(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 994 057**

(51) Int. Cl.:

G06F 7/544 (2006.01)
G06F 7/483 (2006.01)
G06F 7/76 (2006.01)
G06F 7/02 (2006.01)
G06N 3/063 (2013.01)
G06N 3/08 (2013.01)
G06N 20/00 (2009.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **06.01.2017 E 23163158 (1)**

(97) Fecha y número de publicación de la concesión europea: **14.08.2024 EP 4220380**

(54) Título: **Aprendizaje máquina acelerado por hardware**

(30) Prioridad:

07.01.2016 US 201662276169 P
05.01.2017 US 201715399714

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
16.01.2025

(73) Titular/es:

INTEL CORPORATION (100.00%)
2200 Mission College Blvd.
Santa Clara, CA 95054, US

(72) Inventor/es:

BRUESTLE, JEREMY y
NG, CHOONG

(74) Agente/Representante:

LEHMANN NOVO, María Isabel

DESCRIPCIÓN

Aprendizaje máquina acelerado por hardware

ANTECEDENTES

El aprendizaje máquina es un subcampo de la informática que se centra en algoritmos que pueden aprender y hacer predicciones basadas en datos. El aprendizaje máquina se basa en gran medida en las redes neuronales para su puesta en práctica y se volvió práctico para un uso amplio con el advenimiento de la informática en la nube, que hizo económico el acceso a grandes cantidades de poder de procesamiento computacional. De hecho, la informática en la nube y, en general, el procesamiento informático económico ha hecho que las variaciones del aprendizaje máquina sean más costosas desde el punto de vista computacional y que requieran más recursos, tales como las redes neuronales profundas que implican la puesta en práctica de capas de redes accesibles desde el punto de vista económico.

En consecuencia, el aprendizaje máquina se realiza, por lo general, utilizando unidades centrales de procesamiento ("CPU") de uso general, unidades de procesamiento gráfico ("GPU") o hardware de procesamiento de uso general masivo. Se han propuesto procesadores de aprendizaje máquina en donde la aceleración por hardware se basó en técnicas orientadas a un algoritmo específico y/o arquitectura de red. De manera alternativa, se ha propuesto hardware de aprendizaje máquina basado en modelización del hardware con mayor proximidad a la arquitectura neurológica. En estos casos, en donde la aceleración estaba basada en el supuesto de que una arquitectura neurológica proporcionaría, de manera inherente, mejoras en el rendimiento, en lugar de realizar la aceleración por hardware en las instrucciones del ordenador y el procesamiento de dichas instrucciones del ordenador.

El documento de CHEN TIANSHI CHEN ET AL: "DianNao", ACM SIGARCH COMPUTER ARCHITECTURE NEWS, GRUPO DE INTERÉS ESPECIAL DE ACM SOBRE ARQUITECTURA INFORMÁTICA, 2 PENN PLAZA, SUITE 701 NEW YORK NY 10121-0701 USA, vol. 42, nº 1, de fecha 24 de febrero de 2014 (24-02-2014), págs. 269-284, ISSN: 0163-5964, DOI: 10.1145/2654822.2541967 es un ejemplo de arquitectura neurológica.

BREVE DESCRIPCIÓN DE LOS DIBUJOS

La descripción detallada se expone con referencia a las figuras adjuntas.

La Figura 1 es un diagrama de contexto de nivel superior para el aprendizaje máquina acelerado por hardware.

La Figura 2 es un diagrama de bloques de una arquitectura, a modo de ejemplo, para el aprendizaje máquina acelerado por hardware.

La Figura 3 es un diagrama de flujo para una operación, a modo de ejemplo, de un registro de desplazamiento para el aprendizaje máquina acelerado por hardware.

La Figura 4 es un diagrama de flujo para una operación, a modo de ejemplo, para obtener y ejecutar instrucciones legibles por ordenador en un acelerador por hardware de aprendizaje máquina.

DESCRIPCIÓN DETALLADA

Contexto del aprendizaje máquina acelerado por hardware

Tal como se indicó con anterioridad, el aprendizaje máquina es un subcampo de la ciencia informática que se centra en algoritmos que pueden aprender y realizar predicciones basadas en datos. Por lo general, estos algoritmos adoptan la forma de métodos que procesan datos, a modo de ejemplo, y luego hacen generalizaciones a datos similares. A menudo, estos métodos adoptan la forma de regresión o clasificación estadística, aunque el campo también incluye métodos tales como el aprendizaje por refuerzo que modelan el comportamiento de aprendizaje de la interacción de un agente que busca recompensas con el entorno.

Recientemente, un conjunto de métodos, concretamente denominados "aprendizaje profundo", se han vuelto comunes en múltiples usos comercialmente pertinentes. Estos métodos se basan en la investigación de redes neuronales artificiales, aunque en la mayoría de los casos, las técnicas modernas de aprendizaje profundo no intentan modelar directamente aspectos conocidos de la cognición biológica. En cambio, utilizan la idea de una serie de "capas" de "unidades de procesamiento", cada una de las cuales se compone concretamente de una operación lineal seguida de una no linealidad. Estas capas también pueden incluir procesos aleatorios y, por lo general, aunque no universalmente, están densamente conectadas.

Estos métodos pueden incluir perceptrones multicapa, redes convolucionales, memoria a corto plazo, aprendizaje Q profundo, máquinas de Boltzmann restringidas y muchos otros métodos. Si bien estos métodos difieren, en muchos aspectos, comparten una estructura lógica común de capas que procesan entradas y generan salidas, posiblemente recurrentes en el tiempo y posiblemente utilizando ruido estocástico. También pueden utilizar el concepto de una "función de pérdida" que representa para un conjunto dado de datos lo bien que está funcionando la red. Por lo general, esta función de pérdida se descompone aún más en una función de utilidad que representa la pérdida relativa al

conjunto de datos a modo de ejemplo, así como una pérdida de regularización que busca minimizar la complejidad y mejorar el rendimiento de la generalización. Por lo general, pero no de manera universal, estos sistemas aprenden a través del descenso de gradiente estocástico, a menudo con extensiones adicionales.

De manera independiente, tanto el entrenamiento de redes de aprendizaje profundo como el uso de redes entrenadas se pueden dividir en un patrón muy simple de bloques de construcción computacionales fundamentales. Estas son contracciones de tensores generalizadas, ruido aditivo y no linealidades por elementos. Aunque no siempre se abstrae de esta manera, se puede lograr una amplia variedad de métodos de aprendizaje profundo, así como otro aprendizaje máquina o cálculo informático general mediante estos primitivos.

Estos primitivos pueden ponerse en práctica en términos de una arquitectura típica de unidad central de procesamiento ("CPU"), o pueden ponerse en práctica mediante una unidad de procesamiento gráfico ("GPU"), que es un tipo de procesador vectorial que se suele utilizar para cargas de trabajo de gráficos. Sin embargo, el flujo de datos constante, la gran cantidad de multiplicaciones y la naturaleza especializada de las operaciones que no son de multiplicación permiten una aceleración significativa a través del hardware. Se han propuesto métodos de aceleración por hardware, muchos de los cuales son altamente específicos para un algoritmo particular de aprendizaje máquina o arquitectura de red, o son neuromórficos, ya que buscan duplicar el comportamiento de las redes biológicas, en lugar de poner en práctica algoritmos de aprendizaje profundo.

A diferencia de dichos métodos, se propone un método para el aprendizaje máquina acelerado por hardware que no es específico de un algoritmo particular de aprendizaje máquina o arquitectura de red y no es neuromórfico. Conviene señalar que los flujos de trabajo típicos de aprendizaje máquina comprenden tres operaciones básicas: contracciones de tensor, adición de ruido aleatorio y operaciones no lineales por elementos. De las operaciones que anteceden, las contracciones de tensor suelen constituir la mayor parte del trabajo computacional en las arquitecturas informáticas tradicionales. En consecuencia, proponemos una arquitectura informática para realizar estas operaciones junto con un conjunto de instrucciones informáticas asociadas. La Figura 1 es un diagrama de contexto 100 de dicha arquitectura.

La aceleración por hardware de aprendizaje máquina por lo general se encuentra en el contexto de un ordenador 102. El ordenador 102 puede ser cualquier dispositivo informático que incluya un ordenador independiente, un servidor en red o un sistema integrado específico para el aprendizaje máquina. El ordenador 102 suele tener una unidad central de procesamiento ("CPU") que es cualquier procesador que ejecuta instrucciones legibles por ordenador que varían desde la CPUs de uso general hasta los controladores integrados.

La CPU está acoplada, de manera comunicativa, mediante un bus a una memoria general 106. La memoria general 106 puede estar compuesta por una memoria de acceso aleatorio dinámica ("DRAM"), pero este no tiene por qué ser siempre el caso. La memoria general 106 es cualquier medio legible por ordenador y puede almacenar un sistema operativo y otras aplicaciones, cada una de las cuales consta de instrucciones legibles por ordenador.

Los medios legibles por ordenador incluyen, al menos, dos tipos de medios legibles por ordenador, a saber, medios de almacenamiento informático y medios de comunicación. Los medios de almacenamiento informático incluyen medios volátiles y no volátiles, extraíbles y no extraíbles puestos en práctica en cualquier método o tecnología para el almacenamiento de información, tales como instrucciones legibles por ordenador, estructuras de datos, módulos de programas u otros datos. Los medios de almacenamiento informático incluyen, entre otros, memoria RAM, memoria ROM, memoria EEPROM, memoria instantánea u otra tecnología de memoria, CD-ROM, discos versátiles digitales (DVD) u otro almacenamiento óptico, casetes magnéticos, cinta magnética, almacenamiento en disco magnético u otros dispositivos de almacenamiento magnético, o cualquier otro medio que no sea de transmisión que pueda utilizarse para almacenar información para el acceso de un dispositivo informático. Por el contrario, los medios de comunicación pueden incorporar instrucciones legibles por ordenador, estructuras de datos, módulos de programa u otros datos en una señal de datos modulada, tal como una onda portadora u otro mecanismo de transmisión. Tal como se define en el presente documento, los medios de almacenamiento informático no incluyen los medios de comunicación.

La interfaz de entrada/salida ("E/S") 108 facilita la interacción del ordenador 102 con otros dispositivos. La interfaz de E/S 108 puede ser cualquier tarjeta de controlador, tal como un receptor/transmisor asíncrono universal (UART) utilizado junto con un protocolo de interfaz de E/S estándar tal como RS-232 y/o Bus Serie Universal (USB). En el caso de aplicaciones altamente paralelas, tales como el aprendizaje máquina, la interfaz de E/S, puede proporcionar uno o más canales de E/S y/o canales de E/S en paralelo.

La interfaz de red 110 puede funcionar de manera potencial en adaptación con la interfaz de E/S 108 y puede ser una tarjeta de interfaz de red compatible con Ethernet y/o Wi-Fi y/o cualquier número de otros protocolos físicos y/o de enlace de datos. De nuevo, en el caso de aplicaciones altamente paralelas tales como el aprendizaje máquina, el ordenador 102 puede admitir múltiples interfaces de red 110.

El ordenador 102 puede albergar uno o más aceleradores de hardware de aprendizaje máquina 112. El acelerador por hardware 112 se compone de múltiples bancos de memoria de ordenador 114, tal como la memoria de acceso aleatorio estática ("SRAM"). Las memorias SRAMs 114 soportan el acceso aleatorio concurrente. En una forma de

realización, Las memorias SRAMs 114 admiten un acceso de memoria muy amplio (por ejemplo, acceso de múltiples bytes dentro de una única instrucción informática), lo que permite acceder a vectores y conjuntos más grandes de valores en una única instrucción informática. Conviene señalar que con respecto al acelerador por hardware 112, los bancos de memoria de ordenador 114 son locales pero la memoria general 106 no es local. A veces, el acelerador por hardware 112 puede intercambiar datos locales y no locales. Dichas transferencias se describen a continuación.

Las memorias SRAMs 114 son objeto de acceso y funcionamiento mediante múltiples unidades operativas 116. Las unidades operativas 116 son capaces de realizar de manera simultánea operaciones de cálculo y admiten un conjunto de instrucciones específicas de operaciones de aprendizaje máquina. Las unidades operativas 116 se prestan a muchas optimizaciones de aprendizaje máquina. En una forma de realización, las unidades operativas 116 pueden configurarse para realizar operaciones de punto fijo de menor precisión. Por ejemplo, la norma IEEE de 32 bits especifica 23 bits explícitos para la mantisa de un número de coma flotante. Una operación de coma flotante de menor precisión tendría menos de 23 bits para representar la mantisa. Concretamente, aunque la mayoría de las investigaciones de aprendizaje profundo se han realizado utilizando aritmética de coma flotante de precisión de 32 bits para representar números reales, los hallazgos recientes han demostrado que, para la mayoría de las cargas de trabajo de aprendizaje profundo, se pueden utilizar métodos de punto fijo y precisión mucho más bajos (menos de 32 bits) particularmente si se utiliza una ronda aleatoria. Las unidades operativas 116 se describen con mayor detalle a continuación.

Una unidad de transferencia SRAM/DRAM 118 facilita la transferencia rápida de datos entre la memoria general 106 y los bancos de memoria SRAM 114. Concretamente, la unidad de transferencia SRAM/DRAM 118 admite operaciones de copia masiva de modo que las transferencias de datos grandes (multibyte) se puedan realizar en una operación única.

Un acelerador por hardware de aprendizaje máquina 112 contiene características adicionales que se describen con mayor detalle con respecto a la Figura 2 siguiente y en otro lugar en esta descripción.

Definiciones y notación

En general, tanto los datos de entrada (es decir, ejemplos para aprendizaje o accionamiento), así como los datos de ponderación (es decir, parámetros de red), se representan de manera natural como tensores multidimensionales. Por lo general, a los efectos de los casos de uso de aprendizaje máquina, estos tensores no tienen una noción de índices covariantes o contravariantes. Además, los tensores simétricos pueden ser pertinentes, pero dicha simetría a menudo no se formaliza o se mantiene explícitamente mediante las operaciones, en lugar de ser un aspecto del tensor en sí mismo, lo que hace que el uso de tensores en el aprendizaje máquina sea más similar a las matrices multidimensionales. Además, las contracciones tienen algunas propiedades ligeramente variantes. Por ejemplo, las convoluciones a menudo no coinciden exactamente con las nociones típicas de cálculo tensorial de una contracción, pero son bastante similares desde el punto de vista computacional. De este modo se define aquí una noción un poco más computacional de tensores y contracciones.

Tensor

Un tensor se define aquí como una matriz rectangular multidimensional de números reales, que se suele representar por una representación de punto fijo de precisión relativamente baja (tal como 16 bits en lugar de 32 bits). Por lo general, los datos de tensor se almacenan en una memoria de acceso aleatorio dinámica ("DRAM") y tienen un diseño de base de etapa simple, o se componen de un conjunto de "mosaicos", cada uno de los cuales tiene un diseño basado en etapa.

Contracción del tensor

Para los fines del presente documento, una contracción de tensor se define como una operación que toma dos tensores de entrada y obtiene un tensor de salida, mediante un conjunto de operaciones de multiplicación y de acumulación. Las contracciones de tensor se definen mediante un conjunto de "variables de índice". El margen de cada variable de índice es un número entero simple y, a menudo, pero no siempre, es simplemente el tamaño de una de las dimensiones del tensor, y el margen de los índices del tensor se puede especificar de forma independiente. En particular, una contracción tensorial adopta la forma:

$$O[i, j, ...] = A[P, Q, ...] * B[X, Y, ...]$$

En este caso, los índices en minúsculas, i, j, etc., representan un índice de tensor específico, las letras mayúsculas fuera de los corchetes representan tensores, y las letras mayúsculas entre corchetes son polinomios multivariados de grado 1 o 0 sobre los índices de tensores establecidos en toda su generalidad, pero por lo general también son solamente un índice de tensor único. Se supone que la operación recorre el margen de todos los índices no especificados en la salida y suma los resultados de la multiplicación de los elementos tensoriales izquierdo y derecho de forma similar a la notación de Einstein. Por ejemplo, una operación de multiplicación matricial se representaría en esta notación como:

$$O[i, j] = A[i, k] * B[k, j]$$

Una convolución de una imagen tridimensional (compuesta por dimensiones X, Y y C, es decir, ubicación de píxeles horizontal y vertical y un canal de color o característica) con un núcleo convolucional de cuatro dimensiones que funciona sobre un parche de imagen en todos los canales, se representaría como:

$$O[x, y, co] = A[i, j, co, ci] * B[x + i, y + j, ci]$$

En este caso, se observa el uso de un polinomio simple (x + i). Conviene señalar que se debe especificar el margen de x, y, i, j, co, etc., para que esta sea una descripción completa de la operación a realizar. En muchos casos, la opción lógica es suponer que el margen de un índice es idéntico a la magnitud de las dimensiones sobre las que opera, pero esto no es aplicable de manera universal.

También podemos generalizar tanto la operación realizada para cada par de elementos tensoriales (en este caso, la multiplicación) así como el método de acumulación de elementos (en este caso, la sumatoria). Cuando el mecanismo de acumulación no es una suma, lo especificamos como en el siguiente ejemplo, que pone en práctica "*max-pooling*", y también ilustra el uso de un tensor "constante" que es de dimensión 0 y contiene un único elemento (en este caso 1):

$$O[x, y] = \text{máx. sobre } A[x + i, y + j] * B[0]$$

Para una agrupación de 2x2 máx., i y j oscilarían entre [0, 2).

Parámetros

El acelerador por hardware de aprendizaje máquina descrito en este documento es configurable. Para proporcionar ejemplos, para una mayor comprensión, se proporciona un valor típico, así como cualquier restricción probable para algunos parámetros de diseño. Por supuesto, puesto que el acelerador por hardware descrito solamente se entiende como un ejemplo concreto de un método para poner en práctica las operaciones fundamentales descritas mediante una puesta en práctica de hardware eficiente, y estos valores no deben interpretarse como limitantes.

SRAM_BANKS: Número de bancos primarios de memoria de acceso aleatorio estática ("SRAM"), mínimo de 2, concretamente 8.

ELEM_WIDTH: Ancho de bits de un elemento, normalmente 16, mínimo 2.

ACCUM_WIDTH: El ancho de un acumulador, normalmente 48, debe ser mayor o igual a 2 * ELEM_WIDTH.

SRAM_WIDTH: Número de elementos objeto de lectura durante un acceso único a memoria SRAM de un banco único, concretamente 512. Por lo general una potencia de 2.

SRAM_DEPTH: Número de filas de SRAM, cada una de las cuales contiene bits SRAM_WIDTH * ELEM_WIDTH. Por lo general, 4096, lo que da como resultado un tamaño total de SRAM de SRAM_BANKS * ELEM_WIDTH * SRAM_WIDTH * SRAM_DEPTH/8 = 20 MB.

SHIFT_WIDTH: Número de elementos que son la salida de los bancos SRAM después de la unidad de rotación, debiendo ser <= a (SRAM_BANKS - 1) * SRAM_WIDTH, concretamente 2048. Por lo general una potencia de 2.

NUM_UNITS: Número de unidades operativas, debe ser >= a SHIFT_WIDTH, concretamente significativamente mayor, por ejemplo, 131072. Por lo general una potencia de 2.

INDEX_WIDTH: Ancho de un registro de índice, concretamente 32 bits.

NUM_INDEXES: Número de registros de índice, normalmente 8.

Conviene señalar que los anchos de memoria, en particular SRAM_WIDTH, son relativamente grandes. Lo que antecede es, para permitir el procesamiento de elementos tensoriales en una operación única. Concretamente, los grandes anchos de memoria disponibles en un acceso único permiten acceder a vectores y elementos tensoriales completos durante dicha operación única. De no ser así, se utilizarían múltiples accesos y múltiples operaciones, lo que ralentizaría la operación de los cálculos de tensor.

Componentes para un acelerador por hardware de lenguaje máquina

La Figura 2 es un diagrama de bloques 200 de un acelerador por hardware de lenguaje máquina 112. Un acelerador por hardware de lenguaje máquina se pone en práctica concretamente como puertas lógicas dentro de un único circuito integrado. En algunas formas de realización, un acelerador por hardware de lenguaje máquina 112 puede ponerse en práctica en un dispositivo de matriz de puertas programables en campo ("FPGA") o equivalente. En producción, un acelerador por hardware de lenguaje máquina 112 puede ponerse en práctica como un circuito integrado personalizado compuesto por múltiples puertas lógicas.

Un acelerador por hardware de lenguaje máquina 112 se compone de los siguientes componentes:

- Memoria SRAM de datos 114
- Lector de desplazamientos 202
- Red de multidifusión 204
- Unidades operativas 116
- Registros de índice (direccionamiento) 206
- Unidad de transferencia DRAM 118
- Decodificador de instrucciones 208

Estos componentes se describen de manera individual con mayor detalle a continuación.

Memoria SRAM de datos

El acelerador por hardware de aprendizaje máquina mantiene la mayoría de los datos para operar en múltiples bancos de SRAM (concretamente de doble puerto). Existe un número de bancos SRAM_BANKS, cada uno con filas SRAM_DEPTH, compuesto cada uno por bits SRAM_WIDTH * ELEM_WIDTH, que se ven lógicamente como elementos SRAM_WIDTH de magnitud de anchura de elementos ELEM_WIDTH. SRAM se ve lógicamente como un único espacio de direcciones lineales que consta de elementos de magnitud de bits ELEM_WIDTH, con un total de elementos SRAM_BANKS * SRAM_WIDTH * SRAM_DEPTH. La ubicación lógica de SRAM i, denominada M[i] a continuación, existe en una ubicación calculada por:

$$LR = \text{floor}(i / SRAM\_WIDTH)$$

$$LO = i - LR$$

$$B = LR \,\%\, SRAM\_BANKS$$

$$R = LR / SRAM\_BANKS$$

$$O = LO * ELEM\_WIDTH$$

en donde LR representa la "fila lógica", LO representa el "compensación de elemento lógico" dentro de la fila, y B es el número de banco físico, R es el número de fila física dentro de esa parte posterior y O es la compensación binaria dentro de esa fila del primer bit del elemento, comprendiendo los siguientes bits ELEM_WIDTH la totalidad del elemento. (Conviene señalar que el operador % es el operador de módulo).

A continuación, se presentan alternativas a los métodos anteriores junto con otras modificaciones.

Lector de desplazamiento

El lector de desplazamiento 202 proporciona lógica de direccionamiento y una rotación cilíndrica y un desplazador correspondiente para permitir el acceso lógico de SHIFT_WIDTH de elementos lógicamente contiguos que comienzan en cualquier compensación lógica, A, dentro de la memoria SRAM. La Figura 3 es un diagrama de flujo 300 del funcionamiento del lector de desplazamientos 202 dada la dirección del primer elemento.

En el bloque 302, para cada banco B, se debe realizar, de manera simultánea, una lectura de la base de la fila ((A + B * SRAM_WIDTH)/SRAM_WIDTH).

En el bloque 304, se deben concatenar las entradas SRAM_BANKS * SRAM_WIDTH en una única lista lógica de entradas.

En el bloque 306, girar los resultados a la izquierda en elementos (floor(A/SRAM_WIDTH) % SRAM_BANKS) * SRAM_WIDTH.

Desplazar los resultados a la izquierda en elementos A % SRAM_WIDTH.

Mediante una canalización adecuada, lo que antecede da lugar a una lectura de elementos contiguos de SHIFT_WIDTH desde cualquier dirección lógica por ciclo de reloj. Al duplicar el direccionamiento y la lógica de rotación/desplazamiento, para memoria SRAM de doble puerto, lo que antecede, por lo general, se extendería hasta dos lecturas independientes de elementos contiguos de SHIFT_WIDTH por ciclo de reloj.

Red de multidifusión

Volviendo a la Figura 2, la red de multidifusión 204 proporciona un mecanismo general para permutar y duplicar las entradas que son objeto de lectura por el lector de desplazamientos 202 y aplicarlas a las unidades operativas 116. Concretamente, sobre una base por instrucción, a cada unidad operativa 116 se le proporcionará un único elemento desde la red de multidifusión 204, y cada uno de estos elementos deberá proceder de una entrada dentro de las salidas SHIFT_WIDTH del lector de desplazamiento asociado 202, o debe ser 0. El reordenamiento y la duplicación realizados por la unidad de multidifusión 204 no son fijos, y puede variar según el software. La modificación se realiza

mediante un mecanismo de "precarga" mediante el cual se carga un patrón de multidifusión en la red de multidifusión 204, entre grupos de operación normal, o mediante información de "patrón" adicional presente en cada instrucción, o alguna combinación, lo que permite potencialmente una variabilidad por instrucción de la multidifusión. En el uso típico, habrá dos redes de multidifusión independientes 204, cada una conectada a un lector de desplazamientos independiente 202, lo que permitirá enviar dos elementos a cada unidad operativa para cada ciclo de reloj. Múltiples alternativas se contemplan explícitamente a continuación, y aún más alternativas son posibles.

Por lo general, con respecto a las contracciones de tensor, cada salida se mantiene en el acumulador de una unidad operativa 116, y el objetivo de la red de multidifusión 204 es enrutar los dos operandos de la operación de tensor (concretamente multiplicación) desde los tensores de entrada o mosaicos a la unidad operativa apropiada.

A. Red de multidifusión de operación binaria

Una red de multidifusión de operación binaria 204 consta de una serie de capas, cada una de las cuales pasa a través de los elementos de manera idéntica o realiza una permutación fija o una operación en abanico, en función de un bit, de toda la capa, especificado en la instrucción que se ejecuta en ese momento. Las primeras capas son de una anchura de SHIFT_WIDTH elementos y, mediante disposiciones en abanico, se extienden hasta que las capas tienen una anchura de NUM_UNITS elementos. En la puesta en práctica, se prevé que una capa sea un conjunto de trazas fijas con multiplexores de un bit de ancho de capa *ELEM_WIDTH con una línea de selección compartida. Además, se prevé que, para cumplir con los requisitos de temporización, es posible que sea necesario insertar capas de memoria intermedia en partes de la red de multidifusión 204. En este caso, la ejecución de instrucciones se canalizará de modo que la línea de selección para capas posteriores y la operación que llega en las unidades operativas sea retardada de forma adecuada en relación con la lectura de la memoria original.

La elección del número de capas y las permutaciones fijas a utilizar se basa en proporcionar la máxima flexibilidad entre los flujos de trabajo de aprendizaje máquina típicos, concretamente, las contracciones de tensor. Puesto que la salida se considera un mosaico de un tensor, al redondear todos los tamaños de mosaicos a una potencia de dos, los bits del índice binario de una unidad operativa pueden considerarse un conjunto de coordenadas, en donde cada coordenada comprende un número contiguo de bits dentro del índice. Esta limitación, sobre los tamaños de las dimensiones del tensor, simplifica en gran medida los tipos de reorganización de datos necesarios.

Es probable que los tensores de entrada elijan utilizar solamente algunas de estas coordenadas, posiblemente con una compensación (es decir, la adición del índice de tensor que aparece solamente en el lado derecho). Además, es probable que la entrada elija difundir el mismo valor mediante varias coordenadas de salida, en los casos en que las coordenadas del tensor, de una entrada determinada, sean independientes de uno de los índices de salida.

Por ejemplo, imaginando que SHIFT_WIDTH es 16, a cada entrada que llega desde el rotador de desplazamiento se le puede dar un índice en el margen [0,16), que se puede representar en binario como 4 bits (primero el bit bajo) como (i0, i1, i2, i3). Imaginando NUM_UNITS como 256, las unidades de salida pueden recibir un índice similar (o0, o1, o2… o7).

Ahora, se deberá imaginar que se desea tomar un tensor I de 2x2 de la entrada y aplicarlo a un tensor O de 2x2x2x2, tal que O[w,x,y,z] recibe la entrada desde A[x,y].

Lo que antecede se puede hacer proporcionando a la salida (o0, o1...) el valor de la entrada (o2, o3, o4, o5). Es decir, existe una relación entre qué elemento de entrada se envía a qué unidad operativa y los bits de los índices del elemento de entrada y la unidad operativa asociados.

Para realizar estas operaciones, se utilizan dos conjuntos de capas. El primer conjunto de capas, denominado capas de permutación, está diseñado para permutar elementos de manera que sean de utilidad para los flujos de trabajo de aprendizaje máquina. El segundo conjunto de capas, denominado capas de duplicación, duplica las entradas y proporciona un despliegue en abanico.

El primer conjunto de capas consta de una serie de "mezclas" que reorganizan lógicamente los bits del índice para permitir operaciones de transposición en mosaicos que tienen potencia de dos tamaños, así como para permitir entradas de "empaquetado" para varias acumulaciones. Dicho de otro modo, la primera capa permuta las entradas, de forma que los bits, que componen sus índices, sean también permutados.

Para describir los detalles de esta capa, se comienza definiendo una operación de permutación, parametrizada en un número B que está comprendido entre 0 y SHIFT_WIDTH-2 inclusive. Esta operación traslada el bit B+1 al bit inferior y desplaza todos los bits posteriores al bit B+1 a la izquierda, dejando los bits anteriores a B+1 tal como están.

Esta permutación se describe mediante el siguiente código Python:

```
def do_perm(x, B):
    N = len(x)
    P = 2 ** (B+1)
    P2 = 2 ** (B+2)
    return [x[(i/P2)*P2 + 2*(i%P) + ((i%P2)/P)] for i in range(N)]
```

Utilizamos 2*(SHIFT_WIDTH-1) capas, cada una de las cuales realiza una permutación específica, o realiza una permutación de identidad. Las permutaciones comienzan con B = SHIFT_WIDTH-2 y bajan a 0, y luego repiten el mismo patrón de nuevo. Describiendo todo este procedimiento en código Python, se escribe lo siguiente:

```
def do_perms(x, bits):
    B = int(math.log(len(x), 2)) # Número de bits
    L = 2*(B-1)
    for i in range(L):
        if bits[i]:
            x = do_perm(x, B - (i%(B-1)) - 2)
        else:
            x = x
    return x
```

Para un ejemplo simple de una transposición de un tensor 4×4, en donde SHIFT_WIDTH = 4, se desea realizar las permutaciones 0, 2 y 1 en ese orden. Puesto que las opciones de permutación son [2, 1, 0, 2, 1, 0], se realizarán permutaciones en el patrón de bits [0, 0, 1, 1, 1, 0]. Utilizando el código ejemplo anterior se muestra:

```
x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
print do_perms(x, [0, 0, 1, 1, 1, 0])
```

De las cuales, son salidas

[0, 4, 8, 12, 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 15]

Si observamos la X original como un tensor 4x4, esto es claramente una transposición de dicho tensor.

En general, es posible reorganizar los bits para solicitar numerosos tipos de permutaciones de tensor. Para tensores bidimensionales, lo que antecede permite todas las permutaciones. Para tensores tridimensionales, esto admite 5 de las 6 permutaciones posibles. Para tensores de cuatro dimensiones, este método admite 6 de las 12 permutaciones posibles, concretamente, las que se enumeran a continuación:

0 1 2 3

0 1 3 2

0 2 1 3

0 2 3 1

0 3 1 2

1 0 2 3

1 2 0 3

1 2 3 0

1 3 0 2

2 0 1 3

2 3 0 1

3 0 1 2

Las permutaciones no admitidas pueden ponerse en práctica a costa de ciclos adicionales haciendo contracciones contra un tensor constante para poner en práctica parte de la permutación, seguido de una segunda operación para poner en práctica la permutación restante. Sin embargo, mediante la disposición cuidadosa del orden de las dimensiones del tensor mediante software, lo que antecede puede evitarse en la gran mayoría de los casos.

El segundo conjunto de capas (después de salir del lector de desplazamientos) sirven para duplicar entradas. Inicialmente, el ancho de la capa se duplica con cada capa. Si se solicita la duplicación, cada entrada da como resultado dos entradas. Si no se solicita la duplicación, se pasan las entradas con el índice más bajo y las entradas con el índice más alto se rellenan con 0. Una vez que el ancho de la capa coincide con el número de unidades operativas, las duplicaciones posteriores copian la mitad de las entradas y descartan la otra mitad, y en el caso de transferencia es una función de identidad trivial. El patrón de duplicación específico sugerido varía en cada capa y está parametrizado por el número de capa L, que varía de 0 a $\log_2$(NUM_UNITS)-1, de modo que el vector de entradas de salida O[i] se extrae desde el conjunto de admisiones de entradas tal como sigue:

$$P = 2\text{^}L$$

$$O[i] = I[\text{floor}(i / (2 * P)) * P + i \% P]$$

Como ejemplo ilustrativo, se supone que SHIFT_WIDTH = 8 y que NUM_UNITS = 16. Lo que antecede daría como resultado 4 capas. Además, se supone que el valor objeto de lectura del lector de desplazamientos 202 fue el siguiente:

0, 1, 2, 3, 4, 5, 6, 7

Si la duplicación 0ª y 1ª están comprometidas, pero la 2ª y 3ª se dejan como identidad, llegamos a la siguiente entrada en las unidades operativas 116

0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3

Si la 2ª y la 3ª duplicación, están comprometidas, pero la 0ª y la 1ª se dejan como identidad, llegamos a la siguiente entrada en las unidades operativas

0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3

B. Red de multidifusión totalmente reprogramable

Una red multidifusión 204 completamente reprogramable consiste en una red Benes de difusión (también conocida como red sin bloqueo reorganizable) con soporte para nodos de conmutación 2x2 generales que son capaces de difusión. Cada nodo de conmutación contiene dos bits para especificar la operación de permutación o difusión a realizar, que se pone en práctica mediante cerrojos y multiplexores de 1 bit. Lo que antecede permite especificar patrones de multidifusión arbitrarios, lo que permite relacionar cualquier entrada con cualquier conjunto de salidas. Si SHIFT_WIDTH es menor que NUM_UNITS, las primeras capas $\log_2$(NUM_UNITS/SHIFT_WIDTH) de la red se truncarán, y se podrá suponer que cualquier entrada que hubiera venido de la capa anterior es nula.

La reconfiguración de la red de permutación requiere la carga de (2log(NUM_UNITS)-1)NUM_UNITS bits de datos. Estos datos pueden cargarse mediante memoria DRAM o SRAM. Esta carga necesariamente tomará una cantidad significativa de ciclos de reloj y requerirá soporte para direccionar los cerrojos dentro de la red de permutación. Sin embargo, en los casos en que el número de ciclos en los que se utiliza una permutación dada es lo suficientemente grande, la generalidad del soporte de permutación (incluyendo el soporte para anchos de tensor que no son potencias de dos) puede hacer que este método sea conveniente. Concretamente, este método permite poner en práctica el conjunto completo de índices polinómicos arbitrarios en un ciclo para mosaicos de entrada que se ajustan a SHIFT_WIDTH y mosaicos de salida que se ajustan a NUM_UNITS. Además, es posible proporcionar resiliencia a defectos de fabricación o daños que causen unidades operativas defectuosas al permitir que las unidades operativas de repuesto se conecten como sustituciones.

C. Red multidifusión Benes de paridad

La red de multidifusión Benes de paridad es una opción más general que la red de multidifusión de operación binaria, pero menos intensiva para las operaciones de recarga que la red de multidifusión totalmente reprogramable.

Esta red tiene la misma estructura básica de la red Benes de difusión general anterior, sin embargo, en lugar de configurar cada nodo de conmutación de forma independiente, cada nodo de conmutación, para una capa determinada de la red, se configura para que sea la paridad de su índice AND-ed con un valor configurable por capa. Es decir, para la capa 1, elemento i, la elección de qué elemento del nodo de conmutación elegir viene dada por:

$$c[l,i] = \mathrm{PARITY}(\mathrm{AND}(p[l], i))$$

en donde, AND representa el elemento AND del índice i de ancho de bits $\log_2$(NUM_UNITS) con el valor de configuración de ancho de bits $\log_2$(NUM_UNITS) por capa p[1]. PARITY(x) representa la función de paridad sobre el conjunto de bits restantes.

Este mecanismo requiere solamente $\log_2$(NUM_UNITS)(2$\log_2$(NUM_UNITS) - 1) bits para especificar p[l] para todas las capas, o para NUM_UNITS = 64k, 496 bits.

Sin embargo, todas las operaciones de permutaciones de tensor de mosaicos con anchos que son una potencia de dos, así como todas las multidifusiones basadas en potencia de dos son compatibles con este método, lo que lo hace más general que la red de multidifusión de operación binaria, pero menos costosa que la red de multidifusión totalmente reprogramable. Además, mientras que los valores completos de p[l] podrían especificarse por instrucción, también es posible especificar la permutación lógica y la operación de difusión a realizar y calcular los bits requeridos mediante un circuito combinatorio en el circuito integrado, lo que resulta en un ancho de instrucción reducido, o eliminando la necesidad de realizar una búsqueda de memoria SRAM adicional. Por último, las permutaciones podrían almacenarse en una pequeña memoria SRAM independiente, lo que permitiría pequeñas instrucciones que son solamente datos de índices.

Esta idea de configurar el conjunto completo de nodos de conmutación utilizando una función de un conjunto más pequeño de parámetros de configuración y el índice de nodo, puede generalizarse.

D. Red de Permutación Benes y Red de Duplicación Simple

En esta variante, la parte de permutación de la red de multidifusión de operación binaria se sustituye por una red Benes simple (sin difusión) para la que se carga la configuración; sin embargo, se mantiene el conjunto fijo de capas de duplicación. Debido a que la cantidad de bits necesarios para especificar una capa de permutación Benes arbitraria es (2$\log_2$(B)-1)B/2 bits, si SHIFT_WIDTH es <= que ELEMENT_WIDTH, dicha red se puede cargar desde la memoria SRAM en una sola lectura.

Unidades operativas

Las unidades operativas 116 consisten en un gran número (NUM_UNITS) de unidades independientes e idénticas 116, cada una de las cuales normalmente contendría tres registros, A, R y S, y varias unidades funcionales. Además, cada unidad operativa 116 recibe dos elementos por ciclo de reloj, uno de cada red de multidifusión 204. Estos valores se denominan B y C, y tienen una anchura de bits ELEM_WIDTH. Las unidades operativas 116 están todas controladas por un único decodificador de instrucciones, y todas realizan la misma operación en cada ciclo de reloj.

Durante cada ciclo de reloj, cada unidad operativa 116 realiza una operación basada en sus valores de registro anteriores, las nuevas entradas B y C de la memoria, y por elementos de control de instrucción que incluyen D, una constante de 4 bits por instrucción. Cada unidad operativa 116 actualiza entonces sus registros.

Registros de la unidad operativa

La unidad operativa 116 contará con los siguientes registros:

A - Registro acumulador, contiene bits ACCUM_WIDTH interpretados como un número entero con signo

R - Registro de estado RNG, contiene datos utilizados para generar números pseudoaleatorios

S - Registro de estado, contiene 4 booleanos S[0] a S[3]

Operaciones

Durante cada ciclo, la unidad operativa 116 realiza lógicamente operaciones que comprenden instrucciones informáticas limitadas a un conjunto de primitivos de aprendizaje máquina. Los primitivos son operaciones atómicas determinadas para realizar operaciones utilizadas en el aprendizaje máquina que normalmente requieren varias instrucciones en un procesador tradicional. Concretamente, una operación atómica es una operación que se realiza como una unidad discreta de funcionalidad y no se puede subdividir. Por lo tanto, el hardware que admite la funcionalidad que se puede realizar en una única operación que requiere múltiples operaciones en un procesador tradicional acelera el rendimiento de la funcionalidad por la cantidad de operaciones guardadas.

Los ejemplos de funcionalidad a realizar en una operación única en lugar de múltiples operaciones en un procesador tradicional incluyen contracciones de tensor, adición de ruido aleatorio y operaciones no lineales por elementos. Las instrucciones incluyen el conjunto de las siguientes operaciones de primitivos:

SET: A := B // Establecer acumulador

MAC: A += B * C // Multiplicar-Acumular

ARN: A += R // Añadir D bits de ruido aleatorio

LTR: A := (A - B) * C // Transformación lineal

SHR: A;= A » D // Desplazar a la derecha

CMP: S[D] := (B <= A <= C) // Comparar A con el rango, actualizar S[D]

OUT: // Datos de salida de unidades operativas a SRAM, véase operaciones globales

SRD: R = A // Valores RNG iniciales (semilla)

RET: // Retorno, realizado con este cálculo

Además, durante cada ciclo, a menos que la instrucción sea SRD, los registros R se actualizan para obtener un nuevo valor pseudoaleatorio. Lo que antecede se puede realizar mediante un registro de desplazamiento de retroalimentación lineal ("LFSR"), vinculando R registros desde múltiples unidades operativas para aumentar la mezcla.

Este conjunto de instrucciones podría ampliarse según sea necesario. Por ejemplo, para operaciones de agrupación máxima, la siguiente instrucción adicional puede ser de utilidad:

MMA: A := max(A, B * C) // Multiplicar, Acumulación Máxima

Valores de control por instrucción para las unidades operativas

Los valores de control por instrucción para las unidades operativas 116 son los siguientes:

Operación: Qué operación realizar de entre las enumeradas en el apartado anterior.

Condición: Para cada S[i], qué valor debe tener S[i] para realizar esta operación. Puede ser 0, 1 o "no importa" para cada uno. La salida de cada prueba se combina con AND para decidir si se debe realizar la operación. Puesto que cada entrada normalmente se representaría como 2 bits, este componente de la instrucción ocupa 8 bits.

D: Valor utilizado para CMP y SHR.

De nuevo, existen muchas variaciones. Por ejemplo, la ejecución condicional podría definirse como un número de 16 bits, que representa bajo cuál de cada una de las 16 condiciones posibles basadas en S se debe realizar la operación.

Registros de Índice y Unidad de Direcciones

Los registros de índice 206 son globales para las unidades operativas 116 y el decodificador de instrucciones 208, y rastrean diversas variables de índice para controlar el direccionamiento y permitir que se realicen bucles y otras operaciones con facilidad. Cada registro de índice 206 almacena un valor entero, INDEX_WIDTH de bits de anchura, y existen NUM_INDEXES registros.

Cada instrucción contiene cierto número de referencias de dirección, que permiten el cálculo de una dirección SRAM o DRAM mediante los registros de índice.

Cada referencia de dirección contiene una compensación de memoria base, así como un amplio multiplicador INDEX_WIDTH para cada registro de índice, para un total de NUM_INDEXES + 1 valores por referencia de dirección. Para calcular la dirección de memoria, se calcula la suma siguiente:

A = B + Sumar sobre i=[0, NUM_INDEXES) de M[i] * R[i]

en donde B es la dirección base, M[i] es el multiplicador de cada registro de índice y R[i] representa el valor actual de cada registro de índice.

Las instrucciones también contienen una actualización de registro de índice, que nuevamente consta de NUM_INDEXES números de tamaño INDEX_WIDTH, U[i], así como un conjunto de valores booleanos B[i] y realiza la siguiente actualización:

$$R[i] := B[i] \; ? \; (R[i] + U[i]) : U[i]$$

en donde U[i] es la lista de actualizaciones de registros de índice y ? es el operador trinario if-then-else.

Por lo tanto, para cada instrucción, las referencias de memoria se pueden calcular a partir de registros de índice, y los registros de índice se pueden actualizar.

Por último, cada instrucción especifica una prueba de registro de índice, que consiste en una elección del registro de índice, i, para probar, y un valor máximo, M. La prueba se considera verdadera si:

$R[i] < M$

A continuación, cada instrucción especifica un número de instrucción siguiente tanto para el caso en donde la prueba es verdadera como para el caso en donde la prueba es falsa. Lo que antecede permite la creación de construcciones en bucle.

Unidad de transferencia DRAM

La unidad de transferencia de DRAM 118 se solicita en una sección separada de cada instrucción, independientemente de la operación de las unidades operativas. Especifica una memoria SRAM y una ubicación de memoria DRAM (posiblemente con algunos requisitos de alineación), así como una operación (lectura o escritura). La operación también puede ser del tipo *no-op*. Lo que antecede inicia una lectura/escritura hacia/desde DRAM desde/hacia SRAM. Esta operación no se completará al inicio de la siguiente instrucción, debido a la profundidad de la canalización de instrucciones en relación con la latencia de la memoria DRAM, por lo que el software debe tener conocimiento de dicha latencia. El ancho de la transferencia de datos suele ser tan grande como sea posible dado el ancho de banda de la memoria DRAM.

De manera alternativa, la unidad de transferencia de DRAM 118 puede estar restringida para realizar solamente lecturas en SRAM, y las escrituras pueden provenir de las unidades operativas. En este caso, es posible realizar una búsqueda previa desde la DRAM mediante el uso de un segundo decodificador de instrucciones que opera una cierta cantidad de ciclos por delante del decodificador de instrucciones principal y coloca los resultados en una memoria caché asociativa. A continuación, cuando la instrucción se planifica en el decodificador de instrucciones principal, el valor se extrae desde la memoria caché y es objeto de escritura en SRAM, lo que hace que el acceso a DRAM parezca determinista. Para las operaciones de escritura, se puede utilizar una memoria caché de escritura para evitar la necesidad de un desplazamiento completo de ida y vuelta de la memoria DRAM.

Decodificador de instrucciones + SRAM

El decodificador de instrucciones 208 normalmente contendrá su propia memoria SRAM, cuyo ancho está optimizado para la instrucción más grande posible, aunque también puede reutilizar la memoria SRAM del sistema, incluyendo posiblemente un mecanismo de almacenamiento en memoria caché para reducir los conflictos de lectura del banco de memoria. En cada ciclo, el decodificador de instrucciones realiza la lectura de una nueva instrucción y organiza la actividad de las diversas unidades tal como se describe en la sección de operación.

Diseño alternativo de SRAM

De manera alternativa, los bancos de memoria SRAM 114 pueden existir como 2 bancos, cada uno de los cuales tiene una anchura de elementos SHIFT_WIDTH. Cada banco normalmente alimentaría uno de los dos registros de desplazamiento y normalmente tiene muy pocas direcciones (tan solo 4 por banco). Lo que antecede simplifica ese direccionamiento, lo que hace que se cargue exactamente una dirección en el cargador de desplazamiento, que luego simplemente realiza una rotación de los datos desde la dirección de anchura única. Las salidas pueden ser objeto de escritura en cualquier banco, y las secciones de cada banco se pueden planificar para ser objeto de escritura o de lectura desde la memoria DRAM, posiblemente incluyendo una asignación no trivial de múltiples zonas de memoria DRAM a una sola dirección. Por ejemplo, suponiendo que se pueda realizar la escritura de 128 elementos en memoria DRAM para cada ciclo de reloj, un método sería admitir una escritura de cualquier compensación de 128 elementos alineados dentro de una dirección de cualquier banco para programar la lectura o escritura durante cada ciclo de reloj.

Operación del sistema global

En cada ciclo, el decodificador de instrucciones 208 realiza la lectura de una única nueva instrucción. Debido a que las instrucciones están muy canalizadas, se están produciendo operaciones para múltiples instrucciones anteriores mientras el decodificador 208 recupera la nueva instrucción. La Figura 4 es un diagrama de flujo del funcionamiento del decodificador 208. Para aclarar el flujo, en este caso se describirá el orden lógico, pero se podrán programar múltiples etapas independientes en el mismo ciclo de reloj si fuere posible, y algunas etapas, tales como la propagación de los elementos a través de la red de multidifusión, podrán tomar más de un ciclo de reloj, con cerrojos de canalización interna.

En el bloque 402, recuperar y cargar la instrucción del registro del puntero de instrucción actual ("IP").

En el bloque 404, actualizar los registros de índice.

En el bloque 406, realizar la prueba de registro de índice y establecer una nueva IP para la siguiente instrucción.

En el bloque 408, calcular las direcciones para la transferencia de memoria DRAM (si corresponde) y planificar la transferencia mediante la unidad de transferencia de memoria DRAM 118.

En el bloque 410, calcular las direcciones para el acceso a la memoria del elemento tensor B y del elemento tensor C, si es pertinente para el tipo de instrucción extraída.

En el bloque 412, cargar los datos de SRAM para B y C en los bancos de datos de SRAM 114, si es pertinente para el tipo de instrucción obtenida.

En el bloque 414, propagar datos cargados mediante cargadores de desplazamiento, si es pertinente para el tipo de instrucción obtenida.

En el bloque 416, propagar los datos cargados a las unidades operativas 116 mediante las redes de multidifusión 204, si es pertinente para el tipo de instrucción obtenida.

En el bloque 418, realizar la operación en la unidad operativa 116.

Conviene señalar que, si la operación a realizar es "OUT", una parte contigua de los datos de las unidades operativas se escribe en una ubicación de memoria SRAM especificada como una referencia de memoria, siendo la parte una zona alineada con el espacio de índice de la unidad operativa y definida por otra referencia de memoria. Además, se define una longitud de escritura que puede ser menor que SHIFT_WIDTH. De manera alternativa, las salidas se pueden ser objeto de escritura directamente en memoria DRAM.

Ejemplo de un caso de uso de convolución

Las técnicas de aceleración por hardware de aprendizaje máquina anteriores se pueden utilizar para poner en práctica la operación lógica de contracciones de tensor generalizadas, ruido aditivo y no linealidades por elementos. Puesto que estas operaciones por lo general se realizan en el orden anterior repetidamente para redes profundas (una vez por capa), se asume este mismo orden. Sin embargo, lo que antecede no es requerido.

Para concretar las operaciones, se examinará una operación específica, una convolución seguida de una ronda aleatoria, seguida asimismo de una no linealidad RELU con fugas, de una imagen que tiene 256 de anchura por 256 de altura por 64 canales de características de profundidad con un kernel que opera sobre una zona x,y de 3x3 y asigna 64 canales de entrada a 64 canales de salida. Se supondrá que el tensor de entrada que representa la imagen y el kernel ya existe dentro de la SRAM organizada de modo que se tenga:

$$D[ci, x, y] = M[D\_o + ci*65536 + 256*x + y]$$

$$K[i, j, ci, co] = K[K\_o + i*16384 + j*4096 + 64*ci + co]$$

También se supondrá que existe una zona para la salida en:

$$O[co, x, y] = M[O\_o + co*65536 + 256*x + y]$$

En donde M[i] representan posiciones de memoria lógica mediante el diseño de memoria no alternativo. También se supondrán valores predeterminados para todos los parámetros. Se presumirá el bit de funcionamiento de la red multidifusión.

Lógicamente, se realizará la iteración sobre cada valor de x de 0 a 256 en etapas de 6 elementos, y para cada uno, se calculará la salida para todos los valores de *y* y *co* para los valores *x* en cuestión como un mosaico único, realizando una acumulación interna sobre *i*, *j* y *ci*. A continuación, se añadirá ruido, se reducirá la precisión y se realizará la no linealidad, y luego serán objeto de lectura los resultados.

Para realizar lo que antecede, se asignarán los diversos índices de tensores lógicos a registros de índice de la siguiente manera:

R0 = x r1 = ci r2 = i r3 = j r4 = co (para salida a memoria principal)

Se definen dos patrones de multidifusión específicos, uno para el kernel y el otro para la entrada. Para el kernel, se define un patrón que asigna elementos de entrada [0,64) a todas las unidades operativas, de modo que U[i] = I[i/2048]. Lo que antecede se puede hacer habilitando todas las capas de duplicación < que 12, deshabilitando todas las capas de duplicación >= que la capa 12 y deshabilitando todas las capas de permutación. Se denominará a este patrón PK.

Para la entrada, se define un patrón que asigna repeticiones 64 veces para todos los 2048 elementos de entrada. Lo que antecede se puede hacer habilitando todas las capas de duplicación < a 6, deshabilitando todas las capas de duplicación >= que la capa 6 y deshabilitando todas las capas de permutación, denominando a este patrón PI.

También se creará un conjunto de constantes C y se cargarán en la memoria de modo que se pueda utilizar el patrón de capa de permutación que permita que todas las capas de duplicación y las capas sin permutaciones proporcionen la constante a todas las unidades operativas.

A continuación, se indica el flujo de instrucciones lógicas que realiza la operación anterior:

00: SET // Poner a cero todos los acumuladores

B=constant 0x0000

r0 = 0, r1 = 0, r2 = 0, r3 = 0

01: MAC // Sumar cada fila de i

B=M[D_o + 65536*r1 + 256*r0 + 256*r2 + r3] vía PI C=M[K_o +

16384*r2 + 4096*r3 + 64*r1] vía PK

r3 += 1 if (r3 < 2) goto 1, else goto 2

02: MAC // Sumar cada columna de j

B=M[D_o + 65536*r1 + 256*r0 + 256*r2 + r3] vía PI C=M[K_o +

16384*r2 + 4096*r3 + 64*r1] vía PK

r2 += 1, r3 = 0

if (r2 < 2) goto 1, else goto 3

03: MAC // Sobre cada canal de entrada

B=M[D_o + 65536*r1 + 256*r0 + 256*r2 + r3] vía PI C=M[K_o +

l6384*r2 + 4096*r3 + 64*r1] via PK

r1 += 1, r2 = 0, r3 = 0 if (r2 < 63) goto 1, else goto 4

04: ARN 15 bits // Agregar ruido para poner en práctica una ronda aleatoria

05: SHR 16 // Desplazar hacia abajo en 16 para retornar a 8.8

06: CMP -> S[0] // Ver si son mayor o igual que 0.0

B=constant 0x0000

C=constant 0x7fff

07: LTR if !S[0] //Si es menor, multiplicar por .1 * 0x1000

B=constant 0x0000

C=constant 0x199

08: LTR si S[0] // Si es mayor o igual multiplicar por 0x1000

B=constant 0x0000

C=constant 0x1000

09: ARN 11 bits // Añadir ruido para poner en práctica una ronda aleatoria

10: SHR 12 // Desplazar hacia abajo para eliminar el factor de Ox1000

r4 = 0 if (r0 < 252) goto 11, else goto 13

11: OUT // Caso de salida normal

offset = 2048*r4

M[D_o + 65536*r4 + 256*r0]

size = 2048

r4+= 1

if (r4 < 64) goto 11, else goto 12

12; SET // Volver a poner a cero los acumuladores y obtener las siguientes 6 filas

B=constant 0x0000

```
        r0 += 6 goto 0
    13: OUT // Caso de salida final especial para gestionar las últimas 2 filas
        offset = 2048*r4
        M[D_o + 65536*r4 + 256*r0]
        size = 512
        r4+= 1
        if (r4 < 64) goto 11, else goto 14
    14: RET
```

La descripción del tema se puede resumir de la siguiente manera:

Un sistema para realizar la aceleración por hardware para el aprendizaje máquina comprende: múltiples bancos de memoria de acceso aleatorio; un lector de desplazamientos; una pluralidad de unidades operativas configuradas para acceder al menos a una parte de la memoria de acceso aleatorio al mismo tiempo; una red multidifusión acoplada de manera comunicativa a la memoria y al lector de desplazamientos; y un decodificador de instrucciones compuesto por un acumulador, uno o más registros y un generador de números aleatorios, en donde el decodificador de instrucciones está configurado para procesar instrucciones restringidas a un conjunto de instrucciones compuesto por primitivos de aprendizaje máquina, configurado para realizar transferencias de memoria y enviar instrucciones a las unidades operativas.

La memoria de acceso aleatorio puede estar compuesta por una memoria de acceso aleatorio estática y configurada para acceder a la misma con un ancho de memoria suficiente para realizar operaciones vectoriales con un acceso único.

Cada banco de memoria de acceso aleatorio puede estar acoplado con suficientes lectores de desplazamientos para realizar desplazamientos de modo que los lectores de desplazamientos no necesiten cambiar la dirección de la memoria rotando todo el banco de memoria de acceso aleatorio.

El sistema puede comprender, además, una pluralidad de registros de índice, en donde las instrucciones del conjunto de instrucciones están configuradas para calcular direcciones de memoria a partir de los registros de índice y actualizar los registros de índice dentro de la operación atómica de la instrucción.

El lector de desplazamiento puede configurarse para realizar una operación de desplazamiento entre los bancos de memoria de acceso aleatorio mediante: lectura simultánea de datos de una pluralidad de bancos de memoria en función de una compensación lógica; concatenar los datos objeto de lectura en una única lista lógica; rotar los datos objeto de lectura en la lista lógica única basándose en la compensación lógica y el ancho de la memoria que puede ser objeto de lectura en una única operación; y cambiar los datos de lectura en base a la compensación lógica y al ancho de la memoria que puede ser objeto de lectura en una única operación.

La red de multidifusión puede configurarse para permutar datos y copiar datos desde el lector de desplazamientos y la memoria mediante la operación de las unidades operativas. De manera opcional, la red de multidifusión puede estar compuesta por una pluralidad de capas de permutación y una pluralidad de capas de duplicación, estando configuradas las capas de permutación para realizar operaciones de transposición atómica de mosaicos de tensor. De otro modo, la red de multidifusión también puede ser: una red sin bloqueo reorganizable de difusión con unidades de conmutación 2x2, conservando cada unidad el estado en cuanto a una función de permutación o de difusión a realizar, de modo que la red pueda configurarse de manera dinámica para un algoritmo de multidifusión arbitrario; una red sin bloqueo reorganizable por paridad tal que un conjunto completo de nodos de conmutación se configure en base al menos a un conjunto de parámetros menor que el número de nodos de conmutación y un índice de nodo; o una red sin bloqueo reorganizable que no sea de difusión tal que la duplicación se realice mediante una duplicación simple.

El sistema puede comprender, además, una unidad de transferencia configurada para realizar amplias operaciones de copia masiva entre la memoria no local y la memoria de acceso aleatorio local.

Las unidades operativas pueden configurarse para tener un acumulador con un ancho de memoria suficiente para realizar operaciones vectoriales con un acceso único.

Las unidades operativas pueden configurarse con un generador de números pseudoaleatorios y un registro de números pseudoaleatorios.

Las unidades operativas pueden configurarse para procesar una operación atómica de multiplicación y de acumulación máxima (MMA).

Las unidades operativas pueden configurarse para realizar operaciones de coma flotante con una precisión de mantisa de menos de 24 bits.

Conclusión

Aunque el objeto se ha descrito en un lenguaje específico de características estructurales y/o actos metodológicos, debe entenderse que el objeto definido en las reivindicaciones adjuntas no se limita necesariamente a las características o actos específicos descritos con anterioridad. Más bien, las características y acciones específicas descritas con anterioridad se describen como ejemplos de formas de poner en práctica las reivindicaciones.

REIVINDICACIONES

1. Un aparato que comprende:

una unidad central de procesamiento, CPU;

un bus para acoplar la CPU a una memoria de acceso aleatorio dinámica, DRAM; y

un acelerador por hardware de aprendizaje máquina acoplado a la CPU, comprendiendo el acelerador de aprendizaje máquina:

una pluralidad de bancos de memoria de acceso aleatorio estática, SRAM, para almacenar un primer tensor de entrada que comprende una primera matriz multidimensional de números reales y un segundo tensor de entrada que comprende una segunda matriz multidimensional de números reales;

el circuito de transferencia SRAM/DRAM para realizar transferencias multibyte de datos tensoriales entre los bancos de DRAM y de SRAM, incluyendo los datos tensoriales al menos una parte de la primera y segunda matrices multidimensionales de números reales;

un decodificador de instrucciones para decodificar una instrucción vectorial de multiplicación-acumulación, MAC, que incluye un valor de operación que indica una operación MAC, una indicación de una primera pluralidad de los números reales de la primera matriz multidimensional y una segunda pluralidad de los números reales de la segunda matriz multidimensional e información de permutación;

circuitos para permutar, sobre la base de la información de permutación, la primera pluralidad de los números reales de la primera matriz multidimensional de conformidad con la información de permutación para generar una primera pluralidad permutada de números reales;

una pluralidad de unidades operativas para realizar una pluralidad de operaciones paralelas de multiplicación y acumulación, MAC, de conformidad con la instrucción, comprendiendo cada unidad operativa:

un multiplicador para multiplicar un primer número real de la primera pluralidad permutada de números reales y un segundo número real correspondiente de una segunda pluralidad de números reales asociados con la segunda matriz multidimensional para generar un producto, y

un acumulador para añadir el producto a un valor de acumulación para generar un valor de resultado, teniendo cada uno el primer número real y el segundo número real un primer ancho de bits y el valor de acumulación teniendo un segundo ancho de bits al menos dos veces el primer ancho de bits.

2. El aparato según la reivindicación 1, en donde el circuito de transferencia SRAM/DRAM debe realizar transferencias multibyte de datos tensoriales entre los bancos DRAM y SRAM en respuesta a la instrucción.

3. El aparato según la reivindicación 1, en donde el circuito de transferencia SRAM/DRAM debe realizar las transferencias multibyte de datos tensoriales entre los bancos DRAM y SRAM independientemente de las unidades operativas que realizan las operaciones MAC.

4. El aparato según cualquiera de las reivindicaciones 1 a 3 que, además, comprende:

una pluralidad de registros de acumulación, cada registro de acumulación para almacenar uno de los valores de acumulación que se agregarán a uno de los productos mediante el acumulador de una de las unidades operativas.

5. El aparato según cualquiera de las reivindicaciones 1 a 4, en donde el primer tensor de entrada comprende una primera matriz bidimensional que comprende la primera pluralidad de números reales y el segundo tensor de entrada comprende una segunda matriz bidimensional que comprende la segunda pluralidad de números reales.

6. El aparato según la reivindicación 5, en donde la primera matriz bidimensional comprende datos del núcleo y la segunda matriz bidimensional comprende datos de entrada.

7. El aparato según la reivindicación 5 o 6, en donde la pluralidad de operaciones paralelas de multiplicación y acumulación, MAC, comprenden:

multiplicar cada número real de la primera pluralidad permutada de números reales en una fila de la primera matriz bidimensional con un número real correspondiente de la segunda pluralidad de números reales en una columna de la segunda matriz bidimensional para generar una pluralidad de productos; y

añadir la pluralidad de productos para generar un resultado en una matriz de resultados bidimensional, comprendiendo el resultado un único número real en la matriz de resultados bidimensional.

8. El aparato según cualquiera de las reivindicaciones 1 a 7 que, además, comprende:

un procesador acoplado a las unidades operativas; y

una interconexión de entrada/salida, IO, para acoplar el procesador a un bus IO.

9. El aparato según cualquiera de las reivindicaciones 1 a 8, en donde el circuito para permutar comprende una pluralidad de conmutadores configurables.

10. El aparato según cualquiera de las reivindicaciones 1 a 9, en donde el circuito para permutar es para realizar una operación de difusión para enrutar un único elemento de datos origen a cada una de entre la pluralidad de unidades operativas.

11. El aparato según cualquiera de las reivindicaciones 1 a 10, en donde se van a generar valores de índice para realizar la operación de permutación.

12. El aparato según cualquiera de las reivindicaciones 1 a 11, en donde el primer ancho de bits es de 16 bits.

13. El aparato según la reivindicación 12, en donde la primera y segunda pluralidad de números reales comprenden elementos de datos de punto fijo de 16 bits.

14. El aparato según cualquiera de las reivindicaciones 1 a 13 que, comprende, además:

una memoria caché compartida por al menos una o más de entre la pluralidad de unidades operativas.

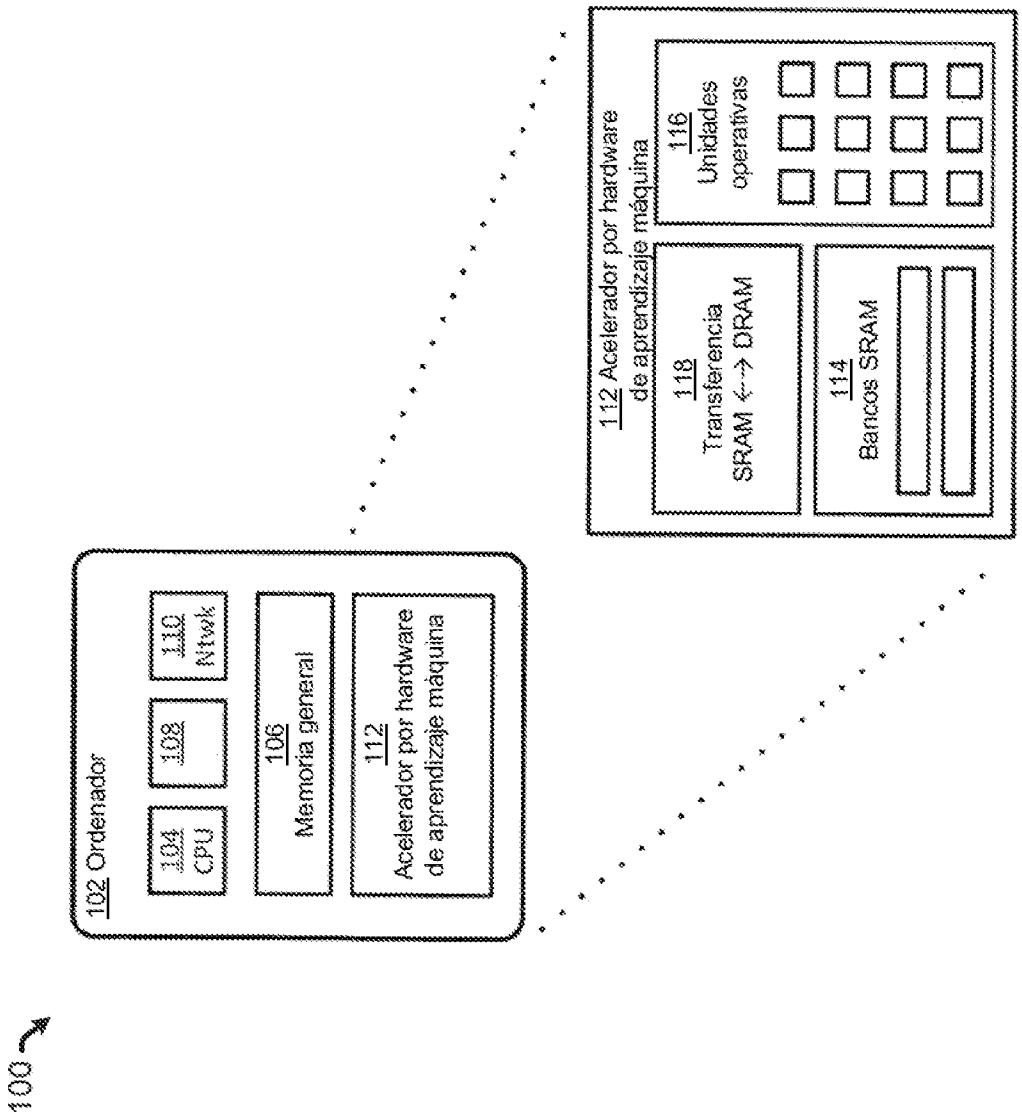
100
102 Ordenador
104 CPU
108
110 Ntwk
106 Memoria general
112 Acelerador por hardware de aprendizaje máquina
112 Acelerador por hardware de aprendizaje máquina
118 Transferencia SRAM ←→ DRAM
114 Bancos SRAM
116 Unidades operativas

Figura 1

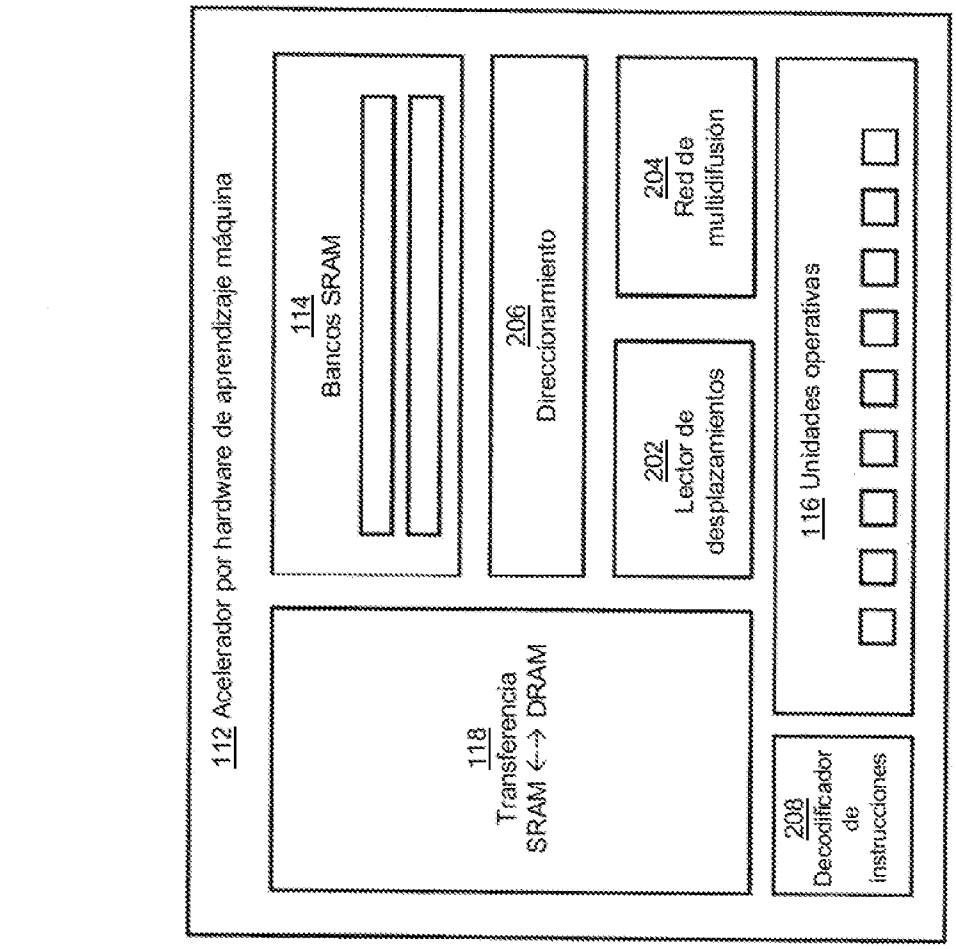
200
112 Acelerador por hardware de aprendizaje máquina
114
Bancos SRAM
206
Direccionamiento
204
Red de
multidifusión
202
Lector de
desplazamientos
118
Transferencia
SRAM ←→ DRAM
116 Unidades operativas
208
Decodificador
de
instrucciones

Figura 2

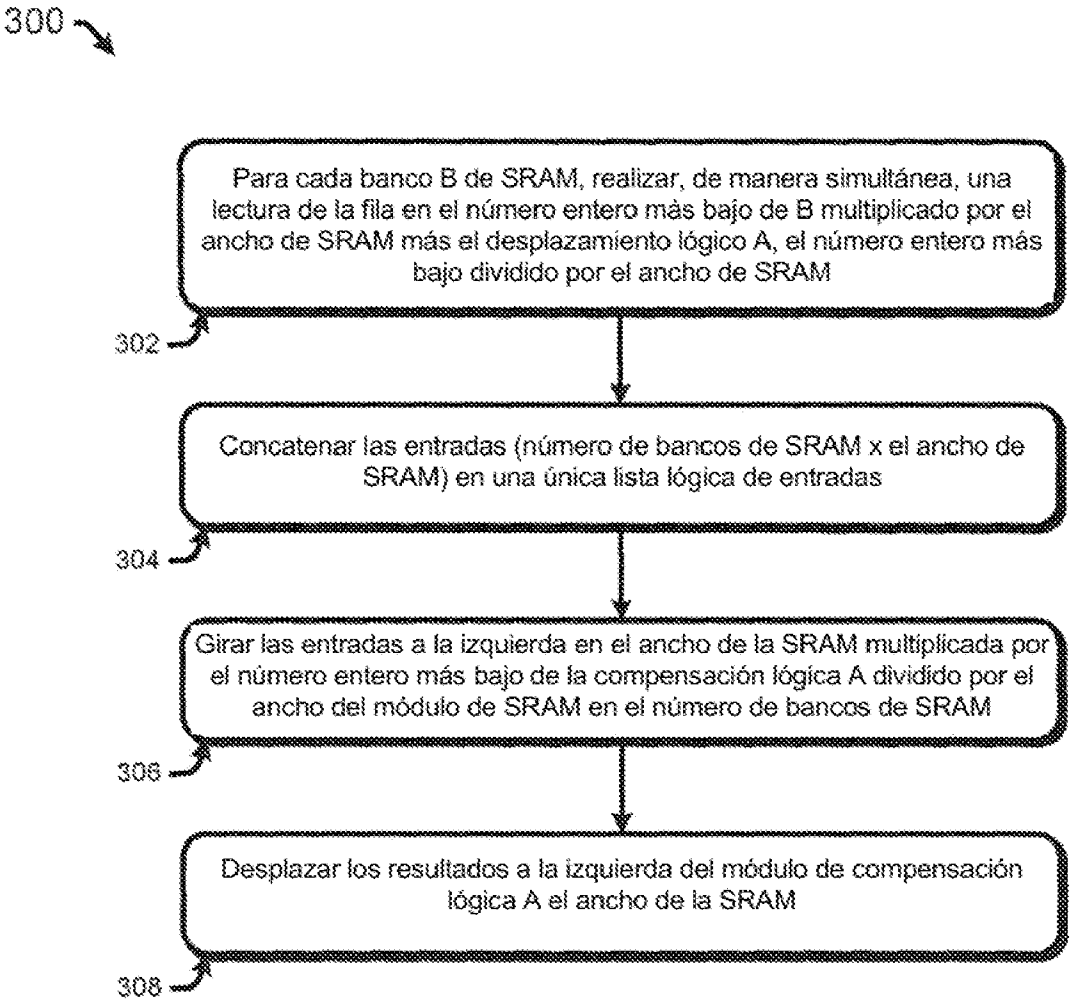
300
Para cada banco B de SRAM, realizar, de manera simultánea, una lectura de la fila en el número entero más bajo de B multiplicado por el ancho de SRAM más el desplazamiento lógico A, el número entero más bajo dividido por el ancho de SRAM
302
Concatenar las entradas (número de bancos de SRAM x el ancho de SRAM) en una única lista lógica de entradas
304
Girar las entradas a la izquierda en el ancho de la SRAM multiplicada por el número entero más bajo de la compensación lógica A dividido por el ancho del módulo de SRAM en el número de bancos de SRAM
306
Desplazar los resultados a la izquierda del módulo de compensación lógica A el ancho de la SRAM
308

Figura 3

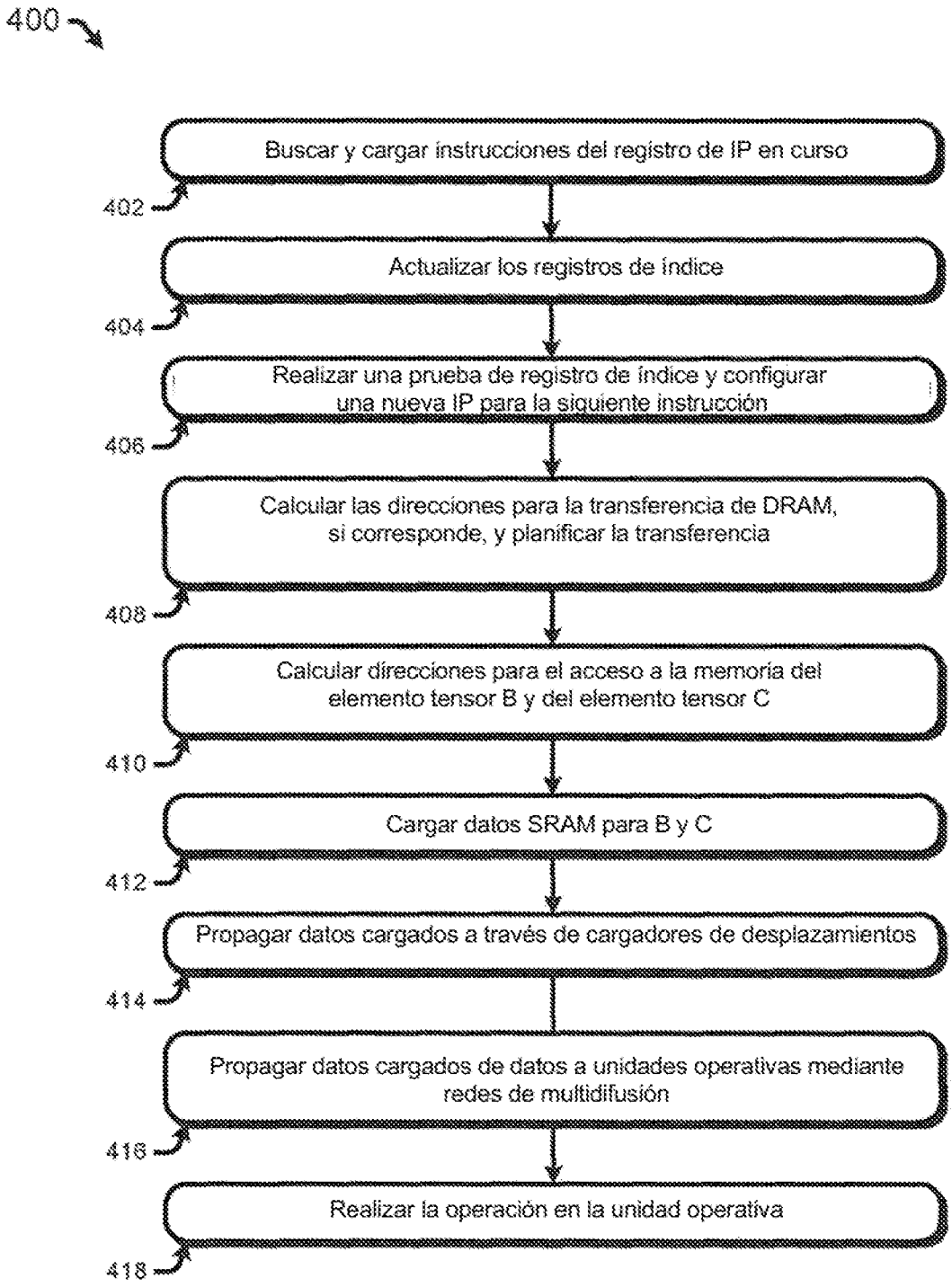
400
Buscar y cargar instrucciones del registro de IP en curso
402
Actualizar los registros de índice
404
Realizar una prueba de registro de índice y configurar una nueva IP para la siguiente instrucción
406
Calcular las direcciones para la transferencia de DRAM, si corresponde, y planificar la transferencia
408
Calcular direcciones para el acceso a la memoria del elemento tensor B y del elemento tensor C
410
Cargar datos SRAM para B y C
412
Propagar datos cargados a través de cargadores de desplazamientos
414
Propagar datos cargados de datos a unidades operativas mediante redes de multidifusión
416
Realizar la operación en la unidad operativa
418

Figura 4