La investigación reportada en esta tesis es parte de los programas de investigación del CICESE (Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California).
La investigación fue financiada por el SECIHTI (Secretaría de Ciencia, Humanidades, Tecnología e Innovación).

Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Maestría en Ciencias
en Ciencias de la Computación

OrthoQuant: Compresión de vectores y matrices mediante una rotación ortonormal, un método universal e independiente de los datos

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de

Maestro en Ciencias

Presenta:

Scarlett Magdaleno Gatica

Ensenada, Baja California, México

2025

Tesis defendida por

Scarlett Magdaleno Gatica

y aprobada por el siguiente Comité

Dr. Edgar Leonel Chávez González
Director de tesis

Dr. José Alberto Fernández Zepeda

Dr. Daniel Sauceda Carvajal

Dr. Mariano José Juan Rivera Meraz

Dr. Pedro Gilberto López Mariscal
Coordinador del Posgrado en Ciencias de la Computación

Dra. Ana Denise Re Araujo
Directora de Estudios de Posgrado

Resumen de la tesis que presenta Scarlett Magdaleno Gatica como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación.

OrthoQuant: Compresión de vectores y matrices mediante una rotación ortonormal, un método universal e independiente de los datos

Resumen aprobado por:

Dr. Edgar Leonel Chávez González
Director de tesis

La creciente escala de los modelos de inteligencia artificial y las bases de datos vectoriales ha generado una crisis de eficiencia computacional y de almacenamiento. Los métodos de compresión actuales a menudo fracasan frente a datos de alta dimensionalidad con distribuciones arbitrarias o matrices de rango completo, y frecuentemente dependen de costosos procesos de reentrenamiento o calibración con datos externos. Para abordar este problema, en este trabajo se propone OrthoQuant, un método de compresión universal e independiente de los datos para vectores y matrices. El método resuelve estas limitaciones mediante la proyección de los datos sobre una matriz de rotación ortonormal aleatoria, sin requerir reentrenamiento ni ajuste fino. Esta transformación, fundamentada en los principios de las representaciones de Kashin, estabiliza la distribución interna de los datos, lo que permite una cuantización robusta y eficiente. La evaluación del método revela una alta fidelidad funcional. En bases de datos vectoriales, se obtuvo una compresión de hasta el 96 % manteniendo un *recall* elevado en la búsqueda del primer vecino más cercano, recuperando 1024 vecinos aproximados y aplicando *re-ranking*. En compresiones menores, es posible reducir la cantidad de vectores recuperados para alcanzar el mismo *recall*, aunque con el correspondiente compromiso en el uso de memoria. Aplicado a Modelos Grandes de Lenguaje (LLMs), los pesos del modelo LLaMA 3.2 3B fueron comprimidos hasta en un 72.35 %, manteniendo un rendimiento cercano al original y alcanzando una perplejidad de 7.85 en WikiText-2 con 4 bits por elemento, cifra que se compara favorablemente con la perplejidad de 6.94 del modelo base. Adicionalmente, la variante para vectores puede operar en modo *streaming*, lo que permite su integración en las capas de atención de LLMs y modelos generativos para procesar contextos más largos y acelerar la búsqueda de similitud. El método OrthoQuant constituye una contribución al estado del arte por su naturaleza general, robusta y libre de datos. Su capacidad para operar sin datos de calibración ni reentrenamiento, junto con su potencial para optimizar mecanismos de atención, lo validan como una herramienta para la democratización y eficiencia de la inteligencia artificial a gran escala.

Palabras clave: compresión de vectores, compresión de matrices, rotación ortonormal aleatoria, OrthoQuant, modelos de lenguaje grandes (LLMs), búsqueda por similitud, almacenamiento eficiente

Abstract of the thesis presented by Scarlett Magdaleno Gatica as a partial requirement to obtain the Master of Science degree in Computer Science.

OrthoQuant: Vector and matrix compression via orthonormal rotation, a universal and data-independent method

Abstract approved by:

Dr. Edgar Leonel Chávez González
Thesis Director

The growing scale of artificial intelligence models and vector databases has led to a crisis in computational and storage efficiency. Current compression methods often fail when dealing with high-dimensional data with arbitrary distributions or full-rank matrices, and frequently depend on costly retraining or calibration processes with external data. To address this problem, in this work we propose OrthoQuant, a universal, data-independent compression method for vectors and matrices. The method overcomes these limitations by projecting the data onto a random orthonormal rotation matrix, without requiring retraining or fine-tuning. This transformation, grounded in the principles of Kashin's representations, stabilizes the internal distribution of the data, enabling robust and efficient quantization. Evaluation results reveal high functional fidelity. In vector databases, compression of up to 96 % was achieved while maintaining a high *recall* in the search for the true nearest neighbor, retrieving 1024 approximate neighbors and applying *re-ranking*. For lower compression ratios, the number of retrieved vectors can be reduced to achieve the same *recall*, albeit with the corresponding trade-off in memory usage. Applied to Large Language Models (LLMs), the weights of the LLaMA 3.2 3B model were compressed by up to 72.35 %, maintaining performance close to the original and achieving a perplexity of 7.85 on WikiText-2 with 4 bits per element, a figure that compares favorably to the 6.94 perplexity of the base model. Additionally, the vector variant can operate in *streaming* mode, enabling integration into the attention layers of LLMs and generative models to process longer contexts and accelerate similarity search. OrthoQuant represents a contribution to the state of the art due to its general, robust, and data-free nature. Its ability to operate without calibration data or retraining, along with its potential to optimize attention mechanisms, validates it as a tool for the democratization and efficiency of large-scale artificial intelligence.

Keywords: vector compression, matrix compression, random orthonormal rotation, OrthoQuant, large language models (LLMs), similarity search, efficient storage

Dedicatoria

A mi familia, mi novio y mis amigos. Quienes, con su cariño y apoyo incondicional, me acompañaron en este camino.

Agradecimientos

Al Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California (CICESE), por brindarme la oportunidad de desarrollar esta investigación.

A la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI), por el apoyo económico durante mis estudios de maestría.

A mi director de tesis, por su orientación, apoyo constante y la confianza depositada en mí durante todo el proceso.

A mi comité de tesis, por sus valiosos comentarios y sugerencias que enriquecieron este trabajo.

Tabla de contenido

Página

Lista de figuras

Figura Página

Lista de tablas

Capítulo 1. Introducción

Actualmente se vive en el auge de la era digital. Gracias a los avances tecnológicos y al uso constante de estas herramientas por parte de los usuarios, cada vez se genera una mayor cantidad de datos. Esta abundancia de información ha permitido mejorar sistemas como los de recomendación, donde ahora se pueden ofrecer experiencias más personalizadas, y también ha ayudado a las empresas a conocer mejor a sus clientes y a ofrecer productos más adecuados. Pero además, esta enorme cantidad de datos, junto con el desarrollo de hardware de alto rendimiento, ha dado paso al desarrollo de modelos cada vez más complejos tanto en aprendizaje de máquina como en aprendizaje profundo. Estos últimos requieren una cantidad considerable de poder de cómputo y grandes volúmenes de datos.

Esta combinación de datos masivos y hardware especializado ha permitido avances notables en áreas como el procesamiento de lenguaje natural y la visión por computadora. Estos modelos ahora forman parte de productos que se usan todos los días, y todo indica que su uso seguirá creciendo. Sin embargo, conforme se lanzan modelos más poderosos, también se vuelven cada vez más costosos computacionalmente. Aunque estos modelos muestran mejoras constantes en precisión, su entrenamiento puede tardar semanas o incluso meses, y su almacenamiento y despliegue se han vuelto cada vez más complicados, incluso usando hardware moderno de última generación, ya que su tamaño masivo requiere una gran cantidad de GPUs tan solo para su operación. Como resultado, su uso queda limitado a quienes cuentan con los recursos económicos suficientes, mientras que para el resto de las personas e incluso para instituciones públicas de investigación, estos modelos resultan inaccesibles.

Además, hoy en día existe una carrera constante entre las principales compañías tecnológicas por desarrollar los modelos más grandes y precisos, donde muchas veces el desempeño no solo se debe a las ideas del modelo, sino al acceso a infraestructura computacional. Por otro lado, gran parte de la investigación se centra en quién logra la mejor métrica, sin enfocarse en cómo aprovechar los modelos ya entrenados, cuyo entrenamiento ya ha supuesto un consumo considerable de datos y energía. En este sentido, la compresión de modelos preentrenados se vuelve una estrategia clave para democratizar su uso y para extender la vida útil de modelos valiosos sin necesidad de volver a entrenar desde cero. Además, la compresión no solo tiene un beneficio práctico o económico, sino también ecológico, pues evita repetir el consumo de energía necesario para entrenar modelos desde cero.

Otra área de oportunidad importante está en las bases de datos vectoriales, que son la base de muchos sistemas actuales. De hecho, incluso antes del auge de la inteligencia artificial, ya se usaban ampliamente; por ejemplo, los motores de búsqueda como Google se apoyan en bases de datos vectoriales para

indexar contenido. Su uso es tan frecuente y su crecimiento tan constante, que simplemente consultarlas puede volverse una tarea computacionalmente costosa. Así, tanto por eficiencia como por escalabilidad, comprimir estas bases también es una necesidad.

Estos son sólo algunos ejemplos, pero en general, vectores y matrices son estructuras centrales en el contexto actual. Saber cómo comprimir estas representaciones abre posibilidades en muchas áreas: desde compresión de modelos y bases de datos, hasta compresión de imágenes o de otros objetos que dependan de estas estructuras.

Actualmente existen distintas estrategias de compresión. En el caso de bases vectoriales, una opción es la cuantización o el uso de índices que permitan búsquedas más rápidas. En el caso de las matrices, como las que se encuentran en redes neuronales, se han propuesto técnicas como la cuantización, la aproximación de rango bajo, la poda o la destilación de conocimiento. Sin embargo, muchos de estos métodos requieren procesos adicionales como reentrenamiento o calibración con datos específicos, lo cual no sólo aumenta la complejidad del proceso, sino que también hace al método dependiente de datos que, en algunos escenarios, pueden no estar disponibles. Además, muchas veces estas estrategias se basan en heurísticas u optimizaciones locales, sin una comprensión más profunda de la distribución de los datos y sin ofrecer garantías sobre la estabilidad de la compresión bajo diferentes condiciones.

1.1. Justificación

Los métodos de compresión actuales han logrado avances significativos en la disminución de los requerimientos de memoria mientras se busca mantener la precisión del modelo o la fidelidad de los datos. No obstante, estas soluciones suelen venir acompañadas de heurísticas locales o procesos adicionales que no sólo aumentan la complejidad del método de compresión, sino que, de manera fundamental, dependen del uso de datos para compensar la pérdida de desempeño que su enfoque inherentemente ocasiona.

Un concepto fundamental en el estudio de matrices es el rango, que indica la cantidad máxima de vectores linealmente independientes que pueden formarse a partir de las filas o columnas de una matriz. Las matrices de rango completo, al alcanzar el rango máximo, no presentan redundancias intrínsecas en sus datos. Esto significa que no se pueden representar con menos parámetros sin incurrir en pérdida de información, ya que la independencia lineal de todos sus vectores implica que ninguno puede expresarse como combinación lineal de los demás. Por el contrario, las matrices que no son de rango completo sí

contienen redundancias que pueden aprovecharse para su compresión eficiente, es decir, para reducir el número de parámetros necesarios para su representación sin pérdida de información.

El problema principal al tratar con matrices de rango completo radica en que cualquier intento de comprimirlas, es decir, representarlas con un menor número de parámetros, implica necesariamente una pérdida de precisión. Esto plantea un desafío crítico en aplicaciones donde la eficiencia del almacenamiento y el procesamiento de datos es esencial, pero la integridad de la información es primordial. Si bien, algunas técnicas como la descomposición en valores singulares (SVD), que constituye la base teórica de la aproximación de rango bajo, permiten una compresión efectiva en matrices con redundancia, no pueden aplicarse a matrices de rango completo sin comprometer de forma significativa la integridad de los datos. Es precisamente en estos escenarios donde los métodos actuales encuentran serias limitaciones, ya que su efectividad a menudo se basa en la explotación de redundancias o en un reentrenamiento costoso que no siempre es factible o deseable.

En el ámbito de las bases de datos vectoriales, la necesidad de compresión es apremiante debido al volumen creciente de colecciones. Algunos métodos como la cuantización vectorial o la indexación aproximada buscan reducir el tamaño de los vectores o acelerar las búsquedas, pero a menudo conllevan una pérdida inevitable de precisión en la similitud calculada. Si bien en algunas aplicaciones esta pérdida es tolerable, en otras, como la recuperación de información de alta fidelidad, puede degradar significativamente el rendimiento. La meta es reducir el espacio sin comprometer drásticamente la capacidad de encontrar los vecinos más cercanos o de preservar las relaciones geométricas cruciales entre los vectores.

Por ende, existe una necesidad evidente de métodos de compresión que sean aplicables de forma más universal a cualquier tipo de datos, independientemente de su distribución original o rango, y cuya metodología no se base únicamente en heurísticas superficiales, sino en una mayor comprensión y aprovechamiento de las distribuciones subyacentes de los elementos de los datos.

1.2. Hipótesis

Para abordar estos desafíos, esta tesis se fundamenta en las siguientes hipótesis:

- Existe un método de compresión que permite comprimir de forma estable tanto vectores como matrices independientemente de su distribución original.

- Aplicar una matriz de rotación ortonormal a los datos permite una compresión más eficiente al estabilizar sus distribuciones internas.
- Conservar los valores de mayor magnitud absoluta sin pérdida asegura una mejor calidad de aproximación y una mayor fidelidad en la conservación del producto punto.

1.3. Método propuesto

En línea con las hipótesis planteadas, se propone un método de compresión basado en la proyección de los datos sobre una matriz de rotación ortonormal. El objetivo principal de esta proyección es transformar los datos de tal manera que las distribuciones de los elementos resultantes sean más estables y predecibles, lo que se puede aprovechar para una compresión más eficiente y efectiva. Además, un paso clave de nuestra metodología, que contribuye significativamente a la calidad de las compresiones obtenidas, es la conservación estratégica de los *outliers* en estas nuevas distribuciones. Al preservar los valores de mayor magnitud sin pérdida durante el proceso de compresión, se busca asegurar una aproximación de mayor calidad a los datos originales y una fidelidad superior en la conservación del producto punto, una operación crítica en muchas aplicaciones de IA y análisis de datos.

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar y evaluar una técnica de compresión basada en la aplicación de una matriz de rotación ortonormal aleatoria, aplicable tanto a conjuntos de vectores como a conjuntos de matrices, con el fin de reducir eficientemente el número de parámetros y preservar la fidelidad de la información.

1.4.2. Objetivos específicos

- Evaluar la calidad de la aproximación obtenida con la técnica propuesta en bases de datos de vectores (mediante tareas de búsqueda por similitud) y en matrices de modelos grandes de lenguaje

(mediante la evaluación de su desempeño).

- Determinar las tasas de compresión alcanzadas por el método propuesto en ambos dominios de aplicación.
- Comparar la calidad de la aproximación obtenida con la de diversas estrategias del estado del arte para bases de datos vectoriales y matrices de modelos de redes neuronales.

Capítulo 2. Trabajo relacionado

2.1. Justificación del uso de una matriz de rotación ortonormal aleatoria

El método de compresión propuesto en esta tesis se basa en la aplicación de una matriz de rotación ortonormal aleatoria para dispersar la energía de los vectores o matrices de entrada, lo que resulta en una distribución cuasi-gaussiana de los valores, centrada en cero. Este fenómeno no es una mera observación empírica, sino que está profundamente arraigado en principios matemáticos fundamentales: el Principio de Incertidumbre, las Representaciones de Kashin y la Concentración de la Medida.

2.1.1. Principio de incertidumbre y dispersión de energía

El Principio de Incertidumbre es una idea fundamental en matemáticas que establece que una señal no puede estar demasiado concentrada en dos dominios diferentes al mismo tiempo. Por ejemplo, si una señal está muy localizada en el tiempo, su representación en el dominio de la frecuencia estará muy dispersa, y viceversa. Este principio, inicialmente formulado en física, ha sido adaptado a espacios discretos y de dimensión finita (Donoho & Stark, 1989; Folland & Sitaram, 1997; Shi et al., 2016).

En el contexto del método propuesto en esta tesis, el principio de incertidumbre se traduce en que, al transformar un vector mediante una matriz de rotación ortonormal, se impide que su energía se concentre en unos pocos coeficientes. Las matrices de rotación ortonormal son isométricas, es decir, preservan tanto las normas como los productos punto entre vectores, lo que implica la conservación de longitudes y ángulos en el espacio euclidiano (Saraeb, 2024). Por tanto, estas transformaciones no destruyen la energía del vector original, sino que la redistribuyen en el espacio transformado.

Este fenómeno encuentra un paralelo conceptual en el trabajo de Tan & Brouwer (2025), quienes estudian la dispersión de operadores en sistemas cuánticos mediante circuitos formados por matrices unitarias aleatorias. Aunque su enfoque se sitúa en el contexto de la mecánica cuántica y utiliza matrices unitarias complejas, estas matrices comparten con las matrices ortonormales la propiedad fundamental de ser isométricas. Su análisis muestra que, bajo la acción de tales transformaciones, la información se dispersa espacialmente sin perderse, lo cual respalda el principio de que las transformaciones lineales isométricas pueden reorganizar la información mediante su redistribución, sin destruirla.

2.1.2. Representaciones de Kashin y coeficientes balanceados

La idea de que el principio de incertidumbre puede traducirse en una dispersión de la energía a lo largo de los valores de un vector transformado se ve respaldada por el concepto de *representación de Kashin*, el cual resulta fundamental para comprender cómo las matrices ortonormales aleatorias permiten distribuir de forma más uniforme la energía entre los coeficientes resultantes. En particular, una representación de Kashin garantiza que los coeficientes transformados de un vector tienen magnitudes pequeñas y comparables, lo cual impide que la energía se concentre en unos pocos elementos. Formalmente, sea $X \in \mathbb{R}^{m \times n}$ una matriz que contiene vectores columna, y sea $\mathbf{x} \in \mathbb{R}^n$ uno de dichos vectores. Si $Q \in \mathbb{R}^{n \times n}$ es una transformación lineal que genera una representación de Kashin, entonces los coeficientes transformados $\mathbf{d} = Q\mathbf{x}$ satisfacen la siguiente desigualdad:

$$\|\mathbf{d}\|_\infty \leq \frac{K}{\sqrt{n}} \|\mathbf{x}\|_2, \tag{1}$$

donde $K > 0$ es una constante universal. Esta cota implica que ningún coeficiente individual domina la energía del vector, lo cual fuerza a que la norma ℓ_2 de $\mathbf{x}$ se reparta de manera equilibrada entre las entradas de $\mathbf{d}$ (Lyubarskii & Vershynin, 2010). En consecuencia, se suprimen los *outliers* que suelen aparecer en los vectores originales, ya que sus magnitudes quedan acotadas por $1/\sqrt{n}$.

Esta propiedad tiene implicaciones importantes para la compresión: cuando los coeficientes son pequeños y de magnitud similar, es posible cuantificarlos de forma más eficiente y con menor pérdida de información. Lyubarskii & Vershynin (2010) han demostrado que las matrices ortonormales aleatorias producen representaciones de Kashin con alta probabilidad, lo que permite que al aplicar dichas transformaciones, los coeficientes resultantes se mantengan balanceados. Esta demostración se apoya en herramientas del análisis funcional y en particular en el fenómeno de concentración de la medida, el cual garantiza que, en espacios de alta dimensión, la mayoría de las imágenes de un vector bajo una transformación ortonormal aleatoria se concentran cerca de su valor esperado (Ledoux, 2001). Dicho de otro modo, la energía del vector transformado se distribuye de forma casi uniforme entre todos los coeficientes, con una probabilidad que crece exponencialmente con la dimensión. Esta característica no sólo favorece una mayor compresión, sino que también incrementa la robustez frente a errores de cuantización, ya que ningún coeficiente individual tiene un peso desproporcionado en la reconstrucción del vector. Este principio ha sido aprovechado recientemente en algoritmos de cuantización aplicados a modelos grandes de lenguaje (Merkulov et al., 2024).

2.2. Compresión de vectores

Las bases de datos vectoriales permiten representar objetos complejos como imágenes, frases, documentos o usuarios, mediante vectores de alta dimensión que capturan sus características esenciales en un espacio continuo. Estas bases de datos están diseñadas para almacenar, indexar y recuperar eficientemente vectores, lo cual es fundamental en tareas modernas de inteligencia artificial, como la recuperación de información, los sistemas de recomendación y la búsqueda semántica.

El proceso central en estas aplicaciones es la búsqueda por similitud, que consiste en encontrar los vectores más cercanos a una consulta en términos de una métrica determinada. A medida que crece el número de vectores y la dimensión de éstos, se vuelve necesario utilizar métodos de compresión para reducir tanto el uso de memoria como el costo computacional de la búsqueda. La compresión no sólo permite almacenar una mayor cantidad de vectores en memoria, sino que también puede acelerar el cálculo de distancias o similitudes, facilitando la búsqueda sobre millones o incluso miles de millones de elementos.

Existen diversos enfoques para la compresión de vectores. Algunos, como el Análisis de Componentes Principales (PCA), proyectan los vectores a un subespacio de menor dimensión que conserva la mayor parte de la varianza de los datos (Pearson, 1901; Jolliffe, 2011). Otros, como la cuantización, restringen los vectores a un conjunto finito de representaciones discretas (Gray & Neuhoff, 1998), de forma análoga a cómo una imagen puede comprimirse al reducir su paleta de colores a un conjunto limitado pero representativo.

Los métodos de compresión son especialmente relevantes para esta tesis, ya que constituyen el fundamento sobre el cual se propone un nuevo enfoque para representar vectores de alta dimensión. A continuación, se presenta una descripción general del proceso de búsqueda por similitud en este contexto, seguida de una revisión de los principales métodos de compresión desarrollados en la literatura.

2.2.1. Búsqueda por similitud en bases de datos vectoriales

El propósito principal de una base de datos vectorial es permitir la recuperación eficiente de los elementos más similares a una consulta dada, tarea conocida como búsqueda de vecinos más cercanos. Para ello, se requiere calcular una medida de similitud entre la consulta y cada vector almacenado, utilizando métricas

como la distancia euclidiana, la similitud coseno o el producto interno.

Formalmente, el problema de búsqueda de vecinos más cercanos (*Nearest Neighbor Search*) consiste en identificar, dado un vector de consulta $\mathbf{y} \in \mathbb{R}^d$, el vector $\mathbf{x}_n$ dentro de una base de datos $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^d$ que minimiza la distancia respecto a $\mathbf{y}$:

$$n^* = \arg \min_{n \in \{1,\ldots,N\}} \|\mathbf{y} - \mathbf{x}_n\|_2.$$

Sin embargo, calcular esta distancia para cada elemento mediante un escaneo lineal resulta costoso para bases de datos a gran escala, donde N puede alcanzar miles de millones de vectores (Matsui et al., 2018; Aumüller et al., 2020). Para superar esta limitación, se han desarrollado diversos métodos de búsqueda aproximada de vecinos cercanos (*Approximate Nearest Neighbor Search*, ANN), los cuales sacrifican una pequeña cantidad de precisión a cambio de mejoras significativas en velocidad y eficiencia computacional (Andoni et al., 2019).

Los métodos ANN pueden clasificarse en dos grandes enfoques. El primero propone estructuras auxiliares como índices basados en árboles, grafos o hashing, que evitan comparar contra todos los vectores al reducir el espacio de búsqueda (Beis & Lowe, 1997; Muja & Lowe, 2014; Malkov & Yashunin, 2020; Andoni et al., 2019). El segundo enfoque, más relevante en esta tesis, se basa en técnicas de compresión de vectores. Estas no sólo reducen el tamaño de la base de datos, sino que también permiten acelerar el cálculo de similitud al operar directamente sobre representaciones comprimidas (Jégou et al., 2011; Matsui et al., 2018).

La calidad de los resultados en métodos ANN se evalúa mediante métricas como el *Recall@k*, que indica la fracción de vecinos verdaderos más cercanos presentes entre los k primeros vectores recuperados. Por ejemplo, un valor de *Recall@10* $= 0.9$ implica que, en promedio, 9 de los 10 vecinos reales más cercanos fueron recuperados por el sistema.

Los sistemas modernos suelen emplear una arquitectura de búsqueda en dos etapas. En la primera etapa, se recupera un conjunto de candidatos utilizando un método ANN eficiente. En la segunda etapa, denominada *reranking*, se recalculan las similitudes utilizando un modelo más preciso (por ejemplo, un modelo neuronal) para reordenar los resultados (Nogueira & Cho, 2019; Zhuang & Zuccon, 2021). Esta estrategia permite alcanzar alta precisión sin necesidad de aplicar modelos costosos sobre toda la base de datos. Por ejemplo, si al recuperar los 20 vectores más cercanos mediante ANN se alcanza un *Recall@10* de 1, entonces el reranking sobre ese subconjunto es suficiente para identificar correctamente a los 10

vecinos verdaderos más cercanos.

Este paradigma de búsqueda eficiente es clave para aplicaciones a gran escala como recuperación de imágenes, sistemas de recomendación o motores de búsqueda semántica. A continuación, se describen las principales técnicas de compresión relevantes para esta tesis, las cuales buscan optimizar la primera parte del proceso de búsqueda, es decir, la recuperación.

2.2.2. Cuantización escalar

La cuantización escalar (Scalar Quantization, SQ) es una de las técnicas más simples y ampliamente utilizadas para la compresión de vectores. Consiste en aplicar un cuantizador unidimensional de manera independiente a cada componente del vector. Sea $\mathbf{x} \in \mathbb{R}^n$ un vector de una base de datos, cada componente $\mathbf{x}_j$ se mapea a un valor discreto mediante un cuantizador Q_j construido específicamente para esa coordenada.

Un cuantizador escalar de tamaño N se define como una función $Q : \mathbb{R} \to \mathcal{C}$, donde el conjunto de salida o *codebook* es:

$$\mathcal{C} \equiv \{y_1, y_2, \ldots, y_N\} \subset \mathbb{R}, \tag{2}$$

y sus elementos y_i se denominan niveles de salida o *centroides*. Dado que N es finito, cada salida se puede representar mediante un índice codificado con $b = \log_2 N$ bits, lo que permite comprimir cada componente utilizando sólo b bits (Gersho & Gray, 2012).

En la práctica, los cuantizadores se construyen analizando estadísticamente todos los vectores en la base de datos. Por ejemplo, para cada dimensión j, se pueden calcular el mínimo y el máximo de los valores observados, y luego definir N intervalos equiespaciados en ese rango. Cada intervalo se asocia a un centroide, y cada valor $\mathbf{x}_j$ se asigna al centroide correspondiente según el intervalo en el que se encuentre. Así, cada dimensión posee su propio conjunto de centroides, optimizado a partir de los datos. Una alternativa ampliamente utilizada para obtener cuantizadores escalares óptimos, en el sentido de minimizar el error cuadrático medio, es el algoritmo de Lloyd-Max (Lloyd, 1982; Max, 1960). Este procedimiento iterativo alterna entre el ajuste de los límites de decisión (fronteras de cuantización) y la actualización de los centroides, permitiendo que el cuantizador se adapte de manera óptima a la

distribución estadística de los datos.

Sea $\mathbf{X} \in \mathbb{R}^{n \times d}$ una base de datos de n vectores de dimensión d. Tras aplicar cuantización escalar con b bits por componente, se requiere:

- $n \times d \times b$ bits para almacenar los códigos (índices de los centroides),
- $d \times 2^b \times 32$ bits para almacenar los centroides, asumiendo una precisión de 32 bits por valor flotante.

Esta técnica permite una compresión significativa con bajo costo computacional y es frecuentemente utilizada como componente base en métodos de cuantización más sofisticados. La cuantización escalar supone independencia entre las coordenadas del vector, por lo que cuantiza cada componente por separado. Sin embargo, otros trabajos en la literatura han explorado enfoques que buscan aprovechar las dependencias estadísticas entre componentes. Un ejemplo destacado es la cuantización escalar secuencial, la cual introduce una estructura secuencial en el proceso de cuantización: las primeras componentes del vector se cuantizan de manera estándar, y las siguientes se cuantizan condicionadas a los valores previamente cuantizados. De esta manera, el cuantizador puede adaptar su comportamiento utilizando información ya conocida, mejorando la eficiencia global del proceso (Balasubramanian et al., 1995).

2.2.3. Cuantización vectorial

La cuantización vectorial (Vector Quantization, VQ) es una técnica clásica de compresión que considera al vector completo como una unidad indivisible, en lugar de cuantizar cada componente por separado como en la cuantización escalar. Sea $\mathbf{x} \in \mathbb{R}^d$, VQ lo mapea al centroide más cercano $\mathbf{c}_i$ dentro de un conjunto finito de representaciones o *codebook* $\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$, de modo que:

$$Q(\mathbf{x}) = \arg\min_{\mathbf{c}_i \in \mathcal{C}} \|\mathbf{x} - \mathbf{c}_i\|_2. \tag{3}$$

Dado que se considera la geometría completa del vector, esta técnica permite capturar correlaciones entre componentes y obtener una mejor relación tasa-distorsión respecto de métodos que operan de manera independiente por coordenada (Gray & Neuhoff, 1998; Gersho & Gray, 2012). Sin embargo, su costo computacional es alto, ya que implica buscar entre K posibles representaciones para cada vector.

Para construir el *codebook*, se suele utilizar el algoritmo de Linde–Buzo–Gray, una adaptación del algoritmo de k-medias que busca minimizar el error cuadrático medio entre los vectores originales y sus representaciones cuantizadas (Linde et al., 1980).

Sea $\mathbf{X} \in \mathbb{R}^{n \times d}$ una base de datos de n vectores. Al aplicar VQ con un *codebook* de tamaño K, se requiere:

- $n \times \log_2 K$ bits para representar los índices de los centroides,
- $K \times d \times 32$ bits para almacenar el *codebook*, bajo el supuesto de 32 bits por valor flotante.

VQ es efectiva cuando el número de centroides es grande y la dimensión del vector es relativamente baja. Para vectores de alta dimensión, la complejidad de búsqueda y el tamaño del *codebook* se vuelven prohibitivos, lo que ha motivado el desarrollo de técnicas más escalables como la cuantización por productos.

2.2.4. Cuantización por productos

La cuantización por productos (Product Quantization, PQ) es una técnica diseñada para escalar la cuantización vectorial a espacios de alta dimensión, manteniendo un bajo costo de almacenamiento y computación. La idea central consiste en dividir el vector original $\mathbf{x} \in \mathbb{R}^d$ en m sub-vectores de dimensión más pequeña:

$$\mathbf{x} = [\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}], \quad \mathbf{x}^{(i)} \in \mathbb{R}^{d/m},$$

y aplicar cuantización vectorial independiente a cada subvector utilizando *codebooks* locales $\mathcal{C}_i$. El vector completo se representa como la concatenación de los índices de los centroides seleccionados por bloque:

$$Q(\mathbf{x}) = [Q_1(\mathbf{x}^{(1)}), \ldots, Q_m(\mathbf{x}^{(m)})].$$

Esta técnica fue introducida por Jégou et al. (2011) como una solución eficiente para búsqueda aproximada de vecinos más cercanos en bases de datos a gran escala. El ahorro de memoria en este método, si

bien no es tan grande como en VQ, se logra al representar cada vector únicamente mediante un código por subvector. Lo anterior reduce significativamente la dimensionalidad de la base de datos de códigos en comparación con SQ. Mientras que la cuantización escalar requiere almacenar $n \times d$ códigos para una base de datos de n vectores de dimensión d, la cuantización por productos solo requiere $n \times m$ códigos, donde m es el número de subespacios. Esta reducción implica un menor uso de memoria respecto a SQ, aunque no tan agresivo como el logrado por VQ, posicionando a PQ como un compromiso intermedio entre compresión y precisión. Además, PQ supera una de las principales limitaciones de VQ: su falta de escalabilidad en espacios de alta dimensión, ya que al operar sobre subespacios, PQ mantiene su viabilidad computacional incluso en dominios de gran dimensionalidad.

Formalmente, sea k el número de centroides por subespacio y $b = \log_2 k$ los bits por bloque. Entonces, la base de datos comprimida requiere:

- $n \times m \times b$ bits para almacenar los códigos (índices por bloque),
- $m \times k \times (d/m) \times 32 = k \times d \times 32$ bits para los centroides (con 32 bits por componente).

Además de ofrecer una compresión eficiente, PQ permite realizar tareas de búsqueda mediante el cómputo de distancia asimétrica (Asymmetric Distance Computation, ADC). Este enfoque consiste en calcular la distancia entre la consulta original y los vectores reconstruidos a partir de sus representaciones cuantizadas, en lugar de cuantizar también la consulta, lo cual correspondería al esquema de distancia simétrica (Symmetric Distance Computation, SDC). Se ha demostrado que ADC introduce errores menores en la estimación de distancias y presenta una complejidad computacional comparable a la de SDC, por lo que se considera el método preferido para realizar búsquedas. Además, ADC resulta especialmente eficiente, ya que permite acelerar el cómputo mediante la construcción de una tabla de consulta (lookup table, LUT). Esta tabla se construye evaluando la distancia entre cada subvector de la consulta y todos los centroides de cada subespacio de la base de datos. Así, las distancias finales entre la consulta y cada vector cuantizado pueden obtenerse como la suma de los valores precomputados en la LUT, evitando realizar operaciones vectoriales completas y reduciendo significativamente el costo computacional (Jégou et al., 2011). Gracias a esta combinación de eficiencia y precisión, PQ se ha adoptado ampliamente en sistemas modernos de recuperación de información, clasificación y detección de objetos, así como en el diseño de redes neuronales profundas más rápidas y eficientes (Xu et al., 2018; Matsui et al., 2018).

2.3. Compresión de matrices

La compresión de matrices es una técnica ampliamente utilizada en diversas áreas de la computación, como la compresión de imágenes y video. No obstante, en esta tesis el enfoque se centra en su aplicación dentro del contexto del aprendizaje profundo, específicamente en la compresión de modelos grandes de lenguaje (LLMs, por sus siglas en inglés). Estos modelos contienen un número considerable de matrices densas de gran tamaño, particularmente en las capas lineales, cuya compresión resulta fundamental para reducir los requerimientos de almacenamiento y memoria durante la inferencia. En esta sección se proporciona una breve introducción a los LLMs y al papel que desempeñan estas matrices dentro de su arquitectura, seguida de una revisión de las principales técnicas de compresión de matrices relevantes para los objetivos de esta tesis.

2.3.1. Modelos grandes de lenguaje (LLMs)

Los modelos grandes de lenguaje se basan en redes neuronales profundas que pueden contener cientos o incluso miles de millones de parámetros. Estos modelos se entrenan con grandes volúmenes de texto, a partir de los cuales aprenden patrones complejos y relaciones semánticas, adquiriendo así la capacidad de ejecutar diversas tareas de procesamiento de lenguaje natural, como síntesis de texto, traducción automática, respuesta a preguntas y análisis de sentimientos (Raiaan et al., 2024).

La arquitectura central utilizada en los LLMs más avanzados es el *Transformer*, propuesto por Vaswani et al. (2017), cuya principal innovación es permitir la modelación del contexto y la interdependencia entre palabras dentro de una secuencia. Esto lo distingue de arquitecturas anteriores, como las redes recurrentes, que eran menos eficientes en la captura de dependencias a largo plazo.

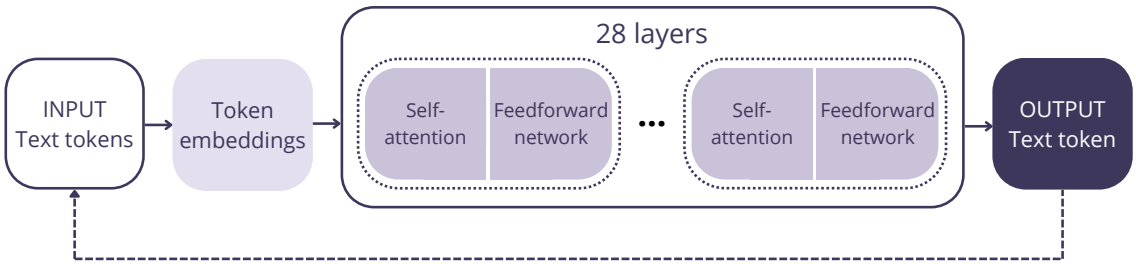


Figura 1. Arquitectura del modelo LLaMA 3.2 de 3 mil millones de parámetros.

Uno de los modelos representativos de esta familia es el *LLaMA 3.2 3B* de Meta, un LLM con arquitectura

Transformer compuesto por 28 capas (Dubey et al., 2024). Este modelo se ha entrenado con una gran diversidad de datos, incluyendo textos provenientes de páginas web, conversaciones, libros, artículos científicos, noticias y código fuente (Raiaan et al., 2024). Su estructura general se presenta en la Figura 1.

La arquitectura *Transformer* se compone de dos bloques fundamentales: el mecanismo de atención (*Self-Attention*) y el perceptrón multicapa (*Multilayer Perceptron*, MLP).

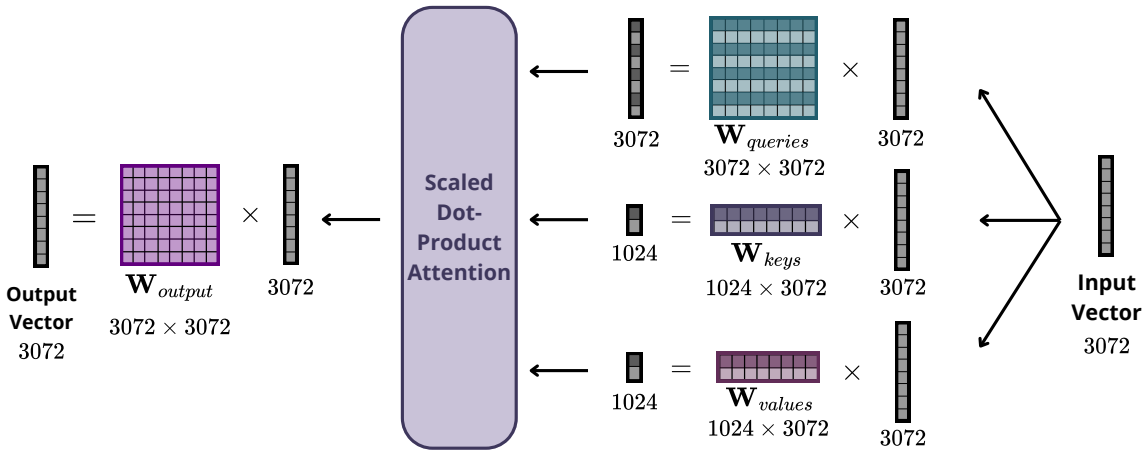


Figura 2. Transformación de un vector en el mecanismo de atención del modelo LLaMA 3.2 3B.

El bloque de atención permite que el modelo determine la relevancia de cada palabra con respecto a las demás en una secuencia. Este mecanismo realiza la siguiente operación:

$$\text{Atención}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V, \tag{4}$$

donde Q, K y V son las matrices de consultas (*queries*), llaves (*keys*) y valores (*values*), respectivamente, y d_k es la dimensión de las llaves. Esta operación puede interpretarse como la búsqueda de una serie de consultas Q sobre una base de datos de llaves K para devolver determinados valores V. La operación $\frac{QK^\top}{\sqrt{d_k}}$ mide la similitud entre las consultas y las llaves; posteriormente, la función *softmax* convierte estas similitudes en probabilidades o pesos de atención normalizados para cada fila. Finalmente, estos pesos ponderan los valores V, generando una salida que representa cuánto debe enfocarse cada token en los demás (Ghojogh & Ghodsi, 2020).

Cada bloque de atención se divide en h cabezas independientes. Cada cabeza i, con $i = 1, \ldots, h$, posee sus propias matrices Q_i, K_i y V_i de dimensión d/h por columna, donde d es la dimensión interna de la atención y h el número de cabezas. En la práctica, y para optimizar el uso de memoria, estas matrices

por cabeza se concatenan para formar las matrices completas W_{queries}, W_{keys} y W_{values} como se muestra en la Figura 2. De esta manera, cada una de estas matrices concatenadas contiene las proyecciones de todas las cabezas, permitiendo operaciones matriciales más eficientes sin perder la correspondencia con las cabezas individuales.

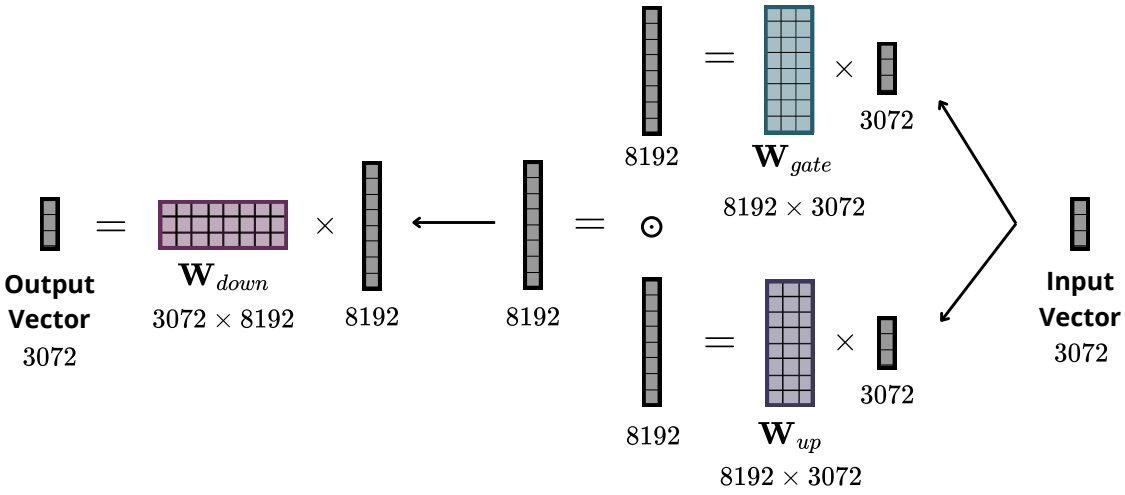


Figura 3. Transformación de un vector en el perceptrón multicapa del modelo LLaMA 3.2 3B.

El bloque MLP permite aumentar la dimensionalidad de los vectores de entrada, lo cual incrementa la capacidad del modelo para representar relaciones no lineales complejas. La operación que realiza este bloque está dada por:

$$Y = (X \cdot W_{\text{up}} \odot \sigma(X \cdot W_{\text{gate}})) \cdot W_{\text{down}}, \tag{5}$$

donde $\odot$ representa la multiplicación elemento a elemento, σ es una función de activación no lineal, y X son los vectores de entrada. Geva et al. (2020) interpreta este bloque como una forma de memoria neural, en la que W_{up} actúa como un conjunto de llaves, y W_{down} como un conjunto de valores. En esta interpretación, el vector de entrada activa un subconjunto de memorias mediante la proyección sobre W_{up}, y la salida se obtiene como una combinación ponderada de los valores correspondientes. Los autores muestran que estas memorias pueden correlacionarse con patrones lingüísticos interpretables, como n-gramas o temas semánticos, y que los valores inducen distribuciones sobre el vocabulario de salida que se correlacionan con las predicciones probables del siguiente token.

Para evaluar el desempeño de estos modelos, es necesario considerar métricas que cuantifiquen su rendimiento funcional tras la compresión. Entre las métricas más utilizadas en la literatura se encuentra

la perplejidad (*perplexity* en inglés), la cual mide la capacidad del modelo para predecir secuencias de texto, siendo especialmente relevante en tareas de modelado de lenguaje. Esta se define mediante la siguiente fórmula:

$$\text{Perplexity} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log p(w_i \mid w_1, \ldots, w_{i-1})\right), \tag{6}$$

donde N representa el número total de palabras en el corpus y $p(w_i \mid w_1, \ldots, w_{i-1})$ es la probabilidad asignada por el modelo a la palabra w_i dado su contexto (Jurafsky & Martin, 2025). Intuitivamente, esta métrica puede interpretarse como una medida del grado de incertidumbre que tiene el modelo al predecir la siguiente palabra: una perplejidad baja indica que el modelo asigna altas probabilidades a las palabras correctas, y por lo tanto es más competente en la tarea de predicción. Formalmente, minimizar la perplejidad es equivalente a maximizar la probabilidad del conjunto de prueba según el modelo.

La perplejidad se evalúa comúnmente en corpus textuales estandarizados como WikiText-2 (Merity et al., 2016), un conjunto de datos derivado de artículos de Wikipedia seleccionados manualmente para eliminar contenido ruidoso como listas, tablas o código HTML. Este corpus contiene aproximadamente 2 millones de palabras y se caracteriza por una estructura lingüística coherente y una distribución léxica rica, lo cual lo convierte en un referente adecuado para tareas de modelado de lenguaje y evaluación de modelos generativos.

Por otro lado, también se emplean métricas que evalúan la capacidad del modelo para razonar y comprender el lenguaje en distintos dominios del conocimiento. Entre ellas destaca el benchmark MMLU (*Massive Multitask Language Understanding*), el cual reúne un conjunto amplio y diverso de tareas de opción múltiple que abarcan más de 50 disciplinas académicas, incluyendo matemáticas, ciencias, humanidades y derecho (Hendrycks et al., 2021). Las preguntas de MMLU fueron elaboradas para cubrir distintos niveles de dificultad, desde secundaria hasta nivel profesional, y están diseñadas para evaluar el razonamiento lógico, el conocimiento factual y la comprensión semántica del modelo.

Estas métricas permiten evaluar la fidelidad funcional de un modelo comprimido con respecto al modelo original y resultan fundamentales para validar la efectividad de los métodos propuestos en este trabajo.

Comprender la arquitectura de los modelos grandes de lenguaje resulta fundamental en el contexto de esta tesis, ya que cada uno de sus componentes (en particular, las matrices de proyección dentro de los bloques de atención y MLP) constituye el principal objetivo de compresión. Como se ilustra en la Figura

4, estas matrices representan la mayor parte de los parámetros del modelo, y por tanto, su compresión puede reducir significativamente el tamaño total sin comprometer su funcionalidad. A continuación se examinan diversas técnicas de compresión dirigidas específicamente a representaciones matriciales, con el objetivo de contextualizar el estado del arte y destacar aquellas metodologías más relevantes en relación con la técnica de compresión propuesta en esta tesis.

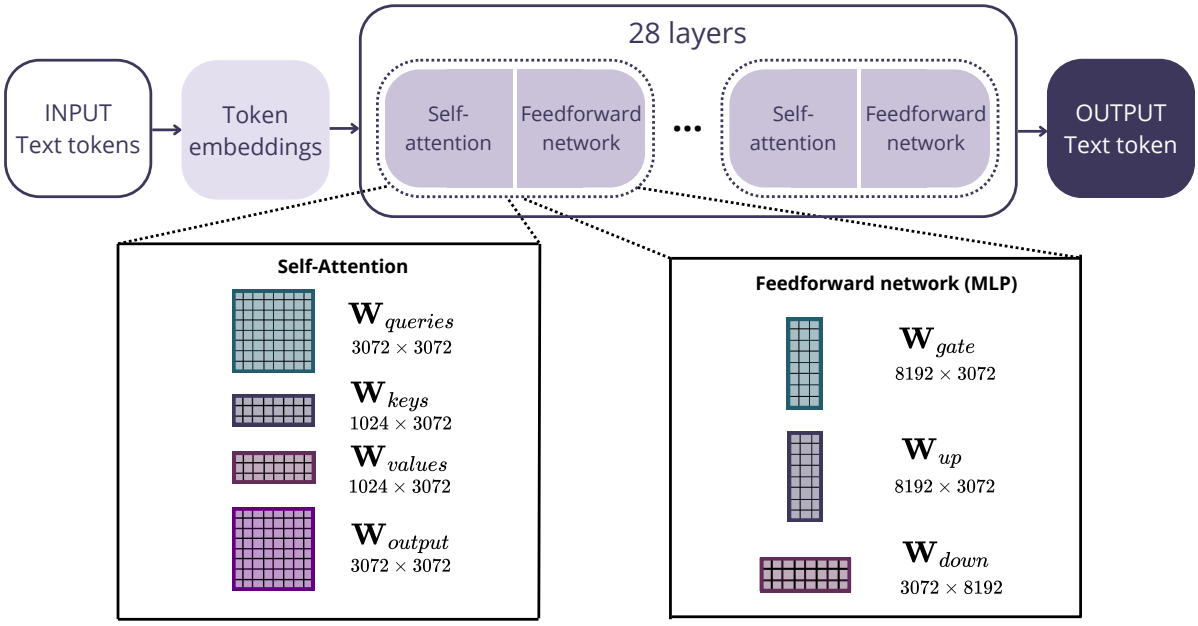


Figura 4. Matrices del modelo LLaMA 3.2 de 3 mil millones de parámetros.

2.3.2. Cuantización

La cuantización es una técnica fundamental en la compresión de matrices, que se enfoca en reducir la precisión de los elementos al representarlos con menos bits. Por ejemplo, esta compresión puede implicar cambiar de 32 bits de punto flotante a 16 bits o incluso a enteros de 8 bits (Rokh et al., 2023). Este proceso disminuye significativamente los requisitos de memoria y mejora la eficiencia computacional sin una pérdida considerable de la precisión de la matriz en muchos casos (Wang et al., 2018).

La cuantización se puede clasificar principalmente en dos categorías: cuantización durante el entrenamiento (*Quantization-Aware Training*, QAT) y cuantización después del entrenamiento *Post-Training Quantization*, PTQ). QAT implica la cuantización de los pesos durante la fase de entrenamiento del modelo. Aunque QAT puede mantener una alta precisión, su principal desventaja es que requiere un re-entrenamiento, a menudo por cientos de épocas, lo que lo hace impráctico para modelos de gran escala

debido al consumo de tiempo y recursos computacionales (Gholami et al., 2022). Por contraste, PTQ comprime modelos preentrenados sin necesidad de acceder al *pipeline* de entrenamiento original (Lang et al., 2024), lo que lo convierte en un enfoque mucho más flexible y eficiente para el despliegue de modelos ya existentes.

Dentro de las técnicas de PTQ, una de las más básicas y ampliamente utilizadas es el método *Round-To-Nearest* (RTN), el cual ejemplifica la esencia de los enfoques de cuantización escalar uniforme. Este método consiste en proyectar cada elemento de la matriz original hacia el nivel de cuantización más cercano dentro de un conjunto predefinido de niveles discretos.

Sea $A \in \mathbb{R}^{m \times n}$ una matriz de valores reales que se desea cuantizar. En primer lugar, los valores de A se dividen en grupos o bloques (los cuales pueden ser renglones, columnas o submatrices de menor tamaño), y a cada uno de estos grupos se le asigna un factor de escala $s \in \mathbb{R}^+$ y un valor de desplazamiento (también llamado *zero-point*) $z \in \mathbb{R}$. Este procedimiento permite adaptar dinámicamente la cuantización a las estadísticas locales de cada grupo, mejorando la capacidad de representar adecuadamente los rangos de valores observados.

El número de niveles de cuantización está directamente determinado por el número b de bits disponibles para representar cada elemento. En consecuencia, el factor de escala s se define en función del rango de valores del bloque y del número total de niveles disponible, que es 2^b. En el caso de cuantización por renglones, el factor de escala para el renglón i se calcula como:

$$s_i = \frac{a_i^{max} - a_i^{min}}{2^b - 1} \tag{7}$$

donde a_i^{min} y a_i^{max}, representan el valor mínimo y máximo, respectivamente, del renglón i. Este valor s_i define la resolución entre niveles consecutivos de cuantización. Para garantizar que el valor mínimo se cuantice al valor entero cero, se introduce un desplazamiento z definido como:

$$z_i = round(-\frac{a_i^{min}}{s_i}) \tag{8}$$

Con estos parámetros, cada elemento a_{ij} se cuantiza como:

$$q_{ij} = \text{round}\left(\frac{a_{ij}}{s_i} + z_i\right). \tag{9}$$

donde $q_{i,j} \in \mathbb{R}$ representa el valor cuantizado. Este valor se almacena como un entero de b bits, restringido al intervalo $\{0, 1, 2, \ldots, 2^b - 1\}$.

Para recuperar una aproximación del valor original, se aplica la operación inversa:

$$\hat{a}_{ij} = s_i \cdot (q_{ij} - z_i)\,, \tag{10}$$

lo cual permite obtener una reconstrucción aproximada de la matriz original a partir de su versión cuantizada. La implementación de este esquema de compresión es extremadamente eficiente, ya que sólo involucra operaciones aritméticas básicas (resta, división y redondeo), y la reconstrucción aproximada de los valores originales se realiza aplicando la fórmula inversa descrita en la ecuación (10), requiriendo únicamente multiplicación y suma.

Este método de cuantización se usa ampliamente en aplicaciones de aprendizaje profundo, especialmente en la cuantización de matrices de pesos de redes neuronales, donde cada bloque de pesos puede tener sus propios parámetros s y z, permitiendo una adaptación más precisa a la distribución de valores de cada subconjunto de datos.

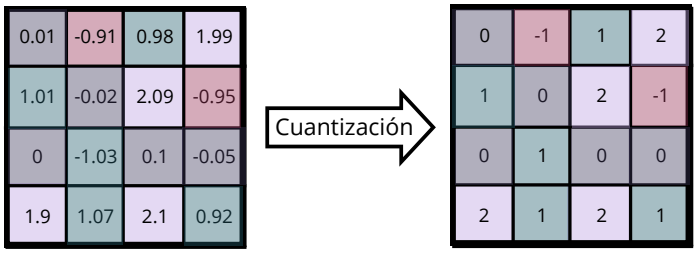


Figura 5. Cuantización de una matriz.

Los métodos recientes de PTQ han avanzado al optimizar los parámetros de cuantización (como factores de escala y desplazamiento) para minimizar el error de reconstrucción, incluso en ausencia de los datos de entrenamiento del modelo original (Nagel et al., 2021). Estos enfoques sin los datos de entrenamiento (*data-free*) típicamente generan datos sintéticos basados en priors estadísticos o heurísticas para aproximar la distribución original de los datos. Por ejemplo, algunos métodos aprovechan las estadísticas de BatchNorm (Cai et al., 2020; Zhang et al., 2021), mientras que otros emplean incrustaciones latentes superpuestas y mapeos auxiliares, como en Qimera Choi et al. (2021). Alternativamente, las técnicas de optimización por capas minimizan el error de cuantización local sin requerir acceso completo al modelo o a muestras de datos (Frantar et al., 2022; Cai et al., 2020). Aunque estas estrategias son efectivas hasta

cierto punto, pueden sufrir de desajuste distribucional o pasar por alto la estructura global del modelo, lo que puede limitar su rendimiento, particularmente a bajas resoluciones de bits.

En el estado del arte, la cuantización se realiza de diversas maneras. Por ejemplo, Zhu et al. (2024) propusieron un método para cuantizar las matrices de pesos de una red neuronal utilizando valores ternarios (-1, 0, 1), lo cual reduce la complejidad computacional. Este enfoque se ha implementado en hardware especializado como FPGAs, mejorando la eficiencia energética y la velocidad de inferencia.

Otra técnica común en la literatura es la cuantización basada en *clustering*. En este enfoque, cada peso de precisión completa se asigna a uno de los niveles de cuantización, similar a cómo cada valor en un proceso de *clustering* se asigna a un grupo. Los pesos que pertenecen al mismo grupo se representan con un único valor, reduciendo así los requisitos de memoria al almacenar el índice del grupo en lugar de los pesos en punto flotante (Rokh et al., 2023).

De manera complementaria, ciertos enfoques recientes destacan la importancia de preservar los pesos más críticos para mantener el desempeño del modelo. Por ejemplo, Yu et al. (2024) propusieron una cuantización basada en RTN asimétrico, en la que los *outliers* se truncan antes de la cuantización y se conserva únicamente un conjunto muy reducido de ellos para la reconstrucción. La retención de esta pequeña cantidad de parámetros, típicamente entre uno y seis por modelo, ha demostrado ser suficiente para sostener de manera significativa la calidad final de la compresión, evidenciando que no todos los pesos contribuyen de igual manera al desempeño del modelo.

A pesar de estos avances, la mayoría de los métodos PTQ aún se basan en transformaciones simples de redondeo (RTN). RTN sirve como la línea base fundacional para la cuantización post-entrenamiento: es *data-free*, *calibration-free* y no requiere optimización más allá del redondeo directo al nivel cuantizado más cercano. Si bien, otros métodos más sofisticados construyen sobre RTN introduciendo calibración por capas, optimización de parámetros o generación de datos sintéticos, la operación central sigue siendo un redondeo local, elemento por elemento. Esto hace de RTN un punto de comparación natural para métodos como el que se desarrolla en esta tesis, que operan de manera similar sin acceso a datos de entrenamiento o ajuste fino adicional, pero que buscan explotar la estructura global para una compresión mejorada. Dado que este trabajo se centra en la compresión de modelos preentrenados sin reentrenamiento ni datos auxiliares, las técnicas sin calibración constituyen las líneas base más relevantes.

2.3.3. Aproximación de rango bajo

La aproximación de rango bajo busca una matriz aproximada de rango reducido que reemplace a la matriz original con el fin de disminuir el número de parámetros y, por ende, el costo computacional y de almacenamiento. Formalmente, dada una matriz $A \in \mathbb{R}^{m \times n}$, el objetivo es encontrar una matriz $\hat{A}$ de rango $r < \min(m, n)$ que minimice la diferencia respecto a A bajo alguna norma, usualmente la norma de Frobenius:

$$\hat{A} = \underset{\mathit{rank}(\hat{A})=r}{\arg\min} \|A - \hat{A}\|_F \tag{11}$$

Una técnica estándar y óptima en este sentido es la descomposición en valores singulares, que factoriza la matriz A como:

$$A = U\Sigma V^\top \tag{12}$$

donde $U \in \mathbb{R}^{m \times m}$ y $V \in \mathbb{R}^{n \times n}$ son matrices ortogonales, y $\Sigma \in \mathbb{R}^{m \times n}$ es una matriz diagonal rectangular, es decir, todos sus elementos fuera de la diagonal son cero, mientras que la diagonal está formada por los valores singulares Σ_{ii}, con $i = 1, \ldots, \min(m, n)$, ordenados de mayor a menor.

La aproximación de rango reducido r se obtiene truncando la SVD a las primeras r componentes:

$$\hat{A} = U_r \Sigma_r V_r^\top \tag{13}$$

donde $U_r \in \mathbb{R}^{m \times r}$ es la submatriz formada por las primeras r columnas de U; $\Sigma_r \in \mathbb{R}^{r \times r}$ es una matriz diagonal que contiene los primeros r valores singulares en orden descendente; y $V_r \in \mathbb{R}^{n \times r}$ corresponde a las primeras r columnas de V.

Este proceso de truncamiento, que se ilustra de manera esquemática en la Figura 6, es óptimo en el sentido de que minimiza la norma de Frobenius del error $\|A - \hat{A}\|_F$ entre todas las matrices de rango r, de acuerdo con el teorema de Eckart-Young-Mirsky (Eckart & Young, 1936; Golub et al., 1987). Así, la calidad de la aproximación depende del ritmo de decaimiento de los valores singulares de A.

En el contexto de compresión, el valor del rango r se selecciona en función de una restricción sobre el

número total de parámetros que se desea almacenar. Dado que la matriz aproximada $\hat{A}$ se representa como el producto $U_r \Sigma_r V_r^\top$, es común evitar el almacenamiento explícito de la matriz diagonal Σ_r. En su lugar, se precomputa el producto $U_r \Sigma_r$, dando lugar a una nueva matriz densa de tamaño $m \times r$. De esta manera, la representación comprimida queda constituida por las matrices $U_r \Sigma_r \in \mathbb{R}^{m \times r}$ y $V_r^\top \in \mathbb{R}^{r \times n}$, lo cual implica un total de

$$p = r(m + n) \tag{14}$$

parámetros almacenados. Esta expresión permite ajustar el valor de r de forma explícita para lograr un compromiso entre el nivel de compresión deseado y la precisión de la aproximación resultante.

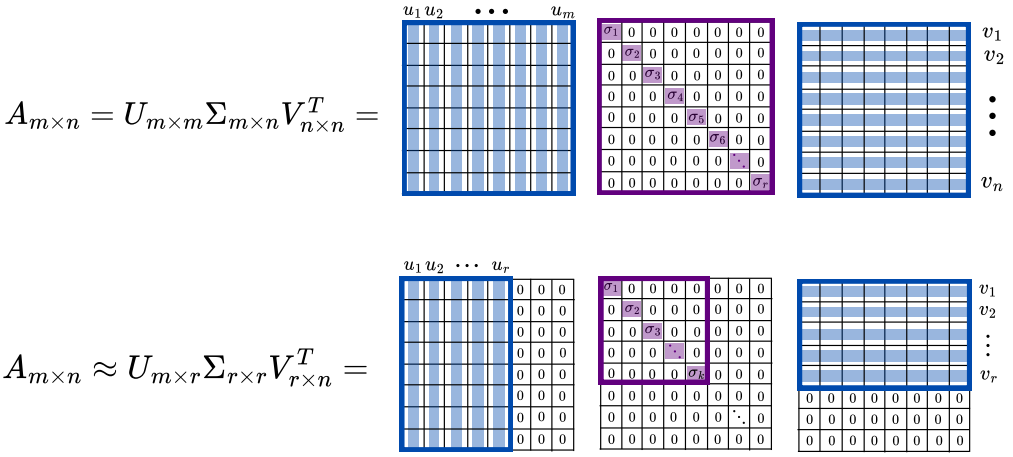


Figura 6. Aproximación de rango bajo de la matriz de pesos mediante SVD.

Diversos trabajos recientes han retomado el principio de aproximación de rango bajo como base para desarrollar métodos más elaborados de compresión de matrices en aplicaciones prácticas. Por ejemplo, en el trabajo de Saha et al. (2023) se propone una técnica que combina la factorización de matrices con cuantización de baja precisión. En este enfoque, se supone que la matriz original A posee estructura de bajo rango y se puede aproximar como $A \approx LR$, donde L y R son matrices de dimensiones reducidas. Posteriormente, los elementos de L y R se cuantizan a un formato de baja precisión, lo que permite una reducción sustancial en el almacenamiento. Los resultados empíricos reportados muestran que esta técnica es efectiva para tareas de compresión de imágenes, clasificación de vecinos más cercanos y en redes neuronales profundas, como LLaMa-7b.

En una línea similar, Li et al. (2023) propusieron LoSparse, una técnica que extiende la idea de aproximación por bajo rango al considerar también componentes dispersos. En este enfoque, cada matriz de

pesos de un modelo grande de lenguaje se representa como la suma de una matriz de bajo rango UV y una matriz dispersa S, es decir:

$$A \approx UV + S \tag{15}$$

La componente de bajo rango UV permite capturar la estructura coherente y dominante de la matriz, mientras que la parte dispersa S captura las partes no coherentes y menos significativas. Al aplicar esta representación híbrida a las distintas capas de los modelos, LoSparse combina de forma efectiva los beneficios de la compresión por rango bajo y la poda estructurada. Los resultados empíricos reportados muestran mejoras significativas con respecto a técnicas convencionales en tareas como comprensión de lenguaje natural, respuesta a preguntas y generación de texto.

Por otro lado, Noach & Goldberg (2020), proponen un enfoque de compresión de modelos en dos etapas para reducir el tiempo de inferencia de modelos preentrenados. Primero, descomponen cada matriz de pesos $W \in \mathbb{R}^{n \times d}$ en dos matrices más pequeñas $A \in \mathbb{R}^{n \times r}$ y $B \in \mathbb{R}^{r \times d}$, de manera que $W' \approx AB$, con r menor que $\frac{nd}{n+d}$. Esta descomposición se realiza en dos fases: inicialmente, utilizando la descomposición en valores singulares truncados (SVD) para obtener una aproximación A' y B' tal que:

$$\|W - A'B'\|_F \tag{16}$$

sea pequeña, y posteriormente se refina usando técnicas de *feature distillation* para conservar el rendimiento del modelo. Este método logra acelerar la inferencia en un factor de $1.45\times$, con pérdidas mínimas en las métricas de desempeño.

En resumen, la aproximación de rango bajo constituye una herramienta eficaz para reducir la complejidad de las matrices densas al capturar su estructura latente mediante descomposiciones matriciales. Su capacidad para conservar la mayor parte de la energía de la matriz original, junto con su bajo costo computacional, la hacen especialmente atractiva en contextos donde el almacenamiento y la velocidad de inferencia son críticos. No obstante, su desempeño puede verse limitado en situaciones donde los datos no presentan un decaimiento pronunciado en sus valores singulares, o donde se requiere una compresión más agresiva. Por esta razón, en la práctica, suele combinarse con otras técnicas complementarias que aprovechan diferentes formas de redundancia en los datos.

2.3.4. Transformada Discreta Coseno

La Transformada Discreta del Coseno (DCT) es una transformación lineal ampliamente utilizada en la compresión de datos y señales debido a su capacidad para concentrar la energía de una señal en unos pocos coeficientes. Para una matriz $A \in \mathbb{R}^{n\times n}$, la DCT bidimensional se define como:

$$\mathsf{DCT}(A) = CAC^{\top} = \tilde{A}, \tag{17}$$

donde $C \in \mathbb{R}^{n\times n}$ es la matriz de transformación, cuyos elementos son combinaciones lineales de funciones coseno evaluadas a distintas frecuencias (Ahmed et al., 1974). Cada componente de C se expresa en función de su fila k y columna m $(k, m = 0, \ldots, n-1)$ como:

$$C_{k,m} = \alpha_k \cos\left(\frac{(2m+1)k\pi}{2n}\right), \tag{18}$$

donde

$$\alpha_k = \begin{cases} \sqrt{\frac{1}{n}}, & \text{si } k = 0, \\ \sqrt{\frac{2}{n}}, & \text{si } k > 0. \end{cases} \tag{19}$$

Esta definición permite interpretar la DCT bidimensional como la aplicación consecutiva de la DCT a cada fila y, posteriormente, a cada columna de la matriz A. La matriz C es ortogonal, es decir, $C^{-1} = C^{\top}$, lo que facilita la recuperación exacta de la matriz original mediante la transformada inversa:

$$A = C^{\top}\tilde{A}C. \tag{20}$$

La aplicación de la DCT redistribuye el contenido de A en el dominio de las frecuencias, concentrando las componentes de baja frecuencia (de mayor magnitud) en la esquina superior izquierda de $\tilde{A}$. Esta característica permite descartar las frecuencias altas, menos significativas, logrando compresión con pérdida (*lossy*) sin degradar notablemente la calidad de reconstrucción. Esta propiedad constituye la base matemática del algoritmo JPEG, en el que la imagen se divide en bloques de 8×8 píxeles y a cada bloque se le aplica la DCT para obtener su representación en el dominio de las frecuencias.

En este contexto, las componentes de alta frecuencia representan detalles finos, como bordes o variaciones pequeñas, que pueden eliminarse sin afectar perceptiblemente la calidad visual. Dado que el ojo humano tiene limitada sensibilidad a estas diferencias sutiles, la eliminación de estas frecuencias se percibe mínimamente, haciendo que la compresión sea eficiente. Por otro lado, las bajas frecuencias capturan las tendencias generales de la imagen, como zonas de color uniforme o gradientes suaves, que constituyen su estructura principal (Benítez López, 2016).

En la literatura, la DCT se ha empleado no sólo en compresión de imágenes y señales, sino también en compresión de modelos neuronales. Por ejemplo, Ulicny et al. (2021) aplican la DCT a filtros convolucionales en CNNs para realizar poda en el dominio espectral, lo que permite reducir la redundancia de parámetros con mínima pérdida de precisión. Validan su método en ResNet-50 y MobileNet-V2, mostrando compresión efectiva sin necesidad de reentrenamiento extenso. Por otro lado, Chen et al. (2016) utilizan la DCT para representar los filtros de redes convolucionales directamente en el dominio de la frecuencia, permitiendo calcular gradientes en ese espacio y reduciendo tanto el uso de memoria como la carga computacional. Más recientemente, Jongho & Hyun (2024) introducen DCT-ViT, una técnica que aplica DCT a modelos Vision Transformer (ViT), realizando poda de tokens de alta frecuencia para disminuir el costo computacional sin sacrificar precisión. Este enfoque logra una reducción de hasta 47 % en los requerimientos computacionales con mínima pérdida de rendimiento, gracias al uso de DCT unidimensional y poda flexible.

En el caso de modelos de lenguaje, Scribano et al. (2023) proponen un mecanismo de atención basado en DCT para arquitecturas Transformer. Su método aplica una DCT sobre el vector de entrada X, generando un vector comprimido $\tilde{X} = DX$, que se utiliza en el módulo de atención. Posteriormente, tras procesarse, se recupera mediante la Inversa de la DCT. Este enfoque es especialmente útil para manejar secuencias largas o de longitud variable, y reduce significativamente los cálculos dentro del mecanismo de atención, mejorando así la eficiencia del modelo.

2.3.5. Compartición de pesos

Una estrategia utilizada dentro de las técnicas de compresión es la compartición de pesos, la cual permite reutilizar eficientemente parámetros entre distintas partes de un modelo, reduciendo así el consumo de memoria y mejorando la eficiencia computacional. Esta técnica se basa en identificar valores numéricos comunes que se pueden compartir por múltiples entradas de la matriz de pesos. En lugar de almacenar

todos los valores individualmente, se mantiene un conjunto reducido de valores representativos, junto con una matriz de índices que indica cuál de estos valores le corresponde a cada posición. Este mecanismo reduce de manera significativa el tamaño efectivo de las matrices de parámetros (Dupuis et al., 2020). La Figura 7 ilustra esquemáticamente el principio de esta técnica.

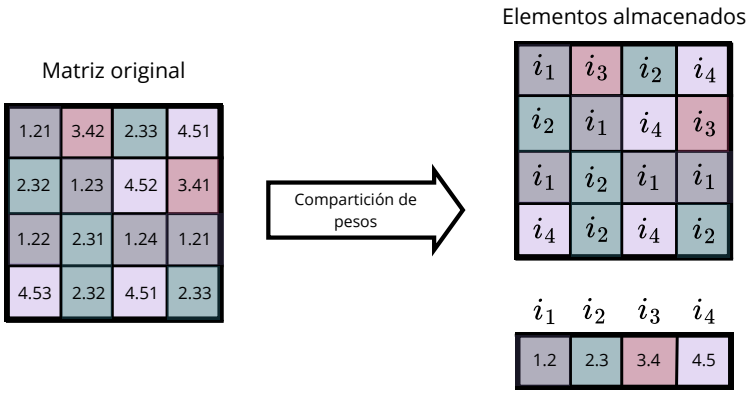


Figura 7. Compartición de pesos de una matriz.

En este contexto, Zechun et al. (2024) introdujeron una técnica denominada "compartición inmediata de pesos por bloques" (*immediate block-wise weight-sharing*), que extiende esta idea al nivel de bloques estructurados dentro del modelo. Su enfoque permite compartir pesos entre bloques adyacentes de la red neuronal sin incrementar el tamaño del modelo y con un impacto mínimo en la latencia de inferencia, mejorando la eficiencia computacional y reduciendo la necesidad de almacenamiento. Esta estrategia es particularmente útil para la implementación en dispositivos móviles, donde los recursos de memoria y procesamiento son limitados.

Capítulo 3. Método de compresión propuesto

El método propuesto tiene como objetivo comprimir conjuntos de datos representados como vectores o matrices reales de dimensión fija. La técnica se fundamenta en la observación de que, al proyectar estos datos sobre una rotación ortonormal aleatoria, es posible revelar una estructura estadística más uniforme que facilita su compresión eficiente en memoria.

En esta sección se presenta una descripción general de la metodología, así como ejemplos concretos de su aplicación en dos escenarios representativos: la compresión de conjuntos de vectores (enfocada en la reducción de bases de datos de vectores para tareas de búsqueda por similitud) y la compresión de conjuntos de matrices (con aplicaciones en modelos de lenguaje de gran escala). Estos casos de uso permiten ilustrar la versatilidad y eficacia del enfoque propuesto en contextos reales de alto impacto.

3.1. Descripción general del método

Sean $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ un conjunto de vectores reales, con $v_i \in \mathbb{R}^d$, o bien un conjunto de matrices reales $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$, donde cada $A_i \in \mathbb{R}^{m \times d}$. La compresión se basa en los siguientes principios:

1. **Rotación ortonormal aleatoria:**

   Se generó una matriz ortonormal $Q \in \mathbb{R}^{d \times d}$ de manera aleatoria, la cual sirvió como transformación común para todos los elementos del conjunto. Esta matriz cumple la propiedad:

   $$Q^\top Q = I, \tag{21}$$

   garantizando la conservación de distancias y normas en el espacio transformado, lo cual es fundamental para preservar las relaciones geométricas originales entre los datos comprimidos.

   El proceso de construcción de la matriz Q consistió en muestrear una matriz $G \in \mathbb{R}^{d \times d}$ con entradas independientes extraídas de una distribución normal estándar, es decir, $G_{ij} \sim \mathcal{N}(0, 1)$. Posteriormente, se aplicó la descomposición QR a G, obteniendo dos matrices Q y R tales que $G = QR$. La matriz ortonormal Q resultante de esta descomposición se utilizó como la rotación compartida entre todos los elementos del conjunto.

2. **Proyección de los datos:**

   - Para cada vector v_i, se calcula su proyección como:

$$u_i = v_i Q^\top. \tag{22}$$

   - Para cada matriz A_i, se calcula su proyección como:

$$D_i = A_i Q^\top. \tag{23}$$

Tras esta rotación, los valores de cada vector o matriz rotado mostraron empíricamente una distribución homogénea al ordenarse de forma ascendente. Este comportamiento resultó ser independiente de la distribución original de los datos sin rotar.

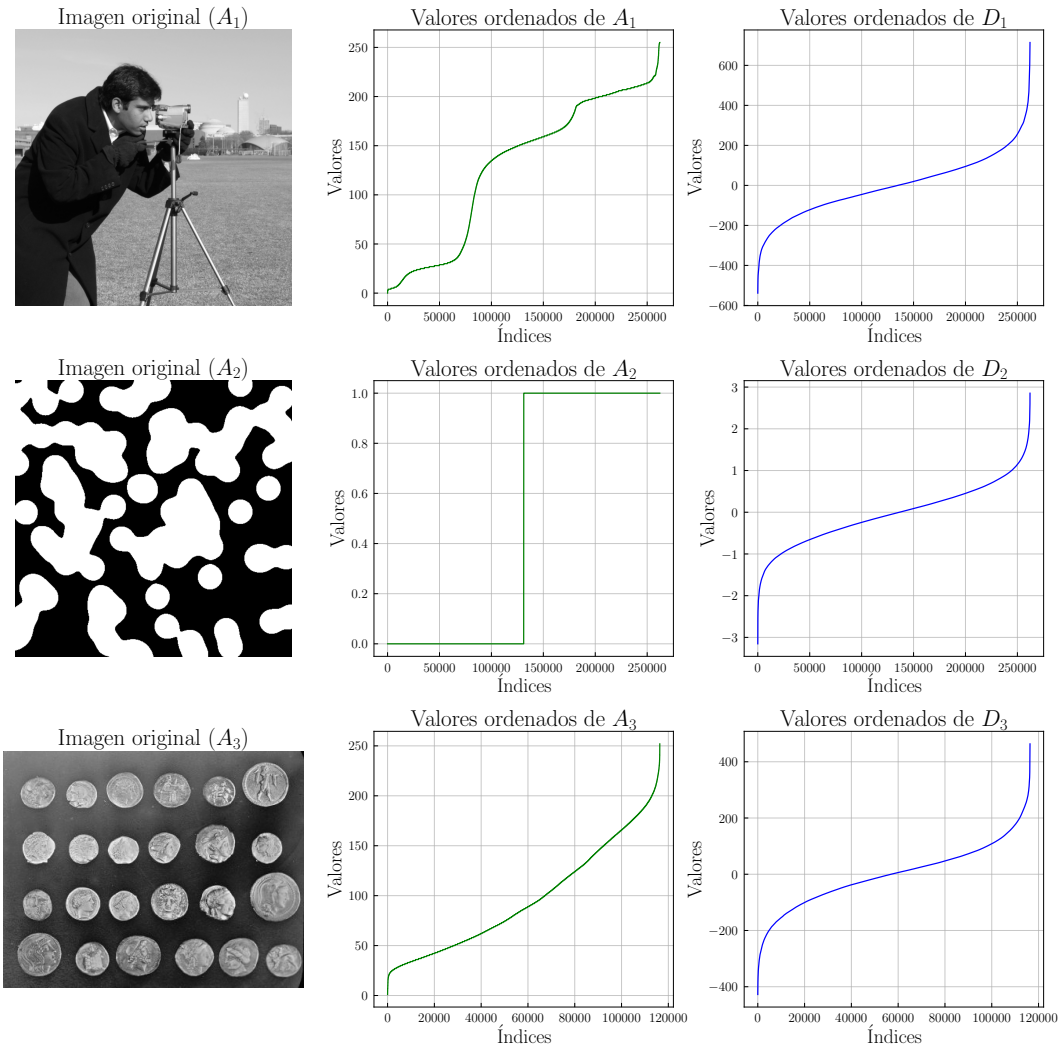


Figura 8. Comparación de la distribución de los elementos ordenados entre las matrices originales y sus contrapartes rotadas. Se observa para tres matrices distintas.

3. **Codificación:**

   Como se observa en la Figura 8, la distribución de los elementos rotados presentó un comportamiento de tipo gaussiano, caracterizado por la presencia de pocos valores de alta magnitud (en valor absoluto) y una gran concentración de elementos con valores cercanos a cero.

   A partir de este patrón se propusieron distintas estrategias de codificación, adaptadas al contexto de aplicación considerado. En este trabajo se estudiaron dos contextos: búsqueda por similitud (vectores) y generación de texto (matrices). Dado que se trata de aplicaciones distintas, sus objetivos también difieren.

   En el caso de la búsqueda por similitud, es fundamental que el método de compresión preserve, en la mayor medida posible, las distancias entre vectores en el espacio. En cambio, en el contexto de la generación de texto, las matrices de los modelos de lenguaje almacenan la información adquirida durante el entrenamiento; por ello, al comprimirlas, es prioritario que la aproximación obtenida sea lo más fiel posible a la matriz original.

   En consecuencia, aunque en ambos casos se sigue un flujo de codificación, almacenamiento eficiente y posterior decodificación, los métodos específicos se ajustan a las particularidades y objetivos de cada aplicación. A continuación se describe la codificación propuesta para cada caso.

3.2. Aplicación del método a vectores: compresión de bases de datos

3.2.1. Codificación de los elementos rotados

En el caso de vectores, como se mencionó previamente, el objetivo principal es preservar con la mayor fidelidad posible las distancias en el espacio original. Para ello, la distribución gaussiana observada se aproximó mediante el algoritmo de Lloyd–Max, el cual, dado un conjunto de datos o una distribución de referencia, determina de forma óptima los umbrales y centroides que minimizan el error cuadrático medio entre la distribución original y su representación cuantizada.

El número de centroides está determinado por el nivel de compresión deseado, expresado en el número de bits b por código, de modo que el cuantizador utiliza 2^b centroides para codificar los valores normalizados.

Para cada vector de la base de datos, se calcularon su media y desviación estándar, parámetros empleados en la normalización individual de cada vector. Estos valores estadísticos se almacenaron para su uso

posterior durante la etapa de decodificación, ya que son necesarios para reconstruir una aproximación fiel del vector original. Una vez normalizados, los elementos se cuantizaron asignándoles el índice correspondiente al centroide más cercano, calculado previamente mediante el algoritmo de Lloyd–Max para una distribución normal estándar. Este índice indica el intervalo definido por los umbrales del cuantizador al que pertenece el valor normalizado.

Con el fin de optimizar el almacenamiento, cada índice se codificó utilizando únicamente b bits, lo que asegura que la representación ocupa la mínima cantidad de memoria necesaria para codificar los 2^b posibles valores. Así, la matriz de códigos resultante requiere significativamente menos espacio que los datos originales en formato de punto flotante.

El procedimiento detallado de codificación se presenta en el Algoritmo 1.

Algoritmo 1: Codificación de base de datos con almacenamiento eficiente

Input: Base de datos $X \in \mathbb{R}^{N \times D}$,
Número de bits por índice b
Output: Centroides $\{c_k\}_{k=0}^{2^b-1}$,
Códigos de centroides I,
Medias $\mu \in \mathbb{R}^N$,
Desviaciones estándar $\sigma \in \mathbb{R}^N$

$\{\beta_k\}_{k=0}^{2^b}, \{c_k\}_{k=0}^{2^b-1} \leftarrow$ Algoritmo de Lloyd-Max(b) ; `// Límites y centroides`
$\mu_i \leftarrow$ media(X_i) para $i = 1, \ldots, N$;
$\sigma_i \leftarrow$ desviación estándar(X_i) para $i = 1, \ldots, N$;
for $i \leftarrow 1$ **to** N **do**
    for $j \leftarrow 1$ **to** D **do**
        $Z_{ij} \leftarrow (X_{ij} - \mu_i)/\sigma_i$; `// Normalización elemento a elemento`
        Encontrar k tal que $\beta_k \leq Z_{ij} < \beta_{k+1}$;
        Asignar $I_{ij} \leftarrow k$;
    end
end
Almacenar I eficientemente usando b bits por elemento;
return $\{c_k\}_{k=0}^{2^b-1}, I, \mu, \sigma$;

La estructura final almacenada consta de: (i) los centroides del cuantizador, (ii) la matriz de códigos con los índices de cada elemento codificado, y (iii) los vectores de medias y desviaciones estándar asociados

a cada vector de la base de datos. Esta representación permite una reconstrucción precisa y eficiente durante la decodificación.

3.2.2. Decodificación

Para recuperar cada vector rotado aproximado, se reemplazan los índices por sus centroides correspondientes y luego se desnormalizan con la media y desviación estándar almacenadas. Finalmente, para obtener una aproximación del vector original, se aplica la matriz de rotación Q:

$$\hat{v}_i = \tilde{u}_i Q, \tag{24}$$

donde $\tilde{u}_i$ es el vector rotado reconstruido.

Formalmente, el proceso de descodificación del vector rotado es el siguiente:

Algoritmo 2: Decodificación de base de datos a partir de índices comprimidos

Input: Códigos de centroides I,
Medias $\mu \in \mathbb{R}^N$,
Desviaciones estándar $\sigma \in \mathbb{R}^N$,
Centroides $\{c_k\}_{k=0}^{2^b-1}$,
Output: Base de datos reconstruida $\hat{X} \in \mathbb{R}^{N \times D}$

for $i \leftarrow 1$ **to** N **do**
  for $j \leftarrow 1$ **to** D **do**
    $k_{ij} \leftarrow I_{ij}$;
    $\hat{X}_{ij} \leftarrow \mu_i + \sigma_i \cdot c_{k_{ij}}$; // $c_{k_{ij}}$ es el centroide correspondiente al código k_{ij}
  end
end
return $\hat{X}$;

3.2.3. Búsqueda por similitud en el espacio rotado

El método propuesto permite realizar búsquedas por similitud directamente en el espacio rotado, sin necesidad de recuperar los vectores originales. Tanto los vectores de la base de datos como la consulta se someten a la misma transformación ortonormal, garantizando que la comparación se realice en un espacio común y consistente. Durante las consultas exhaustivas de los k vecinos más cercanos (k-NN), los productos internos se calculan entre la consulta rotada y los vectores rotados aproximados, obtenidos mediante la decodificación de su representación comprimida. La transformación ortonormal preserva las distancias pareadas en el espacio rotado, mientras que la codificación retiene sus componentes más relevantes, lo que permite estimar con precisión la similitud sin necesidad de revertir al dominio original.

3.2.4. Producto interno acelerado y compresión adicional

Para mejorar la eficiencia en la búsqueda por similitud y aumentar la tasa de compresión global, se incorporó una estrategia basada en el Lema de Johnson-Lindenstrauss, que garantiza la preservación aproximada de las distancias pareadas al proyectar los vectores sobre un subespacio aleatorio de dimensión reducida.

Esta propiedad se aprovechó realizando la búsqueda sobre un subconjunto fijo de coordenadas, seleccionadas aleatoriamente dentro de los vectores rotados. Dicho conjunto de índices se definió una única vez y aplicó uniformemente a todos los vectores durante las consultas. Este enfoque puede interpretarse como una segunda capa de compresión, ya que permite realizar consultas de vecinos más cercanos con alta fidelidad utilizando únicamente una fracción de las dimensiones originales.

Como resultado, se redujo considerablemente la cantidad de datos procesados durante la consulta, incrementando la tasa de compresión efectiva. Simultáneamente, el costo computacional asociado al cálculo de productos internos disminuyó, conduciendo a un proceso de recuperación más rápido y eficiente.

3.3. Aplicación del método a matrices: compresión de pesos en modelos de lenguaje

Para el caso de conjuntos de matrices, el método se aplicó a matrices de pesos extraídas de un modelo

grande de lenguaje, como aquellas correspondientes a capas de atención y perceptrón multicapa. La compresión de cada matriz W_i consistió en su proyección mediante una rotación ortonormal aleatoria, seguida de una codificación adaptativa de sus componentes en el espacio rotado. El método preservó la estructura bidimensional de las matrices durante la codificación, permitiendo una descompresión parcial y eficiente directamente en el proceso de inferencia, sin necesidad de reconstruir la matriz completa en memoria.

Este enfoque tuvo como objetivo reducir el espacio de almacenamiento de los pesos del modelo sin comprometer de manera significativa su desempeño en tareas de evaluación estandarizadas. A continuación, se describe en detalle la metodología empleada para aplicar el método de compresión propuesto a la compresión de los pesos de un LLM.

3.3.1. Aplicación de la rotación ortonormal

Dado un LLM con arquitectura *Transformer*, cuya dimensión de representación interna es n, se genera una única matriz ortonormal aleatoria $Q \in \mathbb{R}^{n \times n}$, común a todo el conjunto de matrices de pesos a comprimir. Si bien el almacenamiento de Q introduce un costo adicional, este se amortiza al emplearse de manera compartida en todas las matrices del modelo.

La rotación se aplica de acuerdo con la orientación de cada matriz en la arquitectura original:

- Para matrices que proyectan desde la dimensión del modelo (por ejemplo, W_{queries}, W_{keys}, W_{values}, W_{output}, W_{gate}, W_{up}), se calcula:

$$D_i = W_i Q^\top \tag{25}$$

- Para matrices que proyectan hacia la dimensión del modelo (por ejemplo, W_{down}), se calcula:

$$D_i = Q^\top W_i \tag{26}$$

Tras la rotación, las matrices D_i exhibieron una distribución aproximadamente gaussiana. Este comportamiento estadístico sugirió que, en el cálculo de productos punto entre matrices, los elementos de mayor magnitud dominan el resultado final. En consecuencia, se decidió extraer dichos valores significativos para almacenarlos sin pérdida, manteniéndolos en su forma original. Para ello, se identificaron los puntos

de cambio de concavidad en la distribución rotada, que funcionaron como umbrales para separar estos valores relevantes del resto de la información.

3.3.2. Codificación de los elementos rotados:

Basándose en la regularidad estadística observada, se diseñó un esquema de codificación mediante segmentación adaptativa que separa explícitamente los *outliers* o valores atípicos de la masa principal de datos con magnitud reducida, favoreciendo una representación comprimida más eficiente.

En primer lugar, se determinaron umbrales globales, denotados como t_{low} y t_{high}, obtenidos del análisis de los puntos de cambio de concavidad en la distribución ordenada de los valores rotados, que delimitaron un rango central conteniendo la mayoría de los componentes con magnitudes bajas, cuya aproximación se considera tolerable para la compresión:

$$t_{low} \leq y \leq t_{high}, \tag{27}$$

donde y representa un elemento individual del vector o matriz rotado.

Dentro de este rango principal, los elementos se cuantizaron uniformemente dividiendo el intervalo $[t_{low}, t_{high}]$ en $2^b - 1$ segmentos de igual tamaño, siendo b el número de bits asignados a codificar cada componente. Cada segmento quedó representado por su valor promedio, almacenado explícitamente. Así, los valores se codificaron mediante índices discretos, lo que redujo el almacenamiento a b bits por componente en esta región.

Por otra parte, los elementos con magnitud fuera del rango definido se clasificaron como valores fuera de rango y se almacenaron por separado en su forma original, asegurando la preservación de la información crítica y evitando pérdidas significativas.

Este esquema permite controlar el equilibrio entre compresión y fidelidad ajustando el parámetro b: un valor más alto aumenta la precisión de cuantización y el tamaño almacenado, mientras que un valor menor reduce el espacio a costa de una aproximación menos precisa.

La Figura 9 ilustra gráficamente este procedimiento, mostrando cómo los datos rotados se dividen en dos subconjuntos: los valores que permanecen dentro de los límites de cuantización y se codifican con

índices discretos de b bits, y los *outliers* que se almacenan de manera explícita y sin pérdida.

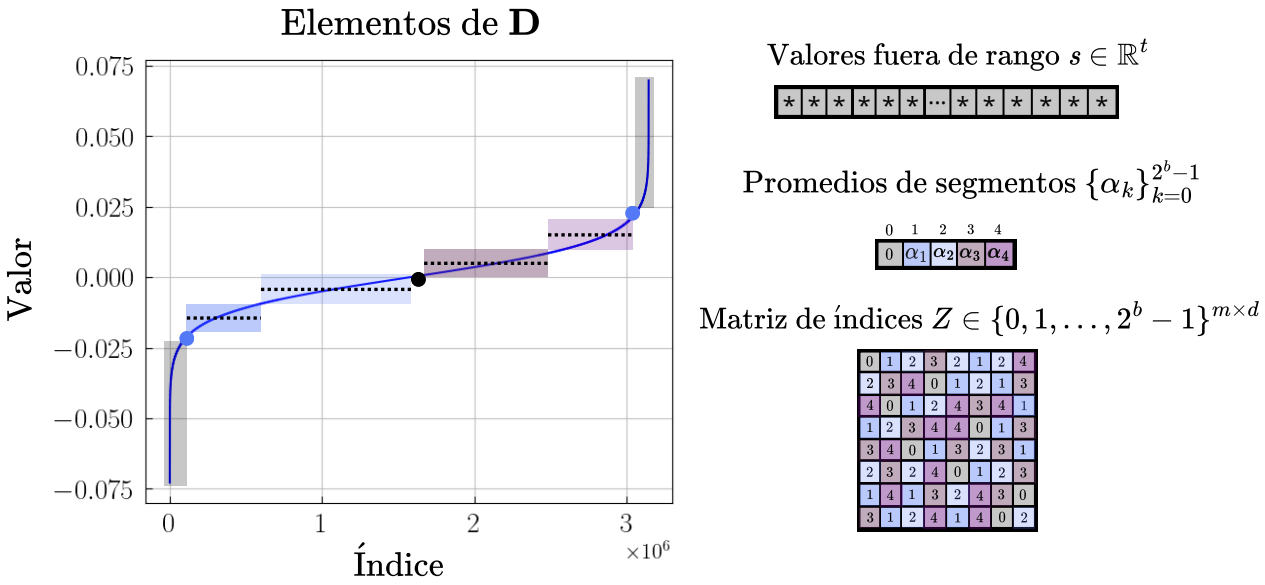


Figura 9. Distribución y esquema de codificación de los elementos de una matriz rotada.

La representación final almacenada incluye:

- La matriz de rotación $Q \in \mathbb{R}^{d \times d}$, común a todo el conjunto, que define la matriz de rotación ortonormal aplicada.
- Un vector $s \in \mathbb{R}^t$ que almacena los t ($t \ll d$) valores correspondientes a las componentes de la matriz rotada que quedaron fuera de los límites definidos por los puntos de concavidad. Estos valores se conservan en su forma original para evitar pérdida de información relevante.
- La codificación compacta de las componentes que permanecieron dentro del rango establecido, representada mediante dos objetos:
  1. Un conjunto de valores promedio $\{\alpha_k\}_{k=1}^{2^b-1}$ representativos de cada segmento.
  2. Una matriz de índices $Z \in \{0, \ldots, 2^b - 1\}^{m \times d}$, donde el índice 0 indica que el valor correspondiente se toma del vector de valores fuera de rango, mientras que un índice $k \geq 1$ hace referencia directa al promedio α_k del conjunto $\{\alpha_k\}$.

El Algoritmo 3 detalla el proceso de compresión aplicado a cada matriz transformada D_i.

Algoritmo 3: Compresión de la matriz D

Input: Matriz $D \in \mathbb{R}^{m \times n}$,
entero b correspondiente al número de bits utilizados para codificar la matriz de índices.
Output: Matriz de índices $Z \in \{0, 1, \ldots, 2^b - 1\}^{m \times n}$,
arreglo de valores promedio $\{\alpha_k\}_{k=0}^{2^b-1}$,
arreglo de valores fuera de rango $s \in \mathbb{R}^t$.

$N \leftarrow 2^b - 1$; `// Número total de códigos disponibles`
Aplanar y ordenar D;
Calcular los puntos de concavidad $\Rightarrow t_{low}, t_{high}$;
Almacenar valores fuera del rango $[t_{low}, t_{high}]$ en el arreglo $s \in \mathbb{R}^t$; `// En el mismo orden en que aparecen en` D `al recorrer la matriz por filas`
Establecer en cero las posiciones fuera de rango en D;
Dividir el intervalo $[t_{low}, t_{high}]$ en subrangos negativos y positivos;
if N *es par* **then**
    Asignar igual número de segmentos a ambos subrangos; `// Distribución simétrica si es posible`
end
else
    Asignar un segmento más al subrango con mayor densidad de valores; `// Prioriza la región con más datos`
end
for *cada segmento en* $[t_{low}, 0)$ *y* $[0, t_{high}]$ **do**
    Calcular el promedio de los valores dentro del segmento α_k;
    Reemplazar todos los valores del segmento con el promedio calculado;
end
Almacenar los valores promedio en un solo arreglo $\{\alpha_k\}$;
Reemplazar valores en D por sus índices correspondientes en $\{\alpha_k\}$;
Codificar los índices usando b bits y almacenar en Z;
return $Z \in \{0, 1, \ldots, 2^b - 1\}^{m \times n}$, $\{\alpha_k\}_{k=0}^{2^b-1}$, $s \in \mathbb{R}^t$

3.3.3. Decodificación

Durante la inferencia, cada matriz aproximada $\tilde{D}$ se reconstruye dinámicamente a partir de su representación comprimida. El Algoritmo 4 describe de manera conceptual la lógica empleada para este proceso.

Aunque este esquema resulta útil para comprender el proceso de reconstrucción, su implementación directa resulta poco eficiente en entornos de cómputo optimizado, tales como unidades de procesamiento gráfico (GPU), unidades tensoriales (TPU) o arquitecturas vectoriales, donde las operaciones secuenciales limitan el aprovechamiento de la paralelización masiva disponible.

Algoritmo 4: Reconstrucción secuencial de la matriz $\widetilde{D}$

Input: Matriz de índices $Z \in \{0, 1, \ldots, 2^b - 1\}^{m \times n}$,
arreglo de valores promedio $\{\alpha_k\}_{k=0}^{2^b-1}$,
arreglo de valores fuera de rango $s \in \mathbb{R}^t$

Output: Matriz reconstruida $\widetilde{D} \in \mathbb{R}^{m \times n}$

```
1  D̃ ∈ ℝ^{m×n};                                   // Inicializar una matriz vacía
2  ℓ ← 0;  // Inicializar contador para recorrer los valores fuera de rango en s
3  for i ← 0 to m − 1 do
4      for j ← 0 to n − 1 do
5          if Z[i, j] = 0 then
6              D̃[i, j] ← s[ℓ];          // Tomar el valor fuera de rango del arreglo s
7              ℓ ← ℓ + 1;             // Avanzar al siguiente valor fuera de rango en s
8          else
9              D̃[i, j] ← α_{Z[i,j]};  // Asignar el valor promedio correspondiente de {α_k}
10         end
11     end
12 end
13 return D̃;                                      // Devolver la matriz reconstruida
```

Por esta razón, en la práctica se adopta una versión paralelizada basada en operaciones vectorizadas, como se ilustra en el Algoritmo 5. Este enfoque permite reconstruir bloques completos de la matriz de manera simultánea, explotando las capacidades de cómputo paralelo que ofrecen bibliotecas como PyTorch, mediante técnicas de difusión de tensores (*broadcasting*) y operaciones por lotes.

Este enfoque evita la necesidad de descomprimir cada elemento de forma secuencial y reduce significativamente el tiempo de reconstrucción, permitiendo integrar la descompresión directamente en el flujo de inferencia del modelo.

Algoritmo 5: Reconstrucción paralelizada de la matriz $\widetilde{D}$

Input: Matriz de índices $Z \in \{0, 1, \ldots, 2^b - 1\}^{m \times n}$,
arreglo de valores promedio $\{\alpha_k\}_{k=0}^{2^b-1}$,
arreglo de valores fuera de rango $s \in \mathbb{R}^t$

Output: Matriz reconstruida $\widetilde{D} \in \mathbb{R}^{m \times n}$

$\widetilde{D} \in \mathbb{R}^{m \times n}$; // Inicializar matriz vacía

$\widetilde{D} \leftarrow \alpha_Z$; // Broadcasting: asigna promedio según índice en Z

$M \leftarrow (Z = 0)$; // Máscara booleana: verdadero donde $Z[i, j] = 0$

$\widetilde{D}[M] \leftarrow s$; // Inserta s en las posiciones verdaderas de M siguiendo su recorrido por filas

return $\widetilde{D}$; // Devuelve matriz reconstruida

Cabe señalar que, en la práctica, no es estrictamente necesario reconstruir explícitamente la matriz $\hat{A}$. Las operaciones de inferencia pueden reformularse como:

$$\mathbf{y} = \tilde{D}(Q\mathbf{x}) \quad \text{o} \quad \mathbf{y} = Q(\tilde{D}\mathbf{x}), \tag{28}$$

según la orientación de la matriz original. Esta estrategia ofrece una ventaja computacional significativa. Si la matriz $\hat{A}$ fuera reconstruida de manera explícita como $\hat{A} = \tilde{D}Q$, la inferencia requeriría un costo computacional del orden de $O(mn^2 + mn)$, correspondiente a la multiplicación matricial $\tilde{D}Q$ seguida de la multiplicación por el vector de entrada $\mathbf{x}$. En contraste, la reformulación directa permite realizar dos productos matriz-vector con un costo total de $O(mn + n^2)$, eliminando la necesidad de una costosa multiplicación entre matrices y reduciendo sustancialmente el número total de operaciones.

3.3.4. Proceso de inferencia con matrices comprimidas

Durante la fase de inferencia, las matrices comprimidas se mantuvieron en su forma compacta a lo largo de todo el proceso, evitando su reconstrucción explícita en memoria. En lugar de almacenar matrices densas $\hat{A}_i$, cada capa del modelo almacenó la matriz $\tilde{D}_i$ en forma compacta. En esta representación, cada matriz comprimida $\tilde{D}_i$ se almacenó mediante un diccionario que contenía tres componentes: el arreglo de valores fuera de rango s, el arreglo de valores promedio $\{\alpha_k\}$ y la matriz de índices Z. Este

diccionario constituyó la versión compacta de la matriz $\tilde{D}_i$, reduciendo significativamente el espacio de almacenamiento en comparación con la matriz densa original. La operación de inferencia se realizó conforme a la ecuación (28), es decir, aplicando la transformación de rotación seguida de la multiplicación con la matriz comprimida, o en el orden inverso, dependiendo de la orientación de la matriz original en la arquitectura del modelo.

Para ello, se diseñaron implementaciones personalizadas de las capas lineales que decodificaron la matriz $\tilde{D}$ al vuelo durante su operación `forward`. La matriz de rotación Q se almacenó una sola vez en memoria y se compartió entre todas las capas que utilizaron esta implementación, lo que permitió aplicarla a los vectores de entrada en cada paso de inferencia sin necesidad de duplicarla para cada capa.

Este esquema se integró en la arquitectura del modelo mediante módulos personalizados en PyTorch. Las capas lineales comprimidas se definieron como subclases de `torch.nn.Module`, incorporando la decodificación de $\tilde{D}$ dentro de su método `forward`. Además, la matriz Q se mantuvo como un tensor compartido entre capas, evitando su replicación en memoria.

Gracias a este diseño, la inferencia se realizó directamente sobre las representaciones comprimidas, sin necesidad de reconstruir previamente las matrices densas, lo que garantizó eficiencia computacional y ahorro de memoria durante la ejecución. Este método permitió ejecutar modelos de gran tamaño de manera eficiente, tanto en uso de memoria como en cómputo, resultando especialmente útil en entornos con recursos limitados, sin afectar la precisión del modelo.

Las Figuras 10 y 11 ilustran este esquema aplicado a un mecanismo de atención y a un bloque MLP, respectivamente. En estos diagramas se observa cómo el modelo permaneció en formato comprimido durante la inferencia y cómo la decodificación de $\tilde{D}$ se realizó al vuelo.

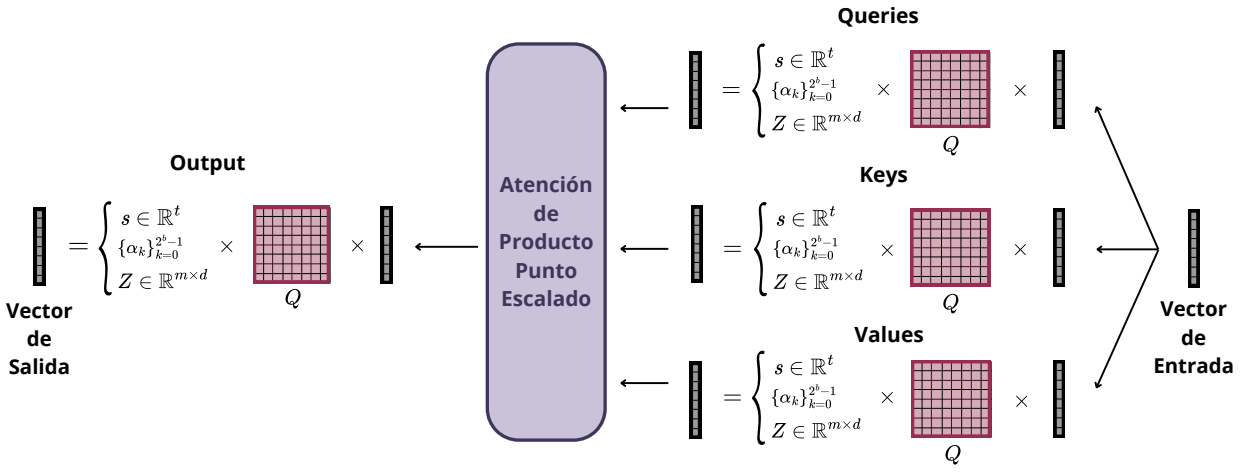


Figura 10. Diagrama del proceso de inferencia en un bloque de mecanismo de atención con matrices comprimidas.

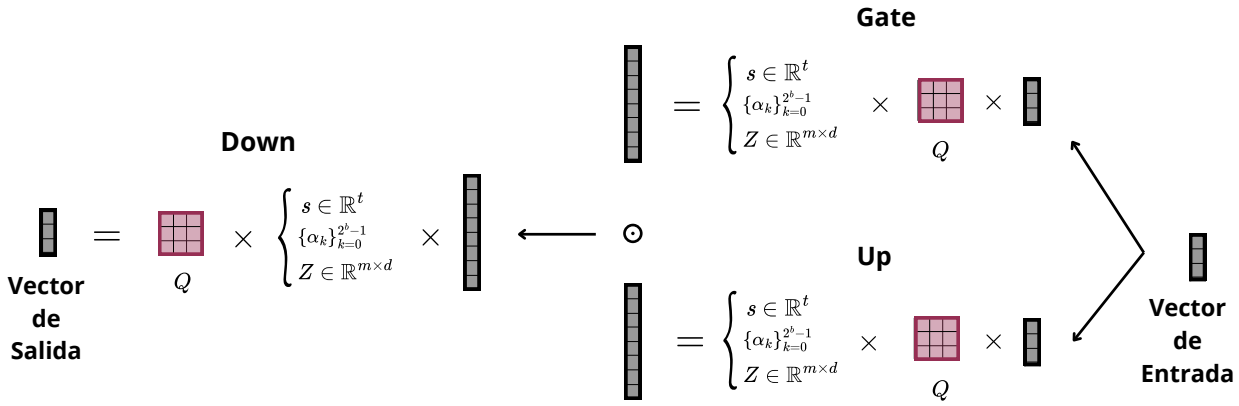


Figura 11. Diagrama del proceso de inferencia en un bloque de perceptrón multicapa con matrices comprimidas.

Capítulo 4. Diseño experimental y resultados

4.1. Evaluación de la compresión en conjuntos de vectores (bases de datos)

Como se mencionó en el capítulo anterior, el método de compresión propuesto (al cual se denomina OrthoQuant) es aplicable a la compresión de conjuntos de vectores. Con el fin de evaluar esta hipótesis, se planteó su aplicación en un escenario representativo: la compresión de bases de datos de vectores. El objetivo principal de esta evaluación es determinar la capacidad del método para reducir de manera eficiente el tamaño de dichas bases, sin comprometer significativamente su utilidad en tareas de búsqueda por similitud. Para ello, se consideró tanto la preservación de la estructura espacial de los vectores como la escalabilidad del método a espacios de mayor dimensionalidad.

4.1.1. Conjuntos de datos utilizados

Para esta evaluación se utilizó el conjunto de datos `CCNEWS`, el cual contiene artículos provenientes de sitios de noticias de todo el mundo, redactados en idioma inglés[1]. Este conjunto está conformado por pares título-artículo, a partir de los cuales es posible obtener representaciones vectoriales utilizando modelos como *Sentence Transformers*. Dichos modelos están diseñados para generar *embeddings* que preserven la similitud semántica entre textos, de manera que artículos similares se representen mediante vectores cercanos en el espacio, mientras que artículos distintos se proyecten a vectores más alejados. Las representaciones vectoriales empleadas en este trabajo se extrajeron del conjunto de datos proporcionado para el desafío SISAP 2025, el cual incluye un total de 614,664 *embeddings* de 384 dimensiones generados mediante el modelo `all-MiniLM-L6-v2`[2][3]. Para las consultas de prueba (*queries*), se utilizaron muestras fuera de distribución proporcionadas en el mismo conjunto, conformadas por 11,000 vectores de consulta.

Con el propósito de evaluar la escalabilidad del método propuesto en espacios de mayor dimensionalidad, se generaron también representaciones vectoriales de 1024 dimensiones para los mismos artículos del conjunto `CCNEWS`, utilizando en este caso el modelo `all-roberta-large-v1`[4]. Para las consultas fuera

[1] `https://huggingface.co/datasets/sentence-transformers/ccnews/`
[2] `https://huggingface.co/datasets/sadit/SISAP2025/blob/main/benchmark-dev-ccnews.h5`
[3] `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`
[4] `https://huggingface.co/sentence-transformers/all-roberta-large-v1`

de distribución correspondientes a este escenario, se extrajeron 11,000 ejemplos del conjunto de datos *Yahoo Answers*[5]. Los *embeddings* de 1024 dimensiones generados a partir de este proceso han sido puestos a disposición pública con fines de replicabilidad y para fomentar futuras investigaciones[6].

Adicionalmente, dado que los *embeddings* de los conjuntos anteriores se obtienen mediante modelos de redes neuronales y suelen tener una distribución aproximadamente gaussiana, se utilizó también el conjunto de datos GIST-1M para evaluar el método en un espacio vectorial con características diferentes (Jégou et al., 2011)[7]. Este conjunto contiene un millón de vectores de dimensión 960 y un conjunto de prueba de mil vectores. Los vectores GIST no son *embeddings* aprendidos, sino que se construyen mediante un método determinista basado en filtros orientados y multiescala aplicados a imágenes. En concreto, cada imagen se divide en bloques y en cada uno se calcula la energía (la magnitud promedio de la respuesta a estos filtros) que captura información sobre texturas y estructuras espaciales. Estas energías se concatenan para formar los vectores, cuya distribución no sigue un patrón gaussiano, a diferencia de los *embeddings* típicos generados por redes neuronales.

Tabla 1. Conjuntos de datos utilizados para la evaluación.

Conjunto	Dimensión	# Vectores	# Consultas	Tamaño (MB)	Distribución
CCNEWS	384	614 664	11 000	927.23	Gaussiana
CCNEWS	1024	614 664	11 000	2 517.66	Gaussiana
GIST	960	1 000 000	1 000	3 839.96	No gaussiana

4.1.2. Métodos y configuración experimental

Adicionalmente, con el fin de evaluar adecuadamente la eficacia de OrthoQuant, se le comparó con técnicas de compresión de bases de datos ampliamente utilizadas en la literatura. Dichas técnicas son la cuantización escalar (Scalar Quantization, SQ) y la cuantización por productos (Product Quantization, PQ). Ambos métodos se encuentran implementados en la biblioteca `faiss`, cuyas implementaciones se utilizaron en este trabajo[8]. Con el objetivo de realizar una comparación justa, se procuró que los tres métodos alcanzaran niveles de compresión equivalentes. Para ello, se exploraron diversas configuraciones posibles de cada técnica y se determinó que es posible obtener compresiones comparables en los casos en que el ancho promedio de bits por elemento sea de aproximadamente 4, 6 y 8 bits. En los escenarios

[5] `https://huggingface.co/datasets/sentence-transformers/yahoo-answers`
[6] `https://huggingface.co/datasets/ScarlettMagdaleno/ccnews-embeddings-dim1024`
[7] `https://huggingface.co/datasets/open-vdb/gist-960-euclidean`
[8] `https://github.com/facebookresearch/faiss/wiki`

de 1 y 2 bits, la comparación sólo fue posible entre OrthoQuant y PQ, dado que la implementación de SQ en `faiss` no contempla configuraciones en esos rangos de cuantización.

4.1.3. Compresión y ancho promedio de bits

En el caso de SQ, el proceso es relativamente directo, ya que consiste simplemente en aplicar una cuantización escalar que utilice la cantidad deseada de bits por coordenada. Así, únicamente es necesario almacenar los códigos de cada vector junto con el *codebook* correspondiente.

Tabla 2. Tamaños de los índices generados por cada método y su porcentaje de compresión respecto al conjunto CCNEWS-384 (927,23 MB). Se incluye el valor estimado de bits por dimensión (*wbits*).

Método	Tamaño (MB)	Compresión (%)	wbits
OrthoQuant	33.81	96.35	1.17
PQ	29.37	96.83	1.01
OrthoQuant	62.78	93.23	2.17
PQ	58.35	93.71	2.01
OrthoQuant	120.73	86.98	4.17
PQ	116.30	87.46	4.01
SQ	115.93	87.50	400
OrthoQuant	178.68	80.73	6,17
PQ	180.15	80.57	6.22
SQ	173.95	81.24	6.00
OrthoQuant	236.64	74.48	8.17
PQ	232.20	74.96	8.01
SQ	232.20	74.96	8.01
Original	927.23	–	32.00

Por otro lado, el método OrthoQuant requiere almacenar los códigos de cada vector, así como un *codebook* precalculado correspondiente a una distribución normal estándar (media cero y desviación estándar uno), el cual contiene únicamente 2^b centroides, donde b es el número de bits deseado. Adicionalmente, se almacenan la media y la desviación estándar de cada vector en precisión de 32 bits. Esto introduce una sobrecarga adicional que, sin embargo, se amortiza conforme aumenta la dimensionalidad de los vectores. En el caso particular de las bases de datos utilizadas, con dimensiones de 384, 1024 y 960, esta sobrecarga corresponde a 0.17, 0.06 y 0.07 bits por elemento, respectivamente. Cabe destacar que estos bits son adicionales a los utilizados por los códigos del vector de índices correspondiente. Esta compensación se refleja en las Tablas 2, 3 y 4 que presentan los niveles de compresión alcanzados para

los conjuntos de datos de dimensión 384, 1024 y 960, respectivamente.

Tabla 3. Tamaños de los índices generados por cada método y su porcentaje de compresión respecto al conjunto CCNEWS-1024 (2517.66 MB). Se incluye el valor estimado de bits por dimensión (*wbits*).

Método	Tamaño (MB)	Compresión (%)	wbits
OrthoQuant	83.59	96.68	1.06
PQ	79.73	96.83	1.01
OrthoQuant	162.27	93.55	2.06
PQ	158.40	93.71	2.01
OrthoQuant	319.63	87.30	4.06
PQ	315.76	87.46	4.01
SQ	314.77	87.50	4.00
OrthoQuant	476.98	81.05	6.06
PQ	488.84	80.58	6.21
SQ	472.32	81.24	6.00
OrthoQuant	634.33	74.80	8.06
PQ	630.46	74.96	8.01
SQ	630.46	74.96	8.01
Original	2517.66	–	32.00

Tabla 4. Tamaños de los índices generados por cada método y su porcentaje de compresión respecto al conjunto GIST (3839.96 MB). Se incluye el valor estimado de bits por dimensión (*wbits*).

Método	Tamaño (MB)	Compresión (%)	wbits
OrthoQuant	128.00	96.67	1.07
PQ	120.98	96.85	1.01
OrthoQuant	248.00	93.54	2.07
PQ	240.98	93.72	2.01
OrthoQuant	488.00	87.29	4.07
PQ	480.98	87.47	4.01
SQ	480.06	87.50	4.00
OrthoQuant	727.99	81.04	6.07
PQ	735.72	80.84	6.13
SQ	720.24	81.24	6.00
OrthoQuant	967.99	74.79	8.07
PQ	960.97	74.97	8.01
SQ	960.97	74.97	8.01
Original	3839.96	–	32.00

En el caso de PQ, se exploraron distintas configuraciones combinando el número de bits por código y la partición de los vectores en subvectores, con el fin de aproximar los niveles deseados de compresión de 1, 2, 4, 6 y 8 *wbits* por elemento. La cuantización por producto implica una sobrecarga adicional asociada al almacenamiento de los *codebooks*, cuya dimensión crece exponencialmente con la cantidad

de bits asignados por código, dado que cada *codebook* debe contener 2^b centroides por subvector, donde b representa el número de bits. Este efecto se vuelve más notable en configuraciones de mayor precisión, como aquellas que emplean 12 bits por código. En todos los casos, se procuró que el promedio de bits por elemento (considerando tanto los códigos como la sobrecarga asociada al almacenamiento del *codebook*) fuera comparable con el de los otros métodos evaluados. La Tabla 5 resume las configuraciones específicas utilizadas para cada nivel de compresión.

Tabla 5. Configuraciones utilizadas para PQ según el objetivo de compresión deseado. El parámetro d es la dimensión original del vector.

Subdimensión	**# Subvectores**	**Bits por código**	**wbits** aproximado
8	$d/8$	8	1
4	$d/4$	8	2
2	$d/2$	8	4
2	$d/2$	12	6
1	d	8	8

4.1.4. Error de reconstrucción

A continuación, se reportan las métricas de error de reconstrucción (mínimo, promedio, máximo y desviación estándar) aplicadas sobre la totalidad de los vectores de las bases de datos, obtenidas para los tres métodos de compresión evaluados, considerando distintos niveles de compresión y las tres bases de datos utilizadas. El error de reconstrucción se calculó conforme a la Ecuación 29, la cual define el error relativo de reconstrucción como la razón entre la norma del error absoluto y la norma del vector original:

$$\text{Error de reconstrucción} = \frac{\|v - \hat{v}\|}{\|v\|} = \frac{\sqrt{\sum_{i=1}^{d}(v_i - \hat{v}_i)^2}}{\sqrt{\sum_{i=1}^{d} v_i^2}} \tag{29}$$

donde v representa el vector original y $\hat{v}$ es el vector aproximado obtenido después del proceso de compresión y posterior descompresión. Esta medida toma valores mayores o iguales a cero, donde un valor de 0 indica una reconstrucción idéntica al vector original. Valores superiores a 1 corresponden a casos en los que el error de reconstrucción excede la norma del vector original, lo cual puede ocurrir en reconstrucciones con errores significativamente elevados.

A partir de este error relativo de reconstrucción, calculado para cada uno de los vectores, se obtuvieron el error mínimo (el menor valor en la base de datos), el error medio o promedio, y el error máximo registrado.

La desviación estándar indica qué tanto se alejan, en promedio, los errores individuales respecto al valor medio, permitiendo así evaluar la dispersión del error. Con estas tablas se obtiene una visión clara del error introducido por cada uno de los métodos de compresión.

Las Tablas 6, 7 y 8 corresponden a las bases de datos de dimensión 384, 1024 y 960, respectivamente. En todas ellas se observó un patrón de comportamiento consistente: aunque OrthoQuant tiende a presentar un error de reconstrucción superior frente a PQ en resoluciones bajas (1 y 2 bits), dicho error disminuye conforme se incrementa la resolución, de modo que, para 8 *wbits*, OrthoQuant exhibe el menor error promedio en los tres conjuntos de datos evaluados. Las tablas con todos los errores de reconstrucción para OrthoQuant en las ocho resoluciones se incluyen en el Anexo A.

Tabla 6. Estadísticas del error de reconstrucción (norma Euclidiana normalizada) para vectores de dimensión 384 comprimidos mediante OrthoQuant, PQ y SQ. Se reportan el mínimo, promedio, máximo y desviación estándar del error respecto a la base original.

Método	wbits	mín	media	máx	std
OrthoQuant	1.17	0.5347	0.6011	0.6703	0.0143
PQ	1.01	0.0570	0.5168	0.6305	0.0627
OrthoQuant	2.17	0.2916	0.3415	0.4145	0.0131
PQ	2.01	0.0401	0.2877	0.3658	0.0343
OrthoQuant	4.17	0.0817	0.0971	0.1805	0.0065
PQ	4.01	0.0230	0.0875	0.1673	0.0104
SQ	4.00	0.1502	0.1672	0.1867	0.0039
OrthoQuant	6.17	0.0285	0.0320	0.0830	0.0009
PQ	6.22	0.0003	0.0247	0.0980	0.0059
SQ	6.00	0.0353	0.0398	0.0439	0.0009
OrthoQuant	8.17	0.0080	**0.0089**	0.0409	0.0002
PQ	8.01	0.0061	0.0155	0.1152	0.0058
SQ	8.01	0.0086	0.0098	0.0109	0.0002

Además, en la mayoría de los casos, la desviación estándar del error de reconstrucción en OrthoQuant resulta menor o igual a la de los métodos comparados, lo que sugiere una mayor estabilidad en su desempeño. Un aspecto particularmente destacable en la Tabla 8 es que la distancia máxima observada en toda la base de datos cuantizada mediante OrthoQuant es considerablemente menor que la registrada con los demás métodos: para la resolución de 1 bit, alcanza un error relativo máximo de 0.6429, mientras que en los métodos comparativos dicho valor supera los 2.00, llegando incluso hasta 85.0626. Esto indica que, si bien los métodos de referencia pueden presentar un buen desempeño en promedio (según su distancia media), los errores de compresión en los casos extremos pueden ser mucho más severos.

Tabla 7. Estadísticas del error de reconstrucción (norma Euclidiana normalizada) para vectores de dimensión 1024 comprimidos mediante OrthoQuant, PQ y SQ. Se reportan el mínimo, promedio, máximo y desviación estándar del error respecto a la base original.

Método	wbits	mín	media	máx	std
OrthoQuant	1.06	0.5591	0.6020	0.6414	0.0088
PQ	1.01	0.0615	0.5130	0.6344	0.0645
OrthoQuant	2.06	0.3085	0.3420	0.3825	0.0080
PQ	2.01	0.0427	0.2863	0.3631	0.0344
OrthoQuant	4.06	0.0864	0.0972	0.1435	0.0041
PQ	4.01	0.0251	0.0870	0.1398	0.0103
SQ	4.00	0.1556	0.1665	0.1772	0.0024
OrthoQuant	6.06	0.0296	0.0320	0.0576	0.0005
PQ	6.21	0.0003	0.0246	0.0667	0.0056
SQ	6.00	0.0369	0.0396	0.0423	0.0006
OrthoQuant	8.06	0.0083	**0.0089**	0.0274	0.0002
PQ	8.01	0.0056	0.0158	0.0767	0.0050
SQ	8.01	0.0091	0.0098	0.0104	0.0001

Tabla 8. Estadísticas del error de reconstrucción (norma Euclidiana normalizada) para vectores de dimensión 960 comprimidos mediante OrthoQuant, PQ y SQ. Se reportan el mínimo, promedio, máximo y desviación estándar del error respecto a la base original.

Método	wbits	mín	media	máx	std
OrthoQuant	1.07	0.5601	0.6036	0.6429	0.0074
PQ	1.01	0.1427	0.2304	29.3499	0.0631
OrthoQuant	2.07	0.3091	0.3402	0.3777	0.0070
PQ	2.01	0.0949	0.1397	20.7037	0.0442
OrthoQuant	4.07	0.0855	0.0961	0.1303	0.0032
PQ	4.01	0.0300	0.0514	13.2306	0.0282
SQ	4.00	0.0620	0.2108	85.0626	0.1942
OrthoQuant	6.07	0.0297	0.0320	0.0462	0.0005
PQ	6.13	0.0065	0.0152	2.6938	0.0094
SQ	6.00	0.0150	0.0521	19.9762	0.0423
OrthoQuant	8.07	0.0083	**0.0089**	0.0134	0.0001
PQ	8.01	0.0023	0.0119	7.2064	0.0163
SQ	8.01	0.0036	0.0129	4.7272	0.0100

Un aspecto que llama la atención en los primeros dos niveles de compresión es que PQ alcanza un menor error de reconstrucción promedio, lo cual podría parecer contraintuitivo si se considera su principio de funcionamiento. En PQ, los subvectores se reemplazan por un único subvector representativo del clúster al que pertenecen, lo que, en principio, debería introducir una mayor distorsión. Sin embargo, esta aparente contradicción se explica por la configuración específica empleada para permitir una comparación justa con OrthoQuant. En particular, al utilizar subvectores de dimensión pequeña con 8 o 12 bits por

código, se dispone de $2^8 = 256$ o $2^{12} = 4096$ centroides por subvector, lo que reduce significativamente el error de reconstrucción.

No obstante, estas configuraciones no son representativas del uso habitual de PQ en escenarios de compresión agresiva, donde se suelen emplear subvectores de mayor dimensión (por ejemplo, 16 o incluso 32) acompañados de códigos de 4 bits ($2^4 = 16$ centroides), con el objetivo de reducir tanto el tamaño como la latencia de las operaciones de *lookup*. En las tablas previamente mencionadas puede observarse que, incluso utilizando subvectores de tamaño 8 con códigos de 8 bits (configuración correspondiente a una resolución de 1 *wbits*), ya se presenta un error de reconstrucción considerable. Por lo tanto, en configuraciones más típicas (con códigos de menor precisión y subvectores más grandes) la degradación del error sería aún más marcada.

4.1.5. Evaluación de búsqueda por similitud en el espacio rotado

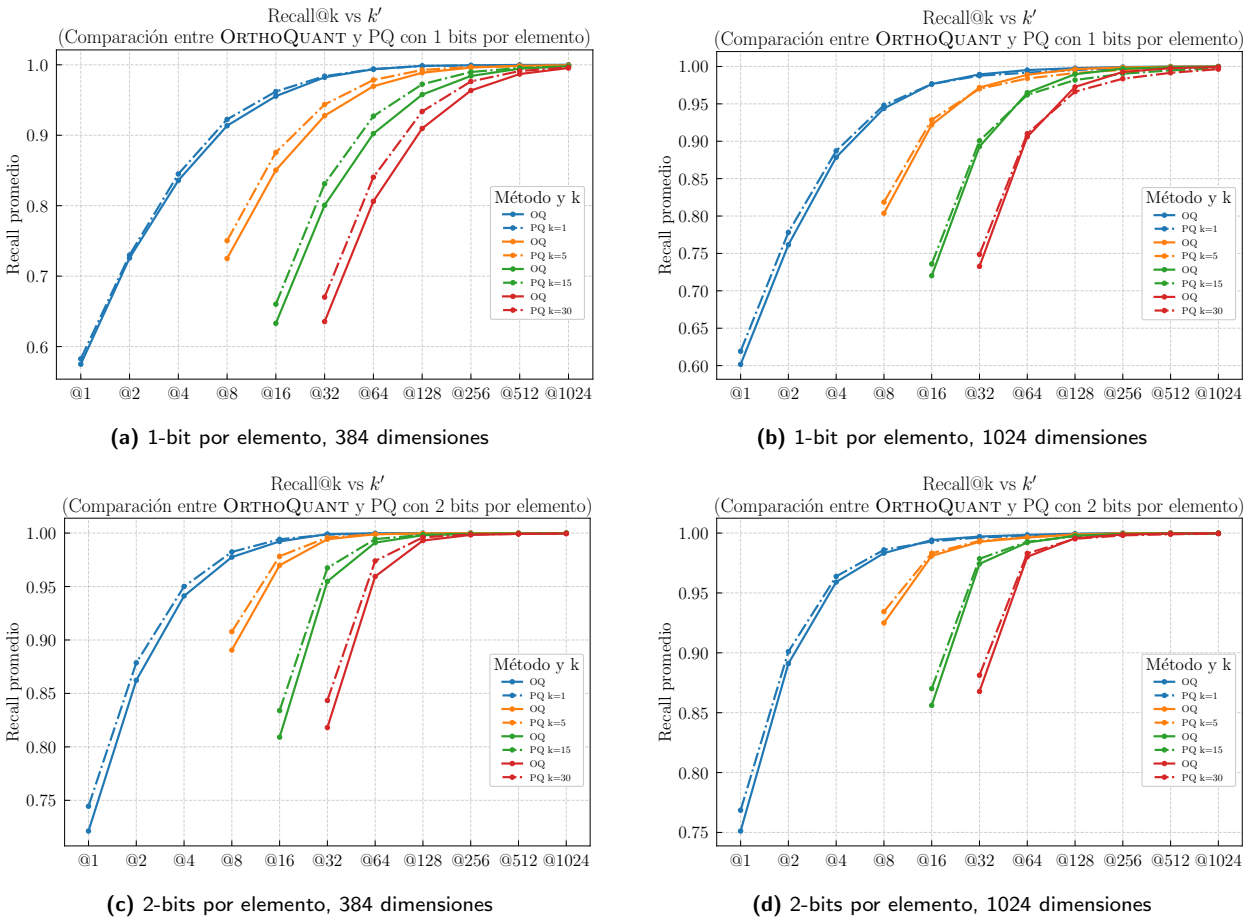
A partir del análisis del error de reconstrucción anterior, surge naturalmente la pregunta de si esta distorsión afecta directamente la capacidad de la base de datos comprimida para recuperar las consultas más relevantes. Para examinar esta cuestión, se calculó el *recall* promedio obtenido por cada método de compresión.

Para ello, es necesario determinar primero los vecinos más cercanos para una consulta dada. Esta operación se realiza mediante el cómputo de la similitud coseno entre el vector de consulta q y cada vector v_i en la base de datos, definida como:

$$\text{Similitud coseno} = \frac{q \cdot v_i}{||q|| \cdot ||v_i||} \tag{30}$$

Para asegurar la reproducibilidad y estandarización del cálculo, se utilizó la función `cosine_similarity` de la librería *scikit-learn*, ampliamente empleada en tareas de análisis de datos. Una vez obtenidos los valores de similitud, se seleccionan los k' vectores más similares a q, es decir, aquellos que presentan los valores más altos de similitud coseno en toda la base de datos comprimida. A partir de estos k' vectores, se calcula el *recall* como la proporción de vecinos relevantes que fueron correctamente recuperados. Un vecino se considera relevante si pertenece al conjunto de los k vecinos más cercanos verdaderos (calculados a partir de la base de datos sin comprimir). Finalmente, este *recall* se promedia sobre las

consultas utilizadas en los experimentos, obteniendo así el *recall* promedio reportado en los resultados.



(a) 1-bit por elemento, 384 dimensiones

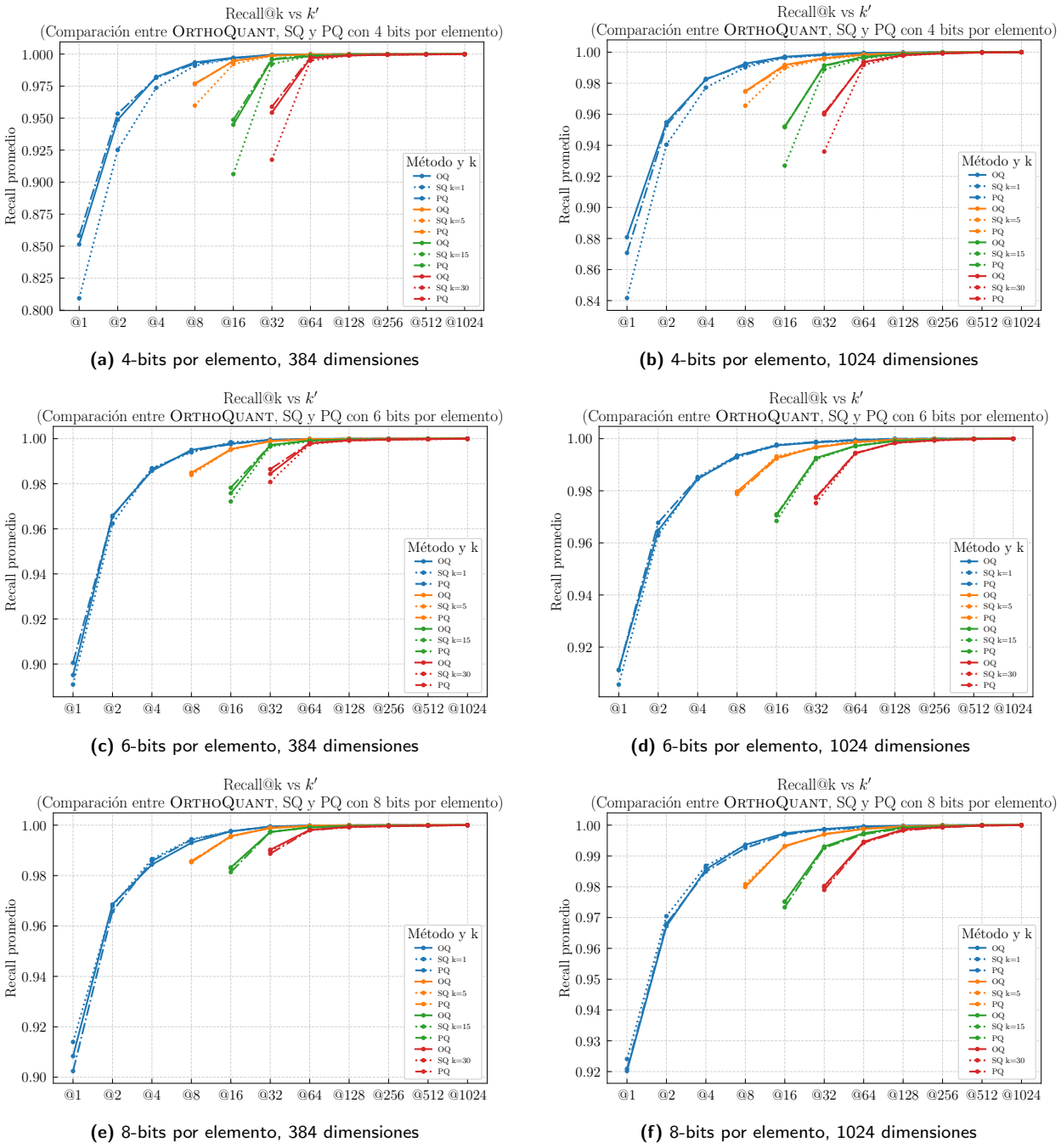
(b) 1-bit por elemento, 1024 dimensiones

(c) 2-bits por elemento, 384 dimensiones

(d) 2-bits por elemento, 1024 dimensiones

Figura 12. Comparación entre OrthoQuant y PQ en términos del *Recall@k@k′* promedio sobre todas las consultas. Cada fila corresponde a una resolución diferente en bits (1 y 2 bits), mientras que las dos columnas representan vectores de dimensión 384 y 1024, respectivamente.

La Figura 12 muestra la comparación del *Recall@k* entre OrthoQuant y PQ, mientras que la Figura 13 incluye también a SQ en la comparación. Ambas figuras corresponden al conjunto de datos `CCNEWS`, donde las gráficas de la izquierda representan los resultados para vectores de dimensión 384 y las de la derecha para dimensión 1024. En dichas gráficas, cada color indica un valor distinto de k (número de vecinos verdaderos que se busca recuperar): azul para $k = 1$, naranja para $k = 5$, y así sucesivamente. El eje horizontal representa el valor de k', es decir, la cantidad de vecinos aproximados extraídos de la base de datos comprimida. Este análisis permite evaluar cuántos vecinos aproximados deben recuperarse para garantizar la inclusión de los verdaderos k vecinos más cercanos. Aunque los métodos comparten color para un mismo valor de k, se distinguen por el estilo de línea: OrthoQuant se representa con línea sólida (-), SQ con línea punteada (.) y PQ con línea guion-punto (.-). Adicionalmente, la Figura 14 presenta los resultados equivalentes para el conjunto `GIST`, compuesto por vectores de dimensión 960.



(a) 4-bits por elemento, 384 dimensiones

(b) 4-bits por elemento, 1024 dimensiones

(c) 6-bits por elemento, 384 dimensiones

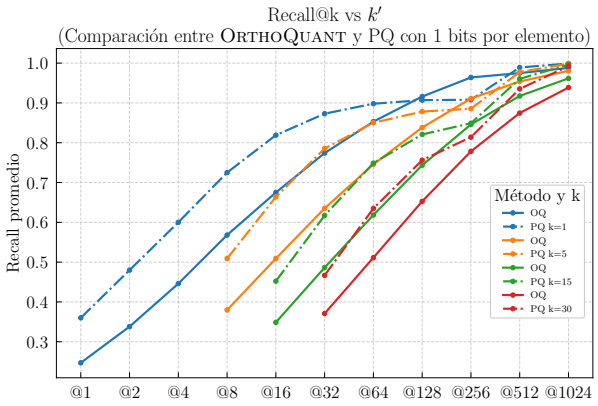
(d) 6-bits por elemento, 1024 dimensiones

(e) 8-bits por elemento, 384 dimensiones

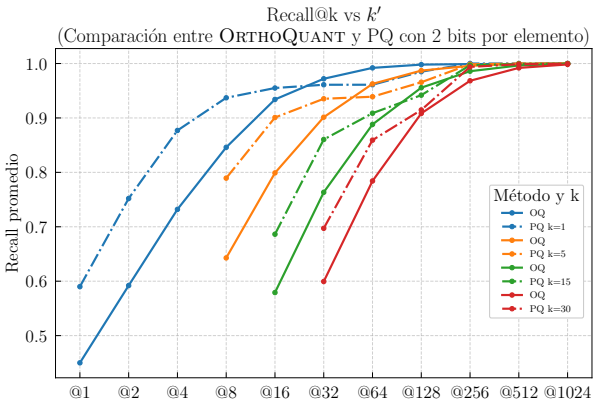
(f) 8-bits por elemento, 1024 dimensiones

Figura 13. Comparación entre OrthoQuant, SQ y PQ en términos del *Recall@k@k′* promedio sobre todas las consultas. Cada fila corresponde a una resolución diferente en bits (4, 6 y 8 bits), mientras que las dos columnas representan vectores de dimensión 384 y 1024, respectivamente.

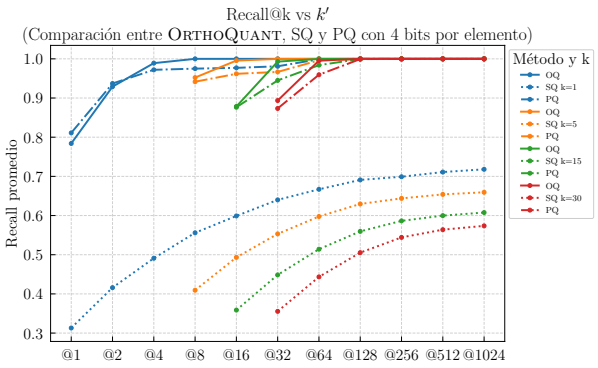
Una primera observación general para el conjunto `CCNEWS` es que el *recall* mejora al aumentar la dimensión de los vectores. Esta tendencia es consistente con el hecho de que, al contar con más componentes, es posible preservar una mayor cantidad de información durante la cuantización, lo cual reduce la pérdida asociada al proceso de compresión, independientemente del método utilizado.



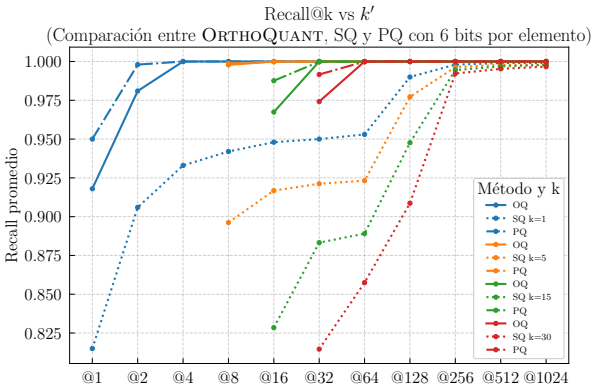
(a) 1-bit por elemento, 960 dimensiones



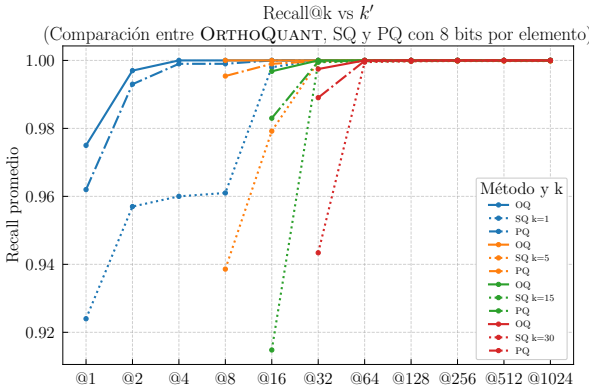
(b) 2-bits por elemento, 960 dimensiones



(c) 4-bits por elemento, 960 dimensiones



(d) 6-bits por elemento, 960 dimensiones



(e) 8-bits por elemento, 960 dimensiones

Figura 14. Comparación entre OrthoQuant, SQ y PQ en términos del *Recall@k@k′* promedio sobre todas las consultas para vectores de dimensión 960. Para los casos de 1 y 2 bits, únicamente se comparan OrthoQuant y PQ, mientras que para el resto de las resoluciones se incluyen los tres métodos.

Por otro lado, OrthoQuant muestra un desempeño muy similar al de los métodos de referencia en todas las resoluciones evaluadas. Incluso en resoluciones de 1 y 2 bits, donde, si bien no supera a PQ, su rendimiento es comparable. Este resultado es especialmente relevante si se considera que OrthoQuant no requiere conocimiento previo de la base de datos para realizar la cuantización. A diferencia de algunos

métodos como PQ, que dependen de procesos de agrupamiento como k-means y, por tanto, requieren acceso a la distribución específica de los vectores en la base de datos para construir sus diccionarios, OrthoQuant puede cuantizar cualquier vector de forma independiente. Esta propiedad le otorga una mayor generalidad y flexibilidad frente a bases de datos arbitrarias o dinámicas.

En contraste, para el conjunto `GIST`, el comportamiento de OrthoQuant se distingue notablemente de los demás métodos, particularmente en comparación con SQ. En las subfiguras 14c, 14d y 14e se observa que SQ presenta un rendimiento inferior e incluso irregular en las resoluciones de 6 y 8 bits. En cuanto a la comparación entre OrthoQuant y PQ, para resoluciones de 1 y 2 bits, PQ suele mostrar mejor desempeño cuando se utilizan valores bajos de k'; no obstante, OrthoQuant tiende a alcanzar valores de *recall* cercanos al 100 % con menor cantidad de candidatos, es decir, converge más rápidamente. Esta característica resulta especialmente favorable en aplicaciones donde se desea minimizar el número de comparaciones durante el *reranking*, reduciendo así el costo computacional en la fase de búsqueda final.

En resoluciones más altas, específicamente 4 y 8 bits, OrthoQuant llega incluso a superar a PQ en varias configuraciones. En contraste, en el caso de 6 bits, PQ exhibe un desempeño superior. Esta diferencia podría atribuirse a que, en dicha configuración, PQ emplea un mayor número de centroides por subvector ($2^{12} = 4096$, en comparación con los $2^6 = 64$ utilizados por OrthoQuant), lo que le permite una reconstrucción más precisa y, por ende, un *recall* más alto.

En resumen, los resultados obtenidos indican que OrthoQuant constituye una alternativa sólida y competitiva frente a los métodos clásicos de cuantización. Su independencia con respecto a los datos específicos de la base, su capacidad para alcanzar altas tasas de *recall* con un menor número de comparaciones en la etapa de *reranking*, así como su buen desempeño incluso en escenarios de compresión extrema, lo posicionan como una herramienta eficaz para sistemas de recuperación aproximada, particularmente en contextos donde se trabaja con bases de datos no gaussianas.

4.1.6. Tiempo de compresión

Posteriormente, se evaluó el tiempo requerido para comprimir la base de datos, cuyos resultados se presentan en la Tabla 9. Esta tabla muestra los tiempos de compresión, medidos en segundos, para un subconjunto de 100,000 vectores de la base de datos, considerando distintas configuraciones de *wbits* y tres niveles de dimensionalidad (384, 960 y 1024). Los resultados permiten observar que el método más

rápido en todos los casos fue SQ, lo cual era de esperarse, ya que este procedimiento consiste únicamente en aplicar una cuantización directa sobre los valores de los vectores. A continuación, OrthoQuant se posiciona como el segundo método más eficiente en términos de tiempo, debido a que, además de realizar una cuantización similar a SQ, incorpora el paso adicional de aplicar una rotación ortonormal y calcular la media y desviación estándar de los vectores. Por otro lado, PQ es el método que presenta los mayores tiempos de compresión, debido a que requiere ejecutar el algoritmo de k-medias de forma independiente sobre cada conjunto de subvectores. Además, el número de centroides (y, por lo tanto, la carga computacional) depende directamente del número de bits utilizados por código.

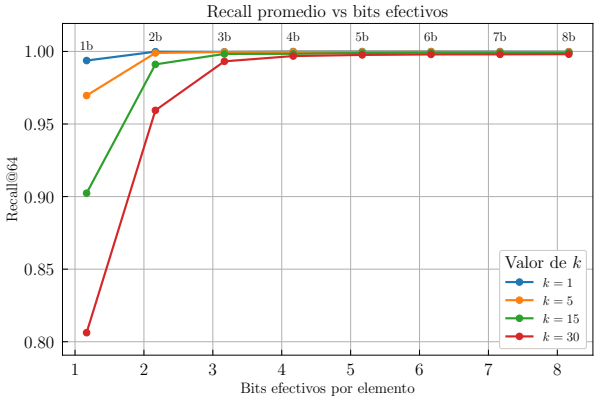
Tabla 9. Tiempos de compresión (en segundos) para 100,000 vectores de dimensión 384, 960 y 1024, utilizando OrthoQuant, PQ y SQ con distintas configuraciones de *wbits*, evaluados en una GPU Nvidia A4000.

Método	Dimensión	4 wbits	6 wbits	8 wbits
OrthoQuant	384	0.2038	0.3758	0.8423
SQ	384	0.0826	0.0986	0.1046
PQ	384	25.2022	71.9935	53.8717
OrthoQuant	960	0.4640	0.8845	2.1256
SQ	960	0.1989	0.2148	0.1966
PQ	960	94.4905	191.9404	138.2600
OrthoQuant	1024	0.4902	0.8939	2.3852
SQ	1024	0.2592	0.2475	0.2017
PQ	1024	79.7788	219.2533	167.3529

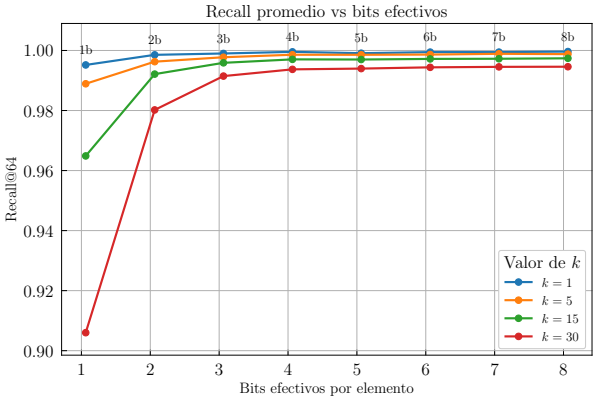
Estos resultados sugieren que, aunque OrthoQuant no alcanza la velocidad de SQ, ofrece una alternativa computacionalmente mucho más eficiente que PQ. Esta diferencia es particularmente relevante si se considera el desempeño obtenido en tareas de recuperación: como se muestra en la Figura 14, para vectores de dimensión 960 y configuraciones de 4 y 8 *wbits*, OrthoQuant supera a PQ en términos de *recall*, mientras que requiere un tiempo de compresión significativamente menor.

4.1.7. Impacto del número efectivo de bits por elemento en el recall

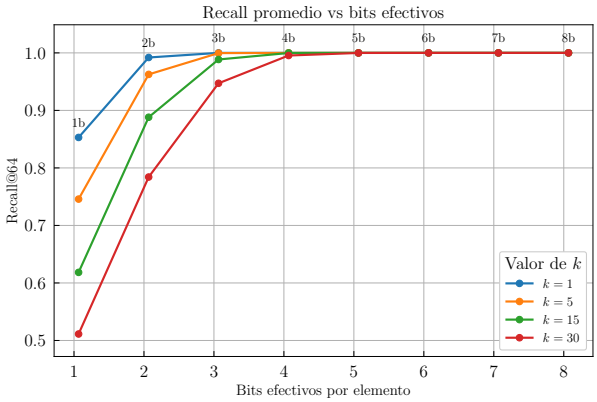
La Figura 15 muestra la relación entre el *recall*@k obtenido con OrthoQuant y el número efectivo de bits por elemento utilizados para la compresión de los vectores. En este experimento, se mantuvo fijo el número de vecinos recuperados en la base comprimida ($k' = 64$), y se evaluaron distintos valores de $k \in \{1, 5, 15, 30\}$, es decir, el número de vecinos más cercanos que se espera recuperar correctamente.



(a) Vectores de dimensión 384.
Sobrecarga de ∼0.17 bits/elemento debido al almacenamiento de los códigos, medias y desviación estándar de cada vector.



(b) Vectores de dimensión 1024.
Sobrecarga de ∼0.06 bits/elemento debido al almacenamiento de los códigos, medias y desviación estándar de cada vector.



(c) Vectores de dimensión 960.
Sobrecarga de ∼0.07 bits/elemento debido al almacenamiento de los códigos, medias y desviación estándar de cada vector.

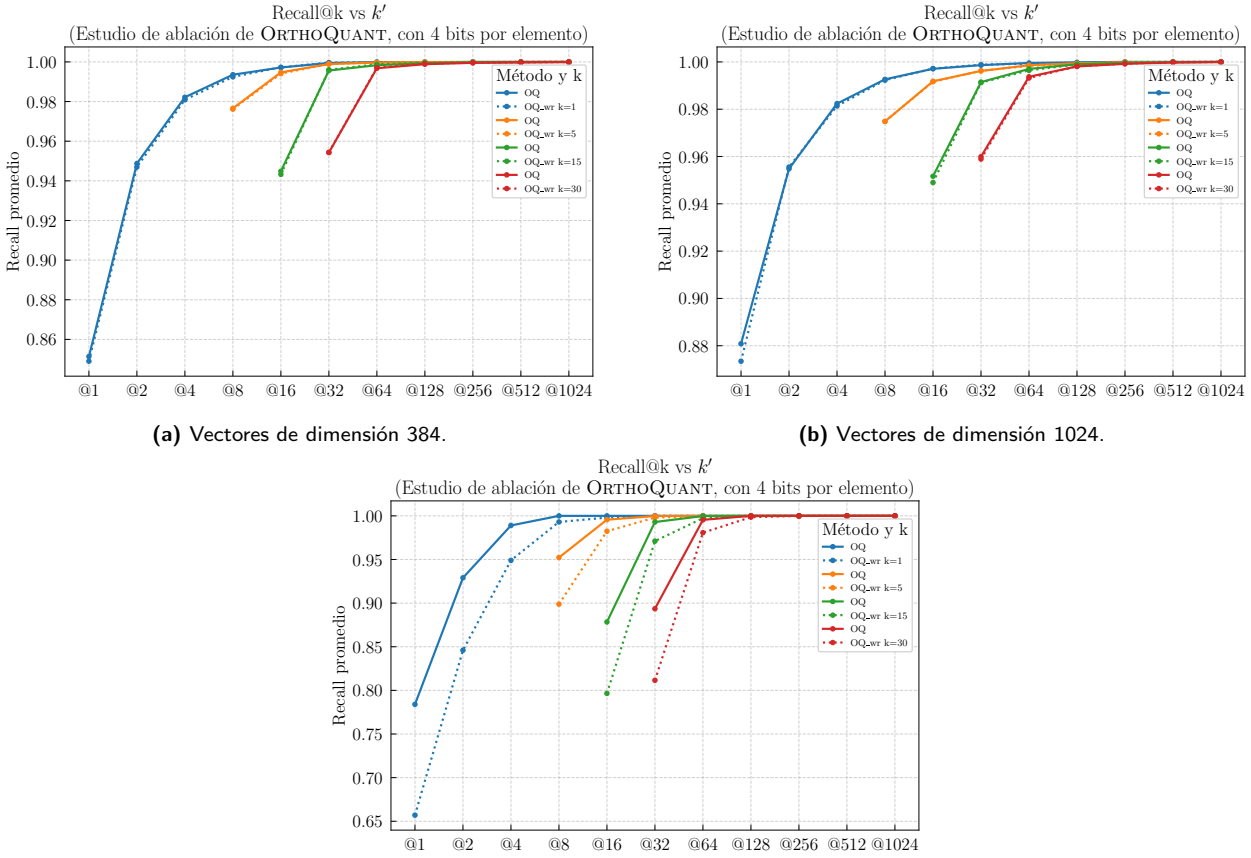
Figura 15. Promedio de *recall*@k' (con $k' = 64$) para distintos valores de k, en función del número efectivo de bits por elemento en memoria. Cada punto está anotado con el número de bits utilizados únicamente por los códigos.

En las tres subgráficas (dimensiones 384 (15a), 1024 (15b) y 960 (15c)) se observa una tendencia clara y consistente: a mayor número de bits por elemento, mayor es el valor de *recall*@k alcanzado, lo cual indica que incrementar la cantidad de información preservada durante la compresión mejora directamente la calidad de los resultados en tareas de recuperación. Esta tendencia es robusta frente al valor de k, lo cual sugiere que la mejora no depende del número de vecinos buscados, sino que es una consecuencia directa de la fidelidad con la que se conserva la estructura del espacio vectorial.

Ambas gráficas incluyen, además, la información relativa a la sobrecarga adicional introducida por el almacenamiento explícito de la media y desviación estándar de cada vector, almacenados como números de punto flotante de 32 bits. La sobrecarga es aproximadamente 0.17, 0.06 y 0.07 bits por elemento para vectores de dimensión 384, 1024 y 960, respectivamente. Dicho costo adicional se consideró al momento de calcular los valores efectivos de *wbits* representados en el eje horizontal de las figuras.

4.1.8. Estudio de ablación: desempeño con y sin rotación

Por otra parte, se llevó a cabo un estudio de ablación para determinar el impacto de aplicar la matriz de rotación ortonormal a los vectores previo a su codificación. La Figura 16 muestra el *recall* promedio para distintos valores de k ($\{1, 5, 15, 30\}$) en una compresión con resolución de 4 bits por código. Con línea sólida se presentan los resultados de OrthoQuant, mientras que con línea punteada se muestran los resultados sin rotación.



(a) Vectores de dimensión 384.

(b) Vectores de dimensión 1024.

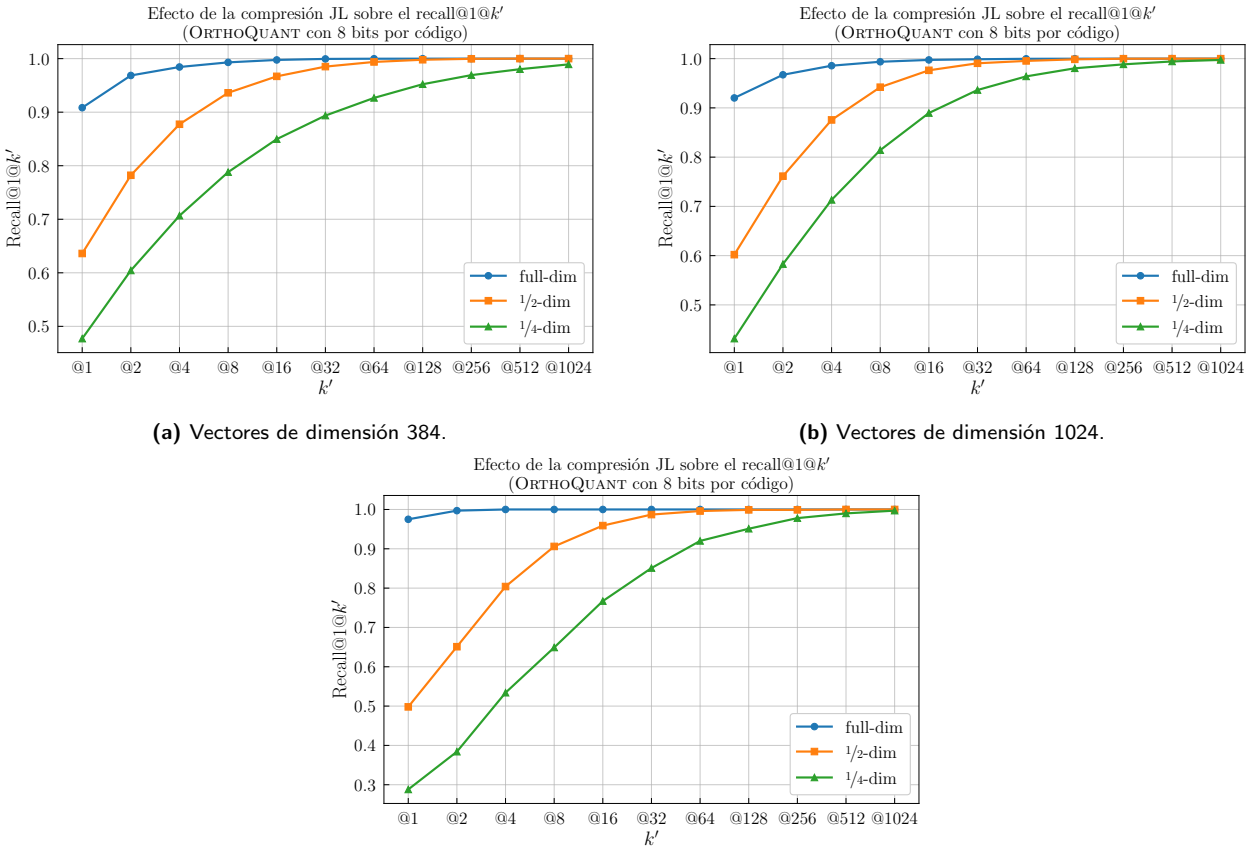
(c) Vectores de dimensión 960.

Figura 16. Estudio de ablación del recall promedio para el caso de OrthoQuant con y sin rotación.

En esta figura se observa que, para los vectores de *embeddings*, cuya distribución es naturalmente gaussiana, la matriz de rotación ortonormal, si bien parece superar ligeramente a la versión sin rotación, no produce una diferencia significativa. Sin embargo, para los vectores de dimensión 960, que corresponden a la base de datos `GIST` y no presentan una distribución naturalmente gaussiana, la rotación sí muestra un impacto claro en la calidad de la búsqueda, por lo que el método aporta un mayor beneficio en bases de datos con distribuciones no gaussianas.

4.1.9. Estudio de ablación mediante proyecciones Johnson-Lindenstrauss: aceleración del producto interno y compresión más agresiva

Por último, se realizó un estudio de ablación, descrito previamente en la Sección 3.2.4, basado en el lema de Johnson-Lindenstrauss, con el objetivo de explorar una variante de OrthoQuant que permita acelerar el cálculo del producto interno o, alternativamente, lograr una compresión más agresiva. Dado que el método propuesto ya implica la aplicación de una transformación de rotación ortonormal sobre los vectores, seleccionar aleatoriamente un subconjunto de sus coordenadas garantiza, de acuerdo con dicho lema, que las distancias entre los vectores muestreados se preservan con un error acotado y probabilísticamente controlado.



(a) Vectores de dimensión 384.

(b) Vectores de dimensión 1024.

(c) Vectores de dimensión 960.

Figura 17. Efecto de seleccionar coordenadas aleatorias como método de reducción de dimensionalidad basado en la proyección de Johnson-Lindenstrauss. Se muestran los resultados para la compresión con 7 bits por elemento en la matriz de índices, considerando proyecciones a la mitad y a una cuarta parte de las coordenadas originales.

Para evaluar esta variante, se experimentó con el muestreo aleatorio de la mitad y de una cuarta parte de las coordenadas de los vectores rotados. Los resultados se presentan en la Figura 17. Como era esperable,

al reducir el número de coordenadas seleccionadas, el *recall* disminuye; sin embargo, esta pérdida en la calidad de recuperación es un efecto natural del intercambio entre compresión y precisión.

A pesar de dicha reducción, se observó que es posible alcanzar un *recall* cercano al 100 % incrementando el número de vecinos k' extraídos de la base de datos comprimida. En particular, al utilizar una cuarta parte de la dimensión original para vectores de dimensión 384, se obtiene un *recall* casi perfecto con $k' = 1024$, mientras que para vectores de dimensión 960 y 1024 se logra un comportamiento similar con $k' = 512$. Esto sugiere que, aunque se pierde información durante la compresión, dicha pérdida puede compensarse mediante un *re-ranking* sobre un conjunto mayor de candidatos. Además, al reducir la dimensionalidad, el cálculo del producto interno se acelera debido a la menor cantidad de operaciones involucradas.

En resumen, esta variante muestra que OrthoQuant permite obtener compresiones más agresivas con un desempeño competitivo en términos de *recall* y una búsqueda más rápida, siempre que se acepte un aumento en la cantidad de candidatos para el *reranking*. Este resultado refuerza la flexibilidad del método propuesto, al habilitar configuraciones adaptadas a diferentes restricciones de almacenamiento y tiempo de inferencia.

4.1.10. Discusión sobre la compresión de vectores

Los resultados obtenidos permiten realizar un análisis detallado del comportamiento de OrthoQuant en el contexto de compresión de bases de datos vectoriales, considerando tanto métricas de reconstrucción como desempeño en tareas de recuperación por similitud.

En primer lugar, se observó que el método propuesto, si bien presenta un mayor error de reconstrucción promedio en las resoluciones más bajas en comparación con los métodos de referencia (SQ y PQ), logra mantener dicha métrica dentro de un rango competitivo a medida que se incrementa la resolución. Para 8 *wbits*, OrthoQuant alcanza incluso el menor error promedio en los tres conjuntos de datos evaluados. Además, exhibe una desviación estándar igual o inferior a la de los métodos comparados en la mayoría de los casos, lo que sugiere un comportamiento más estable en términos estadísticos. Un aspecto particularmente relevante es su bajo error máximo en resoluciones extremas, donde evita los *outliers* severos que sí se presentan en los métodos de referencia. Cabe destacar que los niveles de error obtenidos por PQ fueron notablemente bajos en ciertos escenarios, lo cual se explica por el uso de

configuraciones específicas con subvectores muy cortos y un elevado número de centroides, estrategia que no suele adoptarse en entornos reales debido a su alto costo computacional y limitada escalabilidad, pero que se empleó en este estudio con el objetivo de realizar una comparación justa con OrthoQuant bajo condiciones equivalentes de compresión.

Desde el punto de vista de recuperación, las gráficas de *recall@*k muestran que OrthoQuant alcanza un desempeño competitivo en bases de datos de embeddings de tipo gaussiano, como las generadas por modelos del tipo *Transformer*. En dichos casos, el método mantiene niveles de *recall* comparables a los métodos de referencia, particularmente en configuraciones de compresión intermedia y alta (4, 6 y 8 *wbits*). No obstante, donde OrthoQuant muestra una ventaja más pronunciada es en escenarios con bases de datos no gaussianas, como aquellas representadas por descriptores tradicionales tipo `GIST`. En este contexto, la transformación ortonormal previa a la cuantización tiene un impacto positivo más claro, permitiendo preservar mejor las relaciones de similitud entre vectores, y logrando así un mayor *recall* que los métodos comparados. Esta diferencia sugiere que la propuesta es especialmente útil cuando la distribución de los datos se aleja de una forma gaussiana o no presenta una estructura estadística favorable para técnicas como Product Quantization, cuyo desempeño depende en gran medida de la independencia y homogeneidad estadística en los subespacios definidos.

Otro aspecto relevante es el tiempo de compresión. Aunque OrthoQuant requiere un mayor tiempo que SQ debido al preprocesamiento con rotación ortonormal y al cálculo de la media y desviación estándar para cada vector, se mantiene considerablemente más eficiente que PQ, incluso en configuraciones de alta compresión. Esta característica resulta especialmente importante en aplicaciones que manejan grandes volúmenes de datos, ya que permite alcanzar un equilibrio adecuado entre eficiencia en almacenamiento y costos computacionales.

Finalmente, los estudios complementarios demostraron que la rotación aleatoria utilizada por OrthoQuant es especialmente útil cuando los vectores no siguen una distribución gaussiana, y que, además, gracias a esta proyección, puede adaptarse fácilmente para convertirse en una proyección aleatoria del tipo Johnson-Lindenstrauss mediante un submuestreo de dimensiones. Esta adaptación permite acelerar el cálculo del producto interno, y conserva un *recall* aceptable cuando se complementa con estrategias de *re-ranking* aplicadas sobre conjuntos más amplios de vecinos aproximados.

En conjunto, los resultados sugieren que OrthoQuant constituye una alternativa viable y competitiva para la compresión de bases de datos de vectores, especialmente en escenarios de alta dimensionalidad, donde la mayor cantidad de coordenadas brinda más información para el cálculo de distancias, así como

en vectores con distribuciones no naturalmente gaussianas. Esto lo convierte en una opción atractiva para aplicaciones que requieren un equilibrio entre calidad de recuperación, eficiencia en el almacenamiento y velocidad de compresión.

4.2. Evaluación de la compresión en conjuntos de matrices (pesos de modelos de lenguaje)

Una vez probada la factibilidad de OrthoQuant como método de compresión de vectores en un escenario aplicado a bases de datos vectoriales para resolver tareas de búsqueda por similitud, se procedió a evaluar su desempeño en el contexto de compresión de matrices.

4.2.1. Modelo utilizado

Para evaluar esta hipótesis, se propuso probar su desempeño en una aplicación real: la compresión de las matrices correspondientes a las capas lineales de un modelo de lenguaje de gran tamaño, en particular, el modelo LLaMA 3.2 3B desarrollado por Meta. Con el fin de juzgar adecuadamente la eficacia de OrthoQuant, fue necesario compararlo con otros métodos de compresión de modelos existentes en la literatura, de modo que fuera posible identificar su posicionamiento relativo frente a técnicas consolidadas.

4.2.2. Métodos utilizados

Del capítulo anterior se desprende que OrthoQuant es un método *data-free*, es decir, no requiere acceso a los datos originales de entrenamiento para llevar a cabo la compresión, ni depende de datos sintéticos para realizar un ajuste posterior de los pesos o un proceso de *fine-tuning*. Esta característica motivó la elección de métodos de referencia que compartieran esta propiedad, con el propósito de realizar una comparación justa y equitativa. Los métodos seleccionados para esta evaluación fueron la Aproximación de Rango Bajo (LRA, por sus siglas en inglés), la Transformada Discreta del Coseno (DCT) y el Redondeo al Más Cercano (RTN). La implementación utilizada para cada uno de estos enfoques

se detalla en los anexos correspondientes: en el Anexo B para LRA, en el Anexo C para DCT y en el Anexo D para RTN.

4.2.3. Error de reconstrucción

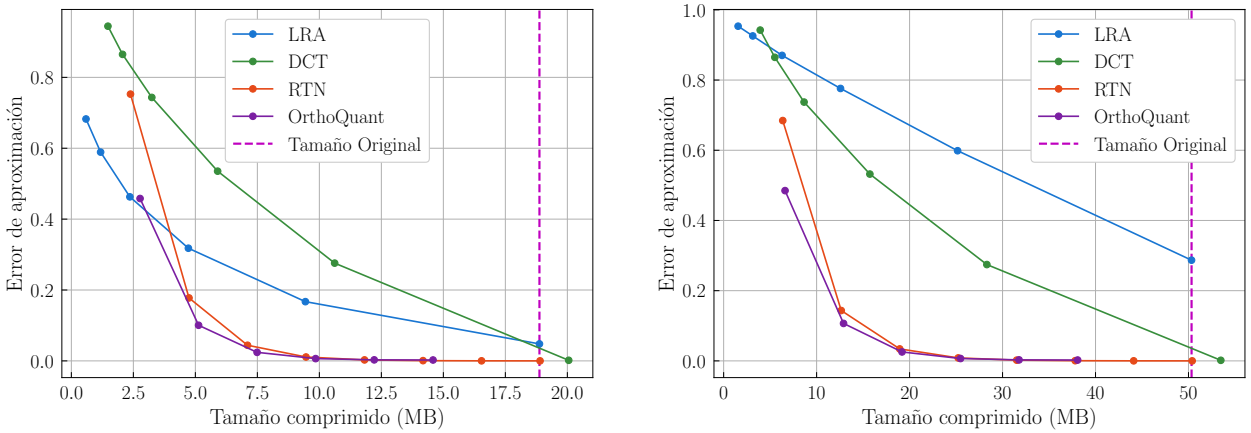
Previo a la implementación completa de la compresión sobre el modelo, se realizaron pruebas preliminares sobre un subconjunto de matrices seleccionadas del LLM con el objetivo de analizar cómo la compresión mediante OrthoQuant y su posterior reconstrucción afectan la calidad de las matrices resultantes, así como evaluar de qué manera se conserva o degrada dicha calidad conforme se incrementa el grado de compresión. Para este fin, se seleccionaron dos matrices: la matriz de *queries* del mecanismo de atención y la matriz de proyección *up* del módulo MLP, ambas pertenecientes a la primera capa del modelo. La primera prueba consistió en comprimir estas matrices utilizando tanto los métodos de referencia como el método propuesto, explorando diferentes niveles de compresión

El error de reconstrucción se calculó mediante la norma de Frobenius entre la matriz original A y la matriz reconstruida $\hat{A}$, de acuerdo con la expresión:

$$\frac{\|A - \hat{A}\|_F}{\|A\|_F} = \frac{\sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |(a_{ij} - \hat{a}_{ij})|^2}}{\sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}} \tag{31}$$

Esta métrica permite cuantificar el error acumulado a lo largo de todos los elementos de la matriz; un valor de 1 indica que la matriz reconstruida es completamente diferente a la original, mientras que un valor de 0 refleja una coincidencia exacta.

En la Figura 18 se presentan los errores de aproximación obtenidos para las matrices comprimidas con los distintos métodos evaluados. A simple vista, es posible observar que los métodos en general presentan un mejor desempeño para la matriz correspondiente al mecanismo de atención que para la del MLP. Este comportamiento puede explicarse por la diferencia en la naturaleza de estas matrices: mientras que la matriz del mecanismo de atención no posee rango completo, la matriz del MLP sí lo tiene. Esta diferencia sugiere que la matriz de *queries* presenta cierta redundancia en sus datos que la pueden aprovechar los métodos de compresión.



(a) Matriz de *queries*.

(b) Matriz de proyección *up*.

Figura 18. Error de aproximación para diferentes niveles de compresión utilizando los métodos LRA, DCT, RTN y OrthoQuant. Las dos figuras corresponden a matrices de pesos extraídas de la primera capa del modelo LLaMA 3.2 3B.

Cabe resaltar que el método que matemáticamente explota de manera más directa esta propiedad de rango bajo es LRA, dado que precisamente su fundamento radica en la capacidad de representar de manera eficiente matrices que no son de rango completo. De manera análoga, la DCT tiende a funcionar mejor cuando la información de la matriz está concentrada en unos pocos coeficientes de baja frecuencia, lo cual también implica la existencia de cierta redundancia o estructura aprovechable. En este sentido, la matriz de *queries* resulta ser más separable (en términos de contenido informativo) que la matriz *up*.

Sin embargo, resulta llamativo que estos dos métodos, a pesar de sus fundamentos matemáticos sólidos, se desempeñen peor que RTN y OrthoQuant en ambos casos, especialmente en el de la matriz *up*, que al ser de rango completo no ofrece redundancias fácilmente explotables para LRA o DCT. Por otro lado, tanto RTN como OrthoQuant muestran un desempeño bastante similar en ambos tipos de matrices, con una ligera ventaja a favor de OrthoQuant en todos los niveles de compresión evaluados.

Un aspecto importante que se desprende de estos resultados es la independencia de OrthoQuant respecto al rango de las matrices. En particular, para el modelo LLaMA, las matrices concatenadas W_{queries}, W_{keys} y W_{values} suelen ser de rango reducido debido a que resultan de la concatenación de las cabezas de atención. Los experimentos muestran que el desempeño de OrthoQuant se mantiene estable tanto para matrices de rango reducido como para matrices de rango completo, a diferencia de métodos como LRA o DCT, cuyo rendimiento disminuye notablemente en el caso de matrices de rango completo. Esto indica que el método propuesto no depende de la estructura de rango de las matrices, reforzando la versatilidad de OrthoQuant frente a distintos tipos de datos.

Así mismo, con el propósito de obtener un ejemplo visual que ilustre el efecto de OrthoQuant sobre la calidad de reconstrucción, se procedió a comprimir una imagen en blanco y negro de un camarógrafo. Este tipo de imagen puede representarse como una matriz bidimensional cuyos valores indican la intensidad de cada píxel, es decir, su nivel de gris. Este experimento se presenta en la Figura 19, cuyo objetivo es permitir la visualización cualitativa tanto del error de reconstrucción como de su causa principal, la cual radica en la cuantización aplicada a los valores centrales de la distribución de la matriz rotada.

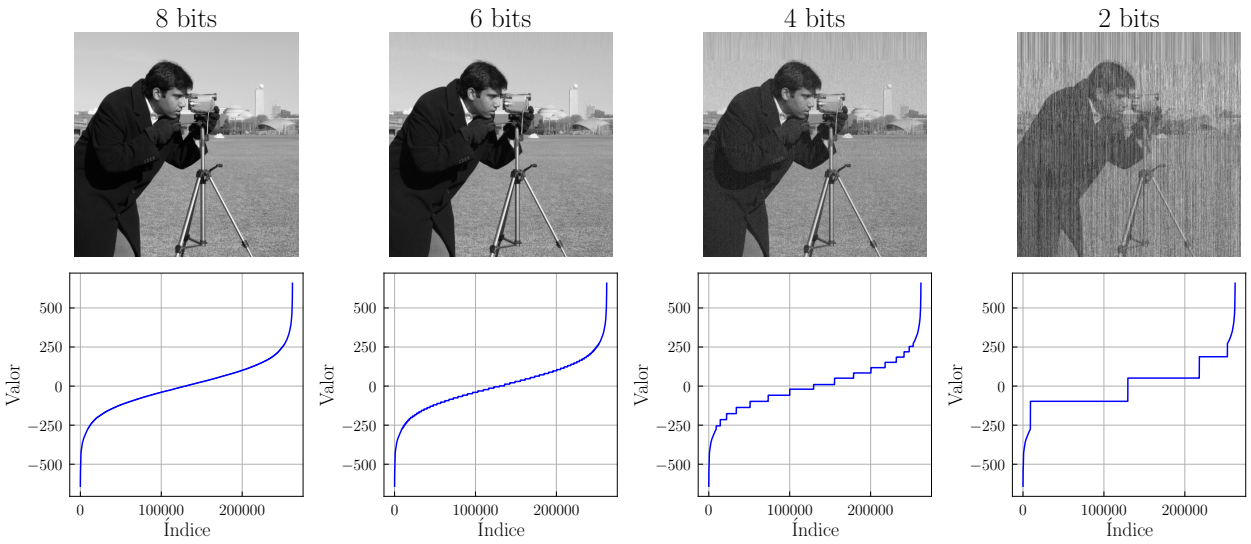


Figura 19. Imágenes reconstruidas $\hat{A}$ (fila superior) y distribución de los valores ordenados de la matriz comprimida $\tilde{D}$ (fila inferior) para diferentes configuraciones de compresión: 8, 6, 4 y 2 bits por elemento en la matriz de índices. Las gráficas de $\tilde{D}$ reflejan el efecto del segmentado aplicado durante la compresión.

En dicha figura se observa cómo la reducción en la cantidad de bits asignados por elemento para representar los índices se traduce en una menor cantidad de segmentos disponibles en la matriz rotada $\tilde{D}$, de acuerdo con la relación $2^b - 1$ segmentos para cada b bits utilizados. Así, cuando se emplean 8 y 6 bits, la calidad de la reconstrucción es elevada, dado que la gran cantidad de segmentos permite aproximar con alta fidelidad la distribución de la matriz comprimida $\tilde{D}$ con respecto a la matriz original D.

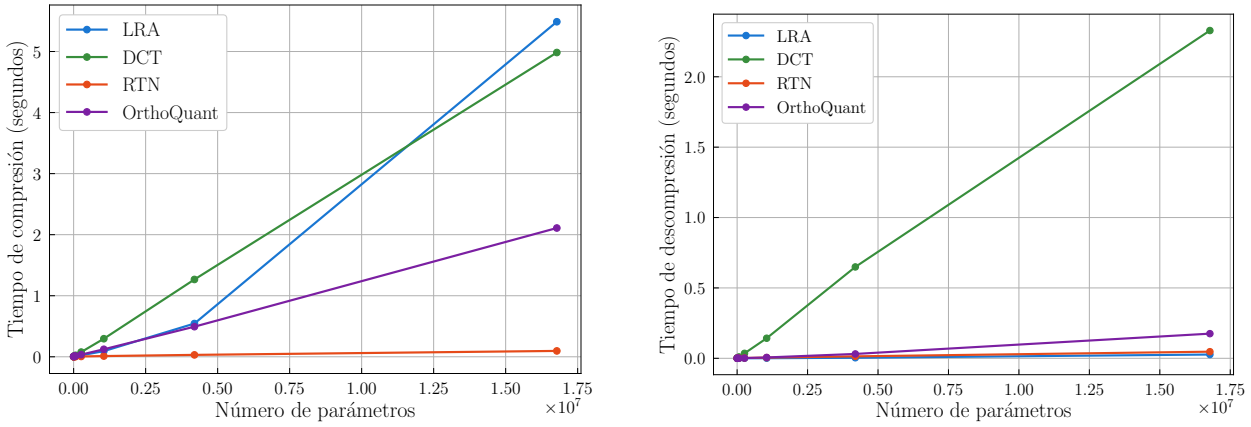
Sin embargo, al reducir el número de bits a 4 y 2, es evidente cómo la compresión comienza a degradar la distribución original de manera más notoria, provocando la aparición de ruido perceptible en las imágenes reconstruidas. A pesar de esta degradación, gracias al almacenamiento explícito de los valores fuera de rango, se observa que incluso en el caso extremo de 2 bits la imagen mantiene su estructura global y permite identificar claramente la figura del camarógrafo.

Este resultado confirma la importancia de conservar los valores fuera de rango durante el proceso de

compresión, pues dichos elementos contienen información esencial que no se pueden representar de manera adecuada mediante la segmentación central. Su preservación resulta, por tanto, clave para garantizar una reconstrucción de calidad y evitar la pérdida de detalles significativos presentes en la matriz original.

4.2.4. Tiempo de compresión y descompresión

Adicionalmente, otra forma de evaluar la viabilidad del método propuesto para la compresión de modelos de lenguaje consiste en medir los tiempos requeridos tanto para la compresión como para la descompresión de las matrices.



(a) Tiempos de compresión.

(b) Tiempos de descompresión.

Figura 20. Tiempos de compresión y descompresión (en segundos) de las diferentes técnicas de compresión (LRA, DCT, RTN y OrthoQuant) evaluadas sobre matrices de distintas dimensiones.

En la Figura 20 se presentan los tiempos de compresión y descompresión obtenidos para OrthoQuant en comparación con los métodos de referencia LRA, DCT y RTN. Para esta evaluación se consideraron matrices aleatorias de diferentes dimensiones con el objetivo de analizar cómo varía el tiempo de procesamiento conforme aumentan las dimensiones o el número de parámetros de la matriz.

Respecto a los tiempos de compresión, se observa que a medida que la dimensión de las matrices crece, los métodos LRA y DCT requieren un mayor tiempo de procesamiento, siendo estos los más costosos computacionalmente en esta etapa. A continuación se encuentra OrthoQuant, cuyo tiempo de compresión resulta intermedio entre LRA/DCT y RTN. Por su parte, RTN presenta los tiempos de compresión más bajos, con valores cercanos a cero segundos incluso para matrices de mayor tamaño;

además, su tiempo de ejecución muestra un crecimiento lineal con una pendiente reducida.

En cuanto a los tiempos de descompresión, DCT es el método que presenta el mayor tiempo de ejecución en comparación con el resto de técnicas evaluadas. A este le sigue OrthoQuant, que muestra un desempeño ligeramente más lento que RTN y LRA, los cuales resultaron ser los métodos más rápidos en esta etapa.

Es importante destacar que, en el contexto específico de la compresión de matrices para modelos de lenguaje de gran tamaño, como los LLMs, resulta preferible minimizar el tiempo de descompresión, dado que esta operación se realiza de manera recurrente durante la inferencia del modelo. En contraste, la compresión es una tarea que típicamente se ejecuta una sola vez durante la preparación del modelo. Por lo tanto, el tiempo de descompresión observado para OrthoQuant representa un indicador favorable de su viabilidad práctica para su aplicación en la compresión de matrices de redes neuronales, ya que se mantiene competitivo frente a los métodos de referencia evaluados.

4.2.5. Evaluación cualitativa del modelo

Con base en el análisis previo, se evaluó el método de compresión propuesto sobre el modelo LLaMA 3.2 3B, centrándose en las matrices de pesos correspondientes a las capas lineales de los bloques de atención y del perceptrón multicapa.

La fase inicial de evaluación consistió en analizar cualitativamente las respuestas generadas por los modelos comprimidos mediante cada una de las técnicas consideradas, verificando la corrección y coherencia de las respuestas frente a un conjunto de tres consultas (*queries*) de prueba. Adicionalmente, se registró el nivel de compresión alcanzado por cada método.

Para esta evaluación cualitativa preliminar se seleccionaron tres consultas diseñadas para verificar si los modelos comprimidos conservaban el conocimiento del modelo original, en particular en tareas que implican enumeraciones o listas de elementos. Las consultas utilizadas fueron las siguientes:

1. *In physics, the primary colors of light are.* Se esperaba que el modelo respondiera con la lista correcta de los colores primarios de la luz: rojo, azul y verde.
2. *The five human senses are.* Se esperaba que el modelo enumerara los cinco sentidos humanos: vista, oído, olfato, gusto y tacto.

3. *The capital city of France is.* Se esperaba que el modelo respondiera correctamente que la capital de Francia es París.

Cada técnica de compresión se evaluó en función de la corrección y coherencia de las respuestas generadas para estas tres consultas. Para ello, se utilizó una escala de evaluación de tres niveles: correcto (verde), coherente (amarillo) e incorrecto (rojo). Una respuesta se consideró correcta si listaba explícitamente los mismos elementos que el modelo original y en un orden equivalente o aceptable; coherente si bien no listaba los elementos esperados, generaba una salida con sentido gramatical y relacionada temáticamente con la consulta; e incorrecta si el modelo no generaba una salida interpretable o retornaba texto irrelevante o corrupto, lo cual se interpretó como un indicio de degradación extrema en el desempeño funcional del modelo comprimido.

Tabla 10. Comparación de respuestas y compresión de pesos para LLaMA 3.2 3B (FP16) con diferentes métodos.

	Original	LRA	DCT	RTN	OrthoQuant
Input: *In physics, the primary colors of light are...*	... **red, blue, and green**. These colors are the result of the interaction	... [salida corrupta: 199199201201 “ ”199 “201199202 “ FW201 FW]	... the same as the other primary color of the light of your primary primary of	... **red, green, and blue**. These colors are the result of the way	... **red, blue, and green**. These are the colors that are produced when
Input: *The five human senses are...*	... **sight, hearing, smell, taste, and touch**. The five senses allow us to interact with	... [salida ilegible: caracteres cirílicos y devanagari no soportados]	... the same as the five The 5 human beings are as the 3rd,	... **sight, smell, touch, hearing and taste**. The five senses of a person are the five	... **sight, hearing, smell, taste, and touch**. The sense of sight is the ability to
Input: *The capital city of France is...*	... **Paris**. It is the most visited city in the world. Paris is a city that is	... [salida ilegible: caracteres cirílicos y devanagari no soportados]	... a very different type of city. The city is the capital of the world’s largest city	... **Paris**. It is located in the north of the country. The city is divided into	... **Paris**. It is the capital of the region of Île-de-France. The city
Compresión de pesos	0 %	50.00 %	43.75 %	74.92 %	72.35 %
Tamaño del modelo	7.21 GB	4.39 GB	4.75 GB	2.99 GB	3.14 GB

En la Tabla 10 se presentan los resultados obtenidos en esta evaluación cualitativa. El modelo original respondió correctamente a las tres consultas planteadas, y dichas respuestas se tomaron como referencia o *ground truth* para la comparación con las variantes comprimidas.

Se observa que los métodos LRA y DCT presentan un deterioro significativo en la capacidad del modelo para generar respuestas coherentes, produciendo en algunos casos salidas ilegibles o con caracteres corruptos, lo que refleja una degradación severa en la representación de los pesos comprimidos. En particular, LRA arrojó respuestas con datos binarios corrompidos o no decodificables, mientras que DCT generó oraciones gramaticalmente válidas pero semánticamente inconsistentes o sin relación directa con las consultas planteadas.

Un aspecto interesante a resaltar es que, de acuerdo con lo mostrado en la Figura 18, el método LRA obtuvo mejores aproximaciones en términos de error de reconstrucción para la matriz asociada al mecanismo de atención en comparación con DCT. Sin embargo, en el caso de la matriz correspondiente al perceptrón multicapa, DCT mostró un desempeño superior en términos de aproximación. Este comportamiento sugiere que una mejor reconstrucción de las matrices del perceptrón multicapa podría tener un impacto más significativo en la preservación de la funcionalidad del modelo comprimido que una buena aproximación de las matrices del mecanismo de atención.

Esta observación es consistente con el hecho de que las capas MLP en modelos de lenguaje de gran escala suelen concentrar una parte sustancial de la capacidad de representación y transformación de información del modelo, por lo que una degradación en estas matrices podría afectar de manera directa la generación de respuestas coherentes, incluso si las matrices de atención se han reconstruido con mayor precisión.

Por el contrario, los métodos RTN y OrthoQuant lograron preservar en gran medida la calidad de las respuestas generadas, reproduciendo de forma correcta y coherente la información esperada en cada consulta. Es importante destacar que, además de mantener la exactitud en las respuestas, OrthoQuant replicó el mismo orden de los elementos listados por el modelo original, lo que sugiere una fidelidad mayor en la conservación de la estructura de salida del modelo. En cambio, aunque RTN también produjo respuestas correctas en términos de contenido, en algunos casos alteró el orden de los ítems enumerados (por ejemplo, en la lista de los cinco sentidos humanos), lo que puede interpretarse como un indicio de una degradación funcional ligeramente superior a la observada con OrthoQuant.

4.2.6. Compresión alcanzada

Además del desempeño cualitativo, otro aspecto relevante es el nivel de compresión alcanzado por cada técnica, expresado tanto en porcentaje de reducción de los pesos como en el tamaño final del modelo.

Como se muestra en la Tabla 10, RTN fue el método que logró la mayor reducción de tamaño, con un nivel de compresión del 74.92 % y un tamaño final de 2.99 GB. Por su parte, OrthoQuant alcanzó un nivel de compresión del 72.35 %, resultando en un tamaño final de 3.14 GB, una reducción muy cercana a la obtenida por RTN, pero con la ventaja adicional de preservar fielmente no sólo el contenido sino también el orden de los elementos en las respuestas generadas. Este equilibrio entre compresión eficiente y fidelidad funcional representa una de las principales fortalezas del método propuesto.

En contraste, LRA y DCT alcanzaron compresiones menores (50.00 % y 43.75 %, respectivamente), obteniendo tamaños finales de 4.39 GB y 4.75 GB. A pesar de su menor agresividad en la reducción de tamaño, estos métodos mostraron un deterioro funcional considerable, evidenciado en la generación de salidas incoherentes o corruptas, lo que pone en duda su viabilidad para la compresión de modelos de lenguaje a gran escala como LLaMA 3.2 3B.

En conjunto, estos resultados permiten concluir que OrthoQuant no sólo es competitivo en términos de reducción del tamaño del modelo, sino que también conserva de manera más precisa la capacidad funcional del modelo original para responder consultas que requieren conocimiento factual estructurado, superando incluso a RTN en la preservación del orden y forma de las respuestas, sin sacrificar de manera significativa el nivel de compresión alcanzado.

4.2.7. Evaluación cuantitativa en benchmarks establecidos

A partir de esta evaluación cualitativa, se decidió continuar con los métodos RTN y OrthoQuant en la fase de evaluación cuantitativa mediante *benchmarks* estandarizados, descartando LRA y DCT debido a su pobre desempeño en la conservación del conocimiento funcional del modelo. La degradación observada en las respuestas generadas por estos dos últimos métodos sugiere que no son adecuados para aplicaciones que requieren preservar la capacidad de inferencia del modelo original.

Adicionalmente, se incluyó en la comparación el método GPTQ, una técnica de cuantización post-entrenamiento que realiza una calibración de los pesos utilizando datos sintéticos con el fin de optimizar la representación comprimida del modelo. Para GPTQ, se utilizaron dos configuraciones: una con agrupaciones por columnas (GPTQ (col)), que es una configuración por defecto común en su implementación, y otra con un tamaño de grupo de 64 elementos (GPTQ (g=64)). Si bien GPTQ no es directamente comparable en condiciones con OrthoQuant (dado que este último no requiere calibraciones adicionales para su compresión), su inclusión resulta valiosa para posicionar la propuesta desarrollada en este

trabajo respecto a técnicas altamente optimizadas y representativas del estado del arte.

La evaluación cuantitativa se llevó a cabo utilizando dos *benchmarks* reconocidos. Para la tarea de modelado de lenguaje, se empleó el conjunto WikiText-2 (versión *raw*) (Merity et al., 2016), procesando el texto mediante una ventana deslizante de 2048 tokens con un desplazamiento de 512 tokens entre ventanas consecutivas. La métrica utilizada en esta tarea fue la *perplejidad*, ampliamente aceptada como indicador del desempeño de modelos generativos de lenguaje.

Por otro lado, para evaluar el desempeño en tareas de uso general, se utilizó el conjunto MMLU (Hendrycks et al., 2021) bajo un esquema de 5 disparos (*5-shot accuracy*), empleando la herramienta *lm-evaluation-harness* (Gao et al., 2023) con un tamaño de lote de 32. Este conjunto permite medir la capacidad del modelo para resolver tareas de razonamiento y conocimiento general sin ajuste fino específico para cada dominio.

Cabe destacar que todos los modelos comprimidos fueron convertidos al formato de Hugging Face para su evaluación directa, sin aplicar procedimientos adicionales de ajuste fino (*fine-tuning*), con excepción de GPTQ, cuyo procedimiento por defecto incluye una fase de calibración con datos sintéticos.

Finalmente, además de las métricas de desempeño mencionadas, se reportó el número promedio de bits por elemento en los pesos comprimidos, así como la reducción total del tamaño del modelo respecto a su versión original en formato FP16. Esto permitió cuantificar de manera precisa la eficiencia de compresión alcanzada por cada técnica, facilitando una comparación integral que contempla tanto la calidad funcional como la reducción efectiva en espacio de almacenamiento.

En cuanto a la eficiencia de compresión alcanzada por cada técnica, los resultados presentados en la Tabla 11 permiten analizar con mayor detalle la relación entre la reducción del tamaño del modelo, la cantidad promedio de bits por elemento comprimido y el desempeño medido en los *benchmarks* WikiText-2 y MMLU. Se observa que OrthoQuant logra una reducción del modelo comparable a RTN y GPTQ, con tasas de compresión que varían entre 37 % y 64 % dependiendo del nivel de cuantización seleccionado. En particular, para configuraciones equivalentes de compresión (por ejemplo, en torno a 4 bits por elemento), OrthoQuant alcanza la menor *perplejidad* en WikiText-2 (7.85) y un desempeño competitivo en MMLU (54.78), superando tanto a RTN como a GPTQ (col) y situándose ligeramente por debajo de GPTQ con agrupamiento por 64 elementos (*g=64*), el cual requiere calibración con datos sintéticos.

Tabla 11. Comparación de OrthoQuant, GPTQ y RTN en términos de compresión del modelo y desempeño. La notación *(col)* indica cuantización por columnas, mientras que *(g=64)* refiere al uso de un tamaño de grupo de 64 en GPTQ.

Método	wbits	Wikitext-2 ↓	MMLU ↑	Reducción del Modelo
GPTQ (col)	3.02	5204.0864	24.72	63.43 %
RTN	3.02	465.8009	25.07	63.43 %
OrthoQuant	3.1872	111.7325	26.51	62.58 %
GPTQ (g=64)	3.25	81.4605	45.85	62.00 %
GPTQ (col)	4.02	2625.0537	25.05	58.54 %
RTN	4.02	9.3143	47.24	58.55 %
OrthoQuant	4.2064	**7.8547**	54.78	57.60 %
GPTQ (g=64)	4.25	12.0092	56.11	57.04 %
RTN	5.02	7.3150	55.76	53.66 %
OrthoQuant	5.2272	**7.1064**	**57.00**	52.62 %
GPTQ (col)	8.02	6.9404	57.73	39.00 %
RTN	8.02	6.9507	57.37	39.01 %
OrthoQuant	8.1984	6.9434	57.42	38.10 %
GPTQ (g=64)	8.25	6.9391	57.66	37.20 %
Modelo Original	16.00	6.9405	57.32	–

Un aspecto crítico a resaltar es el comportamiento de GPTQ (col). Como se observa en la Tabla 11, esta configuración presenta una degradación significativa en el desempeño del modelo, particularmente a bajas tasas de bits (3 y 4 bits), con valores de perplejidad y MMLU considerablemente peores que los de RTN y OrthoQuant. Esto sugiere que, a pesar de su proceso de calibración con datos sintéticos, la cuantización basada en agrupaciones amplias como las columnas limita severamente la capacidad de GPTQ para preservar la información global del modelo, resultando en una pérdida de calidad sustancial en comparación con agrupamientos más pequeños (como g=64) que logran un mejor balance. En contraste, el enfoque de OrthoQuant no se basa en agrupaciones locales sino en la distribución global de la matriz gracias a la rotación ortonormal, lo que le confiere una mayor robustez y estabilidad en la preservación del desempeño, sin necesidad de datos externos.

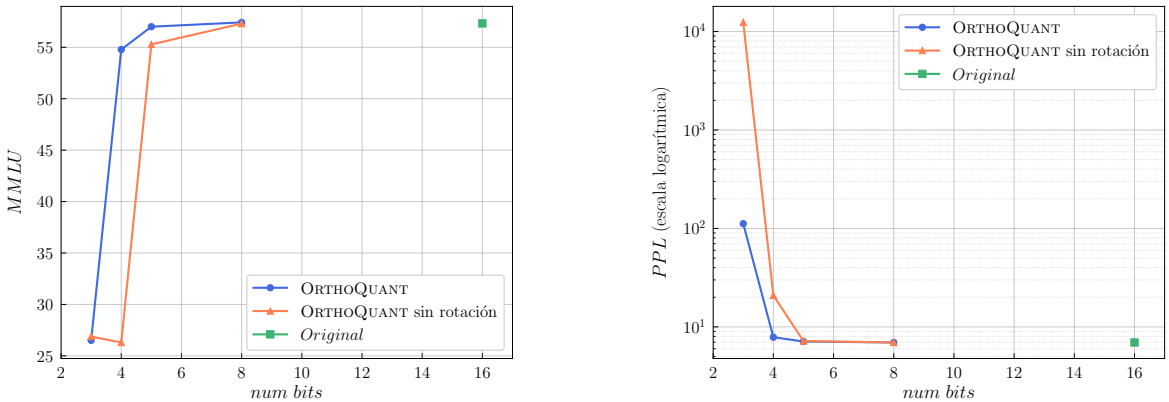
Asimismo, en niveles de compresión más agresivos (alrededor de 3 bits por elemento), OrthoQuant mantiene un desempeño aceptable en ambos *benchmarks*, especialmente si se considera que no requiere datos externos para su compresión, a diferencia de GPTQ. A medida que la cantidad de bits por elemento se incrementa hacia valores cercanos a 8, las diferencias de desempeño entre las técnicas se reducen considerablemente, como era de esperarse, acercándose al comportamiento del modelo original sin compresión. Sin embargo, incluso en este escenario, OrthoQuant conserva una ligera ventaja en términos de *perplejidad* en WikiText-2 respecto a RTN y GPTQ (col), evidenciando su capacidad para preservar la calidad de los pesos comprimidos.

Un aspecto relevante es que OrthoQuant logra este equilibrio entre compresión y calidad de manera independiente a datos de calibración, lo que representa una ventaja en términos de simplicidad del método. Además, esta técnica muestra una reducción promedio del tamaño del modelo ligeramente inferior a la obtenida por RTN o GPTQ en algunos casos, lo cual se explica por el almacenamiento adicional de *outliers*, necesario para garantizar una reconstrucción precisa de las matrices originales.

En resumen, los resultados cuantitativos confirman la competitividad de OrthoQuant respecto a técnicas del estado del arte como GPTQ y RTN, tanto en términos de reducción de tamaño como de mantenimiento de la capacidad funcional del modelo, destacando especialmente en configuraciones de compresión media (4–5 bits) donde logra un balance favorable entre eficiencia y desempeño en tareas representativas de uso real.

4.2.8. Estudio de ablación: desempeño con y sin rotación

Por último, se llevó a cabo un estudio de ablación con el objetivo de evaluar la contribución específica de la rotación ortonormal aleatoria dentro del esquema de compresión propuesto. Para ello, se comparó el desempeño del método OrthoQuant completo con una variante en la que se omitió la aplicación de la matriz de rotación previa a la cuantización, manteniendo inalteradas el resto de las etapas del procedimiento.



(a) Precisión en MMLU.

(b) Perplejidad en WikiText-2.

Figura 21. Comparación del desempeño del modelo con y sin el uso de una matriz de rotación. El eje horizontal representa la cantidad de bits utilizados para codificar los índices en la matriz de índices. La subfigura (a) muestra la precisión en el benchmark MMLU, mientras que la subfigura (b) presenta la perplexidad (en escala logarítmica) en el benchmark WikiText-2.

Los resultados de este análisis se presentan en la Figura 21, donde se muestra la evolución de las métricas de *perplejidad* (WikiText-2) y *accuracy* (MMLU) en función del nivel de compresión aplicado, expresado en términos de bits por elemento en la matriz de índices. Se observa que, a partir de niveles de compresión cercanos a 5 bits por elemento, la variante sin rotación presenta una degradación marcada tanto en *perplejidad* como en precisión en MMLU, en contraste con la versión completa de OrthoQuant, que mantiene un comportamiento estable y controlado bajo compresiones equivalentes.

Este resultado confirma que la inclusión de la rotación ortonormal aleatoria es un componente clave del método, al facilitar una redistribución más uniforme de la energía de las matrices en el espacio rotado. Esta propiedad es consistente con la teoría de representaciones de Kashin, la cual sostiene que dichas rotaciones permiten aproximaciones más estables y eficientes al dispersar la magnitud de los coeficientes. En consecuencia, la rotación previa no sólo mejora la calidad de la reconstrucción bajo fuertes compresiones, sino que también incrementa la compresibilidad de las matrices sin sacrificar de manera significativa la capacidad predictiva del modelo comprimido.

4.2.9. Discusión sobre la compresión de matrices

Los resultados obtenidos en la compresión de matrices correspondientes a los pesos de un modelo de lenguaje de gran tamaño revelan la solidez y versatilidad de OrthoQuant en un contexto distinto al de las bases de datos vectoriales. En primer lugar, el análisis del error de reconstrucción evidenció que OrthoQuant logra mantener una calidad comparable (e incluso superior) a la de técnicas clásicas como RTN, LRA y DCT, tanto en matrices de rango completo como en aquellas con redundancia estructural. Esta capacidad para adaptarse a distintos tipos de estructuras internas, sin requerir conocimiento previo sobre ellas, constituye una ventaja importante frente a métodos como LRA y DCT, cuya efectividad depende en gran medida de supuestos específicos sobre la naturaleza de las matrices (como la presencia de redundancias).

En términos de eficiencia computacional, OrthoQuant presentó un tiempo de compresión intermedio y un tiempo de descompresión competitivo. Dado que en aplicaciones prácticas como la inferencia en LLMs el proceso de descompresión puede ejecutarse en múltiples ocasiones, la eficiencia alcanzada en esta etapa lo posiciona como un candidato viable para utilizarse en sistemas de inferencia comprimida.

Desde el punto de vista funcional, la evaluación cualitativa mostró que OrthoQuant conserva de

manera notable la capacidad del modelo para generar respuestas coherentes y correctas, incluso bajo niveles de compresión elevados. El hecho de que el modelo comprimido reproduzca no sólo el contenido sino también el orden de las respuestas esperadas es indicativo de una preservación precisa de las relaciones semánticas codificadas en los pesos originales. Esta fidelidad supera la alcanzada por métodos como RTN, que si bien también preservan el contenido esencial, muestran signos leves de degradación funcional.

Por otra parte, los resultados obtenidos en la evaluación cuantitativa sobre conjuntos estandarizados como WikiText-2 y MMLU confirmaron que OrthoQuant se mantiene competitivo frente a técnicas especializadas como GPTQ, incluso sin requerir datos sintéticos para calibración ni ajustes posteriores. En escenarios donde no es posible acceder a datos de entrenamiento o calibración, esta propiedad adquiere un valor práctico considerable.

Finalmente, el estudio de ablación permitió confirmar empíricamente que la rotación ortonormal aleatoria es un componente esencial del método. Su eliminación da lugar a una degradación sustancial en el rendimiento del modelo comprimido, particularmente bajo compresiones agresivas. Este hallazgo corrobora el fundamento teórico del método y subraya el papel de la rotación como mecanismo de dispersión energética que incrementa la estabilidad de la cuantización.

En conjunto, estos resultados posicionan a OrthoQuant como una alternativa eficaz, simple y robusta para la compresión de modelos de lenguaje de gran escala, destacando por su independencia de datos, su capacidad de generalización frente a diferentes tipos de matrices y su buen equilibrio entre calidad funcional y eficiencia de compresión.

Capítulo 5. Discusión

Los resultados obtenidos en ambos escenarios de aplicación (compresión de vectores para búsqueda por similitud y compresión de matrices en modelos de lenguaje) permiten realizar una evaluación integral del método propuesto OrthoQuant, evidenciando su eficacia en contextos diversos. En el caso de vectores, se demostró que OrthoQuant no sólo logra compresiones efectivas en términos de bits por elemento, sino que también preserva una alta calidad en tareas de búsqueda por similitud. Estas tareas son esenciales en aplicaciones como los motores de recomendación, cuyo objetivo es recuperar los vectores más relevantes a partir de una consulta. Cabe destacar que, aunque una de las bases de datos empleadas en este trabajo se compone de *embeddings* con distribuciones originalmente gaussianas, el método mantuvo un buen desempeño. Sin embargo, su mayor aporte se observa al aplicarlo sobre bases de datos con distribuciones arbitrarias: el estudio de ablación con y sin rotación mostró que el *recall* mejoraba en este segundo caso. Esta versatilidad es una ventaja importante, ya que muchos métodos de compresión reducen su rendimiento o requieren adaptaciones específicas cuando los datos no siguen distribuciones gaussianas.

Los estudios exploratorios sobre variantes basadas en proyecciones aleatorias del tipo Johnson–Lindenstrauss sugieren que OrthoQuant puede adaptarse a otros objetivos, como la aceleración del cálculo del producto interno. Estas extensiones refuerzan su potencial competitivo, ya que, en este tipo de bases de datos, resulta igualmente importante reducir el tamaño manteniendo un alto *recall* y, al mismo tiempo, acelerar la búsqueda.

En el caso de matrices, OrthoQuant demostró ser competitivo tanto en términos de error de reconstrucción como en el desempeño funcional del modelo comprimido. A diferencia de algunas técnicas como LRA y DCT, cuya eficacia depende de la existencia de redundancia estructural en la matriz, OrthoQuant mostró un comportamiento robusto, manteniendo un rendimiento estable en matrices de rango bajo y de rango completo. Esto fue evidente al comprimir matrices del MLP: a diferencia de las del mecanismo de atención, que exhiben patrones definidos por las cabezas de atención, estas matrices presentan un mayor nivel de ruido. Pese a ello, OrthoQuant mantuvo un error de reconstrucción consistente, mientras que DCT y LRA redujeron su desempeño.

Este resultado es especialmente relevante en el contexto de la compresión de imágenes, donde algunos algoritmos ampliamente utilizados como JPEG (basados en la transformada DCT) pierden eficiencia en imágenes ruidosas, al no poder aprovechar redundancias estructurales y generar así una pérdida excesiva de información. En este escenario, OrthoQuant cubre un vacío de aplicación, mostrando potencial

para la compresión de imágenes y señales con alto contenido de ruido.

En comparación con RTN, OrthoQuant ofreció ventajas cualitativas al preservar con mayor precisión la salida del modelo y alcanzó un rendimiento cuantitativo superior en tareas como modelado de lenguaje (WikiText-2) y evaluación generalista (MMLU), sin requerir ajustes posteriores como los que exige GPTQ. Esta comparación con GPTQ resalta una de las características más distintivas de OrthoQuant: su naturaleza *data-free*. No requiere datos de entrenamiento para llevar a cabo la compresión ni depende de datos sintéticos, a diferencia de GPTQ. Esta cualidad lo diferencia de otros métodos del estado del arte que, si bien pueden ofrecer mejoras marginales en el desempeño, lo hacen a costa de una mayor complejidad de implementación y de una dependencia crítica de datos externos. En cambio, OrthoQuant mantiene un diseño simple y fácilmente replicable.

Como se ha observado, la estrategia de OrthoQuant centrada en preservar los *outliers* logra un desempeño competitivo frente a métodos del estado del arte que requieren procesos adicionales de calibración. La eficacia de esta técnica encuentra sustento indirecto en trabajos previos que destacan la importancia de conservar ciertos parámetros clave para mantener el rendimiento del modelo bajo condiciones de compresión elevada. En particular, Yu et al. (2024) muestran que la eliminación de un conjunto reducido de pesos específicos provoca un deterioro abrupto en la calidad del modelo, lo cual evidencia la existencia de parámetros críticos para la preservación del desempeño. En OrthoQuant, la introducción de una matriz de rotación ortonormal provoca una dispersión de la energía, de modo que un peso crítico en el espacio original puede redistribuirse en múltiples *outliers* en el espacio transformado. Preservar estos *outliers* se interpreta así como una forma indirecta de conservar las contribuciones esenciales identificadas por otros autores, asegurando la estabilidad del desempeño. De aquí se desprende la relevancia de la estrategia de OrthoQuant: mantener los *outliers* durante la codificación significa, en el espacio rotado, resguardar aquellas contribuciones críticas que sostienen la calidad del modelo.

Esta capacidad de preservar contribuciones esenciales adquiere un matiz distinto según el escenario de aplicación. En la búsqueda de vectores, muchos métodos priorizan mantener la similitud (producto punto o distancia) incluso a costa de sacrificar la reconstrucción fiel del vector. En cambio, en la compresión de matrices de pesos de un LLM, la fidelidad de reconstrucción es crucial, ya que estas matrices almacenan el conocimiento adquirido durante el entrenamiento. Una baja fidelidad puede traducirse rápidamente en una degradación severa del rendimiento del modelo en tareas de predicción, razonamiento y generación. OrthoQuant, al ofrecer dos estrategias de codificación, permite seleccionar la más adecuada según el propósito: ya sea priorizar la similitud o maximizar la fidelidad.

Un elemento común en ambas aplicaciones es la eficacia de la rotación ortonormal aleatoria previa a la cuantización. El estudio de ablación mostró que esta rotación mejora la compresibilidad sin degradar el rendimiento, al redistribuir de forma más uniforme la información, en concordancia con propiedades teóricas de las representaciones tipo Kashin. Este beneficio se observó tanto en vectores como en matrices, y permite que el método funcione de manera consistente en casos con alta redundancia (como las matrices de atención) y en casos de rango completo o baja redundancia (como las capas MLP), donde otros métodos suelen perder calidad.

No obstante, se identificaron limitaciones. Una de ellas es el sobrecosto por el almacenamiento de *outliers* en matrices y de medias y desviaciones estándar en vectores. Si bien estos sobrecostos son relativamente bajos (aprox. 0.20 y 0.17 bits por elemento, respectivamente), representan una fracción no despreciable del ancho de bits promedio. Una posible mejora sería emplear formatos de punto flotante más compactos (8 o incluso 4 bits), lo cual no se abordó en esta tesis debido a la falta de soporte en *PyTorch*, pero constituye una línea prometedora para reducir sustancialmente este costo.

Asimismo, la implementación podría optimizarse para reducir los tiempos de descompresión. Actualmente, el proceso incluye empaquetado y desempaquetado de bits, dado que en *PyTorch* la unidad mínima de almacenamiento es 1 byte, incluso para valores booleanos. Una posible optimización sería desarrollar *kernels* especializados que fusionen la descompresión con las operaciones de producto de matrices o vectores, evitando la etapa explícita de desempaquetado y acelerando así la inferencia en LLMs y la búsqueda en sistemas de recuperación (Guo et al., 2024).

En conjunto, la evaluación realizada confirma que OrthoQuant es un método general, simple y eficaz, capaz de adaptarse a distintos tipos de datos y tareas sin comprometer significativamente la calidad. Su bajo requerimiento de recursos, independencia de datos y desempeño competitivo lo posicionan como una alternativa viable para aplicaciones prácticas que exigen compresión eficiente y preservación funcional de la información. Adicionalmente, la versión orientada a vectores incorpora un modo *streaming*, que posibilita su inserción directa en las capas de atención de LLMs y modelos generativos. Esta capacidad habilita el manejo de contextos más extensos y puede optimizar la velocidad en la búsqueda de similitud.

Capítulo 6. Conclusiones y trabajo futuro

La actual era digital se caracteriza por la generación masiva de datos y el uso creciente de modelos de gran escala, lo que ha impulsado avances notables en sistemas de recomendación, procesamiento de lenguaje natural y visión por computadora. El funcionamiento de estos sistemas depende del almacenamiento y manipulación de estructuras de datos como vectores y matrices, cuyo tamaño y complejidad aumentan con cada nueva generación de modelos. Esta tendencia impone altos requerimientos de memoria y cómputo, restringiendo el acceso a instituciones con grandes recursos y elevando el impacto energético. La compresión eficiente surge así como una estrategia clave para optimizar el uso de memoria, acelerar la inferencia y reducir el consumo energético, permitiendo la reutilización de modelos cuyo entrenamiento ya supuso un coste computacional elevado.

En este contexto, esta investigación presentó OrthoQuant, un método de compresión general y *data-free*, capaz de adaptarse a distintos tipos de datos y tareas sin requerir reentrenamiento, calibración externa ni supuestos sobre la distribución original. Su diseño se apoya en dos componentes fundamentales: (i) una rotación ortonormal aleatoria previa a la cuantización, que redistribuye la información de forma más uniforme y mejora la compresibilidad; y (ii) la codificación eficiente de las estructuras rotadas, ajustada al contexto de aplicación.

Se evaluaron dos escenarios prácticos que ilustran esta adaptabilidad:

Compresión de vectores para búsqueda por similitud: OrthoQuant alcanzó altas tasas de compresión manteniendo un *recall* elevado, incluso en bases de datos con distribuciones arbitrarias donde otros métodos reducen su rendimiento. La codificación se optimiza para preservar la similitud entre vectores, priorizando la recuperación precisa de los vecinos más relevantes. Esto se logra mediante la aplicación del algoritmo de Lloyd–Max para aproximar la distribución gaussiana y el almacenamiento de la media y desviación estándar de cada vector. Esta estrategia resulta especialmente útil en motores de recomendación y sistemas de recuperación de información a gran escala.

Compresión de matrices de modelos de lenguaje (LLMs): En este caso, la fidelidad de reconstrucción es prioritaria, dado que las matrices almacenan el conocimiento adquirido durante el entrenamiento. Para ello, se mantiene sin pérdida la codificación de *outliers* y se cuantiza el resto de los elementos. OrthoQuant mostró un rendimiento robusto en matrices con y sin redundancias estructurales, superando a técnicas como DCT y LRA, que dependen de patrones de baja entropía para lograr compresiones efectivas. Destaca su capacidad para mantener un error de reconstrucción estable en capas MLP, más sensibles al ruido, donde otros métodos pierden eficacia.

Esta dualidad, un único método con dos esquemas de codificación ajustados al propósito de los datos, constituye uno de los aportes centrales de OrthoQuant. Su configuración es sencilla, ya que basta con definir el número de bits por código, a diferencia de enfoques como GPTQ o PQ, que requieren ajustar múltiples hiperparámetros dependientes de la dimensión original.

No obstante, se identificaron dos limitaciones principales: el sobrecosto asociado al almacenamiento de *outliers* y de las medias y desviaciones estándar (entre 0.06 y 0.20 bits por elemento), así como el tiempo de descompresión vinculado al empaquetado de bits y a la multiplicación por la matriz de rotación. Futuras líneas de trabajo incluyen la exploración de formatos de menor precisión para la representación de *outliers*, medias y desviaciones estándar (por ejemplo, de 8 o 4 bits), así como el desarrollo de *kernels* especializados que integren la descompresión con las operaciones de multiplicación, eliminando etapas intermedias y acelerando la inferencia, especialmente en modelos de lenguaje a gran escala (LLMs).

En síntesis, OrthoQuant se posiciona como un método de compresión robusto, adaptable y de uso sencillo, capaz de ofrecer reducciones sustanciales en tamaño con una mínima pérdida de calidad, y con potencial de optimización para consolidarse como una herramienta clave en el manejo eficiente de datos y modelos en el panorama tecnológico actual.

Literatura citada

Ahmed, N., Natarajan, T., & Rao, K. (1974). Discrete cosine transform. *IEEE Transactions on Computers*, *C-23*(1), 90–93. https://doi.org/10.1109/T-C.1974.223784.

Andoni, A., Indyk, P., & Razenshteyn, I. (2019). Approximate nearest neighbor search in high dimensions. In *Proceedings of the International Congress of Mathematicians (ICM 2018)*, (pp. 3287–3318). World Scientific. https://doi.org/10.1142/9789813272880_0182.

Aumüller, M., Bernhardsson, E., & Faithfull, A. (2020). Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems*, *87*, Article 101374. https://doi.org/10.1016/j.is.2019.02.006.

Balasubramanian, R., Bouman, C., & Allebach, J. (1995). Sequential scalar quantization of vectors: an analysis. *IEEE Transactions on Image Processing*, *4*(9), 1282–1295. https://doi.org/10.1109/83.413172.

Beis, J. & Lowe, D. (1997). Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1000–1006. https://doi.org/10.1109/CVPR.1997.609451.

Benítez López, J. (2016). El sistema de compresión jpeg. un pequeño paseo por la transformada discreta de fourier y la coseno. *Gaceta de la Real Sociedad Matemática Española*, *19*(1), 25–45. https://riunet.upv.es/bitstreams/46d6bb6e-85c8-463c-939a-347603af3d07/download.

Cai, Y., Yao, Z., Dong, Z., Gholami, A., Mahoney, M. W., & Keutzer, K. (2020). Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://openaccess.thecvf.com/content_CVPR_2020/papers/Cai_ZeroQ_A_Novel_Zero_Shot_Quantization_Framework_CVPR_2020_paper.pdf.

Chen, W., Wilson, J., Tyree, S., Weinberger, K. Q., & Chen, Y. (2016). Compressing convolutional neural networks in the frequency domain. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, 1475–1484. Association for Computing Machinery. https://doi.org/10.1145/2939672.2939839.

Choi, K., Hong, D., Park, N., Kim, Y., & Lee, J. (2021). Qimera: Data-free quantization with synthetic boundary supporting samples. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., & Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume *34*, 14835–14847. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2021/file/7cc234202e98d2722580858573fd0817-Paper.pdf.

Donoho, D. L. & Stark, P. B. (1989). Uncertainty principles and signal recovery. *SIAM Journal on Applied Mathematics*, *49*(3), 906–931. https://doi.org/10.1137/0149053.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*. https://doi.org/10.48550/arXiv.2407.21783.

Dupuis, E., Novo, D., O'Connor, I., & Bosio, A. (2020). On the automatic exploration of weight sharing for deep neural network compression. In *2020 Design, Automation & Test in Europe Conference & Exhibition*, 1319–1322. IEEE. https://past.date-conference.com/proceedings-archive/2020/pdf/0973.pdf.

Eckart, C. & Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, *1*(3), 211–218. https://doi.org/10.1007/BF02288367.

Folland, G. B. & Sitaram, A. (1997). The uncertainty principle: a mathematical survey. *The Journal of Fourier Analysis and Applications*, *3*(3), 207–238. https://doi.org/10.1007/BF02649110.

Frantar, E., Ashkboos, S., Hoefler, T., & Alistarh, D. (2022). Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*. https://doi.org/10.48550/arXiv.2210.17323.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., ... , & Zou, A. (2023). A framework for few-shot language model evaluation. https://doi.org/10.5281/zenodo.10256836.

Ge, T., He, K., Ke, Q., & Sun, J. (2014). Optimized product quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(4), 744–755. https://doi.org/10.1109/TPAMI.2013.240.

Gersho, A. & Gray, R. M. (2012). *Vector Quantization and Signal Compression*, (illustrated ed.), volume *159* of *The Springer International Series in Engineering and Computer Science*. Springer Science & Business Media. https://doi.org/10.1007/978-1-4615-3626-0.

Geva, M., Schuster, R., Berant, J., & Levy, O. (2020). Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*. https://doi.org/10.48550/arXiv.2012.14913Focustolearnmore.

Ghojogh, B. & Ghodsi, A. (2020). Attention mechanism, transformers, bert, and gpt: Tutorial and survey. https://hal.science/hal-04637647v1/file/Transformer_tutorial.pdf.

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., & Keutzer, K. (2022). A survey of quantization methods for efficient neural network inference. In *Low-power Computer Vision*, (pp. 291–326). Chapman and Hall/CRC. https://doi.org/10.48550/arXiv.2103.13630.

Golub, G. H., Hoffman, A., & Stewart, G. W. (1987). A generalization of the eckart-young-mirsky matrix approximation theorem. *Linear Algebra and its applications*, *88*, 317–327. https://doi.org/10.1016/0024-3795(87)90114-5.

Gray, R. & Neuhoff, D. (1998). Quantization. *IEEE Transactions on Information Theory*, *44*(6), 2325–2383. https://doi.org/10.1109/18.720541.

Guo, H., Brandon, W., Cholakov, R., Ragan-Kelley, J., Xing, E. P., & Kim, Y. (2024). Fast matrix multiplications for lookup table-quantized llms. *arXiv preprint arXiv:2407.10960*. https://doi.org/10.48550/arXiv.2407.10960.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., & Steinhardt, J. (2021). Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*. https://doi.org/10.48550/arXiv.2009.03300.

Jolliffe, I. (2011). Principal component analysis. In Lovric, M., editor, *International Encyclopedia of Statistical Science*, (pp. 1094–1096). Springer. https://doi.org/10.1007/978-3-642-04898-2_455.

Jongho, L. & Hyun, K. (2024). Dct-vit: High-frequency pruned vision transformer with discrete cosine transform. *IEEE Access*, *12*, 80386–80396. https://doi.org/10.1109/ACCESS.2024.3410231.

Jurafsky, D. & Martin, J. H. (2025). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*, (3a ed.). Online manuscript released August 24, 2025. https://web.stanford.edu/~jurafsky/slp3/.

Jégou, H., Douze, M., & Schmid, C. (2011). Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(1), 117–128. https://doi.org/10.1109/TPAMI.2010.57.

Lang, J., Guo, Z., & Huang, S. (2024). A comprehensive study on quantization techniques for large language models. In *2024 4th International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC)*, 224–231. https://doi.org/10.1109/ICAIRC64177.2024.10899941.

Ledoux, M. (2001). *The Concentration of Measure Phenomenon*, volume *89* of *Mathematical Surveys and Monographs*. American Mathematical Society. https://books.google.com/books/about/The_Concentration_of_Measure_Phenomenon.html?id=mCX_cWL6rqwC.

Li, Y., Yu, Y., Zhang, Q., Liang, C., He, P., Chen, W., & Zhao, T. (2023). LoSparse: Structured compression of large language models based on low-rank and sparse approximation. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., & Scarlett, J., editors, *Proceedings of the 40th International Conference on Machine Learning*, volume *202* of *Proceedings of Machine Learning Research*, 20336–20350. PMLR. https://proceedings.mlr.press/v202/li23ap/li23ap.pdf.

Linde, Y., Buzo, A., & Gray, R. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communications*, *28*(1), 84–95. https://doi.org/10.1109/TCOM.1980.1094577.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE Transactions on Information Theory*, *28*(2), 129–137. https://doi.org/10.1109/TIT.1982.1056489.

Lyubarskii, Y. & Vershynin, R. (2010). Uncertainty principles and vector quantization. *IEEE Transactions on Information Theory*, *56*(7), 3491–3501. https://doi.org/10.1109/TIT.2010.2048458.

Malkov, Y. A. & Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(4), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473.

Matsui, Y., Uchida, Y., Jégou, H., & Satoh, S. (2018). A survey of product quantization. *ITE Transactions on Media Technology and Applications*, *6*(1), 2–10. https://doi.org/10.3169/mta.6.2.

Max, J. (1960). Quantizing for minimum distortion. *IRE Transactions on Information Theory*, *6*(1), 7–12. https://doi.org/10.1109/TIT.1960.1057548.

Merity, S., Xiong, C., Bradbury, J., & Socher, R. (2016). Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*. https://doi.org/10.48550/arXiv.1609.07843.

Merkulov, D., Cherniuk, D., Rudikov, A., Oseledets, I., Muravleva, E., Mikhalev, A., & Kashin, B. (2024). Quantization of large language models with an overdetermined basis. *arXiv preprint arXiv:2404.09737*. https://doi.org/10.48550/arXiv.2404.09737.

Muja, M. & Lowe, D. G. (2014). Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(11), 2227–2240. https://doi.org/10.1109/TPAMI.2014.2321376.

Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., van Baalen, M., & Blankevoort, T. (2021). A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*. https://doi.org/10.48550/arXiv.2106.08295.

Noach, M. B. & Goldberg, Y. (2020). Compressing pre-trained language models by matrix decomposition. In Wong, K.-F., Knight, K., & Wu, H., editors, *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint*

Conference on Natural Language Processing, 884–889. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.aacl-main.88.

Nogueira, R. & Cho, K. (2019). Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*. https://doi.org/10.48550/arXiv.1901.04085.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572. https://doi.org/10.1080/14786440109462720.

Raiaan, M. A. K., Mukta, M. S. H., Fatema, K., Fahad, N. M., Sakib, S., Mim, M. M. J., Ahmad, J., Ali, M. E., & Azam, S. (2024). A review on large language models: Architectures, applications, taxonomies, open issues and challenges. *IEEE Access*, *12*, 26839–26874. https://doi.org/10.1109/ACCESS.2024.3365742.

Rokh, B., Azarpeyvand, A., & Khanteymoori, A. (2023). A comprehensive survey on model quantization for deep neural networks in image classification. *ACM Transactions on Intelligent Systems and Technology*, *14*(6), 50. https://doi.org/10.1145/3623402.

Saha, R., Srivastava, V., & Pilanci, M. (2023). Matrix compression via randomized low rank and low precision factorization. *Advances in Neural Information Processing Systems*, *36*, 18828–18872. https://proceedings.neurips.cc/paper_files/paper/2023/file/3bf4b55960aaa23553cd2a6bdc6e1b57-Paper-Conference.pdf.

Saraeb, A. (2024). Generation of random (generalized) orthogonal matrices. *arXiv preprint arXiv:2406.18963*. https://doi.org/10.48550/arXiv.2406.18963.

Scribano, C., Franchini, G., Prato, M., & Bertogna, M. (2023). Dct-former: efficient self-attention with discrete cosine transform. *Journal of Scientific Computing*, *94*(3), 67. https://doi.org/10.1007/s10915-023-02125-5.

Shi, J., Han, M., & Zhang, N. (2016). Uncertainty principles for discrete signals associated with the fractional fourier and linear canonical transforms. *Signal, Image and Video Processing*, *10*(8), 1519–1525. https://doi.org/10.1007/s11760-016-0965-7.

Tan, Z. & Brouwer, P. W. (2025). Operator spreading in random unitary circuits with unitary-invariant gate distributions. *Phys. Rev. B*, *111*, Article 184301. https://doi.org/10.1103/PhysRevB.111.184301.

Ulicny, M., Krylov, V. A., & Dahyot, R. (2021). Tensor reordering for cnn compression. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3930–3934. https://doi.org/10.1109/ICASSP39728.2021.9413944.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., & Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., & Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume *30*. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wang, N., Choi, J., Brand, D., Chen, C.-Y., & Gopalakrishnan, K. (2018). Training deep neural networks with 8-bit floating point numbers. *Advances in Neural Information Processing Systems*, *31*. https://proceedings.neurips.cc/paper_files/paper/2018/file/335d3d1cd7ef05ec77714a215134914c-Paper.pdf.

Xu, D., Tsang, I. W., & Zhang, Y. (2018). Online product quantization. *IEEE Transactions on Knowledge and Data Engineering*, *30*(11), 2185–2198. https://doi.org/10.1109/TKDE.2018.2817526.

Yu, H. & Wu, J. (2023). Compressing transformers: Features are low-rank, but weights are not! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume *37*, 11007–11015. https://doi.org/10.1609/aaai.v37i9.26304.

Yu, M., Wang, D., Shan, Q., Reed, C. J., & Wan, A. (2024). The super weight in large language models. *arXiv preprint arXiv:2411.07191*. https://doi.org/10.48550/arXiv.2411.07191.

Zechun, L., Changsheng, Z., Forrest, I., Chen, L., Yuandong, T., Igor, F., Yunyang, X., Ernie, C., Yangyang, S., Raghuraman, K., Liangzhen, L., & Vikas, C. (2024). MobileLLM: Optimizing sub-billion parameter language models for on-device use cases. In *Forty-first International Conference on Machine Learning*. https://openreview.net/forum?id=EIGbXbxcUQ.

Zhang, X., Qin, H., Ding, Y., Gong, R., Yan, Q., Tao, R., Li, Y., Yu, F., & Liu, X. (2021). Diversifying sample generation for accurate data-free quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 15658–15667. https://openaccess.thecvf.com/content/CVPR2021/papers/Zhang_Diversifying_Sample_Generation_for_Accurate_Data-Free_Quantization_CVPR_2021_paper.pdf.

Zhu, R.-J., Zhang, Y., Sifferman, E., Sheaves, T., Wang, Y., Richmond, D., Zhou, P., & Eshraghian, J. K. (2024). Scalable matmul-free language modeling. *arXiv preprint arXiv:2406.02528*. https://doi.org/10.48550/arXiv.2406.02528.

Zhuang, S. & Zuccon, G. (2021). Fast passage re-ranking with contextualized exact term matching and efficient passage expansion. *arXiv preprint arXiv:2108.08513*. https://doi.org/10.48550/arXiv.2108.08513.

Anexo A. Tablas de base de datos vectoriales

Tabla 12. Estadísticas de la distancia Euclidiana entre vectores originales y reconstruidos, normalizada respecto a la norma de los vectores originales, para vectores de dimensión 384 comprimidos con OrthoQuant. Se reportan los valores mínimos, promedio, máximos y desviación estándar de la distancia.

wbits	mín	media	máx	std
1.17	0.5347	0.6011	0.6703	0.0143
2.17	0.2916	0.3415	0.4145	0.0131
3.17	0.1537	0.1851	0.2648	0.0099
4.17	0.0817	0.0971	0.1805	0.0065
5.17	0.0469	0.0536	0.1279	0.0024
6.17	0.0285	0.0320	0.0830	0.0009
7.17	0.0156	0.0174	0.0548	0.0004
8.17	0.0080	0.0089	0.0409	0.0002

Tabla 13. Estadísticas de la distancia Euclidiana entre vectores originales y reconstruidos, normalizada respecto a la norma de los vectores originales, para vectores de dimensión 1024 comprimidos con OrthoQuant. Se reportan los valores mínimos, promedio, máximos y desviación estándar de la distancia.

wbits	mín	media	máx	std
1.06	0.5591	0.6020	0.6414	0.0088
2.06	0.3085	0.3420	0.3825	0.0080
3.06	0.1632	0.1853	0.2341	0.0061
4.06	0.0864	0.0972	0.1435	0.0041
5.06	0.0496	0.0536	0.0903	0.0016
6.06	0.0296	0.0320	0.0576	0.0005
7.06	0.0163	0.0174	0.0373	0.0003
8.06	0.0083	0.0089	0.0274	0.0002

Tabla 14. Estadísticas de la distancia Euclidiana entre vectores originales y reconstruidos, normalizada respecto a la norma de los vectores originales, para vectores de dimensión 960 comprimidos con OrthoQuant. Se reportan los valores mínimos, promedio, máximos y desviación estándar de la distancia.

wbits	mín	media	máx	std
1.07	0.5601	0.6036	0.6429	0.0074
2.07	0.3091	0.3402	0.3777	0.0070
3.07	0.1635	0.1833	0.2190	0.0053
4.07	0.0855	0.0961	0.1303	0.0032
5.07	0.0494	0.0533	0.0798	0.0011
6.07	0.0297	0.0320	0.0462	0.0005
7.07	0.0160	0.0174	0.0247	0.0003
8.07	0.0083	0.0089	0.0134	0.0001

Anexo B. Implementación de la Aproximación de Rango Bajo (LRA)

En este anexo se presenta la implementación utilizada para la aproximación de rango bajo (LRA) aplicada a las matrices de interés en esta tesis. La descripción teórica de esta técnica, basada en la descomposición en valores singulares (SVD), se encuentra detallada en la Sección 2.3.3.

El siguiente pseudocódigo muestra la función utilizada para calcular una aproximación de rango reducido de una matriz $A \in \mathbb{R}^{m \times n}$ mediante su descomposición SVD truncada. El rango r de la aproximación se determina en función del número total deseado de parámetros, conforme a la relación

$$p = r(m + n) \tag{32}$$

Algoritmo 6: Aproximación de Rango Bajo de una Matriz mediante SVD

Input: Matriz $\mathbf{A} \in \mathbb{R}^{m \times n}$, número total de parámetros deseados p

Output: Matrices $\mathbf{U}_r\mathbf{S}_r$ y $\mathbf{V}_r^\top$ que aproximan a $\mathbf{A}$

$\mathbf{U}, \mathbf{S}, \mathbf{V}^\top \leftarrow \text{SVD}(\mathbf{M})$; `// Descomposición en Valores Singulares (SVD)`

$r \leftarrow \left\lfloor \frac{p}{m+n} \right\rfloor$; `// Cálculo del rango truncado` r

`// Truncamiento de las matrices SVD`

$\mathbf{U}_r \leftarrow \mathbf{U}[:, 1:r]$; `// Primeras` r `columnas de` $\mathbf{U}$

$\mathbf{S}_r \leftarrow \mathbf{S}[1:r, 1:r]$; `// Primeros` r `valores singulares`

$\mathbf{V}_r^\top \leftarrow \mathbf{V}^\top[1:r, :]$; `// Primeras` r `filas de` $\mathbf{V}^\top$

$\mathbf{US}_r \leftarrow \mathbf{U}_r\mathbf{S}_r$; `// Multiplicación para obtener una sola matriz`

return $\mathbf{US}_r, \mathbf{V}_r^\top$

Esta implementación fue empleada en los experimentos de compresión de matrices presentados en el Capítulo 4. Como se observa en el Algoritmo 6, la salida corresponde a dos matrices cuyas dimensiones son menores que las de la matriz original. Suponiendo que la matriz original es cuadrada, es decir, que $m = n$, es importante notar que este método proporciona una compresión efectiva únicamente cuando el rango truncado r es menor a $\frac{n}{2}$, dado que el número total de parámetros de ambas matrices está definido por la ecuación (32).

El almacenamiento eficiente se logra al guardar únicamente estas dos matrices en cada capa del modelo, sustituyendo a la matriz original.

Para la descompresión durante la fase de inferencia, es suficiente calcular el producto de las matrices

resultantes:

$$\hat{A} = (US_r)V_r^\top \tag{33}$$

para obtener la matriz aproximada $\hat{A}$ correspondiente a cada capa.

Anexo C. Implementación de Compresión Basada en Transformada Discreta del Coseno (DCT)

En este anexo se describe la implementación utilizada para la compresión de matrices mediante la Transformada Discreta del Coseno (DCT), empleada en los experimentos de compresión presentados en el Capítulo 4. La descripción teórica de la DCT y su aplicación en compresión de datos puede consultarse en la Sección 2.3.4.

El método implementado consiste en aplicar la DCT de manera separada en bloques de 8×8 sobre la matriz de entrada, siguiendo un procedimiento análogo al empleado en algoritmos de compresión de imágenes como JPEG. Posteriormente, se seleccionan los coeficientes de mayor magnitud (en valor absoluto) de cada bloque, reteniendo un porcentaje predefinido de los componentes que concentran la mayor energía de la señal.

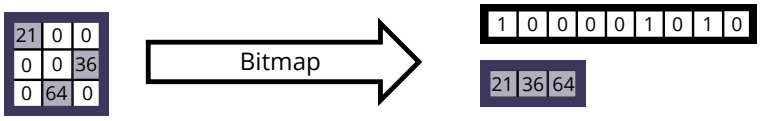


Figura 22. Representación en formato *bitmap* de una matriz dispersa. Este formato consiste en transformar la matriz dispersa en un arreglo binario (de bits) del mismo tamaño que indica la ubicación de los elementos no nulos, junto con un arreglo adicional que almacena únicamente dichos valores no nulos.

Los coeficientes seleccionados se almacenan en una representación dispersa mediante un esquema de codificación basado en *bitmaps*, lo que permite maximizar la eficiencia en el almacenamiento al evitar guardar explícitamente los ceros resultantes del truncamiento. Este esquema se ilustra en la Figura 22.

El pseudocódigo de la función de compresión es el siguiente:

Algoritmo 7: Compresión de matriz mediante DCT

Input: Matriz $A \in \mathbb{R}^{m \times n}$, porcentaje de coeficientes a conservar `percentage`

Output: Bitmap

Aplicar relleno a A para que sus dimensiones sean múltiplos de 8, en caso de ser necesario;
Dividir A en bloques de 8×8;
foreach *bloque* B **do**
    Aplicar DCT a B para obtener B_{DCT};
end
foreach *bloque* B_{DCT} **do**
    Conservar el `percentage` % de coeficientes de mayor magnitud;
end
Codificar la matriz dispersa resultante utilizando un esquema de *bitmap*;
return *bitmap*;

El procedimiento de descompresión llevado a cabo durante la inferencia consiste en revertir este proceso aplicando la transformada inversa DCT (IDCT) a cada bloque, tras reconstruir la matriz dispersa desde su representación en *bitmap*. La descompresión permite recuperar una aproximación de la matriz original:

Algoritmo 8: Descompresión de matriz mediante DCT

Input: Bitmap

Output: Matriz aproximada $\hat{A} \in \mathbb{R}^{m \times n}$

Reconstruir la matriz dispersa a partir del *bitmap*;
Dividir la matriz en bloques de 8×8;
foreach *bloque* B_{DCT} **do**
    Aplicar IDCT para obtener B;
end
Eliminar el relleno para restaurar las dimensiones originales;
return $\hat{A}$;

Esta implementación fue diseñada para preservar un porcentaje configurable de los coeficientes de mayor energía en el dominio de la DCT, permitiendo controlar el nivel de compresión y la calidad de reconstrucción resultante. Además, el uso de codificación por *bitmap* facilita el almacenamiento eficiente de los coeficientes no nulos, lo que reduce de manera significativa el espacio requerido en comparación con el almacenamiento de los bloques completos de DCT.

Anexo D. Implementación de Redondeo al Más Cercano

En este anexo se describe la implementación de la técnica de cuantización *Round-To-Nearest* (RTN), empleada en los experimentos de compresión de matrices presentados en el Capítulo 4. La descripción teórica de esta técnica puede consultarse en la Sección 2.3.2.

El procedimiento implementado realiza una cuantización escalar uniforme de los elementos de la matriz de pesos $A \in \mathbb{R}^{n\times m}$ de manera independiente para cada fila. Para cada fila i se calcula un factor de escala s_i y un desplazamiento (cero) z_i, de modo que los valores reales se transforman en índices enteros $q_{ij} \in \{0, 1, \ldots, 2^b - 1\}$, donde b es el número de bits de cuantización.

El cálculo de los parámetros de cuantización para cada fila se realiza de la siguiente manera:

$$s_i = \frac{\max_j a_{ij} - \min_j a_{ij}}{2^b - 1}, \tag{34}$$

$$z_i = \text{round}\left(-\frac{\min_j a_{ij}}{s_i}\right). \tag{35}$$

Los valores cuantizados se obtienen como:

$$q_{ij} = \text{round}\left(\frac{a_{ij}}{s_i} + z_i\right). \tag{36}$$

El procedimiento de compresión se resume en el siguiente pseudocódigo:

Algoritmo 9: Cuantización Round-To-Nearest (RTN)

Input: Matriz de pesos $A \in \mathbb{R}^{n\times m}$, número de bits de cuantización b

Output: Diccionario con: pesos empaquetados (`packed_weight`), escalas por fila (`s`), ceros por fila (`z`)

foreach *fila* i **do**
- Calcular $\max(a_i)$ y $\min(a_i)$; `// Calcular máximos y mínimos`
- $s_i = \frac{\max(a_i) - \min(a_i)}{2^b - 1}$; `// Calcular escalas`
- $z_i = \text{round}\left(-\frac{\min(a_i)}{s_i}\right)$; `// Calcular ceros`
- $Q_i = \text{round}\left(\frac{a_i}{s_i} + z_i\right)$; `// Cuantizar elementos`

end

Transformar cada fila Q_i a representación binaria con b bits por elemento;

Empaquetar los bits mediante *bit-packing* y almacenarlos en `packed_weight`;

return *Diccionario con* `packed_weight`, `s` *y* `z`;

El procedimiento de descompresión consiste en invertir este proceso. A partir de los pesos empaquetados, se realiza un *bit-unpacking* para obtener la matriz de índices cuantizados Q. Posteriormente, se recuperan los valores reales aproximados mediante la operación inversa de cuantización:

$$\hat{a}_{ij} = s_i \cdot (q_{ij} - z_i)\,, \tag{37}$$

donde $\hat{a}_{ij}$ representa la estimación del elemento original.

El pseudocódigo de la descompresión es el siguiente:

Algoritmo 10: Descompresión Round-To-Nearest (RTN)

Input: Pesos empaquetados (`packed_weight`), escalas (`s`), ceros (`z`), número de bits (b)

Output: Matriz de pesos aproximada $\hat{A} \in \mathbb{R}^{n \times m}$

Desempaquetar los bits de `packed_weight` mediante *bit-unpacking*;

Reconstruir la matriz de índices cuantizados Q con el número de bits b;

foreach *fila* i **do**

  $\hat{a}_i = s_i \cdot (q_i - z_i)$; `// Descuantizar elementos`

end

return $\hat{A}$;

Esta implementación permite controlar el nivel de compresión especificando el número de bits b utilizados para la cuantización, y garantiza una recuperación aproximada eficiente de los pesos originales mediante operaciones simples de escala y desplazamiento por fila. El empaquetamiento binario adicional contribuye a reducir de manera significativa el espacio de almacenamiento requerido.