

PROCESADOR DE RED NEURONAL QUE UTILIZA COMPRESIÓN Y DESCOMPRESIÓN DE DATOS DE ACTIVACIÓN PARA REDUCIR LA UTILIZACIÓN DE ANCHO DE BANDA DE MEMORIA

Antecedentes de la invención

Las redes neuronales profundas (DNN, por sus siglas en inglés) se modelan de manera flexible a partir del procesamiento de información y de los patrones de comunicación en los sistemas nerviosos biológicos, como el cerebro humano. Las DNN pueden utilizarse para resolver problemas de clasificación complejos como, entre otros, detección de objetos, etiquetado semántico y extracción de características. Como resultado, las DNN forman la base de muchas aplicaciones de inteligencia artificial (AI, por sus siglas en inglés), como visión artificial, reconocimiento de voz y traducción automática. Las DNN pueden igualar o superar la precisión humana en muchos de estos dominios.

El alto nivel de rendimiento de las DNN se deriva de su capacidad para extraer características de alto nivel a partir de los datos de entrada después de utilizar el aprendizaje estadístico sobre un conjunto de datos de gran tamaño para obtener una representación efectiva de un espacio de entrada. Sin embargo, el rendimiento superior de las DNN tiene el costo de una alta complejidad computacional. Los procesadores de uso general de alto rendimiento, como las unidades de procesamiento de gráficos (GPU, por sus siglas en inglés), se utilizan comúnmente para proporcionar el alto nivel de rendimiento computacional requerido por muchas aplicaciones de DNN.

Aunque los procesadores de uso general, como las GPU, pueden proporcionar un alto nivel de rendimiento computacional para implementar las DNN, estos tipos de procesadores generalmente no son adecuados para su uso en la ejecución de operaciones de DNN durante largos períodos en dispositivos de cómputo donde el bajo consumo de energía es crítico. Por ejemplo, los procesadores de uso general, como las GPU, pueden no ser adecuados para su uso en la ejecución de tareas de DNN de larga duración en dispositivos portátiles que funcionan con baterías, como teléfonos inteligentes o dispositivos de realidad alternativa/virtual (AR/VR, por sus siglas en inglés), donde se requiere consumo de energía para extender la vida útil de la batería.

El consumo de energía reducido mientras se realizan tareas continuas de DNN, como la detección de movimiento humano, también puede ser importante en dispositivos que no funcionan con baterías, como, por ejemplo, una cámara de seguridad de alimentación por Ethernet (POE, por sus siglas en inglés). En este ejemplo específico, los interruptores POE pueden proporcionar solo una cantidad limitada de energía y la reducción del consumo de energía de los dispositivos POE como las cámaras de seguridad permite el uso de interruptores POE que proporcionan menos energía.

Se han desarrollado circuitos integrados de aplicación específica (ASIC, por sus siglas en inglés) que pueden proporcionar un procesamiento de DNN eficaz y al mismo tiempo reducir el consumo de energía en comparación con los procesadores de uso general. Sin embargo, a pesar de los avances en esta área, existe una necesidad continua de mejorar el rendimiento y reducir el consumo de energía de los ASIC que realizan el procesamiento de DNN, particularmente para su uso en dispositivos de cómputo donde el bajo consumo de energía es crítico.

Es con respecto a estos y otros desafíos técnicos que se presenta la divulgación realizada en el presente documento.

Breve descripción de la invención

Se divulga un módulo o procesador de DNN que puede comprimir y descomprimir datos de activación para reducir la utilización del ancho de banda del bus de memoria. En particular, el módulo de DNN puede utilizar la compresión para reducir la utilización del ancho de banda del bus entre la salida de neuronas y la memoria interna o externa. El módulo de DNN también puede utilizar la descompresión para reducir la utilización del ancho de banda del bus de memoria entre la memoria interna o externa y la entrada de neuronas. La utilización reducida del ancho de banda puede permitir un procesamiento más rápido y, en consecuencia, también puede reducir el consumo de energía. Otros beneficios técnicos no mencionados específicamente en el presente documento también pueden obtenerse a través de implementaciones de la materia objeto divulgada.

Con el fin de obtener los beneficios técnicos mencionados brevemente antes, se divulga un procesador DNN que incluye una o más neuronas y una unidad de compresión. La unidad de compresión puede recibir un fragmento de datos descomprimido generada por una o más de las neuronas. El fragmento de datos descomprimido incluye un número fijo de bytes, como 64 bytes, en algunas modalidades.

Para comprimir el fragmento de datos descomprimido, la unidad de compresión puede generar una porción de máscara y una porción de datos de un fragmento de salida comprimido. La porción de máscara del fragmento de salida comprimido incluye una cantidad de bits equivalente a la cantidad fija de bytes en el fragmento de datos descomprimido. Por ejemplo, si el fragmento de datos descomprimido incluye 64 bytes de datos, la porción de máscara incluirá 64 bits (es decir, 8 bytes).

En algunas modalidades, cada bit en la porción de máscara del fragmento de salida comprimido corresponde a un byte en el fragmento de datos descomprimido. Por ejemplo, el bit uno de la porción de máscara puede corresponder al primer byte en el fragmento de datos descomprimido, el bit dos de la porción de máscara puede corresponder al segundo byte en el fragmento de datos descomprimido, etcétera. En otras modalidades, dos o más bits en la porción de máscara del fragmento de salida comprimido corresponden a un byte en el fragmento de datos descomprimido. En estas modalidades, los bits en la porción de máscara del fragmento de salida comprimido pueden indicar que no solo existen un byte correspondiente en el fragmento descomprimido sino también su magnitud aproximada.

Cuando los bits individuales de la porción de máscara corresponden a bytes en el fragmento descomprimido, la unidad de compresión establece cada bit en la porción de máscara del fragmento de salida comprimido en un falso lógico (que también podría denominarse en el presente documento como un “cero lógico”), donde un byte correspondiente en el fragmento de datos descomprimido contiene todos los ceros (es decir, un “byte cero”). La unidad de compresión también establece cada bit en la porción de máscara del fragmento de salida comprimido en un verdadero lógico (que también podría denominarse en el presente documento como un “uno lógico”), donde un byte correspondiente en el fragmento de datos descomprimido contiene al menos un bit distinto de cero (es decir, un “byte distinto de cero”). De esta

manera, la porción de máscara del fragmento de salida comprimido codifica la presencia y ubicación de los bytes cero y no cero en el fragmento de datos descomprimido.

La unidad de compresión genera la porción de datos del fragmento de salida comprimido al determinar la cantidad de bytes distintos de cero en el fragmento de datos descomprimido. La unidad de compresión entonces determina, con base en la cantidad de bytes distintos de cero en el fragmento de datos descomprimido y la cantidad de bytes disponibles en la porción de datos del fragmento de salida comprimido, la cantidad de bits en la porción de datos del fragmento de salida comprimido que están disponibles para almacenar cada byte distinto de cero del fragmento de datos descomprimido. Por ejemplo, si la porción de datos del fragmento de datos comprimido tiene 24 bytes de ancho (es decir, 192 bits) y existen 47 bytes distintos de cero en el fragmento de datos descomprimido, existen cuatro bits disponibles en la porción de datos para almacenar cada byte cero a partir del fragmento de datos descomprimido.

En algunas modalidades, la unidad de compresión también puede determinar la cantidad de bits adicionales, si están presentes, en la porción de datos del fragmento de salida comprimido que están disponibles para almacenar bytes distintos de cero del fragmento de datos descomprimido. En el ejemplo anterior, por ejemplo, cuatro bits adicionales están disponibles para almacenar bytes distintos de cero (es decir, $192 \bmod 47 = \text{cuatro bits}$). La unidad de compresión puede asignar estos bits adicionales a uno o más de los bytes distintos de cero en el fragmento de datos descomprimido antes de truncar uno o más de los bytes distintos de cero. Por ejemplo, la unidad de compresión podría asignar estos bits adicionales a los primeros bytes en la porción de datos del fragmento de salida comprimido.

La unidad de compresión entonces trunca los bytes distintos de cero en el fragmento de datos descomprimido a la cantidad determinada de bits disponibles en la porción de datos para almacenar cada byte distinto de cero (es decir, cuatro en el ejemplo dado anteriormente). La unidad de compresión trunca los bits menos significativos (LSB, por sus siglas en inglés) de los bytes distintos de cero para ajustarse dentro de la cantidad de bits disponibles en la porción de datos en una modalidad. En otra modalidad, la unidad de compresión trunca los bits más significativos (MSB, por sus siglas en inglés) de los bytes distintos de cero. La unidad de compresión entonces almacena los bytes distintos de cero truncados en la porción de datos del fragmento de salida comprimido. El fragmento de salida comprimido, que incluye la porción de máscara y la porción de datos, puede entonces enviarse, por ejemplo, a la memoria interna en el procesador de DNN o memoria externa de un servidor de aplicación del procesador DNN.

El módulo de DNN también puede incluir una unidad de descompresión que puede descomprimir los fragmentos de datos que se comprimieron de la manera descrita anteriormente. Por ejemplo, la unidad de descompresión puede recibir un fragmento de datos comprimido a partir de la memoria en el procesador de DNN o la memoria de un servidor de aplicación. La unidad de descompresión puede entonces determinar la cantidad de bytes distintos de cero en la porción de datos del fragmento de datos descomprimido con base en la cantidad de bits verdaderos lógicos en la porción de máscara del fragmento de salida comprimido. La unidad de descompresión también puede determinar las ubicaciones de los bytes distintos de cero en el fragmento de datos descomprimido con base en las ubicaciones de los bits verdaderos lógicos en la porción de máscara del fragmento de salida comprimido. La unidad de

descompresión puede determinar la ubicación de los bytes cero en el fragmento de datos descomprimido de manera similar.

La unidad de descompresión también puede determinar la cantidad de bits utilizados por la unidad de compresión para almacenar los bytes distintos de cero truncados en la porción de datos del fragmento de salida comprimido. La unidad de descompresión puede determinar la cantidad de bits utilizados para almacenar cada byte truncado distinto de cero con base en la cantidad de bytes distintos de cero en el fragmento de datos comprimido y la cantidad de bytes disponibles en la porción de datos del fragmento de salida descomprimido.

En el ejemplo anterior, por ejemplo, si la porción de datos del fragmento de datos comprimido tiene 24 bytes de ancho (es decir, 192 bits) y existen 47 bytes distintos de cero en el fragmento de datos descomprimido, la unidad de compresión utilizará cuatro bits para almacenar cada byte truncado distinto de cero del fragmento de datos descomprimido en la porción de datos. La unidad de descompresión también puede determinar la cantidad de bits adicionales, si están presentes, que la unidad de compresión asignó a uno o más de los bytes distintos de cero truncados almacenados en la porción de datos del fragmento de salida comprimido.

Para cada posición de bit en la porción de máscara del fragmento de salida comprimido que es un cero lógico, la unidad de descompresión inserta un byte cero en la posición correspondiente del fragmento de salida descomprimido. Para cada posición en la porción de máscara que es un uno lógico, la unidad de descompresión inserta el byte truncado distinto de cero desde la posición correspondiente del fragmento de entrada comprimido en una posición correspondiente del fragmento de salida descomprimido junto con un número de bits cero equivalentes a la cantidad de bits truncados durante la compresión del fragmento de salida comprimido. Los bits cero pueden insertarse en los LSB o MSB de los bytes distintos de cero truncados, dependiendo de qué bits se truncaron durante la compresión.

En algunas modalidades, la unidad de descompresión también agrega un desplazamiento (por ejemplo, 00000001) a uno o más de los bytes distintos de cero truncados almacenados en el fragmento de salida descomprimido. Por ejemplo, puede agregarse un desplazamiento a bytes distintos de cero del fragmento de datos descomprimido que se convierten en bytes cero después de la compresión. De esta manera, los bytes distintos de cero no se convertirán en bytes cero cuando se comprimen y descomprimen. En otras modalidades, puede agregarse un desplazamiento a todos los bytes en el fragmento de salida descomprimido.

Tal como se abordó brevemente antes, las implementaciones de las tecnologías descritas en el presente documento pueden reducir la utilización del ancho de banda del bus de memoria en un módulo de DNN, permitir que un módulo de DNN complete las operaciones de procesamiento más rápidamente y reducir el consumo de energía. Otros beneficios técnicos no identificados específicamente en el presente documento también pueden obtenerse a través de implementaciones de las tecnologías divulgadas.

Debería apreciarse que la materia objeto descrita anteriormente puede implementarse como un aparato controlado por computadora, un método implementado por computadora, un dispositivo de cómputo o como un artículo de fabricación tal como un medio legible por computadora. Estas y varias otras

características serán evidentes a partir de una lectura de la siguiente descripción detallada de la invención y una revisión de los dibujos asociados.

Esta breve descripción se proporciona para presentar una breve descripción de algunos aspectos de las tecnologías divulgadas en una forma simplificada que se describe más adelante en la descripción detallada de la invención. Esta breve descripción no pretende identificar características importantes o características esenciales de la materia objeto reclamada, ni pretende que esta breve descripción se utilice para limitar el alcance de la materia objeto reclamada. Además, la materia objeto reclamada no se limita a implementaciones que resuelvan cualquiera o todas las desventajas observadas en cualquier parte de esta divulgación.

Breve descripción de los dibujos

La figura 1 es un diagrama de arquitectura de cómputo que muestra aspectos de la configuración y operación de un módulo de DNN que implementa aspectos de las tecnologías divulgadas en el presente documento, de acuerdo con una modalidad.

Las figuras 2A y 2B son diagramas de arquitectura del sistema de cómputo que muestran aspectos de la configuración y operación de un módulo de DNN para comprimir datos de activación, de acuerdo con una modalidad.

La figura 3 es un diagrama de estructura de datos que ilustra aspectos de la operación de un módulo de DNN para comprimir datos de activación con referencia a un fragmento de datos de activación descomprimido ejemplar, de acuerdo con una modalidad.

La figura 4 es un diagrama de flujo que muestra una rutina que ilustra aspectos de la operación del módulo de DNN divulgado para comprimir datos de activación, de acuerdo con una modalidad divulgada en el presente documento.

Las figuras 5A y 5B son diagramas de arquitectura del sistema de cómputo que muestran aspectos de la configuración y operación de un módulo de DNN para descomprimir datos de activación, de acuerdo con una modalidad.

La figura 6 es un diagrama de estructura de datos que ilustra aspectos de la operación de un módulo de DNN para descomprimir datos de activación con referencia a un fragmento de datos de activación comprimido ejemplar, de acuerdo con una modalidad.

La figura 7 es un diagrama de flujo que muestra una rutina que ilustra aspectos de la operación del módulo de DNN divulgado para descomprimir datos de activación, de acuerdo con una modalidad divulgada en el presente documento.

La figura 8 es un diagrama de arquitectura de computadora que muestra una arquitectura ilustrativa de *hardware* y *software* de cómputo para un dispositivo de cómputo que puede actuar como un servidor de aplicación para el módulo de DNN presentado en el presente documento, de acuerdo con una modalidad.

La figura 9 es un diagrama de red que ilustra un entorno de cómputo distribuido en el cual pueden implementarse aspectos de las tecnologías divulgadas, de acuerdo con diversas modalidades presentadas en el presente documento.

Descripción detallada de la invención

La siguiente descripción detallada de la invención está dirigida a un módulo de DNN que puede comprimir y descomprimir los datos de activación para reducir la utilización del ancho de banda del bus de memoria. Tal como se abordó brevemente antes, las implementaciones de las tecnologías divulgadas pueden reducir la utilización del ancho de banda del bus de memoria en un módulo de DNN, permitir que un módulo de DNN complete las operaciones de procesamiento más rápidamente y reducir el consumo de energía. Otros beneficios técnicos no mencionados específicamente en el presente documento también pueden obtenerse a través de implementaciones de la materia objeto divulgada.

Aunque la materia objeto descrita en el presente documento se presenta en el contexto general de un módulo de DNN de *hardware*, los expertos en la materia reconocerán que pueden realizarse otras implementaciones en combinación con otros tipos de sistemas y módulos de cómputo. Los expertos en la materia también apreciarán que la materia descrita en el presente documento puede ponerse en práctica con otras configuraciones de sistemas de cómputo, que incluyen dispositivos portátiles, sistemas multiprocesador, sistemas electrónicos de cómputo o de procesamiento con base en microprocesadores o sistemas de procesamiento integrados en dispositivos (como dispositivos de cómputo vestibles, automóviles, domótica, etcétera), minicomputadoras, unidades centrales y similares.

Tal como se describirá con mayor detalle posteriormente, se describe un módulo de DNN que se configura para comprimir la salida de sus neuronas. La salida comprimida puede almacenarse en la memoria del módulo de DNN o en una memoria externa al módulo de DNN, como la memoria proporcionada por un servidor de aplicación para el módulo de DNN. El módulo de DNN entonces puede descomprimir los datos previamente comprimidos y proporcionar los datos descomprimidos a las neuronas.

De acuerdo con una modalidad, una unidad de compresión en el procesador de DNN comprime fragmentos de longitud fija (por ejemplo, 64 bytes) de datos de activación descomprimidos a una relación de compresión fija (por ejemplo, 2:1). Los datos de activación comprimidos generados por la unidad de compresión pueden incluir fragmentos de datos que tienen una longitud fija (por ejemplo, 32 bytes), que incluyen una porción de máscara de longitud fija (por ejemplo, 8 bytes) y una porción de datos de longitud fija (por ejemplo, 24 bytes).

Los bits de la porción de máscara de un fragmento de salida comprimido corresponden a bytes dentro de un fragmento de entrada descomprimido en una modalidad. Por ejemplo, el primer bit de una porción de máscara puede corresponder al primer byte en un fragmento de entrada descomprimido, el segundo bit de la porción de máscara puede corresponder al segundo byte en el fragmento de entrada descomprimido, etcétera. Los bits en la porción de máscara de los datos de activación comprimidos pueden establecerse en un cero lógico si el byte correspondiente en el fragmento de entrada descomprimido es cero y puede establecerse en uno lógico si el byte correspondiente en el fragmento de entrada descomprimido no es cero.

Tal como se abordó brevemente antes, dos o más bits en la porción de máscara del fragmento de salida comprimido corresponden a un byte en el fragmento de datos descomprimido en algunas

modalidades. En estas modalidades, los bits en la porción de máscara del fragmento de salida comprimido pueden indicar que no solo existe un byte correspondiente en el fragmento descomprimido sino también su magnitud aproximada.

La porción de datos de un fragmento de salida comprimido incluye los bytes distintos de cero de un fragmento de entrada descomprimido que se truncaron para representar los bytes distintos de cero del fragmento de entrada utilizando la cantidad de bits disponibles en el fragmento de datos comprimidos. El número de bits disponibles en la porción de datos del fragmento de salida comprimido para cada byte distinto de cero se determina en algunas modalidades al dividir la cantidad total de bits disponibles en la porción de datos (por ejemplo, 192 bits) entre la cantidad de bytes distintos de cero en el fragmento de entrada descomprimido. El resultado de este cálculo indica la cantidad de bits en la porción de datos del fragmento de salida comprimido que están disponibles para representar cada byte de datos distintos de cero en el fragmento de entrada descomprimido. Cualquiera de los bits restantes puede utilizarse para proporcionar un bit adicional para representar algunos de los valores distintos de cero en el fragmento de datos del fragmento de salida comprimido.

Una vez que se determine la cantidad de bits disponibles en la porción de datos del fragmento de salida comprimido para representar cada byte distinto de cero en el fragmento de entrada descomprimido, los LSB de los valores distintos de cero en el fragmento de entrada descomprimido se truncan para ajustarse dentro de la cantidad disponible de bits. Los MSB de los valores distintos de cero podrían truncarse en otras modalidades. Los valores truncados distintos de cero pueden almacenarse en la porción de datos del fragmento de salida comprimido. Este proceso puede repetirse para cada fragmento de valores de activación de entrada descomprimidos. Los fragmentos de salida comprimidos pueden almacenarse en la memoria del módulo dentro o fuera del módulo para su posterior descompresión y uso por las neuronas.

El módulo de DNN divulgado también puede incluir una unidad de descompresión para descomprimir los valores de activación que se comprimieron por la unidad de compresión de la manera descrita anteriormente. La unidad de descompresión recibe fragmentos de datos de activación comprimidos que incluyen una porción de máscara y una porción de datos. La unidad de descompresión puede utilizar los bits de la porción de máscara para identificar la cantidad de bytes distintos de cero que estarán presentes en un fragmento de salida descomprimido y sus ubicaciones dentro del fragmento de salida descomprimido. La máscara también indica las ubicaciones de cero bytes en el fragmento de salida descomprimido.

En algunas modalidades, la unidad de descompresión determina la cantidad de bits que se utilizaron por la unidad de compresión para representar cada byte distinto de cero al dividir el número total de bits disponibles en el fragmento de datos (por ejemplo, 192 bits) de un fragmento comprimido por la cantidad de bytes distintos de cero en el fragmento de entrada descomprimido según lo especificado por la máscara. La unidad de descompresión también puede suponer que la unidad de compresión utilizó cualquier bit restante para proporcionar un bit adicional para representar algunos de los valores distintos de cero en la porción de datos del fragmento comprimido (por ejemplo, los primeros N valores).

Para cada posición de bit en la máscara que es un cero lógico, la unidad de descompresión puede

insertar un byte cero en el fragmento de salida descomprimido en su posición correspondiente. Para cada posición de bit en la máscara que sea un uno lógico, la unidad de descompresión inserta los bytes distintos de cero truncados desde la posición correspondiente en la porción de datos del fragmento de entrada comprimido en la posición correspondiente en el fragmento de salida descomprimido. La unidad de descompresión también inserta ceros en los LSB o MSB, según corresponda, de los valores distintos de cero para reemplazar los bits que se truncaron durante la compresión.

En algunas modalidades, la unidad de descompresión agrega un valor de compensación a los valores distintos de cero truncados para garantizar que los valores descomprimidos distintos de cero no se conviertan en bytes cero cuando se descomprimen. El fragmento de salida descomprimido puede almacenarse en la memoria del módulo dentro o fuera del módulo para que las neuronas lo utilicen. A continuación se proporcionarán detalles adicionales sobre la operación del módulo de DNN, la unidad de compresión y la unidad de descompresión.

En la siguiente descripción detallada de la invención, hacen referencias a los dibujos adjuntos que forman parte de la misma, y que se muestran a modo de ilustración de las configuraciones o ejemplos específicos. Con referencia ahora a los dibujos, en los cuales números similares representan elementos similares a lo largo de las diversas figuras, se describirán aspectos de un módulo de DNN que puede comprimir y descomprimir datos de activación para reducir la utilización del ancho de banda del bus de memoria.

La figura 1 es un diagrama de arquitectura de cómputo que muestra los aspectos de la configuración y operación de un módulo de DNN 105 que implementa las tecnologías divulgadas en el presente documento, de acuerdo con una modalidad. El módulo de DNN 105 descrito en el presente documento se configura en algunas modalidades para resolver problemas de clasificación (y problemas relacionados) como, entre otros, detección de objetos, etiquetado semántico y extracción de características.

Para proporcionar esta funcionalidad, el módulo de DNN 105 puede implementar una red neuronal de solo recuperación y admitir programáticamente una amplia variedad de estructuras de red. El entrenamiento para la red implementada por el módulo de DNN 105 puede realizarse sin conexión en una granja de servidores, centro de datos u otro entorno de cómputo adecuado. El resultado de entrenar una DNN es un conjunto de parámetros que pueden conocer como “pesos” o “núcleos”. Estos parámetros representan una función de transformación que puede aplicarse a una entrada con el resultado como una clasificación o salida etiquetada semánticamente.

El módulo de DNN 105 divulgado en el presente documento puede considerarse un procesador superescalar. El módulo 105 de DNN puede enviar una o más instrucciones a múltiples unidades de ejecución, denominadas neuronas 105F. Las unidades de ejecución pueden ser “despacho simultáneo compleción simultánea”, donde cada unidad de ejecución se sincroniza con cada una de las otras unidades de ejecución. El módulo 105 de DNN puede clasificarse como una arquitectura de flujo de instrucción única, múltiples flujos de datos (SIMD, por sus siglas en inglés).

El módulo de DNN 105 incluye un número de neuronas 105F (por ejemplo, una potencia de dos).

Una neurona 105F es la unidad base en redes neuronales artificiales que se utiliza para modelar una neurona biológica en el cerebro. El modelo de una neurona 105F puede incluir el producto interno de un vector de entrada con un vector de peso agregado a un sesgo, con una función de activación aplicada. El procesamiento realizado por una neurona 105F en el módulo de DNN 105 divulgado en el presente documento está estrechamente relacionado con una neurona artificial.

Cada neurona 105F en el módulo de DNN 105 es capaz de realizar suma ponderada, agrupación máxima, derivación y potencialmente otros tipos de operaciones. Las neuronas 105F procesan datos de entrada y peso en cada ciclo de reloj. Cada neurona 105F está sincronizada con todas las demás neuronas 105F en términos de progreso dentro de un núcleo para minimizar el flujo de datos del núcleo dentro del módulo de DNN 105.

Cada neurona 105F puede contener un multiplicador, un sumador, un comparador y varios acumuladores (no mostrados en la figura 1). Al tener múltiples acumuladores, las neuronas 105F pueden mantener el contexto de múltiples núcleos activos diferentes a la vez. Cada acumulador puede cargarse desde una lectura de BaSRAM 150 (descrita posteriormente). Los acumuladores pueden sumarse con el contenido de otros acumuladores de otras neuronas 105F.

El módulo de DNN 105 acepta datos planos como entrada, como datos de imagen. Sin embargo, la entrada al módulo de DNN 105 no está limitada a datos de imagen. Más bien, el módulo de DNN 105 puede operar en cualquier dato de entrada presentado al módulo de DNN 105 en un formato plano uniforme. En una modalidad particular, el módulo de DNN 105 puede aceptar como entrada tramas de datos de uno o dos bytes de planos múltiples.

Cada trama de entrada puede convolucionar con un conjunto de núcleos $N \times K \times H \times W$, donde N es la cantidad de núcleos, K es la cantidad de canales por núcleo, H es la altura y W es el ancho. La convolución se realiza en intervalos superpuestos a través de los datos de entrada donde el intervalo se define mediante avances en las direcciones X y Y. Estas funciones se realizan por las neuronas 105F y se gestionan por el módulo de DNN 105 y los registros de control visibles por *software*.

El módulo de DNN 105 admite tres tipos de datos principales: pesos; datos de entrada/mapas de características; y datos de activación. Los datos de entrada/mapas de características y los datos de activación son, en la mayoría de los casos, dos nombres para los mismos datos con la distinción de que cuando se hace referencia a una salida de una capa se usa el término datos de activación. Cuando se hace referencia a la entrada de una capa, se utiliza el término datos de entrada/mapa de características.

Las neuronas 105F en el módulo de DNN 105 calculan una suma ponderada de sus entradas y pasan la suma ponderada a través de una “función de activación” o “función de transferencia”. La función de transferencia comúnmente tiene una forma sigmoidea, pero también puede adoptar la forma de una función lineal por partes, una función escalón u otro tipo de función. La función de activación permite que las neuronas 105F entrenen a un conjunto más grande de entradas y salidas deseadas donde los límites de clasificación no son lineales.

El módulo de DNN 105 opera en una lista de descriptores de capa que corresponden a las capas de una red neuronal. El módulo de DNN 105 puede tratar la lista de descriptores de capa como

instrucciones. Estos descriptores pueden recuperarse previamente desde la memoria al módulo de DNN 105 y ejecutarse en orden. La lista de descriptores actúa como un conjunto de instrucciones para el módulo de DNN 105. Las herramientas de *software* y/o compiladores pueden ejecutarse en dispositivos externos al módulo de DNN 105 para crear las listas de descriptores que se ejecutan en el módulo de DNN 105.

En general, pueden existir dos clases principales de descriptores: descriptores de movimiento de memoria a memoria (M2M, por sus siglas en inglés); y descriptores de operaciones. Los descriptores M2M pueden utilizarse para mover datos hacia/desde la memoria principal hacia/desde un búfer local (es decir, el búfer de línea 125 descrito posteriormente) para que los descriptores de operación los consuman. Los descriptores M2M siguen una canalización de ejecución diferente a la de los descriptores de operación. La canalización objetivo para los descriptores M2M puede ser el motor interno de DMA 105B o los registros de configuración 105G, mientras la canalización objetivo para los descriptores de operación pueden ser las neuronas 105F.

Los descriptores operativos especifican una operación específica que las neuronas 105F deben realizar en una estructura de datos ubicada en la memoria local, es decir, la memoria estática de acceso aleatorio (SRAM, por sus siglas en inglés). Los descriptores operativos se procesan en orden y son capaces de muchas y diferentes operaciones de capa, al menos algunas de las cuales se describen en el presente documento.

Como se ilustra en la figura 1, el módulo de DNN 105 tiene un subsistema de memoria con una estructura de memoria intermedia única L1 y L2. Los búferes L1 y L2 mostradas en la figura 1 están diseñados específicamente para el procesamiento de redes neuronales. A modo de ejemplo, el búfer L2 150 puede mantener una capacidad de almacenamiento seleccionada con una interfaz privada de alta velocidad que opera a una frecuencia seleccionada. El búfer 125 L1 puede mantener una capacidad de almacenamiento seleccionada que puede dividirse entre el núcleo y los datos de activación. El búfer L1 125 podría denominarse en el presente documento como el “búfer de línea 125”, y el búfer L2 150 podría denominarse en el presente documento como BaSRAM 150.

En algunas modalidades, los datos computacionales (es decir, datos de entrada, pesos y datos de activación) se almacenan en la fila principal de la BaSRAM 150. Los datos computacionales pueden organizarse como dos búferes de línea, donde un búfer de línea contiene datos de entrada, que podría denominarse en el presente documento como “búfer de entrada”, y el otro búfer de línea, que podría denominarse en el presente documento como “búfer de peso”, contiene los pesos de núcleo. Los búferes de línea se llenan a partir de la BaSRAM 150 por la unidad de carga/almacenamiento 105C. Los datos se acumulan en cada búfer de línea hasta que alcanza su capacidad predeterminada. Los datos del búfer de línea se copian después en un búfer de respaldo en algunas modalidades y se presentan a las neuronas 105F.

El módulo de DNN 105 también puede comprender una serie de otros componentes que incluyen, entre otros, una interfaz de registro 105G, una unidad de entrenamiento previo 105A, una unidad de guardado/restauración 105E, un controlador de capa 105D y una interfaz de registro 105G. En algunas modalidades, el módulo de DNN 105 puede incluir componentes adicionales o alternativos.

En algunas configuraciones, el módulo de DNN 105 opera en conjunción con otros componentes de cómputo externos. En algunas modalidades, por ejemplo, el módulo de DNN 105 está conectado a un procesador de un servidor de aplicación de sistema en chip (SoC, por sus siglas en inglés) 130. Por ejemplo, el módulo de DNN 105 puede conectarse al SoC servidor 130 a través de una interfaz PCIe. Pueden utilizarse componentes PCIe apropiados, como la terminal PCIe 135 para habilitar estas conexiones.

El SoC servidor 130 sirve como procesador de aplicaciones para el módulo de DNN 105. El sistema operativo principal, la aplicación y el procesamiento de sensor auxiliar se realizan por el SoC servidor 130. El SoC servidor 130 también puede conectarse a una fuente de datos de entrada 102, como una cámara externa, que proporciona datos de entrada, como datos de imagen, al módulo de DNN 105.

La DDR DRAM 155 también puede conectarse al SoC servidor 130 que puede utilizarse como la memoria principal del sistema. Puede accederse a esta memoria desde el SoC servidor 130 a través de la estructura de alto ancho de banda 120 (por ejemplo, bus PCIe) a través de un controlador de memoria 145. La estructura de alto ancho de banda 120 proporciona pequeñas transacciones de mensajes de acceso bidireccional directo a la memoria (DMA, por sus siglas en inglés) y mayores transacciones de DMA. Un puente 115 y una estructura de bajo ancho de banda 110 pueden conectar el módulo de DNN 105 al SoC servidor 130 para la configuración del submódulo y otras funciones.

El módulo de DNN 105 puede incluir un motor de DMA 105B que está configurado para mover datos hacia y desde la memoria principal 155. En algunas modalidades, el motor de DMA 105B tiene dos canales. Un canal está dedicado a buscar descriptores de operación, mientras el otro canal está dedicado a operaciones M2M. Puede incorporarse un descriptor DMA en el descriptor M2M. Los descriptores en este contexto son descriptores DMA que se utilizan para mover los contenidos de la memoria, que no deben confundirse con los descriptores de operación descritos anteriormente.

Para vaciar la memoria local de BaSRAM 150, y para proporcionar más espacio para datos de entrada y datos de peso, la salida de activación puede transmitirse directamente a la memoria DDR 155. Cuando se transmiten datos a la memoria DDR 155, el módulo de DNN 105 acumulará suficientes datos para una transacción por ráfagas en la estructura de alto ancho de banda 120 y almacenará suficientes transacciones para minimizar la contrapresión en las neuronas 105F. A continuación se proporcionarán detalles adicionales sobre la operación del módulo de DNN 105.

Las figuras 2A y 2B son diagramas de arquitectura del sistema de cómputo que muestran aspectos de la configuración y operación del módulo de DNN 105 para comprimir datos de activación, de acuerdo con una modalidad. Tal como se muestra en la figura 2A y se abordó brevemente antes, el módulo de DNN 105 incluye una o más neuronas 105F y una unidad de compresión 200. En otras modalidades, la unidad de compresión 200 se implementa mediante la unidad de carga/almacenamiento 105C en algunas modalidades, pero podría implementarse de otras maneras.

La unidad de compresión 200 puede recibir un fragmento de datos de activación descomprimido 202 generados por una o más de las neuronas 105F. En algunas modalidades, el fragmento de datos descomprimido 202 incluye una cantidad fija de bytes, como 64 bytes.

La unidad de compresión 200 puede comprimir el fragmento de datos descomprimido 202 para generar un fragmento de datos de activación comprimido 204. El fragmento de datos de activación comprimido 204 puede almacenarse en la memoria 206. Por ejemplo, el fragmento de datos de activación comprimido 204 puede almacenarse en la memoria LPDDR4 155 proporcionada por el servidor de aplicación o puede almacenarse en la BASRAM 150 proporcionada por el módulo de DNN 105. Tal como se divulgará con mayor detalle posteriormente, las tecnologías divulgadas en el presente documento pueden utilizar la compresión y descompresión para reducir la utilización del bus de memoria al almacenar o recuperar datos de activación comprimidos o descomprimidos en la memoria LPDDR4 155 o BASRAM 150. Detalles adicionales con respecto a estas tecnologías se divulgan posteriormente con respecto a las figuras 2A a 9.

Como se ilustra en la figura 2B, la unidad de compresión 200 puede generar una porción de máscara 208 y una porción de datos 210 de un fragmento comprimido de datos de salida 204. La porción de máscara 208 del fragmento de salida comprimido 204 incluye una cantidad de bits equivalente a la cantidad fija de bytes en el fragmento de datos descomprimido 202. Por ejemplo, si el fragmento de datos descomprimido 202 incluye 64 bytes de datos, la porción de máscara 208 del fragmento de salida comprimido 204 incluirá 64 bits (es decir, 8 bytes).

En algunas modalidades, cada bit en la porción de máscara 208 del fragmento de salida comprimido 204 corresponde a un byte en el fragmento de datos descomprimido 202. Por ejemplo, el bit uno de la porción de máscara 208 puede corresponder al primer byte en el fragmento de datos descomprimido 202, el bit dos de la porción de máscara 208 puede corresponder al segundo byte en el fragmento de datos descomprimido 202, etcétera.

La unidad de compresión 200 establece cada bit en la porción de máscara 208 del fragmento de salida comprimido 204 en un cero lógico donde un byte correspondiente en el fragmento de datos descomprimido 202 es un byte cero. La unidad de compresión 200 también establece cada bit en la porción de máscara 208 del fragmento de salida comprimido 204 en uno lógico, donde un byte correspondiente en el fragmento de datos descomprimido 202 es un byte distinto de cero. De esta manera, la porción de máscara 208 del fragmento de salida comprimido 204 codifica la presencia y ubicación de los bytes cero y distintos de cero en el fragmento de datos descomprimido 202.

La unidad de compresión 200 genera el fragmento de datos 210 del fragmento de salida comprimido 204 al determinar la cantidad de bytes distintos de cero en el fragmento de datos descomprimido 202. La unidad de compresión 200 determina entonces, con base en la cantidad de bytes distintos de cero en el fragmento de datos descomprimido 202 y la cantidad de bytes disponibles en el fragmento de datos 210 del fragmento de salida comprimido 204, la cantidad de bits en la porción de datos 210 del fragmento de salida comprimido 204 que está disponible para almacenar cada byte distinto de cero del fragmento de datos descomprimido 202. Por ejemplo, si la porción de datos 210 del fragmento de datos comprimido 204 tiene 24 bytes de ancho (es decir, 192 bits) y existen 47 bytes distintos de cero en el fragmento de datos descomprimido 202, existen cuatro bits disponibles en el fragmento de datos 210 para almacenar cada byte distinto de cero del fragmento de datos descomprimido 202.

En algunas modalidades, la unidad de compresión 200 también puede determinar la cantidad de bits adicionales, si están presentes, en la porción de datos 210 del fragmento de salida comprimido 204 que están disponibles para almacenar bytes distintos de cero del fragmento de datos descomprimido 202. En el ejemplo anterior, por ejemplo, cuatro bits adicionales están disponibles para almacenar bytes distintos de cero (es decir, $192 \bmod 47 = \text{cuatro bits}$). La unidad de compresión 200 puede asignar estos bits adicionales a uno o más de los bytes distintos de cero en el fragmento de datos descomprimido 204 antes de truncar uno o más de los bytes distintos de cero. Por ejemplo, la unidad de compresión 200 podría asignar estos bits adicionales a los primeros N bytes en la porción de datos 210 del fragmento de salida comprimido 204.

La unidad de compresión 200 trunca entonces los bytes distintos de cero en el fragmento de datos descomprimido 202 a la cantidad determinada de bits disponibles en el fragmento de datos 210 para almacenar cada byte distinto de cero (es decir, cuatro en el ejemplo dado anteriormente). En una modalidad, la unidad de compresión 200 trunca los LSB de los bytes distintos de cero para ajustarse dentro de la cantidad de bits disponibles en el fragmento de datos 210. En otra modalidad, la unidad de compresión 200 trunca los MSB de los bytes distintos de cero. La unidad de compresión 200 almacena los bytes distintos de cero truncados en la porción de datos 210 del fragmento de salida comprimido 204. El fragmento de salida comprimido 204, que incluye la porción de máscara 208 y el fragmento de datos 210, puede enviarse, por ejemplo, a la memoria interna en el módulo de DNN 105 o a la memoria externa de un servidor de aplicación del módulo de DNN 105. Detalles adicionales con respecto al proceso de compresión descrito anteriormente se proporcionarán posteriormente con respecto a las figuras 3 y 4.

En algunas modalidades, tal como se abordó brevemente antes, dos o más bits en la porción de máscara 208 del fragmento de salida comprimido 204 corresponden a un byte en el fragmento de datos descomprimido 202. En estas modalidades, los bits en la porción de máscara 208 del fragmento de salida comprimido 204 pueden indicar que no solo existe un byte correspondiente en el fragmento descomprimido 202 sino también su magnitud aproximada. Por ejemplo, y sin limitación, la porción de máscara 208 podría incluir dos bits por byte en el fragmento de datos descomprimido 202. En este ejemplo, 00 puede indicar que el MSB del valor distinto de cero correspondiente en el fragmento de datos descomprimido 202 es cero, 01 puede indicar que el MSB es <64 , 10 puede indicar que el MSB es <128 y 11 puede indicar que el MSB >128 . Estos valores pueden utilizarse para identificar qué los MSB de los bytes en el fragmento de datos descomprimido 202 pueden truncarse. Por ejemplo, si el MSB de un byte particular es <64 , entonces los dos MSB superiores pueden truncarse sin pérdida de datos.

La figura 3, de acuerdo con una modalidad, es un diagrama de estructura de datos que ilustra aspectos de la operación del módulo de DNN 105 para comprimir fragmentos de datos de activación descomprimidos 202 con referencia a un fragmento de datos de activación descomprimido 202 ejemplar. En el ejemplo mostrado en la figura 3, un fragmento de datos de activación descomprimido 202 tiene 64 bytes de longitud. Los bytes cero, uno y 63 del fragmento de datos de activación descomprimido 202 son bytes cero. Los bytes dos, tres y 62 del fragmento de datos de activación descomprimido 202 son bytes distintos de cero, que almacenan los valores 112, 121 y dos, respectivamente. Los bytes 4 a 61 del

fragmento de datos de activación descomprimido 202 ejemplar pueden almacenar bytes cero o distintos de cero.

Tal como se abordó anteriormente, la unidad de compresión 200 puede generar una porción de máscara 208 que codifica la presencia y ubicación de los bytes cero y distintos de cero en el fragmento de datos de activación descomprimido 202. En este ejemplo, por ejemplo, los bits cero, uno y 63 de la porción de máscara 208 se han establecido en ceros lógicos para indicar la presencia de bytes cero en las ubicaciones correspondientes en el fragmento de datos de activación descomprimido 202. De manera similar, los bits dos, tres y 62 de la porción de máscara 208 se han establecido en uno lógico para indicar que los bytes dos, tres y 62 del fragmento de datos de activación descomprimido 202 almacenan bytes distintos de cero.

Tal como se abordó anteriormente, la unidad de compresión 200 genera la porción de datos 210 del fragmento de salida comprimido 204 al determinar la cantidad de bytes distintos de cero en el fragmento de datos descomprimido 202. En el ejemplo mostrado en la figura 3, por ejemplo, el fragmento de datos descomprimido 202 incluye 47 bytes distintos de cero (no todos se muestran en la figura 3). La unidad de compresión 200 determina entonces, con base en la cantidad de bytes distintos de cero en el fragmento de datos descomprimido 202 y la cantidad de bytes disponibles en la porción de datos 210 del fragmento de salida comprimido 204, la cantidad de bits en la porción de datos 210 del fragmento de salida comprimido 204 que está disponible para almacenar cada byte distinto de cero del fragmento de datos descomprimido 202.

En el ejemplo mostrado en la figura 3, por ejemplo, la porción de datos 210 del fragmento de datos comprimido 204 tiene 24 bytes de ancho (es decir, 192 bits) y existen 47 bytes distintos de cero en el fragmento de datos descomprimido 202. Como resultado, existen cuatro bits disponibles en el fragmento de datos 210 para almacenar cada byte distinto de cero del fragmento de datos descomprimido 202 (es decir, $192/47 = 4$ restante 4).

Como también se abordó anteriormente, la unidad de compresión 200 también puede determinar la cantidad de bits adicionales, si están presentes, en la porción de datos 210 del fragmento de salida comprimido 204 que están disponibles para almacenar bytes distintos de cero del fragmento de datos descomprimido 202. En el ejemplo mostrado en la figura 3, por ejemplo, cuatro bits adicionales están disponibles para almacenar bytes distintos de cero (es decir, $192 \bmod 47 =$ cuatro bits). La unidad de compresión 200 puede asignar estos bits adicionales a uno o más de los bytes distintos de cero en el fragmento de datos descomprimido 204 antes de truncar uno o más de los bytes distintos de cero. En el ejemplo mostrado en la figura 3, uno de los cuatro bits adicionales se asignó a cada uno de los primeros cuatro bytes distintos de cero en el fragmento de datos de activación descomprimido 202. Como resultado, los primeros cuatro bytes del fragmento de datos de activación descomprimido 202 se truncarán a cinco bits en lugar de cuatro.

La unidad de compresión 200 trunca los bytes distintos de cero en el fragmento de datos descomprimido 202 a la cantidad determinada de bits disponibles en el fragmento de datos 210 para almacenar cada byte distinto de cero (es decir, cinco bits para los primeros cuatro bytes distintos de cero

en el ejemplo dado anteriormente). En el ejemplo mostrado en la figura 3, en una modalidad, la unidad de compresión 200 trunca los LSB de los bytes distintos de cero para que se ajusten dentro de la cantidad disponible de bits (es decir, cuatro en este ejemplo) en el fragmento de datos 210. En otra modalidad, la unidad de compresión 200 trunca los MSB de los bytes distintos de cero.

Tal como se muestra en la figura 3, el segundo byte del fragmento de datos de activación descomprimido 202 almacena el valor 113 (01110001). Debido a que se asignaron cinco bits a los primeros cuatro valores distintos de cero en el fragmento de datos de activación descomprimido 202, los tres LSB de este valor se truncan, lo cual hace que el valor 01110 se almacene en la primera ubicación en el fragmento de datos de activación comprimido 210. El tercer byte del fragmento de datos de activación descomprimido 202 almacena el valor 121 (01111001). Debido a que se asignaron cinco bits a los primeros cuatro valores distintos de cero en el fragmento de datos de activación descomprimido 202, los tres LSB de este valor se truncan, lo cual hace que el valor 01111 se almacene en la segunda ubicación en el fragmento de datos de activación comprimido 210.

En el ejemplo mostrado en la figura 3, el byte 62 del fragmento de datos de activación descomprimido 202 almacena el valor 2 (00000010). Debido a que se asignaron cuatro bits a los valores distintos de cero 5 a 63 en el fragmento de datos de activación descomprimido 202, los cuatro LSB de este valor se truncan, lo cual hace que el valor 0000 se almacene en la ubicación 62 en el fragmento de datos de activación comprimido 210. Otros bytes distintos de cero en el fragmento de datos de activación descomprimido 202 pueden truncarse y almacenarse en la porción de datos 210 del fragmento de datos de activación comprimido 204 de manera similar.

Una vez que todos los bytes distintos de cero del fragmento de datos de activación descomprimido 202 se almacenaron en el fragmento de datos 203, la unidad de compresión 200 almacena el fragmento de salida comprimido 204, que incluye la porción de máscara 208 y la porción de datos 210, por ejemplo, en memoria interna en el módulo de DNN 105 o memoria externa de un servidor de aplicación del módulo de DNN 105. A continuación se proporcionan detalles adicionales con respecto al proceso de compresión con respecto a la figura 4.

La figura 4 es un diagrama de flujo que muestra una rutina 400 que ilustra aspectos de la operación del módulo de DNN 105 para comprimir fragmentos de datos de activación descomprimidos 202, de acuerdo con una modalidad descrita en el presente documento. Debe apreciarse que las operaciones lógicas descritas en el presente documento con respecto a la figura 4, y otras figuras, pueden implementarse (1) como una secuencia de acciones implementadas por computadora o módulos de programa que se ejecutan en un dispositivo de cómputo y/o (2) como circuitos lógicos de máquina interconectados o módulos de circuito dentro de un dispositivo de cómputo.

La implementación particular de las tecnologías divulgadas en el presente documento es una cuestión de elección que depende del rendimiento y otros requisitos del dispositivo de cómputo. Por consiguiente, las operaciones lógicas descritas en el presente documento se denominan de manera diversa como estados, operaciones, dispositivos estructurales, acciones o módulos. Estos estados, operaciones, dispositivos estructurales, acciones y módulos pueden implementarse en *hardware*, *software*, *firmware*,

lógica digital de uso especial y cualquier combinación de los mismos. Debe apreciarse que pueden realizarse más o menos operaciones que las mostradas en las figuras y descrito en el presente documento. Estas operaciones también pueden realizarse en un orden diferente al descrito en el presente documento.

La rutina 400 comienza en la operación 402, donde la unidad de compresión 200 determina la cantidad de bytes distintos de cero en el fragmento de datos de activación descomprimido 202. La rutina 400 entonces procede a la operación 404, donde la unidad de compresión 200 determina si la cantidad de bytes distintos de cero en el fragmento de datos de activación descomprimido 202 es menor o igual que la cantidad de bytes disponibles en la porción de datos 210 del fragmento de datos de activación comprimido 204. Los bytes distintos de cero del fragmento de datos de activación descomprimido 202 no necesitan comprimirse si la cantidad de bytes distintos de cero es menor o igual que la cantidad de bytes disponibles en la porción de datos 210 del fragmento de datos de activación comprimido 204. Por consiguiente, en este caso, la rutina 400 procede a la operación 408, donde los bytes distintos de cero se almacenan en la porción de datos 210 sin truncamiento.

Si la cantidad de bytes distintos de cero en el fragmento de datos de activación descomprimido 202 es mayor que la cantidad de bytes disponibles en la porción de datos 210 del fragmento de datos de activación comprimido 204, la rutina 400 procede desde la operación 406 a la operación 412. En la operación 412, la unidad de compresión 200 determina la cantidad de bits de la porción de datos 210 del fragmento de datos de salida comprimido 204 disponibles para almacenar los bytes distintos de cero truncados del fragmento de datos de activación descomprimido 202 de la manera descrita anteriormente. La rutina 400 entonces procede desde la operación 412 a la operación 414.

En la operación 414, la unidad de compresión 200 determina la cantidad de bits adicionales, si están presentes, en la porción de datos 210 del fragmento de salida comprimido 204 que están disponibles para almacenar bytes distintos de cero del fragmento de datos descomprimido 202. Tal como se abordó anteriormente, la unidad de compresión 200 puede asignar estos bits adicionales a uno o más de los bytes distintos de cero en el fragmento de datos descomprimido 204 antes de truncar uno o más de los bytes distintos de cero. Esto ocurre en la operación 416.

Desde la operación 416, la rutina 400 procede a la operación 418, donde la unidad de compresión 200 establece bits en la porción de máscara 208 del fragmento de datos de activación comprimido 204 a uno lógico donde el byte correspondiente en el fragmento de activación descomprimido 202 es distinto de cero. La unidad de compresión 200 también establece bits en la porción de máscara 208 del fragmento de datos de activación comprimido 204 a un cero lógico donde el byte correspondiente en el fragmento de activación descomprimido 202 es cero.

Desde la operación 418, la rutina 400 procede a la operación 420, donde la unidad de compresión 200 trunca los LSB o MSB de los bytes distintos de cero en el fragmento de datos descomprimido 202 a la cantidad determinada de bits disponibles en la porción de datos 210 para cada byte distinto de cero. Los bytes distintos de cero truncados se almacenan en la porción de datos 210 del fragmento de datos de activación comprimido 204. La unidad de compresión 200 almacena entonces el fragmento de salida comprimido 204, que incluye la porción de máscara 208 y la porción de datos 210 en la memoria interna

en el módulo de DNN 105 o en la memoria externa de un servidor de aplicación del módulo de DNN 105. Desde las operaciones 408 y 420, la rutina 400 procede a la operación 410, donde concluye.

Las figuras 5A y 5B, de acuerdo con una modalidad, son diagramas de arquitectura del sistema de cómputo que muestran aspectos de la configuración y la operación del módulo de DNN 105 para descomprimir datos de activación comprimidos. Tal como se abordó brevemente antes, y tal como se muestra en las figuras 5A y 5B, el módulo de DNN 105 también puede incluir una unidad de descompresión 500 que puede descomprimir fragmentos de datos de activación 204 que se comprimieron de la manera descrita anteriormente.

Por ejemplo, la unidad de descompresión 500 puede recibir un fragmento de datos de activación comprimido 204 desde el almacenamiento 206, como la memoria en el procesador de DNN o la memoria de un servidor de aplicación. La unidad de descompresión 500 puede entonces determinar la cantidad de bytes distintos de cero en la porción de datos 210 del fragmento de datos comprimido 204 con base en la cantidad de bits verdaderos lógicos en la porción de máscara 208 del fragmento comprimido 204. La unidad de descompresión 500 también puede determinar las ubicaciones de los bytes distintos de cero en el fragmento de datos descomprimido 502 con base en las ubicaciones de los bits verdaderos lógicos en la porción de máscara 208 del fragmento de salida comprimido 204. La unidad de descompresión 500 puede determinar las ubicaciones de los bytes cero en el fragmento de datos descomprimido 502 de manera similar.

La unidad de descompresión 500 también puede determinar la cantidad de bits utilizados por la unidad de compresión 200 para almacenar cada uno de los bytes distintos de cero truncados en la porción de datos 210 del fragmento de salida comprimido 204. La unidad de descompresión 500 puede determinar la cantidad de bits utilizados para almacenar cada byte distinto de cero truncado con base en el número de bytes distintos de cero en el fragmento de datos comprimido 204 (tal como se indica por la porción de máscara 208) y el tamaño objetivo del fragmento de salida descomprimido 502.

En el ejemplo anterior, por ejemplo, si la porción de datos del fragmento de datos comprimido 204 tiene 24 bytes de ancho (es decir, 192 bits) y existen 47 bytes distintos de cero en el fragmento de datos descomprimido 202, esto significa que la unidad de compresión 200 utilizó cuatro bits para almacenar cada byte distinto de cero truncado del fragmento de datos descomprimido 202 en la porción de datos 210. La unidad de descompresión 500 también puede determinar la cantidad de bits adicionales, si están presentes, que la unidad de compresión 200 asignó a uno o más de los bytes distintos de cero truncados almacenados en la porción de datos 210 del fragmento de salida comprimido 204.

Para cada posición de bit en la porción de máscara 208 del fragmento de salida comprimido 204 que es un cero lógico, la unidad de descompresión 500 inserta un byte cero en la posición correspondiente del fragmento de salida descomprimido 502. Para cada posición en la porción de máscara 208 que es un uno lógico, la unidad de descompresión 500 inserta el byte distinto de cero truncado desde la posición correspondiente del fragmento de entrada comprimido 204 en una posición correspondiente del fragmento de salida descomprimido 502 junto con una cantidad de bits cero equivalentes a la cantidad de bits truncados durante la compresión del fragmento de salida comprimido 204. Los bits cero pueden insertarse

en los LSB o MSB de los bytes distintos de cero truncados, dependiendo de qué bits se truncaron durante la compresión.

En algunas modalidades, tal como se mencionó anteriormente, la unidad de descompresión 500 también agrega un desplazamiento (por ejemplo, 00000001) a uno o más de los bytes distintos de cero truncados almacenados en el fragmento de salida descomprimido 502. Por ejemplo, puede agregarse un desplazamiento a bytes distintos de cero del fragmento de datos descomprimido 202 que se convierten en bytes cero después de la compresión. De esta manera, los bytes distintos de cero no se convertirán en bytes cero cuando se descompriman.

La figura 6, de acuerdo con una modalidad, es un diagrama de estructura de datos que ilustra aspectos de la operación del módulo de DNN 105 para descomprimir datos de activación con referencia a un fragmento de datos de activación comprimido ejemplar. El ejemplo que se muestra en la figura 6 ilustra la descompresión de los datos de activación comprimidos 204 generados en el ejemplo descrito anteriormente con respecto a la figura 3. Tal como se muestra en la figura 6, la porción de máscara 208 almacena ceros en los bits cero, uno y 63 y almacena los unos en los bits dos, tres y 62. La porción de datos 210 almacena los valores 01110, 01111 y 0000 de la manera mostrada en la figura 6.

A medida que la unidad de descompresión 500 realiza las operaciones de procesamiento descritas anteriormente, el cero lógico en la posición del primer bit de la porción de máscara 208 hará que la unidad de descompresión 500 almacene un byte cero como el primer byte del fragmento de datos de activación descomprimido 502. De manera similar, el cero lógico en la posición del segundo bit de la porción de máscara 208 hará que la unidad de descompresión 500 almacene un byte cero como el segundo byte del fragmento de datos descomprimido 502.

El uno lógico en la tercera posición de la porción de máscara 208 hará que la unidad de descompresión 500 recupere los primeros cinco bits (es decir, 01110) de la porción de datos 210 e inserte tres LSB, lo cual da como resultado que el valor 01110000 (112) se almacene como el tercer byte del fragmento de datos de activación descomprimido 502. De forma similar, el uno lógico en la posición del cuarto bit de la porción de máscara 208 hará que la unidad de descompresión 500 recupere los segundos cinco bits (es decir, 01111) de la porción de datos 210 e inserte tres LSB, lo cual da como resultado que el valor 01111000 (120) se almacene como el cuarto byte del fragmento de datos de activación descomprimido 502.

El uno lógico en la posición de bit 63 de la porción de máscara 208 hará que la unidad de descompresión 500 recupere los últimos cuatro bits de la porción de datos 210 (es decir, 0000) e inserte cuatro LSB cero, lo cual da como resultado que el valor de cero se almacene en la posición de del byte 63 del fragmento de datos de activación descomprimido 502. El cero lógico en la posición del último bit de la porción de máscara 208 hará que la unidad de descompresión 500 almacene un byte cero como el último byte del fragmento de datos descomprimido 502.

Tal como se abordó anteriormente, la unidad de descompresión 500 puede agregar un valor de desplazamiento a ciertos bytes en el fragmento de datos de activación descomprimido 502. Por ejemplo, la unidad de descompresión 500 puede agregar un valor de desplazamiento, como 00000001, a los bytes

distintos de cero en el fragmento de datos de activación descomprimido 202 pero que se comprimieron a bytes cero en el fragmento de datos de activación comprimido 204.

En el ejemplo mostrado en la figura 6, el último byte en la porción de datos 210 no fue cero (es decir, dos) en el fragmento de datos de activación descomprimido 202 pero se convirtió en cero en el fragmento de datos de activación comprimido 504. En consecuencia, la unidad de descompresión 500 puede agregar un valor de desplazamiento, como 00000001, a este byte, asegurando así que los bytes distintos de cero en los fragmentos de datos de activación descomprimidos 202 no se comprimirán a bytes cero.

La figura 7 es un diagrama de flujo que muestra una rutina 700 que ilustra aspectos de la operación del módulo de DNN 105 para descomprimir datos de activación, de acuerdo con una modalidad divulgada en el presente documento. La rutina 700 comienza en la operación 702, donde la unidad de descompresión 500 utiliza la porción de máscara 208 de un fragmento de datos de activación comprimido 204 para determinar la cantidad de bytes distintos de cero y sus ubicaciones en el fragmento de datos de activación descomprimido 502.

La rutina 700 procede desde la operación 702 a la operación 704, donde la unidad de descompresión 500 determina si la cantidad de bytes distintos de cero en el fragmento de datos de activación comprimido 204 es menor o igual que la cantidad de bytes del fragmento de datos de activación descomprimido 502. Tal como se abordó anteriormente, los bytes distintos de cero del fragmento de datos de activación comprimido 204 no necesitan descomprimirse si la cantidad de bytes distintos de cero es menor o igual que la cantidad de bytes del fragmento de datos de activación descomprimido 502. Por consiguiente, en este caso, la rutina 700 procede a la operación 708, donde los bytes distintos de cero en el fragmento de datos de activación comprimido 204 se almacenan en el fragmento de datos de activación descomprimido 502 sin modificación.

Si la cantidad de bytes distintos de cero en el fragmento de datos de activación comprimido 504 es mayor que la cantidad de bytes en el fragmento de datos de activación descomprimido 502, la rutina 700 procede desde la operación 706 a la operación 712. En la operación 712, la unidad de descompresión 500 determina la cantidad de bits de la porción de datos 210 del fragmento de datos de salida comprimido 204 que la unidad de compresión 200 utilizó para almacenar cada byte distinto de cero truncado del fragmento de datos de activación descomprimido 202. La rutina 700 entonces procede desde la operación 712 a la operación 714 de la manera descrita anteriormente.

En la operación 714, la unidad de descompresión 500 determina la cantidad de bits adicionales, si están presentes, que se utilizaron para almacenar bytes distintos de cero del fragmento de datos descomprimido 202. La unidad de descompresión 500 puede asignar estos bits adicionales a uno o más de los bytes distintos de cero en el fragmento de datos descomprimido 502 de la manera descrita anteriormente. Esto ocurre en la operación 716.

Desde la operación 716, la rutina 700 procede a la operación 718, donde la unidad de descompresión 500 inserta un byte cero en la posición correspondiente del fragmento de salida descomprimido 502 para cada posición de bit en la porción de máscara 208 del fragmento de salida

comprimido 204 que es un cero lógico. Para cada posición de bit en la porción de máscara 208 del fragmento de salida comprimido 204 que es un uno lógico, la unidad de descompresión 500 inserta los bytes distintos de cero truncados desde las posiciones correspondientes del fragmento de entrada comprimido 204 en una posición correspondiente del fragmento de salida descomprimido 502 junto con una cantidad de bits cero equivalente a la cantidad de bits truncados durante la compresión del fragmento de salida comprimido 204. Los bits cero pueden insertarse en los LSB o MSB de los bytes distintos de cero truncados, dependiendo de qué bits se truncaron durante la compresión. Esto ocurre en la operación 720.

En algunas modalidades, la unidad de descompresión 500 también puede agregar un valor de desplazamiento a uno o más de los bytes distintos de cero truncados almacenados en el fragmento de salida descomprimido 502. Por ejemplo, puede agregarse un desplazamiento a bytes distintos de cero del fragmento de datos descomprimido 202 que se convierten en bytes cero después de la compresión. De esta manera, los bytes distintos de cero no se convertirán en bytes cero cuando se comprimen y descomprimen. En otras modalidades, puede agregarse un desplazamiento a todos los bytes en el fragmento de datos de activación descomprimido 502.

La unidad de descompresión 500 almacena el fragmento de salida descomprimido 502 en la memoria interna en el módulo de DNN 105 o en la memoria externa de un servidor de aplicación del módulo de DNN 105 para uso de las neuronas 105F. Desde las operaciones 708 y 720, la rutina 400 procede a la operación 710, donde concluye.

La figura 8 es un diagrama de arquitectura de cómputo que muestra una arquitectura ilustrativa de *hardware* y *software* de cómputo para un dispositivo de cómputo que puede actuar como un servidor de aplicación para el módulo de DNN 105 presentado en el presente documento. En particular, la arquitectura ilustrada en la figura 8 puede utilizarse para implementar una computadora servidor, teléfono móvil, un lector electrónico, un teléfono inteligente, una computadora de escritorio, un dispositivo AR/VR, una tableta, una computadora portátil u otro tipo de dispositivo de cómputo adecuado para utilizarse con el módulo de DNN 105.

La computadora 800 ilustrada en la figura 8 incluye una unidad de procesamiento central (CPU, por sus siglas en inglés) 802, una memoria de sistema 804, que incluye una memoria de acceso aleatorio (RAM, por sus siglas en inglés) 806 y una memoria exclusivamente de lectura (ROM, por sus siglas en inglés) 808, y un bus de sistema 810 que conecta la memoria 804 a la CPU 802. Un sistema básico de entrada/salida (BIOS, por sus siglas en inglés o *firmware*) que contiene las rutinas básicas que ayudan a transferir información entre elementos dentro de la computadora 800, como durante el arranque, puede almacenarse en la ROM 808. La computadora 800 incluye, además, un dispositivo de almacenamiento de gran capacidad 812 para almacenar un sistema operativo 822, programas de aplicación y otros tipos de programas. El dispositivo de almacenamiento de gran capacidad 812 también puede configurarse para almacenar otros tipos de programas y datos.

El dispositivo de almacenamiento de gran capacidad 812 está conectado a la CPU 802 a través de un controlador de almacenamiento de gran capacidad (no mostrado) conectado al bus 810. El dispositivo de almacenamiento de gran capacidad 812 y sus medios legibles por computadora asociados proporcionan

almacenamiento no volátil para la computadora 800. Aunque la descripción de los medios legibles por computadora contenidos en el presente documento se refiere a un dispositivo de almacenamiento de gran capacidad, como un disco duro, una unidad de CD-ROM, una unidad de DVD-ROM o un dispositivo de almacenamiento USB, los expertos en la materia deben apreciar que los medios legibles por computadora pueden ser cualquier medio de almacenamiento por computadora disponible o medios de comunicación a los cuales pueda acceder la computadora 800.

Los medios de comunicación incluyen instrucciones legibles por computadora, estructuras de datos, módulos de programa u otros datos en una señal de datos modulada, como una onda portadora u otro mecanismo de transporte, e incluyen cualquier medio de suministro. El término “señal de datos modulada” significa una señal que tiene una o más de sus características modificadas o configuradas de manera que codifique información en la señal. A modo de ejemplo, y sin limitación, los medios de comunicación incluyen medios alámbricos como una red alámbrica o conexión directa, y medios inalámbricos como acústicos, radiofrecuencia, infrarrojos y otros medios inalámbricos. Las combinaciones de cualquiera de los anteriores también deben incluirse dentro del alcance de los medios legibles por computadora.

A modo de ejemplo, y sin limitación, los medios de almacenamiento por computadora pueden incluir medios volátiles y no volátiles, extraíbles y no extraíbles implementados en cualquier método o tecnología para el almacenamiento de información, como instrucciones legibles por computadora, estructuras de datos, módulos de programa u otros datos. Por ejemplo, los medios de almacenamiento por computadora incluyen, entre otros, RAM, ROM, EPROM, EEPROM, memoria flash u otra tecnología de memoria en estado sólido, CD-ROM, discos versátiles digitales (DVD, por sus siglas en inglés), HD-DVD, Blu-ray, u otro almacenamiento óptico, casetes magnéticos, cinta magnética, almacenamiento en disco magnético u otros dispositivos de almacenamiento magnético, o cualquier otro medio que pueda utilizarse para almacenar la información deseada y a la que pueda acceder la computadora 800. Para los fines de las reivindicaciones, la frase “medio de almacenamiento de cómputo”, y sus variaciones, no incluye ondas o señales *per se* o medios de comunicación.

De acuerdo con diversas configuraciones, la computadora 800 puede operar en un entorno en red utilizando conexiones lógicas a computadoras remotas a través de una red tal como la red 820. La computadora 800 puede conectarse a la red 820 a través de una unidad de interfaz de red 816 conectada al bus 810. Debe apreciarse que la unidad de interfaz de red 816 también puede utilizarse para conectarse a otros tipos de redes y sistemas de cómputo remotos. La computadora 800 también puede incluir un controlador de entrada/salida 818 para recibir y procesar la entrada de varios otros dispositivos, que incluyen un teclado, ratón, entrada táctil, un lápiz electrónico (no mostrado en la figura 8), o un sensor físico como una cámara de video. De manera similar, el controlador de entrada/salida 818 puede proporcionar salida a una pantalla de presentación u otro tipo de dispositivo de salida (tampoco mostrado en la figura 8).

Debe apreciarse que los componentes de *software* descritos en el presente documento, cuando se cargan en la CPU 802 y se ejecutan, pueden transformar la CPU 802 y la computadora 800 en general de un dispositivo de cómputo de uso general en un dispositivo de cómputo de uso especial personalizado para

facilitar la funcionalidad presentada en el presente documento. La CPU 802 puede construirse a partir de cualquier número de transistores u otros elementos de circuito discreto, que pueden asumir de forma individual o colectiva cualquier cantidad de estados. Más específicamente, la CPU 802 puede funcionar como una máquina de estados finitos, en respuesta a las instrucciones ejecutables contenidas en los módulos de *software* divulgados en el presente documento. Estas instrucciones ejecutables por computadora pueden transformar la CPU 802 al especificar cómo la CPU 802 cambia entre estados, transformando así los transistores u otros elementos de *hardware* discretos que constituyen la CPU 802.

La codificación de los módulos de *software* presentados en el presente documento también puede transformar la estructura física de los medios legibles por computadora presentados en el presente documento. La transformación específica de la estructura física depende de varios factores, en diferentes implementaciones de esta descripción. Los ejemplos de tales factores incluyen, entre otros, la tecnología utilizada para implementar los medios legibles por computadora, si los medios legibles por computadora se caracterizan como almacenamiento primario o secundario, y similares. Por ejemplo, si los medios legibles por computadora se implementan como memoria basada en semiconductores, el *software* descrito en el presente documento puede codificarse en los medios legibles por computadora transformando el estado físico de la memoria de semiconductores. Por ejemplo, el *software* puede transformar el estado de transistores, condensadores u otros elementos de circuito discreto que constituyen la memoria de semiconductores. El *software* también puede transformar el estado físico de tales componentes para almacenar datos.

Como otro ejemplo, los medios legibles por computadora divulgados en el presente documento pueden implementarse utilizando la tecnología magnética u óptica. En tales implementaciones, el *software* presentado en el presente documento puede transformar el estado físico de los medios magnéticos u ópticos, cuando el *software* está codificado en los mismos. Estas transformaciones pueden incluir alterar las características magnéticas de ubicaciones particulares dentro de los medios magnéticos dados. Estas transformaciones también pueden incluir la alteración de las características físicas o características de ubicaciones particulares dentro de medios ópticos dados, para cambiar las características ópticas de esas ubicaciones. Son posibles otras transformaciones de los medios físicos sin apartarse del alcance y la esencia de la presente descripción, con los ejemplos anteriores proporcionados solo para facilitar este análisis.

Considerando lo anterior, debe apreciarse que muchos tipos de transformaciones físicas se producen en la computadora 800 para almacenar y ejecutar los componentes de *software* presentados en el presente documento. También debe apreciarse que la arquitectura mostrada en la figura 8 para la computadora 800, o una arquitectura similar, puede utilizarse para implementar otros tipos de dispositivos de cómputo, que incluyen computadoras portátiles, dispositivos de videojuegos, sistemas de cómputo integrados, dispositivos móviles como teléfonos inteligentes, tabletas y dispositivos AR/VR, y otros tipos de dispositivos de cómputo conocidos para los expertos en la materia. También se contempla que la computadora 800 podría no incluir todos los componentes mostrados en la figura 8, puede incluir otros componentes que no se muestran explícitamente en la figura 8, o puede utilizar una arquitectura

completamente diferente a la mostrada en la figura 8.

La figura 9 es un diagrama de red que ilustra un entorno de cómputo de red distribuido 900 en el que pueden implementarse aspectos de las tecnologías divulgadas, de acuerdo con diversas modalidades presentadas en el presente documento. Tal como se muestra en la figura 9, uno o más servidores de cómputo 900A pueden interconectarse a través de una red de comunicaciones 820 (que puede ser de una LAN inalámbrica o fija, WAN, intranet, extranet, red entre pares, red privada virtual, internet, red de comunicaciones mediante Bluetooth, red propietaria de comunicaciones de bajo voltaje u otra red de comunicaciones) con una serie de dispositivos clientes de cómputo, como, entre otros, computadora tipo tableta 900B, una consola de juegos 900C, un reloj inteligente 900D, un teléfono 900E, como un teléfono inteligente, una computadora personal 900F y un dispositivo AR/VR 900G.

En un entorno de red en el cual la red de comunicaciones 820 es internet, por ejemplo, la computadora servidor 900A puede ser una computadora servidor dedicada operable para procesar y comunicar datos hacia y desde los dispositivos de cómputo cliente 900B a 900G a través de cualquiera de una serie de protocolos, como el protocolo de transferencia de hipertexto (HTTP, por sus siglas en inglés), protocolo de transferencia de archivos (FTP, por sus siglas en inglés) o protocolo simple de acceso a objetos (SOAP, por sus siglas en inglés). Además, el entorno de cómputo de red 900 puede utilizar varios protocolos de seguridad de datos, como la capa de puertos seguros (SSL, por sus siglas en inglés) o privacidad bastante buena (PGP, por sus siglas en inglés). Cada uno de los dispositivos de cómputo cliente 900B a 900G puede estar equipado con un sistema operativo que funciona para ser compatible con una o más aplicaciones de cómputos o sesiones de terminal, como un navegador web (no mostrado en la figura 9) u otra interfaz gráfica de usuario (no mostrada en la figura 9), o un entorno de escritorio móvil (no mostrado en la figura 9) para obtener acceso a la computadora servidor 900A.

La computadora servidor 900A puede conectarse comunicativamente a otros entornos de cómputo (no mostrados en la figura 9) y recibir datos con respecto a las interacciones/red de recursos de un usuario participante. En una operación ilustrativa, un usuario (no mostrado en la figura 9) puede interactuar con una aplicación de cómputo que se ejecuta en un dispositivo de cómputo cliente 900B a 900G para obtener los datos deseados y/o utilizar otras aplicaciones de cómputos.

Los datos y/o aplicaciones de cómputo pueden almacenarse en el servidor 900A, o servidores 900A, y comunicarse a los usuarios colaborativos mediante los dispositivos de cómputo cliente 900B a 900G a través de una red de comunicaciones 820 ejemplar. Un usuario participante (no mostrado en la figura 9) puede solicitar acceso a datos y aplicaciones específicos alojados en su totalidad o parcialmente en la computadora servidor 8800A. Estos datos pueden comunicarse entre los dispositivos de cómputo cliente 900B a 900G y la computadora servidor 900A para su procesamiento y almacenamiento.

La computadora servidor 900A puede alojar aplicaciones de cómputos, procesos y miniaplicaciones para la generación, autenticación, cifrado y comunicación de datos y aplicaciones, y puede cooperar con otros entornos de servidores de cómputo (no mostrados en la figura 9), proveedores de servicios externos (no mostrados en la figura 9), almacenamiento conectado a la red (NAS, por sus siglas en inglés) y redes de área de almacenamiento (SAN, por sus siglas en inglés) para realizar transacciones

de aplicaciones/datos.

Debe apreciarse que la arquitectura de cómputo mostrada en la figura 8 y el entorno de cómputo de red distribuida mostrado en la figura 9 se simplificaron para facilitar el análisis. También debe apreciarse que la arquitectura de cómputo y la red de cómputo distribuida pueden incluir y utilizar muchos más componentes de cómputo, dispositivos, programas de *software*, dispositivos de red y otros componentes que no se describen específicamente en el presente documento.

La divulgación planteada en el presente documento también abarca la materia objeto establecida en las siguientes cláusulas:

Cláusula 1. Un procesador de red neuronal, que comprende: una o más neuronas; y una unidad de compresión configurada para recibir un fragmento de datos descomprimido generado por al menos una de las neuronas en el procesador de red neuronal, el fragmento de datos descomprimido comprende una cantidad fija de bytes; generar una porción de máscara de un fragmento de salida comprimido, la porción de máscara comprende una cantidad de bits equivalente a la cantidad fija de bytes en el fragmento de datos descomprimido, cada bit en la porción de máscara corresponde a un byte en el fragmento de datos descomprimido, y donde cada bit en la porción de máscara se establece en cero lógico donde un byte correspondiente en el fragmento de datos descomprimido es cero y se establece en uno lógico donde un byte correspondiente en el fragmento de datos descomprimido no es cero; generar una porción de datos del fragmento de salida comprimido al determinar una cantidad de bytes distintos de cero en el fragmento de datos descomprimido, al determinar, con base en el número de bytes distintos de cero en el fragmento de datos descomprimido, una cantidad de bits en la porción de datos del fragmento de salida comprimido disponible para almacenar bytes distintos de cero truncados del fragmento de datos descomprimido, al truncar los bytes distintos de cero en el fragmento de datos descomprimido a la cantidad determinada de bits y al almacenar los bytes distintos de cero truncados en la porción de datos del fragmento de salida comprimido; y generar el fragmento de salida comprimido, el fragmento de salida comprimido comprende la porción de máscara y la porción de datos.

Cláusula 2. El procesador de red neuronal de la cláusula 1, donde el procesador de red neuronal comprende, además, una unidad de descompresión configurada para: recibir el fragmento de salida comprimido; determinar la cantidad de bytes distintos de cero en la porción de datos del fragmento de datos descomprimido con base en la porción de máscara del fragmento de salida comprimido; determinar las ubicaciones de los bytes distintos de cero en el fragmento de datos descomprimido con base en la porción de máscara del fragmento de salida comprimido; determinar la cantidad de bits utilizados por la unidad de compresión para almacenar los bytes distintos de cero truncados en la porción de datos del fragmento de salida comprimido; para cada posición en la porción de máscara del fragmento de salida comprimido que es un cero lógico, insertar un byte cero en la posición correspondiente de un fragmento de salida descomprimido; y para cada posición en la porción de máscara que es un uno lógico, insertar el byte distinto de cero truncado desde la posición correspondiente del fragmento de entrada comprimido en una posición correspondiente del fragmento de salida descomprimido y una cantidad de bits cero equivalente a la cantidad de bits truncados durante la compresión del fragmento de salida comprimido.

Cláusula 3. El procesador de red neuronal de cualquiera de las cláusulas 1 y 2, donde la unidad de compresión está configurada, además, para: determinar una cantidad de bits adicionales en la porción de datos del fragmento de salida comprimido disponible para almacenar bytes truncados distintos de cero del fragmento de datos descomprimido; y asignar los bits adicionales a uno o más bytes distintos de cero en el fragmento de datos descomprimido antes de truncar uno o más bytes distintos de cero.

Cláusula 4. El procesador de red neuronal de cualquiera de las cláusulas 1 a 3, donde la unidad de descompresión está configurada, además, para determinar la cantidad de bits adicionales asignados a uno o más de los bytes distintos de cero almacenados en la porción de datos del fragmento de salida comprimido.

Cláusula 5. El procesador de red neuronal de cualquiera de las cláusulas 1 a 4, donde la unidad de descompresión está configurada, además, para agregar un desplazamiento a uno o más de los bytes distintos de cero truncados almacenados en el fragmento de salida descomprimido.

Cláusula 6. El procesador de red neuronal de cualquiera de las cláusulas 1 a 5, donde se truncan uno o más bits menos significativos (LSB) de los bytes distintos de cero.

Cláusula 7. El procesador de red neuronal de cualquiera de las cláusulas 1 a 6, donde se truncan uno o más bits más significativos (MSB) de los bytes distintos de cero.

Cláusula 8. Un procesador de red neuronal, que comprende: una o más neuronas; y una unidad de descompresión configurada para recibir una porción comprimida de datos que comprende una porción de máscara y un fragmento de datos; determinar una cantidad de bytes distintos de cero en un fragmento de datos descomprimido con base en los bits en la porción de máscara; determinar, con base, al menos en parte, en la cantidad de bytes distintos de cero, una cantidad de bits utilizados para almacenar bytes distintos de cero truncados en la porción de datos del fragmento de datos de salida comprimido; para cada posición de bit en la porción de máscara del fragmento de datos comprimido que es un cero lógico, insertar un byte cero en la posición correspondiente del fragmento de datos descomprimido; y para cada posición en la porción de máscara del fragmento de datos comprimido que es un uno lógico, insertar un byte distinto de cero truncado desde la posición correspondiente en la porción de datos del fragmento de datos comprimido en una posición correspondiente en el fragmento de datos descomprimido y una cantidad de bits cero equivalente a una cantidad de bits truncados durante la compresión del fragmento de datos comprimido.

Cláusula 9. El procesador de red neuronal de la cláusula 8, que comprende, además, una unidad de compresión configurada para: recibir un fragmento de datos descomprimido generado por al menos una de las neuronas en el procesador de red neuronal, el fragmento de datos descomprimido comprende una cantidad fija de bytes; generar la porción de máscara del fragmento de datos comprimido, la porción de máscara comprende una cantidad de bits equivalente a la cantidad fija de bytes en el fragmento de datos descomprimido, cada bit en la porción de máscara corresponde a un byte en el fragmento de datos descomprimido, y donde cada bit en la porción de máscara comprende un cero lógico donde un byte correspondiente en el fragmento de datos descomprimido es cero y comprende un uno lógico donde un byte correspondiente en el fragmento de datos descomprimido no es cero; generar la porción de datos del

fragmento de datos comprimido determinando una cantidad de bytes distintos de cero en el fragmento de datos descomprimido, al determinar, con base en la cantidad de bytes distintos de cero en el fragmento de datos descomprimido, una cantidad de bits en los datos porción de la porción comprimida de datos disponible para almacenar bytes distintos de cero truncados de la porción de datos descomprimida, al truncar los bytes distintos de cero en el fragmento de datos descomprimido a la cantidad determinada de bits y al almacenar los bytes distintos de cero truncados en la porción de datos del fragmento de datos comprimido; y generar el fragmento de datos comprimido, el fragmento de datos comprimido comprende la porción de máscara y la porción de datos.

Cláusula 10. El procesador de red neuronal de cualquiera de las cláusulas 8 y 9, donde la unidad de compresión se configura, además, para almacenar los bytes distintos de cero en el fragmento de datos descomprimido en la porción de datos del fragmento de datos comprimido sin truncamiento si la cantidad de bytes cero en el fragmento de datos descomprimido es menor o igual que una cantidad de bytes en la porción de datos del fragmento de datos comprimido.

Cláusula 11. El procesador de red neuronal de cualquiera de las cláusulas 8 a 10, en donde la unidad de compresión está configurada, además, para: determinar una cantidad de bits adicionales en la porción de datos del fragmento de salida comprimido disponible para almacenar bytes distintos de cero truncados del fragmento de datos descomprimido; y asignar los bits adicionales a uno o más bytes distintos de cero en el fragmento de datos descomprimido antes de truncar uno o más bytes distintos de cero.

Cláusula 12. El procesador de red neuronal de cualquiera de las cláusulas 8 a 11, donde la unidad de descompresión está configurada, además, para determinar la cantidad de bits adicionales asignados a uno o más de los bytes distintos de cero almacenados en la porción de datos del fragmento de salida comprimido.

Cláusula 13. El procesador de red neuronal de cualquiera de las cláusulas 8 a 12, donde uno o más bits menos significativos (LSB) de los bytes distintos de cero se truncan durante la compresión del fragmento de datos comprimido.

Cláusula 14. El procesador de red neuronal de cualquiera de las cláusulas 8 a 13, donde uno o más bits más significativos (MSB) de los bytes distintos de cero se truncan durante la compresión del fragmento de datos comprimido.

Cláusula 15. Un método implementado por computadora, que comprende: recibir, en una unidad de compresión de un procesador de red neuronal, un fragmento de datos descomprimido generado por al menos una neurona en el procesador de red neuronal, el fragmento de datos descomprimido comprende una cantidad fija de bytes; generar una porción de máscara de un fragmento de salida comprimido, la porción de máscara comprende una cantidad de bits equivalente a la cantidad fija de bytes en el fragmento de datos descomprimido, cada bit en la porción de máscara corresponde a un byte en el fragmento de datos descomprimido, y donde cada bit en la porción de máscara comprende un cero lógico donde un byte correspondiente en el fragmento de datos descomprimido es cero y comprende un uno lógico donde un byte correspondiente en el fragmento de datos descomprimido no es cero; generar una porción de datos del fragmento de salida comprimido al determinar una cantidad de bytes distintos de cero en el fragmento

de datos descomprimido, al determinar, con base en la cantidad de bytes distintos de cero en el fragmento de datos descomprimido, una cantidad de bits en la porción de datos del fragmento de salida comprimido disponible para almacenar bytes distintos de cero truncados del fragmento de datos descomprimido, al truncar los bytes distintos de cero en el fragmento de datos descomprimido a la cantidad determinada de bits y al almacenar los bytes distintos de cero truncados en la porción de datos del fragmento de salida comprimido; y almacenar el fragmento de salida comprimido en una memoria del procesador de red neuronal, el fragmento de salida comprimido comprende la porción de máscara y la porción de datos.

Cláusula 16. El método implementado por computadora de la cláusula 15, que comprende además: determinar una cantidad de bits adicionales en la porción de datos del fragmento de salida comprimido disponible para almacenar bytes distintos de cero truncados del fragmento de datos descomprimido; y asignar los bits adicionales a uno o más de los bytes distintos de cero en el fragmento de datos descomprimido antes de truncar uno o más de los bytes distintos de cero.

Cláusula 17. El método implementado por computadora de cualquiera de las cláusulas 15 y 16, que comprende, además, almacenar los bytes distintos de cero en el fragmento de datos descomprimido en la porción de datos del fragmento de datos comprimido sin truncamiento si la cantidad de bytes distintos de cero en el fragmento de datos descomprimido es menor o igual a una cantidad de bytes en la porción de datos del fragmento de datos comprimido.

Cláusula 18. El método implementado por computadora de cualquiera de las cláusulas 15 a 17, que comprende, además: recibir, en una unidad de descompresión de un procesador de red neuronal, el fragmento de salida comprimido; determinar la cantidad de bytes distintos de cero en la porción de datos del fragmento de datos descomprimido con base en la porción de máscara del fragmento de salida comprimido; determinar las ubicaciones de los bytes distintos de cero en el fragmento de datos descomprimido con base en la porción de máscara del fragmento de salida comprimido; determinar la cantidad de bits utilizados por la unidad de compresión para almacenar los bytes distintos de cero truncados en la porción de datos del fragmento de salida comprimido; para cada posición de bit en la porción de máscara del fragmento de salida comprimido que es un cero lógico, insertar un byte cero en la posición correspondiente de un fragmento de salida descomprimido; y para cada posición en la porción de máscara del fragmento de salida comprimido que es un uno lógico, insertar el byte distinto de cero truncado desde la posición correspondiente del fragmento de salida comprimido en una posición correspondiente del fragmento de salida descomprimido y una cantidad de bits cero equivalente a la cantidad de bits truncados durante la compresión del fragmento de salida comprimido.

Cláusula 19. El método implementado por computadora de cualquiera de las cláusulas 15 a 18, que comprende, además, agregar un desplazamiento a uno o más de los bytes distintos de cero truncados almacenados en el fragmento de salida descomprimido.

Cláusula 20. El método implementado por computadora de cualquiera de las cláusulas 15 a 19, donde el desplazamiento se agrega a uno o más bits menos significativos (LSB) de los bytes distintos de cero truncados almacenados en el fragmento de salida descomprimido.

Con base en lo anterior, debe apreciarse que un módulo de DNN que puede comprimir y

descomprimir los datos de activación para reducir la utilización del ancho de banda del bus de memoria se divulgó en el presente documento. Aunque la materia objeto planteada en el presente documento se describió en un lenguaje específico para las características estructurales de cómputo, las acciones metodológicas y transformadoras, la maquinaria de cómputo específica y los medios legibles por computadora, se entenderá que la materia objeto expuesta en las reivindicaciones adjuntas no está necesariamente limitada a las características, acciones o medios específicos descritos en el presente documento. Por el contrario, las características, acciones y medios específicos se divulgan como formas ejemplares de implementación de la materia objeto reclamada.

La materia objeto descrita anteriormente se proporciona solo a modo de ilustración y no debe interpretarse como limitante. Pueden realizar varias modificaciones y cambios en la materia objeto descrita en el presente documento sin seguir las configuraciones y aplicaciones ejemplares ilustradas y descritas, y sin apartarse del alcance de la presente divulgación, el cual se establece en las siguientes reivindicaciones.