(19) Deutsches Patent- und Markenamt

(10) **DE 10 2020 209 254 A1** 2022.01.27

(12) # Offenlegungsschrift

(21) Aktenzeichen: **10 2020 209 254.0**
(22) Anmeldetag: **22.07.2020**
(43) Offenlegungstag: **27.01.2022**

(51) Int Cl.: ***G06N 3/06*** (2006.01)
G06V 30/19 (2022.01)

(71) Anmelder:
OFFIS e.V., 26121 Oldenburg, DE; Universität Duisburg-Essen, 47057 Duisburg, DE

(74) Vertreter:
Fink Numrich Patentanwälte PartmbB, 80634 München, DE

(72) Erfinder:
Helms, Domenik, 26655 Westerstede, DE; Schiele, Gregor, 40477 Düsseldorf, DE; Könnecke, Moritz, 26655 Westerstede, DE

(56) Ermittelter Stand der Technik:

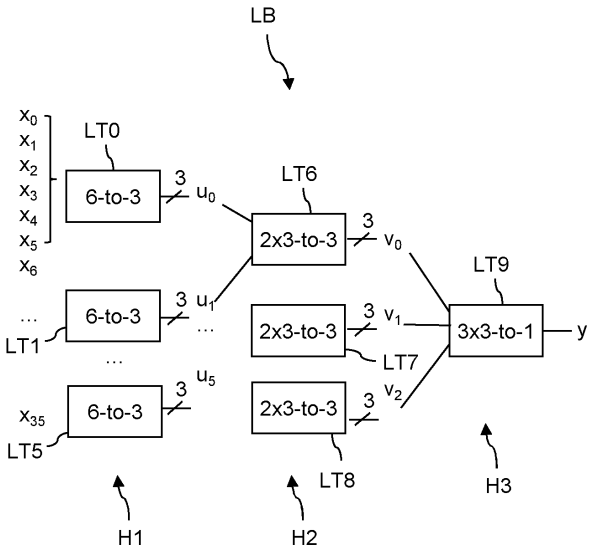
FARAONE, Julian, et al.: Addnet: Deep neural networks using fpga-optimized multipliers. In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 28, 2019, No. 1, 115-128. https://doi.org/10.1109/TVLSI.2019.2939429 [abgerufen am 31.03.2021]

Prüfungsantrag gemäß § 44 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **Vorrichtung zur rechnergestützten Klassifikation von Sensordaten unter Verwendung einer trainierten neuronalen Netzstruktur**

(57) Zusammenfassung: Die Erfindung betrifft einer Vorrichtung zur rechnergestützten Klassifikation von Sensordaten (SD) unter Verwendung einer trainierten neuronalen Netzstruktur (NN), die ein oder mehrere neuronale Netze umfasst und eine Vielzahl von Schichten (CL1, PL1, .., CLn, PLn, FC1, ..., FCn) aus künstlichen Neuronen (NE) enthält. Die neuronale Netzstruktur (NN) ist auf einem FPGA (100) ausgebildet, der derart programmiert ist, dass Lookup-Tabellen (LTO, LT1, ..., LT9) definiert sind, welche jeweils aus einer oder mehreren vorgegebenen Lookup-Basiszellen (BZ) des FPGA (100) gebildet werden und aus mehreren Eingangssignalen (x_0, x_1, ..., v_2) ein Ausgangssignal (uo, u_1, ..., y) erzeugen. Das oder die Neuronen (NE) zumindest eines Teils der Neuronen (NE) sind spezifische Neuronen (NE), die jeweils einen Lookup-Block (LB) bilden, wobei der Lookup-Block (LB) ein Baum aus mehreren Lookup-Tabellen (LTO, LT1, ..., LT9) in mehreren Hierarchieebenen (H1, H2, H3) ist.



Beschreibung

[0001] Die Erfindung betrifft eine Vorrichtung zur rechnergestützten Klassifikation von Sensordaten unter Verwendung einer trainierten neuronalen Netzstruktur.

[0002] Künstliche neuronale Netze werden heutzutage in einer Vielzahl von Anwendungsbereichen eingesetzt. Häufig dienen neuronale Netze dazu, die Messungen von Sensoren auszuwerten und die entsprechenden Sensordaten geeignet zu klassifizieren, beispielsweise um aus den Sensordaten Fehlerbilder in einer industriellen Produktionsanlage abzuleiten. In vielen Fällen besteht der Bedarf an einer Umsetzung eines neuronalen Netzes, das Sensordaten mit geringer Latenz und geringem Bedarf an elektrischer Energie klassifiziert.

[0003] Im Stand der Technik werden Ansätze beschrieben, die neuronale Netze unter Verwendung von Logikblöcken in FPGAs (FPGA = Field Programmable Gate Array) realisieren (siehe beispielsweise US 2019/0171420 A1).

[0004] In der Druckschrift Erwei Wang et al., „LUTNet: Learning FPGA Configurations for Highly Efficient Neural Network Interference“, IEEE Transactions On Computers, 2. März 2020, wird ein neuronales Netz beschrieben, das über Lookup-Tabellen auf einem FPGA realisiert ist. Für die Gewichte sowie Ein- und Ausgänge der Neuronen sind nur binäre Werte zugelassen, was die Genauigkeit der mit dem neuronalen Netz durchgeführten Berechnungen einschränkt. Ferner können solche Netze nicht mit gängigen Methoden, wie z.B. Backpropagation, trainiert werden.

[0005] Aufgabe der Erfindung ist es, eine Vorrichtung zur rechnergestützten Klassifikation von Sensordaten unter Verwendung einer trainierten neuronalen Netzstruktur zu schaffen, die genaue Klassifikationsergebnisse mit geringem Rechenaufwand und geringem Energieverbrauch liefert.

[0006] Diese Aufgabe wird durch die Vorrichtung gemäß Patentanspruch 1 gelöst. Weiterbildungen der Erfindung sind in den abhängigen Ansprüchen definiert.

[0007] Die erfindungsgemäße Vorrichtung dient zur rechnergestützten Klassifikation von Sensordaten unter Verwendung einer trainierten neuronalen Netzstruktur, die ein oder mehrere neuronale Netze umfasst und eine Vielzahl von Schichten aus künstlichen Neuronen enthält. Jedes Neuron umfasst eine Vielzahl von Eingängen und einen Ausgang, um ein Signal am Ausgang in Abhängigkeit von jeweiligen Signalen an der Vielzahl von Eingängen zu erzeugen. In der nachfolgenden Beschreibung der erfindungsgemäßen Vorrichtung sind Merkmale, die Verfahrensschritte enthalten, immer dahingehend zu verstehen, dass die Vorrichtung derart ausgestaltet ist, dass der entsprechende Verfahrensschritt im Betrieb der Vorrichtung ausgeführt wird. Hier und im Folgenden bezieht sich der Begriff des Signals bzw. des Sensorsignals bzw. des Eingangssignals bzw. des Ausgangssignals immer auf ein digitales Signal.

[0008] Die neuronale Netzstruktur der erfindungsgemäßen Vorrichtung umfasst eine Eingangsschicht zur Zufuhr von Sensorsignalen, wobei die Sensorsignale auf Sensordaten in der Form von Messungen einer oder mehreren Sensoren basieren. Vorzugsweise resultieren die Sensorsignale aus einer Vorverarbeitung der ursprünglichen Sensordaten. Die neuronale Netzstruktur umfasst ferner eine Ausgangsschicht, über deren Ausgangssignale die der Eingangsschicht zugeführten Sensorsignale klassifiziert werden, indem diese Sensorsignale zumindest einer Klasse aus einer Mehrzahl von Klassen zugeordnet werden.

[0009] Die neuronale Netzstruktur ist auf einem FPGA ausgebildet, das auch als programmierbares Logikgatter bezeichnet wird. In an sich bekannter Weise enthält das FPGA ein Feld aus vorgegebenen (physikalisch vorhandenen) Lookup-Basiszellen, die jeweils aus Signalen aus mehreren 1-Bit-Eingängen ein Signal an einem 1-Bit-Ausgang erzeugen. Das FPGA ist derart programmiert, dass Lookup-Tabellen definiert sind, welche jeweils aus einer oder mehreren Lookup-Basiszellen des FPGA gebildet werden und aus mehreren Eingangssignalen ein Ausgangssignal erzeugen. Die Eingangssignale und Ausgangssignale können dabei auch Mehrbit-Signale sein.

[0010] Erfindungsgemäß sind das oder die Neuronen zumindest eines Teils der Neuronen spezifische Neuronen, die jeweils einen Lookup-Block bilden, der das jeweilige spezifische Neuron (d.h. dessen Parameter) modelliert. Dabei wird der entsprechende Lookup-Block durch eine neuartige baumartige Struktur aus mehreren Lookup-Tabellen gebildet. In dieser baumartigen Struktur sind die Lookup-Tabellen in mehreren Hierarchieebenen angeordnet, die durch die folgenden Merkmale charakterisiert sind.

[0011] Die niedrigste Hierarchieebene besteht aus mehreren Lookup-Tabellen, deren Eingangssignale den Signalen an den Eingängen des jeweiligen spezifischen Neurons entsprechen. Den Lookup-Tabellen einer jeweiligen Hierarchieebene, die eine nächsthöhere Hierarchieebene zu einer benachbarten Hierarchieebene ist (d.h. alle Hierarchieebenen außer der niedrigsten), werden ausschließlich digitale Ausgangssignale der Lookup-Tabellen der benachbarten Hierarchieebene (d.h. keine anderen Signale als diese Ausgangssignale) zugeführt, wobei in jeder Lookup-Tabelle in der jeweiligen Hierarchieebene mehrere Ausgangssignale von Lookup-Tabellen der benachbarten Hierarchieebene kombiniert werden. Aufgrund der Kombination der Ausgangssignale nimmt die Anzahl der Lookup-Tabellen von einer Hierarchieebene zu der nächsthöheren Hierarchieebene ab. Ferner besteht die höchste Hierarchieebene aus einer einzelnen Lookup-Tabelle, deren Ausgangssignal dem Signal am Ausgang des jeweiligen spezifischen Neurons entspricht.

[0012] Die erfindungsgemäße Vorrichtung weist den Vorteil auf, dass durch die baumartige Verschaltung von Lookup-Tabellen auch Neuronen mit einer großen Anzahl von Eingängen auf einem FPGA ohne die Einschränkung auf binäre Gewichte und Ein- und Ausgänge realisiert werden können. Gleichzeitig wird durch die Verwendung von Lookup-Tabellen, die ohne aufwändige Berechnungen lediglich durch Auslesen eines Speichers Eingangssignale auf ein Ausgangssignal abbilden, ein energieeffizienter Betrieb der Vorrichtung ermöglicht. In der detaillierten Beschreibung wird dargelegt, wie die oben definierte neuronale Netzstruktur geeignet trainiert werden kann.

[0013] Je nach verwendetem FPGA können die vorgegebenen Lookup-Basiszellen in der erfindungsgemäßen Vorrichtung unterschiedliche Größen aufweisen. In einer Variante sind die vorgegebenen Lookup-Basiszellen jeweils 32-Bit-Basiszellen mit fünf 1-Bit-Eingängen. Ebenso kann ein FPGA verwendet werden, in dem die vorgegebenen Lookup-Basiszellen jeweils 64-Bit-Basiszellen mit sechs 1-Bit-Eingängen sind.

[0014] In einer weiteren bevorzugten Variante werden in einem jeweiligen spezifischen Neuron zwischen benachbarten Hierarchieebenen Signale mit einer Bitbreite von mindestens zwei Bit, vorzugsweise von mindestens drei Bit, übertragen. Hierdurch wird die Genauigkeit der Klassifikation verbessert.

[0015] In einer weiteren, besonders bevorzugten Ausführungsform liegen das oder die spezifischen Neuronen, welche die oben definierte baumartige Struktur aufweisen, in einem oder mehreren sog. Convolutional Layers eines Convolutional Neural Networks, das häufig auch als CNN bezeichnet wird und eine Anzahl von Convolutional Layers (d.h. zumindest einen Convolutional Layer) umfasst. Vorzugsweise sind alle Neuronen in den jeweiligen Convolutional Layers die oben definierten spezifischen Neuronen. Convolutional Neural Networks mit entsprechenden Convolutional Layers sind dem Fachmann dabei an sich bekannt. Die Neuronen in einem solchen Netz führen eine Faltung auf den Eingangssignalen durch, um hierdurch spezifische Merkmale in den Eingangssignalen zu detektieren. Optional können sich in dem Convolutional Neural Network an die entsprechenden Convolutional Layers jeweilige Pooling-Layers anschließend, die ebenfalls bekannt sind und die Datenmenge reduzieren. Convolutional Neural Networks kommen häufig bei der Verarbeitung von Bilddaten zum Einsatz. Ein Convolutional Neural Network kann im Rahmen der Erfindung allgemein zur Verarbeitung von Sensordaten genutzt werden.

[0016] In einer bevorzugten Variante der soeben beschriebenen Ausführungsform enthält das Convolutional Neural Network die Eingangsschicht der neuronalen Netzstruktur, wobei die Eingangsschicht ein Convolutional Layer ist, dessen Neuronen spezifische Neuronen sind, wobei sich an die Anzahl von Convolutional Layers ein Fully Connected Neural Network aus einem oder mehreren Fully Connected Layers anschließt, das die Ausgangsschicht der neuronalen Netzstruktur enthält. Eine konkrete Ausgestaltung dieser Architektur wird in der detaillierten Beschreibung anhand von EKG-Signalen erläutert. In einer bevorzugten Variante sind nur die Neuronen in den Convolutional Layers als spezifische Neuronen in der Form von baumartig verschalteten Lookup-Tabellen realisiert.

[0017] In einer weiteren bevorzugten Variante ist das Fully Connected Neural Network in einem oder mehreren DSP-Slices des FPGA implementiert (DSP = Digital Signal Processing). Die Integration solcher Slices in FPGAs ist an sich bekannt, und die Slices fungieren als digitale Signalprozessoren, die eigenständig Berechnungen durchführen können. Bei der Realisierung des Fully Connected Neural Networks über DSP-Slices werden dessen Neuronen vorzugsweise auf herkömmliche Weise und nicht basierend auf der baumartigen Verschaltung von Lookup-Tabellen realisiert.

[0018] In einer weiteren bevorzugten Ausführungsform ist die erfindungsgemäße Vorrichtung derart ausgestaltet, dass die Sensordaten vorverarbeitet werden, wobei die vorverarbeiteten Sensordaten die Sensorsig-

nale darstellen, die der Eingangsschicht der neuronalen Netzstruktur zugeführt werden. Die Vorverarbeitung kann beispielsweise eine Dynamic Range Compression und/oder eine Quantisierung umfassen.

[0019] In einer weiteren, besonders bevorzugten Ausführungsform repräsentieren die Sensordaten eine oder mehrere Zeitreihen der Messungen des oder der Sensoren, wobei die Sensorsignale, die der Eingangsschicht zugeführt werden, zu einem oder mehreren aufeinander folgenden und ggf. überlappenden Zeitabschnitten der Zeitreihe oder Zeitreihen gehören. Vorzugsweise wird diese Ausführungsform mit der obigen Ausführungsform eines Convolutional Neural Networks kombiniert.

[0020] In einer besonders bevorzugten Variante der soeben beschriebenen Ausführungsform umfassen die Sensordaten eine Anzahl von EKG-Signalen, wobei die EKG-Signale jeweils eine Zeitreihe einer elektrischen Potentialdifferenz eines am menschlichen oder tierischen Körper gemessenen Elektrokardiogramms repräsentieren. Die Sensorsignale, die der Eingangsschicht zugeführt werden, gehören dabei zu einem oder mehreren aufeinander folgenden Zeitabschnitten des oder der EKG-Signale, wobei die Sensorsignale dahingehend klassifiziert werden, ob und/oder welcher oder welche Krankheitszustände für das Herz vorliegen, von dem die EKG-Signale stammen. Durch die erfindungsgemäße Realisierung der neuronalen Netzstruktur auf einem FPGA kann dabei mit wenig Energieverbrauch eine Herzaktivität überwacht werden.

[0021] Die erfindungsgemäße Vorrichtung kann auch zur Klassifikation von beliebigen anderen Sensordaten als EKG-Signalen eingesetzt werden. In einer weiteren bevorzugten Variante stammen die Sensordaten aus einem Herstellungsprozess, in dem ein beliebiges Produkt produziert wird. Dabei werden die Sensorsignale dahingehend klassifiziert, ob und/oder welche Fehlerbilder im Herstellungsprozess auftreten.

[0022] In einer bevorzugten Variante der obigen Ausführungsform, die vorzugsweise mit einem Convolutional Neural Network realisiert wird, sind die Sensordaten Dickemessungen, die eine zeitliche Entwicklung einer Dicke einer Ummantelung eines Kabels und insbesondere eines elektrischen Kabels an einer oder mehreren vorgegebenen Messstellen während der Aufbringung der Ummantelung auf das Kabel repräsentieren, wobei die Sensorsignale dahingehend klassifiziert werden, ob und/oder welches oder welche Fehlerbilder bei der Aufbringung der Ummantelung des Kabels auftreten.

[0023] Ausführungsbeispiele der Erfindung werden nachfolgend anhand der beigefügten Figuren detailliert beschrieben.

[0024] Es zeigen:

Fig. 1 eine Darstellung, die die Funktionsweise eines konventionellen Neurons in einem neuronalen Netz verdeutlicht;

Fig. 2 ein Beispiel einer Verschaltung von Basiszellen eines FPGA zu einer Lookup-Tabelle;

Fig. 3 ein Beispiel eines Lookup-Blocks, der in einer Ausführungsform der Erfindung zur Implementierung eines Neurons auf einem FPGA genutzt wird;

Fig. 4 den Aufbau der im Lookup-Block der **Fig. 3** verwendeten Lookup-Tabellen basierend auf der Verschaltung entsprechender Basiszellen;

Fig. 5 eine Darstellung eines Neurons mit quantisierter Aktivierungsfunktion, die im Rahmen des Trainings einer Variante des erfindungsgemäßen neuronalen Netzes verwendet wird;

Fig. 6 eine schematische Darstellung einer Variante der erfindungsgemäßen Vorrichtung zur Klassifikation von EKG-Signalen unter Verwendung eines Convolutional Neural Networks; und

Fig. 7 eine Darstellung, die den Aufbau eines jeweiligen Neurons in einem Convolutional Layer des Convolutional Neural Networks aus **Fig. 6** verdeutlicht.

[0025] **Fig. 1** zeigt den Aufbau eines konventionellen Neurons in einem neuronalen Netz. Das Verhalten eines solchen Neurons wird mit einem weiter unten näher beschriebenen Lookup-Block eines FPGA approximativ modelliert. Das Neuron ist als Kreis dargestellt und mit Bezugszeichen NE bezeichnet. Es umfasst eine Vielzahl von Eingängen IN, wobei in dem Beispiel der **Fig. 1** sechs Eingänge angedeutet sind. An den Eingängen liegen entsprechende Eingangssignale x_0, x_1, ..., x_5 an. Die Eingangssignale stammen von Neuronen einer tieferen Schicht bzw. sind Eingangssignale des neuronalen Netzes, sofern das Neuron in der Eingangsschicht des neuronalen Netzes vorgesehen ist. Jedes der Eingangssignale wird mit reelwertigen Gewichten w_0, w_1, ..., w_5 gewichtet und diese gewichteten Signale werden zusammen mit einem reelwertigen Bias b im Neuron NE summiert. Mittels einer sog. Aktivierungsfunktion AF wird dann aus dieser Summe an einem ent-

sprechenden Ausgang OU des Ausgangssignal y des Neurons generiert. Häufig wird als Aktivierungsfunktion AF die sog. Rectifier-Funktion genutzt, die den Maximalwert aus der gewichteten Summe und dem Wert Null angibt. Diese Rectifier-Funktion ist auch in **Fig. 1** angedeutet.

[0026] Mathematisch lässt sich die Ausgabe y des Neurons NE wie folgt beschreiben:

$$y\left(x_0, x_1, \ldots, x_{N-1}\right) = max\left(0, \sum_{i=0}^{N-1} w_i \cdot x_i + b\right) \qquad (1)$$

wobei N die Anzahl der Eingänge IN des Neurons NE repräsentiert

[0027] Aus dem Stand der Technik ist es unter anderem bekannt, die Gleichung (1) basierend auf binären Gewichten w_i sowie binären Eingangs- und Ausgangswerten x_i und y durch einfache Binäroperationen (XNOR) über Lookup-Tabellen auf einem FPGA umzusetzen. Ein großer Nachteil dieser Methode ist, dass die Gewichte binär sind, und sich daher kleine Änderungen der Gewichte (insbesondere während des Lernprozesses des neuronalen Netzes) nicht realisieren lassen.

[0028] Im Unterschied zum Stand der Technik sind die Neuronen des erfindungsgemäßen neuronalen Netzes nicht auf binäre Gewichte sowie binäre Eingangs- und Ausgangswerte beschränkt. Hierzu werden Lookup-Tabellen eines FPGA auf neuartige Weise miteinander verknüpft.

[0029] Wie bekannt, sind FPGAs hochkonfigurierbare Logikbausteine, die vom Benutzer auf elektronische Weise konfiguriert und so zu nahezu beliebigen Logikschaltungen (z.B. einem Prozessor) verknüpft werden können. Aktuelle FPGAs enthalten viele 10000 sog. Basiszellen, die in den Patentansprüchen als Lookup--Basiszellen bezeichnet werden. Jede Basiszelle stellt eine physikalisch vorhandene Einheit auf dem FPGA dar und hat eine vorbestimmte Anzahl von mehreren 1-Bit-Eingägen und gibt für entsprechende Bitkombinationen an diesen Eingängen einen Binärwert an einem 1-Bit-Ausgang aus. Durch die Basiszelle wird somit eine Lookup-Tabelle realisiert, welche jeder Bitkombination an den Eingängen einen vorgegebenen Bitwert am Ausgang zuweist. Diese Zuweisung ist bei der Programmierung des FPGAs beliebig festlegbar. Je nach Ausgestaltung des FPGAs kann die Anzahl der 1-Bit-Eingänge an einer Basiszelle unterschiedlich festgelegt sein. Üblicherweise hat eine Basiszelle fünf Eingänge und damit 32 Speicherbits oder sechs Eingänge und damit 64 Speicherbits. Mehrere Basiszellen können im Rahmen der Programmierung des FPGA über frei konfigurierbare Verbindungsleitungen miteinander verknüpft werden. Im Folgenden wird ein FPGA mit 64-Bit-Basiszellen (d.h. sechs 1-Bit-Eingängen) betrachtet.

[0030] Es ist möglich, mehrere Basiszellen eines FPGA so miteinander zu verschalten, dass diese wie eine binäre Tabelle mit größerer oder kleinerer Eingangsanzahl funktioniert.

[0031] Beispielhaft ist dies in **Fig. 2** dargestellt, welche die Verschaltung von fünf 64-Bit-Basiszellen BZ_0, BZ_1, BZ_2, BZ_3 und BZ_4 wiedergibt. Die 1-Bit-Eingangswerte x_0 bis x_5 werden gleichzeitig den Basiszellen BZ_0 bis BZ_3 zugeführt, die insgesamt einen 4-Bit-Wert liefern, welcher der Basiszelle BZ_4 zugeführt wird. In Kombination mit den zwei weiteren 1-Bit-Werten x_6 und x_7, die ebenfalls der Basiszelle BZ_4 zugeführt werden, wird dann durch die Basiszelle BZ_4 ein Binärwert y am Ausgang generiert. Es entsteht somit durch diese Verschaltung eine binäre Tabelle mit 8 1-Bit-Eingängen und einem 1-Bit-Ausgang.

[0032] Im Folgenden wird eine binäre Tabelle mit n 1-Bit-Eingängen und einem 1-Bit-Ausgang als n-Block bezeichnet. Ein solcher n-Block stellt eine Ausführungsform einer Lookup-Tabelle im Sinne der Patentansprüche dar. Aus praktischen Gründen ist n limitiert, da der Hardwareaufwand exponentiell mit n steigt. Für einen n-Block benötigt man über $2^{(n-6)}$ Basiszellen, n ist also auf Werte deutlich unter 20 beschränkt.

[0033] Fasst man mehrere n-Blöcke zusammen, die die gleichen Eingangswerte zugeführt bekommen, aber unterschiedliche Werte in den elementaren Basiszellen gespeichert haben, kann man damit Mehr-Bit-Ausgangswerte erzeugen. Werden m dieser n-Blöcke kombiniert, erhält man einen Logikblock, der im Folgenden als n-to-m-Block bezeichnet wird und ebenfalls eine Ausführungsform einer Lookup-Tabelle im Sinne der Patentansprüche darstellt. Beispielsweise kann ein 6-to-4-Block für jede der 2^6=64 möglichen Kombinationen seiner n=6 Eingänge einen Wert aus m=4 Bits (0 bis 15) am Ausgang erzeugen.

[0034] Es ist nun möglich, die n Eingänge von n-to-m Blöcken nicht als n unabhängige binäre Werte aufzufassen, sondern jeweils b von diesen als eine b-Bit-Zahl zu interpretieren. Dies hat keinen Einfluss auf die Umsetzung in der Hardware. Ein 6-to-2-Block kann entweder für sechs 1-Bit-Eingänge oder für drei 2-Bit-Ein-

gänge bzw. für zwei 3-Bit-Eingänge je einen 2-Bit-Ausgang erzeugen. Im Folgenden werden Blöcke, die aus n b-Bit-Eingängen einen m-Bit-Ausgang generieren als nxb-to-m-Blöcke bezeichnet.

[0035] Auch diese Blöcke sind Varianten von Lookup-Tabellen im Sinne der Patentansprüche.

[0036] Erfindungsgemäß wird ein jeweiliges Neuron durch eine baumartige Verschaltung der oben beschriebenen Lookup-Tabellen auf einem FPGA realisiert, wie weiter unten noch näher beschrieben wird. Das Ausgangssignal y einer jeweiligen Lookup-Tabelle wird dabei in Abhängigkeit von den Eingangssignalen X_i wie folgt modelliert:

$$y(x_i) = max\left(0, min\left(2^{B_y} - 1, \left\lfloor \sum_{i=0}^{N-1} w_i \cdot x_i + b \right\rfloor\right)\right) \qquad (2)$$

wobei die Gaussklammer [x] das Abrunden auf ganze Werte beschreibt.

[0037] Der Ausgang y einer jeweiligen Lookup-Tabelle weist dabei die Bitbreite B_y auf und die Eingänge x_i haben die Bitbreite B_x. Es sind somit beliebige Quantisierungen für die Eingangs- und Ausgangssignale möglich. Ferner sind die Größen w_Z und b reelwertig, d.h. sie werden beim Trainieren der Neuronen als Fließkommazahlen realisiert. Die Lookup-Tabellen bilden somit alle $x_i \in \left[0, 2^{B_x} - 1\right]$ auf $y_i \in \left[0, 2^{B_y} - 1\right]$ ab. Da w_i und b reelwertig bleiben können, kann ein Lernen per Backpropagation durchgeführt werden.

[0038] Die erfindungsgemäße baumartige Verschaltung der Lookup-Tabellen lässt sich allgemein als ein Baum aus mehreren Hierarchieebenen beschreiben, der einem Lookup-Block und damit einem spezifischen Neuron im Sinne der Patentansprüche entspricht. Dabei besteht die niedrigste Hierarchieebene aus mehreren Lookup-Tabellen, die entsprechende Blattknoten des Baums bilden und deren digitale Eingangssignale den Signalen an den Eingängen des jeweiligen spezifischen Neurons entsprechen. Ferner werden den Lookup-Tabellen einer jeweiligen Hierarchieebene, die eine nächsthöhere Hierarchieebene zu einer benachbarten Hierarchieebene ist, ausschließlich digitale Ausgangssignale der Lookup-Tabellen der benachbarten Hierarchieebene zugeführt, wobei in jeder Lookup-Tabelle der jeweiligen Hierarchieebene mehrere Ausgangssignale von Lookup-Tabellen der benachbarten Hierarchieebene kombiniert werden.

[0039] Darüber hinaus besteht die höchste Hierarchieebene aus einer einzelnen Lookup-Tabelle, die einen Wurzelknoten in dem Baum darstellt und deren digitales Ausgangssignal dem Signal am Ausgang des jeweiligen spezifischen Neurons entspricht.

[0040] **Fig. 3** zeigt eine konkrete Realisierung eines spezifischen Neurons in der Form eines Lookup-Blocks LB gemäß der obigen Definition. Wie man aus **Fig. 3** erkennt, besteht der Lookup-Block aus drei Hierarchieebenen H1, H2 und H3. Die Hierarchieebene H1 ist die niedrigste Hierarchieebene, die Hierarchieebene H2 eine Zwischenebene und die Hierarchieebene H3 die höchste Hierarchieebene. Die Hierarchieebene H1 enthält insgesamt sechs Lookup-Tabellen LT0, LT1, ..., LT5, wobei nur drei dieser Tabellen wiedergegeben sind. Jede dieser Lookup-Tabellen ist ein 6-to-3-Block, der sechs Ein-Bit-Signale x_i erhält, so dass den sechs 6-to-3-Blöcken insgesamt 36 1-Bit-Signale x_0 bis x_{35} zugeführt werden.

[0041] Die einzelnen Lookup-Blöcke LT0 bis LT5 erzeugen ein 3-Bit-Ausgangssignal. Mit anderen Worten erzeugt die Lookup-Tabelle LT0 das 3-Bit-Ausgangssignal u_0, die Lookup-Tabelle LT1 das Drei-Bit-Ausgangssignal u_1, die Lookup-Tabelle LT2 das 3-Bit-Ausgangssignal u_2, die Lookup-Tabelle LT3 das 3-Bit-Ausgangssignal u_3, die Lookup-Tabelle LT4 das 3-Bit-Ausgangssignal u_4 und die Lookup-Tabelle LT5 das 3-Bit-Ausgangssignal u_5. Jeweils zwei dieser benachbarten Ausgangssignale werden dann Lookup-Tabellen LT6, LT7 und LT8 in der Hierarchieebene H2 zugeführt. Die einzelnen Lookup-Tabellen LT6 bis LT8 entsprechen dabei jeweiligen 2x3-to-3-Blöcken. Konkret werden der Lookup-Tabelle LT6 die beiden Ausgangssignale u_0 und u_1, der Lookup-Tabelle LT7 die beiden Ausgangssignale u_2 und u_3 sowie der Lookup-Tabelle LT8 die beiden Ausgangssignale u_4 und u_5 zugeführt. Die Lookup-Tabelle LT6 erzeugt hieraus das 3-Bit-Ausgangssignal vo, die Lookup-Tabelle LT7 das 3-Bit-Ausgangssignal v_1 und die Lookup-Tabelle LT8 das 3-Bit-Ausgangssignal v_2. Diese drei Ausgangssignale v_0 bis v_2 werden anschließend der höchsten Hierarchieebene H3 zugeführt, die aus der einzelnen Lookup-Tabelle LT9 in der Form eines 3x3-to-1-Blocks besteht. Diese Lookup-Tabelle erzeugt den 1-Bit-Wert y.

[0042] Wie bereits oben erwähnt, werden die einzelnen Lookup-Tabellen durch eine geeignete Verschaltung von Basiszellen realisiert. Die entsprechende Verschaltung der Basiszellen für die in **Fig. 3** dargestellten

6-to-3-Blöcke bzw. 2x3-to-3- Blöcke bzw. den 3x3-to-1-Block ist in **Fig. 4** wiedergegeben. Das entsprechende FPGA enthält dabei 64-Bit-Basiszellen, die in **Fig. 4** allgemein mit BZ bezeichnet sind. Wie man erkennt, ist ein 6-to-3-Block durch drei Basiszellen realisiert, die parallel die gleichen 6-Bit-Werte erhalten und einen 3-Bit-Wert ausgeben. Demgegenüber ist ein 2x3-to-3-Block durch drei Basiszellen umgesetzt, die jeweils die gleichen Eingangswerte in der Form von zwei 3-Bit-Werten erhalten und auf diese Weise einen Drei-Bit-Wert generieren. Ferner ist ein 3x3-to-1-Block durch insgesamt elf Basiszellen realisiert. Acht links angeordnete Basiszellen erhalten dabei die gleichen Eingangssignale in der Form von zwei 3-Bit-Werten. Hierdurch entstehen zwei 4-Bit-Werte, die separat in zwei mittlere Basiszellen eingespeist werden, die zusätzlich die gleichen zwei Bits eines 3-Bit-Werts erhalten. Als Ausgang erzeugen die beiden mittleren Basiszellen zwei 1-Bit--Werte, die einer rechts angeordneten Basiszelle neben dem dritten Bit des 3-Bit-Werts zugeführt werden, dessen ersten beiden Bits den mittleren Basiszellen zugeführt wurden. Hieraus erzeugt die rechte Basiszelle einen 1-Bit-Wert. Die rechte Basiszelle hat dabei noch weitere Eingänge, die jedoch nicht genutzt werden.

[0043] Im Folgenden wird konkret für den Baum der **Fig. 3** die effektive Ausgabefunktion des hierdurch realisierten spezifischen Neurons angegeben. Für andere Zahlen von Eingängen x, für andere innere Bitbreiten von u und v und für andere Hardwarevorgaben ergeben sich andere Bäume, die zu entsprechenden Ausgaben führen.

[0044] Entsprechend der Gleichung (2) lauten die u_j (j = 0, 1, ..., 5) in **Fig. 3** wie folgt:

$$u_j\left(x_{6\cdot j},\ldots,x_{6\cdot j+5}\right)=max\left(0,min\left(7,\left\lfloor\sum_{i=0}^{5}w_{j,i}\cdot x_{i+6\cdot j}+b_j\right\rfloor\right)\right) \qquad (3)$$

[0045] Die v_k(k = 0, 1, 2) ergeben sich dann mit Gewichten mk,h und Bias c_k (um die Verwechslung mit den Gewichten $w_{j,i}$ und dem Bias b_j der Hierarchieebene H1 zu vermeiden):

$$v_k\left(u_{2\cdot k},u_{2\cdot k+1}\right)=max\left(0,min\left(7,\left\lfloor\sum_{h=0}^{1}m_{k,h}\cdot u_{h+2\cdot k}+c_k\right\rfloor\right)\right) \qquad (4)$$

[0046] Gleichung (3) in (4) eingesetzt ergibt dann für ein v_k:

$$v_k\left(u_{2\cdot k},u_{2\cdot k+1}\right)=v_k\left(x_{12\cdot k},\ldots,x_{12\cdot k+11}\right)=$$
$$=max\left(0,min\left(7,\left\lfloor\sum_{h=0}^{1}m_{k,h}\cdot max\left(0,min\left(7,\left\lfloor\sum_{i=0}^{5}w_{h+2\cdot k,i}\cdot x_{i+6\cdot(h+2\cdot k)}+b_{(h+2\cdot k)}\right\rfloor\right)\right)+c_k\right\rfloor\right)\right) \qquad (5)$$

[0047] Für eine Verschaltung von n-to-m-Blöcken in lediglich zwei Hierarchieebenen würde Gleichung (5) mit k=0 bereits der Ausgabefunktion entsprechen. Im hier betrachteten Beispiel gibt es noch eine dritte Hierarchieebene mit Gewichten n_g und Bias d:

$$y\left(v_0,\ldots,v_2\right)=max\left(0,min\left(1,\left\lfloor\sum_{g=0}^{2}n_g\cdot v_g+d\right\rfloor\right)\right) \qquad (6)$$

[0048] Durch Einsetzen von Gleichung (5) in Gleichung (6) ergibt sich hieraus:

$$y\left(x_0,\ldots,x_{35}\right)=max\left(0,min\left(1,\left\lfloor\sum_{g=0}^{2}n_g\cdot max\left(0,min\left(7,\left\lfloor\sum_{h=0}^{1}m_{g,h}\cdot max\left(0,min\left(7,\left\lfloor\sum_{i=0}^{5}w_{h+2\cdot g,i}\cdot x_{i+6\cdot(h+2\cdot g)}+b_{(h+2\cdot g)}\right\rfloor\right)\right)+c_g\right\rfloor\right)\right)+d\right\rfloor\right)\right) \qquad (7)$$

[0049] Gleichungen (3), (5) und (7) verhalten sich in der Anwendung in Neuronalen Netzen ähnlich; die doppelte bzw. dreifache min-Funktion wird für immer größere innere Bitbreite weniger relevant und die Parameter n_g, $m_{k,h}$ und $w_{i,j}$ können während des Trainings bestimmt werden. Als einzige entscheidende (den Unterschied zwischen (3), (5) und (7) definierende) Änderung bleibt die doppelte/dreifache max-Funktion: Es wird für Teilsummen bereits aufgerundet, was zu einem leicht veränderten Ergebnis führt.

[0050] Auch hier ist es hilfreich, dass das Netz so wie es ist trainiert werden kann. Im Training werden die Lookup-Tabellen in den einzelnen Hierarchieebenen wie unabhängige Schichten betrachtet und können so innerhalb ihrer Einschränkungen so weit wie möglich optimiert werden.

[0051] Im Folgenden wird beschrieben, wie ein neuronales Netzes mit Neuronen, die entsprechend den obigen Erläuterungen aus einer baumartigen Verschaltung von Lookup-Tabellen bestehen, gelernt bzw. trainiert werden kann. Zunächst wird ein reguläres, uneingeschränktes (unquantisiertes) Netz basierend auf konventionellen Neuronen entsprechend Gleichung (1) erzeugt und dessen Metaparameter mittels Backpropagation solange optimiert, bis das Netz die gewünschte Genauigkeit erreicht hat. In der Aktivierungsfunktion der jeweiligen Neuronen wird dann zunächst eine grobe Quantisierung (z.B. auf 4 Bit) eingeführt und das Training wiederholt, um die Parameter auf das durch die Quantisierung eingeführte Rauschen anzupassen. Beim wiederholten Training wird eine Erweiterung einer normalen Backpropagation-Trainingsmethode genutzt, bei der Quantisierungsartefakte (als Teil der Aktivierungsfunktion) regulär dargestellt werden können. Für die Backpropagation werden die durch die Quantisierung erzeugten Unstetigkeiten ignoriert und nur die zusätzliche Sättigung beim maximalen Wert des Ausgangssignals berücksichtigt, wodurch ein herkömmlicher, hocheffizienter Lernprozess genutzt werden kann. Dies wird nochmals in **Fig. 5** verdeutlicht, die ein Neuron mit einer quantisierten Aktivierungsfunktion AF zeigt, die im Rahmen des Trainings mittels Backpropagation genutzt wird. Der Aufbau des Neurons unterscheidet sich vom Aufbau des Neurons aus **Fig. 1** lediglich darin, dass die Aktivierungsfunktion eine Quantisierung und Sättigung beschreibt.

[0052] Anschließend werden Neuronen mit vielen Eingängen durch die oben beschriebenen Lookup-Blocks mit wenigen Eingängen in mehreren Hierarchieebenen ersetzt. Am Übergang zwischen den Hierarchieebenen werden zunächst Werte mit einer Bitbreite von 3 Bit (optional auch mehr) gewählt. Danach wird für jedes Neuron einzeln entschieden, ob es noch gröber quantisiert werden kann. Hierzu können dem Fachmann geläufige Methoden genutzt werden. Beispielsweise wird eine Kostenfunktion : $L = 2^{B_x * N-6} \cdot B_y$ berechnet, welche die Zahl der benötigten Lookup-Tabellen in einem Neuron angibt, und diese dann mit dem durch die gröbere Quantisierung einhergehenden Genauigkeitsverlust verglichen. Entsprechende Vergleichskriterien, basierend auf denen dann entschieden wird, ob gröber quantisiert wird, liegen im Rahmen von fachmännischem Handeln.

[0053] Bisher wurde beschrieben, wie auf einem FPGA ganze Neuronen basierend auf einer baumartigen Verschaltung von Lookup-Blöcken realisiert werden können. Im Folgenden wird beispielhaft die Verwendung einer neuronalen Netzstruktur mit solchen Neuronen zur Überwachung von EKG-Signalen beschrieben, um Vorhofflimmern zu detektieren. Die hier betrachtete EKG-Sensorik, die elektrische Potentialdifferenzen zwischen Elektroden auf der Haut eines menschlichen Patienten misst, erfasst zwei Kanäle (d.h. zwei Potentialdifferenzen) und liefert somit zwei Messsignale in der Form von entsprechenden Zeitreihen jeweiliger Potentialdifferenzen. Jedes dieser Messsignale wird mit einer bestimmten Abtastfrequenz und Bitbreite abgetastet (konkret 500Hz, 16bit). Die Bitbreite ist hoch genug, um alle relevanten Artefakte im Signal korrekt abzubilden. Die Signale können als zwei Funktionen $f_1(t)$ und $f_2(t)$ interpretiert werden. Ein medizinischer Fachmann kann diesen Signalen das Auftreten von Vorhofflimmern entnehmen.

[0054] Aufgabe der hier beschriebenen Ausführungsform ist es, mittels eines neuronalen Netzes auf einem FPGA automatisch basierend auf entsprechenden Zeitspannen der beiden Messsignale (z.B. den letzten 1024 Werten (~2 Sekunden)) eine Signalklassifikation dahingehend vorzunehmen, ob Vorhofflimmern vorliegt oder nicht. Das Klassifikationsergebnis wird dabei durch den Vergleich von Ausgabewerten am Ausgang der neuronalen Netzstruktur repräsentiert.

[0055] In der hier beschriebenen Ausführungsform wird das zur Klassifikation verwendete neuronale Netz dadurch realisiert, dass ein Convolutional Neural Network, das zur Bilderkennung genutzt wird, auf eine Dimension reduziert wird. Die entsprechende Architektur, die eine Ausführungsform der erfindungsgemäßen Vorrichtung darstellt, ist in **Fig. 6** wiedergegeben.

[0056] Ausgangspunkt sind die oben beschriebenen Sensordaten SD in der Form von zwei EKG-Signalen, die als 16-Bit-Eingangsstrom der neuronalen Netzstruktur NN zugeführt werden. Die neuronale Netzstruktur ist dabei auf einem entsprechenden FPGA 100 realisiert, das lediglich schematisch durch eine gestrichelte Umrandung angedeutet ist. Die Sensordaten SD werden zunächst einer Vorverarbeitung unterzogen. In der hier beschriebenen Ausführungsform werden die beiden EKG-Signale einer an sich bekannten Dynamic Range Compression unterzogen, die in **Fig. 1** mit DRC bezeichnet ist. Dabei wird berücksichtigt, dass die EKG-Signale teils sehr schlecht ausgesteuert sind (dicker Patient, Schweiß, Position der Elektroden, usw.).

Die spätere Quantisierung auf geringe Bitbreiten benötigt jedoch die volle Ausnutzung des Wertebereichs, was durch die Dynamic Range Compression erreicht wird.

[0057] Nach der Dynamic Range Compression werden die 16-Bit-Signale in einem weiteren Vorverarbeitungsschritt einer Eingangsquantisierung IQ unterzogen. Mittels der Quantisierung werden die beiden 16-Bit--Messsignale auf jeweils 2-Bit-Signale SI abgebildet, welche Sensorsignale im Sinne der Patentansprüche darstellen. Gegebenenfalls kann auch eine weitere Vorverarbeitung der Sensordaten je nach Anforderung der Anwendung erfolgen (z.B. Tiefpassfilterung, Downsampling und dergleichen).

[0058] Nach der Vorverarbeitung werden die beiden 2-Bit-Signale SI in ein Schieberegister SR geschrieben, das sich einfach auf dem FPGA 100 realisieren lässt. In an sich bekannter Weise wird dem Schieberegister ein Wert eingeschoben und alle anderen Werte werden eine Position weiter geschoben, so dass das letzte Datum im Schieberegister entfällt.

[0059] An das Schieberegister SR schließt sich als Eingangsschicht des neuronalen Netzes ein an sich bekannter Convolutional Layer CL1 an. In dieser Schicht werden zwei Wertereihen mit 1024 aufeinander folgenden Werten der beiden EKG-Signale verarbeitet, was einer Zeitspanne von etwa 2 Sekunden entspricht. Dabei wird jeweils ein kurzer Zeitabschnitt von neun Werten (auch als Kernel-Größe bezeichnet) beider Wertereihen (also 18 Werte) parallel in zwölf Filterneuronen NE gegeben, die als Lookup-Blocks LB auf dem FPGA 100 entsprechend der obigen Beschreibung realisiert sind. Diese Filterneuronen führen in an sich bekannter Weise eine Faltung durch, um Signalformen zu erkennen. Die Faltung ist dabei für jedes Filterneuron unterschiedlich und wird durch unterschiedliche Gewichte der jeweiligen Filterneuronen realisiert. Aus Übersichtlichkeitsgründen sind in **Fig. 6** nur vier von den zwölf Filterneuronen wiedergegeben.

[0060] Anschließend wird der Zeitabschnitt um fünf Werte bzw. Messzeitpunkte (als Stride bezeichnet) verschoben, und die nächsten 18 Werte werden den gleichen zwölf Filterneuronen zugeführt. Für die betrachtete Zeitspanne von 1024 Werten erfolgt somit die erste Berechnung auf den Werten 1 bis 9, die zweite auf den Werten 6 bis 14 und die zweihundertvierte (und letzte) auf den Werten 1016 bis 1024. Der Ausgang der Schicht CL1 wird folglich durch 204 mal 12 Werte gebildet, wobei jeder dieser Werte angibt, wie sehr eine bestimmte elementare Signalform (z.B. eine fallende Flanke) erkannt wurde.

[0061] Das Schieberegister SR übernimmt die Speicherung von so vielen Werten, die entsprechend der Kernel-Größe der Schicht CL1 gleichzeitig verarbeitet werden, d.h. von $2 \cdot 9 = 18$ Werten je 2 Bit. Nach einem Berechnungsschritt schiebt das Schieberegister SR nicht um ein Bit weiter, sondern genau so weit, wie es einer Verschiebung um fünf Werte gemäß dem Stride entspricht. Im hier beschriebenen Beispiel werden somit für einen neuen Berechnungsschritt immer 20 neue Bits eingeschoben (Produkt aus fünf Werten von jeweils 2 Bit für zwei EKG-Signale).

[0062] Das für die Schicht CL1 beschriebene Verfahren wird im Prinzip für den darauf folgenden zweiten Convolutional Layer (nicht explizit in **Fig. 6** gezeigt) wiederholt, wobei nunmehr dieser zweiten Schicht anstatt von zwei EKG-Signalen die 204 Ausgabewerte der zwölf Filterneuronen unter Verwendung eines Schieberegisters SR' zugeführt werden, das weiter unten näher beschrieben wird. Auch dieser Convolutional Layer und alle weiteren Convolutional Layers enthalten Filterneutonen, die als Lookup-Blocks LB auf dem FPGA 100 entsprechend der obigen Beschreibung realisiert sind. Jeder einzelne Wert stellt dabei keine Amplitude mehr dar, sondern eine der 12 „Kategorien“, die in der Schicht CL1 durch die Filterneuronen erkannt wurden (z.B. „steigende Flanke“). Was genau diese Kategorien sind, braucht nicht vorgegeben werden, sondern wird beim Training von selbst bestimmt. Wenn beispielsweise der zweite Convolutional Layer acht Filterneuronen hat und jeweils fünf aufeinander folgende Kategorien (d.h. Kernel-Größe = 5) zusammenfasst, werden acht Neuronen benötigt, jedes davon mit $5 \cdot 12 = 60$ Eingängen. Bei einem Stride von 5 ergeben sich 8(Filter) · 41 (=204/Stride) Werte am Ausgang, jeder fasst die Kategorien der ersten Schicht CL1 zu „Metakategorien“ zusammen (z.B. „stark steigend, dann konstant, dann stark fallend“ = „Positiver Peak“).

[0063] Der darauf folgende dritte Convolutional Layer (nicht explizit in **Fig. 6** gezeigt) liest 41-mal die acht „Metakategorien“ ein und fasst davon jeweils z.B. sieben Stück zusammen, das Ganze für zehn Filterneuronen. Bei einem Stride von 7 ergeben sich $6 \cdot 10$ „Metametakategorien“ (z.B. „kleine Zacke, dann große Zacke, dann kleine negative Zacke“).

[0064] Die obige Vorgehensweise lässt sich beliebig weit fortsetzen. Bemerkenswert ist die Tatsache, dass jeder einzelne Wert in jedem Convolutional Layer eine immer größere Zeitbasis repräsentiert. Die Kategorien der ersten Schicht haben jeweils 9 Werte zusammengefasst, also Abschnitte von 18ms (9/500Hz) beschrie-

ben. Die zweite Schicht kombiniert bei einem Stride von 5 je fünf dieser Blöcke ohne Überlapp, daher beschreibt jeder Ausgangswert etwa 90ms. Die dritte Schicht hat dann bei einem Stride von 7 schon pro Wert einen Überblick über Abschnitte von 630 ms. Allgemein kann eine beliebige Anzahl von n Convolutional Layers CL1, ..., CLn verwendet werden, wie in **Fig. 6** angedeutet ist. Jedem Convolutional Layer ist dabei ein geeignetes Schieberegister vorgeschaltet.

[0065] Da im hier beschriebenen Ausführungsbeispiel eine Granularität von 630 ms ausreichend ist, werden die $6 \cdot 10$ Werte der dritten Schicht in eine Fully Connected Schicht eines regulären Fully Connected Neural Networks gegeben. Diese Schicht ist in **Fig. 3** mit FC1 bezeichnet. In einem Fully Connected Neural Network ist jedes Neuron einer Schicht mit allen Neuronen von benachbarten Schichten verbunden. Die Neuronen in dem Fully Connected Neural Network sind dabei nicht über Lookup-Blöcke realisiert. Das Fully Connected Neural Network wird weiter unten näher beschrieben.

[0066] Wie oben erwähnt, sind alle Neuronen der Convolutional Layers basierend auf entsprechenden Lookup-Blöcken in der Form einer baumartigen Struktur von Lookup-Tabellen in mehreren Hierarchieebenen realisiert. Beispielhaft ist in **Fig. 7** die Struktur eines entsprechenden Neurons NE bzw. Lookup-Blocks LB aus der Schicht CL1 wiedergegeben. Wie man erkennt, hat der Lookup-Block LB drei Hierarchieebenen H1, H2 und H3, ähnlich wie in der Variante der **Fig. 3**. Es werden nunmehr jedoch als Eingangssignale 18 2-Bit--Signale xo, x_1, ..., x_{17} zugeführt, die aus der Eingangsquantisierung IQ der **Fig. 6** stammen. In der Hierarchieebene H1 sind sechs Lookup-Tabellen LT0 bis LT5 vorgesehen, die jeweils drei benachbarte 2-Bit-Signale als Eingänge erhalten. Die Lookup-Tabellen der Hierarchieebene H1 sind als 3x2-to-3-Blocks ausgebildet, so dass 3-Bit-Ausgangswerte an den entsprechenden Ausgängen der Lookup-Tabellen erhalten werden, nämlich Ausgangswerte u_0, u_1, ..., u_5 für die entsprechenden Lookup-Tabellen LT0, LT1, ..., LT5.

[0067] Diese Ausgangswerte werden wiederum als Eingangswerte den drei Lookup-Tabellen LT6, LT7 und LT8 der zweiten Hierarchieebene H2 zugeführt, wobei jede Lookup-Tabelle dieser Hierarchieebene immer zwei benachbarte 3-Bit-Werte aus den Lookup-Tabellen der Hierarchieebene H1 erhält. Die Lookup-Tabellen LT6 bis LT8 sind als 2x3-to-3-Blöcke ausgestaltet, so dass als Ausgangssignale für die jeweiligen drei Lookup-Tabellen 3-Bit-Werte generiert werden, die in **Fig. 6** als v_0, v_1 und v_2 bezeichnet sind. Diese Werte werden schließlich der als 3x3-to-2-Block ausgebildeten Lookup-Tabelle LT9 der Hierarchieebene H3 zugeführt, die daraus den 2-Bit-Wert y erzeugt.

[0068] Es werden somit durch die zwölf Neuronen der Schicht CL1 zwölf 2-Bit-Werte ausgegeben, die dann in dem Schieberegister SR' zwischengespeichert werden. Genauso wie das Schieberegister SR sammelt das Schieberegister SR' genug Daten für den entsprechenden Verarbeitungsschritt im zweiten Convolutional Layer. Wie oben angedeutet, kann für den zweiten Convolutional Layer sowohl die Kernel-Größe als auch die Stride-Größe den Wert 5 annehmen. In diesem Fall nimmt das Schieberegister SR' je Aktivierung 24 Bit an, speichert $5 \cdot 24$ Bit und triggert die nachfolgende Schicht alle fünf Aktivierungen. Demzufolge ergibt sich am Eingang der zweiten Convolutional Schicht ein Signal von $60 \cdot 2$ Bit mit einer Frequenz von 20 Hz.

[0069] Optional schließt sich an jeden der Convolutional Layer CL1, ..., CLn ein an sich bekannter Pooling Layer an. Entsprechende Pooling Layer sind in **Fig. 6** beispielhaft durch Bezugszeichen PL1, ..., PLn angedeutet. Jeder Pooling Layer reduziert mit einer herkömmlichen Methodik die Datenmenge am Ausgang des vorgeschalteten Convolutional Layers. Dies geschieht durch einen Addierer, welcher dem hinter dem entsprechenden Convolutional Layer liegenden Schieberegister nachgelagert ist, oder ggf. durch Maximum--Funktionen. Addierer sind sehr effizient auf FPGAs realisierbar. Für Maximum-Funktionen können konventionelle 3x2-to-2-Blöcke baumartig verwendet werden.

[0070] Allgemein nimmt der erste Convolutional Layer aus **Fig. 6** die Daten direkt aus dem Eingang mit einer um den Stride-Faktor reduzierten Frequenz an und gibt sie mit wesentlich geringerer, um den nächsten Stride-Faktor reduzierte Frequenz an den nächsten Convolutional Layer weiter, der genauso verfährt. Die immer geringer werdende Frequenz reduziert die Aktivität und damit den Energieverbrauch der späteren Layer deutlich. Dies ist insbesondere für die Fully Connected Layer des nachfolgenden Fully Connected Neural Networks hilfreich. Diese Layer sind wesentlich weniger kompakt und daher weniger energiesparend als die Convolutional Layers gebaut.

[0071] Gemäß **Fig. 6** befindet sich am Ende des letzten Convolutional Layers CLn bzw. Pooling Layers PLn das Schieberegister SR", in das die in Frequenz und Menge stark reduzierten Daten geschrieben werden. Die letzte Schicht wird mit deutlich höherer Bitbreite (z.B. vier Bit) realisiert. An das Schieberegister SR" schließt sich das Fully Connected Neural Network aus einer oder mehreren aufeinander folgenden Fully Con-

nected Layers FC1, ..., FCn an. Die Daten im Schieberegister SR" werden dabei in einen Speicherblock mit wahlfreiem Zugriff geschrieben, um anschließend durch das Fully Connected Neural Network verarbeitet zu werden. Wie oben erwähnt, sind in dem Fully Connected Neural Network die Neuronen in konventioneller Weise und nicht basierend auf Lookup-Blocks realisiert.

[0072] Zur Realisierung des Fully Connected Neural Networks werden sog. DSP-Slices des verwendeten FPGA 100 genutzt. Dazu wird der Ausgang aus der Übergabeschicht in die DSP-Slices gespeist. Aktuelle DSPs sind in der Lage, nach dem SIMD-Prinzip (Single Instruction Multiple Data) statt einer Multiplikation hoher Bitbreite auch mehrere Multiplikationen geringer Bitbreite gleichzeitig zu verarbeiten und auch anschließend getrennt zu akkumulieren. Dies bedeutet, dass jedes einzelne DSP-Slice mehrere Neuronen gleichzeitig berechnen kann und das hunderte vorhandene DSP-Slices gemeinsam eine beachtliche Menge von Berechnungen parallel und dennoch bei geringem Energieverbrauch durchführen können. Daher kann der Beitrag eines jeden Werts einer Schicht für alle Neuronen der nächsten Schicht parallel berechnet werden. Es müssen dazu nur die Gewichte aus dem Speicher geladen werden, die Zwischenergebnisse verbleiben in den Akkumulatorregistern der DSPs. Auf diese Weise werden die sonst dominierenden Speicherkosten stark reduziert. Da auch diese Gewichte in der Bitbreite eingeschränkt sein sollen (auf etwa 8 Bit), können mehrere Gewichte in jedem einzelnen Datenwort gespeichert werden. Statt zwei Speicherzugriffen (Eingang und Gewicht lesen) pro Eingang jedes Neurons wird nur noch ein Lesezugriff pro Eingang je vier Neuronen benötigt, was einem Effizienzgewinn von Faktor 8 entspricht.

[0073] Im hier beschriebenen Ausführungsbeispiel reicht lediglich eine Fully Connected Schicht mit nur zwei Neuronen aus, wobei ein Neuron kodiert, wie sehr ein Vorflimmern erkannt wurde, und das andere Neuron, wie sehr kein Vorhofflimmern erkannt wurde. Der Vergleich der nunmehr wieder in etwa 8 Bit kodierten Werte ergibt die Klassifikation, ob Vorhofflimmern vorliegt oder nicht. Mit anderen Worten liegt Vorhofflimmern dann vor, wenn der Ausgabewert des Neurons, das kodiert, wie sehr Vorhofflimmern erkannt wurde, größer ist als der Ausgabewert des Neurons, welches kodiert, wie sehr kein Vorhofflimmern erkannt wurde. Darüber hinaus erhält man eine Abschätzung der Konfidenz über den Absolutwert des Differenzbetrags der Ausgabewerte der beiden Neuronen. Mit Bezug auf **Fig. 6** erhält man somit über die Ausgangssignale des Fully Connected Neural Networks die Information, ob die Sensordaten zu der Klasse C1 (Vorhofflimmern) oder zu der Klasse C2 (kein Vorhofflimmern) gehören.

[0074] Die erfindungsgemäße Struktur eines neuronalen Netzes umfassend Neuronen in der Form von Lookup-Blocks kann auch zur Klassifikation von beliebigen anderen Sensordaten als EKG-Signalen eingesetzt werden. Im Folgenden wird als weiteres Beispiel eine Klassifikation in einem Produktionsprozess beschrieben.

[0075] An einer Produktionsstraße, in der Kupferkabel mit Kunststoff als Isolationsschicht ummantelt werden, wird direkt nach dem Extruder (dem Gerät, dass den Kunststoff auf die Kupferader aufbringt) die Dicke dieser Schicht in mehrere Richtungen (z.B. in vier orthogonalen Richtungen, die senkrecht auf die Längsrichtung des Kabels stehen) mit sehr hoher Zeitauflösung gemessen. Das Kabel durchläuft die Produktion mit einer Geschwindigkeit von mehreren Metern pro Sekunde, die Dicke soll aber mehrmals pro Millimeter gemessen werden, weshalb Messungen im hohen kHz-Bereich benötigt werden.

[0076] Ein Experte kann der Analyse der Dickeverteilung der Ummantelung (Absolute Dicke, Exzentrizität quer zur Leitungsrichtung, Gleichmäßigkeit der Dicke entlang der Leitungsrichtung, etc.) Fehler im Herstellungsprozess ansehen (z.B. Extrudertemperatur zu gering, Vorschub zu hoch, Beimischung zu gering, etc.). Allerdings muss er dazu ein Stück des Kabels untersuchen. Dies ist erst am Ende der Produktion möglich, wenn das Kabel auf Rollen aufgeteilt wird. In der Zwischenzeit sind viele Kilometer mangelhaftes Kabel hergestellt worden.

[0077] Mit der erfindungsgemäßen neuronalen Netzstruktur kann nunmehr eine automatische Erkennung und Klassifizierung der Artefakte der Dicke der Ummantelung im Mikrosekundenbereich durchgeführt werden. Die vier Sensormessgrößen (Kabeldicke quer zur Leitungsrichtung) kommen im hohen kHz-Bereich in das Erkennersystem. Dieses wird auf die Detektion von sechs klassischen Fehlerbildern (plus Normalfall) trainiert und soll innerhalb weniger Millisekunden aus den Sensordaten das Auftreten von Fehlerbildern ermitteln. Mithilfe der Vorhersage der Artefakte kann eine verbesserte Einregelung der Produktionsanlage (Steuerung relevanter Parameter in der Produktion) realisiert werden. Neu ergibt sich die Möglichkeit, auch kurzfristige Schwankungen (die nur über wenige Meter der Leitung auftreten) durch die schnelle Regelung auszugleichen.

[0078] Die im Vorangegangenen beschriebenen Ausführungsformen der Erfindung weisen eine Reihe von Vorteilen auf. Insbesondere kann auf einem FPGA eine neuronale Netzstruktur mithilfe von baumartig verschalteten Lookup-Tabellen realisiert werden, ohne dass die Gewichte und die Ein- und Ausgangssignale der Neuronen binär quantisiert werden müssen. Es können somit Sensordaten hochgenau lediglich mit den Ressourcen eines FPGA und folglich mit geringem elektrischen Energiebedarf klassifiziert werden. Ferner ist es weiterhin möglich, das neuronale Netz mit einem herkömmlichen Backpropagation-Verfahren zu trainieren.

ZITATE ENTHALTEN IN DER BESCHREIBUNG

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

Zitierte Patentliteratur

- US 2019/0171420 A1 [0003]

Patentansprüche

1. Vorrichtung zur rechnergestützten Klassifikation von Sensordaten (SD) unter Verwendung einer trainierten neuronalen Netzstruktur (NN), die ein oder mehrere neuronale Netze umfasst und eine Vielzahl von Schichten (CL1, PL1, ..., CLn, PLn, FC1, ..., FCn) aus künstlichen Neuronen (NE) enthält, wobei jedes Neuron (NE) eine Vielzahl von Eingängen (IN) und einen Ausgang (OU) umfasst, um ein Signal am Ausgang (OU) in Abhängigkeit von jeweiligen Signalen an der Vielzahl von Eingängen (IN) zu erzeugen, wobei die Vorrichtung derart konfiguriert ist, dass:
- die neuronale Netzstruktur (NN) eine Eingangsschicht (CL1) zur Zufuhr von Sensorsignalen (SI) umfasst, wobei die Sensorsignale (SI) auf Sensordaten (SD) in der Form von Messungen einer oder mehrerer Sensoren basieren;
- die neuronalen Netzstruktur (NN) eine Ausgangsschicht (FCn) umfasst, über deren Ausgangssignale die der Eingangsschicht (CL1) zugeführten Sensorsignale (SI) klassifiziert werden, indem die Sensorsignale (SI) zumindest einer Klasse (CL1, CL2) aus einer Mehrzahl von Klassen (CL1, CL2) zugeordnet werden;
- die neuronale Netzstruktur (NN) auf einem FPGA (100) ausgebildet ist, das ein Feld aus vorgegebenen Lookup-Basiszellen (BZ) umfasst, die jeweils aus Signalen an mehreren 1-Bit-Eingängen ein Signal an einem 1-Bit-Ausgang erzeugen;
- wobei das FPGA (100) derart programmiert ist, dass Lookup-Tabellen (LTO, LT1, ..., LT9) definiert sind, welche jeweils aus einer oder mehreren vorgegebenen Lookup-Basiszellen (BZ) des FPGA (100) gebildet werden und aus mehreren Eingangssignalen (x_0, x_1, ..., v_2) ein Ausgangssignal (uo, u_1, ..., y) erzeugen;
- wobei das oder die Neuronen (NE) zumindest eines Teils der Neuronen (NE) spezifische Neuronen (NE) sind, die jeweils einen Lookup-Block (LB) bilden, der das jeweilige spezifische Neuron (NE) modelliert, wobei der Lookup-Block (LB) ein Baum aus mehreren Lookup-Tabellen (LTO, LT1, ..., LT9) in mehreren Hierarchieebenen (H1, H2, H3) ist, wobei
i. die niedrigste Hierarchieebene (H1) aus mehreren Lookup-Tabellen (LTO, LT1, ..., LT5) besteht, deren Eingangssignale (xo, x_1, ..., x_{35}) den Signalen an den Eingängen (IN) des jeweiligen spezifischen Neurons (NE) entsprechen;
ii. den Lookup-Tabellen (LT6,..., LT9) einer jeweiligen Hierarchieebene (H2, H3), die eine nächsthöhere Hierarchieebene (H2, H3) zu einer benachbarten Hierarchieebene (H1, H2) ist, ausschließlich Ausgangssignalen (uo, u_1, ..., v_2) der Lookup-Tabellen (LTO, LT1, ..., LT8) der benachbarten Hierarchieebene (H1, H2) zugeführt werden und dabei in jeder Lookup-Tabelle (LT6,..., LT9) der jeweiligen Hierarchieebene (H2, H3) mehrere Ausgangssignale (u_0, u_1, ..., v_2) von Lookup-Tabellen (LTO, LT1, ..., LT8) der benachbarten Hierarchieebene (H1, H2) kombiniert werden;
iii. die höchste Hierarchieebene (H3) aus einer einzelne Lookup-Tabelle (L9) besteht, deren Ausgangssignal (y) dem Signal am Ausgang (OU) des jeweiligen spezifischen Neurons (NE) entspricht.

2. Vorrichtung nach Anspruch 1, **dadurch gekennzeichnet**, dass die vorgegebenen Lookup-Basiszellen (BZ) jeweils 32-Bit-Basiszellen mit fünf 1-Bit-Eingängen sind oder dass die vorgegebenen Lookup-Basiszellen (BZ) jeweils 64-Bit-Basiszellen mit sechs 1-Bit-Eingängen sind.

3. Vorrichtung nach Anspruch 1 oder 2, **dadurch gekennzeichnet**, dass in einem jeweiligen spezifischen Neuron (NE) zwischen benachbarten Hierarchieebenen (H1, H2, H3) Signale (uo, u_1, ..., v_2) mit einer Bitbreite von zumindest 2 Bit, vorzugsweise von zumindest 3 Bit, übertragen werden.

4. Vorrichtung nach einem der vorhergehenden Ansprüche, **dadurch gekennzeichnet**, dass das oder die spezifischen Neuronen (NE) in einem oder mehreren Convolutional Layers (CL1, ..., CLn) eines Convolutional Neural Networks liegen, das eine Anzahl von Convolutional Layers (CL1, ..., CLn) umfasst, wobei vorzugsweise alle Neuronen (NE) in den jeweiligen Convolutional Layers (CL1, .., CLn) spezifische Neuronen (NE) sind.

5. Vorrichtung nach Anspruch 4, **dadurch gekennzeichnet**, dass das Convolutional Neural Network die Eingangsschicht (CL1) der neuronalen Netzstruktur (NN) enthält, wobei die Eingangsschicht (CL1) ein Convolutional Layer ist, dessen Neuronen (NE) spezifische Neuronen sind, wobei sich die Anzahl von Convolutional Layers (CL1, ..., CLn) ein Fully Connected Neural Network aus einem oder mehreren Fully Connected Layers (FC1, ..., FCn) anschließt, wobei das Fully Connected Neural Network die Ausgangsschicht (FCn) der neuronalen Netzstruktur (NN) enthält.

6. Vorrichtung nach Anspruch 5, **dadurch gekennzeichnet**, dass das Fully Connected Neural Network in einem oder mehreren DSP-Slices des FPGA (100) implementiert ist.

7. Vorrichtung nach einem der vorhergehenden Ansprüche, **dadurch gekennzeichnet**, dass die Vorrichtung derart ausgestaltet ist, dass die Sensordaten (SD) vorverarbeitet werden, wobei die vorverarbeiteten Sensordaten die Sensorsignale (SI) darstellen, die der Eingangsschicht (CL1) der neuronalen Netzstruktur (NN) zugeführt werden.

8. Vorrichtung nach einem der vorhergehenden Ansprüche, **dadurch gekennzeichnet**, dass die Sensordaten (SD) eine oder mehrere Zeitreihen der Messungen des oder der Sensoren repräsentieren, wobei die Sensorsignale (SI), die der Eingangsschicht (CL1) zugeführt werden, zu einem oder mehreren aufeinander folgenden Zeitabschnitten der Zeitreihe oder Zeitreihen gehören.

9. Vorrichtung nach Anspruch 8, **dadurch gekennzeichnet**, dass die Sensordaten (SD) eine Anzahl von EKG-Signalen umfassen, die jeweils eine Zeitreihe einer elektrischen Potentialdifferenz eines am menschlichen oder tierischen Körper gemessenen Elektrokardiogramms repräsentieren, wobei die Sensorsignale (SI), die der Eingangsschicht (CL1) zugeführt werden, zu einem oder mehreren aufeinander folgenden Zeitabschnitten des oder der EKG-Signale gehören, wobei die Sensorsignale (SI) dahingehend klassifiziert werden, ob und/oder welcher oder welche Krankheitszustände für das Herz vorliegen, von dem die EKG-Signale stammen.

10. Vorrichtung nach einem der Ansprüche 1 bis 8, **dadurch gekennzeichnet**, dass die Sensordaten (SD) aus einem Herstellungsprozess stammen, wobei die Sensorsignale (SI) dahingehend klassifiziert werden, ob und/oder welches oder welche Fehlerbilder im Herstellungsprozess auftreten.

11. Vorrichtung nach Anspruch 10, **dadurch gekennzeichnet**, dass die Sensordaten (SD) Dickemessungen sind, die eine zeitliche Entwicklung einer Dicke einer Ummantelung eines Kabels an einer oder mehreren vorgegebenen Messstellen während der Aufbringung der Ummantelung auf das Kabel repräsentieren, wobei die Sensorsignale (SI) dahingehend klassifiziert werden, ob und/oder welches oder welche Fehlerbilder bei der Aufbringung der Ummantelung des Kabels auftreten.

Es folgen 6 Seiten Zeichnungen

Anhängende Zeichnungen

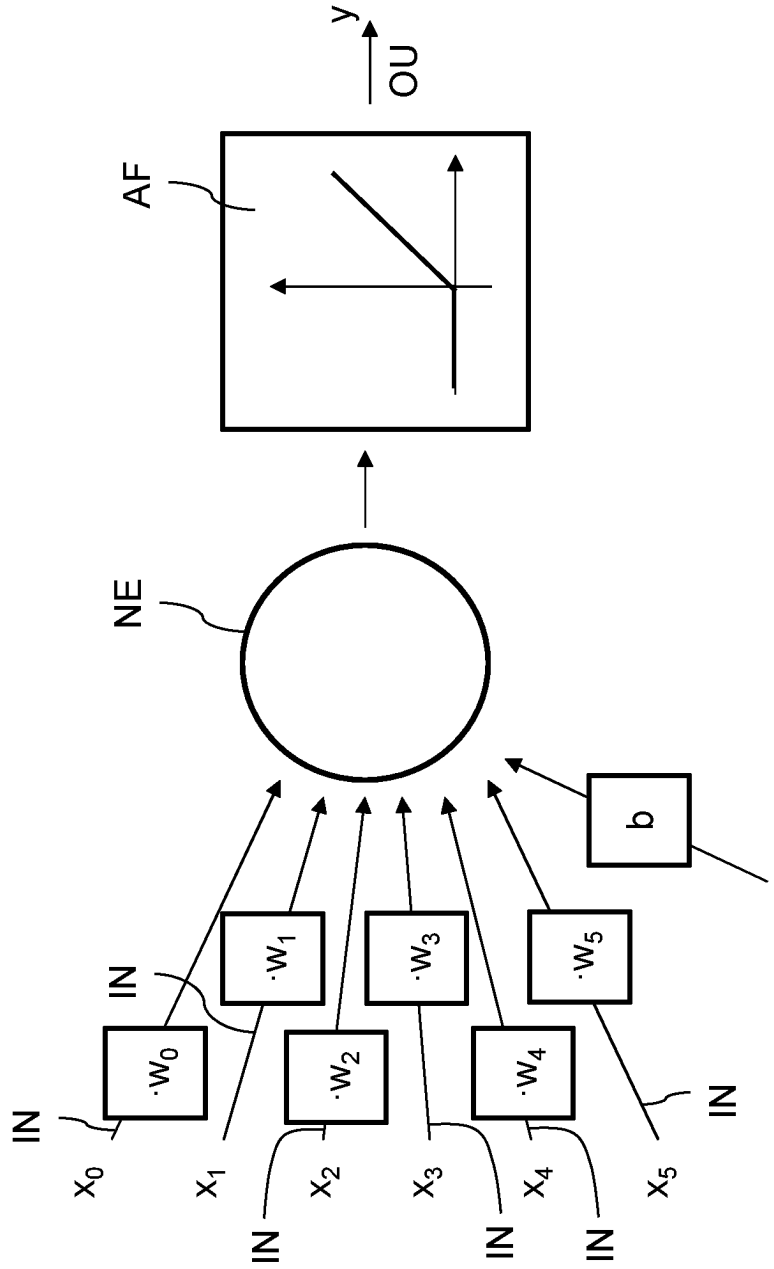


Fig. 1

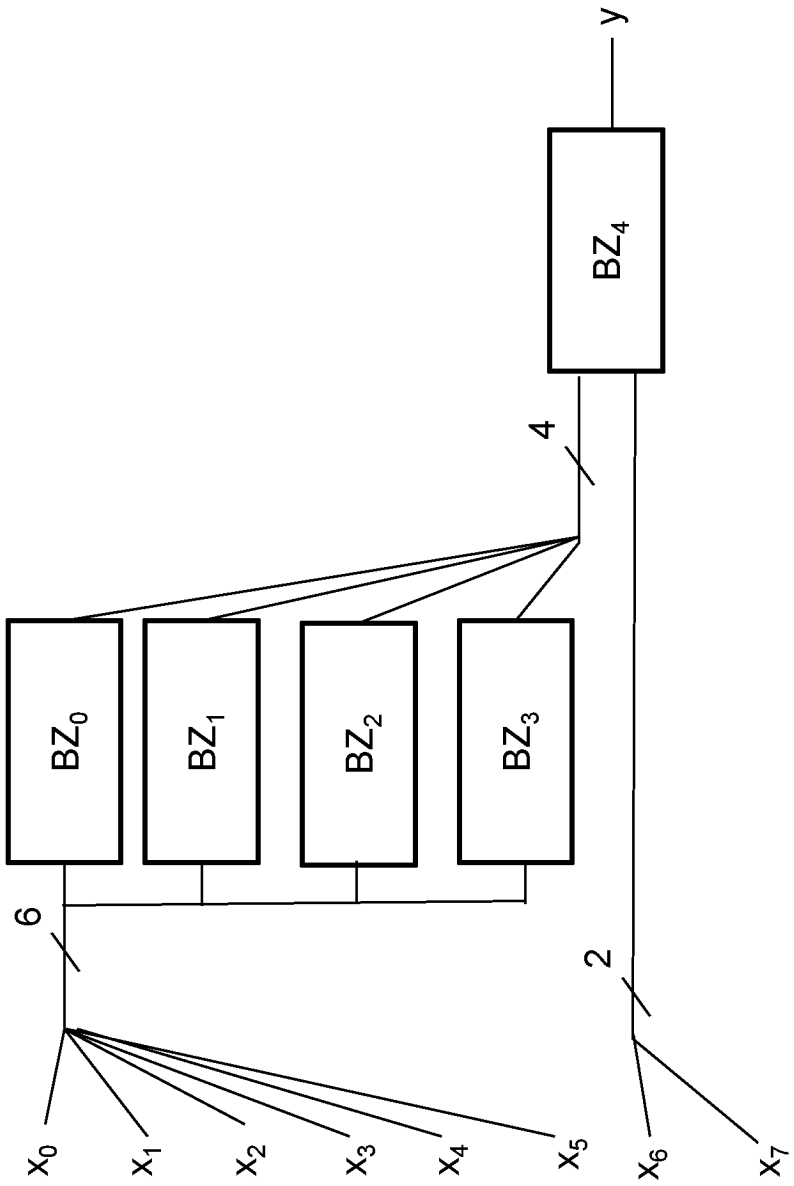


Fig. 2

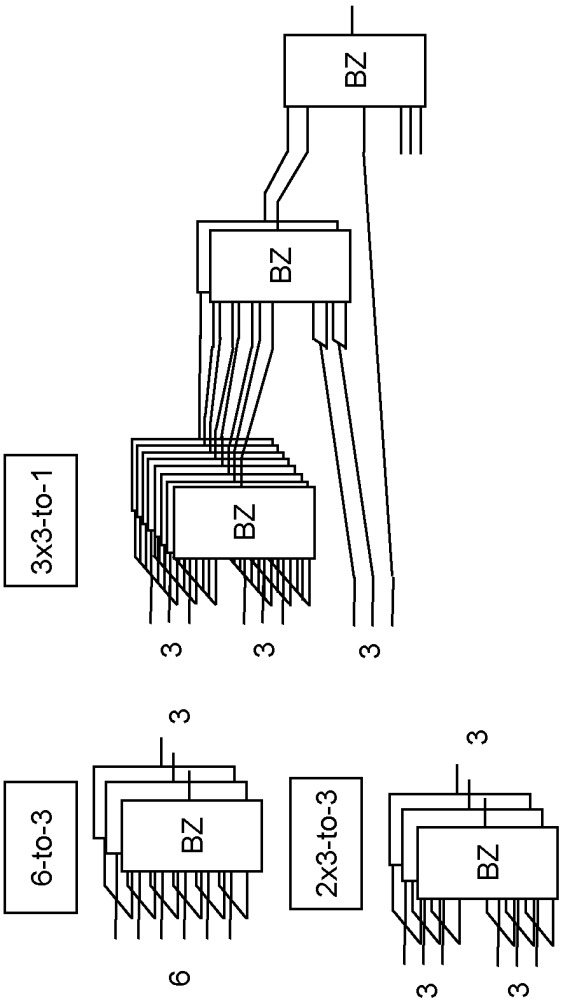

BZ
BZ
BZ
3x3-to-1
3
3
3
6-to-3
BZ
3
6
2x3-to-3
BZ
3
3
3

Fig. 4

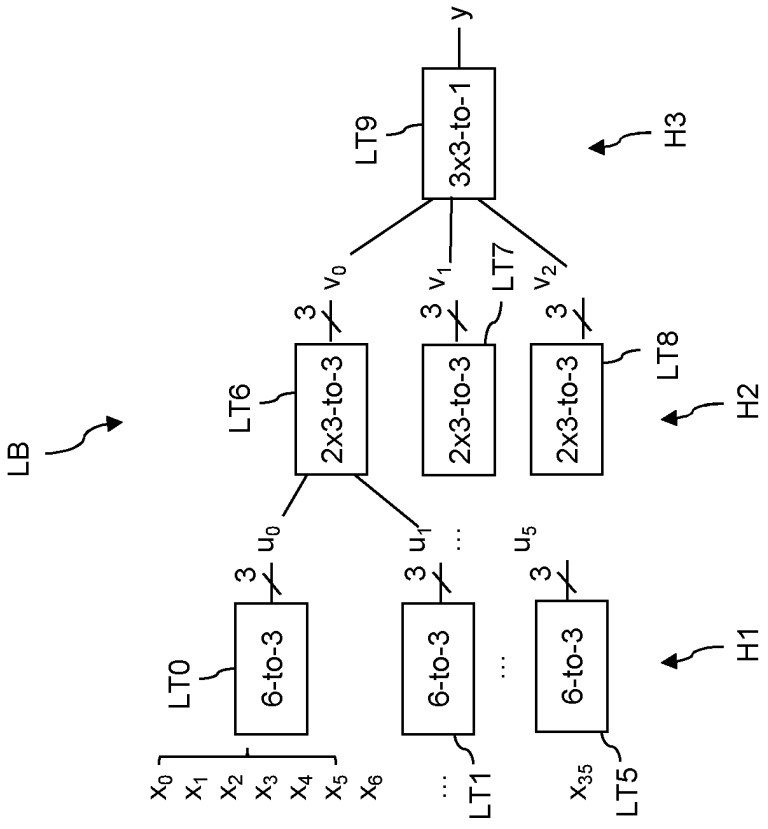

LB
LT0
6-to-3
LT1
6-to-3
LT5
6-to-3
H1
LT6
2x3-to-3
LT7
2x3-to-3
LT8
2x3-to-3
H2
LT9
3x3-to-1
H3
y
x_0 x_1 x_2 x_3 x_4 x_5 x_6
x_{35}
u_0
u_1
u_5
v_0
v_1
v_2
3

Fig. 3

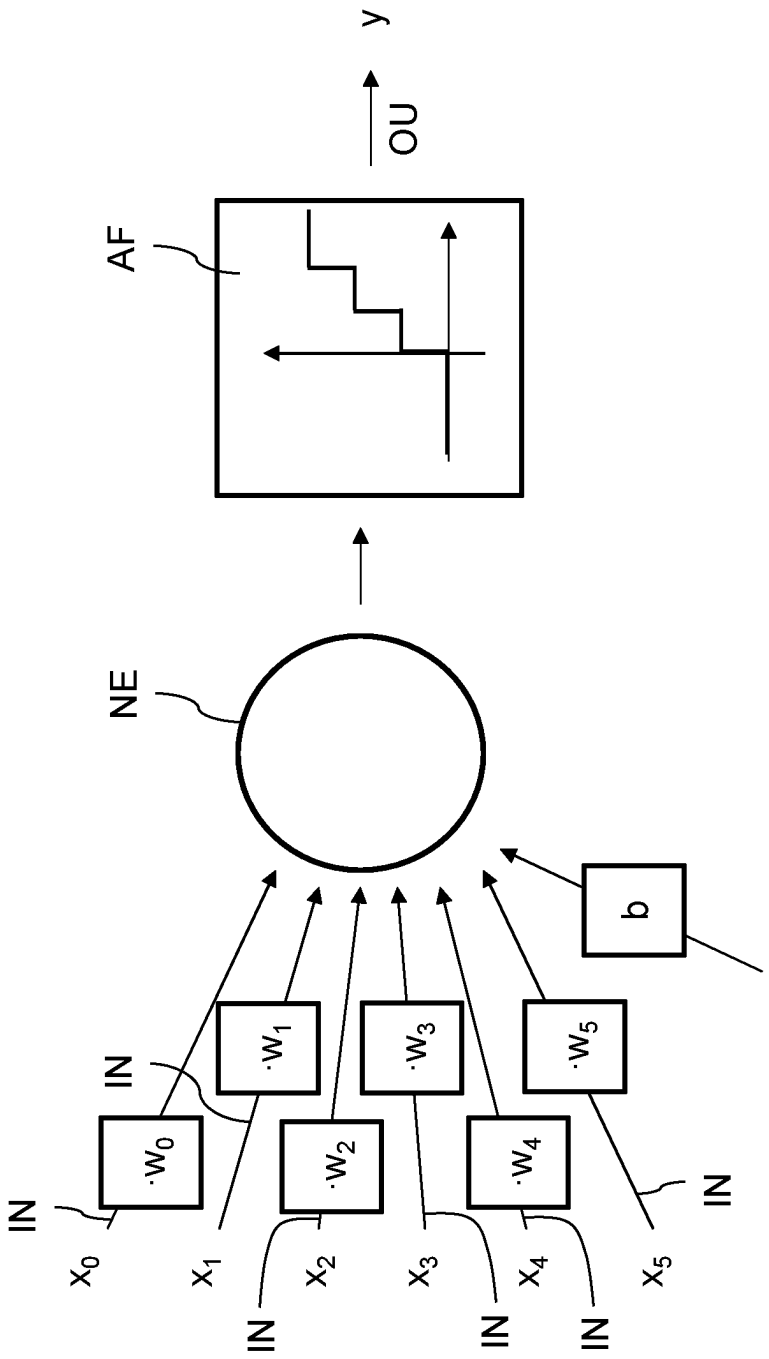

y
OU
AF
NE
b
IN
IN
IN
IN
IN
IN
$\cdot w_0$
$\cdot w_1$
$\cdot w_2$
$\cdot w_3$
$\cdot w_4$
$\cdot w_5$
x_0
x_1
x_2
x_3
x_4
x_5

Fig. 5

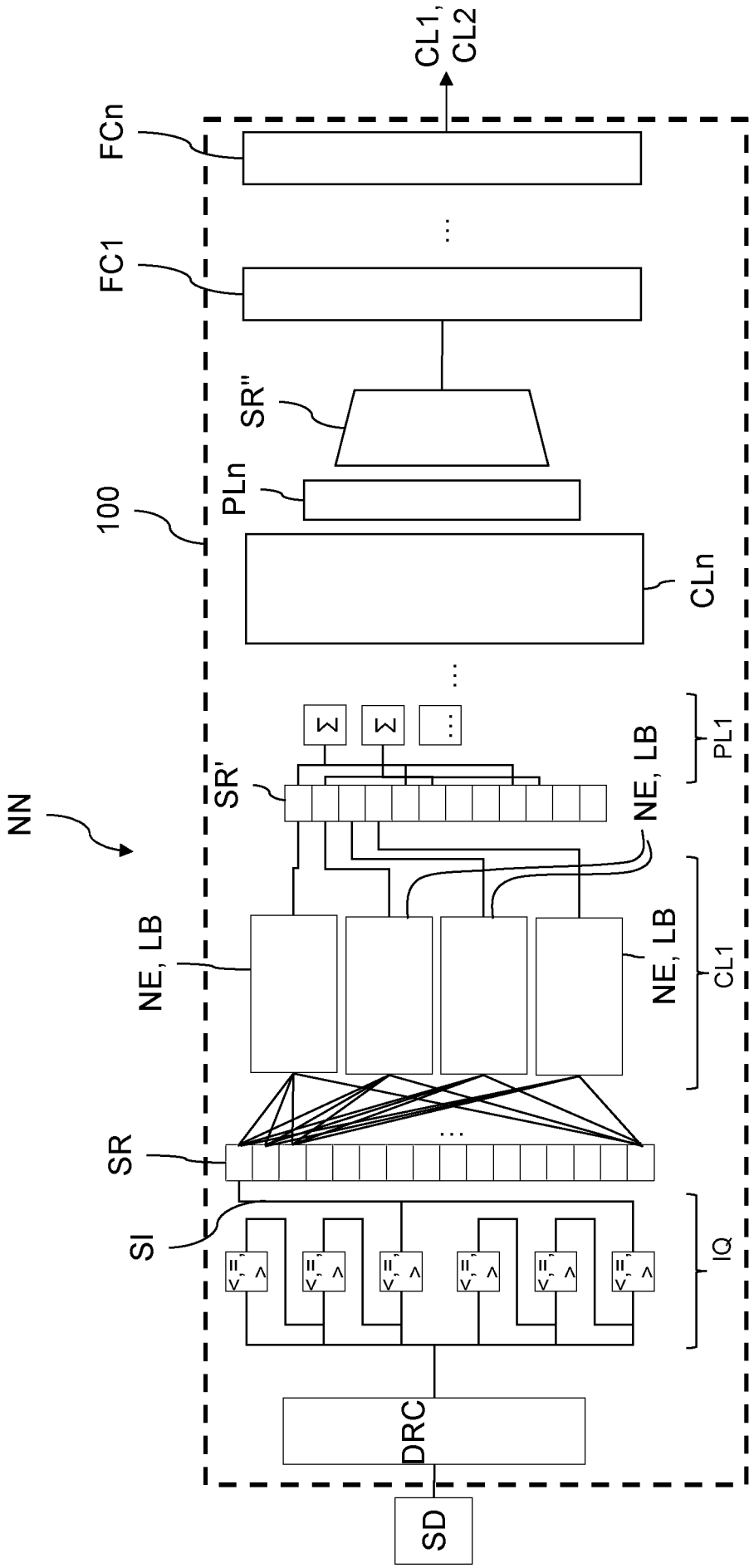

CL1, CL2
FCn
FC1
SR''
PLn
100
CLn
NN
SR'
NE, LB
PL1
NE, LB
NE, LB
CL1
SR
SI
IQ
DRC
SD

Fig. 6

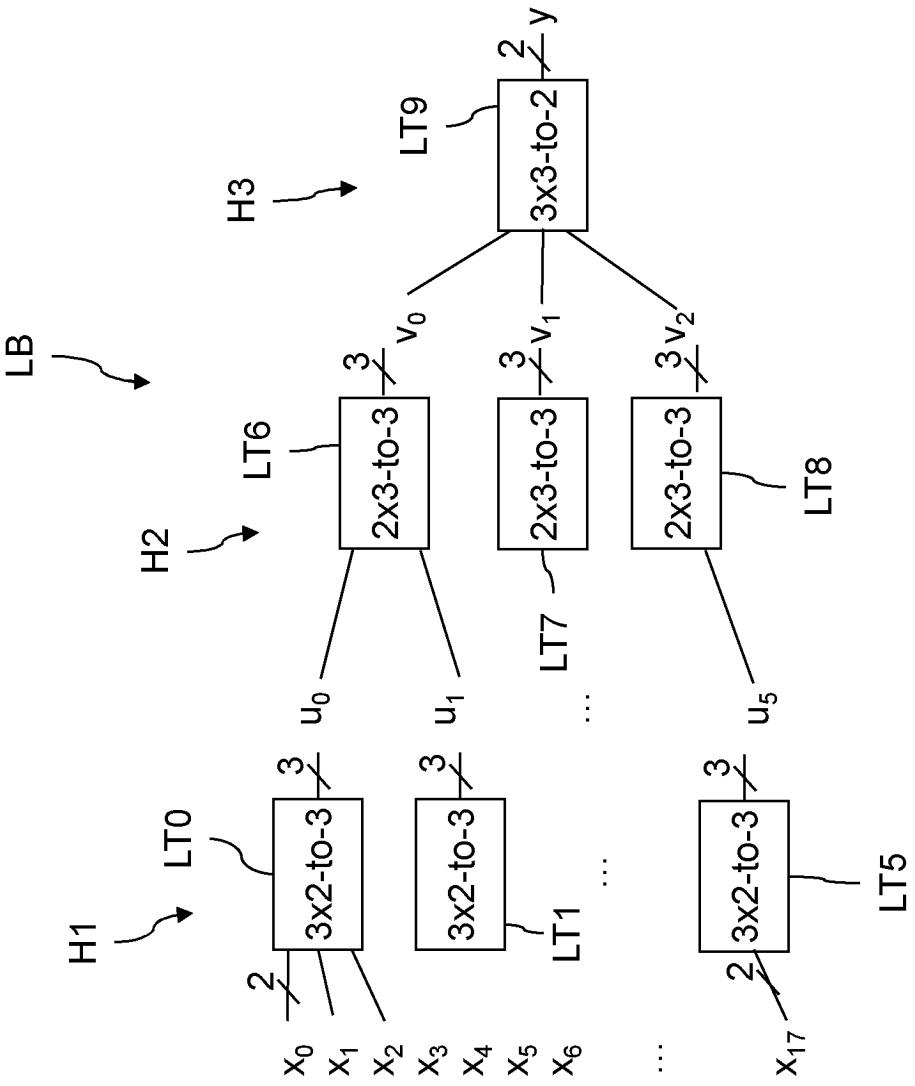


Fig. 7

 Das Dokument wurde durch die Firma Jouve hergestellt.