(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 558 952**

(21) Número de solicitud: 201500841

(51) Int. CI.:

G06N 3/00 (2006.01)

(12) PATENTE DE INVENCIÓN CON EXAMEN PREVIO B2

(22) Fecha de presentación:

20.11.2015

(43) Fecha de publicación de la solicitud:

09.02.2016

Fecha de la concesión:

23.06.2016

(45) Fecha de publicación de la concesión:

30.06.2016

(73) Titular/es:

UNIVERSIDAD DE CANTABRIA (100.0%)
Pabellón de Gobierno, Avda. de los Castros s/n
39005 Santander (Cantabria) ES

(72) Inventor/es:

GREGORIO MONASTERIO, José Ángel y
PUENTE VARONA, Valentín

(54) Título: **Sistema y método escalable de aceleración por hardware para almacenar y recuperar información**

(57) Resumen:

La presente invención se refiere a un método y un sistema de aceleración por hardware para almacenar y recuperar información, que implementa un algoritmo de aprendizaje cortical a través de una red de conmutación de paquetes. El sistema comprende: un módulo codificador para proveer una entrada SDR y enviar paquetes multidifusión a ciertos módulos columnados conectados entre sí mediante la red de conmutación de paquetes; donde los módulos columnados comprenden a su vez: un encaminador, una pluralidad de módulos de memoria configurados para almacenar las entradas recibidas desde el encaminador y almacenar información de contexto; y un módulo de cálculo que calcula el solapamiento de las entradas, selecciona los módulos de memoria con mayor solapamiento, determina un contexto temporal para los módulos de memoria seleccionados y envía una predicción de salida del sistema a un módulo clasificador, el cual selecciona una salida del sistema entre un grupo de salidas preestablecidas, en función de dicha predicción.

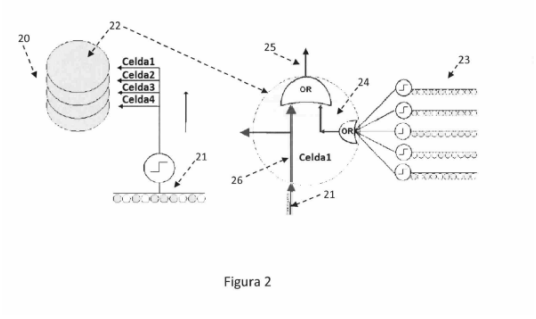

Figura 2

ES 2 558 952 B2

SISTEMA Y MÉTODO ESCALABLE DE ACELERACIÓN POR HARDWARE PARA ALMACENAR Y RECUPERAR INFORMACIÓN

DESCRIPCIÓN

CAMPO TÉCNICO DE LA INVENCIÓN

La presente invención se refiere al campo técnico de la inteligencia artificial y más concretamente a las redes neuronales artificiales implementadas en hardware para almacenar y recuperar información.

ANTECEDENTES

La aplicación de algoritmos basados en redes neuronales consiste en el procesamiento automático de información inspirado en el modo en que funciona el sistema nervioso de los animales, las neuronas y sus conexiones. Las neuronas pueden distinguirse agrupadas en *columnas,* las cuales están conectadas a través de los axones de bajo-rango a otras columnas cercanas (a través de la capa I del cortex) o a otras columnas distantes y a la interface sensor-motor, es decir, el tálamo (a través de la capa VI). La figura 1 representa estas estructuras de columnas micro-corticales (1) y las hiper-columnas corticales (2). Independientemente de la funcionalidad de cada zona, el cortex es morfológicamente muy regular.

La evidencia empírica sugiere que el sistema neuronal representa la información siguiendo una representación distribuida dispersa SDR para almacenar y recuperar información. En esta representación, en contraste con la representación convencional de datos binarios (también acuñado como representación localista), cada bit tiene significado semántico y la representación de datos es altamente resistente al ambiente ruidoso y propenso a los fallos (como es el biológico), es decir, el cambio indeseado de un numero bajo de bits en la representación original siempre produce un valor similar a la original.

El cortex puede entenderse que funciona como una memoria auto-asociativa, jerárquicamente estructurada como una memoria temporal jerárquica (HTM). Esta afirmación, basada puramente en las observaciones de la neurociencia, presenta un algoritmo preciso, llamado algoritmo de aprendizaje cortical (CLA), que proporciona las

reglas para almacenar y recuperar información, es decir, aprender y hacer predicciones. Este concepto ha sido utilizado en problemas prácticos, tales como la detección de anomalías, predicción de secuencias, identificación de patrones, etc. imitando el comportamiento de capas superiores de la columna cortical.

El algoritmo CLA se centra en replicar parcialmente la funcionalidad de las micro-columnas corticales, donde la capa I se utiliza principalmente para la interconexión de las diferentes columnas en la misma hiper-columna; el nivel II/III, denominado generalmente como la capa de inferencia, está supuestamente dedicado a predecir el estado de la columna en los próximos pasos de la entrada; y la capa IV, denominada capa sensorial, se ocupa de las señales de entrada a la columna. Los principios de funcionamiento de las capas V y VI aún no se comprenden bien y actualmente CLA no los modela, pero el punto clave de esta organización es que la misma hiper-columna puede ser reutilizada por diferentes hiper-columnas en el siguiente nivel y a través de toda la jerarquía, el nivel de información que una columna puede identificar será cada vez mayor (condensando la semántica de los niveles más bajos).

El algoritmo CLA define el término *columna (20)*, representado en la figura 2, lo que es suficiente para manejar la predicción sin la estructura jerárquica. En la parte inferior, un segmento dendrítico proximal (21) podría estar conectado a un subconjunto de los bits de la entrada del SDR. Esta restricción modela el hecho de que la actividad del axón de entrada será observada por un subconjunto columnas. Dichos segmentos modelan el crecimiento dendrítico de la conexión de alimentación directa del sistema, lo que es bien conocido que es responsable del aprendizaje en el cortex. En contraste con otras redes neuronales artificiales, cada sinapsis del segmento se caracteriza por un valor binario, es decir, está conectado o no. Para una entrada codificada dada, en cada segmento proximal de dendritas se determina el número de sinapsis activas, es decir, el número de entradas activas conectadas al segmento con una sinapsis conectada (esto se llama solapamiento de la entrada). Una vez que esto se sabe, al igual que en los sistemas biológicos, comienza un proceso de inhibición y únicamente se seleccionan aproximadamente el 2% de las mejores de las columnas con más sinapsis activas. Las columnas restantes son inhibidas. Las sinapsis, que han sido activadas por la entrada en las columnas ganadoras, se fortalecen y las sinapsis conectadas a las entradas inactivas se debilitan. Con el fin de manejar el aprendizaje, para cada conexión sináptica se realiza un seguimiento con un valor de permanencia. Si el valor está por encima de un umbral predefinido, la sinapsis se considera conectada. En el momento del arranque,

los valores se eligen al azar, cerca del valor umbral. Típicamente, tres o cuatro bits pueden ser suficientes. En las implementaciones software, por defecto, el umbral puede ser 0,2, máximo 1,0 y aprender con incremento de 0,1 (la amortiguación u *olvido* es, por lo general, un orden de magnitud más pequeña, pero se puede evitar el aumento de resolución mediante el uso de una resta al azar). De esta manera se emula la Capa IV de las micro-columnas corticales que, en la terminología CLA/HTM, se llama agrupación espacial (spatial pooling). La intuición detrás de esta agrupación es la de "filtrar" las características más notables de la entrada con el fin de almacenar posteriormente la secuencia.

Por otro lado, cuando se activa una columna, es decir gana el proceso de inhibición, las celdas (temporales) tienen que procesar dicha información. Cada columna tendrá unas pocas decenas de celdas (22). Un número de columnas SDR-compatible representará una entrada codificada. Por lo tanto, después de la inhibición, las columnas ganadoras representan las características más sobresalientes de la entrada. En un entorno de "sin-contexto", una sola celda sería suficiente para hacer la predicción. Sin embargo, para obtener una predicción que dependa del contexto, se necesita tanto un valor actual (26) como un contexto de secuencia temporal. Para ello, cada celda por columna representa el valor de la entrada en una secuencia temporal (es decir, la memoria debe ser capaz de predecir las secuencias sucesivas). Incluso con un bajo número de celdas por columna, el número de "contextos" que el sistema puede almacenar para el mismo valor, es enorme. Por ejemplo, en un sistema con 2048 columnas y 32 celdas por columna, será capaz de capturar 40^{32} contextos temporales diferentes para la misma entrada.

Cada celda podría predecir el estado de la columna en la siguiente entrada en el secuenciador. Para ello, utiliza segmentos dendríticos para el modelado de las relaciones columna. Cada segmento dendrítico distal (23) almacena potenciales sinapsis con otras columnas del cortex. Las reglas para manejar tales sinapsis son similares a las del segmento proximal. Si alguno de los segmentos de la celda alcanza un umbral dado, ésta entra en el estado predictivo (24), lo que significa que dicha columna se activará (25) en el siguiente periodo o época. Cuando una columna no se predijo correctamente, todas las celdas de la columna intentan conectarse con la secuencia vista previamente. En primer lugar, se construyen, sobre la marcha, nuevos segmentos distales según las activaciones remotas previas y, en segundo lugar, se buscan celdas que deben predecir la activación en el siguiente periodo, lo que imita las capas II/III en las columnas biológicas. La intuición es utilizar la sinapsis entre las diferentes columnas en el sistema para obtener un camino serpenteante entre celdas

que representen los diferentes contextos temporales. La terminología CLA/HTM utilizada para esta tarea es la agrupación temporal (temporal pooling).

Hoy en día, los avances realizados en HTM se implementan en software, lo que limita técnicamente los sistemas a unos pocos miles de columnas. En lugar de conexiones precisas ponderadas, HTM utiliza una topología dinámica compleja para almacenar y recuperar información, lo que, desde la perspectiva hardware simplista, no es posible (una sola columna puede estar potencialmente relacionada con decenas de miles de diferentes columnas). Las soluciones existentes son muy demandantes en memoria y requieren millones de ciclos de reloj para producir cada predicción. Problemas como el reconocimiento de patrones basado en el mecanismo sacádico requieren sistemas mucho más grandes y rápidos y aunque se están haciendo esfuerzos en enfoques basados en FPGA o tecnologías emergentes como apilamiento 3D y memorias no volátiles que podrían aliviar de alguna manera estos estrictos requisitos, el estado del arte recibiría como una valiosa contribución cualquier solución que presentase una implementación hardware factible para superar ese problema y redujese los costes y tiempo de ejecución.

DESCRIPCIÓN DE LA INVENCIÓN

La presente invención resuelve los problemas mencionados anteriormente, presentando la arquitectura de una implementación hardware que emplea técnicas y metodologías arquitecturales tales como chips o multiprocesadores de propósito general. Específicamente, las limitaciones que implican las implementaciones software conocidas en el estado del arte, para el algoritmo de aprendizaje cortical CLA, son superadas por la presente invención, la cual se refiere en un primer aspecto a un sistema de aceleración por hardware para almacenar y recuperar información, que implementa dicho algoritmo de aprendizaje cortical a través de una red de conmutación de paquetes. El sistema comprende:

- **al menos un módulo codificador** configurado para codificar una entrada binaria en una representación distribuida dispersa (SDR), y para enviar, por cada bit activo de la SDR, un paquete multidifusión a un módulo columnado determinado a través de la red de conmutación de paquetes, en función de una tabla de correspondencias previamente establecidas;
- **una pluralidad de módulos columnados** conectados mediante dicha red de conmutación de paquetes, configurados para recibir los paquetes multidifusión

enviados desde el codificador, donde cada uno de los módulos columnados comprende a su vez:

- **un encaminador** con soporte multidifusión configurado para recibir paquetes desde el módulo codificador, entregar dichos paquetes a ciertos módulos de memoria del módulo columnado y enviar paquetes desde los módulos de memoria a un clasificador de salida;
- **una pluralidad de módulos de memoria** configurados para almacenar las entradas recibidas desde el encaminador y almacenar información de contexto;
- **un módulo de cálculo** configurado para determinar un grado de solapamiento entre el contenido de los ciertos módulos de memoria y la entrada actual, seleccionar un número determinado de módulos de memoria con mayor grado de solapamiento, determinar un contexto temporal para cada uno de los módulos de memoria seleccionados, realizar una predicción de la salida del sistema en función de la entrada actual y la información de contexto temporal y enviar un paquete de salida que contiene dicha predicción a un módulo clasificador de salida;

- **un módulo clasificador** de salida configurado para recibir un paquete de salida, enviado a través de la red de conmutación desde cualquiera de los módulos columnados, y para seleccionar una salida del sistema entre un grupo de salidas preestablecidas en función del paquete de salida recibido.

El sistema de la presente invención, de acuerdo a una de sus realizaciones particulares, contempla que el módulo de cálculo comprenda un comparador, un sumador y un contador.

La presente invención contempla, en una de sus realizaciones, que cada módulo de memoria de la pluralidad de módulos de memoria comprenda una pluralidad de celdas temporales, las cuales adoptan un estado activo o un estado no activo y su combinación representa un determinado contexto temporal para el módulo de memoria. Ventajosamente se consigue así la representación de diferentes contextos temporales que permiten predecir las siguientes entradas y, además, las secuencias que puede almacenar el sistema contribuyen directamente al aprendizaje para futuras entradas.

Adicionalmente, la presente invención contempla que el módulo de cálculo esté configurado para comprobar si su predicción de salida es correcta; en caso de predicción errónea se produce una ráfaga que pone todas las celdas temporales del módulo de memoria en estado activo. Así, se afina ventajosamente el aprendizaje del sistema.

Opcionalmente, la presente invención, de acuerdo a una de sus realizaciones, contempla que el módulo de cálculo esté además configurado para simultanear etapas y, dada una secuencia de entrada, producir una predicción en tres intervalos de dicha secuencia. Ventajosamente se aprovechan así las capacidades de la red y se puede segmentar el algoritmo CLA para inyectar los resultados de cada etapa en la red sin necesidad de esperar a terminar todas las fases de cálculo.

Adicionalmente, una de las realizaciones de la presente invención, contempla la posibilidad de que el módulo de cálculo esté además configurado para agregar tráfico de diferentes etapas en un mismo paquete. Es una medida más de optimización que puede incorporar la presente invención para potenciar las ventajas de la segmentación del algoritmo comentada anteriormente.

Los módulos columnados ubicados en los extremos de la red, de acuerdo a una de las realizaciones de la invención, se contempla que estén configurados para inyectar en la red de conmutación de paquetes un paquete escoba, el cual se replica en el resto de módulos columnados únicamente cuando el encaminador correspondiente no tiene más paquetes en cola hasta que dicho paquete escoba alcanza el extremo opuesto de la red, lo que indica que la red ha sido vaciada. Esto ventajosamente sirve de mecanismo para garantizar la correcta ejecución de las etapas de cálculo de solapamiento y determinar el contexto temporal.

Una de las realizaciones particulares de la invención contempla la posibilidad de que el número de módulos de memoria que comprende cada uno de los módulos columnados esté determinado por un equilibrio entre el retardo de propagación y el ciclo de reloj del sistema.

El módulo codificador de la presente invención, o uno de los módulos codificadores, puede configurarse para enviar los paquetes de entrada a una selección de módulos columnados preestablecida aleatoriamente que representa en torno al 20% del total de módulos columnados. Así ventajosamente se proporciona otra de las optimizaciones de la presente invención donde, en función de las entradas o de las aplicaciones concretas, podría variarse dinámicamente el tamaño de la selección, o parche proximal.

El sistema propuesto por la presente invención se implementa, de acuerdo a diferentes realizaciones particulares, en una placa de silicio, un chip o un microprocesador utilizando tecnología CMOS.

Un segundo aspecto de la invención se refiere a un método escalable de aceleración por hardware para almacenar y recuperar información a través de una red de conmutación de paquetes, el método comprende los pasos de:

a) codificar, en un módulo codificador, una entrada binaria en una representación distribuida dispersa (SDR)

b) enviar, por cada bit activo de la SDR, un paquete multicast desde el módulo codificador a un módulo columnado determinado de una pluralidad de módulos columnados a través de la red de conmutación de paquetes, en función de una tabla de correspondencias previamente establecidas;

c) recibir los paquetes enviados desde el módulo codificador, a través de la red de conmutación de paquetes, en un encaminador del módulo columnado;

d) entregar dichos paquetes a ciertos módulos de memoria del módulo columnado;

e) almacenar en los ciertos módulos de memoria los paquetes recibidos;

f) determinar, en un módulo de cálculo del módulo columnado, un grado de solapamiento entre el contenido de los módulos de memoria que han recibido el paquete de entrada y la entrada actual;

g) seleccionar, por el módulo de cálculo, un número determinado de módulos de memoria con mayor grado de solapamiento;

h) determinar, por el módulo de cálculo, un contexto temporal para cada uno de los módulos de memoria seleccionados;

i) realizar, por el módulo de cálculo, una predicción de la salida del sistema en función de la entrada actual y la información de contexto temporal almacenada en los módulos de memoria;

j) enviar un paquete de salida que contiene dicha predicción a un módulo clasificador de salida;

k) recibir un paquete de salida en el clasificador de salida, enviado a través de la red de conmutación desde cualquiera de los módulos columnados;

l) seleccionar, en el clasificador de salida, una salida del sistema entre un grupo de salidas preestablecidas en función del paquete de salida recibido.

De acuerdo a una de las realizaciones de la presente invención, el método propuesto contempla comprobar si la predicción de salida realizada por el módulo de cálculo es correcta, donde, en caso de predicción errónea se produce una ráfaga que pone todas las celdas temporales del módulo de memoria en estado activo.

Adicionalmente, la presente invención puede incluir el paso de comprobar que la red de conmutación de paquetes está vacía antes de ejecutar las etapas de calcular el solapamiento y determinar el contexto temporal, donde, para comprobar que la red está vacía, se proporciona un paquete escoba que recorre la red de conmutación de paquetes.

De forma opcional, la presente invención contempla en una de sus realizaciones el paso de restringir los paquetes enviados por el módulo codificador a una selección de módulos columnados, preestablecida aleatoriamente, que representa en torno al 20% del total de módulos columnados.

Inspirándose en las propiedades biológicas de axones y dendritas, la presente invención define, por tanto, un sistema que utiliza una construcción lógica para satisfacer la flexibilidad topológica del conocido algoritmo CLA través de una red on-chip. A diferencia de otros sistemas de aprendizaje del estado del arte, los cálculos del algoritmo CLA son simples (sumas y restas de baja precisión, comparaciones simples), por lo que, añadiendo alguna lógica de cálculo a los encaminadores de dicha red y algunos módulos de memoria para almacenar el estado de conectividad, la presente invención implementa el algoritmo CLA sin necesidad de procesadores de propósito general complejos, donde el substrato de comunicación, y los procedimientos para conseguir una implementación hardware factible del algoritmo CLA conocido, se basan en el uso de una red de conmutación de paquetes y diversas técnicas empleadas en arquitectura de computadores, que garantizan así mismo la escalabilidad del sistema. La combinación de todas las técnicas presentadas en la presente invención permite reducir, en promedio, aproximadamente un 95% el retardo de la red y energía necesaria.

Además, la implementación hardware propuesta por la presente invención implica multitud de ventajas adicionales como ampliar el espectro de aplicación del algoritmo permitiendo, por ejemplo, combinarlo fácilmente con computación tipo von-Neumann, utilizarlo como acelerador de procesamiento neuronal, permitir explorar el potencial de la organización jerárquica, o investigar sobre los mecanismos subyacentes y

desconocidos del neo-cortex. Por ello, una implementación basada en silicio como la propuesta en la presente invención supone una valiosa contribución al estado del arte.

DESCRIPCIÓN DE LOS DIBUJOS

Para complementar la descripción que se está realizando y con objeto de ayudar a una mejor comprensión de las características de la invención, se acompaña, como parte integrante de dicha descripción, un juego de figuras en donde, con carácter ilustrativo y no limitativo, se ha representado lo siguiente:

Figuras 1a, 1b.- representan unas estructuras de columnas micro-corticales (1a) y estructuras de hiper-columnas corticales (1b) en las que se basa la presente invención.

Figura 2.- representa una de las columnas según el algoritmo CLA.

Figura 3a.- representa una descripción de alto nivel de la arquitectura propuesta por una de las realizaciones de la presente invención.

Figura 3b.- representa un bosquejo en alto nivel de uno de los módulos columnados de la figura 3a.

Figura 4.- representa las etapas necesarias para el algoritmo CLA

Figura 5.- representa la segmentación del algoritmo CLA y cómo las etapas son simultaneadas.

Figura 6.- representa un ejemplo de optimización de acuerdo a una de las realizaciones de la invención, donde se muestra un parche proximal en una topología de tipo panal.

Figura 7.- representa un ejemplo de optimización de acuerdo a una de las realizaciones de la invención, donde se muestran varias zonas de scale-out.

Figura 8.- representa gráficamente el número de ciclos de reloj, por intervalo, para diferentes tamaños de malla cuadrada 2D.

Figura 9.- representa gráficamente el número de ciclos de reloj, por intervalo, para diferentes mallas cuadradas 2D, empleando tráfico agregado.

La **figura 10.-** representa gráficamente el número de ciclos de reloj, por intervalo, para diferentes mallas cuadradas 2D, con el algoritmo segmentado, agregación de tráfico y parches proximales aplicados.

La **figura 11.**- representa gráficamente el número de ciclos de reloj, variando la anchura del enlace.

La **figura 12.**- representa gráficamente los ciclos de reloj requeridos por la red para procesar un intervalo con diferentes anchuras del enlace.

La **figura 13.**- representa gráficamente los requerimientos de energía dinámica de la red para procesar un intervalo desde el flujo de entrada.

La **figura 14.**- representa gráficamente los ciclos, normalizados al algoritmo base, por intervalo de entrada (malla 16x16).

La **figura 15.**- representa gráficamente la energía dinámica de la red por intervalo, normalizada al algoritmo base.

La **figura 16.**- representa gráficamente la probabilidad de columnas mal predichas, normalizada al algoritmo base.

DESCRIPCIÓN DETALLADA DE LA INVENCIÓN

Lo definido en esta descripción detallada se proporciona para ayudar a una comprensión exhaustiva de la invención. En consecuencia, las personas medianamente expertas en la técnica reconocerán que son posibles variaciones, cambios y modificaciones de las realizaciones descritas en la presente memoria sin apartarse del ámbito de la invención. Además, la descripción de funciones y elementos bien conocidos en el estado del arte se omite por claridad y concisión.

Por supuesto, las realizaciones de la invención pueden ser implementadas en una amplia variedad de plataformas arquitectónicas, protocolos, dispositivos y sistemas, por lo que los diseños e implementaciones específicas, presentadas en este documento, se proporcionan únicamente con fines de ilustración y comprensión, y nunca para limitar aspectos de la invención.

La presente invención divulga la implementación de un acelerador hardware basado en el algoritmo de aprendizaje cortical para almacenar y recuperar la información, donde los detalles, desde la perspectiva de la arquitectura de computadores, se ofrecen a continuación.

La suposición básica de las memorias HTM y algoritmos CLA es que la plasticidad sináptica (a través del crecimiento dendrítico) es el elemento clave del cortex para realizar el aprendizaje. Esto supone que la información se almacena en la relación entre las columnas, definida dinámicamente mediante las conexiones establecidas durante el aprendizaje. Por lo tanto, la capacidad de almacenamiento es proporcional al producto del número de columnas por el número máximo de conexiones por columna.

Aunque la conectividad de las neuronas puede ser potencialmente muy alta (las estrías dendríticas pueden proporcionar hasta decenas de miles de sinapsis potenciales), muchas de estas sinapsis no están activas (es decir, el axón pre-sináptico está demasiado distante de la dendrita) o múltiples sinapsis activas corresponden al mismo par de las neuronas (como mecanismo de redundancia), por lo que, en lugar de replicar eléctricamente la morfología de los sistemas biológicos, que actualmente sería imposible, la presente invención introduce tal funcionalidad en una red de conmutación de paquetes.

Principalmente, es el substrato de comunicación el objeto de organización y optimización para emular la actividad del axón y aplicar correctamente los algoritmos de predicción y aprendizaje del HTM/CLA. La presente invención, en lugar de utilizar las sinapsis para establecer una conexión activa entre dos columnas, recurre a estructuras de memoria asociadas a una pluralidad de encaminadores (routers) para modelar dichos segmentos dendríticos y empleando una lógica de cálculo simple para realizar la tareas de agrupación espacial (spatial pooling) y agrupación temporal (temporal pooling).

La **Figura 3a** presenta una descripción de alto nivel de la arquitectura propuesta, donde puede identificarse un codificador (31) en la entrada del sistema, encargado de convertir una entrada localista en una representación SDR; y un clasificador (32) a la salida del sistema, encargado de llevar a cabo la finalidad prevista, por ejemplo detectar anomalías en una secuencia de entrada, predecir el próximo valor en la secuencia de entrada, comparar determinados patrones etc. Entre los módulos codificador y clasificador, la mecánica CLA se implementa mediante un componente al que, en este documento, se hará referencia como Columnar Core (CC) o módulos columnados (33). En una de las realizaciones particulares ilustrada por la Figura 3a, con fines únicamente explicativos, se recurre a un sistema de 16 módulos columnados, CC0-CC15, conectados mediante una red de conmutación de paquetes, por ejemplo con una topología en forma de malla cuadrada, pero configuraciones de otras dimensiones serían igualmente posibles aprovechando una de las mayores ventajas de la presente invención, su escalabilidad.

La **Figura 3b** representa un bosquejo en alto nivel de un CC. En este caso particular, se supone que cada CC tiene *B* columnas y *t* celdas temporales por columna. El sistema es homogéneo, al igual que el cortex biológico, y la presente invención, para su implementación hardware, tiene en cuenta los siguientes requisitos para las tres diferentes secciones: comunicación (34), cálculo (35) y memoria (36):

A. Requerimientos de la Comunicación

La red de interconexión tiene que manejar todo el tráfico generado por el algoritmo CLA, es decir, el tráfico de entrada proveniente del codificador, el tráfico de inhibición y de la actividad lateral de las activaciones de las celdas temporales (38), así como el envío del patrón de activación al clasificador. Tal actividad se realiza, en la presente invención, a nivel lógico, empleando paquetes en lugar de cables físicos. Por ejemplo, de acuerdo a una de las realizaciones, cada bit de salida del codificador se conecta a un conjunto de columnas (37) estáticamente definido con lo que para una entrada dada, cada uno de los bits activos en la representación SDR enviará un paquete de multidifusión (multicast) al CC donde residen las columnas, o módulos de memoria, que tienen que recibirlos. El codificador cuenta, en una de las realizaciones, con una tabla que relaciona columnas y entradas. Por lo tanto, el paquete de multidifusión utilizado por la presente invención emula la actividad de cada axón. Del mismo modo, cuando una columna alcanza un estado predictivo, se enviará un único paquete al clasificador.

Internamente, el encaminador (39) recibirá entradas de la lógica de un módulo de cálculo (35) para la agrupación espacial "spatial pooler" (el solapamiento de columnas utilizado en el procedimiento de inhibición) y para la agrupación temporal "temporal pooler" (eventos de activación de celdas). Esas entradas deben ser enviadas a los potenciales receptores de paquetes. En los algoritmos CLA del estado del arte implementados en software se supone que, en la mayoría de los casos, todas las columnas del sistema deben ser conscientes de ello, es decir, los receptores potenciales son todas las columnas. Por ejemplo, en la inhibición global (que es el método por defecto), cualquier columna debe ser consciente del solapamiento con la entrada del resto de columnas. El solapamiento se calcula como el número de sinapsis conectadas en el segmento proximal de las columnas para una entrada dada. Con este tipo de información, la lógica de cálculo puede determinar si la columna actual se encuentra dentro del conjunto del 2% con un mayor solapamiento y alimentar la lógica de agrupación temporal. Del mismo modo, para la construcción de los segmentos distales, aunque probabilísticamente limitado, el algoritmo supone que cada columna es consciente de todas las celdas

(temporales) en estado predictivo, lo que equivale a suponer que los efectos de los axones se difunden a todas las celdas en el sistema.

Hay por tanto una gran cantidad de tráfico multidifusión que requiere un considerable ancho de banda de red y un notable consumo de energía. Además, para cualquiera de los cálculos realizados en la parte de cálculo sólo debe poder accederse a información local, por lo que no puede confiarse en ningún componente centralizado para ampliar el sistema a miles de CCs y por tanto, la sincronización del sistema supone un reto.

B. Requerimientos Computacionales

La mayor parte de la actividad del axón, tal y como ya se ha comentado, se modela de acuerdo a la presente invención como tráfico multidifusión. La lógica de cálculo en el destino será la encargada de realizar la predicción y los procedimientos de aprendizaje para cada paquete entrante.

Hay dos etapas en el algoritmo de CLA que tiene que ser aplicadas de forma secuencial, una vez que todo el tráfico del ciclo actual ha sido drenado fuera de la red:

- *La agrupación espacial,* donde el módulo de cálculo evaluará el solapamiento o superposición de la entrada con su segmento proximal (es decir, el número de sinapsis activas) y, suponiendo que la inhibición es global), difundirá su valor al resto de las columnas en el sistema. En la inhibición local se empleará un paquete de multidifusión. Cada columna es consciente de su propio solapamiento, con una simple comparación con el paquete entrante sabrá si está entre el 2% más activo, una vez que se drena todo el tráfico. Se actualizarán las sinapsis en la tabla de segmentos proximales de las entradas activas si la columna estaba activa. Por lo tanto, de acuerdo a una de las realizaciones de la invención, el módulo de cálculo comprende, para realizar estas operaciones de lógica espacial, un comparador, un sumador de 4 bits y un contador. Notar que el solapamiento máximo requiere ~ *Log2Input* bits. Para un codificador de 2048 entradas, son suficientes 12 bits.

- *La agrupación temporal,* donde el módulo de cálculo evaluará cualquier actividad lateral. Suponiendo que el axón de las celdas (temporales) es global, se generará una difusión. El paquete de entrada incluirá la columna y celda temporal origen. Esto se mantiene en una lista de activaciones actuales. Una vez finalizado el ciclo actual, la lógica determinará, para cada columna, si la activación se predijo correctamente. En tal caso, el segmento

distal correspondiente de la celda temporal en el estado predictivo se actualizará en consecuencia (esto emula el crecimiento de dendritas). Si la columna no fue predicha correctamente, la lógica debe mantener las activaciones del ciclo anterior para buscar el segmento distal más cercano (o crear uno nuevo si no existiese). Desde la perspectiva del hardware, esto requiere una extensa búsqueda a través de todos los segmentos dendríticos de la columna y determinar qué segmentos dendríticos de la columna están activos. Las celdas (temporales) con un segmento dendrítico activo generarán una difusión/multidifusión en la red y, finalmente, las columnas que no fueron correctamente predichas producirán un estallido o ráfaga (burst), como hace el sistema biológico, que equivale a poner todas las celdas temporales de la columna en estado activo (seleccionando solamente una para realizar el aprendizaje).

C. Requerimientos de memoria

Los segmentos proximales almacenan la permanencia de las sinapsis con cada bit de entrada potencialmente conectado. Debe tenerse en cuenta que cada bit de la representación SDR producida por el codificador está potencialmente conectado (es decir, se podría formar una sinapsis) al subconjunto elegido de columnas en el arranque (que puede ser seleccionado de manera uniforme). En general, puede suponerse que cada bit puede conectarse a cualquier columna en el sistema y, por lo tanto, el segmento proximal tiene que tener una entrada para cada potencial entrada, pero de acuerdo a la presente invención, cada columna se conectará (es decir, se formará una sinapsis) a un subconjunto muy pequeño de entradas de codificador, como ocurre en la práctica. Por lo tanto, el segmento proximal se estructura, de acuerdo a una de las realizaciones, como una memoria cache convencional indexada por el índice de entrada. En la práctica, una capacidad para 64-128 entradas parece ser suficiente para un sistema de 2K columnas. El valor de la permanencia necesita almacenarse ahí. Como en los sistemas biológicos, la precisión requerida por el algoritmo es baja (típicamente menos de 4 bits son suficientes). Por ejemplo, suponiendo un sistema de 2K columnas, con 1K entradas, la agregación de todos los segmentos proximales del cortex requerirán (incluyendo etiquetas) entre 0,25MB y 0,5MB ($12bits \cdot 64 \cdot 2K$; $12bits \cdot 128 \cdot 2K$), que desde una perspectiva de implementación hardware, la tarea de manipulación de esos segmentos parece sencilla.

En cambio, los segmentos distales parecen significativamente más difíciles de manejar. En un enfoque simplista, cada segmento distal requerirá tantas sinapsis como columnas

haya en el sistema. Además, cada celda temporal podría requerir múltiples segmentos (típicamente en el rango de 128 a 256). Para un sistema con 2K columnas, de 32 celdas temporal cada una y 256 segmentos por celda, suponiendo 4 bits para almacenar la permanencia, los segmentos de cada celda requerirán 8MB. Por lo tanto, la memoria total requerida para el sistema será prohibitiva para un sistema físico real. Sin embargo, como en el caso de la biología, sólo son requeridas unas pocas de las potenciales conexiones. Por ejemplo, restringiendo cada segmento a las sinapsis más activas (utilizando, por ejemplo, una aproximación basada en stack o pila) el número de conexiones potenciales que deben reservarse se puede reducir considerablemente.

Por tanto, una vez identificado los tres problemas mencionados anteriormente para los módulos columnados (comunicación y sincronización, complejidad de la lógica temporal de agrupación y organización de los segmentos distales), desde el punto de vista de la escalabilidad, y por tanto el más relevante para la presente invención, es el primero. Teniendo en cuenta que en los sistemas biológicos, la diferencia fundamental entre las especies parece estar dominado por el número de neuronas y no por el número de sinapsis por neurona, parece oportuno pensar que los problemas de la lógica interna de los CCs no son un inconveniente importante ya que las tablas, y el tiempo requerido por la lógica de cálculo para la agrupación temporal, no es necesario que escalen con el número total de columnas.

Por otro lado, es evidente que, al aumentar el número de columnas en el algoritmo de CLA, los requerimientos para comunicar y sincronizar los diferentes CCs serán sustancialmente mayores. El substrato de comunicaciones será el encargado de facilitar el aprendizaje de nuevos patrones temporales y espaciales y, desde esta perspectiva, se aborda el problema más exigente para la presente invención: el substrato de comunicaciones que se necesita para modelar la actividad del axón y la sincronización de los CCs de una manera eficiente y rápida. A continuación se detallan los aspectos clave para dicho substrato de comunicaciones:

A. Características de la Red

Puesto que toda la actividad del axón se modela como paquetes multidifusión, el encaminador utilizado por la presente invención requiere soporte multidifusión. Con el apoyo de la red, las necesidades de energía de los paquetes multidifusión serán menores, ya que se realiza la copia del paquete cerca del destino y permite obtener una latencia más baja, ya que no es necesaria su replicación en la inyección y, por lo tanto, no se produce ningún retraso por este motivo.

El tamaño requerido de paquete es bastante pequeño. Por ejemplo, el tráfico de inhibición requerirá la identificación de la columna de origen y el solapamiento (*Log2Numcolumns* + *Log2NumEncoderInputs*). La actividad Lateral requerirá la columna origen y la identificación de las celdas temporales (*Log2Numcolumns* + *Log2NumTemporalCells*). La actividad de entrada requerirá identificación de la fuente (*Log2Numinputs*). Para un sistema de 2.048 columnas/ entradas, con 32 celdas temporales por columna, el tamaño requerido será de 22, 16 y 11 bits respectivamente. Por lo tanto, algunas de las realizaciones de la invención contemplan enlaces de comunicación estrechos, que disminuyen aún más las necesidades de energía y los costes del encaminador.

De acuerdo a una de las realizaciones particulares de la invención, suponiendo *Log2Numcolumns* enlaces anchos, se pueden emplear paquetes compuestos de un único *flit (Flow Control digiT)* en la mayoría de los casos y, bajo tales circunstancias, las necesidades de almacenamiento dentro de los encaminadores serán bajas.

Durante el estado estacionario, aproximadamente el 2% de las columnas del cortex tendrán actividad. Por lo tanto, una de las realizaciones de la presente invención contempla una red de grado bajo y enlaces estrechos para satisfacer los requisitos. Redes de grado alto requerirían aumentar la complejidad de los encaminadores y el coste del cableado. Por ejemplo, toros o mallas bidimensionales, o incluso las redes de tipo panal, podrían satisfacer tales requerimientos. Otra realización de la invención, en la que se supone una red tolerante a fallos, contempla incrementar el tamaño del sistema sin problemas de defectos de producción (yield) e incluso la utilización de técnicas de integración oblea-oblea en un entorno 3D.

B. Sincronización

El algoritmo CLA comprende principalmente cuatro fases: calcular el solapamiento o superposición de las dendritas proximales con la entrada codificada actual, determinar las columnas ganadoras del cortex, determinar la actividad lateral en cada celda temporal de la columna y producir la predicción. Solapado con esas fases se realiza la adaptación (es decir, el aprendizaje) de los segmentos sinápticos.

La dificultad de ejecutar esas fases de una manera totalmente distribuida, es saber cuándo debe ejecutarse cada una. Por ejemplo, la determinación del solapamiento de la entrada no debe ejecutarse hasta que se reciba toda la actividad de entrada (es decir, que cada columna sea consciente de toda las actividades de los axones). Puesto que no hay mensaje de confirmación de la recepción de la actividad del axón, cada CC debe

ser consciente de cuándo ejecutar la parte correspondiente del algoritmo. Del mismo modo, la inhibición no se puede activar hasta que cada columna sea consciente de si ella misma está dentro de las más activas y, finalmente, la predicción no se puede realizar hasta que se conozca la actividad lateral de las celdas temporales relacionados. La forma más sencilla, pero eficaz, para evitar este problema consiste en vaciar el contenido de la red antes de avanzar a la siguiente fase. Si la red está vacía, hay garantía de que todos los paquetes que influyen ya habrán llegado a su destino.

La **Figura 4** detalla todas las etapas necesarias para el algoritmo CLA. Además de la codificación (40) y clasificación (50), hay nueve etapas adicionales, tres de ellas realizan el cálculo de la lógica espacial y temporal (43, 46 y 49), tres corresponden a la actividad del axón (41, 44 y 47) y, finalmente, otras tres son necesarias para drenar la red (42, 45 y 48).

El problema de la sincronización en la presente invención se reduce entonces a proporcionar un mecanismo escalable de drenaje de la red. Para garantizar la escalabilidad de dicho mecanismo, se necesita una manera simple y efectiva de hacerlo dentro de la propia red. La presente invención contempla, de acuerdo a una de las realizaciones, utilizar encaminamiento en orden de dimensión, inyectar un paquete de difusión especial, llamado *paquete escoba*, en los CCs de los extremos de la red, correspondientes a los identificadores (IDs) más pequeño y el más grande (en el ejemplo de la Figura 3a corresponderían a CC0 y CC15). A cada paquete escoba se le permitirá pasar al siguiente encaminador únicamente si el encaminador local no tiene más paquetes y los buffers de tránsito en los puertos, por los que el encaminador ha recibido las copias del paquete, están vacíos. El paquete se replica en todos los puertos restantes. Por ejemplo, cuando CC5 recibe, de CC4 y CC1, el paquete escoba de CC0, sabemos que no hay paquetes de actividad que pueda afectar a las columnas que maneja CC5. Cuando las colas de tránsito del Oeste y Norte están vacías, el encaminador replica el paquete escoba CC0 a través de los puertos de Sur y Este. Esta operación se aplicará en todo el cortex hasta que el núcleo CC15 reciba el paquete escoba de CC0. En este punto, CC15 es consciente de que ya no hay paquetes en la red para él y puede avanzar a la siguiente etapa del algoritmo. Del mismo modo, cuando un CC intermedio recibe todos los paquetes escoba de CC0 y CC15, sabe que no hay paquetes pendientes en la red para él. Cabe destacar que este mecanismo opera de una manera completamente distribuida y escala según el ancho de banda disponible de la red.

En los sistemas biológicos, tales drenajes parecen no ser necesarios porque la tasa de entrada de los cambios es lo suficientemente espaciada como para garantizar que la actividad espacial y temporal se realiza satisfactoriamente. Cuando la tasa de entrada

es demasiado alta, el sistema será incapaz de aprender o de predecir. Como ejemplo simple, un cambio excesivamente rápido de una imagen será percibido como ruido por el cortex visual. Aunque se podría aplicar una solución similar a la presente invención, el codificador y los datos no están tan sintonizados como en los sistemas biológicos y harán más que recomendable incorporar la solución del drenaje de la red propuesta.

C. Segmentación del Algoritmo

Las etapas del algoritmo requieren una cantidad sustancial de tiempo y energía. Sin embargo, como puede verse en la Figura 4, pueden identificarse *etapas* como en el caso un procesador de propósito general. Por lo tanto, la presente invención recurre a las mismas técnicas de optimización empleadas allí. En particular, de acuerdo a una de las realizaciones, el algoritmo es segmentado para simultanear actividades de distintas etapas y reducir su número a, solamente, tres por para cada dato de entrada. La **Figura 5** muestra cómo esa organización será beneficiosa una vez que se carga el pipeline. La idea es comenzar a calcular el solapamiento de la próxima entrada tan pronto como se haya calculado el solapamiento actual. Entonces, en el intervalo 54, se llevan a cabo dos operaciones en la red simultáneamente. Si avanzamos en el tiempo comprobamos que pueden superponerse tres operaciones de entrada diferentes en una sola etapa. En el intervalo 57 estamos transmitiendo la comunicación de la actividad distal del primer valor de entrada, el tráfico de inhibición del segundo valor de entrada y la realización de la actividad proximal del tercer dato de entrada. En el intervalo 59 se lleva a cabo, de forma simultánea, la predicción para la primera época, el cálculo de la actividad lateral para el segundo y el cálculo del solapamiento para el último. Y, lo que es más importante aún, solamente se necesita un drenaje de la red por cada valor de entrada. Una vez que se carga el pipeline, sólo se necesitan tres intervalos en la secuencia de entrada para producir una predicción.

D. Superposición de Comunicación y Computación

Simultanear las etapas del algoritmo, como se ha explicado anteriormente, abre la posibilidad a mejoras adicionales, ya que no es necesario terminar la fases de cálculo (es decir, el cálculo del solapamiento, de la actividad lateral y de la predicción), antes de comenzar a enviar el resultado de cada una. Tan pronto como la lógica de cálculo comienzan a generar la actividad del axón, ésta se puede inyectar en la red. Por lo tanto, el número de ciclos de reloj necesarios para procesar un valor en la secuencia de entrada será determinado por la parte más lenta: comunicación o cálculo. El número de ciclos requeridos por el más lento y el tiempo de ciclo de reloj determinarán el tiempo necesario para procesar una muestra de la secuencia de entrada. Por último, la red de

drenaje debe ser cuidadosamente manejada: los paquetes escoba se envían al encaminador de cada CC si todas las columnas locales han finalizado las acciones que se deben realizar en el intervalo actual.

E. Tráfico Agregado.

La organización óptima de acuerdo a una de las realizaciones de la presente invención y desde el punto de vista de la latencia, es la combinación de varias columnas en un solo CC. Utilizar muchos encaminadores con enlaces muy cortos puede aumentar innecesariamente la latencia media en la red. Para optimizar dicha latencia, el tamaño del CC (es decir, el número de columnas que maneja) debe ajustarse para que el retardo de propagación y el ciclo de reloj de la red sean similares. Con este enfoque es posible agregar múltiples activaciones de los axones procedentes de columnas en el mismo CC, en un único paquete. Aunque esto podría aumentar el número de flits del paquete (su longitud), ello reducirá significativamente la carga de la red.

Adicionalmente, el algoritmo segmentado permite combinar, en un único paquete, las acciones procedentes de diferentes etapas en el algoritmo. Por ejemplo, la información de la inhibición se puede combinar con la de las activaciones laterales de la época anterior y por consiguiente, agruparla en un solo paquete. Para llevarlo a cabo, se supone la existencia de colas de inyección de agrupamiento (similar a la estructura para soportar caches no bloqueantes, generalmente llamada MSHR (miss information/status handling registers), donde cualquier nuevo paquete inyectado se coteja con los que están esperando a ser inyectados. Si hay una coincidencia en la máscara de destino, el paquete anterior se modifica para contener la información del que acaba de llegar y así, el nuevo paquete puede ser desechado.

F. Escalabilidad

El sistema biológico sugiere que el mejor enfoque para aumentar el almacenamiento es aumentar el número de columnas y no el número de celdas temporales (y segmentos distales) por columna. Desde una perspectiva práctica, si aumentamos el número de columnas podríamos reducir el número de segmentos distales requeridos por celda temporal. Aunque desde la perspectiva software esto parece interesante, desde el punto de vista del hardware es realmente relevante porque podría reducir el coste de la interconexión y tal vez la complejidad de cada CC. Por lo tanto, la presente invención, en una de sus realizaciones, contempla aumentar el número de columnas tanto como lo permita la tecnología. Desafortunadamente, el sistema de comunicación, tal como se ha descrito hasta el momento, podría escalar hasta un número limitado de columnas, pero, claramente, los requisitos de energía no escalarán empleando tecnología CMOS. A

continuación, se tratará este aspecto de acuerdo a diferentes realizaciones de la invención que introducen dos estrategias complementarias:

1) Tráfico proximal escalable: Parches proximales

Las implementaciones software del algoritmo conocidas en el estado del arte suponen que las sinapsis proximales, en una columna dada, no tienen en cuenta la topología del sistema. En el momento del arranque, la entrada codificada está potencialmente conectada a un subconjunto de las columnas elegidas al azar (por defecto, en torno al 20%). Durante el agrupamiento espacial, el sistema aprende las inter-relaciones relevantes según sea la secuencia de entrada. Aunque desde la perspectiva del software esto es beneficioso, ya que equilibra la utilización de las columnas, para una implementación hardware resulta muy exigente. Por ejemplo, empleando CCs con 5 columnas y un subconjunto del 20% de columnas, este enfoque implica que la activación del axón en el codificador requerirá una multidifusión a todos los CCs del sistema. Ciertamente, este enfoque se aparta del funcionamiento de los sistemas biológicos. Desde tal perspectiva, la presente invención opta por, de acuerdo a una de sus realizaciones, limitar las columnas potencialmente conectadas a una zona restringida topológicamente en la red, que se hará referencia como *parche proximal.* La **figura 6** muestra un ejemplo de un parche proximal (60) en una topología de tipo panal. De acuerdo a esta realización particular de la invención, el codificador se conecta a la red a través de colas de inyección, donde cada bit se conecta a un encaminador diferente en la periferia del circuito. En aras de la simplicidad, la figura 6 sólo ilustra esto para dos bits separados, pero téngase en cuenta que el codificador tendrá miles de bits de salida. De esta forma las columnas que podrían conectarse a cada CC quedan restringidas dentro del parche proximal.

La presente invención, en una de sus realizaciones, define la posición del parche de forma aleatoria en el momento del arranque. Su tamaño es un parámetro de diseño y se puede redefinir según la naturaleza de la entrada o la aplicación concreta. Experimentalmente, se ha observado que una regla de tamaño del 20% es válida, aunque el aumento o disminución dinámicos del parche también se pueden contemplar para equilibrar la utilización de columnas.

Bajo tales circunstancias, cuando se active la entrada unida al módulo columnado R1 (61), se generará una multidifusión a los CCs dentro del parche. El paquete se inyecta en la red y se comporta como unicast hasta llegar al módulo columnado CC1 (62). La información de cabecera debe incluir tal módulo columnado 62 como nodo intermedio y la máscara de multidifusión para los nodos restantes.

2) Trafico distal y de inhibición escalable

Del mismo modo, el tráfico distal y de inhibición se supone global para la implementación del software (aunque la inhibición puede ser local). Desde la perspectiva de la red, el retardo y la potencia requeridos se incrementarán significativamente a medida que aumentemos el número de CCs. Conviene señalar que el número de columnas involucrado en el proceso de inhibición es notablemente mayor que las entradas activas en el codificador (cualquier solapamiento de entrada distinto de cero requerirá una multidifusión). Sin embargo, los sistemas biológicos, sin duda, no utilizan una comunicación global. A partir de esta hipótesis, la presente invención, de acuerdo a una de sus realizaciones, divide la red en zonas separadas y restringe la inhibición y el tráfico distal a su interior. Dichas regiones serán referidas como zonas de *scale-out*.

La **figura 7** representa gráficamente cómo incrementar el número de CCs desde 16 hasta 64 en una de las realizaciones de la invención. En lugar de requerir multidifusión o emisiones completas, el tráfico generado por columnas, en cualquiera de las cuatro zonas representadas (71-74), se restringe a circular por su interior. Si necesitamos aumentar aún más el número de columnas, sólo debemos aumentar el número de zonas, con lo que el tráfico se mantiene casi constante aunque el sistema escale enormemente.

El codificador, es decir, el tráfico proximal, que ve la red a nivel global, selecciona las columnas potencialmente conectadas sin hacer distinciones entre las zonas. De esta forma se aumenta el número de columnas disponibles y, por tanto, la capacidad de representación de valores (precisión). Por ejemplo, con un número de columnas de aproximadamente 2K-4K por zona, esta flexibilidad adicional podría no ser útil para reducir los requisitos de memoria en los segmentos distales e incrementar la complejidad del codificador. La presente invención, en una de sus realizaciones, contempla utilizar tantos valores consecutivos en la secuencia de entrada codificada como número de zonas de scale-out. En este ejemplo, se utilizan 4 codificadores para codificar, simultáneamente, cuatro intervalos diferentes de la secuencia de entrada. De esta manera, no sólo aumenta el rendimiento del sistema, sino también la carga sobre cada columna individual. Además, el aumento del número de zonas mantendrá constante el tráfico proximal total (ya que cada entrada en la secuencia, activa un número de bits de entrada proporcional al número de columnas por zona).

Diferentes realizaciones de la presente invención han sido testadas en simuladores adaptados para emplear estructuras de datos y mecanismos apropiados para una implementación hardware factible y obtener resultados energéticos y de rendimiento

precisos. El codificador SDR de entrada se ha implementado utilizando un generador pseudo-aleatorio Mersenne Twister. Para el modelado de entrada se han utilizado series temporales sintéticas, específicamente la serie periódica de datos enteros de 32 bits, generados a partir de polinomios definidos aleatoriamente (hasta cuarto grado con coeficientes elegidos al azar). Se define una serie temporal mediante veinte valores de cada uno de ellos. Se repite cada serie temporal hasta que es aprendida por el sistema, lo que se produce cuando el número de elementos en la secuencia, con columnas sin predicciones incorrectas (es decir, no hay ráfagas de columna), es igual a la mitad de todos los puntos de datos. De esta manera, el sistema se mantiene la mitad del tiempo aprendiendo nuevas secuencias y la otra mitad simplemente prediciéndolas. Por lo tanto, durante la mitad de los intervalos se producirá el tráfico adicional producto de las ráfagas de columnas o el tráfico de inhibición que generará la aparición de una nueva secuencia de entrada. Durante la segunda mitad del tiempo, el sistema tendrá una representación estable de la entrada, siendo mucho benigna para el tráfico de la red. El número de series temporales (es decir polinomios) necesarias para cumplir un intervalo de confianza del 98% es aproximadamente de 500. Por último, el clasificador utilizado en estas pruebas es el más simple de la aplicación NuPIC, que proporciona una puntuación de anomalía como la fracción de las columnas con fallo de predicción.

Una vez presentadas las condiciones de las pruebas, se presentan a continuación algunos de los resultados obtenidos por la presente invención, demostrando que logra mejorar los resultados de retraso y energía del sistema, a la vez que el análisis de la escalabilidad demuestra su viabilidad para decenas de miles de módulos columnados CCs en el sistema.

En una primera realización de la invención, se utiliza la configuración por defecto utilizada por la aplicación NuPIC: 2048 columnas, con 32 celdas temporales por columna, e inhibición global. Se emplea una topología de malla 2D, con un encaminador básico convencional, encaminamiento determinista DOR, un pipeline de 4-ciclos, y empleando *virtual-cut through* como control de flujo. Suponemos cables de enlace *low-swing* y que requieren de un ciclo de reloj para trasladar un flit de un encaminador a otro y se incluyen buffers de entrada de 1280 bytes, sin canales virtuales. Hay que tener en cuenta que, con esta configuración, todo el tráfico de multidifusión se transmite a todos los CC del sistema. Por lo tanto, empleando replicación en orden de dimensión en los encaminadores intermedios, se obtiene una red libre de bloqueos. El encaminador ha incorporado el mecanismo de drenaje de la red, descrito anteriormente.

La **figura 8** representa gráficamente el número de ciclos de reloj, por intervalo, para diferentes tamaños de malla cuadrada 2D. Es decir, el número de ciclos de red necesarios para llevar a cabo las tareas de cada intervalo de entrada, siguiendo un enfoque secuencial y segmentado para los diferentes tamaños de red (diferente número de columnas por CC). Como se puede apreciar, hasta 300 ciclos pueden ser ahorrados segmentando y solapando el proceso de aprendizaje. Otra observación que puede no resultar intuitiva es el comportamiento de la aproximación segmentada, ya que cuando se aumenta el tamaño de la red, el tiempo requerido apenas se modifica. La razón de este comportamiento es la contención. En este caso, la red recibe una carga mayor (ya que las tres fases de la comunicación se solapan). Po ello, cuando se reduce el tamaño de la red se reduce el ancho de banda disponible y consecuentemente la contención crece. En estas condiciones, parece que la ventaja de ancho de banda compensa el incremento de distancia promedio. Con la aproximación secuencial, la contención únicamente es notable en la malla 4x4.

La **figura 9** representa gráficamente el número de ciclos de reloj por intervalo para diferentes mallas cuadradas 2D, empleando tráfico agregado (enlaces de 16 bytes de anchos con paquetes de 5 flits). En esta figura se ha introducido tráfico agregado. Bajo las mismas condiciones que la realización anterior, se modela cuidadosamente la longitud del paquete, suponiendo su tamaño de 5-flit con enlaces de 16 bytes de ancho, pero se agrega otro paquete cuando se supera ese límite por el proceso de agrupamiento. Como puede verse en dicha figura, los beneficios son notables, siendo capaz de procesar las necesidades de comunicación del sistema de un intervalo, en menos de 60 ciclos de reloj de la red. Respecto a la configuración de la red, el resultado obtenido invierte la observación anterior sobre su tamaño, ya que la reducción de tráfico es tan drástica que la contención no está presente en ningún caso. Por lo tanto, bajo tal configuración, el factor dominante es la distancia promedio de la red.

La **figura 10** representa gráficamente el número de ciclos de reloj por intervalo para diferentes mallas cuadradas 2D, con el algoritmo segmentado, agregación de tráfico y parches proximales aplicados. En esta realización se han introducido parches proximales, lo que implica un beneficio significativo. En ambos casos, han sido seleccionadas el 20% de las columnas en el sistema. Teniendo más de 5 columnas en cada CC, la distribución uniforme implica una difusión. Los parches proximales reducen esto de manera significativa, especialmente cuando el tamaño de la red crece (y el beneficio de convertir una difusión en una multidifusión localizada es más grande). El tráfico proximal no cambia significativamente la puntuación de anomalía (es decir, la

probabilidad de perder una activación de la columna en la agrupación temporal), siendo alrededor del 6% al final de la simulación, en ambos casos.

Teniendo la contención tan poco impacto en la red, parece razonable reducir el ancho de banda del enlace. Los resultados anteriores corresponden a enlaces de 16 bytes de ancho, que es un tamaño bastante convencional para muchos sistemas contemporáneos donde se utilizan redes en chip. Por ejemplo, para una malla 16x16 el ancho de banda de la bisección es aproximadamente de 512 GB/s, suponiendo 1ns de ciclo de reloj. Por lo tanto, en tales circunstancias, parece interesante explorar el efecto de la reducción de la anchura de enlace con el fin de reducir el consumo de energía y el área.

La **figura 11** representa gráficamente el número de ciclos de reloj, variando la anchura del enlace. Para ello se utiliza un ejemplo concreto en el que puede verse la variación del tiempo requerido para procesar un intervalo cuando el ancho del enlace varía desde 16 bytes hasta 1 byte. En este punto, con el fin de seleccionar la mejor configuración de la red, hay que equilibrar su retardo y el coste de la lógica de cálculo temporal/espacial en los CCs, para lo que es importante tener en cuenta, al menos, los siguientes 3 aspectos: (1) el aprendizaje, en ambos casos, está fuera del camino crítico; (2) la lógica espacial es bastante simple (necesita calcular el solapamiento de las entradas) y ya que opera en paralelo con la lógica temporal, tampoco estarán en la ruta crítica del circuito; (3) la generación de la actividad lateral en la celda temporal, que está en el camino crítico del algoritmo, está dominada por el acceso a la memoria donde se almacenan los segmentos distales. Por consiguiente, el número de accesos a dicha memoria resulta un elemento clave en toda la lógica que, de acuerdo a diferentes realizaciones de la invención, podrá estructurarse de diferentes formas. En cualquier caso, la hipótesis optimista es que no se necesitará 1 ciclo de reloj por segmento distal, que cada columna tiene 32 celdas temporales y cada celda temporal requiere un promedio de 1 segmento, con lo que se necesitarán 32 ciclos de reloj para procesar una sola columna. En un sistema de de 2K columnas, como el utilizado con estos resultados, el cómputo requerirá aproximadamente desde 4000 ciclos (32 2048/16) en una red de 4x4, a 64 ciclos de reloj en una red 32x32. Por lo tanto, en tales supuestos, la red más apropiada es una red de 256 nodos con enlaces de 2 bytes de ancho. La escalabilidad del retraso de la red permitirá un adecuado ajuste respecto al coste de la lógica de computación.

Una de las mayores ventajas que se persiguen, y que soluciona un gran problema del estado del arte, es incrementar notablemente la capacidad de estos sistemas y ofrecer una escalabilidad real que contemple millones de columnas. La presente invención consigue dichas ventajas y la prueba de ello queda reflejada en la figura 12.

En la **figura 12**, a partir de una configuración fija para la red que mantiene las optimizaciones anteriores, es posible comparar sus resultados con un sistema con 4 zonas scale-out, es decir, en total 8K columnas, pero manteniendo inalterada la configuración de red. Se muestran así los ciclos de reloj de red requeridos para procesar un intervalo de la entrada actual. Se muestra el tiempo requerido empleando dos anchuras de enlace, para diferentes tamaños de red, y con zonas de scale-out y sin ellas. Como puede apreciarse, el retardo es mucho menos sensible al diámetro de la red, pudiendo utilizar una malla de 32x32 (M1024) con enlaces de 2 bytes de ancho y con unos resultados similares a los del sistema plano (alrededor de 200 ciclos de reloj). Según estos resultados, la inhibición y el tráfico distal dominan la carga de la red, ya que el aumento del tráfico proximal en los destinos es insignificante. El uso de 8 zonas scale-out (es decir, 16K columnas) reporta los mismos resultados. Por lo tanto, se evidencia que la presente invención ofrece un sistema donde el retardo de comunicación es independiente del número de columnas.

En este punto, suponiendo un ciclo de reloj de red de 1ns y que la comunicación es el elemento crítico del algoritmo, la presente invención es capaz de procesar hasta 100 millones de valores de la secuencia de entrada por segundo. Este rendimiento no es alcanzable por ningún enfoque software. Por ejemplo, la implementación actual más rápida de Nupic puede procesar en 1 segundo en torno a 1000 entradas (para un tamaño de sistema similar ejecutándose en una máquina media actual).

Incrementar el número de CCs en el sistema aumentará la energía dinámica de red (cada paquete recorrerá más enlaces y routers). La **figura 13** presenta la energía dinámica requerida por la red para procesar un intervalo. Como se puede ver, el uso de las zonas de scale-out reduce, hasta en 3 veces, los requisitos de la red, siendo posible reducir, casi cuadráticamente, los efectos negativos de su tamaño.

Para finalizar esta descripción, se detalla a continuación el rendimiento, los requisitos de almacenamiento y la eficiencia de predicción de todas las realizaciones anteriores. La **figura 14** muestra, desde el punto de vista de rendimiento, hasta un 92% de los efectos de red que podrían solaparse con las otras actividades en el camino crítico.

La **figura 15** muestra cómo, del mismo modo, se reduce la energía dinámica de la red, lo que confirma que los problemas de comunicación son casi inexistentes y que, por lo tanto, afianza la idea de la presente invención de que la solución basada en la conmutación de paquetes es una propuesta factible.

La **figura 16** presenta la probabilidad de fallos de predicción por columna, por cada experimento (cada experimento tiene aproximadamente 3 millones de intervalos diferentes). Algunos de los cambios introducidos alteran, ligeramente, el algoritmo original de CLA, pero en la mayoría de los casos, los márgenes de confianza indican que el promedio de los casos es similar (es decir, no hay cambios de exactitud en los resultados). Como se señaló anteriormente, el valor no normalizado es alrededor del 6%. Sin embargo, los parches proximales parecen mejorar ligeramente esta cifra a menos del 5%. Al parecer, esta optimización del tráfico sugerida por la biología, es beneficiosa desde el punto de vista de precisión. Como era de esperar, el uso de las zonas scale-out disminuye ligeramente la precisión del sistema de nuevo a los resultados obtenidos con el algoritmo de base.

REIVINDICACIONES

1. Un sistema de aceleración por hardware para almacenar y recuperar información, que implementa un algoritmo de aprendizaje cortical a través de una red de conmutación de paquetes, el sistema comprende:

- **al menos un módulo codificador** configurado para codificar una entrada binaria en una representación distribuida dispersa (SDR), y para enviar, por cada bit activo de la SDR, un paquete multidifusión a un módulo columnado determinado a través de la red de conmutación de paquetes, en función de una tabla de correspondencias previamente establecidas;
- **una pluralidad de módulos columnados** conectados mediante dicha red de conmutación de paquetes, configurados para recibir los paquetes multidifusión enviados desde el codificador, donde cada uno de los módulos columnados comprende a su vez:
  - **un encaminador** con soporte multidifusión configurado para recibir paquetes desde el módulo codificador, entregar dichos paquetes a ciertos módulos de memoria del módulo columnado y enviar paquetes desde los módulos de memoria a un clasificador de salida;
  - **una pluralidad de módulos de memoria** configurados para almacenar las entradas recibidas desde el encaminador y almacenar información de contexto;
  - **un módulo de cálculo** configurado para determinar un grado de solapamiento entre el contenido de los ciertos módulos de memoria y la entrada actual, seleccionar un número determinado de módulos de memoria con mayor grado de solapamiento, determinar un contexto temporal para cada uno de los módulos de memoria seleccionados, realizar una predicción de la salida del sistema en función de la entrada actual y la información de contexto temporal y enviar un paquete de salida que contiene dicha predicción a un módulo clasificador de salida;
- **un módulo clasificador** de salida configurado para recibir un paquete de salida, enviado a través de la red de conmutación desde cualquiera de los módulos columnados, y para seleccionar una salida del sistema entre un grupo de salidas preestablecidas en función del paquete de salida recibido.

2. Sistema de acuerdo a la reivindicación 1, donde el módulo de cálculo comprende un comparador, un sumador y un contador.

3. Sistema de acuerdo a cualquiera de las reivindicaciones anteriores, donde cada módulo de memoria de la pluralidad de módulos de memoria comprende una pluralidad de celdas temporales que adoptan un estado activo o un estado no activo y su combinación representa un determinado contexto temporal para el módulo de memoria.

4. Sistema de acuerdo a la reivindicación 3, donde el módulo de cálculo está además configurado para comprobar si su predicción de salida es correcta; en caso de predicción errónea se produce una ráfaga que pone todas las celdas temporales del módulo de memoria en estado activo.

5. Sistema de acuerdo a cualquier de las reivindicaciones anteriores, donde el módulo de cálculo está configurado para simultanear etapas y, dada una secuencia de entrada, producir una predicción en tres intervalos de dicha secuencia.

6. Sistema de acuerdo a la reivindicación 5 donde el módulo de cálculo está además configurado para agregar tráfico de diferentes etapas en un mismo paquete.

7. Sistema de acuerdo a cualquiera de las reivindicaciones anteriores, donde los módulos columnados de los extremos de la red, están configurados para inyectar en la red de conmutación de paquetes un paquete escoba, el cual se replica en el resto de módulos columnados únicamente cuando el encaminador correspondiente no tiene más paquetes en cola hasta que dicho paquete escoba alcanza el extremo opuesto de la red, lo que indica que la red ha sido vaciada.

8. Sistema de acuerdo a cualquiera de las reivindicaciones anteriores, donde el número de módulos de memoria que comprende cada uno de los módulos columnados está determinado por un equilibrio entre el retardo de propagación y el ciclo de reloj del sistema.

9. Sistema de acuerdo a cualquiera de las reivindicaciones anteriores donde, al menos, un módulo codificador está configurado para enviar un paquete de entrada a una selección de módulos columnados preestablecida aleatoriamente que representa en torno al 20% del total de módulos columnados.

10. Sistema de acuerdo a cualquiera de las reivindicaciones anteriores fabricado en una placa de silicio, un chip o un microprocesador utilizando tecnología CMOS.

11. Método escalable de aceleración por hardware para almacenar y recuperar información a través de una red de conmutación de paquetes, el método comprende los pasos de:

a) codificar, en un módulo codificador, una entrada binaria en una representación distribuida dispersa (SDR)

b) enviar, por cada bit activo de la SDR, un paquete multicast desde el módulo codificador a un módulo columnado determinado de una pluralidad de módulos columnados a través de la red de conmutación de paquetes, en función de una tabla de correspondencias previamente establecidas;

c) recibir los paquetes enviados desde el módulo codificador, a través de la red de conmutación de paquetes, en un encaminador del módulo columnado;

d) entregar dichos paquetes a ciertos módulos de memoria del módulo columnado;

e) almacenar en los ciertos módulos de memoria los paquetes recibidos;

f) determinar, en un módulo de cálculo del módulo columnado, un grado de solapamiento entre el contenido de los módulos de memoria que han recibido el paquete de entrada y la entrada actual;

g) seleccionar, por el módulo de cálculo, un número determinado de módulos de memoria con mayor grado de solapamiento;

h) determinar, por el módulo de cálculo, un contexto temporal para cada uno de los módulos de memoria seleccionados;

i) realizar, por el módulo de cálculo, una predicción de la salida del sistema en función de la entrada actual y la información de contexto temporal almacenada en los módulos de memoria;

j) enviar un paquete de salida que contiene dicha predicción a un módulo clasificador de salida;

k) recibir un paquete de salida en el clasificador de salida, enviado a través de la red de conmutación desde cualquiera de los módulos columnados;

l) seleccionar, en el clasificador de salida, una salida del sistema entre un grupo de salidas preestablecidas en función del paquete de salida recibido.

12. Método de acuerdo a la reivindicación 11, que además comprende comprobar si la predicción de salida realizada por el módulo de cálculo es correcta, donde, en caso de

predicción errónea se produce una ráfaga que pone todas las celdas temporales del módulo de memoria en estado activo.

13. Método de acuerdo a cualquiera de las reivindicaciones 10-12 que además comprende comprobar que la red de conmutación de paquetes está vacía antes de ejecutar la etapa de calcular el solapamiento y antes de determinar el contexto temporal, donde, para comprobar que la red está vacía, se proporciona un paquete escoba que recorre la red de conmutación de paquetes.

14. Método de acuerdo a cualquiera de las reivindicaciones 10-13 que además comprende el paso de restringir los paquetes enviados por el módulo codificador a una selección de módulos columnados preestablecida aleatoriamente que representa en torno al 20% del total de módulos columnados.

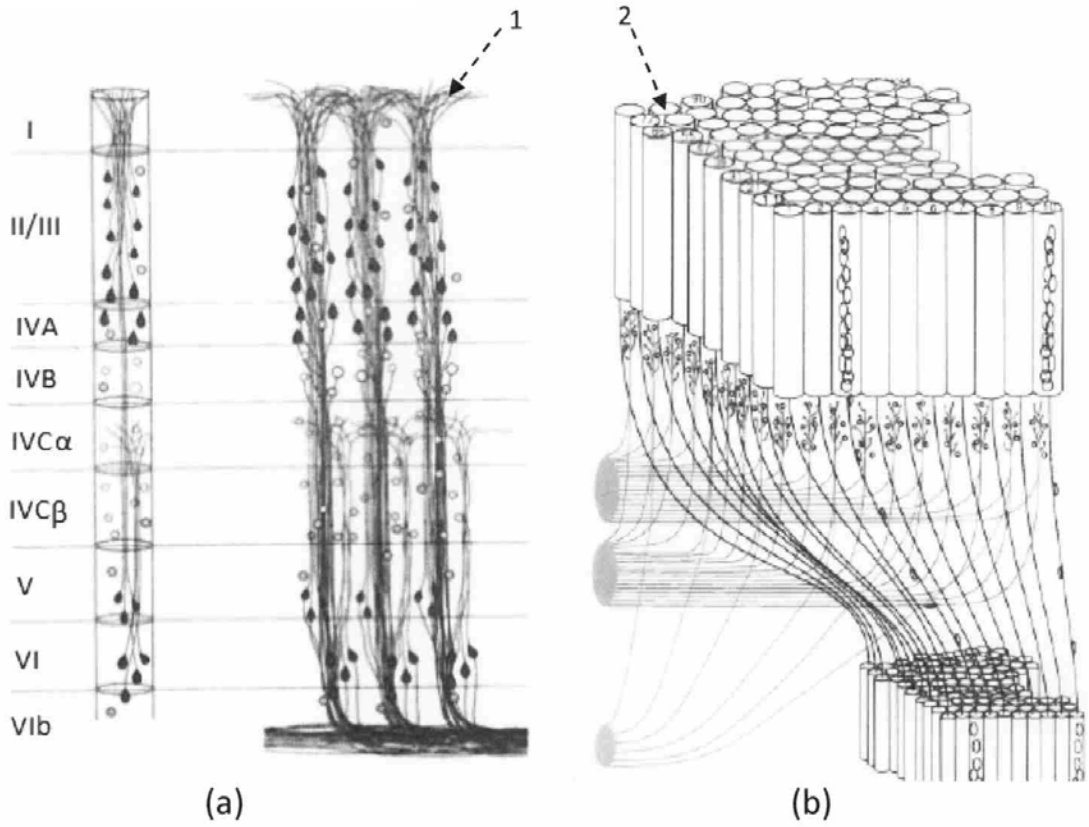

Figura 1

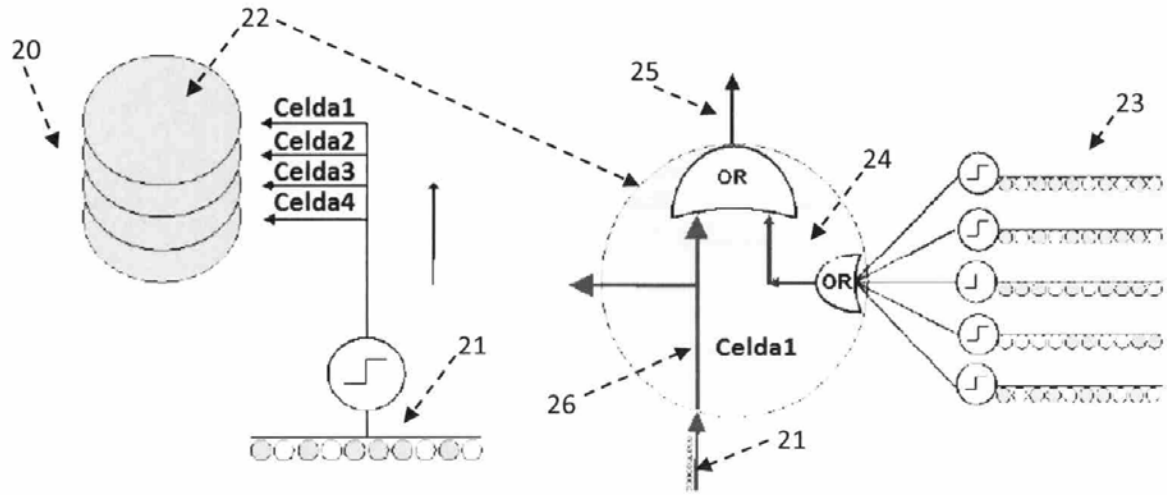

Figura 2

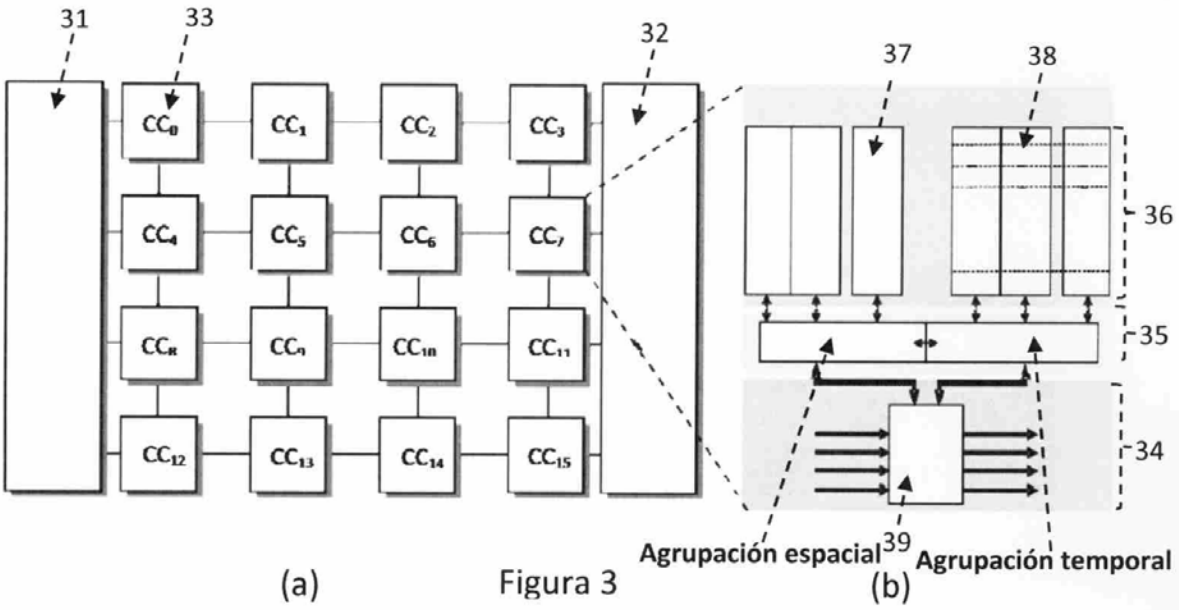

Figura 3

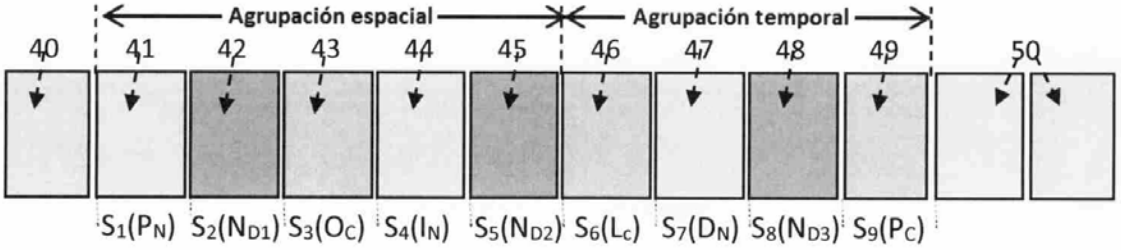

Figura 4

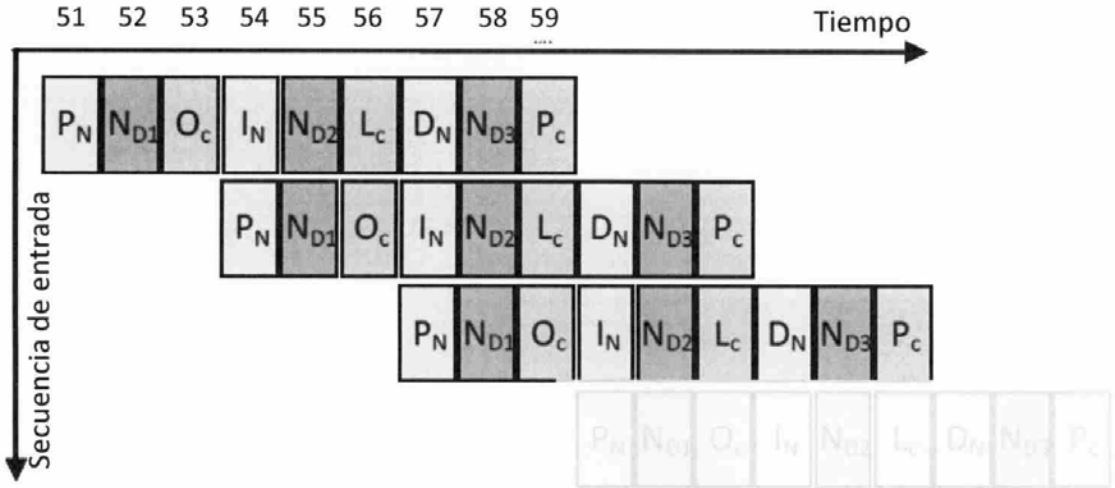

Figura 5

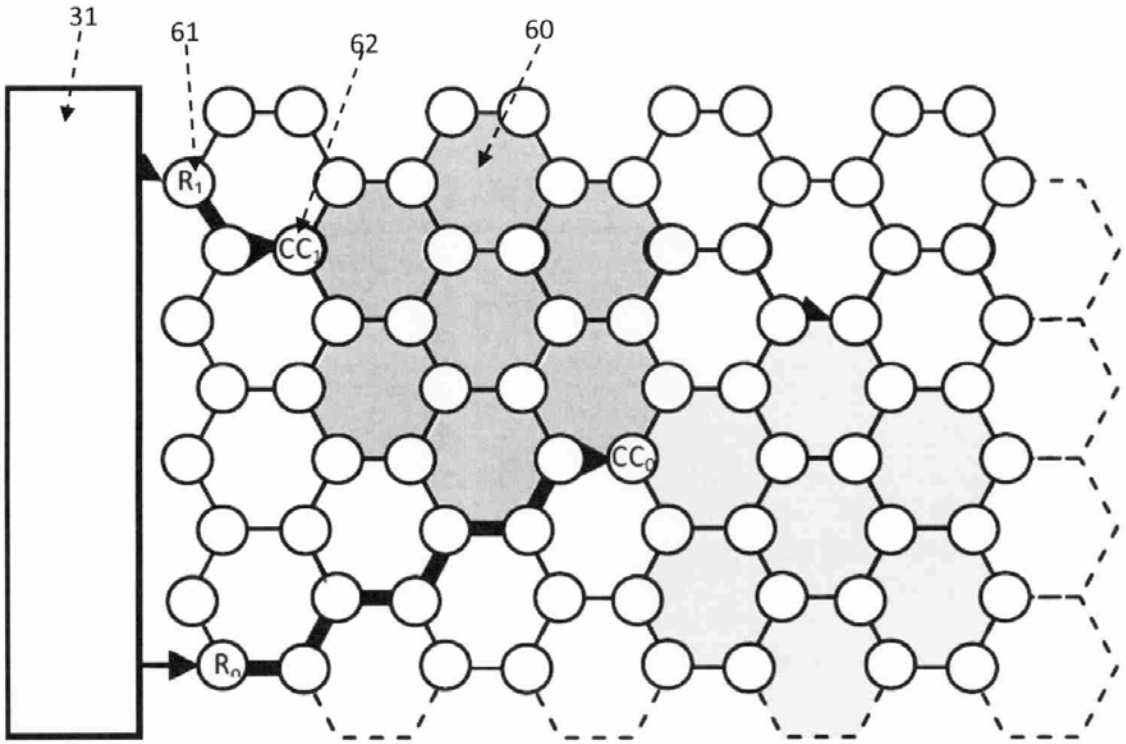

Figura 6

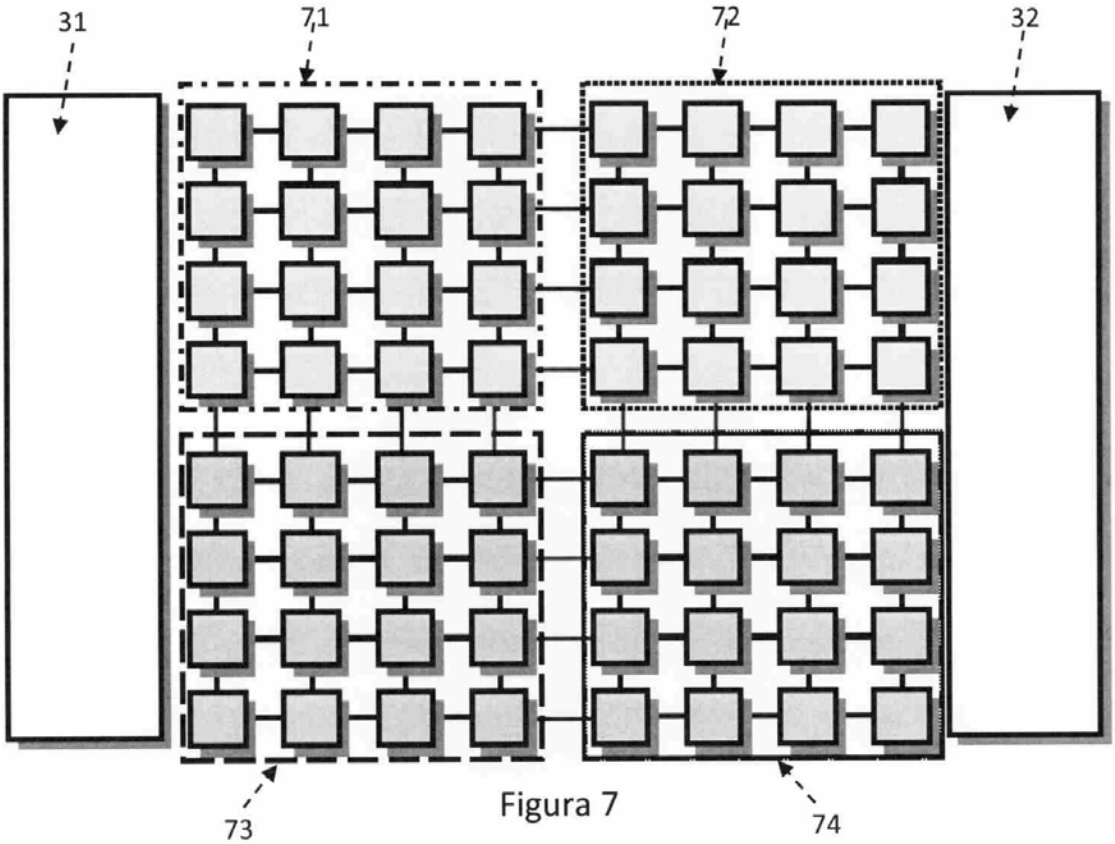

Figura 7

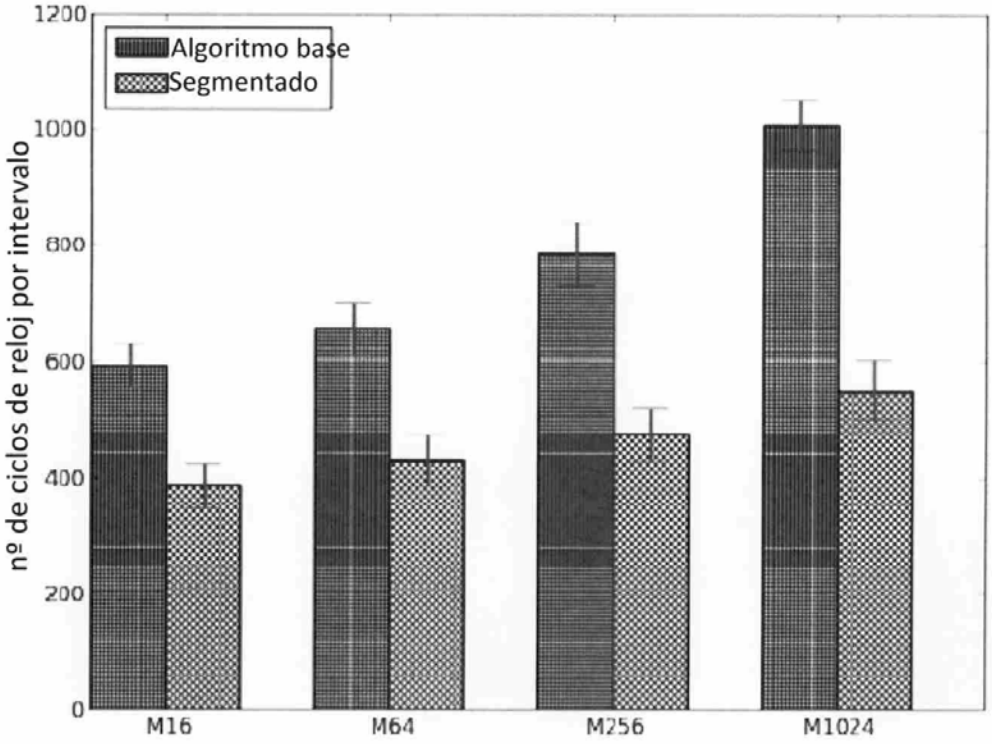

Figura 8

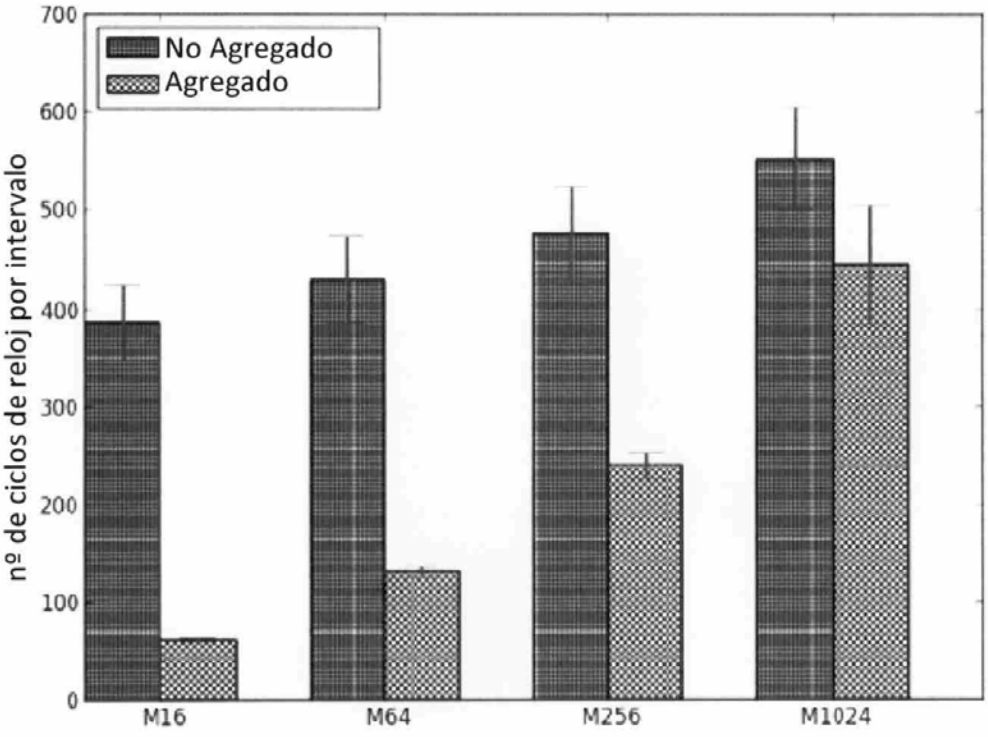

Figura 9

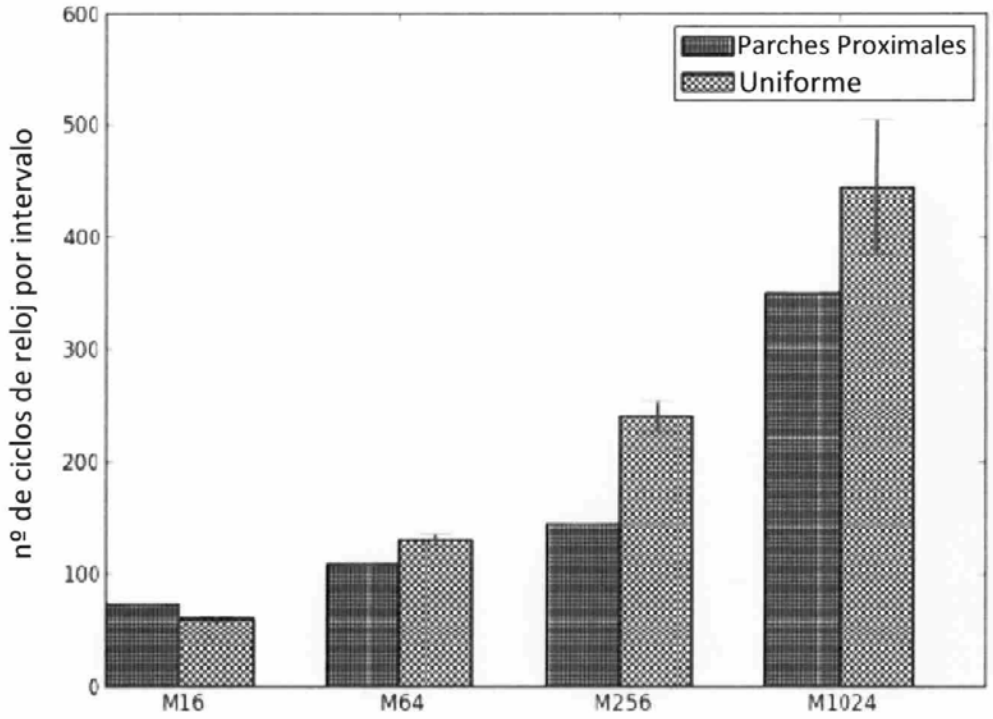

Figura 10

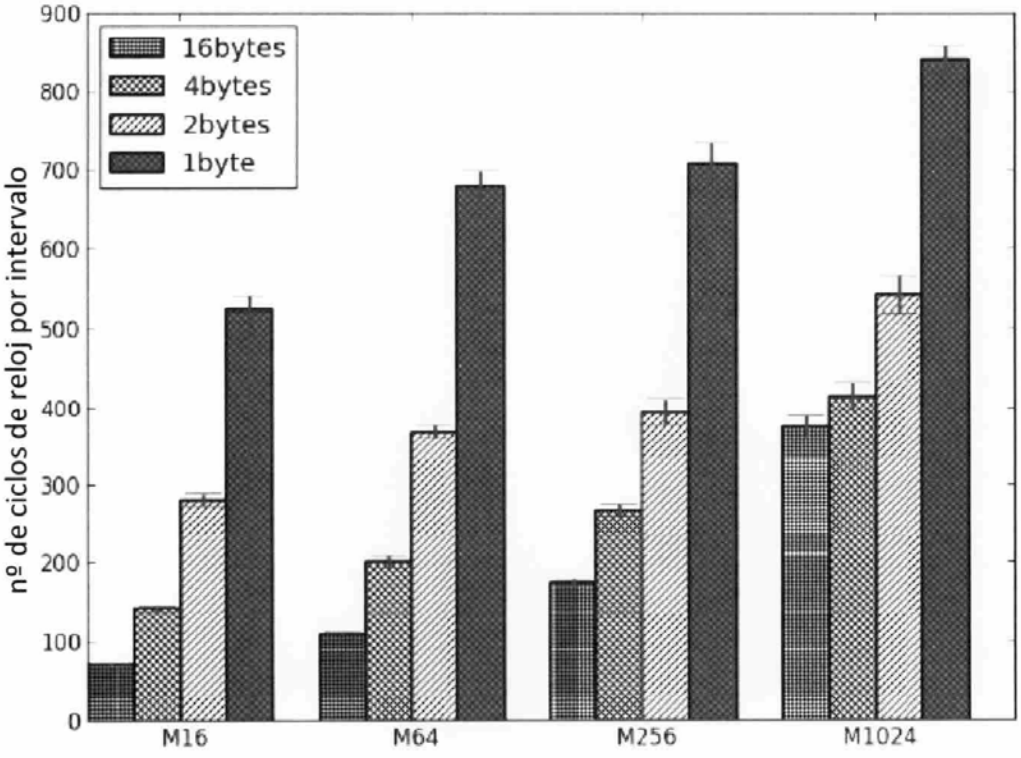

Figura 11

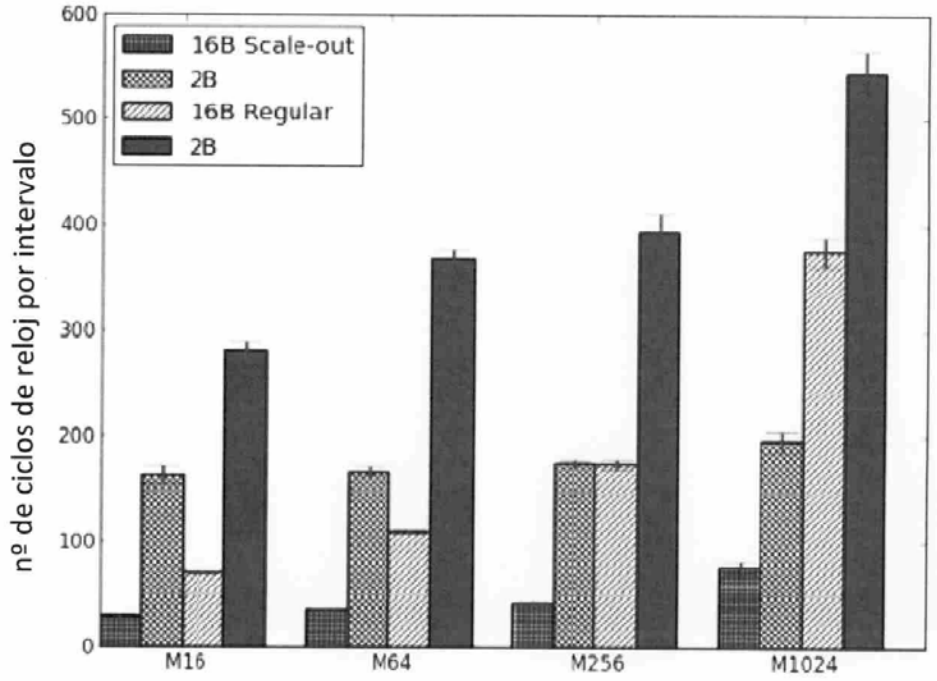

Figura 12

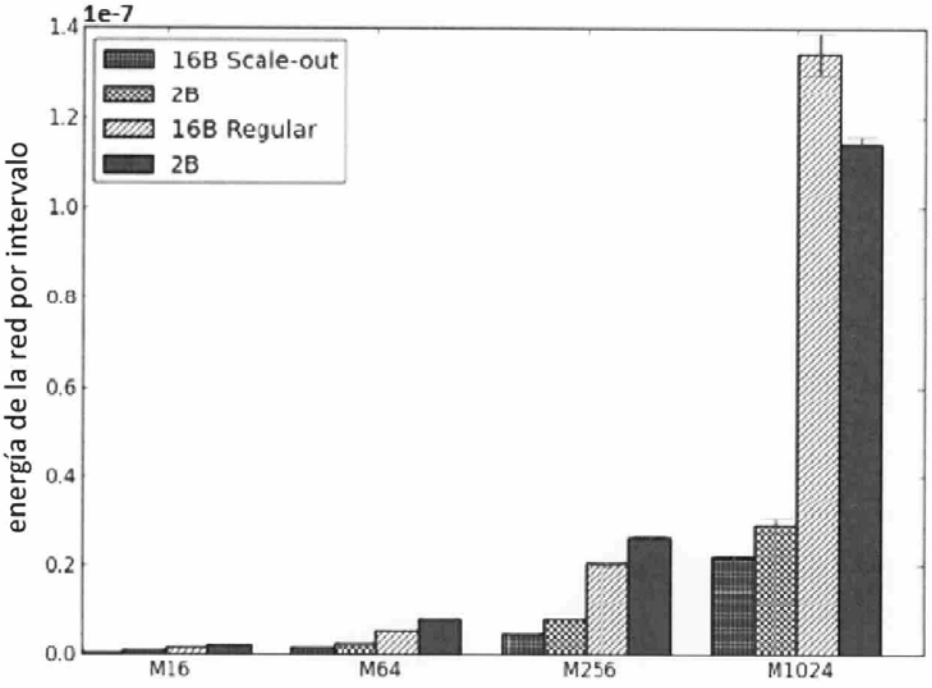

Figura 13

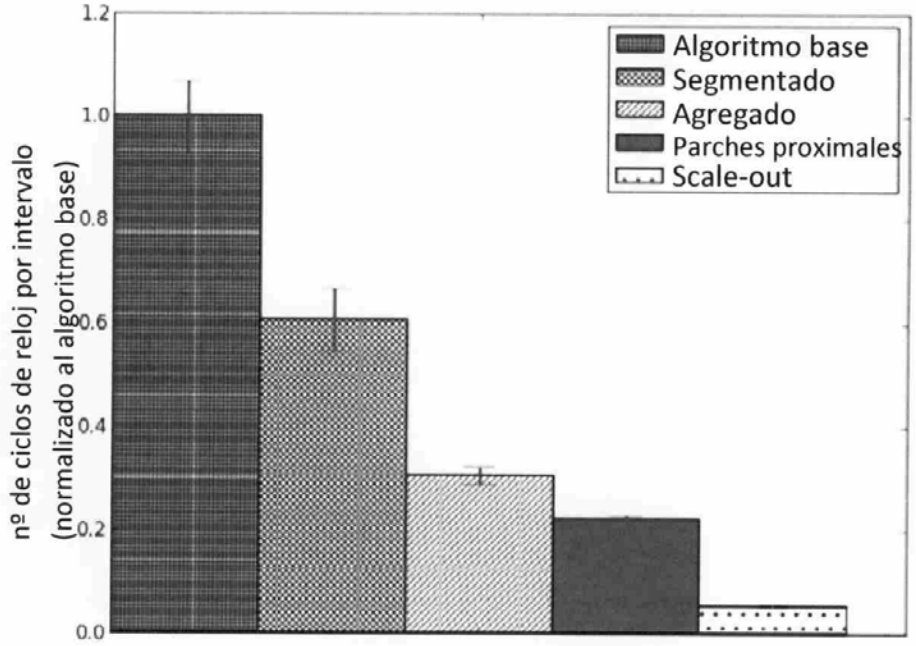

Figura 14

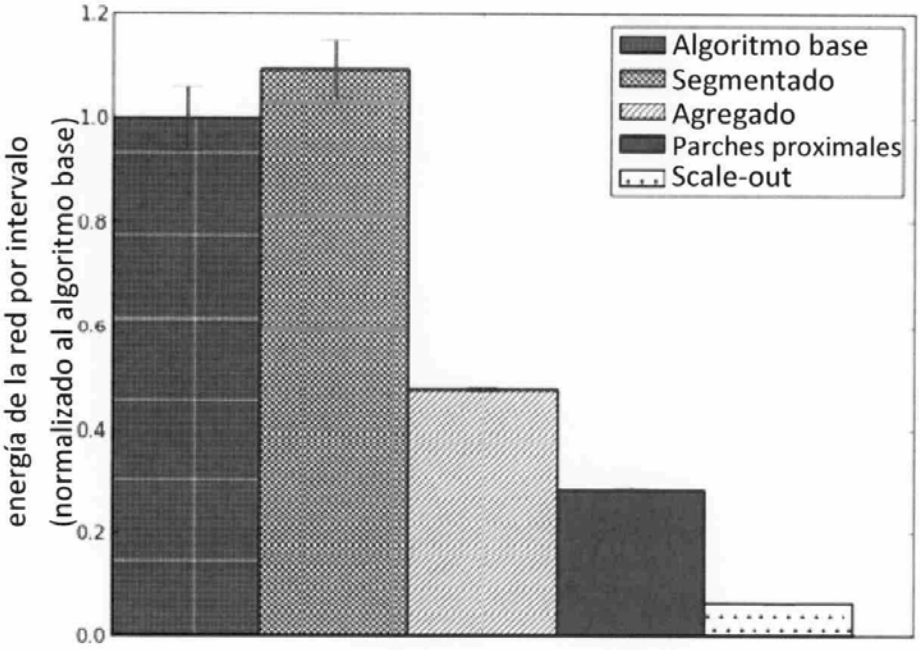

Figura 15

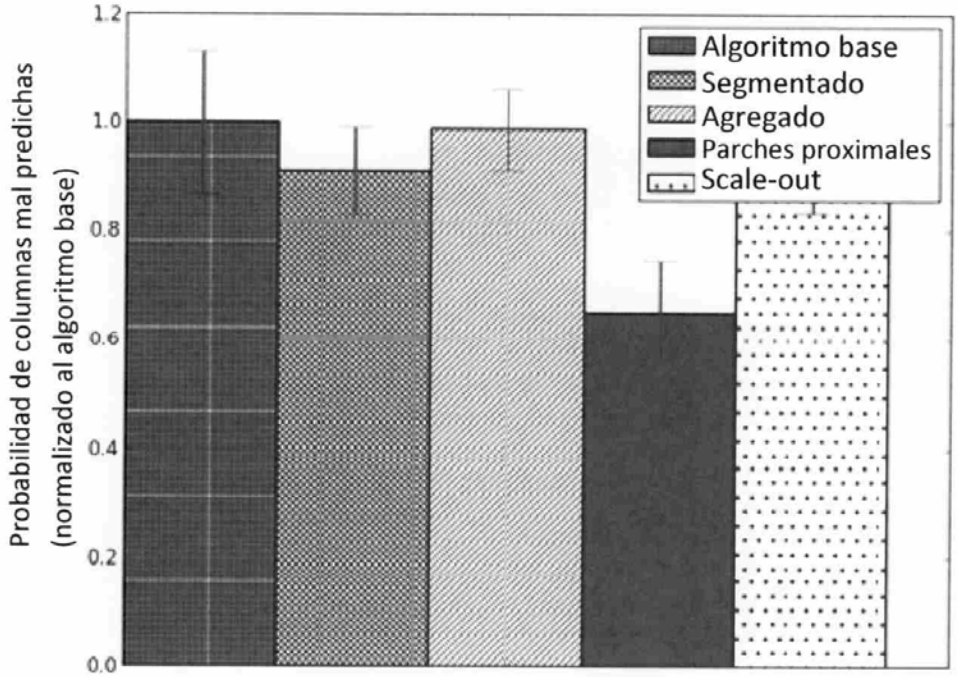
Probabilidad de columnas mal predichas
(normalizado al algoritmo base)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Algoritmo base
Segmentado
Agregado
Parches proximales
Scale-out

Figura 16

OFICINA ESPAÑOLA
DE PATENTES Y MARCAS

ESPAÑA

(21) N.º solicitud: 201500841

(22) Fecha de presentación de la solicitud: 20.11.2015

(32) Fecha de prioridad:

INFORME SOBRE EL ESTADO DE LA TECNICA

(51) **Int. CI. :** ***G06N3/00*** (2006.01)

DOCUMENTOS RELEVANTES

Categoría	**(56) Documentos citados**	**Reivindicaciones afectadas**
A	US 2013054495 A1 (HAWKINS JEFFREY C et al.) 28.02.2013, Descripción: párrafos 118-121,123-127.	1-14
A	US 2012078827 A1 (NUGENT ALEX) 29.03.2012, todo el documento.	1-14
A	NERE ANDREW et al. Simulating cortical networks on heterogeneous multi-GPU systems. JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING, 20120221 ELSEVIER, AMSTERDAM, NL 21.02.2012 VOL: 73 No: 7 Págs: 953-971 ISSN 0743-7315: doi:10.1016/j.jpdc.2012.02.006 McIntosh-Smith Simon; Gillan Charles; Sanna Nico; Scott Stan; Steinke Thomas. Todo el documento.	1-14
A	US 2011225108 A1 (HAWKINS JEFFREY C et al.) 15.09.2011, todo el documento.	1-14
A	PADILLA DANIEL E et al. Performance of a hierarchical temporal memory network in noisy sequence learning.2013 IEEE International Conference on Computational Intelligence and Cybernetics (CYBERNETICSCOM), 20131203 IEEE 03.12.2013 VOL: Págs: 45-51: doi:10.1109/CyberneticsCom.2013.6865779. Todo el documento.	1-14

Categoría de los documentos citados
X: de particular relevancia
Y: de particular relevancia combinado con otro/s de la misma categoría
A: refleja el estado de la técnica
O: referido a divulgación no escrita
P: publicado entre la fecha de prioridad y la de presentación de la solicitud
E: documento anterior, pero publicado después de la fecha de presentación de la solicitud

El presente informe ha sido realizado

☒ para todas las reivindicaciones ☐ para las reivindicaciones nº:

Fecha de realización del informe	**Examinador**	**Página**
01.02.2016	M. Muñoz Sánchez	1/4

Documentación mínima buscada (sistema de clasificación seguido de los símbolos de clasificación)

G06F, G06N

Bases de datos electrónicas consultadas durante la búsqueda (nombre de la base de datos y, si es posible, términos de búsqueda utilizados)

INVENES, EPODOC, WPI, XPIEE, XPI3E, NPL

Fecha de Realización de la Opinión Escrita: 01.02.2016

Declaración

Novedad (Art. 6.1 LP 11/1986)	Reivindicaciones	1-14	**SI**
	Reivindicaciones		**NO**
Actividad inventiva (Art. 8.1 LP11/1986)	Reivindicaciones	1-14	**SI**
	Reivindicaciones		**NO**

Se considera que la solicitud cumple con el requisito de aplicación industrial. Este requisito fue evaluado durante la fase de examen formal y técnico de la solicitud (Artículo 31.2 Ley 11/1986).

Base de la Opinión.-

La presente opinión se ha realizado sobre la base de la solicitud de patente tal y como se publica.

1. Documentos considerados.-

A continuación se relacionan los documentos pertenecientes al estado de la técnica tomados en consideración para la realización de esta opinión.

Documento	Número Publicación o Identificación	Fecha Publicación
D01	US 2013054495 A1 (HAWKINS JEFFREY C et al.)	28.02.2013
D02	US 2012078827 A1 (NUGENT ALEX)	29.03.2012
D03	NERE ANDREW et al. Simulating cortical networks on heterogeneous multi-GPU systems. JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING, 20120221 ELSEVIER, AMSTERDAM, NL 21.02.2012 VOL: 73 No: 7 Pags: 953 - 971 ISSN 0743-7315 doi:10.1016/j.jpdc.2012.02.006 McIntosh-Smith Simon; Gillan Charles; Sanna Nico; Scott Stan; Steinke Thomas. Todo el documento.	21.02.2012
D04	US 2011225108 A1 (HAWKINS JEFFREY C et al.)	15.09.2011
D05	PADILLA DANIEL E et al. Performance of a hierarchical temporal memory network in noisy sequence learning.2013 IEEE International Conference on Computational Intelligence and Cybernetics (CYBERNETICSCOM), 20131203 IEEE 03.12.2013 VOL: Págs: 45-51 doi:10.1109/CyberneticsCom.2013.6865779. Todo el documento.	03.12.2013

2. Declaración motivada según los artículos 29.6 y 29.7 del Reglamento de ejecución de la Ley 11/1986, de 20 de marzo, de Patentes sobre la novedad y la actividad inventiva; citas y explicaciones en apoyo de esta declaración

Se considera D01 el documento más próximo del estado de la técnica al objeto de la solicitud.

Reivindicaciones independientes

Reivindicación 1: El documento D01, divulga un método y un sistema de memoria temporal y espacial con un codificador de representación distribuida dispersa (párs. 118-121). El documento D01 también incluye módulos de memoria, módulo clasificador y módulo de cálculo y esboza el funcionamiento de un sistema de aprendizaje cortical (párs. 123-127).
El documento D01 no recoge ni los módulos columnados ni las características *multicast* del codificador y del encaminador en su relación con dichos módulos columnados y con los módulos de memoria que constituyen una implementación alternativa útil y de mayor factibilidad técnico-económica. Por tanto, el problema técnico objetivo consistiría en cómo concretar o desarrollar la implementación del sistema.
El documento D02 por su parte implementa un sistema de memoria temporal jerárquica utilizando encaminadores pero sin ninguna referencia a módulos columnados ni al procesamiento basado en aprendizaje cortical.
El documento D03 por su parte presenta una simulación de aprendizaje cortical en un sistema *multi-GPU* definiendo un equivalente a los módulos columnados (hipercolumnas) pero no se usa la lógica de encaminadores.
Teniendo en cuenta que ninguno de los documentos D01, D02 o D03 divulga la lógica de encaminadores reivindicada y que esta característica no es obvia para el experto en la materia, debido a que su implementación no es inmediata, la reivindicación 1 tiene novedad según el art. 6.1 y actividad inventiva según el art. 8.1 de la ley 11/86 de patentes.

Reivindicación 11: En paralelo con el análisis de la reivindicación 1 cabe concluir que la reivindicación 11 también tiene novedad según el art. 6.1 y actividad inventiva según el art. 8.1 de la ley 11/86 de patentes.

Reivindicaciones dependientes

Reivindicación 2-10, 12-14: estas reivindicaciones igualmente poseen novedad según el art. 6.1 y actividad inventiva según el art. 8.1 de la ley 11/86 de patentes al depender de las reivindicaciones 1 y 11 por cumplir estas ambos requisitos de patentabilidad.