**Mamba: Modelo SSM como alternativa a los Transformers, nuevas posibilidades en IA.**

**Ana B. Bestard, Daniela E. Cruz, y Lianet Soler**

**Universidad de Oriente, Santiago de Cuba, Cuba**

**Resumen**

Mamba es un modelo innovador que aborda las limitaciones de Transformers en el manejo de secuencias largas. Desarrollado por Albert Gu y Tri Dao, Mamba implementa modelos de espacio de estados selectivos que logran rendimientos comparables a los de Transformers, pero con escalabilidad lineal en términos de longitud de secuencia. Esta eficiencia computacional y uso reducido de recursos durante la inferencia lo convierten en una opción prometedora para aplicaciones que requieren procesar datos extensos y complejos.

A pesar de las críticas iniciales, investigaciones posteriores han demostrado la efectividad de Mamba en diversas tareas, desde el procesamiento de lenguaje natural hasta el aprendizaje de representaciones visuales. Variantes como MoE-Mamba y Vision Mamba han abordado desafíos específicos, como la tokenización y el manejo de datos en bruto, resaltando la adaptabilidad del modelo a múltiples contextos.

Mamba se está configurando como uno de los grandes hallazgos en inteligencia artificial, situándose en el lugar de una alternativa efectiva y escalable a los Transformers para el modelado de secuencias.

**Introducción**

En los últimos años, específicamente desde 2017 hasta la actualidad, la inteligencia artificial (IA) ha ganado una atención sin precedentes, especialmente en el ámbito de la generación de contenido. La aparición de chatbots avanzados como Chat-GPT, Claude y otros sistemas de IA generativa ha revolucionado la forma en que interactuamos con la tecnología. Este auge se intensificó a partir de finales de 2022, cuando estos modelos comenzaron a integrarse en aplicaciones comerciales y cotidianas, facilitando una interacción más natural y efectiva entre humanos y máquinas.

Los modelos están basados en la arquitectura de los Transformers, introducida en 2017 por Vaswani et al. en su influyente artículo "Attention Is All You Need". Esta arquitectura ha dominado la industria de IA generativa gracias a la comprensión de contextos complejos y a la forma eficiente de manejar las secuencias de datos. Sin embargo, a medida que los modelos se han vuelto más grandes y complejos, también han surgido limitaciones significativas, especialmente en términos de eficiencia y escalabilidad.

Mamba se presenta como una solución a las limitaciones inherentes a los Transformers, particularmente en el manejo de largas secuencias. Los modelos secuenciales tradicionales, aunque efectivos, enfrentan desafíos en la escalabilidad y la complejidad computacional. Mamba, desarrollado por Albert Gu y Tri Dao, propone una nueva clase de modelos selectivos de espacio de estados que alcanzan el poder de modelado de Transformers mientras escalan linealmente con la longitud de la secuencia. Por lo tanto, esta innovación es muy crítica para aplicaciones que requieren el procesamiento de datos extensos y complejos.

El 1 de diciembre de 2023, los desarrolladores de Mamba (Albert Gu y Tri Dao) presentaron su trabajo, destacando cómo este modelo se aleja de los enfoques tradicionales al integrar modelos de espacio de estado selectivos. Mamba no solo mejora la eficiencia computacional, sino que también ofrece un rendimiento comparable al de los Transformers, pero con un uso significativamente menor de recursos durante la inferencia. Este modelo, que se apoda "Mamba" en referencia a la serpiente negra, simboliza su rapidez y agilidad en el procesamiento de datos.

A pesar de su potencial, el artículo original de Mamba fue rechazado inicialmente por Ethics Review, lo que generó cuestionamientos sobre su viabilidad. Sin embargo, investigaciones posteriores, como "MoE-Mamba: Effective Selective Space Models with Mixture of Experts" y "Vision Mamba: Efficient Visual Representation Learning with Bidirectional State Space Model", han demostrado la eficacia de Mamba en diversas aplicaciones, desde el procesamiento de lenguaje natural hasta la representación visual. Estas variantes no solo abordan problemas específicos, como la tokenización y el manejo de datos en bruto, sino que también resaltan la adaptabilidad de Mamba en múltiples contextos.

Mamba representa un avance significativo en el campo de la inteligencia artificial, especialmente en el modelado de secuencias. Al abordar las limitaciones de los Transformers y ofrecer una solución eficiente y escalable, Mamba se posiciona como una opción prometedora para futuros desarrollos en IA.

**Desarrollo**

Los modelos de secuencia son una clase de modelos de aprendizaje automático diseñados para tareas que involucran datos secuenciales, donde el orden de los elementos en la entrada es importante. Los datos secuenciales incluyen datos textuales, datos de series temporales, señales de audio, transmisiones de video o cualquier otro dato ordenado.[1]

**RNN Y TRANSFORMERS**

Antes, para analizar una secuencia se hacía con redes neuronales, donde se procesaba cada token y el output de un token se conectaba con el input del siguiente token y con el ultimo output tenías toda la secuencia procesada, a eso se le llamó Redes Neuronales Recurrentes (RNN) [2]**.**

Un token de IA es la unidad más pequeña en la que se puede dividir una palabra o frase cuando la procesa un Large Language Model (LLM). Los tokens representan palabras, signos de puntuación o subpalabras, que permiten a los modelos analizar e interpretar texto de manera eficiente y, posteriormente, generar contenido de una manera similar basada en unidades. Esto es similar a cómo una computadora convierte datos en ceros y unos para facilitar el procesamiento. Los tokens permiten que un modelo determine un patrón o relación dentro de palabras y frases para que puedan predecir términos futuros y responder en el contexto del mensaje. Cuando se ingresa un mensaje, la frase y las palabras son demasiado largas para que un chatbot las interprete tal como están.[3]

Los modelos toman el mensaje, convierten la entrada en una lista de tokens, procesan el mensaje y convierten los tokens predichos nuevamente en las palabras que vemos en la respuesta.

Lo que a nosotros nos podrían parecer dos palabras idénticas puede generarse en diferentes tokens dependiendo de cómo estén estructurados dentro del texto [4].

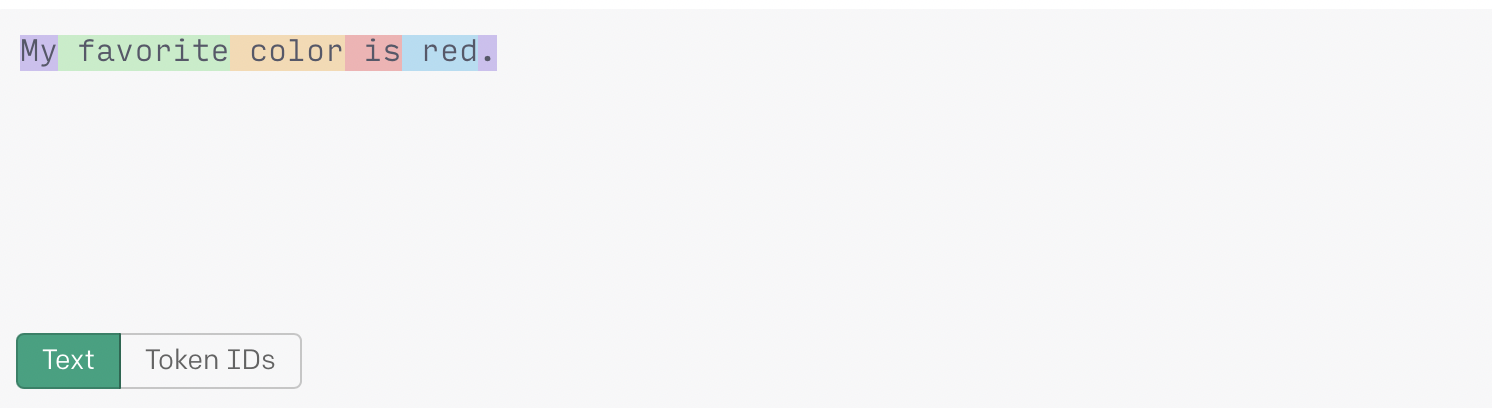
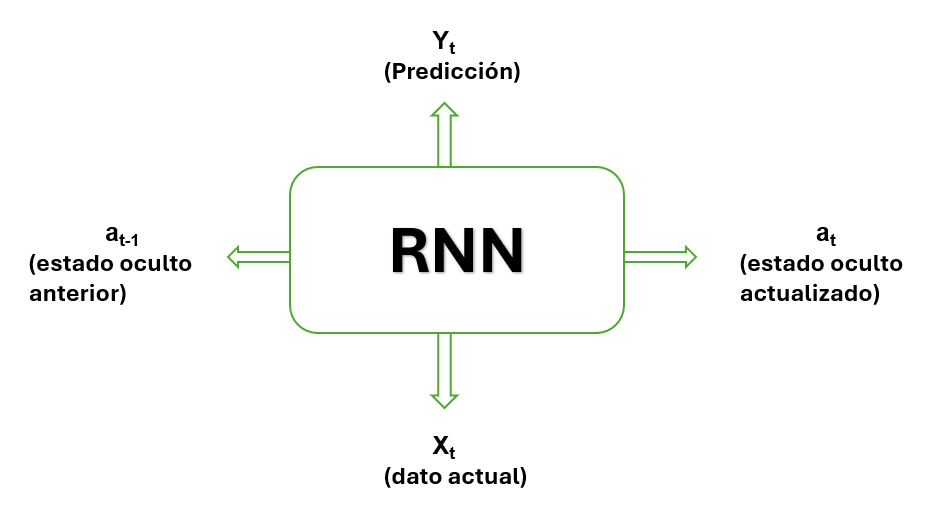


Imagen que contiene Aplicación

Descripción generada automáticamente *Figura 1. Representación del procesamiento de mensajes a tokens. Vector numérico.*

Las RNN se basan en una idea muy intuitiva, tal como en el cerebro humano se procesan las frases al leer, una a una en secuencia, apoyándose del contexto de las palabras anteriores, las redes neuronales lo hacen de la misma forma.

Una RNN básica (Figura 2) tiene dos entradas: el estado actual, y el estado oculto anterior, y proporciona dos salidas, la predicción y el valor actualizado del estado oculto. [5]

****

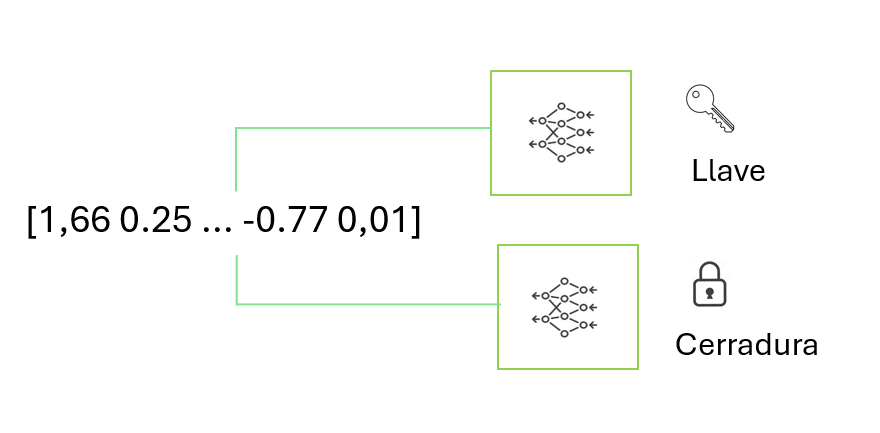
***Figura 2.*** *Red Neuronal Recurrente básica*

Las Redes Neuronales Recurrentes (RNN) tienen una limitación inherente a su capacidad para procesar información. Un RNN sólo es capaz de "considerar" el último estado actualizado junto con la entrada actual. Esta es una restricción bastante importante, ya que el próximo resultado se determina basándose únicamente en estos dos elementos. Aunque el proceso de generación de la salida se realiza mediante una transformación lineal, el cálculo computacional requerido para esta operación no es excesivamente elevado. Sin embargo, esta limitación en la capacidad de memoria de las RNN puede afectar su rendimiento en tareas que requieren la consideración de contextos más amplios o secuencias más largas.

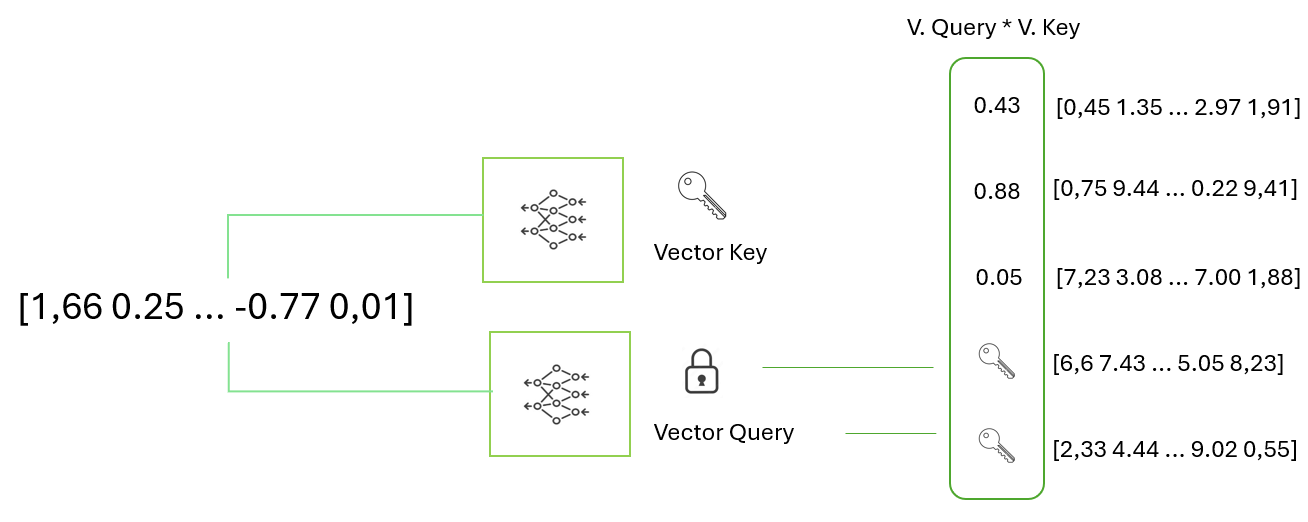
En la película Buscando a Nemo, el personaje Dory [6] sufre un trastorno de la memoria que le impide recordar información reciente o formar nuevos recuerdos. De manera análoga a las redes neuronales del presente o las redes neuronales recurrentes (RNN), Dory tiene dificultades para mantener la información a medida que avanza en su procesamiento. A medida que procesa nuevos datos, el peso que otorga a los primeros elementos disminuye, lo que le impide recordar adecuadamente la secuencia completa. Este fenómeno, conocido como “olvido gradual”, es una de las mayores limitantes para la implementación de la RNN. Tienen dificultades para retener información a largo plazo de la misma forma en que Dory olvida rápidamente los detalles más recientes. Al igual que Dory, las redes RNN no pueden mantener una memoria a corto plazo estable y coherente a través de secuencias cortas o largas. Para resolver este problema se llevaron a cabo los mecanismos de atención[[1]](#footnote-2).

Para comprender los mecanismos de atención se debe conocer que cada token está representado por un vector numérico (e.d., vector de embedding[[2]](#footnote-3)), para poder calcular las distancias entre los otros tokens. **(**Figura 1**)**

El problema de las RNN era la dificultad de analizar las relaciones que existen entre los tokens de entrada distanciados. Para solucionar esto se utilizan dos redes neuronales por token, una actúa como llave y otra como cerradura (Figura 3). Cada llave puede interactuar con el resto de las cerraduras de los otros tokens y funcionar con mejor o peor resultado [7].



***Figura 3.*** *Redes neuronales (Llave y cerradura) por token*)[8]**.**



***Figura 4.*** *Producto escalar a los valores de los vectores. Vector Atención*

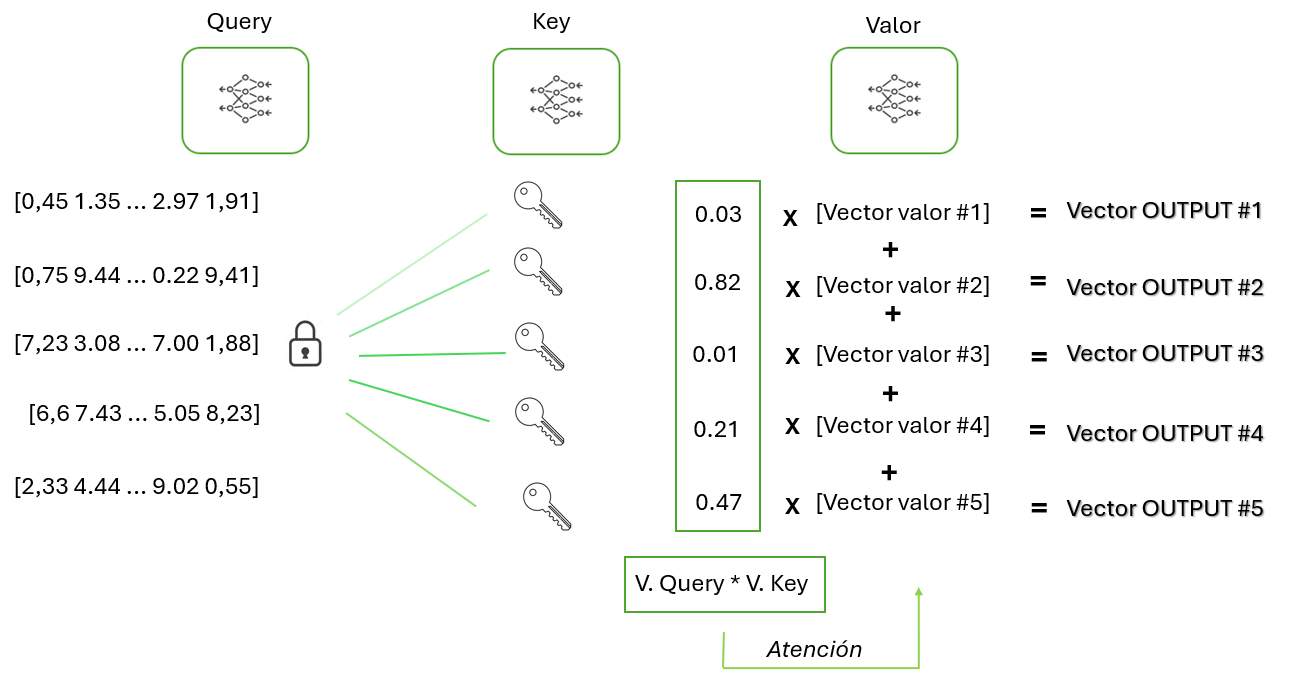
En el contexto de las Redes Neuronales Recurrentes, el mecanismo de atención se utiliza para calcular la compatibilidad de un token de consulta con todos los tokens en una secuencia de entrada. El método se basa en calcular el producto escalar del vector de consulta del modelo con los vectores clave. Cuanto mayor sea el valor resultante de este producto escalar, mayor será la compatibilidad entre los tokens [9].

Al realizar este cálculo para cada token de la secuencia, se obtiene una matriz de compatibilidad que representa las relaciones entre todos los tokens. A este vector resultante se le denomina Vector de Atención, ya que refleja el grado de atención que el modelo presta a cada token de la secuencia de entrada al generar la salida actual. (Figura 4)

Al asociar un vector de representación (incrustación) a cada token de entrada, el modelo puede estimar qué parte de la secuencia de tokens de entrada está recibiendo una mayor atención en el proceso de generación de la salida. Esto permite al modelo enfocar su atención en las partes más relevantes de la entrada para producir una salida más precisa y contextualmente adecuada.

El mecanismo de atención en las RNN representa un avance significativo en la capacidad de estas redes para capturar relaciones a largo plazo y procesar secuencias de manera más efectiva. Al ponderar la importancia relativa de cada token en la secuencia, el modelo puede concentrar sus recursos computacionales en las partes más informativas de la entrada, mejorando así su rendimiento en tareas que involucran el procesamiento de lenguaje natural y otras aplicaciones que requieren el análisis de secuencias. Si se realiza la misma operación para cada token de la secuencia, queda una matriz de atención para cada uno, lo que sirve para conocer la importancia que le da el modelo a cada token del resto de la secuencia. Al resultado del procesamiento de cada token se le conoce como vector valor.

Se aplica el algoritmo de atención[[3]](#footnote-4) sobre el vector valor, el resultado de este cálculo es un conjunto de vectores de tokens. Estos vectores se ordenan de mayor a menor, basándose en sus puntuaciones de atención otorgando mayor importancia a los tokens que han recibido más atención durante el cálculo para así poder identificar los elementos importantes en el contexto dado.



***Figura 5.*** *Vector Atención a todas las entradas.*

De esta forma se resuelve la falta de memoria utilizando el cálculo de atención (Figura 5). Attention is all you need [8] publicado en 2017 por [Ashish Vaswani](https://arxiv.org/search/cs?searchtype=author&query=Vaswani,+A), [Noam Shazeer](https://arxiv.org/search/cs?searchtype=author&query=Shazeer,+N), [Niki Parmar](https://arxiv.org/search/cs?searchtype=author&query=Parmar,+N), [Jakob Uszkoreit](https://arxiv.org/search/cs?searchtype=author&query=Uszkoreit,+J), [Llion Jones](https://arxiv.org/search/cs?searchtype=author&query=Jones,+L), [Aidan N. Gomez](https://arxiv.org/search/cs?searchtype=author&query=Gomez,+A+N), [Lukasz Kaiser](https://arxiv.org/search/cs?searchtype=author&query=Kaiser,+L) e [Illia Polosukhin](https://arxiv.org/search/cs?searchtype=author&query=Polosukhin,+I), demostró que no son necesarias las redes recurrentes, sino que con el cálculo de la atención y una arquitectura diferente se conseguían mejores resultados que al combinar las redes neuronales con los mecanismos de atención.

A ese nuevo tipo de red neuronal se le conoce como Transformers, los cuales utilizan mecanismos de autoatención[[4]](#footnote-5) para procesar secuencias en paralelo y capturar dependencias a largo alcance de manera eficiente.[10]

Los Transformers, al igual que las Redes Neuronales Recurrentes (RNN), enfrentaban el desafío de la falta de memoria a la hora de procesar secuencias de datos. Sin embargo, los Transformers abordan esta limitación mediante la implementación de mecanismos de autoatención. Estos mecanismos, si bien los hacen más efectivos en tareas de procesamiento de lenguaje natural, presentan una dificultad relacionada a la capa de autoatención.

Para calcular la salida de cada posición en la secuencia, los Transformers deben ponderar la importancia relativa de cada token en relación con los demás. Este proceso, conocido como autoatención, requiere que se calcule el peso de cada token con respecto a todos los demás tokens en la secuencia. A medida que la longitud de la secuencia aumenta, el número de cálculos necesarios crece exponencialmente.

Para entender mejor este concepto, consideremos un ejemplo numérico. Supongamos que estamos procesando una secuencia de texto que contiene 10 palabras (tokens). En este caso, para calcular la autoatención, el modelo debe evaluar las relaciones entre cada uno de los 10 tokens. Esto implica que el modelo realizará 10 x 10 = 100 cálculos para establecer las relaciones de atención entre todos los pares de tokens.

Si la longitud de la secuencia se duplica, pasando de 10 a 20 tokens, el número de cálculos necesarios para la autoatención aumenta a 20 x 20 = 400 cálculos. Este incremento significa que, al duplicar la longitud de la secuencia, el modelo no solo realiza el doble de cálculos, sino que el requerimiento de memoria y tiempo computacional se cuadruplica. En términos generales, el requerimiento de memoria para la autoatención es proporcional a L², donde L es la longitud de la secuencia.

Este aumento exponencial en los recursos computacionales necesarios puede tener un impacto considerable en el rendimiento del modelo. Aunque los Transformers son capaces de generar predicciones más precisas y efectivas en comparación con otros modelos lineales, el tiempo requerido para procesar secuencias largas puede ser significativamente mayor. Por lo tanto, al diseñar aplicaciones que utilizan Transformers, es crucial considerar este costo computacional, especialmente en contextos donde se manejan secuencias extensas.

**Modelos de Espacio de Estado (*State Space Models* (SSM))**

Modelos Transformers, como Chat-GPT, se vuelven exponencialmente más caros de entrenar y ejecutar mientras más grandes son, (como Meta que quiere adquirir una computadora de alrededor 40K solo para entrenar LLMs a finales de 2024, lo que es aproximadamente 10 millones en GPU) es justamente el mecanismo de atención lo que hace estos modelos más ineficientes de escalar [11].

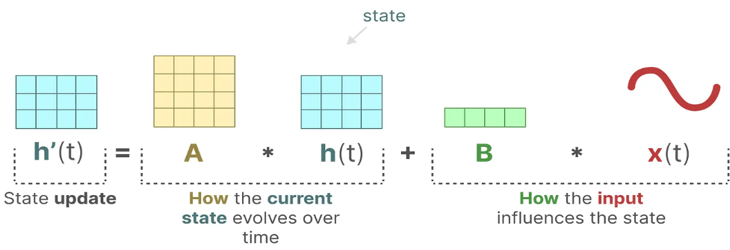
Para abordar el problema de la falta de memoria en los modelos de aprendizaje automático, los investigadores redescubrieron una arquitectura previamente establecida conocida como Modelos de Espacio de Estado (State Space Models, SSM). Esta arquitectura fue refinada y evolucionada para dar lugar a los Modelos S4 (Structured State Space Sequence Models), que constituyen el núcleo de la arquitectura de Mamba [12].

El desarrollo de los Modelos S4 representa un avance significativo en la capacidad de manejar secuencias de datos, permitiendo a los modelos superar las limitaciones de memoria que enfrentan las arquitecturas anteriores, como las Redes Neuronales Recurrentes (RNN). Al integrar los principios de los SSM con innovaciones en el diseño de modelos, los S4 brindan una solución más eficiente y efectiva para el procesamiento de secuencias, lo que mejora a su vez el rendimiento en variedad de tareas de aprendizaje automático.

Para entender en que se basa el modelo Mamba, se debe comprender que son los modelos SSM y los S4.

Los Modelos de Espacio de Estado (SSM) procesan la información de manera secuencial, manteniendo un estado interno que se actualiza con cada elemento de la secuencia. Estas arquitecturas son conceptualmente similares a las Redes Neuronales Recurrentes (RNN) lineales, donde la salida del procesamiento anterior se combina con la entrada del token actual. Sin embargo, a diferencia de las RNN, los SSM utilizan dos ecuaciones y cuatro conjuntos de matrices de parámetros:

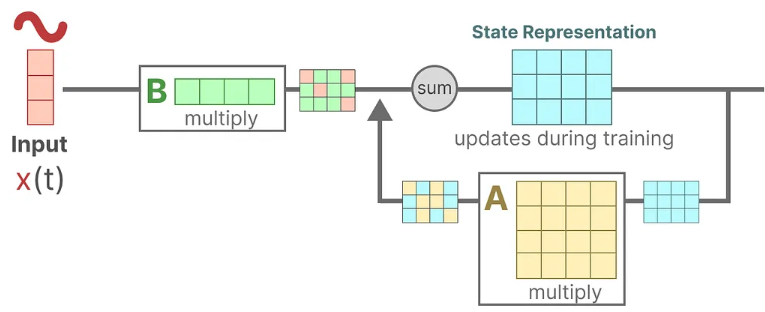
La ecuación de estado, que describe cómo cambia el estado (a través de la matriz A) en función de cómo la entrada influye en el estado (a través de la matriz B). (Figura 6)



***Figura 6****: Ecuación de estado.*

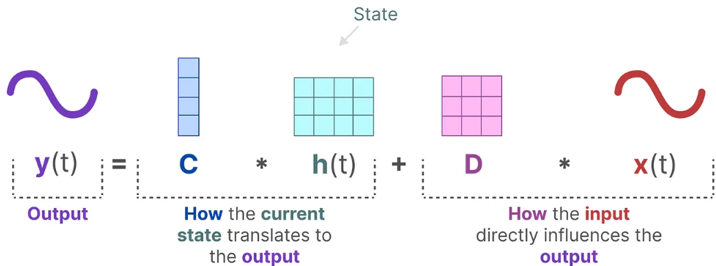
La **matriz A** es crucial para la dinámica del sistema, ya que describe cómo el estado anterior influye en el nuevo estado. En la ecuación de estado, se utiliza para multiplicar el estado latente anterior, lo que permite que el modelo retenga información sobre los estados pasados y capture las relaciones temporales entre ellos. Esto es esencial para construir un estado que refleje la historia completa del sistema. La matriz A se aplica antes de crear las representaciones de estado y se actualiza después de que se haya actualizado la representación de estado.

La **matriz B** se encarga de modelar cómo las entradas actuales afectan el estado del sistema (como influyen en él). Multiplicando la entrada actual por esta matriz, se incorpora la influencia de las nuevas entradas en la actualización del estado. Esto permite que el modelo adapte su comportamiento en función de las señales externas que recibe. (Figura 7)



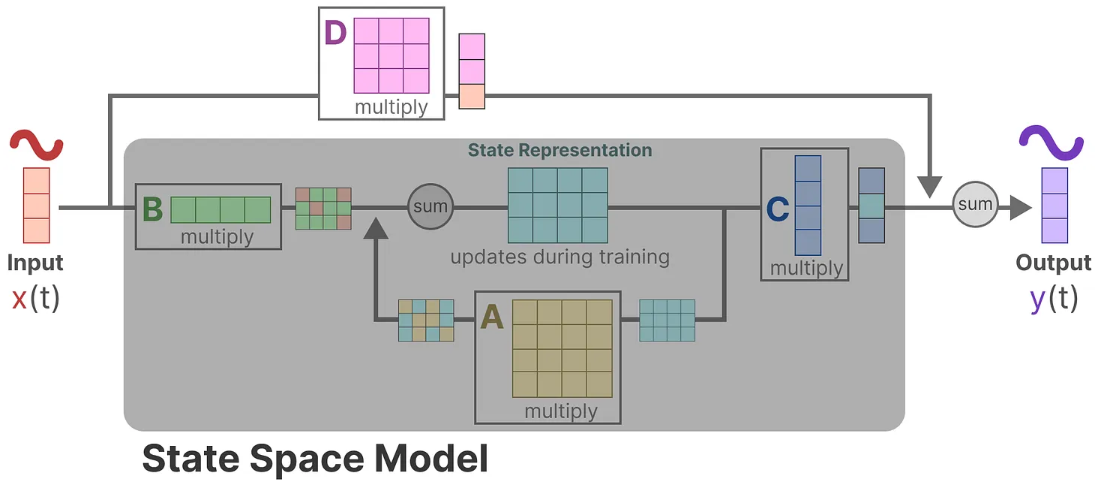
***Figura 7****: Arquitectura de los SSM (Matriz A y Matriz B).*

Y la ecuación de salida, que describe cómo se traduce el estado a la salida (a través de la matriz C) y cómo la entrada influye en la salida (a través de la matriz D). (Figura 8)

 ***Figura 8****: Ecuación de salida. [13]*

La **matriz C** se utiliza en la ecuación de salida y describe cómo el estado latente se traduce en la salida observable del modelo. Multiplicando el estado actualizado por la matriz C, se obtiene la salida del sistema, que puede ser una predicción o una acción por tomar.

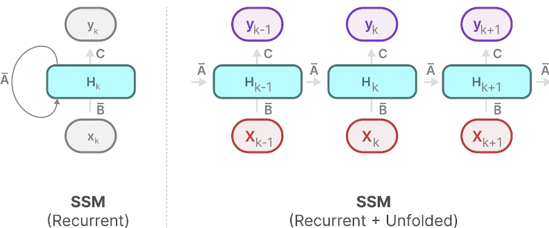
La **matriz D** proporciona una conexión directa entre la entrada y la salida, actuando como un "skip connection". Esto significa que permite que la entrada influya directamente en la salida sin pasar por el estado, lo que puede ser útil para capturar información relevante de manera más inmediata. (Figura 9) (Sección 1.1 y 1.2)



***Figura 9****: Arquitectura de los SSM. [13]*

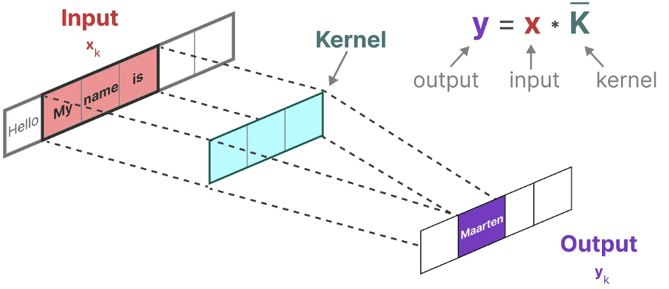
Posteriormente se realiza el proceso de discretización (sobre las matrices A y B), se realiza para convertir señales continuas en representaciones discretas, lo que es esencial para trabajar con datos como secuencias de texto. Este proceso implica la introducción de un parámetro de **tamaño de paso** o **delta (Δ)**, que define la duración de cada intervalo de tiempo en el que se mantiene el valor de la señal de entrada, este se retiene hasta que se recibe un nuevo valor en el siguiente intervalo de tiempo, lo que permite que el modelo procese entradas discretas de manera efectiva. Ahora tanto A como B representan parámetros discretizados del modelo.[13]

La representación recurrente en los Modelos de Espacio de Estados (SSM) permite formular el problema en pasos discretos en lugar de señales continuas, lo que facilita el procesamiento de secuencias. En esta representación, en cada paso de tiempo se calcula cómo la entrada actual, multiplicada por la matriz B, influye en el estado anterior, que se actualiza mediante la matriz A, y luego se predice la salida utilizando la matriz C. Este enfoque es similar al de las Redes Neuronales Recurrentes (RNN), donde el estado oculto se actualiza de manera eficiente, permitiendo que el modelo retenga información de pasos anteriores sin necesidad de recalcular toda la secuencia. Al descomponer la representación en pasos discretos, se logra una mayor eficiencia en el procesamiento y se facilita el aprendizaje de patrones temporales en los datos secuenciales. (Figura 10)



***Figura 10****: Representación Recurrente en los SSM.*

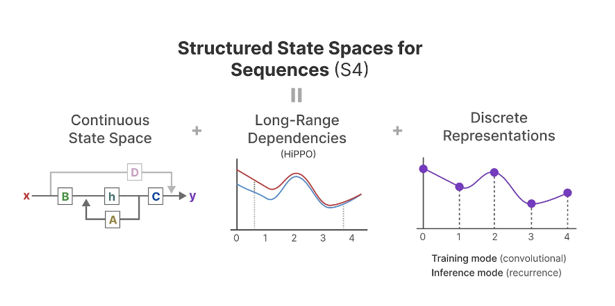
Los SSM también pueden representarse de manera convolucional, lo que permite aplicar filtros (kernels)[[5]](#footnote-6) a las secuencias de entrada para extraer características relevantes. A diferencia de las Redes Neuronales Recurrentes (RNN), donde el procesamiento se realiza de manera secuencial, la representación convolucional facilita el entrenamiento en paralelo, similar a las Redes Neuronales Convolucionales (CNN). Esto se debe a que los filtros se aplican independientemente a cada posición de la secuencia, lo que permite procesar múltiples pasos simultáneamente. Sin embargo, la representación convolucional tiene la limitación de que el tamaño del kernel es fijo, lo que implica que la capacidad de capturar dependencias a largo plazo está restringida por el tamaño del filtro. (Sección 1.3)



***Figura 10****: Representación Convolucional con Kernel en los SSM.*

Con estas representaciones se puede seleccionar la representación más adecuada según la tarea en cuestión. Durante la fase de entrenamiento, optamos por la representación convolucional, que permite un procesamiento en paralelo, mientras que, para la inferencia, utilizamos la representación recurrente, que es más eficiente.

Los S4 (Structured State Space for Sequences) son una evolución de los Modelos de Espacio de Estados (SSM) diseñados para abordar las limitaciones de estos últimos, especialmente en el manejo de secuencias largas y complejas. La principal diferencia entre los S4 y los SSM radica en su capacidad para manejar dependencias a largo plazo de manera más eficiente. Mientras que los SSM tradicionales pueden sufrir al intentar retener información relevante a lo largo de secuencias extensas, los S4 utilizan la técnica HiPPO (High-order Polynomial Projection Operators) para construir la matriz A, lo que les permite capturar mejor la memoria histórica y mejorar la retención de información (Figura 11).[13] Sin embargo, los S4 también enfrentan desafíos, como la necesidad de optimizar el rendimiento en tareas específicas de modelado de lenguaje, donde pueden no ser tan efectivos como las arquitecturas más recientes, como los Transformers.

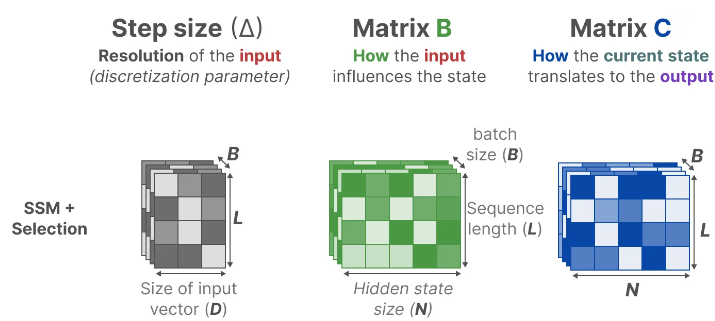


***Figura 11****: Características de los S4*

Los Selective SSM(S6) son modelos S4(Structured State Space for Sequences) con un mecanismo de selección y calculado con un escaneo.[12]

O sea, cuenta con un algoritmo de escaneo selectivo que permite al modelo filtrar información (ir)relevante, y un algoritmo que reconoce el hardware (Hardware aware) y permite el almacenamiento eficiente de resultados (intermedios) mediante escaneo paralelo, fusión de kernel y recálculo.[13] (Sección 2.3)

La característica principal (Figura 11) de los Selective SSM (S6) es que emplean capas lineales para calcular diferentes Delta B y C, (las matrices) para cada token de entrada. Obteniendo entonces diferentes matrices delta B y C para cada input el modelo puede aprender a enfocarse en algunos tokens más que en otros, que es básicamente el trabajo que hace la atención en los Transformers. (Sección 2.2)



***Figura 11****: Características de los S6*

Los desarrolladores de Mamba [12] encontraron una forma de computar de forma alternativa y en paralelo todo lo necesario para hacer al modelo más rápido manteniendo una arquitectura lineal. Presentan un truco para calcularlo rápidamente, basándose en un principio matemático llamado escaneo asociativo paralelo, (Parallel Asociative Scan) [13]. Se basa en la idea algorítmica de que, aunque algo se sienta inherentemente secuencial se pueden almacenar pasos intermedios para hacerlo más rápidamente.

Para llevar a cabo estos algoritmos eficientes de forma rápida se necesita una implementación de hardware específica para guardar y calcular las cosas en el tipo adecuado de GPU. Los desarrolladores de Mamba para hacer esto eficientemente, toman Delta, A, B, y C, desde la GPU HBM (la memoria lenta dentro de la GPU) a la GPU SRAM (la rápida). Hacen la discretización de A y B en la SRAM, y el Parallel asociative scan en la SRAM también, lo multiplican con la matriz C y ponen el resultado de vuelta a la HBM [14].

**Arquitectura de Mamba**

El modelo Mamba introduce una nueva arquitectura para el procesamiento de secuencias, que se centra en la eficiencia computacional y el manejo de secuencias largas. [15]

Mamba utiliza espacios de estados selectivos en lugar de la atención tradicional. Esto le permite recordar y olvidar selectivamente información, mejorando la eficiencia computacional. El modelo mantiene un estado de tamaño fijo que se actualiza paso a paso a medida que se procesan las entradas, lo que lo hace efectivo para manejar secuencias largas. Utiliza una función de activación no lineal para calcular el nuevo estado en cada paso y utiliza una capa de salida para generar predicciones basadas en el estado actual.

La arquitectura de Mamba se fundamenta en una red neuronal que optimiza la eficiencia computacional mediante el uso combinado de memoria en la GPU. A diferencia de los Transformers, que dependen en gran medida de mecanismos de atención y capas de preenfoque multicapa, Mamba emplea una estructura de red neuronal de extremo a extremo que no incorpora estos elementos. A pesar de esta diferencia, Mamba genera texto con bastante eficacia y logra una mayor eficiencia computacional que los modelos basados ​​en Transformers [16].

Mamba posee una arquitectura híbrida estructurada por capas de SSM intercaladas con capas de redes neuronales feed-forward, a diferencia de los Transformers que están estructurados por capas de atención alternadas con feed-forward.

Las capas SSM procesan la información secuencial y mantienen el estado interno. Capturan dependencias temporales o secuenciales en los datos, permiten el procesamiento eficiente de secuencias largas y actualizan un estado interno basado en la entrada actual y el estado anterior.

Las capas feed-forward transforman la representación de los datos. Consisten en neuronas completamente conectadas, aplican transformaciones no lineales a los datos y no tienen conexiones recurrentes o estado interno.

Típicamente, una capa SSM es seguida por una capa feed-forward debido a que las capas SSM capturan patrones secuenciales y las capas feed-forward procesan y transforman estas representaciones.

Cada tipo de capa aporta fortalezas diferentes permitiendo al modelo capturar tanto patrones secuenciales como transformaciones complejas. Contribuyendo a mantener la eficiencia computacional de los SSM mientras añade poder expresivo.

La entrada pasa primero por una capa SSM. La salida de la SSM se procesa a través de la capa feed-forward. Este proceso se repite varias veces a lo largo del modelo.

El número y tamaño de estas capas son hiperparámetros que pueden ajustarse. La configuración óptima puede variar según la tarea y los datos.

Imagine una línea de producción donde cada estación (capa SSM) procesa una parte específica del producto, y entre estaciones hay áreas de refinamiento (capas feed-forward) que ajustan y mejoran el trabajo realizado.

Esta estructura intercalada permite a Mamba combinar la eficiencia de procesamiento secuencial de los SSM con la capacidad de transformación no lineal de las redes feed-forward, resultando en un modelo poderoso y eficiente para tareas de procesamiento de secuencias.

Además, Mamba utiliza la memoria disponible en la GPU de forma inteligente haciendo uso de dos tipos diferentes: SRAM (memoria de acceso rápida y costosa) y HBM (memoria de alta velocidad y ancho de banda). La memoria SRAM se utiliza para almacenar el estado al que se accede con frecuencia, mientras que otros cálculos se realizan en HBM. Esta diferenciación en el uso de la memoria facilita un acceso más rápido a los datos críticos, lo que contribuye significativamente a la eficiencia computacional de Mamba.[16] (Sección 2.4)

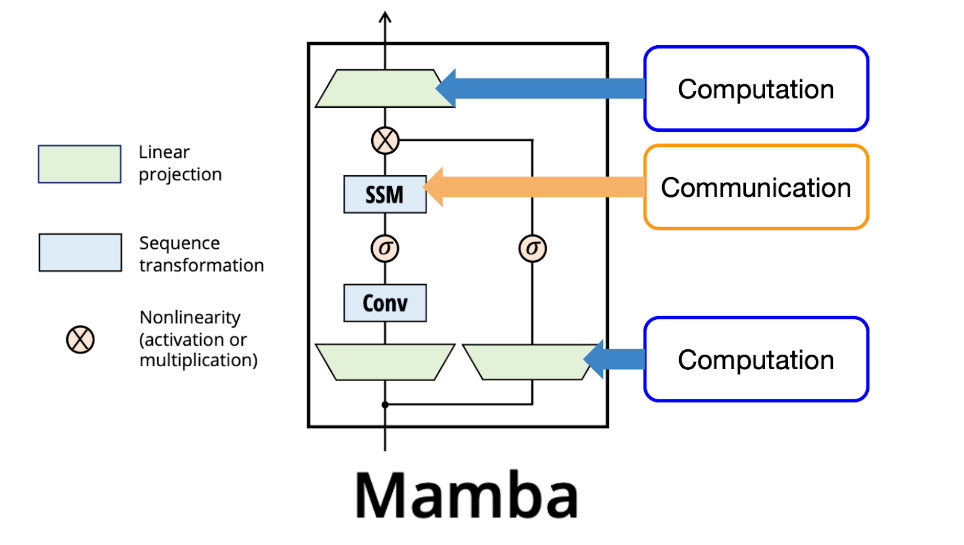
**Principales diferencias entre Mamba y los modelos de lenguaje tradicionales:**

La principal ventaja de Mamba sobre Transformers es la mejora de la eficiencia computacional, buscando reducir los requisitos de cómputo y memoria sin sacrificar el rendimiento. Mamba opera con una arquitectura de espacio de estados selectiva que permite el cálculo de la atención en tiempo lineal en lugar del tiempo cuadrático requerido por los Transformers. Esto significa que Mamba puede procesar entradas mucho más grandes sin perder eficiencia, convirtiéndose así en una alternativa más eficiente y escalable para aplicaciones de procesamiento de lenguaje natural.

Mamba fue diseñado para escalar eficientemente a modelos más grandes y tareas más compleja y es significativamente más eficiente en términos de complejidad computacional, especialmente al manejar secuencias largas como análisis de series de tiempo o procesamiento de texto extenso, mientras que los LLM tradicionales pueden tener limitaciones en la longitud de las secuencias.

Mamba generalmente se entrena más rápido que los LLM tradicionales debido a su arquitectura simplificada.

La arquitectura del modelo Mamba (Figura 12) se basa en un diseño simplificado que combina el bloque H3[[6]](#footnote-7), fundamental en la mayoría de las arquitecturas de Modelos de Espacio de Estados (SSM), con el bloque MLP (perceptrón multicapa) común en las redes neuronales modernas. En lugar de intercalar estos dos bloques, se repite homogéneamente el bloque Mamba. A diferencia del bloque H3, Mamba reemplaza la primera puerta multiplicativa con una función de activación, y en comparación con el bloque MLP, añade un SSM a la rama principal. Para la función de activación, se utiliza la activación SiLU / Swish, lo que permite que el modelo mantenga un enfoque eficiente y efectivo en el procesamiento de secuencias.[18] (Sección 2.1)



*Figura 12: Arquitectura Mamba* [17]

La arquitectura de los Transformers se parece más a la de los modelos RNN y LSTM (Long-Short Term Memory Models. Tipo especial de RNN que es capaz de recordar un dato relevante en la secuencia y de preservarlo por varios instantes de tiempo).

Los LSTM necesitan el output de su estado oculto anterior y el input global para generar la siguiente predicción, por lo que lo hace más lento debido a que cada capa necesita que su capa anterior acabe con su procesamiento para poder continuar.

Para los modelos de S4 que Mamba usa, cada estado oculto solo depende de la entrada global por lo que no hay pérdida de tiempo en esperar que la capa anterior genere el resultado. Lo que hace el cálculo mucho más rápido porque solo se necesitan las multiplicaciones de las matrices con la entrada global.

**Cálculo cuadrático de Transformers cambia a lineal.**

Lo que significa es que si alguna vez existiera un modelo Mamba como Chat-GPT 4 significaría que se obtendrían más cantidad de tokens por menos precio, una mínima de 1K al cuadrado de tokens por el precio de 1 centavo, lo que es mucho más barato que en la actualidad.

Las ventajas que Mamba mostró en el entrenamiento demuestran su posible superioridad frente a otros modelos secuenciales. No solo resuelve el problema de escalar que tienen los Transformers, sino que no está utilizando los mecanismos de atención y aun así puede recordar cualquier detalle que se le proporcione con total precisión.

La analogía entre la teoría detrás del modelo MAMBA y el teorema de muestreo de Nyquist-Shannon es interesante. Aunque parecen conceptos diferentes, hay una conexión profunda entre ellos.

El teorema de muestreo de Nyquist-Shannon establece que una señal continua en el tiempo puede ser reconstruida perfectamente a partir de sus muestras si la frecuencia de muestreo es mayor que dos veces la frecuencia más alta de la señal. En otras palabras, para capturar toda la información de una señal, se debe muestrear a una frecuencia lo suficientemente alta [18].

El modelo Mamba, por otro lado, utiliza un mecanismo de atención selectiva para enfocarse en partes específicas de la secuencia de entrada. La idea clave detrás de Mamba es que no todas las partes de la secuencia de entrada son igualmente importantes para hacer predicciones, y que seleccionar cuidadosamente las partes relevantes puede mejorar la precisión y la eficiencia.

Ahora, imagine que la secuencia de entrada es como una señal continua en el tiempo. En este sentido, el modelo Mamba puede ser visto como un muestreador de la secuencia de entrada, seleccionando cuidadosamente las partes más relevantes de la secuencia para hacer predicciones.

De la misma manera que el teorema de muestreo de Nyquist-Shannon establece que una frecuencia de muestreo lo suficientemente alta es necesaria para capturar toda la información de una señal, el modelo Mamba establece que una selección cuidadosa de las partes relevantes de la secuencia de entrada es necesaria para hacer predicciones precisas.

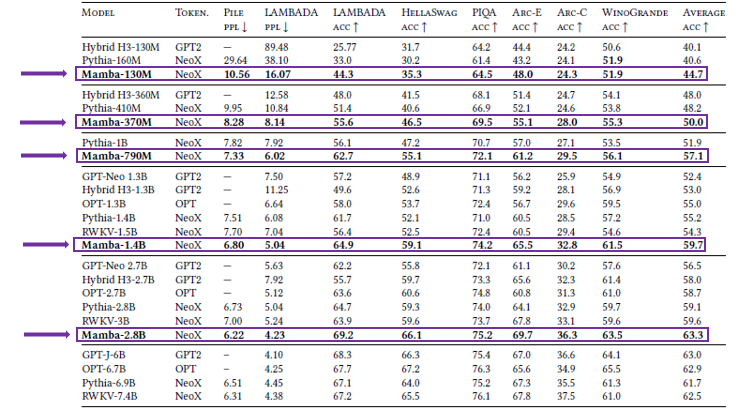
En otras palabras, el modelo Mamba puede ser visto como una aplicación del principio de muestreo de Nyquist-Shannon al dominio de las secuencias, donde la frecuencia de muestreo se reemplaza por la selección de las partes relevantes de la secuencia.

**Ventajas de la utilización de Mamba**

Mamba mantiene un rendimiento constante incluso con secuencias muy largas, comparado con Transformers también superando todos los demás modelos de SSM y los modelos que no usan atención, todo esto siendo muy rápida. O sea, es tan o más eficaz que ellos.

Superioridad en pruebas de referencia: Mamba ha demostrado un rendimiento superior en pruebas de referencia de modelado de lenguaje. En comparación con los Transformers, Mamba supera a modelos de igual tamaño tanto en el preentrenamiento como en la evaluación descendente.

Se comparó el rendimiento del modelo en una variedad de tareas de evaluación cero-shot[[7]](#footnote-8) con los de los modelos de código abierto más conocidos en tamaños similares, destacando especialmente a Pythia y RWKV, donde los resultados muestran que Mamba se destaca como el líder en su categoría para cada tamaño de modelo analizado y, en términos generales, logra resultados comparables a los modelos de referencia que son el doble de grandes.[18] (Figura 13)



*Figura 13: Resultados en Zero-shot Evaluations.*

Se demostró que los Transformers ante secuencias más largas toman más tiempo. De 14000 millones de parámetros, Mamba puede procesar 1814 tokens por segundo, mientras que Transformers bajo la misma cantidad de parámetros arrojó un error de falta de memoria.

Eficiencia computacional: Mamba utiliza una arquitectura de espacio de estados selectiva que permite un cálculo de atención en tiempo lineal, lo que mejora significativamente la eficiencia computacional en comparación con los Transformers.

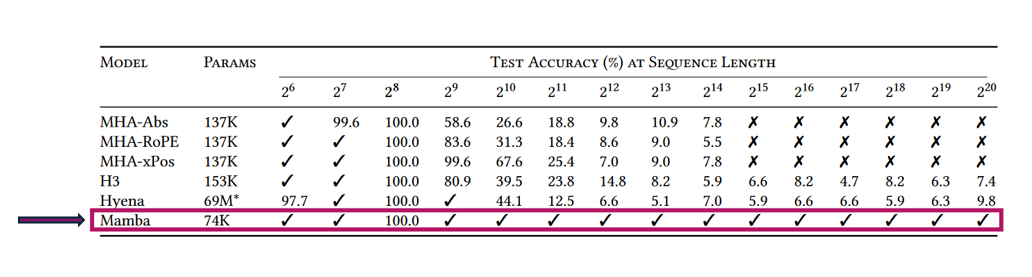
Manejo de entradas más grandes: Gracias a su eficiencia computacional mejorada, Mamba puede manejar entradas mucho más grandes sin perder rendimiento. Esto es especialmente útil para aplicaciones que requieren un contexto más extenso, como el procesamiento de documentos largos o secuencias genéticas.

Dentro de Las "tareas sintéticas" (synthetic tasks)[[8]](#footnote-9) que se evaluaron para Mamba, se encuentran los induction heads, que permiten que el modelo realice inferencias basadas en patrones previos en los datos que ha procesado, facilitando así su capacidad para completar o predecir secuencias de tokens de manera efectiva.

Los resultados de Mamba, específicamente de su capa de SSM selectiva, indican que este modelo tiene la capacidad de resolver tareas de manera perfecta al recordar selectivamente los tokens relevantes mientras ignora la información irrelevante en el medio. Mamba generaliza de manera efectiva a secuencias de longitud de un millón, lo que equivale a 4000 veces más que lo que se utilizó durante su entrenamiento, mientras que ningún otro método supera el doble de esa longitud. (Figura 14)

Los modelos fueron entrenados con una longitud de secuencia de 28 = 256 y se probaron en longitudes de secuencia que aumentan desde 26 = 64 hasta 220 = 1048576. Esto demuestra que Mamba no solo es eficiente en el manejo de secuencias largas, sino que también supera significativamente a otros enfoques en términos de escalabilidad y rendimiento en tareas de modelado de secuencias.[18]

Bajo estos parámetros, indica que el modelo puede aplicar lo que ha aprendido a nuevas situaciones con total precisión.

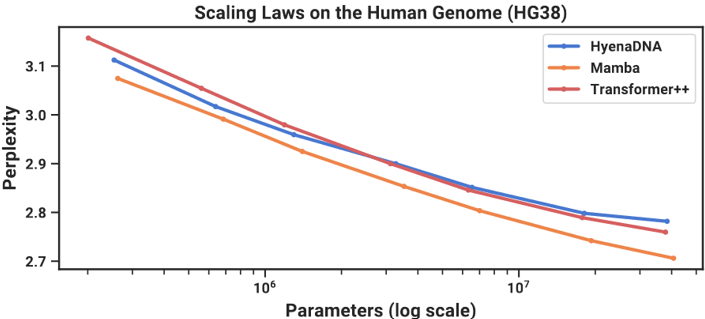


*Figura 14: Resultados en Induction heads. (✓exactitud de generalización perfecta, ✗ falta de memoria)*

Audio y Genómica: Mamba puede cubrir una gran diversidad de datos, funciona muy bien para clasificación de secuencias de ADN mostrando mejores resultados que los otros modelos última generación, como SaShiMi, Hyena y Transformers en el modelado de formas de onda de audio y secuencias de ADN, tanto en la calidad previa al entrenamiento como en métricas posteriores. En ambos entornos, su rendimiento mejora con secuencias de hasta un millón de longitud.

En el modelamiento de lenguaje Mamba es el primer modelo de secuencia lineal que alcanza el rendimiento de un Transformer tanto en perplexity[[9]](#footnote-10) previa al entrenamiento como en las evaluaciones posteriores. Con scaling laws de hasta 1B de parámetros, se demostró que Mamba supera el rendimiento de incluso modelos basados en Transformers. Tiene un rendimiento de generación 5 veces mayor que el de Transformers de tamaño similar, y la calidad de Mamba-3B coincide con la de Transformers que duplican su tamaño [17].

Mamba escala su rendimiento en comparación con otros modelos al aumentar tanto el tamaño del modelo como la longitud del contexto, utilizando el conjunto de datos del genoma humano (HG38). Muestra que, al fijar una longitud de contexto corta de 210=1024, y aumentar el tamaño del modelo desde aproximadamente 200,000 hasta 40 millones de parámetros, Mamba escala mejor que los modelos de referencia como HyenaDNA y Transformer++. Se evidencia que la perplexity previa al entrenamiento de Mamba mejora de manera continua a medida que aumenta el tamaño del modelo, lo que permite que Mamba iguale el rendimiento de los modelos Transformer++ y HyenaDNA utilizando entre 3 y 4 veces menos parámetros.(Figura15)[17].



*Figura 15: Resultados de DNA Scaling Laws (Tamaño del modelo)*

Por otro lado, al mantener fijo el tamaño del modelo y aumentar las longitudes de las secuencias, el mecanismo de selección de Mamba facilita un mejor rendimiento a medida que se incrementa la longitud del contexto. A diferencia de los modelos de referencia, que tienden a empeorar su rendimiento con longitudes de secuencia más largas, Mamba muestra una mejora en su perplexity de preentrenamiento a medida que se aumenta el contexto, alcanzando secuencias extremadamente largas de hasta un millón de tokens. Esto resalta la efectividad de Mamba en el manejo de contextos largos, en contraste con el modelo HyenaDNA, que presenta un rendimiento decreciente bajo las mismas condiciones. (Figura 16). [17]

Gráfico, Gráfico de líneas

Descripción generada automáticamente

*Figura 16: Resultados de DNA Scaling Laws (Longitud de la secuencia)*

Algunos desarrolladores han aplicado Mamba en campos como la Vision por computadora, que a diferencia del texto que tiene relaciones secuenciales, los datos visuales son más sensibles y necesitan la imagen completa para darle sentido.

Se publicó el artículo **Vision Mamba: Efficient Visual Representation Learning with Bidirectional State Space Model** [19]. Que propuso una arquitectura de visión genérica con Bloques Mamba en un modelo llamado VIM, donde, de acuerdo con una lista de imágenes de elección se obtuvo información posicional y se comprimió direccionalmente en los SSM, mostrando resultados prometedores en comparación con los Vision Transformers, porque aparte de las ventajas que ofrece la arquitectura Mamba, el modelo VIM es tres veces más pequeño que el vision Transformers, incluso con un rendimiento comparable [20].

Gracias a las propiedades de Mamba de analizar largos contextos, puede ser útil para analizar imágenes de alta resolución de satélites o microscopios [20].

Otra investigación de este mismo tipo **VMamba: Visual State Space Model** [21] , pero con un modelo distinto, llamado VMamba, muestra importantes mejoras de rendimiento en diferentes tareas: clasificación de imágenes, detección de objetos y segmentación.

Mixture of experts (MoE) [22] está siendo aplicado en Mamba, se supo a través del artículo **MoE-Mamba: Efficient Selective State Space Models with Mixture of Experts**[23], integrando SSM con MoE (que combina la capacidad de aprendizaje de múltiples modelos especializados, cada uno de los cuales se enfoca en una subtarea específica) para mejorar el modelado secuencial. Esta combinación tiene como objetivo aprovechar las fortalezas de ambos SSM, conocidos por su desempeño eficiente, y MoE, una técnica para ampliar modelos, y su combinación requiere menos pasos de entrenamiento que el modelo Mamba tradicional, manteniendo incluso la misma calidad.

**Propuesta para reducir el tamaño del modelo con MambaByte**

Del artículo Mamba: Linear-Time Sequence Modeling with Selective State Spaces [18] se conoce que es ideal para el modelado de ADN por su capacidad para procesar secuencias largas, entonces, ¿qué pasaría si en lugar de modelar ADN, se modelaran bytes? El problema de la tokenización[[10]](#footnote-11) que detenía a los Transformers se resolvería, el modelo será capaz de aprender de patrones de bytes, luego aprenderá a formar caracteres, y posteriormente palabras hasta llegar a textos.

Los Transformers toman el texto de entrada, lo tokenizan en piezas de palabras, y luego procesan esos tokens a través de la red neuronal.

Mamba aprende directamente de los bytes del texto en bruto, sin necesidad de tokenización; evitando los sesgos[[11]](#footnote-12) introducidos por el proceso de tokenización.

Un ejemplo generado por el modelo MambaByte que tiene casi mil millones de parámetros, fue capaz de continuar en el estilo del contexto que se le propuso en un texto y recodar de manera efectiva los nombres de los personajes ciento de bytes después, pero dado que es un modelo pequeño, la historia generada carecía de sentido en algunas partes, sin embargo, muestra resultados competitivos con los Transformers.

En el artículo **MambaByte: Token-free Selective State Space Model** [24], se menciona que uno de los inconvenientes potenciales de Mamba es lo que llaman Lost in The Middle, lo que significa que cuando la longitud del contexto es demasiado larga, se perdería cierta información debido a sus propiedades selectivas.

Sin el mecanismo de atención para verificar el contexto, Mamba tenía dificultades para igualar el mismo nivel de calidad de respuestas comparado con modelos existentes, especialmente con tareas relacionadas con retroceso debido a su mecanismo selectivo que descartaría información específica.

Otros artículos como MoE-Mamba: Effective Selective Space Models with Mixture of Experts, Vision Mamba: Efficient Visual Representation Learning with Bidirectional State Space Model y MambaByte: Token-free Selective State SpaceModel probaron lo contrario, donde comparados con los Transformers, Mamba es mejor para recordar bits específicos de información.

**Jamba**

Otro artículo publicado luego del nacimiento de Mamba fue Jamba: A Hybrid Transformer-Mamba Language Model [25] que propone un modelo que combina Mamba con Transformers para arreglar algunos de los inconvenientes de Mamba.

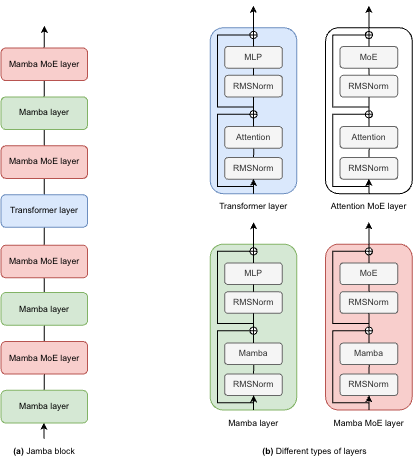
Jamba es un modelo de lenguaje de próxima generación en la unión de las arquitecturas SSM y Transformer, por lo tanto, un híbrido LLM. Esta es la primera implementación de producción a gran escala de Mamba, lo que abre muchas oportunidades interesantes para la investigación y la práctica. Aunque los resultados de esta fase inicial de experimentación son prometedores, anticipamos que se lograrán mejoras aún mayores a través de futuras optimizaciones y exploraciones.

Jamba propone atenuar los problemas fundamentales de los Transformers y resolver el problema de Mamba en cuando a los niveles de calidad de respuesta.

Jamba emplea el MoE y es un modelo con 12 mil millones de parámetros activos de 52 mil millones de parámetros totales en todos los expertos, lo que le permite manejar contextos más largos con una longitud de 256.000 caracteres, alrededor de 210 páginas de texto[26]. Es capaz de procesar hasta 140.000 tokens con sólo una GPU de 80 GB.[27]

La arquitectura de Jamba se basa en un diseño de capas y bloques que permite una integración efectiva de las arquitecturas Transformer y Mamba. Cada bloque de Jamba contiene una capa de atención o una capa de Mamba, seguida de un perceptrón multicapa [[12]](#footnote-13)(MLP), esto significa que hay una capa de Transformador por cada ocho capas en total.[27]

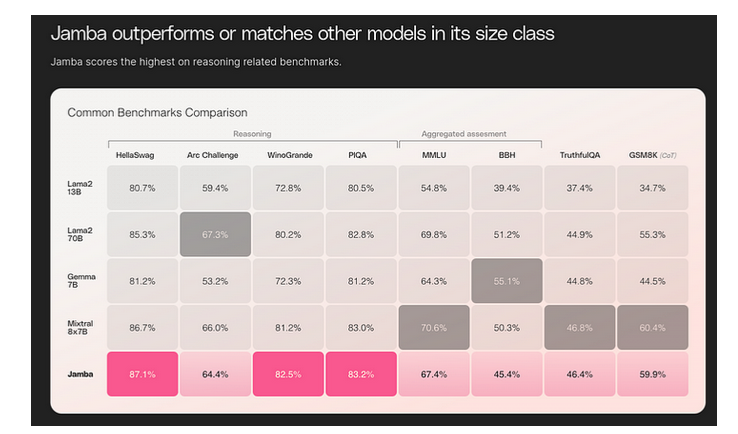
Un bloque Jamba estaría compuesto por estas capas intercaladas donde 1 de 8 capas (Figura 17 lado izquierdo) es una capa Transformer (aunque se demostró que si se ponen menos capas el resultado será el mismo) y el resto intercaladas es una capa Mamba y otra de MoE Mamba. Esta composición toma las ventajas de esos modelos y los combina. [22].



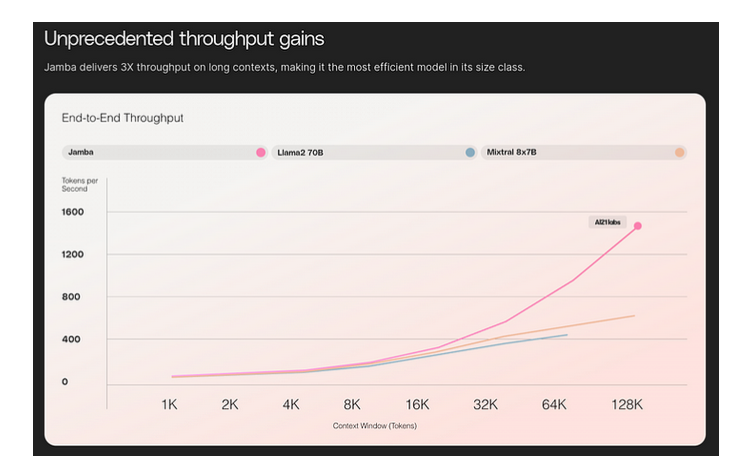
*Figura 17: Arquitectura del modelo Jamba: a) Bloque de Jamba b) Tipos de capas de Jamba*

Para contextualizar el impacto de Jamba en la evolución de los modelos Mamba, es importante señalar que antes de su lanzamiento, el modelo de Mamba más grande probado contenía aproximadamente 3 mil millones de parámetros. En contraste, Jamba ha demostrado ser capaz de igualar o superar el rendimiento de otros modelos populares en su categoría de tamaño en una amplia gama de puntos de referencia. La capacidad de Jamba para competir con modelos establecidos continúa demostrando su potencial para avanzar en el campo de la inteligencia artificial proporcionando soluciones más eficientes y efectivas para el procesamiento de datos. (Figura 18).

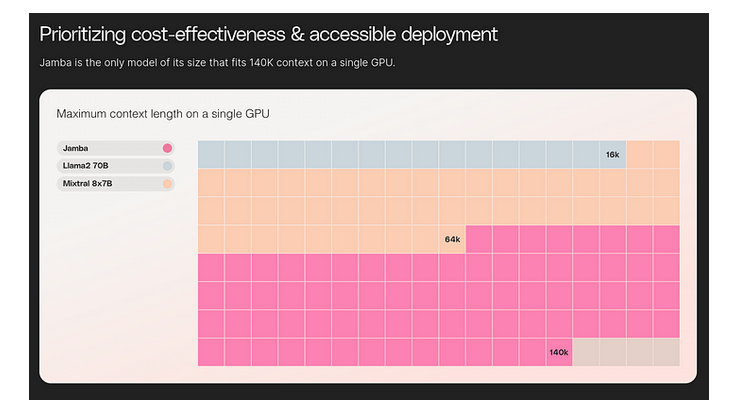
Ofrece un rendimiento tres veces mayor q Llama 2 13B, Llama2 70 B y Mixtral 8x7B[[13]](#footnote-14). (Figura 18) Esto favorece el desarrollo de diversas arquitecturas relacionadas con Mamba, especialmente debido a su naturaleza de open source[[14]](#footnote-15). [26]



*Figura 18: Resultados de Jamba en comparación con otros modelos* [26]



*Figura 19: Rendimiento de Jamba en procesamiento te tokens* [26]



*Figura 20: Comparación de la Longitud de contexto en GPU.* [26]

Gracias a las propiedades de Mamba, Jamba admite un de longitud de contexto hasta de 140k en una sola GPU de 80GB, con tres veces más rendimiento en comparación con los modelos basados en Transformers. (Figura 20). [26]

Aunque Mamba ofrece muchas ventajas en comparación con otros modelos de lenguaje, también enfrenta algunas limitaciones y desafíos. Algunos de estos incluyen:

Precisión de respuestas: Aunque Mamba tiene un rendimiento sólido en la generación de respuestas en tiempo real, no siempre puede garantizar respuestas 100% precisas. La calidad de las respuestas generadas puede depender de la calidad, la cantidad de los datos de entrenamiento y requisitos de precisión o restricciones de tiempo que sean necesarias para una predicción más acertada.

Capacidad de adaptación: Mamba puede necesitar una cantidad significativa de entrenamiento y ajuste para adaptarse a aplicaciones específicas. Esto puede requerir una inversión de tiempo y recursos para garantizar resultados óptimos.

Aunque requiere una inversión de tiempo y recursos para asegurar resultados óptimos, Mamba puede adaptarse a una serie de aplicaciones específicas:

Procesamiento de lenguaje natural: Mamba puede utilizarse para una variedad de tareas de procesamiento de lenguaje natural (PLN), incluyendo generación de respuestas en tiempo real, análisis de sentimientos y extracción de datos.

Clasificación de lenguaje natural y análisis de sentimientos: Mamba ha sido probado y evaluado en escenarios del mundo real, como clasificación de lenguaje natural y análisis de sentimientos, mostrando buenos resultados.

Modelado de secuencias en general: Mamba se presenta como un modelo de secuencia general que logra un rendimiento de vanguardia en varias modalidades, como lenguaje, audio y genómica.

Modelado de lenguaje: En modelado de lenguaje, modelos Mamba específicos como Mamba-3B superan a Transformers del mismo tamaño y coinciden con Transformers del doble de su tamaño, tanto en preentrenamiento como en evaluación posterior

**Mamba Chat**

En el artículo Descubre Mamba: ¡El Futuro de los Modelos de Lenguaje! [16] mencionan que Mamba Chat es una versión afinada de Mamba que ha sido entrenada para generar respuestas en tiempo real. La variante ya se ha equipado utilizando el conjunto de datos Ultra Chat con 16.000 ejemplos. Mamba Chat tiene varias ventajas para aplicaciones como la atención al cliente, en las que las consultas de los usuarios deben ser atendidas de forma correcta y rápida. Su capacidad para generar respuestas en tiempo real ha demostrado ser cinco veces más rápida que la de los modelos basados ​​en Transformers, lo que se traduce en una mayor eficiencia y escalabilidad en aplicaciones de procesamiento del lenguaje natural.(Sección 4.2 y 4.5)

Mamba Chat ha demostrado una ventaja de velocidad cinco veces mayor que los modelos básicos basados ​​en transformadores para la generación de respuestas en tiempo real. Equipado con su arquitectura de espacio de estado selectivo y eficiencia computacional mejorada, Mamba Chat puede ofrecer respuestas rápidas y precisas a las consultas de los clientes en una aplicación de atención al cliente. (Sección 4.3 y 4.4)

Lo que diferencia a Mamba Chat es la capacidad de ofrecer respuestas correctas al instante a todas las consultas de los clientes, algo especialmente relevante en el ámbito de la atención al cliente, donde cada momento cuenta. Mediante la implementación de Mamba Chat, las empresas pueden mejorar no sólo la eficiencia operativa sino también la satisfacción del cliente al brindar respuestas instantáneas y de calidad.[16]

La contribución mayor de Mamba es mejorar los SSM que ya eran más rápidos que los Transformers, usando menos cálculo computacional y por lo tanto menos memoria, lo que es más importante a la hora de modelar secuencias largas. Pero los SSM no superaban a los Transformers en ejecución, haciendo a estos últimos, aunque más caros, mucho más efectivos en la precisión de sus sentencias.

Mamba toma los Selective Space Models y los mejora, haciéndolos selectivos (Selective State Space Models) que son más eficientes que los Transformers.

Aunque Mamba se encuentra en sus primeras etapas de desarrollo, muestra un gran potencial para diversas aplicaciones en el ámbito del procesamiento del lenguaje natural, como:

Atención automatizada al cliente: Mamba Chat podría implementarse para proporcionar respuestas rápidas y precisas a las consultas de los clientes, mejorando la eficiencia operativa y la satisfacción del usuario.(sección 4.1)

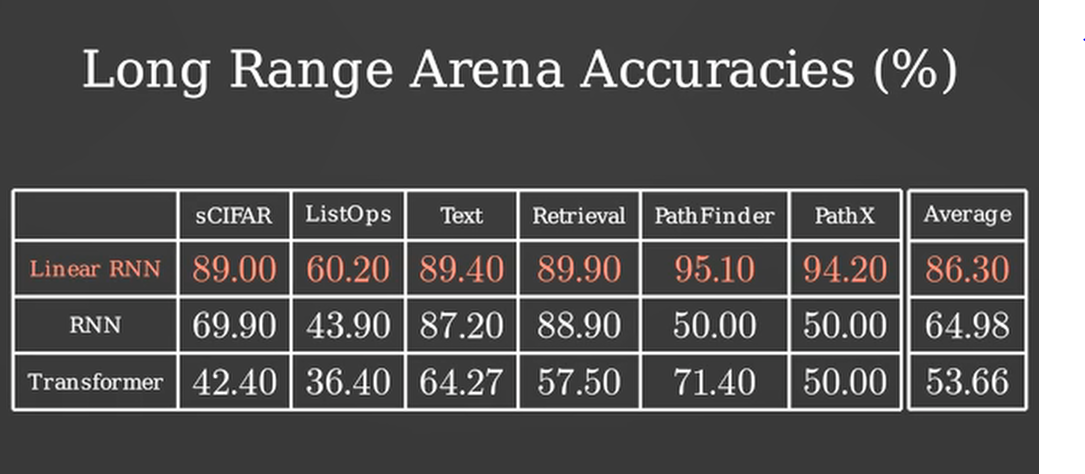
Análisis de sentimientos: Mamba podría usarse en el análisis de sentimientos, mediante el cual esto sería la identificación y comprensión de las emociones expresadas en un texto. Esto ayudaría cuando se quiera evaluar la percepción del público hacia determinados temas o productos.

Extracción de datos: con una eficiencia computacional mejorada, Mamba puede procesar entradas más largas y extraer fácilmente información valiosa de documentos largos o secuencias complejas, como en el análisis de datos genéticos.

**Rechazo inicial de Mamba**

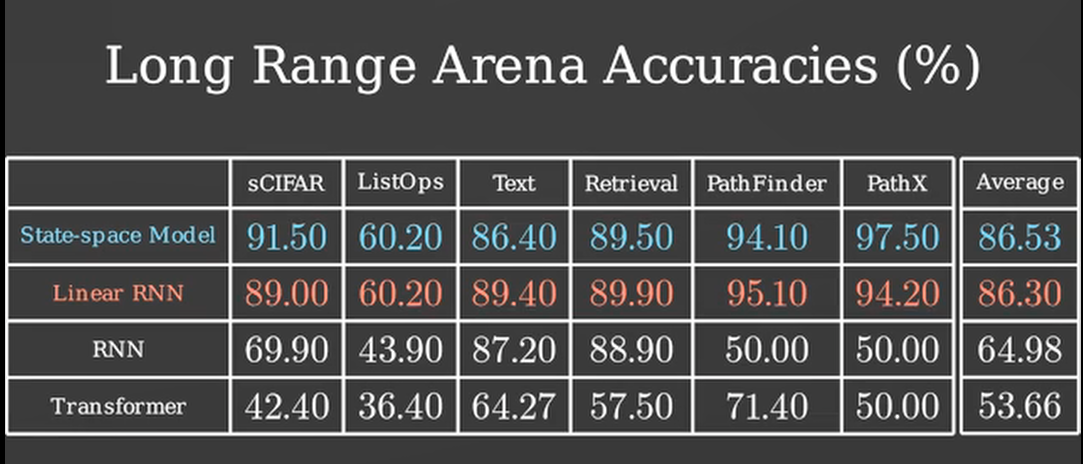
El artículo de Mamba, Mamba: Linear-Time Sequence Modeling with Selective State Spaces se presentó en la ICLR 2024 (una de las conferencias sobre aprendizaje automático más prestigiosas del mundo) y en enero fue rechazado por Ethics Review. Varios grupos diferentes han reimplementado Mamba y todos reprodujeron con éxito los resultados afirmados en el artículo, es decir, que Mamba funciona mejor que los transformadores y utiliza menos computación. Todos en la comunidad esperaban que el artículo Mamba fuera aceptado si no ganaban el premio al mejor artículo.[28]

Según Meta Review, Mamba no se probó en el Long Range Arena Bench Mark.



*Figura 21: Resultados de rendimiento de LM en el Long Range Arena. (Superioridad de RNN Lineal sobre Transformers)*

El Long Range Arena es una tarea completamente diferente al modelado de lenguaje y Mamba es específicamente un modelo de lenguaje. Los Transformadores tuvieron un rendimiento mucho peor que los RNN lineales en Long Range Arena (Figura 21), pero los transformadores siguen siendo un modelo de lenguaje mucho mejor, por lo que el rendimiento en Long Range Arena no es de ninguna manera indicativo de la capacidad de modelado de lenguaje, Mamba establece un nuevo estado del arte para el modelado de lenguaje, no debe rechazarse porque no resuelve también alguna otra tarea no relacionada. La única otra crítica en Meta Review es que Mamba solo fue evaluado en el modelado del lenguaje, es decir, la precisión al predecir la siguiente palabra en un texto.[28]



*Figura 22: Resultados de rendimiento de LM en el Long Range Arena. (SSM alcanzó una precisión similar a la del modelo de RNN lineal)*

Los desarrolladores de Mamba siguieron mejorando el modelo. Se publicaron resultados comparativos más sólidos, que muestran sus ventajas. A medida que aumentaba la demanda de modelos más eficientes, Mamba empezó a llamar la atención por su escalabilidad lineal y el hecho de que puede manejar secuencias largas se volvió más atractivo.

Investigadores influyentes en el campo comenzaron a reconocer el potencial de Mamba. Esto llevó a que más equipos experimentaran con la arquitectura y reportaran resultados positivos.

Los creadores de Mamba continuaron publicando y presentando en conferencias y foros. Esto aumentó la visibilidad y credibilidad del modelo en la comunidad de IA.

La disponibilidad de implementaciones de Mamba en frameworks populares facilitó la experimentación, lo que llevó a más validación independiente y a descubrir nuevas aplicaciones.

El modelo recibió atención en plataformas como Twitter y Reddit entre los profesionales de la IA. Los debates y discusiones públicas aumentaron el interés y la intriga en Mamba.

**Mamba-2**

# En State Space Duality (Mamba-2) Part I - The Model [29], Tri Dao explica que a pesar del potencial que Mamba había mostrado, no estaban completamente satisfechos con la primera versión.

Desde una perspectiva computacional, a pesar de los esfuerzos realizados para optimizar Mamba y hacerlo más rápido, especialmente con su implementación de escaneo selectivo con reconocimiento de hardware, sigue siendo considerablemente menos eficiente en términos de hardware en comparación con mecanismos como la Atención. Un factor clave en esta ineficiencia es que los aceleradores modernos, como las GPU y TPU, están altamente especializados para realizar multiplicaciones de matrices. Aunque esto no representa un problema significativo durante la inferencia, puede convertirse en un desafío considerable durante el proceso de entrenamiento, donde las consideraciones son bastante diferentes [29].

El punto principal del artículo de Mamba-2 es lo que se conoce como dualidad del espacio de estados estructurado (SSD) [29].

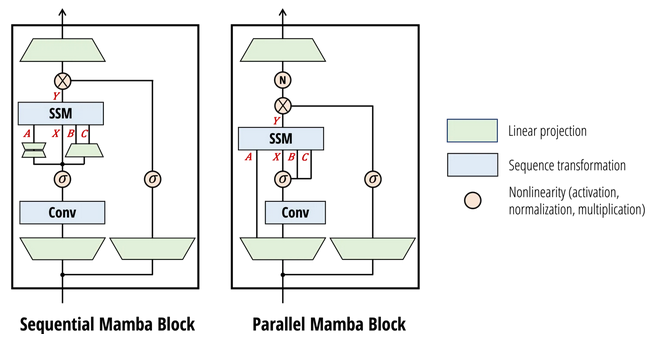
El modelo SSD se refiere a un componente específico e independiente, como una capa de atención o una capa de Modelo de Espacio de Estado (SSM), que puede integrarse en redes neuronales profundas. El marco SSD proporciona un enfoque general para analizar el modelo [29].

El algoritmo SSD es un método computacional diseñado para calcular las capas SSD de una manera mucho más eficiente en comparación con los SSM desarrollados anteriormente. Este algoritmo optimizado permite un procesamiento más rápido y una mayor eficiencia en la implementación de las capas SSD dentro de las arquitecturas de redes neuronales profundas [29].(Sección 3.)

**Arquitectura de Mamba-2**

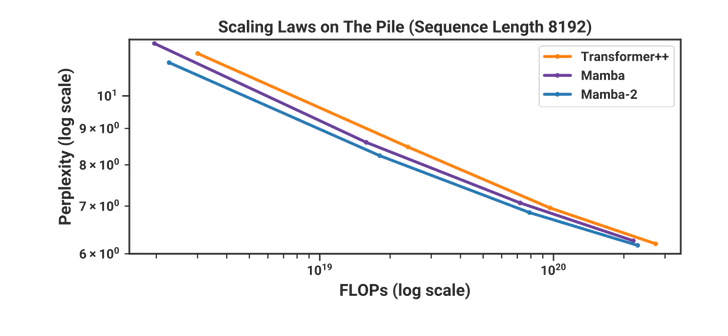
Aunque la contribución principal de Mamba-2 es la nueva capa y teoría SSD, también se realizaron algunos cambios en la arquitectura de la red neuronal de Mamba. (Figura 23)

El cambio principal es producir los parámetros SSM (A, B, C) en paralelo con la entrada X, en lugar de secuencialmente. Esto está motivado en parte por las conexiones con la atención; pero de manera más pragmática, es más simple y más adaptable a técnicas de escalado como el paralelismo tensorial [29]. (Sección 3)



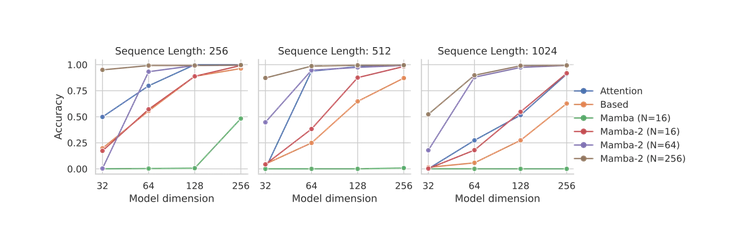
*Figura 23: Arquitectura Mamba-2*

Mamba-2 no fue tan probado como Mamba-1, pero los desarrolladores afirman que debe estar a la par o ser mejor que Mamba-2 en todos los aspectos. Los resultados del modelo de lenguaje completo siguen el mismo protocolo que Mamba y han mostrado una ligera mejora en las leyes de Chinchilla[[15]](#footnote-16). Los modelos que han sido completamente entrenados en el conjunto de datos Pile [[16]](#footnote-17)y evaluados bajo los estándares de disparo cero(cero-shot) presentan tendencias similares. A pesar de que el rendimiento es comparable, Mamba-2 es considerablemente más rápido de entrenar que Mamba-1 [29].(Figura 24)



*Figura 24: Resultados de Mamba-2 en Scaling Laws (leyes de escalado).*

Se destacó la única tarea sintética que ejecutaron, desde el artículo original de Mamba, que exploraba materiales sintéticos como la copia sintética y los cabezales de inducción (Induction heads). Se ejecutó una versión de esta tarea que es significativamente más difícil que la que suele informarse en la literatura y se descubrió que Mamba-2 supera parcialmente a Mamba-1. Una razón para el rendimiento mejorado es el tamaño del estado mucho mayor (hasta 16 veces mayor que Mamba-1 en este caso), que fue una de las principales motivaciones para desarrollar Mamba-2 en primer lugar. (Figura 25)

****

*Figura 25: Comparación de Rendimiento en tareas de recuperación de información (Recall Task) de Mamba-2 vs Mamba-1 y Transformer*

Curiosamente, Mamba-2 también parece ser notablemente mejor que Mamba-1 en esta tarea particular, incluso cuando se controla el tamaño del estado. Los desarrolladores admiten que no están seguros de la razón exacta y sugieren que sería fantástico eliminar los otros aspectos del modelo para investigar más a fondo, además plantean la posibilidad de que la estructura restringida de SSD pueda ser realmente útil en este contexto específico.

**Conjunto de Datos en Mamba**

Para maximizar el rendimiento y la aplicabilidad de Mamba, se han desarrollado y utilizado varios conjuntos de datos que permiten explorar sus capacidades en diferentes dominios. Es fundamental abordar la calidad de estos datos y su adecuación para tareas específicas, ya que son aspectos críticos para asegurar que Mamba pueda competir efectivamente con modelos establecidos.

El conjunto de datos Mamba Variants, disponible en Kaggle[30], ofrece una colección exhaustiva de modelos centrados en el modelado de secuencias, incluyendo diversas configuraciones del modelo Mamba. Es particularmente útil para investigadores que buscan experimentar con diferentes arquitecturas y optimizaciones del modelo. Las variantes de Mamba permiten evaluar cómo pequeñas modificaciones pueden impactar el rendimiento en tareas específicas, facilitando la investigación en el aprendizaje automático y el modelado de secuencias. Este conjunto de datos ha sido utilizado en estudios académicos y proyectos de investigación para probar la efectividad de Mamba en comparación con otros modelos.

Pretrained Mamba Model Classification on Intel Images también disponible en Kaggle[31], este conjunto de datos se centra en la clasificación de imágenes utilizando un modelo Mamba preentrenado. Permite a los usuarios evaluar el rendimiento del modelo en tareas de clasificación de imágenes dentro de conjuntos de datos de Intel. Este recurso es invaluable para aplicaciones prácticas en visión por computadora, donde Mamba puede ser utilizado para identificar y clasificar imágenes de manera eficiente. La capacidad de Mamba para procesar imágenes rápidamente lo convierte en una opción atractiva para empresas y desarrolladores que buscan implementar soluciones de clasificación de imágenes en tiempo real.

IMDB Movie Review Dataset [32]en un conjunto de datos utilizado en un proyecto de clasificación de texto con el modelo Mamba, contiene 50,000 reseñas de películas que se emplean para realizar análisis de sentimientos. La clasificación de reseñas como positivas o negativas demuestra la aplicabilidad de Mamba en tareas de procesamiento de lenguaje natural. Este conjunto ha sido utilizado en aplicaciones comerciales para mejorar la comprensión del sentimiento del cliente y en investigaciones académicas para explorar técnicas de análisis de texto.

OpenHermes Dataset [33]se utiliza para entrenar variantes del modelo Mamba, como el Mamba 2.8B. Contiene 242,000 entradas generadas principalmente por GPT-4 y se basa en datos abiertos de diversas fuentes. Su relevancia radica en su aplicación en el modelado de lenguaje y tareas de instrucción, permitiendo a los investigadores y desarrolladores entrenar modelos más robustos y versátiles. Este conjunto ha sido utilizado en aplicaciones de generación de texto y chatbots, donde se requiere un modelo que comprenda y genere lenguaje humano de manera efectiva.

**Presentaciones y Trabajos sobre Mamba en ICML 2024**

La Conferencia Internacional sobre Aprendizaje Automático (ICML, por sus siglas en inglés) es la principal reunión de profesionales dedicados al avance de la rama de la inteligencia artificial conocida como aprendizaje automático. Es reconocida mundialmente por presentar y publicar investigaciones de vanguardia sobre todos los aspectos del aprendizaje automático que se utilizan en áreas estrechamente relacionadas, como la inteligencia artificial, la estadística y la ciencia de datos, así como en áreas de aplicación importantes como la visión artificial, la biología computacional, el reconocimiento de voz y la robótica.[34]

En su edición 41, que tuvo lugar del 21 al 27 de julio del 2024 en Viena, Austria, se aceptaron alrededor de 2,600 investigaciones, entre las cuales contaron **Vision Mamba: Efficient Visual Representation Learning with Bidirectional State Space Model** [19]**, antes mencionada, y** Can Mamba Learn How to Learn? A Comparative Study on In-Context Learning Tasks[35], que presenta un modelo híbrido, llamado MambaFormer, que combina la arquitectura Mamba con bloques de atención. Una combinación que busca aprovechar las ventajas de ambos enfoques al integrar la eficiencia de Mamba en el procesamiento de secuencias largas con la flexibilidad y capacidad de atención de los transformadores, lo que permite un mejor manejo de las relaciones a largo plazo en los datos.

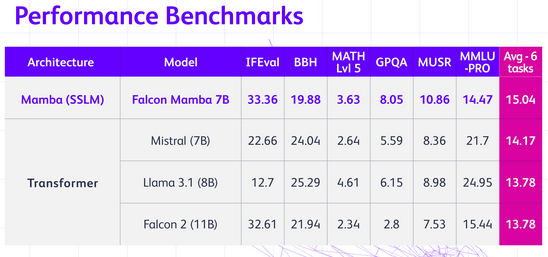
Es importante destacar que MambaFormer fue publicado previamente a Jamba, estableciendo una base sólida en la investigación de modelos de espacio de estados y su aplicación en el aprendizaje automático. Esta secuencia de publicaciones resalta la evolución y el desarrollo continuo en el campo, donde MambaFormer introdujo innovaciones significativas que posteriormente fueron integradas y ampliadas en el modelo Jamba.

A pesar de que el modelo Mamba fue rechazado en la conferencia ICLR 2024, ha encontrado aceptación y reconocimiento en la ICML 2024. La aceptación de Mamba sugiere que, a pesar de las críticas recibidas, el modelo tiene el potencial de contribuir significativamente al campo del aprendizaje automático y el procesamiento de lenguaje natural.

**Falcon Mamba 7B**

Falcon Mamba 7B es el primer modelo de lenguaje espacial estatal (SSLM) de código abierto, una nueva arquitectura revolucionaria para los modelos Falcon, que permite procesar bloques de texto más grandes; es otro ejemplo del enfoque pionero de Abu Dhabi en la investigación y el desarrollo de la IA.[36]

El Falcon Mamba 7B es el no. 1 lenguaje de modelo de espacio de estado (SSLM) de código abierto con rendimiento global en el mundo, según lo verifica de forma independiente Hugging Face. Supera a los modelos tradicionales de arquitectura de transformadores como el Llama 3.1 8B de Meta y el 7B de Mistral.[36]



*Figura 26: Comparación de Rendimiento entre Falcon Mamba 7B, Mistral (7B), Llama 3.1 (8B) y Falcon 2 (11B)*

Para los modelos de arquitectura de transformador, Falcon Mamba 7B supera al Llama 3.1 8B de Meta y al 7B de Mistral. Mientras tanto, para los otros SSLM, Falcon Mamba 7B supera a todos los demás modelos de código abierto en los puntos de referencia antiguos y será el primer modelo en la nueva tabla de clasificación de puntos de referencia más resistente de Hugging Face.[36]

Se probó la longitud de secuencia más grande que cabe en una sola GPU A10 de 24 GB. El tamaño del lote está fijo en 1 y se utilizó precisión float32. Los modelos basados ​​en transformadores utilizan un tamaño de vocabulario redimensionado para coincidir con el modelo Falcon Mamba. Se pueden ajustar secuencias más grandes que los modelos basados ​​en transformadores SoTA.[37]

**Mamba VS RNA**

Mientras que Mamba es un modelo de secuencia basado en modelos de estados selectivos (SSM) y utiliza una arquitectura que permite un procesamiento rápido y eficiente de secuencias largas. Las Redes Neuronales Artificiales pueden tener diversas arquitecturas como feedforward, recurrentes (RNN), o convolucionales (CNN) y pueden ser más lentas en secuencias largas.

Las RNN y LSTM pueden manejar secuencias, pero pueden tener dificultades con dependencias a largo plazo. A diferencia de Mamba que fue específicamente diseñado para manejar secuencias largas de manera eficiente y para capturar dependencias a largo plazo, podiendo obtener un rendimiento similar al de las RNA con el empleo de menos parámetros.

En el campo de la gestión energética, la predicción precisa del consumo de energía es fundamental para la planificación y optimización de recursos. Con la creciente demanda de energía y la necesidad de una gestión más eficiente, se requieren herramientas avanzadas que puedan procesar datos históricos y proporcionar pronósticos confiables. Esta temporada es del desarrollo de un predictor de consumo energético utilizando técnicas de aprendizaje profundo, implementando especificamente una versión simplificada del modelo Mamba. La herramienta busca dar predicciones precisas sobre el consumo futuro de energía, basadas en patrones históricos y factores temporales. It develops an interactive application in Python con Streamlit, Pandas, NumPy, y PyTorch. El modelo proporciona a los usuarios la funcionalidad de cargar archivos que contienen datos históricos de consumo de energía, entrenar un modelo predictivo basado en Mamba y generar predicciones futuras. Este enfoque utilizará la normalización de datos, entrenará modelos en tiempo real y visualizará los resultados para facilitar la interpretabilidad de las predicciones.

En el campo de la predicción de consumo energético, el uso de modelos de aprendizaje profundo ha demostrado ser una herramienta poderosa para capturar patrones complejos y realizar pronósticos precisos. En este contexto, dos arquitecturas de redes neuronales se destacan por su eficacia: las Redes Neuronales Recurrentes (RNN), específicamente las Redes de Memoria de Corto y Largo Plazo (LSTM), y las Redes Neuronales Convolucionales (CNN), como las implementadas en la biblioteca Mamba.

En este estudio, realizaremos una comparación entre dos códigos de predicción de consumo energético: MAMENERGY1, que utiliza la biblioteca Mamba basada en CNN, y UNE8, que emplea LSTM. Nuestro objetivo es evaluar el rendimiento de cada modelo en términos de precisión, eficiencia y capacidad de generalización, con el fin de determinar cuál es más adecuado para la predicción de consumo energético.

Las redes LSTM han demostrado ser especialmente efectivas en el modelado de series temporales debido a su capacidad para capturar dependencias a largo plazo. Por otro lado, las CNN han ganado popularidad en tareas de visión artificial, pero también han sido aplicadas con éxito en problemas de series temporales, como la predicción de consumo energético.

Mediante pruebas con diversos conjuntos de datos históricos, buscaremos comprender el comportamiento de los modelos en diferentes condiciones y su capacidad de generalización. Esto nos permitirá hacer recomendaciones informadas sobre cuál modelo es más adecuado para la predicción de consumo energético en función de los requisitos específicos del problema.

La comparación de estos dos códigos de predicción de consumo energético, basados en LSTM y CNN, respectivamente, contribuirá a la comprensión de las fortalezas y debilidades de cada arquitectura en este dominio. Los resultados de este estudio serán de gran utilidad para los investigadores y profesionales que trabajan en la predicción de consumo energético, al proporcionar una guía para seleccionar el modelo más apropiado para sus necesidades específicas.

Comparación entre MAMENERGY1 y UNE8:

• Arquitectura del modelo:

MAMENERGY1 utiliza una arquitectura SimpleMambaSSM, que es una implementación simplificada del modelo Mamba State Space Model.

UNE8 utiliza una arquitectura LSTM (Long Short-Term Memory), que es un tipo de red neuronal recurrente (RNN).

• Complejidad del modelo:

SimpleMambaSSM en MAMENERGY1 es más simple, con una capa RNN y una capa lineal.

El modelo LSTM en UNE8 es más complejo, con múltiples capas LSTM, capas de Dropout y capas Densas.

• Preparación de datos:

MAMENERGY1 utiliza solo dos características: día de la semana y mes del año.

UNE8 utiliza una secuencia de 7 días de datos históricos, incluyendo el consumo total de la empresa, día de la semana y mes del año.

• Entrenamiento:

MAMENERGY1 realiza el entrenamiento en tiempo real dentro de la aplicación Streamlit.

UNE8 entrena el modelo y lo guarda como un archivo .h5, permitiendo cargarlo posteriormente para predicciones.

• Escalabilidad:

MAMENERGY1 escala los datos de entrada y salida utilizando MinMaxScaler.

UNE8 también utiliza MinMaxScaler, pero aplica la escala a cada característica por separado en la secuencia de entrada.

• Interactividad:

MAMENERGY1 ofrece más opciones interactivas, como desactivar el entrenamiento y desentrenar el modelo.

UNE8 tiene una interfaz más simple con opciones para entrenar y predecir.

• Visualización:

Ambos códigos proporcionan visualizaciones similares de los resultados y la pérdida durante el entrenamiento.

• Eficiencia y efectividad:

- Eficiencia computacional:

El modelo SimpleMambaSSM en MAMENERGY1.py es probablemente más eficiente en términos de tiempo de entrenamiento y uso de recursos. Esto se debe a su arquitectura más simple y al uso de menos datos de entrada. La línea que demuestra esto es:

self.rnn = nn.RNN(input\_dim, hidden\_dim, batch\_first=True)

En comparación, el modelo LSTM en UNE8.py es más complejo y potencialmente más costoso computacionalmente:

LSTM(50, activation='relu', input\_shape=(X.shape[1], X.shape[2]), return\_sequences=True)

- Efectividad en la captura de patrones temporales:

El modelo LSTM en UNE8.py probablemente sea más efectivo en capturar patrones temporales complejos debido a su arquitectura más sofisticada y al uso de una secuencia de datos históricos. Esto se evidencia en la preparación de datos:

sequence\_length = 7

X, y = [], []

for i in range(len(df) - sequence\_length):

X.append(df[['TOTAL EMPRESA', 'DiaSemana', 'MesAno']].values[i:i+sequence\_length])

y.append(df['TOTAL EMPRESA'].values[i+sequence\_length])

• Flexibilidad y adaptabilidad:

MAMENERGY1 ofrece mayor flexibilidad al permitir el entrenamiento en tiempo real y la capacidad de desentrenar el modelo. Esto puede ser beneficioso para adaptarse a cambios rápidos en los patrones de consumo energético.

• Robustez del modelo:

UNE8 probablemente produce un modelo más robusto debido a su uso de Dropout y una arquitectura más profunda, lo que puede ayudar a prevenir el sobreajuste:

Dropout(0.2),

LSTM(50, activation='relu', return\_sequences=False),

Dropout(0.2),

| **Característica** | **MAMENERGY1 (Mamba)** | **UNE8 (LSTM)** |
| --- | --- | --- |
| Arquitectura | SimpleMambaSSM (más simple) | LSTM (más compleja) |
| Complejidad | Una capa RNN y una capa lineal | Múltiples capas LSTM, Dropout y Densas |
| Datos de entrada | Día de la semana y mes del año | Secuencia de 7 días de datos históricos |
| Entrenamiento | En tiempo real en Streamlit | Previo, guardado como archivo .h5 |
| Escalabilidad | MinMaxScaler para entrada y salida | MinMaxScaler para cada característica |
| Interactividad | Más opciones (desactivar/desentrenar) | Interfaz más simple |
| Eficiencia computacional | Probablemente mayor | Menor debido a la complejidad |
| Captura de patrones temporales | Posiblemente menos efectiva | Probablemente más efectiva |
| Flexibilidad | Mayor (entrenamiento en tiempo real) | Menor |
| Robustez | Posiblemente menor | Mayor (uso de Dropout) |

Medidas de evaluación para comparar los modelos:

1. Error Cuadrático Medio (MSE) y Raíz del Error Cuadrático Medio (RMSE): Estas medidas evalúan la diferencia entre los valores predichos y los valores reales. Un valor más bajo indica mejor rendimiento.
2. Error Absoluto Medio (MAE): Similar al MSE, pero menos sensible a valores atípicos. Útil para entender la magnitud promedio del error en las mismas unidades que los datos originales.
3. Coeficiente de Determinación (R²): Indica qué tan bien el modelo explica la variabilidad de los datos. Un valor más cercano a 1 indica un mejor ajuste.
4. Error Porcentual Absoluto Medio (MAPE): Expresa el error como un porcentaje, lo que facilita la interpretación y comparación entre diferentes escalas de datos.
5. Tiempo de entrenamiento y predicción: Mide la eficiencia computacional de cada modelo.
6. Uso de memoria: Evalúa la cantidad de recursos computacionales necesarios para cada modelo.
7. Capacidad de generalización: Se puede medir utilizando técnicas de validación cruzada para evaluar cómo se desempeña cada modelo con datos no vistos.
8. Estabilidad de las predicciones: Evalúa la consistencia de las predicciones a lo largo del tiempo y bajo diferentes condiciones.
9. Interpretabilidad: Aunque es más subjetiva, se puede evaluar la facilidad con la que se pueden interpretar las decisiones del modelo.
10. Adaptabilidad a cambios: Mide qué tan bien cada modelo se adapta a cambios repentinos en los patrones de consumo energético.
11. **MAMENERGY1 (Mamba):**

Error Cuadrático Medio (MSE) y Raíz del Error Cuadrático Medio (RMSE):

* El código utiliza MSE como función de pérdida (nn.MSELoss()).

Tiempo de entrenamiento y predicción:

* El entrenamiento se realiza en tiempo real en la aplicación Streamlit.
* Utiliza 100 épocas para el entrenamiento.

Uso de memoria:

* El modelo es relativamente simple (una capa RNN y una capa lineal), lo que sugiere un uso de memoria eficiente.

Capacidad de generalización:

* No se implementa validación cruzada, lo que podría limitar la evaluación de la generalización.

Estabilidad de las predicciones:

* Permite visualizar la predicción en el contexto de datos históricos.

Interpretabilidad:

* El modelo es relativamente simple, lo que podría facilitar su interpretación.

Adaptabilidad a cambios:

* Ofrece la opción de desentrenar el modelo, lo que permite adaptarse a nuevos patrones.

1. **UNE8 (LSTM):**

Error Cuadrático Medio (MSE):

* Utiliza MSE como función de pérdida.
* Muestra gráficas de pérdida de entrenamiento y validación.

Tiempo de entrenamiento y predicción:

* El entrenamiento se realiza por separado y el modelo se guarda.
* Utiliza 100 épocas para el entrenamiento.

Uso de memoria:

* El modelo es más complejo (múltiples capas LSTM y Dense), lo que sugiere un mayor uso de memoria.

Capacidad de generalización:

* Implementa división de datos en entrenamiento y prueba, lo que ayuda a evaluar la generalización.

Estabilidad de las predicciones:

* Permite visualizar la predicción en el contexto de datos históricos.

Interpretabilidad:

* El modelo es más complejo, lo que podría dificultar su interpretación.

Adaptabilidad a cambios:

* No ofrece opciones explícitas para adaptar el modelo a nuevos patrones sin reentrenamiento completo.

**Comparación:**

1. Precisión: Ambos modelos utilizan MSE, pero UNE8 proporciona una mejor evaluación al mostrar pérdidas de entrenamiento y validación.
2. Eficiencia computacional: MAMENERGY1 probablemente sea más eficiente debido a su arquitectura más simple.
3. Uso de memoria: MAMENERGY1 probablemente use menos memoria que UNE8.
4. Capacidad de generalización: UNE8 tiene una ventaja al implementar división de datos en entrenamiento y prueba.
5. Estabilidad de predicciones: Ambos ofrecen visualizaciones similares para evaluar la estabilidad.
6. Interpretabilidad: MAMENERGY1 podría ser más fácil de interpretar debido a su simplicidad.
7. Adaptabilidad: MAMENERGY1 ofrece más flexibilidad para adaptarse a cambios con la opción de desentrenar.
8. Robustez: UNE8 podría ser más robusto debido a su arquitectura más compleja y el uso de Dropout.

**¿Dónde encontrar los modelos de Mamba?**

* mamba-ssm: Esta es la implementación oficial de Mamba por parte de los autores originales. Está disponible en GitHub y proporciona la implementación base del modelo Mamba.
* PyTorch: Aunque PyTorch no tiene una implementación nativa de Mamba, se puede usar para construir modelos Mamba personalizados.
* Hugging Face Transformers: La biblioteca Transformers de Hugging Face está trabajando en la integración de modelos Mamba, aunque al momento de mi última actualización, esta integración aún estaba en desarrollo.
* MambaForge: Es un fork de Miniforge que incluye el paquete mamba-ssm preinstalado, facilitando el uso de Mamba en entornos de desarrollo de Python.
* FastMamba: Una implementación optimizada de Mamba diseñada para ser más rápida que la implementación original.
* xformers: Aunque principalmente conocida por sus implementaciones eficientes de transformadores, xformers está explorando la implementación de Mamba.
* torchmamba: Una implementación de Mamba en PyTorch que busca ser eficiente y fácil de usar.

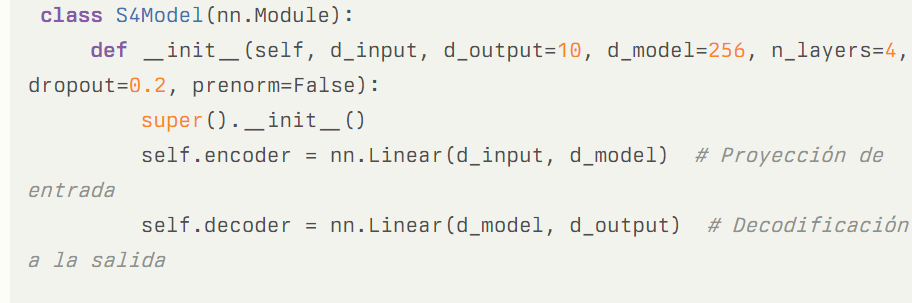
**Sección 1**

**Código como Reflejo de los Aspectos Clave de S4/SSM**

Ejemplo de scripts de entrenamiento para utilizar S4 de forma externa.

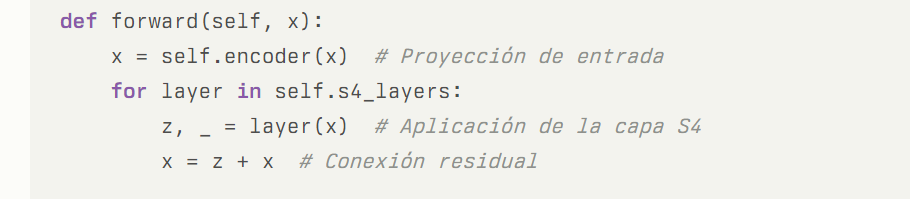
<https://github.com/state-spaces/s4/blob/main/example.py>

1.1 Ecuación de Estado y de Salida:



En el constructor de S4Model, self. encoder actúa como la matriz B que proyecta la entrada a un espacio de mayor dimensión (similar a cómo la entrada influye en el estado), mientras que self.decoder representa la matriz Cque traduce el estado a la salida.

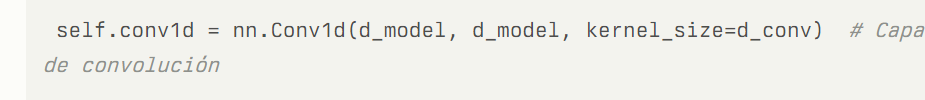
1.2 Actualización del Estado



En el método forward, la aplicación de layer(x) representa la actualización del estado utilizando la matriz A. La conexión residual x = z + x permite que el modelo mantenga información del estado anterior, reflejando cómo se actualiza el estado en un SSM.

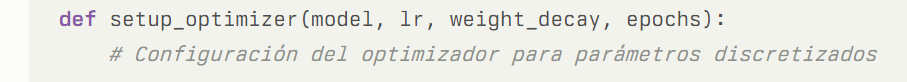
(Dado q este código es de entrenamiento):

1.3 Representación Convolucional



La inicialización de self. conv1d permite que el modelo aplique convoluciones a lo largo de la secuencia, capturando patrones locales y mejorando la eficiencia del procesamiento.

1.4 Discretización y Parámetros



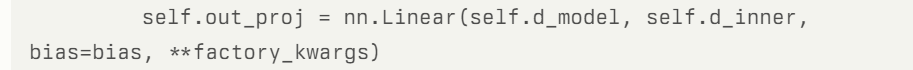
La configuración del optimizador puede incluir ajustes específicos para los parámetros discretizados del modelo, asegurando que el aprendizaje se realice de manera efectiva en el contexto de las SSM.

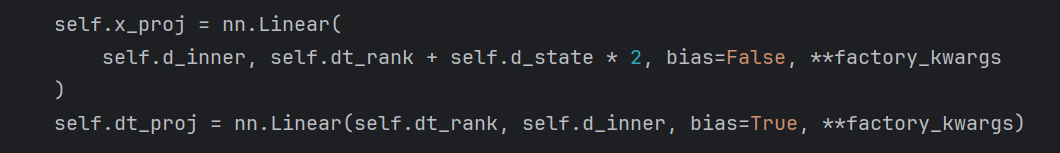
**Sección 2**

**Código como Reflejo de los Aspectos Clave de Mamba**

2.1 El bloque Mamba se inicializa en el método `\_\_init\_\_`, donde se configuran varios parámetros y capas que son fundamentales para su funcionamiento.

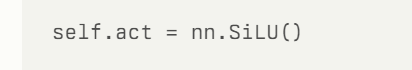






La arquitectura de Mamba combina bloques de espacio de estados (SSM) con bloques de perceptrón multicapa (MLP), lo que se refleja en cómo se organizan las capas en el código, en: la inicialización de `in\_proj`, `conv1d`, y `out\_proj`. En el código, la proyección de entradas y la proyección de tiempo discreto son ejemplos de cómo se utilizan estos bloques para manejar la información secuencial y realizar transformaciones no lineales. Esto permite al modelo capturar tanto patrones secuenciales como complejas transformaciones, optimizando así la eficiencia computacional.

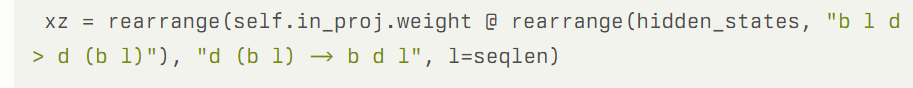
El bloque Mamba utiliza una función de activación SiLU (Swish) para procesar las entradas.



Mamba ha sido diseñado para ser eficiente en el manejo de secuencias largas, la elección de la función de activación SiLU permite que el modelo mantenga un enfoque eficiente y efectivo en el procesamiento de secuencias.

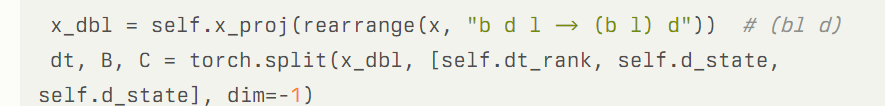
2.2 El método `forward` del bloque Mamba se encarga de procesar las entradas y aplicar las proyecciones necesarias.

El código muestra cómo se realiza la proyección inicial de las entradas utilizando self.in\_proj que permite calcular las matrices necesarias para el procesamiento:



Mamba utiliza capas lineales para calcular diferentes matrices Delta para cada token de entrada.

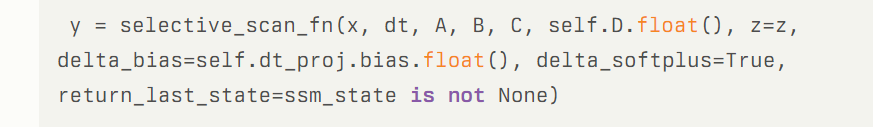
Se divide la salida de la proyección en las matrices `dt`, `B` y `C`, que son esenciales para el funcionamiento del modelo.



los Selective SSM capturan dependencias temporales en los datos. La separación de estas matrices es crucial para el cálculo del estado del modelo en cada paso. Esto permite que el modelo mantenga un estado interno que se actualiza en función de las entradas.

2.3 Escaneo Selectivo

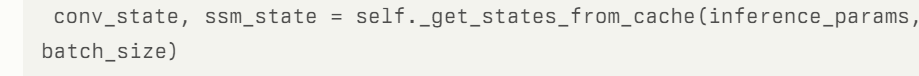
La función `selective\_scan\_fn` se invoca para realizar el escaneo selectivo, lo que permite que el modelo actualice su estado interno de manera eficiente.



Mecanismo de Selección: Este fragmento de código refleja directamente el mecanismo de selección donde el modelo puede filtrar información irrelevante y enfocarse en patrones significativos dentro de las secuencias. Esto es fundamental para la arquitectura Mamba, que se basa en el escaneo selectivo para mejorar la eficiencia.

2.4 Manejo de Memoria en GPU

La arquitectura Mamba también se beneficia del uso eficiente de la memoria, utilizando SRAM y HBM para optimizar el rendimiento.



Este fragmento de código ilustra cómo el modelo gestiona el estado en la memoria de manera eficiente, almacenando y accediendo a datos críticos en la memoria rápida.

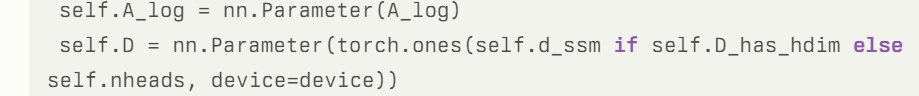
Se llama al método \_get\_states\_from\_cache, que se encarga de recuperar los estados de convolución (conv\_state) y el estado del modelo de espacio de estados selectivo (ssm\_state) desde un caché. Esto es crucial para mantener la continuidad en el procesamiento de secuencias largas.

**Sección 3**

**Código como Reflejo de los Aspectos Clave de Mamba2**

Inicialización de Parámetros

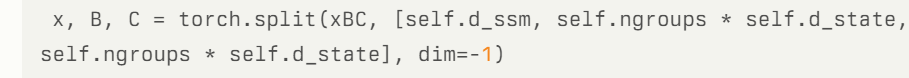
La clase `Mamba2` define varios parámetros en su constructor, incluyendo:



Estos parámetros son esenciales para la implementación de las capas SSD, donde A y D son fundamentales para el cálculo de los estados en el modelo SSD.

División de Parámetros

Los parámetros SSM se separan en el método `forward`:



Esta línea muestra cómo se extraen los parámetros B y C, permitiendo que se utilicen simultáneamente con X en el cálculo posterior.

Proceso de Entrada

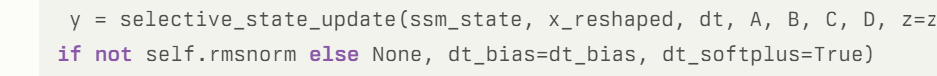
En el método `forward`, se realiza la proyección de entrada y el cálculo de los estados SSD:



Aquí, la entrada u es transformada en un espacio de características que incluye los parámetros necesarios para las capas SSD, lo que permite el procesamiento paralelo.

Cálculo de Estados

El método `step` implementa la actualización de los estados SSD:

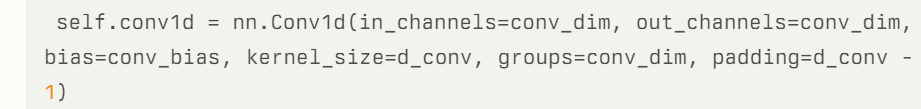


Este fragmento muestra cómo se aplican las actualizaciones de estado de manera eficiente, utilizando el algoritmo SSD para optimizar el cálculo de los estados A, B, y C.

Y la entrada X (representada como x\_reshaped), evidenciando claramente el procesamiento en paralelo.

Uso de Conv1D

El uso de convoluciones en el modelo se refleja en la definición de `conv1d`, que permite integrar la atención y el procesamiento de secuencias de manera paralela:



Esto es un ejemplo de cómo la arquitectura de Mamba-2 ha sido diseñada para ser más simple y adaptable, alineándose con la teoría de la dualidad del espacio de estados estructurado.

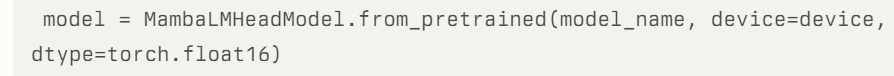
El cambio principal en el modelo Mamba-2, que implica la producción de los parámetros SSM (A, B, C) en paralelo con la entrada X, se evidencia en varias secciones clave del código. A continuación, se detallan los fragmentos relevantes que ilustran esta implementación:

**Sección 4**

**Código como Reflejo de los Aspectos Clave de MambaChat**

4.1. Estructura del Modelo

El modelo `MambaLMHeadModel` se define en el archivo `mixer\_seq\_simple.py`, que implementa una arquitectura basada en Selective State Space Models (SSM). Este modelo se inicializa en el primer código mediante:

Mamba Chat se basa en la mejora de los SSM, haciéndolos más eficientes en términos de cálculo y memoria. Esto permite que Mamba Chat genere respuestas más rápidas y precisas en tiempo real, lo cual es crucial para aplicaciones como la atención al cliente.

4.2. Configuración de Parámetros:

El código permite ajustar parámetros como `temperature`, `top\_p` y `max\_length`, que son cruciales para controlar la creatividad y la longitud de las respuestas generadas. Esto se relaciona con la capacidad de Mamba Chat para adaptarse y ofrecer respuestas precisas y relevantes en tiempo real.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

4.3. Generación de Respuestas

Mamba Chat está diseñado para ofrecer respuestas instantáneas, lo que se traduce en una mejora en la eficiencia operativa y en la satisfacción del cliente. La capacidad de generar respuestas en tiempo real es una de las principales ventajas.

La función `chat\_with\_mamba` utiliza el modelo para generar respuestas a partir de la entrada del usuario y el historial de conversación. La generación se realiza con:

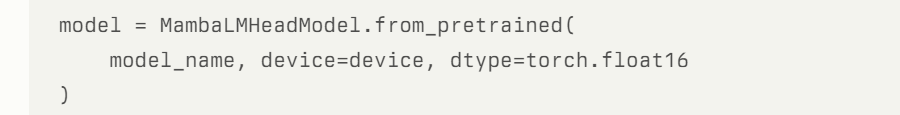
Texto, Carta

Descripción generada automáticamente

4.4. Eficiencia Computacional

Mamba Chat es cinco veces más rápido que los modelos tradicionales basados en Transformers. Esta eficiencia se logra gracias a la arquitectura mejorada de los SSM, que permite manejar secuencias largas de manera más efectiva.

El uso de `torch.float16` en la inicialización del modelo permite que Mamba Chat utilice menos memoria y recursos computacionales:



4.5. Uso de Tokenizers: El código utiliza `AutoTokenizer` para manejar la tokenización de las entradas del usuario. Esto es esencial para preparar los datos antes de pasarlos al modelo, asegurando que las consultas sean interpretadas correctamente. La capacidad de Mamba Chat para procesar entradas largas se ve reflejada en la configuración del tokenizer y en cómo se gestionan las secuencias.



**Conclusiones**

Los modelos de inteligencia artificial han ido evolucionando desde la aparición de la arquitectura Transformer en 2017 y llega hasta el muy reciente modelo Mamba en 2023. En el proceso, lo que se ha presenciado es una rápida mejora en la capacidad de Sistemas de IA para procesar y generar contenidos, culminando con la integración general de chatbots avanzados y sistemas generativos de IA en aplicaciones cotidianas.

El dominio de los Transformers en el campo de la IA generativa ha sido innegable, demostrando una capacidad sin precedentes para comprender contextos complejos y manejar secuencias de datos de manera eficiente. Sin embargo, a medida que estos modelos han crecido en tamaño y complejidad, han surgido limitaciones significativas, particularmente en términos de eficiencia computacional y escalabilidad.

La introducción de Mamba por parte de Albert Gu y Tri Dao es, por tanto, un paso hacia la solución de estas limitaciones. Propone una nueva clase de modelos selectivos de espacio de estados, mediante los cuales Mamba logrará un poder de modelado como el de Transformers mientras escala linealmente con la longitud de la secuencia. Esta innovación es muy crítica, especialmente en aplicaciones que requieren un procesamiento de datos extenso y complejo.

Nuestro análisis comparativo entre la versión simplificada del modelo Mamba (SimpleMambaSSM) y un modelo LSTM tradicional ha arrojado luz sobre los beneficios potenciales de este nuevo enfoque. La eficiencia y flexibilidad computacional demostradas por el modelo basado en Mamba indican una ruta muy prometedora para futuras aplicaciones en tiempo real y sistemas con recursos limitados.

Sin embargo, es importante reconocer que, aunque los resultados iniciales son prometedores, se requiere una investigación más exhaustiva para validar plenamente la efectividad de Mamba en una gama más amplia de aplicaciones. Los desafíos iniciales enfrentados por Mamba, incluido el rechazo inicial por Ethics Review, subrayan la necesidad de un escrutinio riguroso y una evaluación continua de nuevas tecnologías de IA.

Las investigaciones posteriores, como "MoE-Mamba" y "Vision Mamba", han comenzado a demostrar la versatilidad y adaptabilidad de Mamba en diversos contextos, desde el procesamiento de lenguaje natural hasta la representación visual. Estos desarrollos sugieren que Mamba podría tener un impacto significativo en Múltiples áreas de la IA, potencialmente redefiniendo los enfoques estándar en el campo.

Mamba es un importante avance en el modelado de secuencias, y se establece como una prometedora alternativa a los tradicionales Transformers. Su eficiencia para mantener el rendimiento, con una computacional significativamente mayor, podría ser transformadora para el futuro de la IA generativa y el procesamiento de lenguaje natural. Es importante continuar investigando y desarrollando modelos como Mamba, probando su potencial en diversas aplicaciones y probando rigurosamente su rendimiento frente a las tecnologías existentes. Parece que el futuro de la IA está en un punto de inflexión, donde modelos como Mamba pueden ofrecer nuevas oportunidades hacia la creación de sistemas de IA más eficientes, escalables y adaptables.

Referencias:

[1] «Sequence Models: An in-depth look at Key Algorithms and their real-world applications». Accedido: 26 de julio de 2024. [En línea]. Disponible en: https://www.linkedin.com/pulse/sequence-models-in-depth-look-key-algorithms-ritika-dokania-tqzdc

[2] Dot CSV, *¿Qué es un TRANSFORMER? La Red Neuronal que lo cambió TODO!*, (27 de septiembre de 2021). Accedido: 2 de julio de 2024. [En línea Video]. Disponible en: https://www.youtube.com/watch?v=aL-EmKuB078

[3] «What is an AI token?», Digital Trends. Accedido: 2 de julio de 2024. [En línea]. Disponible en: https://www.digitaltrends.com/computing/what-is-an-ai-token/

[4] «What are tokens and how to count them? | OpenAI Help Center». Accedido: 2 de julio de 2024. [En línea]. Disponible en: https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them

[5] «¿Qué son las Redes LSTM? | Codificando Bits». Accedido: 2 de julio de 2024. [En línea]. Disponible en: https://www.codificandobits.com/blog/redes-lstm/

[6] «Buscando a Nemo», *Wikipedia, la enciclopedia libre*. 26 de junio de 2024. Accedido: 2 de julio de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Buscando\_a\_Nemo&oldid=160963870

[7] A. Vaswani *et al.*, «Attention Is All You Need», 1 de agosto de 2023, *arXiv*: arXiv:1706.03762. doi: 10.48550/arXiv.1706.03762.

[8] A. Vaswani *et al.*, «Attention Is All You Need», 1 de agosto de 2023, *arXiv*: arXiv:1706.03762. doi: 10.48550/arXiv.1706.03762.

[9] «SensIO». Accedido: 29 de julio de 2024. [En línea]. Disponible en: https://juansensio.com/blog/060\_attention

[10] «¿Qué son los transformadores?: explicación de los transformadores en inteligencia artificial: AWS». Accedido: 29 de julio de 2024. [En línea]. Disponible en: https://aws.amazon.com/es/what-is/transformers-in-artificial-intelligence/

[11] «El algoritmo de transformador con la menor complejidad de tiempo óptima posible | HackerNoon». Accedido: 25 de julio de 2024. [En línea]. Disponible en: https://hackernoon.com/es/el-algoritmo-transformador-con-la-menor-complejidad-de-tiempo-optimo-posible

[12] A. Gu y T. Dao, «Mamba: Linear-Time Sequence Modeling with Selective State Spaces», 31 de mayo de 2024, *arXiv*: arXiv:2312.00752. doi: 10.48550/arXiv.2312.00752.

[13] «Chapter 39. Parallel Prefix Sum (Scan) with CUDA | NVIDIA Developer». Accedido: 3 de julio de 2024. [En línea]. Disponible en: https://developer.nvidia.com/gpugems/gpugems3/part-vi-gpu-computing/chapter-39-parallel-prefix-sum-scan-cuda

[14] «Atención inmediata: revolucionando la eficiencia de los transformadores - Unite.AI». Accedido: 25 de julio de 2024. [En línea]. Disponible en: https://www.unite.ai/es/flash-attention-revolutionizing-transformer-efficiency/

[15] «Últimos avances en arquitecturas de IA: Transformers, Mamba, I-JEPA | NEDNEX». Accedido: 25 de julio de 2024. [En línea]. Disponible en: https://nednex.com/es/latest-advancements-in-ai-architectures-transformers-mamba-i-jepa/

[16] «¡Descubre Mamba: El Futuro de los Modelos de Lenguaje!» Accedido: 26 de julio de 2024. [En línea]. Disponible en: https://www.toolify.ai/es/ai-news-es/descubre-mamba-el-futuro-de-los-modelos-de-lenguaje-2484887

[17] A. Gu y T. Dao, «Mamba: Linear-Time Sequence Modeling with Selective State Spaces», 31 de mayo de 2024, *arXiv*: arXiv:2312.00752. doi: 10.48550/arXiv.2312.00752.

[18] «Teorema de muestreo de Nyquist-Shannon - Wikipedia, la enciclopedia libre». Accedido: 26 de julio de 2024. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Teorema\_de\_muestreo\_de\_Nyquist-Shannon

[19] L. Zhu, B. Liao, Q. Zhang, X. Wang, W. Liu, y X. Wang, «Vision Mamba: Efficient Visual Representation Learning with Bidirectional State Space Model», 10 de febrero de 2024, *arXiv*: arXiv:2401.09417. doi: 10.48550/arXiv.2401.09417.

[20] «Vision Mamba: Efficient Visual Representation Learning with Bidirectional State Space Model». Accedido: 25 de julio de 2024. [En línea]. Disponible en: https://arxiv.org/html/2401.09417v2

[21] Y. Liu *et al.*, «VMamba: Visual State Space Model», 26 de mayo de 2024, *arXiv*: arXiv:2401.10166. doi: 10.48550/arXiv.2401.10166.

[22] «MoE-Mamba: Efficient Selective State Space Models with Mixture of Experts». Accedido: 25 de julio de 2024. [En línea]. Disponible en: https://arxiv.org/html/2401.04081v2

[23] M. Pióro *et al.*, «MoE-Mamba: Efficient Selective State Space Models with Mixture of Experts», 26 de febrero de 2024, *arXiv*: arXiv:2401.04081. doi: 10.48550/arXiv.2401.04081.

[24] J. Wang, T. Gangavarapu, J. N. Yan, y A. M. Rush, «MambaByte: Token-free Selective State Space Model», 2 de abril de 2024, *arXiv*: arXiv:2401.13660. doi: 10.48550/arXiv.2401.13660.

[25] O. Lieber *et al.*, «Jamba: A Hybrid Transformer-Mamba Language Model», 28 de marzo de 2024, *arXiv*: arXiv:2403.19887. doi: 10.48550/arXiv.2403.19887.

[26] Y. Bhaskar, «AI21 Labs Introduces Jamba: The First Production-Grade Hybrid SSM-Transformer Model», Medium. Accedido: 27 de julio de 2024. [En línea]. Disponible en: https://medium.com/@yash9439/ai21-labs-introduces-jamba-the-first-production-grade-hybrid-ssm-transformer-model-7f6f500441a4

[27] «Towards AI en LinkedIn: JAMBA, the First Powerful Hybrid Model is Here». Accedido: 27 de julio de 2024. [En línea]. Disponible en: https://www.linkedin.com/posts/towards-artificial-intelligence\_jamba-the-first-powerful-hybrid-model-is-activity-7186635678888153088-LpYH

[28] Algorithmic Simplicity, *MAMBA from Scratch Neural Nets Better and Faster than Transformers*, (mayo de 2024). [En línea Video]. Disponible en: https://www.youtube.com/watch?v=N6Piou4oYx8&t=10s

[29] «State Space Duality (Mamba-2) Part I - The Model | Tri Dao». Accedido: 27 de julio de 2024. [En línea]. Disponible en: https://tridao.me/blog/2024/mamba2-part1-model/

[30] «mamba-variants». Accedido: 9 de agosto de 2024. [En línea]. Disponible en: https://www.kaggle.com/datasets/verracodeguacas/mamba-variants

[31] «Pretrained Mamba Model Classification on Intel Img». Accedido: 9 de agosto de 2024. [En línea]. Disponible en: https://www.kaggle.com/datasets/mahmudulhasantasin/pretrained-mamba-model-classification-on-intel-img

[32] V. P. D. Hoang, *VuBacktracking/mamba-text-classification*. (1 de agosto de 2024). Python. Accedido: 9 de agosto de 2024. [En línea]. Disponible en: https://github.com/VuBacktracking/mamba-text-classification

[33] «clibrain/mamba-2.8b-instruct-openhermes · Hugging Face». Accedido: 9 de agosto de 2024. [En línea]. Disponible en: https://huggingface.co/clibrain/mamba-2.8b-instruct-openhermes

[34] «About Us». Accedido: 8 de agosto de 2024. [En línea]. Disponible en: https://icml.cc/About

[35] J. Park *et al.*, «Can Mamba Learn How to Learn? A Comparative Study on In-Context Learning Tasks», 25 de abril de 2024, *arXiv*: arXiv:2402.04248. Accedido: 8 de agosto de 2024. [En línea]. Disponible en: http://arxiv.org/abs/2402.04248

[36] «TII release Falcon Mamba SSLM; and more Middle East AI News». Accedido: 23 de agosto de 2024. [En línea]. Disponible en: https://www.linkedin.com/pulse/tii-release-falcon-mamba-sslm-more-middle-east-ai-news-malin-hkk9f

[37] «Falcon LLM». Accedido: 23 de agosto de 2024. [En línea]. Disponible en: https://falconllm.tii.ae/tii-releases-first-sslm-with-falcon-mamba-7b.html

1. Son componentes en modelos de aprendizaje profundo que permiten al modelo enfocarse en partes específicas de la entrada al procesar secuencias. Ayudan al modelo a "prestar atención" a información relevante para una tarea dada. [↑](#footnote-ref-2)
2. Es una representación de datos en un espacio vectorial de baja dimensión. En el contexto del procesamiento del lenguaje natural, los embeddings convierten palabras o tokens en vectores numéricos que capturan relaciones semánticas. [↑](#footnote-ref-3)
3. Un método computacional que permite a un modelo asignar diferentes pesos o importancia a diferentes partes de los datos de entrada. Esto es crucial en tareas como la traducción automática o el resumen de texto. [↑](#footnote-ref-4)
4. Un tipo específico de mecanismo de atención donde el modelo aprende a atender a diferentes partes de su propia representación interna. Es fundamental en arquitecturas como los transformers. [↑](#footnote-ref-5)
5. Los filtros o kernels son matrices utilizadas en el procesamiento de secuencias para extraer características relevantes mediante la operación de convolución [↑](#footnote-ref-6)
6. H3 (Hungry Hungry Hippos) son una nueva capa de Modelos de Espacio de Estados (SSM) desarrollada para el modelado de lenguaje [↑](#footnote-ref-7)
7. Tipo de evaluación en tareas específicas sin haber recibido ejemplos de entrenamiento previo [↑](#footnote-ref-8)
8. Evaluaciones diseñadas para probar la capacidad de memoria y recuerdo asociativo de los LM [↑](#footnote-ref-9)
9. Una de las métricas para evaluar modelos de lenguaje. [↑](#footnote-ref-10)
10. El proceso de dividir texto en unidades más pequeñas llamadas tokens. Puede ser a nivel de palabras, subpalabras o caracteres, y es un paso crucial en el procesamiento del lenguaje natural. [↑](#footnote-ref-11)
11. En aprendizaje profundo, se refiere a la tendencia de un modelo a favorecer ciertas predicciones sobre otras. También puede referirse a sesgos injustos en los datos o modelos que llevan a discriminación. [↑](#footnote-ref-12)
12. Red neuronal artificial (RNA) formada por múltiples capas. [↑](#footnote-ref-13)
13. Un modelo de lenguaje de gran escala desarrollado por Mistral AI. Es un modelo de mezcla de expertos (MoE) con 8 expertos por capa y aproximadamente 7 mil millones de parámetros por experto. [↑](#footnote-ref-14)
14. Se refiere al software cuyo código fuente está disponible públicamente para que cualquiera pueda ver, modificar y distribuir. Promueve la colaboración y transparencia en el desarrollo de software. [↑](#footnote-ref-15)
15. Chinchilla AI es una familia de grandes modelos de lenguaje desarrollados por el equipo de investigación de DeepMind. [↑](#footnote-ref-16)
16. Es un conjunto de datos diverso de código abierto de 886,03 GB de texto en inglés creado como un conjunto de datos de entrenamiento para modelos grandes de lenguaje (LLM). [↑](#footnote-ref-17)