(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7374236号
(P7374236)

(45)発行日　令和5年11月6日(2023.11.6)

(24)登録日　令和5年10月26日(2023.10.26)

(51)国際特許分類

Ｇ０６Ｆ　17/16　(2006.01)	Ｇ０６Ｆ	17/16	Ｍ
Ｇ０６Ｎ　3/063(2023.01)	Ｇ０６Ｎ	3/063	

ＦＩ

請求項の数　20　外国語出願　（全17頁）

(21)出願番号　特願2022-2202(P2022-2202)
(22)出願日　令和4年1月11日(2022.1.11)
(62)分割の表示　特願2020-503780(P2020-503780)の分割
原出願日　平成30年7月19日(2018.7.19)
(65)公開番号　特開2022-64892(P2022-64892A)
(43)公開日　令和4年4月26日(2022.4.26)
審査請求日　令和4年2月8日(2022.2.8)
(31)優先権主張番号　62/536,399
(32)優先日　平成29年7月24日(2017.7.24)
(33)優先権主張国・地域又は機関　米国(US)
(31)優先権主張番号　15/710,433
(32)優先日　平成29年9月20日(2017.9.20)
(33)優先権主張国・地域又は機関
最終頁に続く

(73)特許権者　510192916
テスラ，インコーポレイテッド
アメリカ合衆国 テキサス州 ７８７２５，オースティン，テスラ ロード １
(74)代理人　110000659
弁理士法人広江アソシエイツ特許事務所
(72)発明者　バノン，ピーター ジョセフ
アメリカ合衆国 カリフォルニア州 ９４０６２，ウッドサイド，グレンウッド アベニュ １２８
(72)発明者　ハード，ケヴィン アルテア
アメリカ合衆国 カリフォルニア州 ９４０６１，レッドウッド シティ，セント フランシス ストリート １１２５
(72)発明者　タルペス，エミル
最終頁に続く

(54)【発明の名称】　加速数学エンジン

(57)【特許請求の範囲】

【請求項１】

センサデータを受信するように構成された第１入力回路と、

複数のフィルタのうちの１つ以上のフィルタを受信するように構成された第２入力回路と、

前記センサデータおよび前記フィルタを受信するように構成された複数のサブ回路であって、各サブ回路は、算術論理ユニットを備え、前記サブ回路は、前記センサデータおよび前記フィルタを畳み込みするように構成された、複数のサブ回路と、を備え、

前記センサデータおよび前記フィルタを畳み込むために、前記サブ回路は、前記サブ回路を介して、前記１つ以上のフィルタで前記センサデータの個々のサブセットを順に畳み込みするように構成され、残りのフィルタのうちの１つ以上が、続いて、畳み込みのために受信され、前記サブ回路の各々が、前記畳み込みに関連する各出力ピクセルを保存する、行列プロセッサ。

【請求項２】

前記センサデータは、画像データ、ライダデータ、超音波データまたはレーダデータを含む、請求項１に記載の行列プロセッサ。

【請求項３】

前記受信されたセンサデータは、線形化されたセンサデータを表す再フォーマットされたオペランドを含む、請求項１に記載の行列プロセッサ。

【請求項４】

前記サブ回路のうちの１つ以上は、エンコーダをさらに備える、請求項１に記載の行列プロセッサ。
【請求項５】
前記サブ回路の少なくとも一部は、特定のエンコーダを共有し、前記特定のエンコーダはブースエンコーダである、請求項４に記載の行列プロセッサ。
【請求項６】
前記行列プロセッサは、冗長データを識別するように構成された状態機械を実装する、請求項１に記載の行列プロセッサ。
【請求項７】
冗長データの識別は、前記複数のフィルタのうちの個々のフィルタおよび／または１つ以上のストライドのうちの個々のストライドに関連付けられた各サイズを含む入力に基づいている、請求項６に記載の行列プロセッサ。
【請求項８】
第１のサブセットの畳み込みは、前記フィルタの１つ以上での前記第１のサブセットの畳み込みを決定することを含み、１つ以上の残りのサブセットは、残りのフィルタの１つ以上で順に畳み込みされる、請求項１に記載の行列プロセッサ。
【請求項９】
前記行列プロセッサは、タイルのアレイを備え、前記タイルは、前記サブ回路の各サブセットを備える、請求項１に記載の行列プロセッサ。
【請求項１０】
出力アレイの各列は、前記複数のフィルタのうちの各フィルタで畳み込みされた前記センサデータの個々のサブセットを含む、請求項１に記載の行列プロセッサ。
【請求項１１】
センサデータをフォーマットし、前記フォーマットされたセンサデータを行列プロセッサに提供するように構成された第１論理回路と、
複数のフィルタのうちの１つ以上のフィルタを前記行列プロセッサに提供するように構成された第２論理回路と、
複数のサブ回路を備える行列プロセッサであって、前記サブ回路は、前記センサデータの個々のサブセットを、前記１つ以上のフィルタで順に畳み込みするように構成され、残りのフィルタの１つ以上が、続いて畳み込みのために受信され、前記サブ回路の各々が、前記畳み込みに関連する各出力ピクセルを保存する、行列プロセッサと、
を備える、システム。
【請求項１２】
前記センサデータの個々のサブセットを順に畳み込みすることは、前記１つ以上のフィルタでの前記センサデータの第１のサブセットの畳み込みを決定することを含み、前記複数のフィルタのうちの残りのフィルタの１つ以上は、前記第１のサブセットでの畳み込みのために受信される、請求項１１に記載のシステム。
【請求項１３】
前記システムは出力アレイを備え、前記出力アレイの各列は、前記複数のフィルタのうちの各フィルタで畳み込みされた前記センサデータの個々のサブセットを含む、請求項１１に記載のシステム。
【請求項１４】
前記第１論理回路は、前記センサデータの一部を格納する複数のデータレジスタを備え、前記複数のデータレジスタは、センサデータから取得された入力領域のサイズに対応する第１幅を有する、請求項１１に記載のシステム。
【請求項１５】
前記入力領域は前記センサデータの個々のサブセットに対応する、請求項１４に記載のシステム。
【請求項１６】
前記システムは、センサデータを複数のベクトルに線形化するように構成され、各ベク

トルは前記センサデータの各サブセットを表す、請求項１１に記載のシステム。
【請求項１７】
行列プロセッサによって実行される方法であって、
第１論理回路から、複数のサブセットを含むセンサデータを受信するステップと、
第２論理回路から、複数のフィルタのうちの１つ以上のフィルタを受信するステップと、
前記行列プロセッサの複数のサブ回路を使用して、前記１つ以上のフィルタで、個々のサブセットを順に畳み込みするステップであって、残りのフィルタのうちの１つ以上が、前記個々のサブセットの順の畳み込みのために、続いて受信され、前記サブ回路の各々が、前記畳み込みに関連する各出力ピクセルを保存する、ステップと、
を含む、方法。
【請求項１８】
前記１つ以上のフィルタで第１のサブセットを畳み込みすることに続いて、第２のサブセットが前記１つ以上のフィルタで畳み込みされる、請求項１７に記載の方法。
【請求項１９】
前記行列プロセッサは、前記複数のフィルタのうちの１つ以上のフィルタの各々を順に受信する、請求項１７に記載の方法。
【請求項２０】
出力アレイの各列に、前記列に関連付けられたフィルタで畳み込みされた前記センサデータの個々のサブセットを出力するステップをさらに含む、請求項１７に記載の方法。
【発明の詳細な説明】
【技術分野】
【０００１】
[関連出願の相互参照]
本出願は、２０１７年7月２４日に提出された「Ａｃｃｅｌｅｒａｔｅｄ Ｍａｔｈｅｍａｔｉｃａｌ Ｅｎｇｉｎｅ」と題された米国仮出願第６２／５３６３９９（２０１５０－２１５４Ｐ（Ｐ０８２２－１ＰＵＳ））の優先権を主張し、発明者としてＰｅｔｅｒ Ｊｏｓｅｐｈ Ｂａｎｎｏｎ、Ｋｅｖｉｎ Ａｌｔａｉｒ Ｈｕｒｄ、およびＥｍｉｌ Ｔａｌｐｅｓを掲載する。前述の特許文書は、その全体が参照により本明細書に組み込まれる。
【０００２】
また、本出願は、２０１７年9月２０日に提出された「Ａｃｃｅｌｅｒａｔｅｄ Ｍａｔｈｅｍａｔｉｃａｌ Ｅｎｇｉｎｅ」と題され、発明者としてＰｅｔｅｒ Ｊｏｓｅｐｈ Ｂａｎｎｏｎ、Ｋｅｖｉｎ Ａｌｔａｉｒ Ｈｕｒｄ、およびＥｍｉｌ Ｔａｌｐｅｓを掲載した共通所有米国特許出願第１５／７１０４３３（２０１５０－２１５４（Ｎ０８２２－１ＮＵＳ））に優先権を主張する。前述の特許文書のそれぞれは、その全体が参照により本明細書に組み込まれる。
【０００３】
本開示は、大量のデータで動作する加速数学エンジンに関し、より具体的には、行列乗算演算に基づいて複雑な畳み込み演算を実行する加速数学エンジンに関する。
【背景技術】
【０００４】
当業者は、時間に敏感で複雑な数学的演算を実施するために使用される一般的なプロセッサおよびシステムに対する速度および性能のますます高まる要求を認識するであろう。これらの一般的なシステムは、大量のデータを処理し、複雑な数学的演算を実行するために使用されるため、計算リソースと計算速度は、これらの計算を実行する既存の一般的なハードウェア設計の機能によって制限される。たとえば、行列演算を実行する汎用コンピュータ装置およびプロセッサは、特定の状況下でこれらの演算をタイムリーに実行できない場合がある。デジタル信号処理操作を実行する多くの従来の乗算器は、一連のソフトウェアおよびハードウェア行列操作手順（アドレス生成、転置、ビットごとの加算およびシフトなど）に依存しており、時間に敏感なシステム内においてボトルネックになり得る。

多くの場合、これらの操作ステップでは、プロセッサの算術関数を使用して中間結果を生成する必要があるが、さまざまな場所から中間結果を保存およびフェッチして操作を完了するため、計算時間が無駄になる。

【０００５】

図１は、従来の乗算器システムの一例を示している。乗算器システム１００は、計算ユニット１０２、レジスタ１０４、キャッシュ１０６、およびメモリ１０８を含むスカラマシンである。動作中、計算ユニット１０２は、レジスタ１０４およびキャッシュ１０６を使用して、メモリ１０８に格納されたデータを検索する。典型的には、計算ユニット１０２は、例えば乗算を加算に変換し、結果を何らかの内部レジスタに出力することにより、結果の行列を得るために入力行列で行列乗算を含む様々な計算手順を実行できるＣＰＵまたはＧＰＵなどのマイクロプロセッサである。

【０００６】

たとえば、画像の出力ピクセルを表す内積は、通常、２つの行列の個々の行列要素をドット乗算して部分的な結果を取得し、最終結果を加算して最終的な内積を取得することによって生成される。個々の行列要素の乗算、つまりスカラ乗算は、通常、ドット乗算を一連の個々のサブ操作に分割することにより、個々のデータ要素に対して実行される。その結果、単一の算術演算を完了するために、レジスタ１０４、キャッシュ１０６、およびメモリ１０８の１つまたは複数に部分積を格納およびフェッチする必要がある。

【０００７】

畳み込みなどの計算が要求されるアプリケーションでは、多くの場合、計算ユニット１０２にソフトウェア関数を組み込み、畳み込み演算を代替の行列乗算演算に変換するために使用する必要がある。これは、データを２つの行列に並べ替えて再フォーマットし、それをそのまま行列乗算できるようにすることで実現される。しかし、各スカラ演算を実行するために必要なデータをレジスタから何度も再格納および再フェッチする必要があるため、スカラマシン１００でデータを効率的に共有または再利用するメカニズムは存在しない。これらの操作の複雑さと管理オーバーヘッドは、畳み込み演算の対象となる画像データの量が増加するにつれて著しく大きくなる。

【０００８】

算術演算を完了するためにレジスタ１０４、キャッシュ１０６、およびメモリ１０８から中間結果を格納およびフェッチする追加された非効率的なステップと相まって、スカラマシン１００内のデータの多くを再利用できないことは、乗算器システム１００などの既存のシステムの欠点の一部にすぎない。

【発明の概要】

【発明が解決しようとする課題】

【０００９】

したがって、必要なのは、行列数学演算を迅速かつ効率的に実行できる高計算スループットのシステムと方法である。

【図面の簡単な説明】

【００１０】

本発明の実施形態を参照し、その例を添付の図面に示すことができる。これらの図は、限定ではなく例示を目的としている。本発明をこれらの実施形態の文脈で一般的に説明するが、本発明の範囲をこれらの特定の実施形態に限定することを意図するものではないことを理解されたい。図の項目は縮尺どおりではない場合がある。

【００１１】

【図１】従来の乗算器システムの例を示す。

【００１２】

【図２】本開示の様々な実施形態に従って算術演算を実行するための例示的な行列プロセッサアーキテクチャを示す。

【００１３】

【図３】図２に示される行列プロセッサアーキテクチャの例示的な構成の詳細を示す。

【００１４】
【図４】図３に示される論理回路の例示的な乗加算回路の実装を示す。
【００１５】
【図５】本開示の様々な実施形態による例示的な畳み込み演算を示す。
【００１６】
【図６】本開示の様々な実施形態による例示的な畳み込み演算の詳細を示す。
【図７】本開示の様々な実施形態による例示的な畳み込み演算の詳細を示す。
【図８】本開示の様々な実施形態による例示的な畳み込み演算の詳細を示す。
【００１７】
【図９】本開示の様々な実施形態による例示的な逆畳み込み演算を示す。
【００１８】
【図１０】本開示の様々な実施形態による、畳み込みニューラルネットワークをより高速にするために算術演算を実行するプロセスを示す図である。
【発明を実施するための形態】
【００１９】
以下の説明では、説明の目的で、本発明の理解を提供するために特定の詳細が述べられている。ただし、これらの詳細なしで本発明を実施できることは当業者には明らかであろう。さらに、当業者は、以下で説明する本発明の実施形態が、プロセス、装置、システム、デバイス、または有形のコンピュータ可読媒体上の方法などの様々な方法で実装され得ることを認識するであろう。
【００２０】
図に示される構成またはモジュールは、本発明の例示的な実施形態の例示であり、本発明を曖昧にすることを避けることを意図している。また、この説明全体を通して、構成はサブユニットを含み得る別個の機能ユニットとして説明されるが、当業者は、さまざまな構成またはその一部が別個の構成に分割されてもよく、または単一のシステムまたは構成内に統合されることを含め、一緒に統合されてもよいことがわかるであろう。本明細書で説明する機能または動作は、構成として実装できることに留意されたい。構成は、ソフトウェア、ハードウェア、またはそれらの組み合わせで実装できる。多くの構成は、多くのサブ構成の相互接続を通じて形成される。これらの論理的に異なるサブ構成を集約して他のサブ構成と組み合わせることができ、ここで説明したものと同様のまたは同一の機能を集約構成レベルで提供できる場合は、ここで示されているものと動作が論理的に異なるサブ構成を選択できる（たとえば、アクティブなｈｉｇｈ信号をアクティブなｌｏｗ信号に、ＡＮＤゲートを反転入力ＮＯＲゲートなどに置き換えることができる、など）。
【００２１】
さらに、図内の構成またはシステム間の接続は、直接接続に限定されるものではない。むしろ、これらの構成間のデータは、中間構成によって変更、再フォーマット、または変更され得る。また、追加の接続またはより少ない接続を使用してもよい。また、「結合」、「接続」、または「通信結合」という用語には、直接接続、１つまたは複数の中間デバイスを介した間接接続、および無線接続が含まれることを理解されたい。
【００２２】
本明細書における「一実施形態」、「好ましい実施形態」、「実施形態」、または「いくつかの実施形態」への言及は、実施形態に関連して説明される特定の特徴、構造、特性、または機能が本発明の少なくとも１つの実施形態に含まれ、２つ以上の実施形態に含まれてもよいことを意味する。また、本明細書の様々な場所での上記のフレーズの出現は、必ずしもすべてが同じ実施形態または複数の実施形態を指しているわけではない。
【００２３】
本明細書のさまざまな場所での特定の用語の使用は、説明のためであり、限定するものとして解釈されるべきではない。サービス、機能、またはリソースは、単一のサービス、機能、またはリソースに限定されず、これらの用語の使用は、分散または集約され得る関連するサービス、機能、またはリソースのグループ化を指す。

【００２４】
「含む」、「含んでいる」、「備え」、「備えている」という用語は、限定されていない（ｏｐｅｎ）用語であると理解され、以下の如何なるリストも例であり、リストされた項目に限定されることを意図するものではなく、追加項目とともに、部分集合または上位集合の項目を含み得る。ここで使用されているあらゆる見出しは、組織的な目的のみに使用され、説明または如何なる請求項の範囲をも制限するために使用されることはない。この特許文献に記載されている各文献は、その全体が参照により本明細書に組み込まれる。
【００２５】
さらに、当業者は、（１）特定のステップをオプションで実行できること、（２）ステップは、ここに記載されている特定の順序に限定されない場合があること、（３）特定のステップが異なる順序で実行される場合があること、（４）特定のステップを同時に実行できること、を認識しなければならない。
【００２６】
本明細書の実施形態は主に畳み込みの文脈で説明されているが、当業者は、逆畳み込みおよび他の行列演算も行列－行列型乗算演算として構成できること、したがって本発明の原理も同様に逆畳み込みに適用できることを理解するであろう。さらに、本開示の様々な実施形態に従って、他のタイプの数学的演算を実施してもよい。
【００２７】
図２は、本開示の様々な実施形態に従って算術演算を実行するための例示的な行列プロセッサアーキテクチャを示している。システム２００は、論理回路２３２、２３４、キャッシュ／バッファ２２４、データフォーマッタ２１０、重みフォーマッタ２１２、データ入力行列２０６、重み入力行列２０８、行列プロセッサ２４０、出力アレイ２２６、後処理ユニット２２８、および制御ロジック２５０を含む。行列プロセッサ２４０は、算術論理ユニット（Ａｒｉｔｈｍｅｔｉｃ　Ｌｏｇｉｃ　Ｕｎｉｔｓ：ＡＬＵ）、レジスタ、および、いくつかの実施形態ではエンコーダ（ブースエンコーダなど）を含む複数のサブ回路２４２を含む。論理回路２３２は、Ｎ個の入力演算子およびデータレジスタを表す回路であり得る。論理回路２３４は、Ｍ個の重みオペランドを行列プロセッサ２４０に入力する回路であり得る。論理回路２３２は、画像データオペランドを行列プロセッサ２４０に入力する回路であり得る。重み入力行列２０８およびデータ入力行列２０６は、ＳＲＡＭデバイスを含む様々なタイプのメモリに格納されてもよい。当業者は、様々なタイプのオペランドが行列プロセッサ２４０に入力され得ることを認識するであろう。
【００２８】
特定の実施形態による動作において、システム２００は、システム内の冗長動作を削減し、ハードウェア固有のロジックを実装してデータおよび重みの大きなセットにわたって特定の数学的演算を実行することにより畳み込み演算を加速する。この加速は、行列プロセッサ２４０内の数学的演算のタイミングを大規模に計るだけでなく、画像データと重みとを取得して行列プロセッサ２４０に入力する方法（および対応するハードウェア構成）の直接的な結果である。
【００２９】
実施形態において、図２のフォーマッタ２１０、２１２の例はインラインフォーマッタとして実装される。特定の実施形態では、フォーマッタ２１０、２１２は別個の構成であり、他の実施形態では、フォーマッタ２１０、２１２は一緒におよび／または１つ以上の他の構成と統合される。それぞれがハードウェアで実装され、行列をオペランド上のベクトルに変換して、行列プロセッサ２４０内で動作する。他の実施形態では、フォーマッタ２１０、２１２はソフトウェアで実装されるが、これは通常、速度の低下をもたらす。データフォーマッタ２１０は、データ入力行列２０６を含む２次元または三次元（例えば、３×３×３立方）のデータを単一のベクトルまたは行もしくは列で表されるストリングに変換し、それによりデータ入力行列２０６を線形化またはベクトル化する。詳細には、フォーマッタ２１０はデータ入力行列２０６を受け取り、行列プロセッサ２４０によって処理される入力データを準備する。実施形態において、これは、行列プロセッサ２４０が出

力ピクセルを生成するときに畳み込み計算の一部として行列乗算を効率的に実行できるように、行列プロセッサ２４０のハードウェア要件に従ってデータ入力行列２０６のパラメータを適切なフォーマットにマッピングすることによって達成される。

【００３０】

一例として、行列プロセッサ２４０が９６行９６列を含むと仮定すると、９６×９６フォーマットにマッピングされたデータは、行列プロセッサ２４０をその全計算能力まで利用させ、したがって、好ましい効率を提供する。その場合、フォーマッタ２１０は９６列幅の出力を生成する必要がある。同様に、フォーマッタ２１２は、重み入力行列２０８に基づいて９６行幅の出力を生成する必要がある。

【００３１】

実施形態では、フォーマッタ２１０は、いくつかのマルチプレクサまたはスイッチを使用して、データ入力行列２０６の一部またはすべてをフェッチし、そこから異なる要素を選択して、行列プロセッサ２４０の列に従って並べられるデータを生成する。実施形態では、選択により、データ入力行列２０６からの適切なデータが、規定のクロックサイクルで各列に渡されることが保証される。実施形態において、重みが静的である場合、それらはオフラインで事前にフォーマットされ、メモリに格納され、一度だけフェッチされ、フォーマッタ２１２を使用せずに修正されたベクトル化されたフォーマットで行列プロセッサ２４０に直接供給されてもよい。他の実施形態では、様々なフォーマットおよびフェッチ操作に従って、重みを動的に調整し、行列プロセッサ２４０に供給してもよい。実施形態では、行列プロセッサ２４０は、さまざまなサイズの列および行の入力を可能にする。すなわち、行列プロセッサ２４０は、任意のサイズのＮ×Ｍ計算を計算するように設計されている。

【００３２】

他の実施形態では、データ入力行列２０６の列の数（例えば、Ｘ）が行列プロセッサ２４０の列の数よりも大きくなるように行列プロセッサ２４０の列の数が制限（例えば、Ｎ列に）された場合（すなわち、Ｘ＞Ｎ）の場合、制御ロジック２５０は、データ入力行列２０６を複数のサブ行列に分割し、各サブ行列が行列プロセッサ２４０によって計算されるようにすることができる。そのような場合、各行列プロセッサ２４０は異なるスレッドで実行されていてもよい。たとえば、データ入力行列２０６が１９２×９６のデータポイントで構成され、行列プロセッサが９６列９６行を有している場合（すなわち、１クロックサイクルで９６×９６の計算が発生する場合）、制御ロジック２５０はデータ入力行列２０６を２つのサブ行列（データ入力行列２０６の左半分とデータ入力行列２０６の右半分など）に分割してもよい。各サブ行列は、９６×９６のデータポイントで構成される。個別にスレッド化された各行列プロセッサ２４０は、送信されるサブ行列の出力チャネルを計算し、全チャネルからの値（つまり１９２の値）を保持するのに十分大きいはずである最終出力アレイ２６０に結果を配置する。より一般的には、データ入力行列２０６は、任意の数のサブ行列に分割され、それぞれが別個のスレッドで実行される異なる行列プロセッサ２４０に送信されてもよい。出力アレイ２２６と同様に、データ入力行列２０６、データフォーマッタ２１０、キャッシュ／バッファ２２４、論理回路２３２、および後処理ユニット２２８は、同様により大きなデータを収容できなければならない。

【００３３】

代替実施形態では、制御ロジック２５０が内積に沿って計算を分割することにより、複数の行列プロセッサ２４０間でＣＮＮを計算することができる。内積のセグメントがそれぞれ異なる行列プロセッサ２４０で計算され、次いで、入力された積が一緒に加算されて出力ベクトルが計算され、出力ベクトルは出力アレイ２６０に格納される。

【００３４】

高速行列乗算に適した代替形式にデータを再配置することにより畳み込み演算を行列乗算に変換するための、ＣＰＵまたはＧＰＵによって実行されるフォーマット機能の一般的なソフトウェア実装とは異なり、本開示のさまざまなハードウェア実装は、データを臨機応変にフォーマットし、実行できるようにする。たとえば、サイクルごとに９６個のデー

タが有効になり、行列の非常に多数の要素を並列処理できるため、データを行列演算に効率的にマッピングすることができる。実施形態では、２Ｎ個のフェッチされた入力データについて、$２Ｎ^{2}$の計算データが単一のクロックサイクルで取得され得る。このアーキテクチャは、複数のデータ入力にわたって多数の数学的演算を実行する並列で効率的な同期プロセスを提供するだけでなく、一般的なプロセッサアーキテクチャで使用される読み取り操作またはフェッチ操作の数を効果的に削減することにより、処理速度を大幅に向上させる。

【００３５】

実施形態では、任意の数の列および行を有する行列プロセッサ２４０の効率を高めるために、フォーマッタ２１２、２１４は、入力行列データの異なる形状を行列プロセッサ２４０に適した列および行に再フォーマットすることができる。実施形態では、異なる入力サイズを有する行列の処理に対応するために、フォーマットが動的に実行される。実施形態において、入力チャネルを含む再フォーマットされた行列は、キャッシュ／バッファ２２４に供給される。

【００３６】

キャッシュ／バッファ２２４は、様々なデータが再利用されるため、ｋを畳み込みカーネル幅として、データ入力行列２０６から１／ｋ回だけデータをフェッチすることができる。たとえば、特定のサイクルで行がフェッチされると、特定の列はその行のすべてのデータにアクセスできる。いくつかの実施形態において、キャッシュ／バッファ２２４は、ＳＲＡＭに再アクセスしてＳＲＡＭからデータ読み取る必要なく、畳み込みによって再利用され得るデータの部分的なコピーを格納する部分的なバッファであってもよい。

【００３７】

行列プロセッサ２４０が計算を完了すると、結果のセットは、例えば、行列プロセッサ２４０の最下行のアキュムレータから、例えば、内積を受信するシフトレジスタを効果的に形成する出力フリップフロップ（図示せず）にシフトされ得る。実施形態では、出力チャネルに対応する行から、例えばクロックサイクルごとに１つ、出力アレイ２２６に結果をプルまたはシフトすることは、状態機械（図示せず）によって達成され得る。状態機械は、例えば、データをＳＲＡＭおよび／または後処理ユニット２２８に送信する前に、出力チャネル上で追加の動作を実行してもよい。行列プロセッサ２４０の内部動作は、以下により詳細に説明される。

【００３８】

実施形態では、行列プロセッサ２４０は、行列プロセッサ２４０を介して出力アレイ２２６に渡される結果のコピーを保存することにより並列処理を可能にするシャドーレジスタを含む。実施形態では、演算結果を出力レジスタからシャドーレジスタに移動することは、次の値のセットをＡＬＵにロードすることを伴う。

【００３９】

累積が完了すると、畳み込みが開始され、前の畳み込みのすべてのデータが出力アレイ２２６に出力される前に蓄積がやり直され得る。その結果、各クロックサイクルにおいて、行列プロセッサ２４０のデータが１行下に移動し、各サイクルにおいて最後の行が出力アレイ２２６に出力されるようになる。実際、この動作モードは、ＳＲＡＭにデータを保存するなどの追加の処理操作とは無関係に、中断することなく連続した各サイクルで新しい計算が行われることを保証する。

【００４０】

後処理ユニット２２８は、ハードウェア加速プーリングユニット、メモリからデータを取得してＳＲＡＭなどにデータ（重みや結果など）を保存するダイレクトメモリアクセス（ｄｉｒｅｃｔ ｍｅｍｏｒｙ ａｃｃｅｓｓ：「ＤＭＡ」）の一部であり得るＤＲＡＭなどの多数のデバイス（図示せず）を含むか、またはそれらと相互作用し得る。デバイスは、システム２００内のフォーマッタ２１０、２１２および他の構成を管理する制御ロジック２５０によって部分的または全体的に制御されてもよい。

【００４１】

データを読み取るためのアドレスを生成し、結果を書き込み、畳み込みの後続のステップで使用されるデータを取得する場所と実行する方法とを計算するために、システム２００が畳み込みのどこにあるかを追跡するシーケンサなど、管理機能を実行する補助デバイスは図２には示されていない。

【００４２】

特定の実施形態では、重み入力行列２０８は物理的に分割され、行列プロセッサ２４０の２つの異なる側から重みを駆動し、２次元アレイが、重み入力行列２０８のデータの一部をそれぞれが受け取る２つの領域（例えば、左側と右側）に分割されるようにする。このような実装では、重みが既知であるという事実を活用することにより、データの待ち時間が短縮される。実施形態では、ピーク電力消費を低減するために、重みとデータとの乗算が特定のサイクル数にわたって分散されるように動作タイミングが選択されてもよい。この効率的な動作タイミングにより、行列プロセッサによって実行される読み取り操作の数の減少や、行列内（サブ回路間など）でのデータ移動の効率の改善など、エネルギー消費ステップが削減される。

【００４３】

実施形態では、冗長データを識別するように構成された状態機械（図示せず）を使用することができる。識別された冗長データは列全体で再利用できるため、データを再フェッチする必要はない。状態機械は、例えば、画像サイズ、フィルタサイズ、ストライド、チャネル数、および同様のパラメータに関連する入力に基づいて、実行されるデータをどのように、およびどこにシフトするかを決定するように構成されてもよい。

【００４４】

実施形態では、ブースエンコーダは、行列プロセッサ２４０の乗算アーキテクチャのいくつかの要素にわたって共有される。ブースエンコーダは、当技術分野で知られている如何なるブースエンコーダであってもよく、２つの数値を乗算し、２つの数値の１つを、たとえば、８ビット値から１２ビットまたは乗算器ロジックでの乗算演算を容易にするその他の値にエンコードするために使用されてもよく、したがって、より高速である。実施形態では、ブースエンコーダは、同一のコード化された代替の重み値をすべての列にわたって共有するように、行全体にわたって並列に適用されてもよい。すべての列にオペランドをロードすることにより、行全体にわたって１クロックサイクルで乗算を実行できる。したがって、Ｎ個の計算要素で同じデータ（たとえば、重み）を共有するために再エンコードを活用するためのコストは、各列（または行）に対して１回だけ支払われる。これとは対照的に、既存の計算アーキテクチャでは、すべての単一スカラにおいて、すべての乗算演算のためにブースエンコーダが必要になる。

【００４５】

図３は、図２に示される行列プロセッサアーキテクチャの例示的な構成の詳細を示す。実施形態では、行列プロセッサ３００は、各軸線上において所定のベクトル長に対応し得る。図３に示されるように、行列プロセッサ３００は、行列形式で配置された６×６タイル３０２のアレイを備えてもよい。各タイル３０２は、サブ回路３５０を順に含む行列３２０を含み得る。図４を参照して以下で詳細に説明するように、各サブ回路３５０は、算術演算を実行できるセルであってもよい。実施形態では、サブ回路３５０は、乗算、累積、およびシフト演算を同時に実行する。

【００４６】

実施形態では、行列プロセッサ３００の複数の行および列を利用してＮ×Ｎタイル出力を生成することにより、算術演算が並列化される。たとえば、９６の行サイズと９６の対応する列サイズとは、２＊９２１６の数学計算の出力を容易にする。他の実施形態では、行の数と列の数とは異なっていてもよい。つまり、Ｎ行Ｍ列であり得、Ｎ×Ｍタイル出力が生成され得る。たとえば、行サイズが９６で対応する列サイズが１９２の場合、１クロックサイクルで２＊１８，４３２の計算結果が生成される。

【００４７】

図４は、図３に示されるサブ回路の例示的な乗加算回路の実装を示す。図４に示すよう

に、乗加算回路４００は、乗算器４３０、加算器４３２、論理４３４、４３６、４３８、アキュムレータ４２４、シャドーレジスタ４２８、および出力レジスタ４４０を含む。実施形態では、アキュムレータ４２４は累算レジスタとして実装されてもよい。

【００４８】

実施形態では、アキュムレータ４２４は、レジスタを含むＡＬＵのセットと、ＡＬＵの出力を受信するように構成され得るシャドーレジスタ４２８とを含み得る。

【００４９】

動作中、乗算器４３０は、重み４０２およびデータ４０４を受信して乗算し、それらから積を生成する。各積は、乗算器４３０から積を受け取ることに応答して、積をアキュムレータ４２４の現在値に加算する加算器４３２に提供されてもよい。

【００５０】

実施形態において、アキュムレータ４２４は、例えば出力レジスタ４４０に格納される累算値を生成する。累積値は畳み込みの結果であり、図２を参照して述べたように、２つのフォーマットされた行列の内積に対応し得る。

【００５１】

実施形態では、結果４５０を出力するシャドーレジスタ４２８に出力レジスタ４４０の結果のコピーを提供し、アキュムレータ４２４に再度アクセスして新しい計算を開始することができる。実施形態では、図４の乗加算回路４００は、乗算、加算演算、およびシフト演算を同時に、すなわち単一のサイクル内で実行することができ、それにより各サイクルで発生する演算の総数を倍にすることができる。

【００５２】

実施形態において、ＣｌｅａｒＡｃｃ信号４０８は、例えば、乗算器４３０が乗算演算を実行したときに、累積演算がやり直せるように、アキュムレータ４２４の内容を消去する。実施形態において、ＲｅｓｕｌｔＥｎａｂｌｅ信号４１２は、データ４０４が有効であるという判定に応じて起動される。アキュムレータ４２４は、データを累積して保存し、データを累積して消去し、または単にデータを消去し得ることが理解される。

【００５３】

実施形態では、結果は、単一のクロックサイクルで、すなわち、中間の実行動作および保存動作を必要とせずに、出力レジスタ４４０からシャドーレジスタ４２８に移動される。

【００５４】

図５は、本開示の様々な実施形態による例示的な畳み込み演算を示している。畳み込み５００は、入力画像５０２の入力チャネルＩＣ、重み５３２、内積５１４、出力チャネルＯＣ、およびアキュムレータ５４０を含む。

【００５５】

実施形態では、畳み込み演算５００は、例えば入力画像５０２内の小さな特徴を検出するために、個々のフィルタ（すなわち、重み）５３２を入力画像５０２に適用する。異なる順序で一連の異なる特徴を分析することにより、入力画像５０２でマクロ特徴を識別することができる。他の実施形態では、入力５０２は非画像データである。例えば、入力５０２は、超音波、レーダ、ライダ（ＬＩＤＡＲ）、または他のセンサデータなどの非画像センサデータであってもよい。入力５０２は、一般的な数学的計算または当業者に知られている他のタイプのデータでもよい。

【００５６】

畳み込み５００は、各入力チャネルＩＣが異なる情報のセットを含み得るため、各入力チャネルＩＣに対して異なる重みのセット５３２を使用してもよく、各重み行列５３２は、異なる特徴の識別を助けるように設計され得る。実施形態では、畳み込み５００は、長方形の入力行列５０４に長方形の重み行列５３２を乗算して、部分的な内積を取得する。次に、出力画像内の出力ピクセル５１４を表す累積された内積５１４（すなわち整数）を生成するために、部分内積が加算器５４６によって合計され得る。

【００５７】

実施形態では、出力チャネルＯＣの各ピクセルは、乗算器５４２および加算器５４４に

よって生成される。実施形態において、部分的な内積の値は、入力画像５０２の領域５０４への重み行列５３２の全体の適用に対応する。換言すれば、各重み５３２は、乗算器５４２により領域５０４とドット乗算されて部分的な内積を生成し、次いで、部分的な内積がアキュムレータ５４０に累積されて、畳み込みを表す累積出力を生成する。

【００５８】

１つまたは複数の入力チャネルＩＣ、たとえば各色（たとえばＲＧＢ）に１つを使用することができる。例えば、各畳み込みは、各色に１つずつ、３つの異なる行列を表す重み５３２を使用してもよい。各出力チャネルＯＣ５１２は、入力データ５０２の異なる特徴を表す異なるフィルタまたは重み５３２を使用して生成され得る。出力チャネルの数は、機能の数に依存し得る。畳み込みの数は、出力チャネルＯＣの数に入力チャネルＩＣの数を掛けたものに等しく、各畳み込みは、各入力チャネルＩＣに対してＮ個の畳み込みを有し得る。当業者は、入力チャネルの数およびタイプが変化してもよく、色および／またはクリアな入力を含み得ることを認識するであろう。

【００５９】

図５に示すように、入力行列５０４は、３つの入力チャネル、すなわち３×３×ＩＣにわたって３×３の重み行列５３２と組み合わせることができるＫｘ×Ｋｙ（すなわち、３×３）行列であり、深さは一致し、出力プレーンにおいて単一の要素、内積５１４を生成する。出力チャネル５１２の各内積５１４は、ドット乗算の結果である。

【００６０】

図６～図８は、本開示の様々な実施形態による例示的な畳み込み演算の詳細を示している。畳み込み６００は、入力データ行列６０２、重みデータ行列６０４、アレイ６０６、および内積６３０を含む。実施形態では、アレイ６０６は、図２および図３に示されるような行列プロセッサアーキテクチャである。

【００６１】

図６の入力データ行列６０２は、実施形態において、図５に示される矩形入力行列５０４などの入力行列を線形化することにより得られ、入力行列のベクトル化形態を得ることができる列６１０を含む。同様に、重みデータ行列６０４は、図５の矩形重み行列５３２などの重み行列のベクトル化された形式であり得る行６２０を含む。例として、３×３入力行列と３つの入力チャネルとは、入力データ行列６０２で使用するために２７要素列６１０を生成できる３×３×３＝２７要素を含むベクトルに再フォーマットできる。逆に、同じ３つの入力チャネルの３×３重み行列を使用して、重みデータ行列６０４で使用する２７要素の行６２０を生成することができる。当業者は、異なるアプリケーションにわたって入力行列のサイズおよび入力チャネルの数が変化し得ることを認識するであろう。

【００６２】

実施形態において、図５において長方形として描かれた入力チャネルおよび入力重みは、例えば、図２を参照して論じられたフォーマッタによって、行列乗算器／プロセッサ（図２の要素２４０として示されている）に提供されるベクトルフォーマット（例えば、９６要素を有するベクトル）に再フォーマットされ、９６×９６要素の内積演算を並行して実行できる。詳細には、各入力チャネルの長方形として図５に示されている入力データ５０４および入力重み５３２は、ベクトル形式に再フォーマットされる。

【００６３】

実施形態では、図６に入力データ６０２および入力重み６０４（例えば、それぞれ９６個の要素を含む）として示される結果のベクトル形式は、９６×９６要素の内積演算を並列に実行する行列プロセッサまたは行列乗算器２４０に提供される。実施形態では、出力チャネルの計算において、同じ入力データのセットを使用するが重みの異なるセット（すなわち、フィルタ）を使用して同じ出力ピクセルが生成され、入力データを読み取ることで一度に多くの出力チャネルを生成できる。上述のように、入力チャネルの数および出力チャネルの数は任意に選択できることが理解される。

【００６４】

さらに、入力データ行列６０２、重みデータ行列６０４、およびアレイ６０６は、図６

に示されているものとは異なる数の列および行を有し得ることが理解される。特に、入力データ行列６０２および重みデータ行列６０４の形状は、アレイ６０６の任意の調停構成の列および行に対応するようにフォーマットされてもよい。加えて、重みデータ行列６０４が既知の状況では、フォーマッタを使用せずに行６２０を生成し、ベクトル化された形式で保存することができる。

【００６５】

実施形態では、図６の内積６３０は、列６１０に対応するベクトルに行６２０に対応するベクトルをドット乗算することにより生成される。実施形態では、図７に示されるように、次の内積６３２は、列６１２に対応するベクトルと行６２０に対応するベクトルをドット乗算することにより取得され得る。当業者が認識するように、アレイ６０６の第１行のすべての内積が満たされると、アレイ６０６の第２行の内積は、入力データ行列６０２の第１列６１０の要素を重みデータ行列６０４の２行目などとドット乗算することにより計算され得る。

【００６６】

図６～図８は単に例示の目的を果たすだけであり、上述のドット乗算が同時に実行されてワンショット行列－行列乗算演算を生成できることに留意することが重要である。

【００６７】

図９は、本開示の様々な実施形態による例示的な逆畳み込み演算を示している。逆畳み込みシステム９００は、入力画像９０２の入力チャネルＩＣ、重み９２２、内積９０４、９０６、および出力チャネルＯＣを含む。当業者は、逆畳み込み演算９００が、実際には、畳み込み演算、例えば図５に示される畳み込みの数学的転置（ほぼ逆）であることを認識するであろう。当業者はさらに、通常の畳み込みニューラルネットワークに使用される手順と同様の手順を適用することにより、ニューラルネットワークを使用して逆畳み込み演算９００を学習できることを認識するであろう。簡潔にするために、図５のものと同様の構成の説明または機能はここでは繰り返さない。

【００６８】

実施形態では、図９の逆畳み込み演算９００は、重み９２２を使用して内積９０４９０６を分解することにより行列９１２を再構築する。畳み込み演算と同様に、逆畳み込み９００は、各入力チャネルＩＣに対して異なるセットの重み９２２を使用してもよい。実施形態では、逆畳み込み９００を画像に有利に適用して、例えばアーチファクトに対するロバスト性を改善するために画像逆畳み込みを実行することができる。他のアプリケーションは、画像データの分析および復元などを含み得る。

【００６９】

図１０は、本開示の様々な実施形態による畳み込みニューラルネットワークを加速するために算術演算を実行するプロセスを示す。

【００７０】

算術演算を実行するためのプロセス１０００は、データ行列の行を表すことができるオペランドの第１セットが第１論理回路から受信されるとき、ステップ１００２で始まる。このオペランドの第１セットは、オペランドが行列プロセッサへの入力と整列するようにベクトル化できる。特定の実施形態では、ベクトル化されたオペランドのサイズは、軸線に沿った行列プロセッサへの入力の数に直接関係する。

【００７１】

ステップ１００４で、重み行列の列を表し得るオペランドの第２セットが、第２論理回路から受信される。オペランドのこの第２セットは、オペランドが行列プロセッサへの対応する入力内で整列するようにベクトル化される。特定の実施形態では、ベクトル化されたオペランドのサイズは、異なる軸線に沿った行列プロセスへの入力の数に直接関係する。

【００７２】

ステップ１００６で、オペランドの第１セットは、オペランドの第２セットとドット乗算され、１つ以上の内積を取得する。特定の実施形態では、オペランドのセットにわたるこのセット操作は、単一のクロックサイクルで実行される。

【００７３】
ステップ１００８で、内積を使用して、画像をフィルタで畳み込み、畳み込み結果を生成することができる。
【００７４】
ステップ１０１０で、畳み込み結果がさらに処理されて、画像出力が強化される。このさらなる処理は、非線形関数、正規化操作、またはプーリング操作を使用して発生し得る。
【００７５】
当業者は、計算システムまたはプログラミング言語が本発明の実施に重要ではないことを認識するであろう。当業者はまた、上記の多くの要素が物理的および／または機能的にサブモジュールに分離されるか、または一緒に結合され得ることを認識するであろう。
【００７６】
以下の請求項の要素は、複数の依存関係、構成、および組み合わせを含むなど、さまざまに配置できることに留意されたい。例えば、実施形態において、様々な請求項の主題は他の請求項と組み合わされてもよい。
【００７７】
前述の例および実施形態は例示であり、本発明の範囲を限定するものではないことを当業者は理解するであろう。明細書を読み、図面を検討することで当業者に明らかなすべての置換、強化、同等物、組み合わせ、および改良は、本発明の真の精神および範囲内に含まれることが意図されている。

【図面】

【図１】

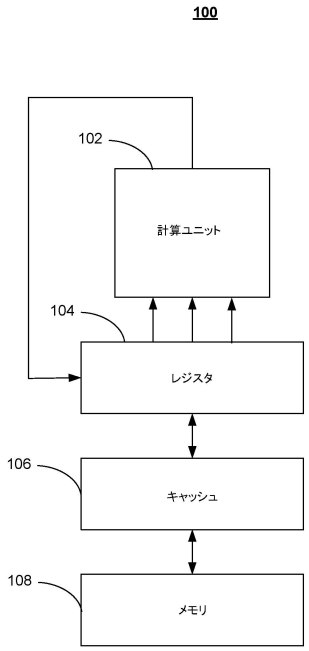


FIGURE 1
（従来技術）

【図２】

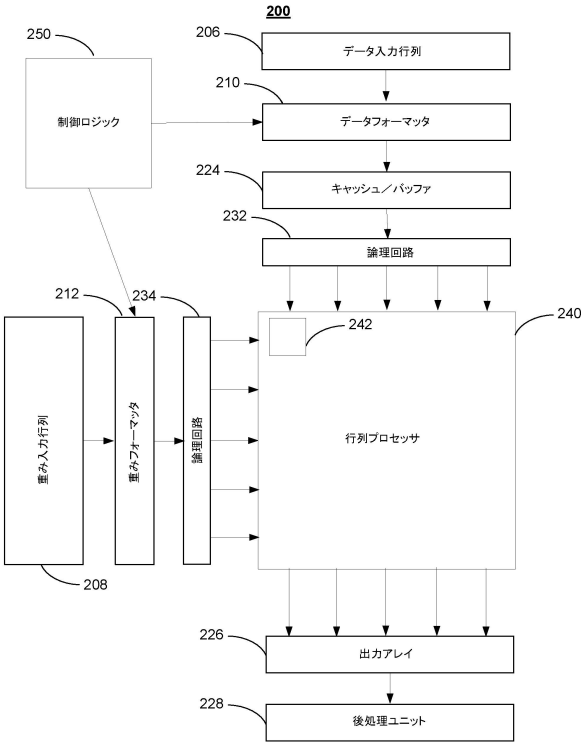


FIGURE 2

【図３】

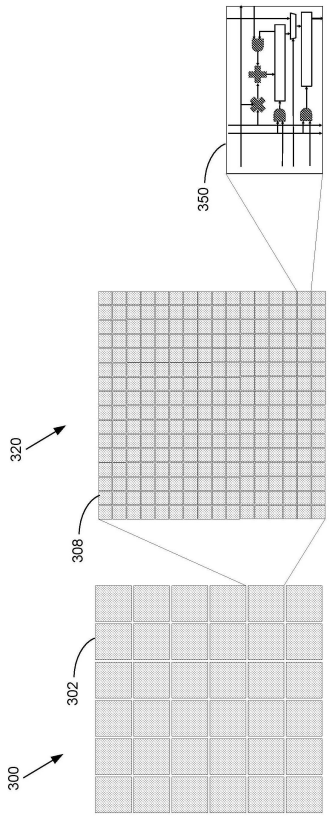


FIGURE 3

【図４】

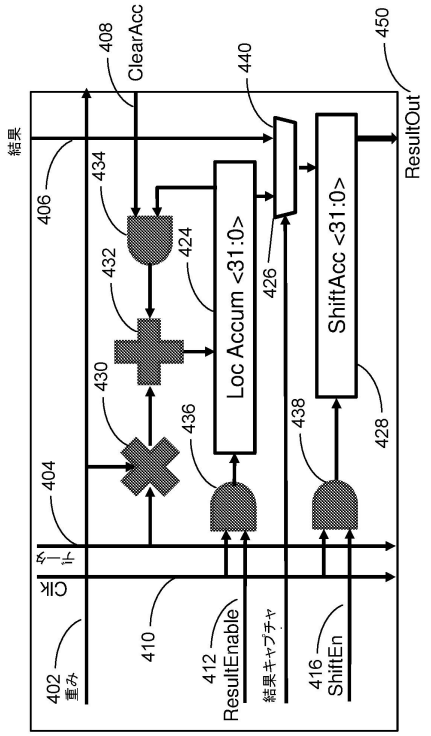


FIGURE 4

【図５】

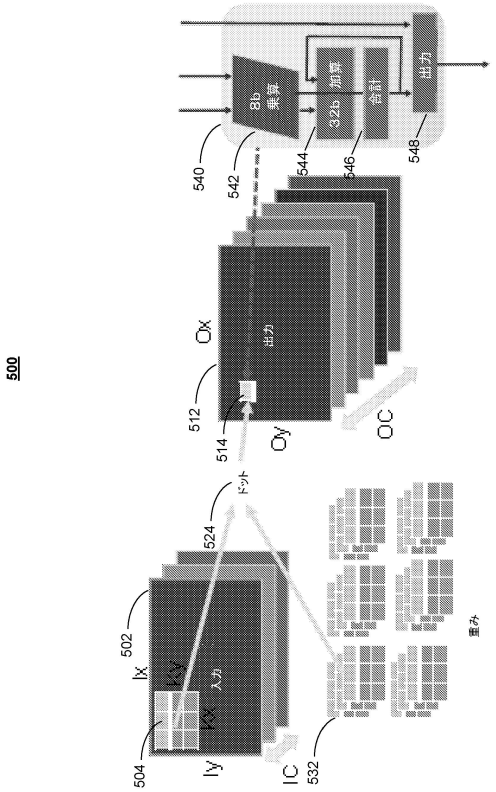


FIGURE 5

【図６】

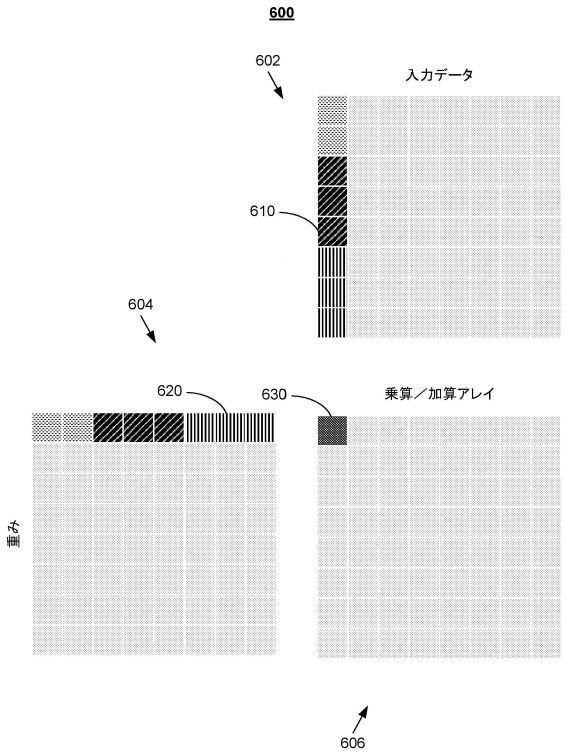


FIGURE 6

【図７】

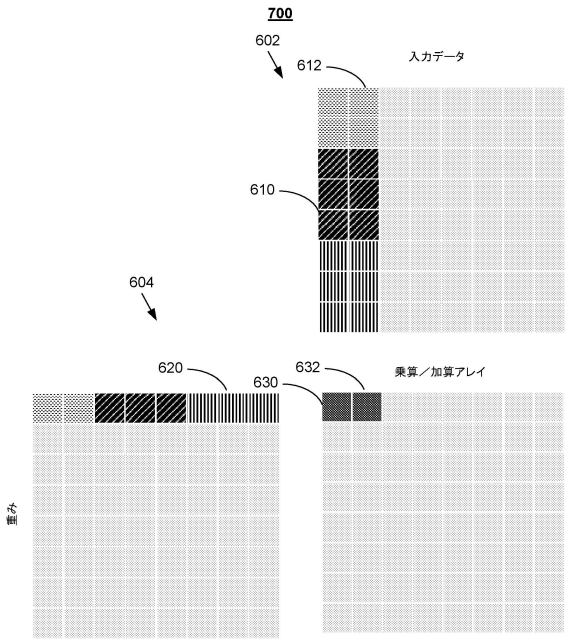


FIGURE 7

【図８】

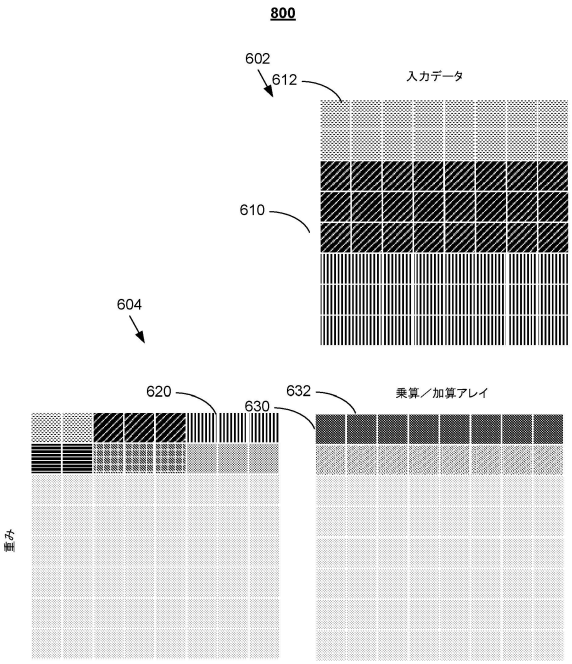
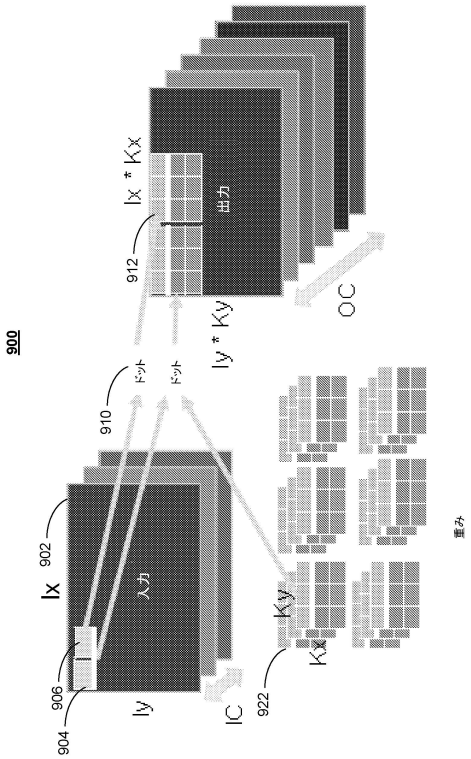


FIGURE 8

【図９】



【図１０】

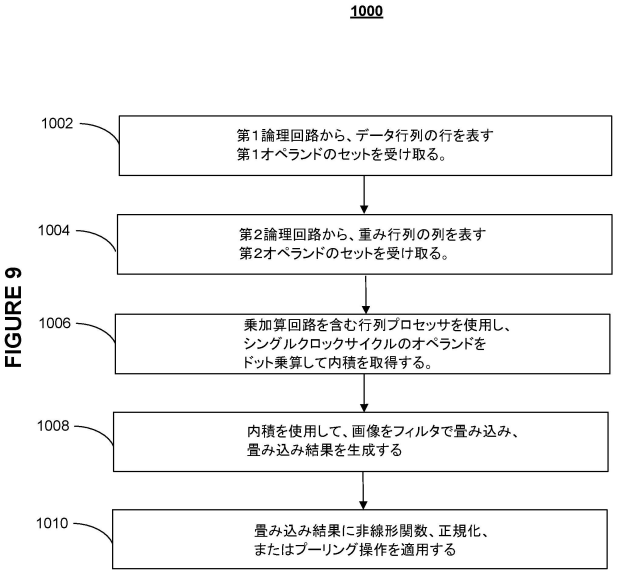


FIGURE 9

FIGURE 10

フロントページの続き

米国(US)
アメリカ合衆国 カリフォルニア州 ９４４０３，サン マテオ，バーバンク アベニュ １６０

審査官 坂庭 剛史

(56)参考文献 米国特許出願公開第２０１７／００９７８８４（ＵＳ，Ａ１）
米国特許出願公開第２０１６／０３４２８９１（ＵＳ，Ａ１）
特開２０１０－０７９８４０（ＪＰ，Ａ）
特開２０１５－０５６１２４（ＪＰ，Ａ）
米国特許出願公開第２０１６／０３７９１０９（ＵＳ，Ａ１）

(58)調査した分野 (Int.Cl.，ＤＢ名)
Ｇ０６Ｆ １７／１６
Ｇ０６Ｎ ３／０６３