressierbarer Verbindung Daten austauschen kann (z.B. über einen Web-Browser).

[0300] Mit Bezug nunmehr auf **Fig.** 11 wird ein Satz funktionaler Abstraktionsschichten gezeigt, die von der Cloud-Computing-Umgebung 50 (**Fig.** 10) bereitgestellt werden. Es versteht sich von vornherein, dass die in **Fig.** 11 dargestellten Komponenten, Schichten und Funktionen nur veranschaulichend sein sollen und Ausführungsformen der Erfindung nicht darauf beschränkt sind. Wie dargestellt, werden die folgenden Schichten und entsprechenden Funktionen bereitgestellt:

Die Hardware- und Software-Schicht 60 enthält Hardware- und Software-Komponenten. Zu Beispielen für Hardware-Komponenten gehören: die Großrechner 61; die Server 62 auf Grundlage der RISC-Architektur (RISC = Reduced Instruction Set Computer, Computer mit reduziertem Befehlssatz), die Server 63; die Blade-Server 64; die Speichereinheiten 65; sowie die Netzwerke und Netzwerkkomponenten 66. In einigen Ausführungsformen enthalten die Software-Komponenten die Netzwerkanwendungs-Serversoftware 67 und die Datenbank-Software 68.

[0301] Die Virtualisierungsschicht 70 stellt eine Abstraktionsschicht bereit, aus der die folgenden Beispiele für virtuelle Entitäten bereitgestellt werden können: virtuelle Server 71; virtuelle Speicher 72; virtuelle Netzwerke 73; darunter virtuelle private Netzwerke; virtuelle Anwendungen und Betriebssysteme 74; und virtuelle Clients 75.

[0302] In einem Beispiel kann die Verwaltungsschicht 80 die nachfolgend beschriebenen Funktionen bereitstellen. Die Ressourcenbereitstellung 81 ermöglicht eine dynamische Bereitstellung von Datenverarbeitungsressourcen und anderen Ressourcen, die verwendet werden, um Aufgaben in der Cloud-Computing-Umgebung durchzuführen. Messen und Preisfindung 82 stellen Kostenverfolgung beim Verwenden von Ressourcen in der Cloud-Computing-Umgebung sowie Abrechnung oder Rechnungsstellung für die Inanspruchnahme dieser Ressourcen bereit. In einem Beispiel können diese Ressourcen Lizenzen für Anwendungssoftware umfassen. Die Sicherheitsfunktion stellt eine Identitätsprüfung für Cloud-Nutzer und Aufgaben sowie Schutz für Daten und andere Ressourcen bereit. Ein Benutzerportal 83 stellt Nutzern und Systemadministratoren den Zugang zur Cloud-Computing-Umgebung bereit. Die Verwaltung der Dienstgüte 84 stellt Zuordnung und Verwaltung von Cloud-Computing-Ressourcen bereit, sodass die erforderliche Dienstgüte erreicht wird. Die Planung und Erfüllung der Dienstgütevereinbarung (Service Level Agreement, SLA) 85 stellt eine Vorabeinteilung und eine Beschaffung von Cloud-Computing-Ressourcen bereit, deren künftiger Bedarf auf der Grundlage einer Dienstgütevereinbarung vorausgesehen wird.

[0303] Die Arbeitslastschicht 90 stellt Beispiele für Funktionalitäten bereit, für die die Cloud-Computing-Umgebung verwendet werden kann. Beispiele von Arbeitslasten und Funktionen, die von dieser Schicht bereitgestellt werden können, beinhalten: Abbildung und Navigation 91; Software-Entwicklung und Lebenszyklusverwaltung 92; Lieferung von Bildung aus dem virtuellen Klassenzimmer 93; Datenanalyseverarbeitung 94; Transaktionsverarbeitung 95; und Verarbeitung zum Unterstützen der Verarbeitung in einem neuronalen Netzwerk und/oder Verarbeitung zur Ausnahmeprüfung/-meldung 96.

[0304] Bei Aspekten der vorliegenden Erfindung kann es sich um ein System, ein Verfahren und/oder ein Computerprogrammprodukt auf jedem möglichen technischen Detaillierungsgrad der Integration handeln. Das Computerprogrammprodukt kann (ein) durch einen Computer lesbare(s) Speichermedium (oder -medien) umfassen, auf dem/denen durch einen Computer lesbare Programmanweisungen gespeichert ist/-sind, um einen Prozessor dazu zu veranlassen, Aspekte der vorliegenden Erfindung auszuführen.

[0305] Bei dem durch einen Computer lesbaren Speichermedium kann es sich um eine physische Einheit handeln, die Anweisungen zur Verwendung durch eine Einheit zur Ausführung von Anweisungen behalten und speichern kann. Bei dem durch einen Computer lesbaren Speichermedium kann es sich zum Beispiel

um eine elektronische Speichereinheit, eine magnetische Speichereinheit, eine optische Speichereinheit, eine elektromagnetische Speichereinheit, eine Halbleiterspeichereinheit oder jede geeignete Kombination daraus handeln, ohne auf diese beschränkt zu sein. Zu einer nicht erschöpfenden Liste spezifischerer Beispiele des durch einen Computer lesbaren Speichermediums gehören die Folgenden: eine tragbare Computerdiskette, eine Festplatte, ein Direktzugriffsspeicher (RAM), ein Nur-Lese-Speicher (ROM), ein löschbarer programmierbarer Nur-Lese-Speicher (EPROM bzw. Flash-Speicher), ein statischer Direktzugriffsspeicher (SRAM), ein tragbarer Kompaktspeicherplatte-Nur-Lese-Speicher (CD-ROM), eine DVD (digital versatile disc), ein Speicher-Stick, eine Diskette, eine mechanisch codierte Einheit wie zum Beispiel Lochkarten oder gehobene Strukturen in einer Rille, auf denen Anweisungen gespeichert sind, und jede geeignete Kombination daraus. Ein durch einen Computer lesbares Speichermedium soll in der Verwendung hier nicht als flüchtige Signale an sich aufgefasst werden, wie zum Beispiel Funkwellen oder andere sich frei ausbreitende elektromagnetische Wellen, elektromagnetische Wellen, die sich durch einen Wellenleiter oder ein anderes Übertragungsmedium ausbreiten (z.B. Lichtwellenleiterkabel durchlaufende Lichtimpulse) oder durch einen Draht übertragene elektrische Signale.

[0306] Hier beschriebene, durch einen Computer lesbare Programmanweisungen können von einem durch einen Computer lesbaren Speichermedium auf jeweilige Datenverarbeitungs-/Verarbeitungseinheiten oder über ein Netzwerk wie zum Beispiel das Internet, ein lokales Netzwerk, ein Weitverkehrsnetzwerk und/oder ein drahtloses Netzwerk auf einen externen Computer oder eine externe Speichereinheit heruntergeladen werden. Das Netzwerk kann Kupferübertragungskabel, Lichtwellenübertragungsleiter, drahtlose Übertragung, Router, Firewalls, Vermittlungseinheiten, Gateway-Computer und/oder Edge-Server aufweisen. Eine Netzwerkadapterkarte oder Netzwerkschnittstelle in jeder Datenverarbeitungs-/Verarbeitungseinheit empfängt durch einen Computer lesbare Programmanweisungen aus dem Netzwerk und leitet die durch einen Computer lesbaren Programmanweisungen zur Speicherung in einem durch einen Computer lesbaren Speichermedium innerhalb der entsprechenden Datenverarbeitungs-/Verarbeitungseinheit weiter.

[0307] Bei durch einen Computer lesbaren Programmanweisungen zum Ausführen von Arbeitsschritten der vorliegenden Erfindung kann es sich um Assembler-Anweisungen, ISA-Anweisungen (Instruction-Set-Architecture), Maschinenanweisungen, maschinenabhängige Anweisungen, Mikrocode, Firmware-Anweisungen, zustandssetzende Daten, Konfigurationsdaten für integrierte Schaltungen oder entweder Quellcode oder Objektcode handeln, die in einer beliebigen Kombination aus einer oder mehreren Programmiersprachen geschrieben werden, darunter objektorientierte Programmiersprachen wie Smalltalk, C++ o.ä. sowie prozedurale Programmiersprachen wie die Programmiersprache „C“ oder ähnliche Programmiersprachen. Die durch einen Computer lesbaren Programmanweisungen können vollständig auf dem Computer des Benutzers, teilweise auf dem Computer des Benutzers, als eigenständiges Software-Paket, teilweise auf dem Computer des Benutzers und teilweise auf einem entfernt angeordneten Computer oder vollständig auf dem entfernt angeordneten Computer oder Server ausgeführt werden. In letzterem Fall kann der entfernt angeordnete Computer mit dem Computer des Benutzers durch eine beliebige Art Netzwerk verbunden sein, darunter ein lokales Netzwerk (LAN) oder ein Weitverkehrsnetzwerk (WAN), oder die Verbindung kann mit einem externen Computer hergestellt werden (zum Beispiel über das Internet unter Verwendung eines Internet-Dienstanbieters). In einigen Ausführungsformen können elektronische Schaltungen, darunter zum Beispiel programmierbare Logikschaltungen, vor Ort programmierbare Gatter-Anordnungen (FPGA, field programmable gate arrays) oder programmierbare Logikanordnungen (PLA, programmable logic arrays) die durch einen Computer lesbaren Programmanweisungen ausführen, indem sie Zustandsinformationen der durch einen Computer lesbaren Programmanweisungen nutzen, um die elektronischen Schaltungen zu personalisieren, um Aspekte der vorliegenden Erfindung durchzuführen.

[0308] Aspekte der vorliegenden Erfindung sind hier unter Bezugnahme auf Ablaufpläne und/oder Blockschaubilder von Verfahren, Vorrichtungen (Systemen) und Computerprogrammprodukten gemäß Ausführungsformen der Erfindung beschrieben. Es wird darauf hingewiesen, dass jeder Block der Ablaufpläne und/oder der Blockschaubilder sowie Kombinationen von Blöcken in den Ablaufplänen und/oder den Blockschaubildern mittels durch einen Computer lesbare Programmanweisungen ausgeführt werden können.

[0309] Diese durch einen Computer lesbaren Programmanweisungen können einem Prozessor eines Computers oder einer anderen programmierbaren Datenverarbeitungsvorrichtung bereitgestellt werden, um eine Maschine zu erzeugen, sodass die über den Prozessor des Computers bzw. der anderen programmierbaren Datenverarbeitungsvorrichtung ausgeführten Anweisungen ein Mittel zur Umsetzung der in dem Block bzw. den Blöcken der Ablaufpläne und/oder der Blockschaubilder festgelegten Funktionen/Schritte erzeugen. Diese durch einen Computer lesbaren Programmanweisungen können auch auf einem durch einen Computer lesbaren Speichermedium gespeichert sein, das einen Computer, eine programmierbare Datenverarbei-

tungsvorrichtung und/oder andere Einheiten so steuern kann, dass sie auf eine bestimmte Art funktionieren, sodass das durch einen Computer lesbare Speichermedium, auf dem Anweisungen gespeichert sind, einen Herstellungsartikel aufweist, darunter Anweisungen, welche Aspekte der/des in dem Block bzw. den Blöcken des Ablaufplans und/oder der Blockschaubilder angegebenen Funktion/Schritts umsetzen.

[0310] Die durch einen Computer lesbaren Programmanweisungen können auch auf einen Computer, eine andere programmierbare Datenverarbeitungsvorrichtung oder eine andere Einheit geladen werden, um das Ausführen einer Reihe von Prozessschritten auf dem Computer bzw. der anderen programmierbaren Vorrichtung oder anderen Einheit zu verursachen, um einen auf einem Computer ausgeführten Prozess zu erzeugen, sodass die auf dem Computer, einer anderen programmierbaren Vorrichtung oder einer anderen Einheit ausgeführten Anweisungen die in dem Block bzw. den Blöcken der Ablaufpläne und/oder der Blockschaubilder festgelegten Funktionen/Schritte umsetzen.

[0311] Die Ablaufpläne und die Blockschaubilder in den Figuren veranschaulichen die Architektur, die Funktionalität und den Betrieb möglicher Ausführungen von Systemen, Verfahren und Computerprogrammprodukten gemäß verschiedenen Ausführungsformen der vorliegenden Erfindung. In diesem Zusammenhang kann jeder Block in den Ablaufplänen oder Blockschaubildern ein Modul, ein Segment oder einen Teil von Anweisungen darstellen, die eine oder mehrere ausführbare Anweisungen zur Ausführung der bestimmten logischen Funktion(en) aufweisen. In einigen alternativen Ausführungen können die in dem Block angegebenen Funktionen in einer anderen Reihenfolge als in den Figuren gezeigt stattfinden. Zwei nacheinander gezeigte Blöcke können zum Beispiel in Wirklichkeit als ein Schritt ausgeführt, gleichzeitig ausgeführt, im Wesentlich gleichzeitig ausgeführt, ganz oder teilweise zeitlich überlappend ausgeführt werden, oder die Blöcke können manchmal je nach entsprechender Funktionalität in umgekehrter Reihenfolge ausgeführt werden. Es ist ferner anzumerken, dass jeder Block der Blockschaubilder und/oder der Ablaufpläne sowie Kombinationen aus Blöcken in den Blockschaubildern und/oder den Ablaufplänen durch spezielle auf Hardware beruhende Systeme umgesetzt werden können, welche die festgelegten Funktionen oder Schritte durchführen, oder Kombinationen aus Spezial-Hardware und Computeranweisungen ausführen.

[0312] Zusätzlich zu dem oben Genannten können ein oder mehrere Aspekte durch einen Diensteanbieter, der eine Verwaltung von Kundenumgebungen anbietet, bereitgestellt, angeboten, eingesetzt, verwaltet, gepflegt usw. werden. Der Diensteanbieter kann zum Beispiel einen Computercode und/oder eine Computerinfrastruktur erstellen, verwalten, unterstützen usw., die einen oder mehrere Aspekte für einen oder mehrere Kunden ausführt. Im Gegenzug kann der Diensteanbieter eine Vergütung vom Kunden beispielsweise im Rahmen eines Abonnementvertrags und/oder eine Honorarvereinbarung erhalten. Zusätzlich oder alternativ kann der Diensteanbieter eine Vergütung aus dem Verkauf von Werbeinhalten an einen oder mehrere Dritte erhalten.

[0313] Bei einem Aspekt kann eine Anwendung eingesetzt werden, um eine oder mehrere Ausführungsformen auszuführen. Beispielsweise weist das Einsetzen einer Anwendung ein Bereitstellen einer Computerinfrastruktur auf, die in der Lage ist, eine oder mehrere Ausführungsformen auszuführen.

[0314] Als weiterer Aspekt kann eine Datenverarbeitungsinfrastruktur eingesetzt werden, die ein Integrieren eines durch einen Computer lesbaren Codes in ein Datenverarbeitungssystem aufweist, wobei der Code zusammen mit dem Datenverarbeitungssystem in der Lage ist, eine oder mehrere Ausführungsformen auszuführen.

[0315] Als noch weiterer Aspekt kann ein Prozess zum Integrieren einer Datenverarbeitungsinfrastruktur bereitgestellt werden, der ein Integrieren eines durch einen Computer lesbaren Codes in ein Computersystem aufweist. Das Computersystem weist ein durch einen Computer lesbares Medium auf, wobei das Computermedium eine oder mehrere Ausführungsformen aufweist. Zusammen mit dem Computersystem ist der Code in der Lage, eine oder mehrere Ausführungsformen auszuführen.

[0316] Vorstehend sind zwar verschiedene Ausführungsformen beschrieben, dabei handelt es sich jedoch lediglich um Beispiele. Datenverarbeitungsumgebungen von anderen Architekturen können zum Beispiel verwendet werden, um einen oder mehrere Aspekte zu integrieren und/oder zu verwenden. Darüber hinaus können verschiedene Befehle oder Operationen verwendet werden. Zusätzlich können verschiedene Arten von Registern und/oder verschiedene Register verwendet werden. Darüber hinaus können andere Datenformate, Datenlayouts und/oder Datengrößen unterstützt werden. In einer oder mehreren Ausführungsformen können ein oder mehrere universelle Prozessoren, ein oder mehrere spezielle Prozessoren oder eine Kombination von universellen und speziellen Prozessoren verwendet werden. Zahlreiche Varianten sind möglich.

[0317] Hier werden verschiedene Aspekte beschrieben. Darüber hinaus sind viele Varianten möglich, ohne von einem Grundgedanken der Aspekte der vorliegenden Erfindung abzuweichen. Es sei darauf hingewiesen, dass, sofern nicht anders angegeben, jeder hier beschriebene Aspekt oder jedes hier beschriebene Merkmal und Varianten davon mit jedem beliebigen Aspekt oder Merkmal kombiniert werden können.

[0318] Andere Arten von Datenverarbeitungsumgebungen können Nutzen daraus ziehen und verwendet werden. Als Beispiel kann ein zum Speichern und/oder Ausführen von Programmcode geeignetes Datenverarbeitungssystem verwendet werden, das mindestens zwei Prozessoren enthält, die direkt oder indirekt über einen Systembus mit Speicherelementen verbunden sind. Die Speicherelemente enthalten zum Beispiel einen lokalen Speicher, der während der eigentlichen Ausführung des Programmcodes eingesetzt wird, einen Massenspeicher und einen Zwischenspeicher, die eine zeitweilige Speicherung von mindestens einem Teil des Programmcodes bereitstellen, um die Häufigkeit zu verringern, mit der der Code während der Ausführung aus dem Massenspeicher abgerufen werden muss.

[0319] Eingabe/Ausgabe- oder E/A-Einheiten (darunter Tastaturen, Anzeigen, Zeigeeinheiten, Direktzugriffsspeichereinheiten (direct access storage devices, DASD), Magnetbänder, CDs, DVDs, USB-Speichersticks und sonstige Speichermedien usw., ohne auf diese beschränkt zu sein) können entweder direkt oder indirekt über dazwischengeschaltete E/A-Steuereinheiten mit dem System verbunden sein. Netzwerkadapter können ebenfalls mit dem System verbunden werden, um es dem Datenverarbeitungssystem zu ermöglichen, über dazwischengeschaltete private oder öffentliche Netzwerke mit anderen Datenverarbeitungssystemen oder entfernt angeordneten Druckern oder Speichereinheiten verbunden zu werden. Modems, Kabelmodems und Ethernet-Karten sind nur einige der derzeit erhältlichen Arten von Netzwerkadaptern.

[0320] Die hier verwendete Terminologie dient lediglich zum Zweck des Beschreibens von speziellen Ausführungsformen und soll die Erfindung nicht einschränken. Wie hier verwendet, sollen die Singularformen „ein/eine/einer/eines“ und „der/die/das“ ebenfalls die Pluralformen umfassen, es sei denn, der Zusammenhang zeigt eindeutig etwas anderes auf. Es versteht sich ferner, dass die Begriffe „aufweisen“ und/oder „aufweisend“, wenn sie in dieser Beschreibung verwendet werden, die Anwesenheit von angegebenen Merkmalen, Ganzzahlen, Schritten, Operationen, Elementen und/oder Komponenten spezifizieren, jedoch nicht die Anwesenheit oder Hinzufügung von einem oder mehreren anderen Merkmalen, Ganzzahlen, Schritten, Operationen, Elementen, Komponenten und/oder Gruppen davon ausschließen.

[0321] Die entsprechenden Strukturen, Materialien, Maßnahmen und Äquivalente aller Mittel oder Schritt-plus-Funktion-Elemente in den nachfolgenden Ansprüchen sollen alle Strukturen, Materialien oder Maßnahmen zur Durchführung der Funktion in Kombination mit anderen beanspruchten Elementen umfassen, wie dies speziell beansprucht wird. Die Beschreibung einer oder mehrere Ausführungsformen wurde zum Zwecke der Veranschaulichung und Beschreibung vorgestellt, soll jedoch nicht erschöpfend oder auf die Erfindung in der offenbarten Form beschränkt sein. Für Fachleute ist offensichtlich, dass viele Änderungen und Abwandlungen möglich sind. Die Ausführungsform wurde ausgewählt und beschrieben, um verschiedene Aspekte und die praktische Anwendung am besten zu erläutern und um anderen Fachleuten ein Verständnis verschiedener Ausführungsformen mit verschiedenen Änderungen zu ermöglichen, wie sie für die jeweils beabsichtigte Verwendung geeignet sind.

Patentansprüche

1. Computerprogrammprodukt zum Ermöglichen der Verarbeitung in einer Datenverarbeitungsumgebung, wobei das Computerprogrammprodukt aufweist:
ein oder mehrere durch einen Computer lesbare Speichermedien und
Programmanweisungen, die gemeinsam auf dem einen oder mehreren durch einen Computer lesbaren Speichermedien gespeichert sind, um ein Verfahren durchzuführen, das umfasst:
Abrufen einer Meldung, dass ein als ungültig ermittelter Wert in den Eingabedaten für eine Berechnung einer oder mehrerer Berechnungen oder in den Ausgabedaten enthalten ist, die sich aus der einen oder mehreren Berechnungen ergeben haben, wobei der Wert aufgrund einer Ausnahme einer Mehrzahl von Ausnahmen als ungültig ermittelt wird; und
Setzen eines Übersichtsanzeigers auf der Grundlage des Abrufens der Meldung, dass der Wert als ungültig ermittelt wurde, wobei der Übersichtsanzeiger die Mehrzahl von Ausnahmen insgesamt darstellt.

2. Computerprogrammprodukt nach dem vorherigen Anspruch, wobei der Übersichtsanzeiger die Mehrzahl von Ausnahmen darstellt, ohne dass zwischen der Mehrzahl von Ausnahmen unterschieden wird.

3. Computerprogrammprodukt nach einem der vorherigen Ansprüche, wobei der Übersichtsanzeiger unabhängig davon gesetzt wird, welche Ausnahme der Mehrzahl von Ausnahmen verwendet wird, um zu ermitteln, ob der Wert ungültig ist.

4. Computerprogrammprodukt nach einem der vorherigen Ansprüche, wobei es sich bei dem Übersichtsanzeiger um einen Bereichsverletzungsanzeiger eines Ausnahme-Merkers handelt, der an einem Ort bereitgestellt wird, der durch eine Anweisung spezifiziert wird, die zum Durchführen der einen oder mehrerer Berechnungen ausgegeben wird.

5. Computerprogrammprodukt nach einem der vorherigen Ansprüche, wobei die Mehrzahl von Ausnahmen einen nichtnumerischen Wert, einen nichtdarstellbaren numerischen Wert und einen Wert außerhalb des gültigen Bereichs umfasst.

6. Computerprogrammprodukt nach einem der vorherigen Ansprüche, wobei das Abrufen der Meldung, dass der Wert als ungültig ermittelt wurde, auf Ausführen einer Anweisung beruht, die die eine oder mehrere Berechnungen durchführt.

7. Computerprogrammprodukt nach dem vorherigen Anspruch, wobei die Anweisung so konfiguriert ist, dass sie eine Mehrzahl von Funktionen durchführt, die Berechnungen durchführen, und die Mehrzahl von Funktionen den Übersichtsanzeiger verwendet.

8. Computerprogrammprodukt nach einem der beiden vorherigen Ansprüche, wobei es sich bei der Anweisung um eine Anweisung in einem neuronalen Netzwerk handelt, die Berechnungen bei Eingabe-Tensoren durchführt, um Ausgabe-Tensoren bereitzustellen, die für die Verarbeitung durch künstliche Intelligenz verwendet werden, wobei die Anweisung so konfiguriert ist, dass sie eine Mehrzahl von Funktionen durchführt, die Berechnungen durchführen, und die Mehrzahl von Funktionen den Übersichtsanzeiger verwendet.

9. Computerprogrammprodukt nach einem der vorherigen Ansprüche, wobei der Übersichtsanzeiger für eine bestimmte Anweisung definiert wird, wobei eine andere Anweisung einen anderen Übersichtsanzeiger verwendet.

10. Computerprogrammprodukt nach dem vorherigen Anspruch, wobei das Verfahren weiterhin Ermitteln eines Wertes eines Bedingungscodes umfasst, der auf der Grundlage des Ausführens der bestimmten Anweisung gesetzt wurde, und wobei die Gültigkeit des Übersichtsanzeigers auf dem Feststellen beruht, dass der Wert des Bedingungscodes ein ausgewählter Wert ist.

11. Computersystem zum Ermöglichen der Verarbeitung in einer Datenverarbeitungsumgebung, wobei das Computersystem aufweist:
einen Speicher; und
mindestens einen Prozessor, der mit dem Speicher Daten austauscht, wobei das Computersystem so konfiguriert ist, dass es ein Verfahren durchführt, wobei das Verfahren umfasst:
Abrufen einer Meldung, dass ein als ungültig ermittelter Wert in den Eingabedaten für eine Berechnung einer oder mehrerer Berechnungen oder in den Ausgabedaten enthalten ist, die sich aus der einen oder mehreren Berechnungen ergeben haben, wobei der Wert aufgrund einer Ausnahme einer Mehrzahl von Ausnahmen als ungültig ermittelt wird; und
Setzen eines Übersichtsanzeigers auf der Grundlage des Abrufens der Meldung, dass der Wert als ungültig ermittelt wurde, wobei der Übersichtsanzeiger die Mehrzahl von Ausnahmen insgesamt darstellt.

12. Computersystem nach dem vorherigen Anspruch, wobei der Übersichtsanzeiger die Mehrzahl von Ausnahmen darstellt, ohne dass zwischen der Mehrzahl von Ausnahmen unterschieden wird.

13. Computersystem nach einem der beiden vorherigen Ansprüche, wobei der Übersichtsanzeiger unabhängig davon gesetzt wird, welche Ausnahme der Mehrzahl von Ausnahmen verwendet wird, um zu ermitteln, ob der Wert ungültig ist.

14. Computersystem nach einem der drei vorherigen Ansprüche, wobei das Abrufen der Meldung, dass der Wert als ungültig ermittelt wurde, auf Ausführen einer Anweisung beruht, die die eine oder mehrere Berechnungen durchführt, und wobei es sich bei der Anweisung um eine Anweisung in einem neuronalen Netzwerk handelt, die Berechnungen bei Eingabe-Tensoren durchführt, um Ausgabe-Tensoren bereitzustellen, die für die Verarbeitung durch künstliche Intelligenz verwendet werden, wobei die Anweisung so konfi-

guriert ist, dass sie eine Mehrzahl von Funktionen durchführt, die Berechnungen durchführen, und die Mehrzahl von Funktionen den Übersichtsanzeiger verwendet.

15. Computersystem nach einem der vier vorherigen Ansprüche, wobei der Übersichtsanzeiger für eine bestimmte Anweisung definiert wird, wobei eine andere Anweisung einen anderen Übersichtsanzeiger verwendet.

16. Durch einen Computer implementiertes Verfahren zum Ermöglichen der Verarbeitung in einer Datenverarbeitungsumgebung, wobei das durch einen Computer implementierte Verfahren umfasst:
Abrufen einer Meldung, dass ein als ungültig ermittelter Wert in den Eingabedaten für eine Berechnung einer oder mehrerer Berechnungen oder in den Ausgabedaten enthalten ist, die sich aus der einen oder mehreren Berechnungen ergeben haben, wobei der Wert aufgrund einer Ausnahme einer Mehrzahl von Ausnahmen als ungültig ermittelt wird; und
Setzen eines Übersichtsanzeigers auf der Grundlage des Abrufens der Meldung, dass der Wert als ungültig ermittelt wurde, wobei der Übersichtsanzeiger die Mehrzahl von Ausnahmen insgesamt darstellt.

17. Durch einen Computer implementiertes Verfahren nach dem vorherigen Anspruch, wobei der Übersichtsanzeiger die Mehrzahl von Ausnahmen darstellt, ohne dass zwischen der Mehrzahl von Ausnahmen unterschieden wird.

18. Durch einen Computer implementiertes Verfahren nach einem der beiden vorherigen Ansprüche, wobei der Übersichtsanzeiger unabhängig davon gesetzt wird, welche Ausnahme der Mehrzahl von Ausnahmen verwendet wird, um zu ermitteln, ob der Wert ungültig ist.

19. Durch einen Computer implementiertes Verfahren nach einem der drei vorherigen Ansprüche, wobei das Abrufen der Meldung, dass der Wert als ungültig ermittelt wurde, auf Ausführen einer Anweisung beruht, die die eine oder mehrere Berechnungen durchführt, und wobei es sich bei der Anweisung um eine Anweisung in einem neuronalen Netzwerk handelt, die Berechnungen bei Eingabe-Tensoren durchführt, um Ausgabe-Tensoren bereitzustellen, die für die Verarbeitung durch künstliche Intelligenz verwendet werden, wobei die Anweisung so konfiguriert ist, dass sie eine Mehrzahl von Funktionen durchführt, die Berechnungen durchführen, und die Mehrzahl von Funktionen den Übersichtsanzeiger verwendet.

20. Durch einen Computer implementiertes Verfahren nach einem der vier vorherigen Ansprüche, wobei der Übersichtsanzeiger für eine bestimmte Anweisung definiert wird, wobei eine andere Anweisung einen anderen Übersichtsanzeiger verwendet.

Es folgen 17 Seiten Zeichnungen

Anhängende Zeichnungen

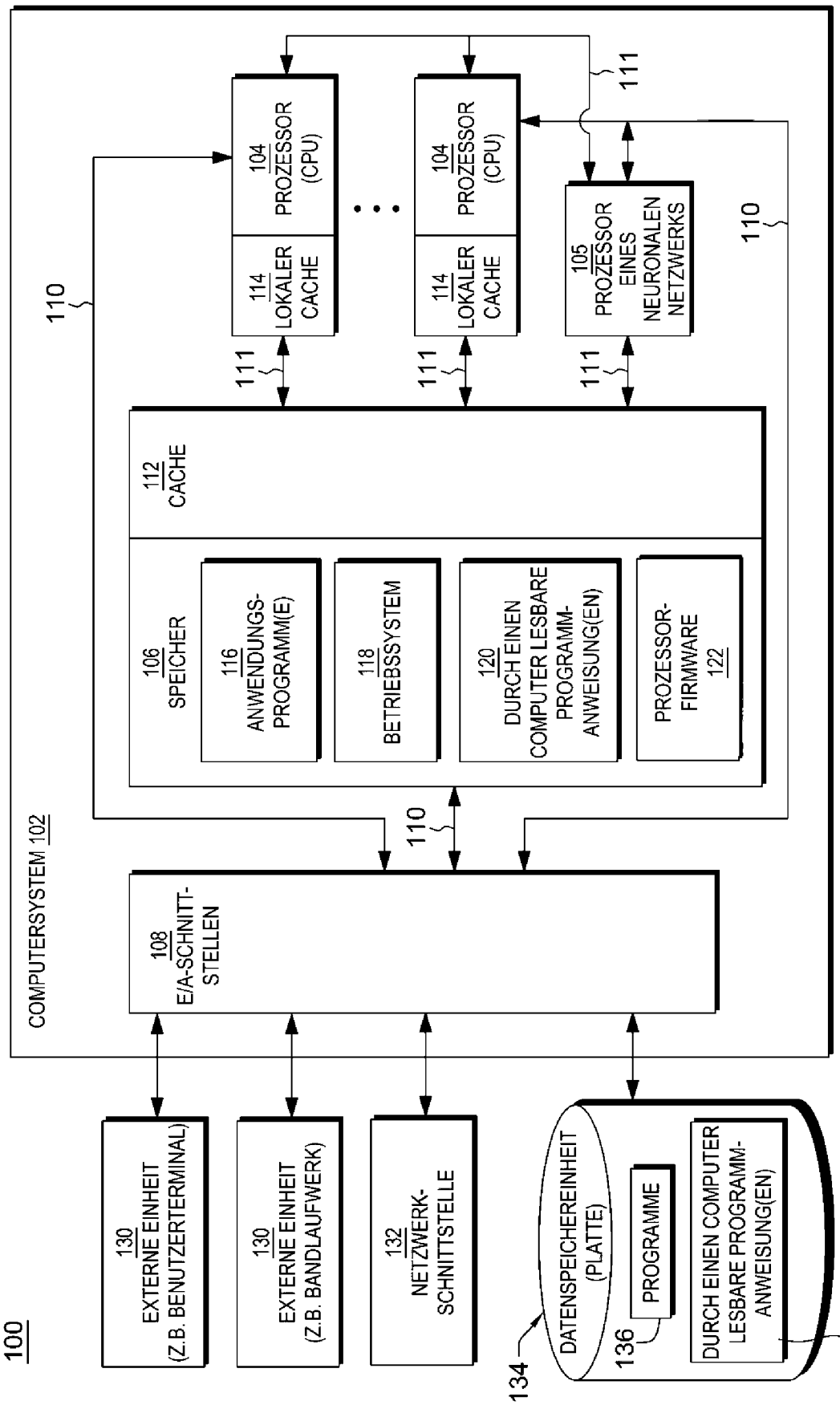


FIG. 1A

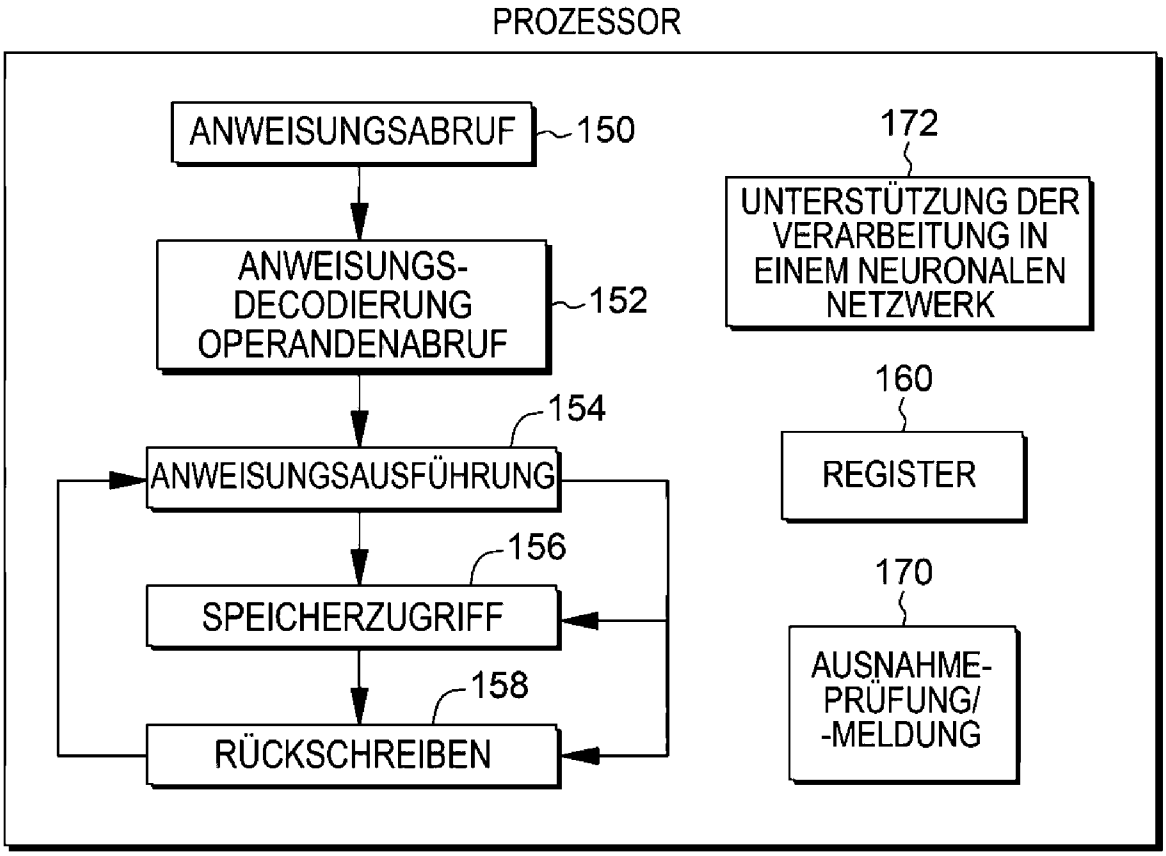

PROZESSOR
ANWEISUNGSABRUF
150
ANWEISUNGS-
DECODIERUNG
OPERANDENABRUF
152
154
ANWEISUNGSAUSFÜHRUNG
156
SPEICHERZUGRIFF
158
RÜCKSCHREIBEN
172
UNTERSTÜTZUNG DER
VERARBEITUNG IN
EINEM NEURONALEN
NETZWERK
160
REGISTER
170
AUSNAHME-
PRÜFUNG/
-MELDUNG

FIG. 1B

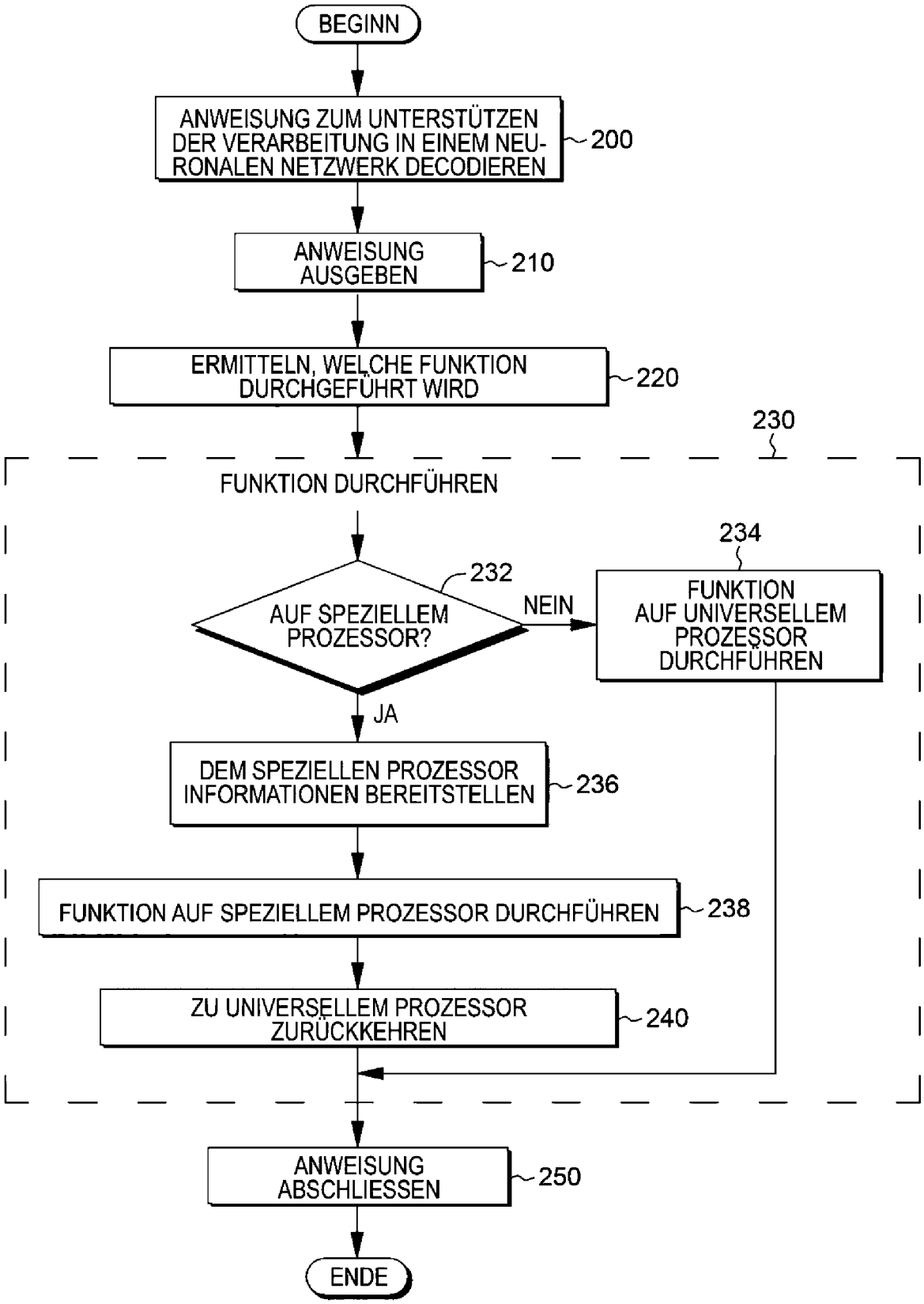

BEGINN
ANWEISUNG ZUM UNTERSTÜTZEN DER VERARBEITUNG IN EINEM NEURONALEN NETZWERK DECODIEREN
200
ANWEISUNG AUSGEBEN
210
ERMITTELN, WELCHE FUNKTION DURCHGEFÜHRT WIRD
220
230
FUNKTION DURCHFÜHREN
232
AUF SPEZIELLEM PROZESSOR?
NEIN
234
FUNKTION AUF UNIVERSELLEM PROZESSOR DURCHFÜHREN
JA
DEM SPEZIELLEN PROZESSOR INFORMATIONEN BEREITSTELLEN
236
FUNKTION AUF SPEZIELLEM PROZESSOR DURCHFÜHREN
238
ZU UNIVERSELLEM PROZESSOR ZURÜCKKEHREN
240
ANWEISUNG ABSCHLIESSEN
250
ENDE

FIG. 2A

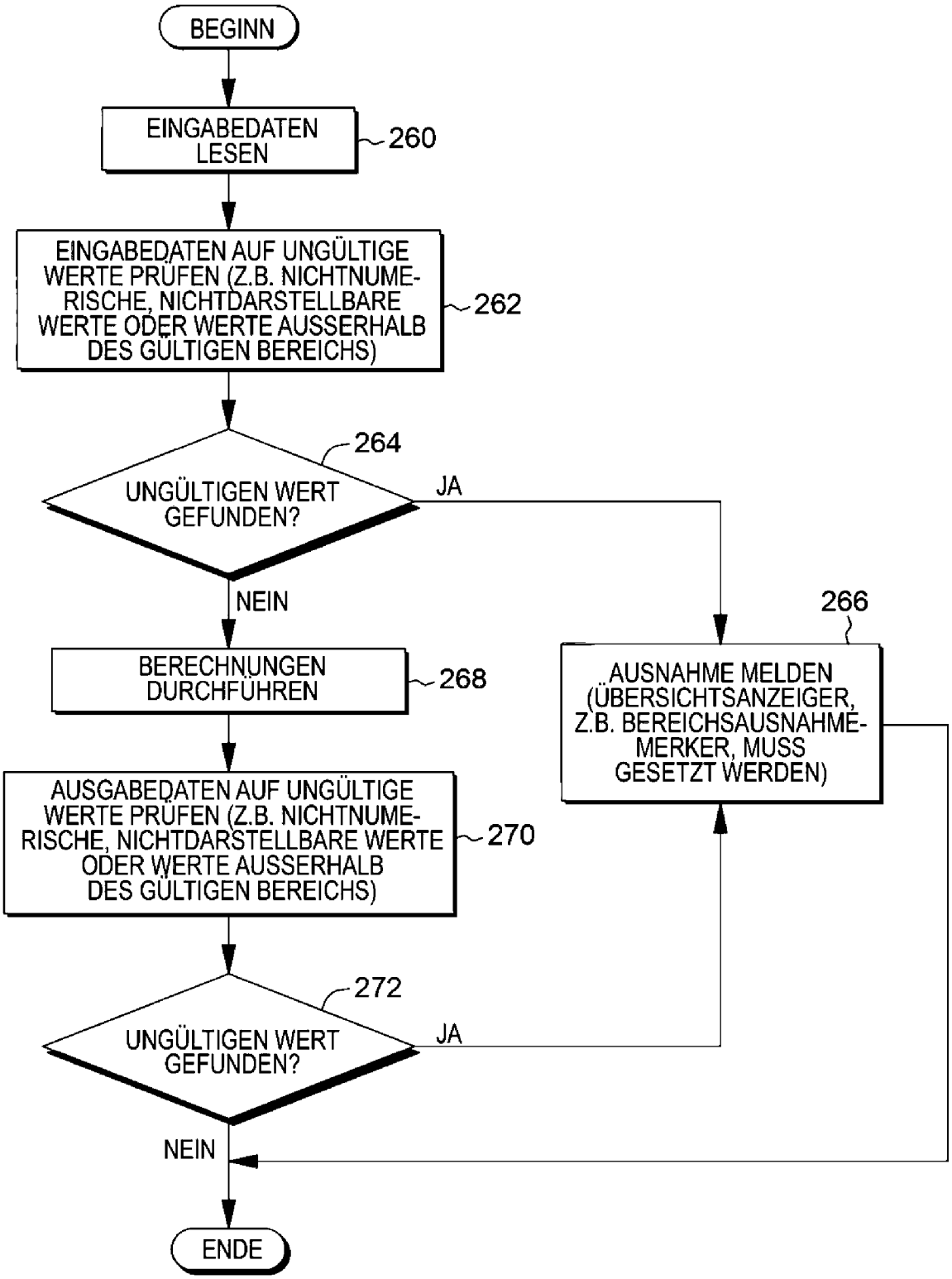

BEGINN
EINGABEDATEN LESEN
260
EINGABEDATEN AUF UNGÜLTIGE WERTE PRÜFEN (Z.B. NICHTNUMERISCHE, NICHTDARSTELLBARE WERTE ODER WERTE AUSSERHALB DES GÜLTIGEN BEREICHS)
262
264
UNGÜLTIGEN WERT GEFUNDEN?
JA
NEIN
266
AUSNAHME MELDEN (ÜBERSICHTSANZEIGER, Z.B. BEREICHSAUSNAHME-MERKER, MUSS GESETZT WERDEN)
BERECHNUNGEN DURCHFÜHREN
268
AUSGABEDATEN AUF UNGÜLTIGE WERTE PRÜFEN (Z.B. NICHTNUMERISCHE, NICHTDARSTELLBARE WERTE ODER WERTE AUSSERHALB DES GÜLTIGEN BEREICHS)
270
272
UNGÜLTIGEN WERT GEFUNDEN?
JA
NEIN
ENDE

FIG. 2B

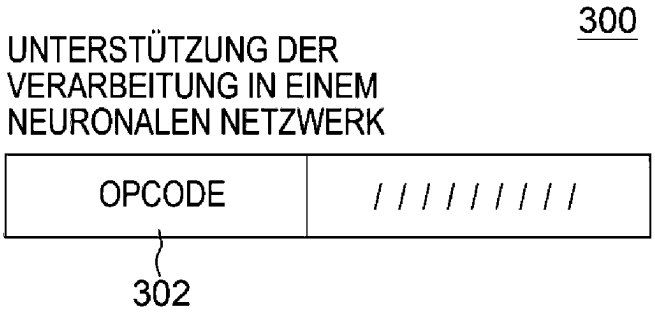

300
UNTERSTÜTZUNG DER VERARBEITUNG IN EINEM NEURONALEN NETZWERK
OPCODE
/ / / / / / / / /
302

FIG. 3A

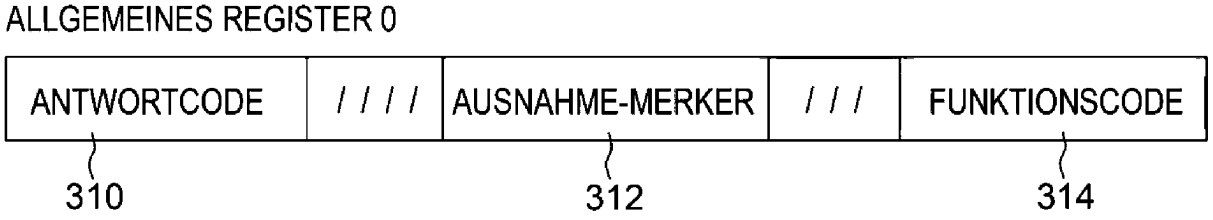

ALLGEMEINES REGISTER 0
ANTWORTCODE
/ / / /
AUSNAHME-MERKER
/ / /
FUNKTIONSCODE
310
312
314

FIG. 3B

314

CODE (DEZ)	CODE (HEX)	FUNKTION	GRÖSSE DES PARAMETER-BLOCKS (BYTES)
0	0	NNPA - QAF	256
16	10	NNPA - ADD	4096
17	11	NNPA - SUB	4096
18	12	NNPA - MUL	4096
19	13	NNPA - DIV	4096
20	14	NNPA - MIN	4096
21	15	NNPA - MAX	4096
32	20	NNPA - LOG	4096
33	21	NNPA - EXP	4096
49	31	NNPA - RELU	4096
50	32	NNPA - TANH	4096
51	33	NNPA - SIGMOID	4096
52	34	NNPA - SOFTMAX	4096
64	40	NNPA - BATCHNORM	4096
80	50	NNPA - MAXPOOL2D	4096
81	51	NNPA - AVGPOOL2D	4096
96	60	NNPA - LSTMACT	4096
97	61	NNPA - GRUACT	4096
112	70	NNPA - CONVOLUTION	4096
113	71	NNPA - MATMUL-OP	4096
114	72	NNPA - MATMUL-OP-BCAST23	4096

FIG. 3C

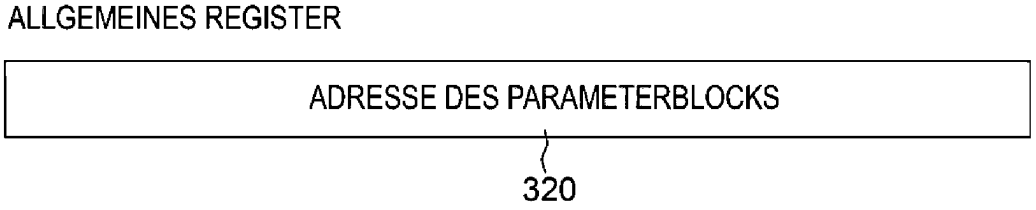

ALLGEMEINES REGISTER
ADRESSE DES PARAMETERBLOCKS
320

FIG. 3D

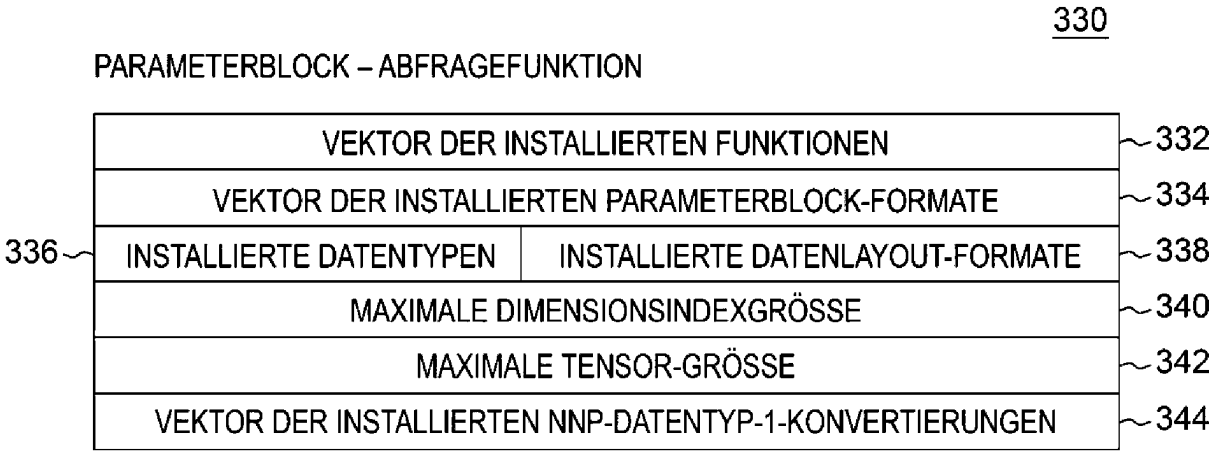

330
PARAMETERBLOCK – ABFRAGEFUNKTION
VEKTOR DER INSTALLIERTEN FUNKTIONEN
332
VEKTOR DER INSTALLIERTEN PARAMETERBLOCK-FORMATE
334
336
INSTALLIERTE DATENTYPEN
INSTALLIERTE DATENLAYOUT-FORMATE
338
MAXIMALE DIMENSIONSINDEXGRÖSSE
340
MAXIMALE TENSOR-GRÖSSE
342
VEKTOR DER INSTALLIERTEN NNP-DATENTYP-1-KONVERTIERUNGEN
344

FIG. 3E

350

PARAMETERBLOCK – NICHTABFRAGEFUNKTIONEN

352	VERSIONSNUMMER DES PARAMETERBLOCKS	VERSIONSNUMMER DES MODELLS - 354	FORTSETZUNGSMERKER 356
	FUNKTIONSSPEZIFISCHE SICHERUNGSBEREICHSADRESSE		358
	AUSGABE-TENSOR-DESKRIPTOR 1		360
	AUSGABE-TENSOR-DESKRIPTOR 2		
	EINGABE-TENSOR-DESKRIPTOR 1		365
	EINGABE-TENSOR-DESKRIPTOR 2		
	EINGABE-TENSOR-DESKRIPTOR 3		
	FUNKTIONSSPEZIFISCHER PARAMETER 1	FUNKTIONSSPEZIFISCHER PARAMETER 2	370
	FUNKTIONSSPEZIFISCHER PARAMETER 3	FUNKTIONSSPEZIFISCHER PARAMETER 4	
	FUNKTIONSSPEZIFISCHER PARAMETER 5		
	FORTSETZUNGSZUSTANDPUFFER		375

FIG. 3F

360, 365

TENSOR-DESKRIPTOR

382	DATENLAYOUT-FORMAT	DATENTYP	384
	DIMENSION-4-INDEXGRÖSSE	DIMENSION-3-INDEXGRÖSSE	386
	DIMENSION-2-INDEXGRÖSSE	DIMENSION-1-INDEXGRÖSSE	
	TENSOR-ADRESSE		388

FIG. 3G

400

NNP-DATENTYP-1

S	EXPONENT + 31	BRUCHZAHL
402	404	406

FIG. 4

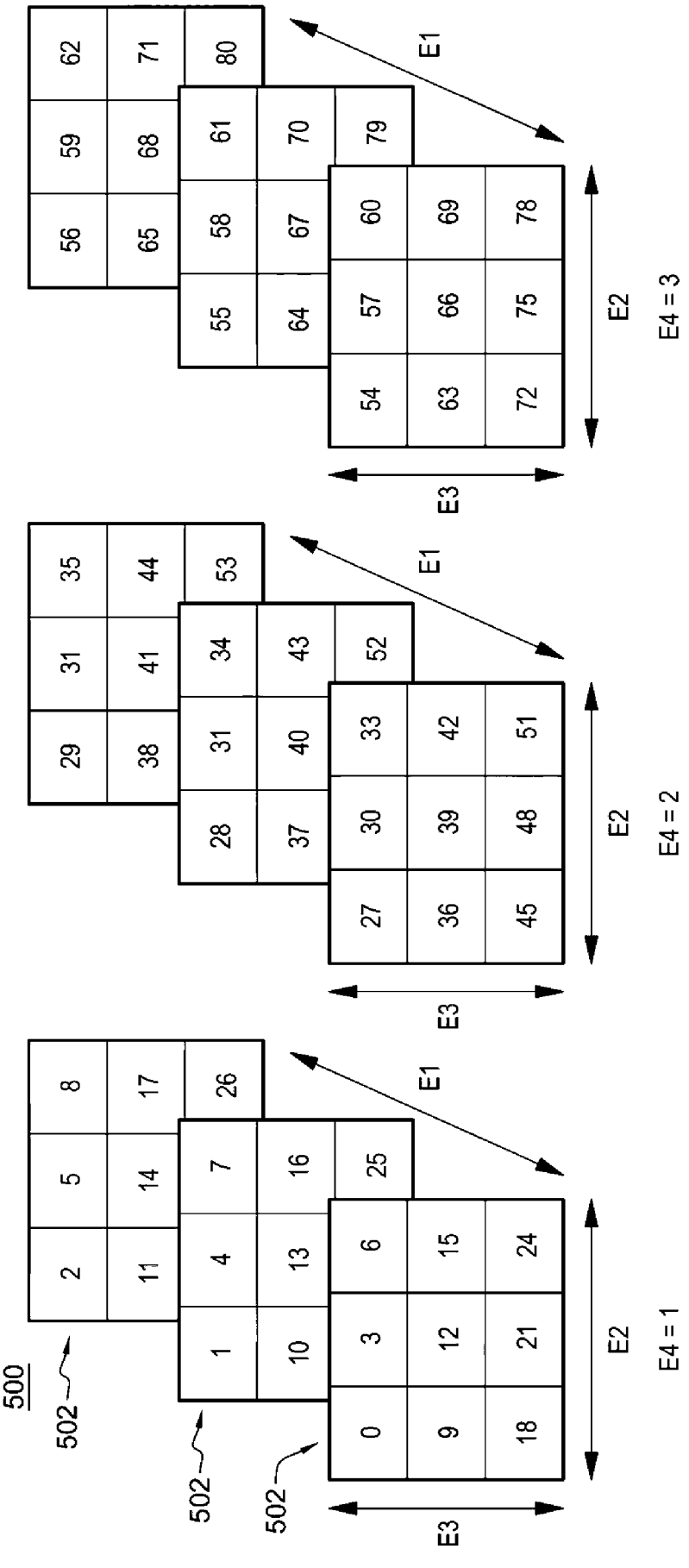


FIG. 5A

FIG. 5B

FIG. 5C

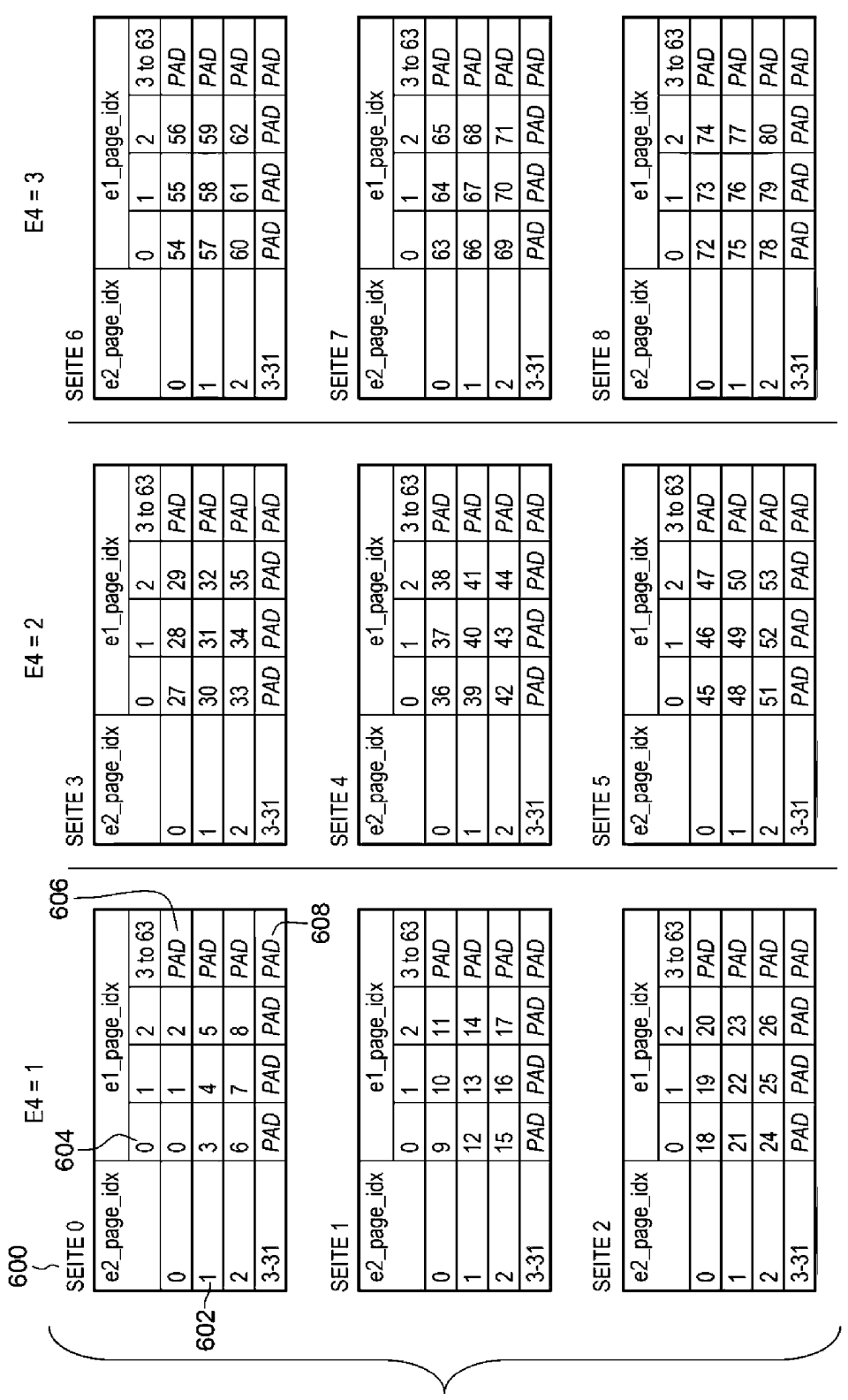


FIG. 6A

FIG. 6B

FIG. 6C

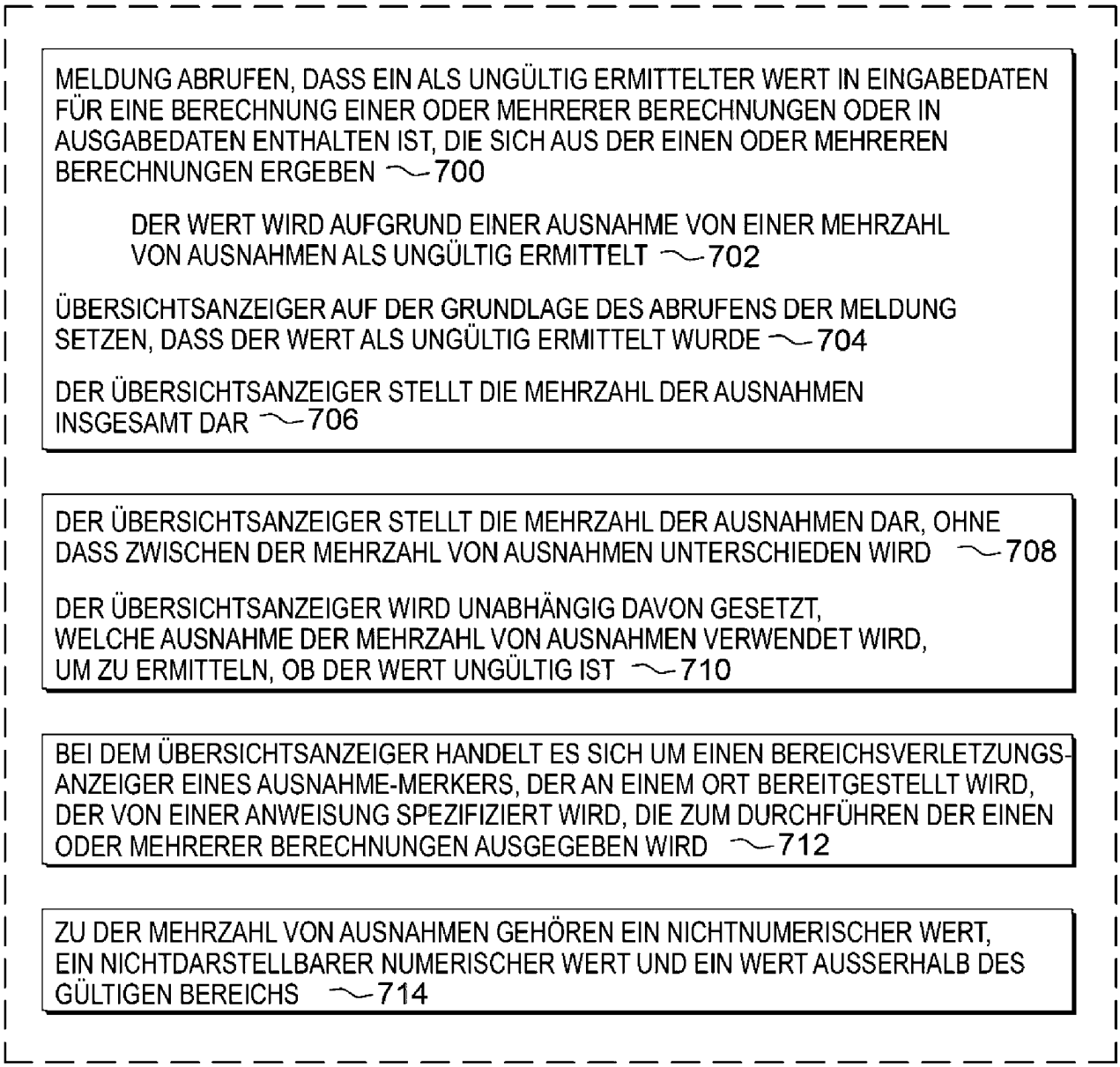

MELDUNG ABRUFEN, DASS EIN ALS UNGÜLTIG ERMITTELTER WERT IN EINGABEDATEN FÜR EINE BERECHNUNG EINER ODER MEHRERER BERECHNUNGEN ODER IN AUSGABEDATEN ENTHALTEN IST, DIE SICH AUS DER EINEN ODER MEHREREN BERECHNUNGEN ERGEBEN 700
DER WERT WIRD AUFGRUND EINER AUSNAHME VON EINER MEHRZAHL VON AUSNAHMEN ALS UNGÜLTIG ERMITTELT 702
ÜBERSICHTSANZEIGER AUF DER GRUNDLAGE DES ABRUFENS DER MELDUNG SETZEN, DASS DER WERT ALS UNGÜLTIG ERMITTELT WURDE 704
DER ÜBERSICHTSANZEIGER STELLT DIE MEHRZAHL DER AUSNAHMEN INSGESAMT DAR 706
DER ÜBERSICHTSANZEIGER STELLT DIE MEHRZAHL DER AUSNAHMEN DAR, OHNE DASS ZWISCHEN DER MEHRZAHL VON AUSNAHMEN UNTERSCHIEDEN WIRD 708
DER ÜBERSICHTSANZEIGER WIRD UNABHÄNGIG DAVON GESETZT, WELCHE AUSNAHME DER MEHRZAHL VON AUSNAHMEN VERWENDET WIRD, UM ZU ERMITTELN, OB DER WERT UNGÜLTIG IST 710
BEI DEM ÜBERSICHTSANZEIGER HANDELT ES SICH UM EINEN BEREICHSVERLETZUNGS-ANZEIGER EINES AUSNAHME-MERKERS, DER AN EINEM ORT BEREITGESTELLT WIRD, DER VON EINER ANWEISUNG SPEZIFIZIERT WIRD, DIE ZUM DURCHFÜHREN DER EINEN ODER MEHRERER BERECHNUNGEN AUSGEGEBEN WIRD 712
ZU DER MEHRZAHL VON AUSNAHMEN GEHÖREN EIN NICHTNUMERISCHER WERT, EIN NICHTDARSTELLBARER NUMERISCHER WERT UND EIN WERT AUSSERHALB DES GÜLTIGEN BEREICHS 714

FIG. 7A

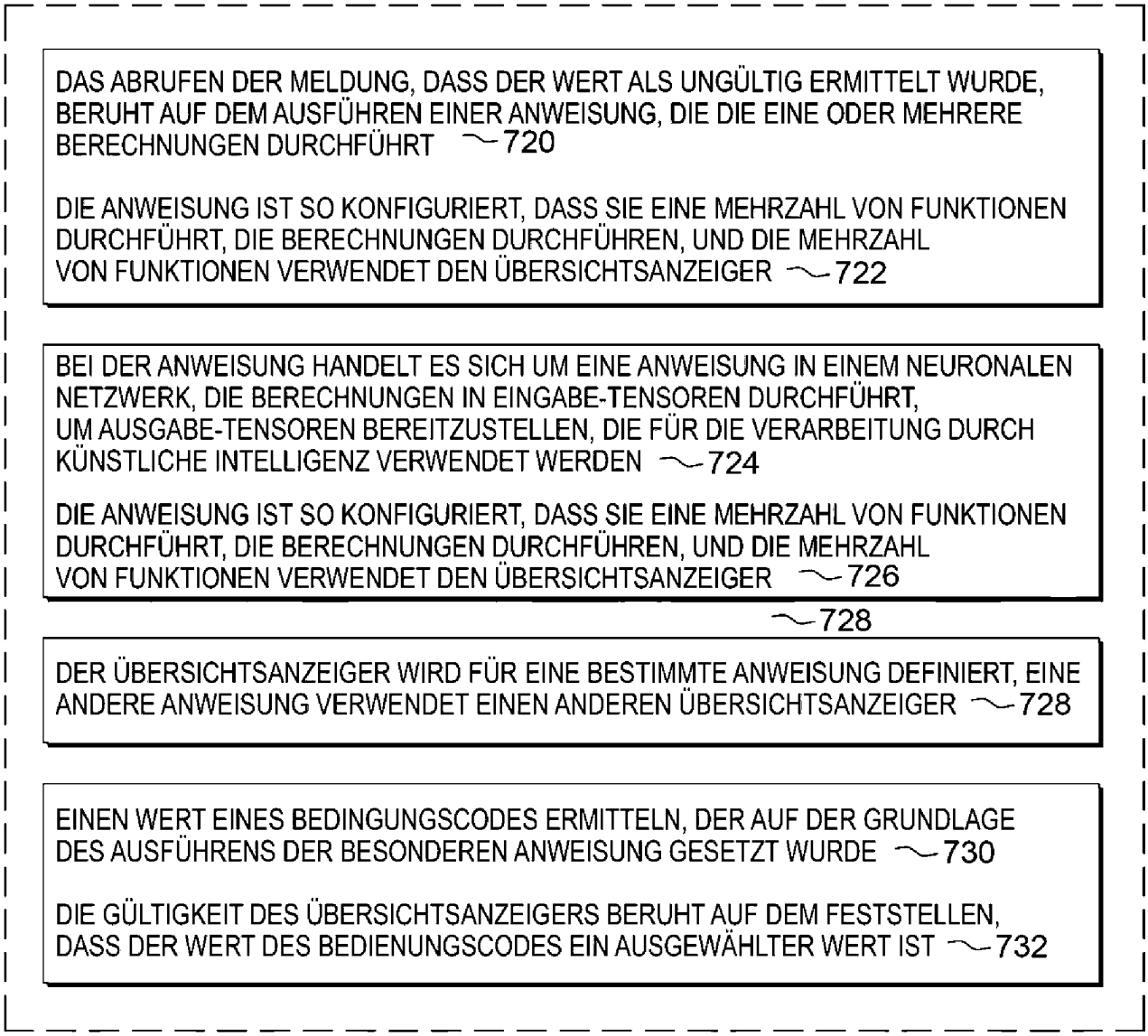

DAS ABRUFEN DER MELDUNG, DASS DER WERT ALS UNGÜLTIG ERMITTELT WURDE, BERUHT AUF DEM AUSFÜHREN EINER ANWEISUNG, DIE DIE EINE ODER MEHRERE BERECHNUNGEN DURCHFÜHRT 720
DIE ANWEISUNG IST SO KONFIGURIERT, DASS SIE EINE MEHRZAHL VON FUNKTIONEN DURCHFÜHRT, DIE BERECHNUNGEN DURCHFÜHREN, UND DIE MEHRZAHL VON FUNKTIONEN VERWENDET DEN ÜBERSICHTSANZEIGER 722
BEI DER ANWEISUNG HANDELT ES SICH UM EINE ANWEISUNG IN EINEM NEURONALEN NETZWERK, DIE BERECHNUNGEN IN EINGABE-TENSOREN DURCHFÜHRT, UM AUSGABE-TENSOREN BEREITZUSTELLEN, DIE FÜR DIE VERARBEITUNG DURCH KÜNSTLICHE INTELLIGENZ VERWENDET WERDEN 724
DIE ANWEISUNG IST SO KONFIGURIERT, DASS SIE EINE MEHRZAHL VON FUNKTIONEN DURCHFÜHRT, DIE BERECHNUNGEN DURCHFÜHREN, UND DIE MEHRZAHL VON FUNKTIONEN VERWENDET DEN ÜBERSICHTSANZEIGER 726
728
DER ÜBERSICHTSANZEIGER WIRD FÜR EINE BESTIMMTE ANWEISUNG DEFINIERT, EINE ANDERE ANWEISUNG VERWENDET EINEN ANDEREN ÜBERSICHTSANZEIGER 728
EINEN WERT EINES BEDINGUNGSCODES ERMITTELN, DER AUF DER GRUNDLAGE DES AUSFÜHRENS DER BESONDEREN ANWEISUNG GESETZT WURDE 730
DIE GÜLTIGKEIT DES ÜBERSICHTSANZEIGERS BERUHT AUF DEM FESTSTELLEN, DASS DER WERT DES BEDIENUNGSCODES EIN AUSGEWÄHLTER WERT IST 732

FIG. 7B

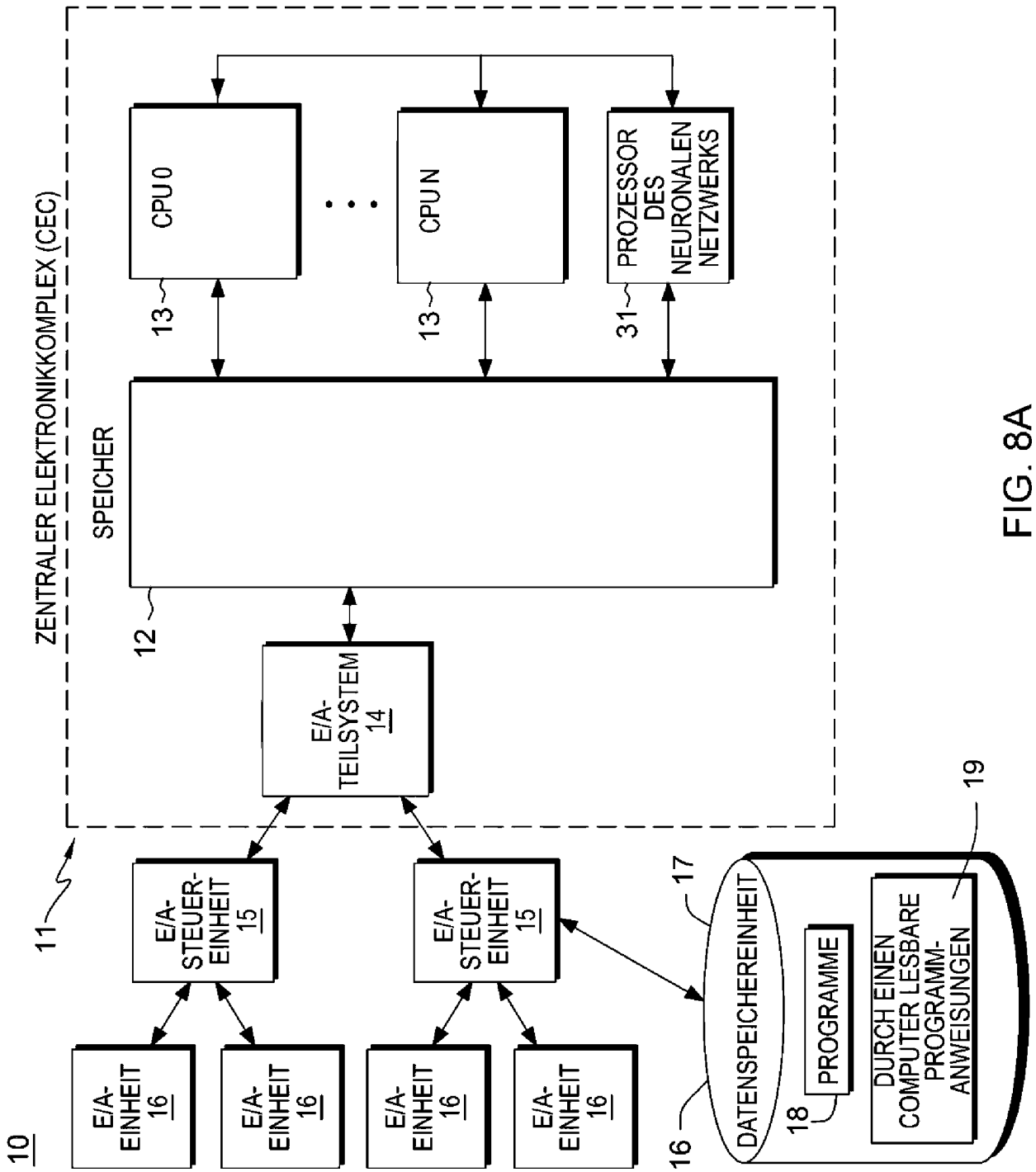

10
11
ZENTRALER ELEKTRONIKKOMPLEX (CEC)
CPU 0
CPU N
PROZESSOR DES NEURONALEN NETZWERKS
13
13
31
SPEICHER
12
E/A-TEILSYSTEM 14
E/A-STEUER-EINHEIT 15
E/A-STEUER-EINHEIT 15
E/A-EINHEIT 16
E/A-EINHEIT 16
E/A-EINHEIT 16
E/A-EINHEIT 16
16
17
DATENSPEICHEREINHEIT
18
PROGRAMME
19
DURCH EINEN COMPUTER LESBARE PROGRAMM-ANWEISUNGEN

FIG. 8A

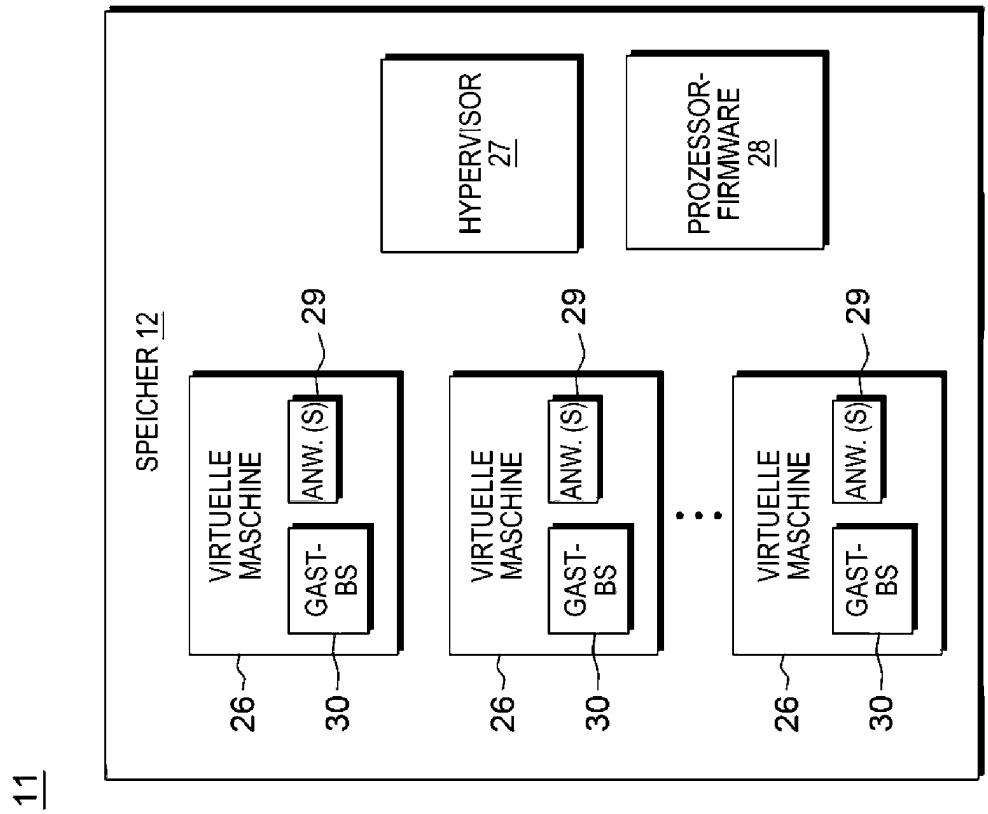

11
SPEICHER 12
VIRTUELLE MASCHINE
ANW. (S)
GAST-BS
29
26
30
HYPERVISOR 27
PROZESSOR-FIRMWARE 28

FIG. 8C

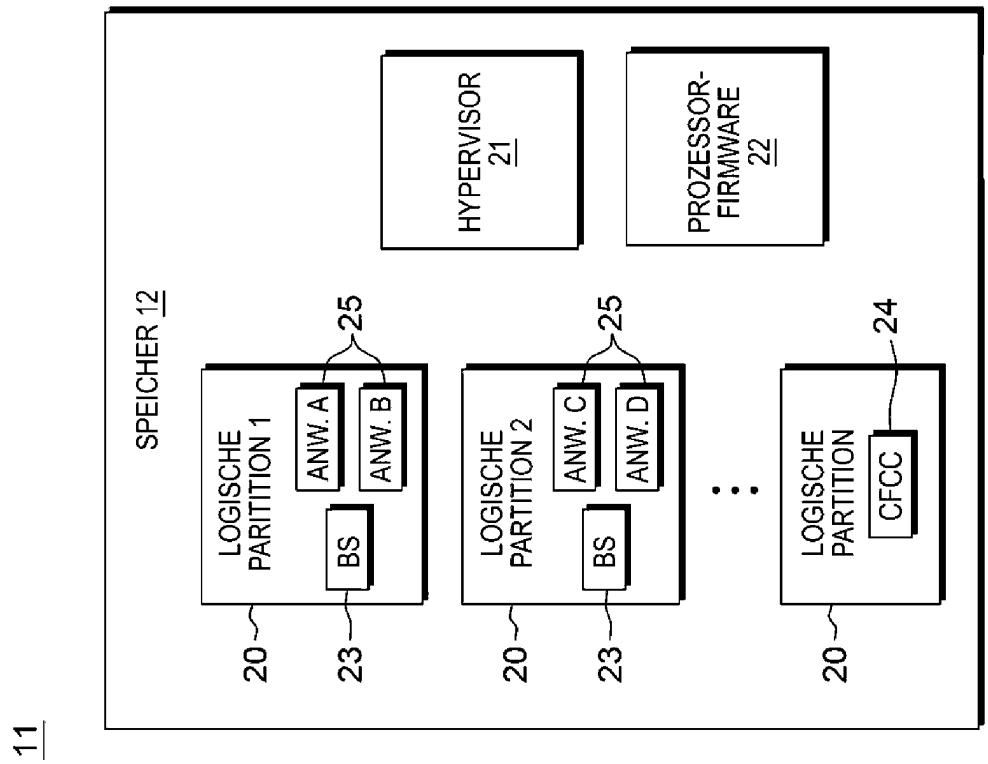

11
SPEICHER 12
LOGISCHE PARITITION 1
ANW. A
ANW. B
BS
25
20
23
LOGISCHE PARTITION 2
ANW. C
ANW. D
LOGISCHE PARTITION
CFCC
24
HYPERVISOR 21
PROZESSOR-FIRMWARE 22

FIG. 8B

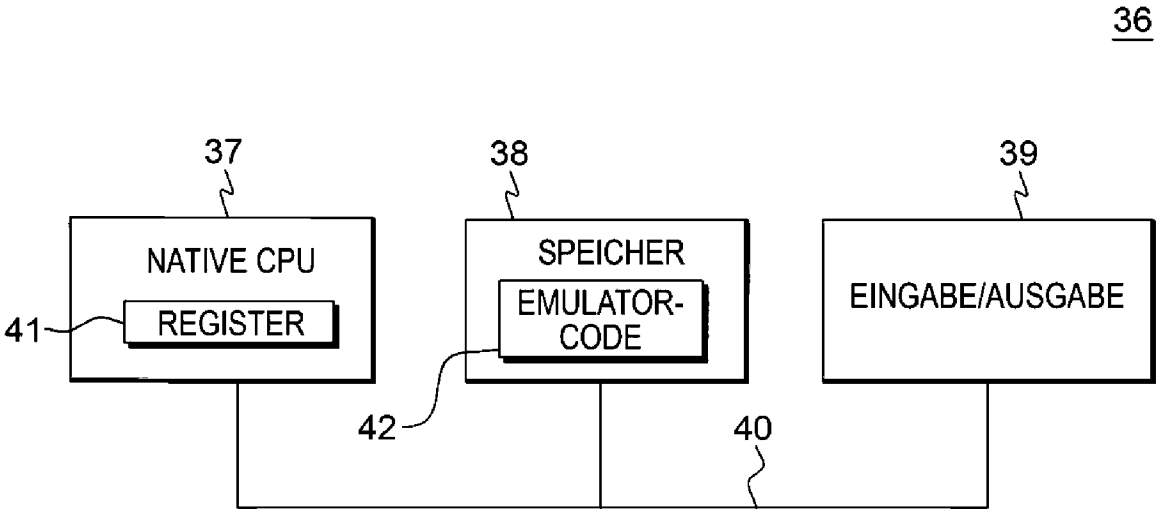

36
37
38
39
NATIVE CPU
REGISTER
41
SPEICHER
EMULATOR-CODE
42
EINGABE/AUSGABE
40

FIG. 9A

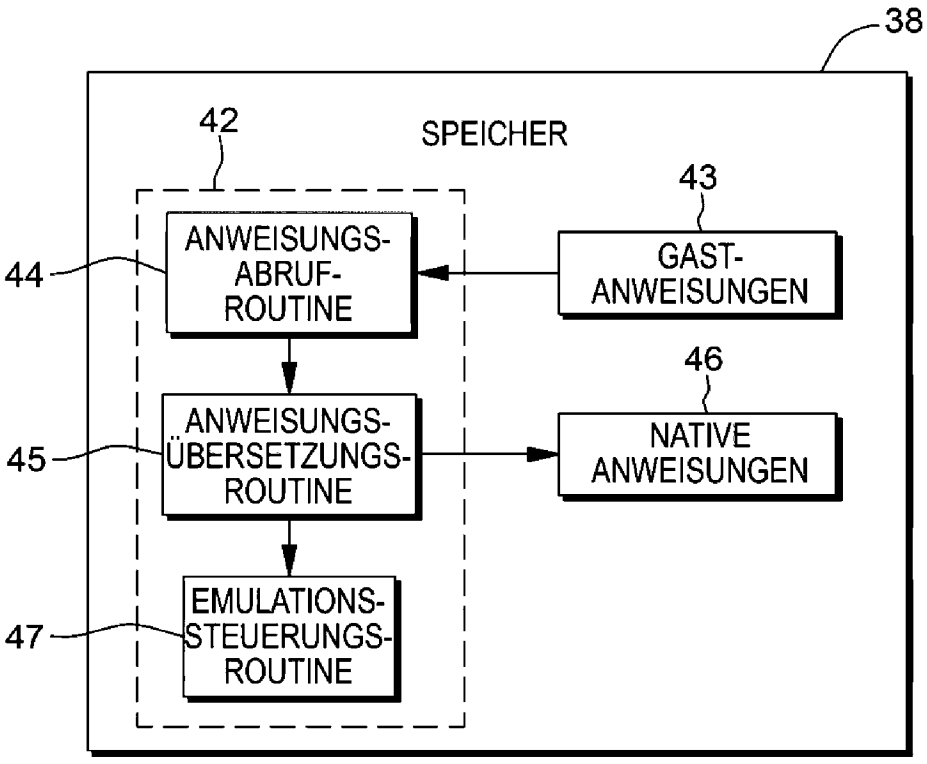

38
42
SPEICHER
43
ANWEISUNGS-ABRUF-ROUTINE
44
GAST-ANWEISUNGEN
46
ANWEISUNGS-ÜBERSETZUNGS-ROUTINE
45
NATIVE ANWEISUNGEN
EMULATIONS-STEUERUNGS-ROUTINE
47

FIG. 9B

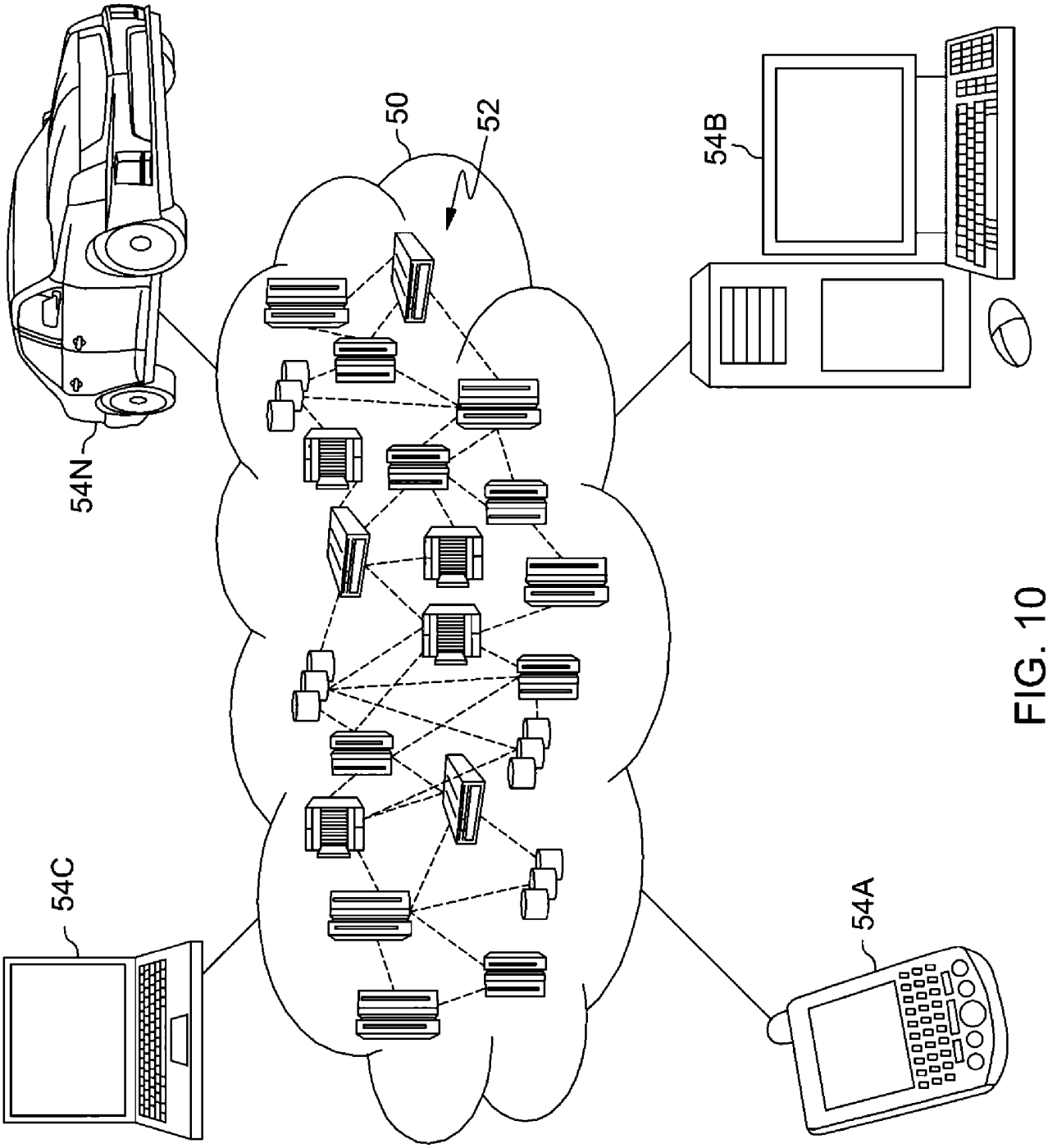


FIG. 10

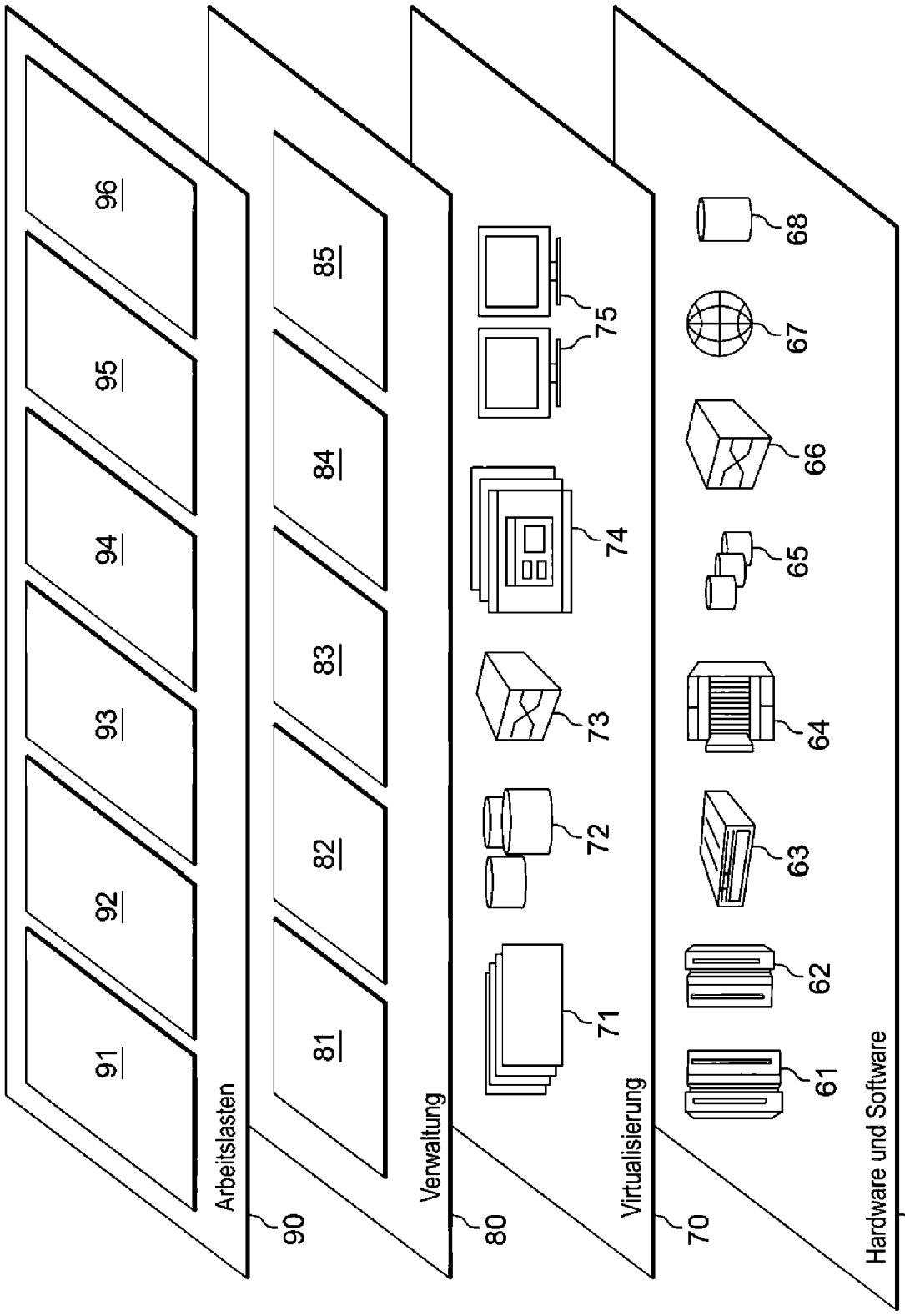


FIG. 11

