UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

EXTRACCIÓN DE VECTORES DE CARACTERÍSTICAS COMPACTOS PARA LA RECUPERACIÓN DE IMÁGENES BASADAS EN DIBUJOS USANDO MODELOS CONVOLUCIONALES

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

PABLO IGNACIO TORRES DESSI

PROFESOR GUÍA:
JOSÉ M. SAAVEDRA RONDO

MIEMBROS DE LA COMISIÓN:
FELIPE BRAVO MARQUEZ
MAURICIO CERDA VILLABLANCA

SANTIAGO DE CHILE
2021

Resumen

La recuperación de imágenes basada en dibujos (en inglés Sketch-Based Image Retrieval o SBIR) ha demostrado un creciente interés en la comunidad de visión por computadora, lo que genera un gran impacto en aplicaciones reales. Por ejemplo, SBIR aporta un mayor beneficio a los motores de búsqueda de comercio electrónico porque permite a los usuarios formular una consulta simplemente dibujando lo que necesitan comprar.

La eficacia lograda por los métodos del estado del arte en la recuperación basada en dibujos ha hecho posible pasar rápidamente del contexto científico a las aplicaciones industriales. Sin embargo, aplicaciones como un motor de búsqueda de comercio electrónico también imponen nuevos retos más allá de la eficacia en sí. En esta línea, el tiempo de búsqueda y el consumo de memoria son dos aspectos que los métodos SBIR deben considerar. Para solucionar esto, algunos autores han propuesto el uso de representaciones compactas, sin embargo, estas degradan drásticamente el rendimiento del sistema para altos niveles de reducción.

En este trabajo se abordó el problema planteado mediante redes convolucionales. Para ello se entrenó una red que hizo de modelo base inspirándose en trabajos del estado del arte y se compararon seis métodos de reducción de dimensión en dos datasets diferentes. Se utilizó Flickr15K y un dataset de eCommerce; este último es otro aporte de este trabajo.

Metodológicamente, se abordó este trabajo a través de la comparación de distintos métodos de reducción de dimensión, el análisis de componentes principales (PCA) y Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP), los cuales permiten generar representaciones de baja dimensión en base a representaciones de mayor dimensión, como las generadas por el modelo base. Adicionalmente, se evaluaron métodos cuya función es generar representaciones de baja dimensión en base a redes convolucionales, como es el caso de Deep Supervised Hashing (DSH) y Deep Triplet Supervised Hashing (DTSH). Finalmente se propuso una Capa de Reducción Simple (CRS) y una Capa Binaria Sigmoidal (CBS) como estrategias adicionales a las anteriormente descritas.

Los resultados permitieron determinar las fortalezas de cada método. Se observo la gran eficacia que presenta UMAP, logrando obtener un mAP de 0.57 en el dataset de Flickr15K y 0.21 en el dataset de eCommerce, incluso mejorando el modelo base hasta en un 35 %, a la vez que reduce los tiempos de búsqueda en un 80 %.

Como conclusión, se determinó que los métodos evaluados lograron el objetivo de obtener representaciones compactas sin degradar la calidad de los resultados, logrando así reducir el tiempo y la memoria utilizada por los sistemas SBIR.

A mis padres, Julio y Gianinna, a quienes les debo todo.

Agradecimientos

En primer lugar quisiera agradecerle a mis padres, Julio y Gianinna. Gracias a mi padre, por toda la preocupación y esfuerzo que me ha dedicado, por enseñarme matemáticas desde pequeño, por todos esos sábados que tuvo que levantarse a las 6 de la mañana, por siempre motivarme a aprender más. Gracias a mi madre, que siempre me ha impulsado a ser una mejor persona y que lo ha dado todo con tal de sacarnos a mi y a mis hermanos adelante.

Quiero agradecerle a todos los amigos que hice en la universidad, por hacer estos años mucho más amenos. En especial a Sven, con quien pasamos incontables aventuras. Agradecer también a los chicos de Zippedi, por todas las enseñanzas y su amistad.

A mis profesores del colegio, Ana, Magaly y Lucho, por ser excelentes docentes y fomentar mi interés por la matemática y las ciencias.

También quisiera agradecerle a mi profesor guía, José Saavedra, por todo el apoyo brindado a lo largo de este trabajo y por empujarme a ser un profesional más integral.

Y finalmente a Kentaki, por subirme el ánimo con sus ridiculeces.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación

La recuperación de imágenes basada en dibujos (Sketch-Based Image Retrieval o SBIR) es un subproblema de recuperación de imágenes por contenido que consiste en recibir un dibujo y buscar dentro de un catálogo las imágenes más parecidas a la consulta. En la figura 1.1 se observan ejemplos de SBIR. Para cada fila se muestra la consulta junto con las primeras imágenes recuperadas.

Figura 1.1: Ejemplos de consultas mediante SBIR.

SBIR ha cobrado gran popularidad en el último tiempo dado que representa una forma efectiva de realizar consultas. Otros tipos de búsqueda requieren imágenes de ejemplo, las cuales no siempre se tienen a mano, o describir lo que se busca con palabras clave, cosa que no es tan fácil a la hora de describir formas o diseños. En el caso de SBIR basta con que la persona tenga una idea de lo que busca para que pueda dibujarlo y hacer las consultas. Otro factor que ha vuelto relevante este tipo de búsqueda es su fácil adopción, con la masificación de los celulares y otros dispositivos táctiles cualquier persona es capaz de realizar un dibujo desde cualquier parte.

SBIR muestra un gran potencial para aplicaciones en el retail y comercio electrónico (e-commerce), donde permite buscar productos dentro de los catálogos de tiendas a través de dibujos de estos mismos. Como ejemplo del potencial de esta tecnología se puede mencionar a Impresee, empresa chilena orientada al retail. Actualmente cuenta con una aplicación que combina múltiples sistemas de búsqueda para hacer consultas en catálogos de tiendas, incluyendo búsqueda basada en dibujos.

La figura 1.2 corresponde a una captura del sitio *pepeganga.com*, en el que se muestra cómo esta herramienta permite buscar productos mediante dibujos.

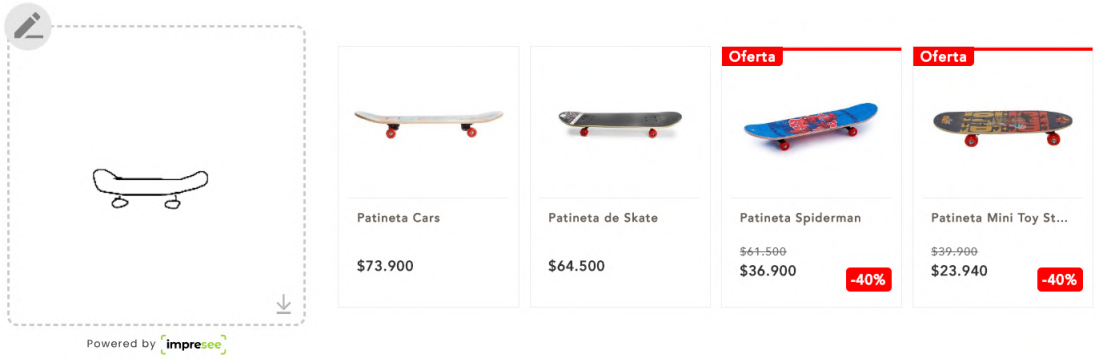

Figura 1.2: Consulta de SBIR en catálogo de tienda.

Sin embargo, un problema de estos sistemas es que mientras más datos se posean se vuelve más difícil la tarea de buscar imágenes similares, requiriendo más almacenamiento y mayores tiempos de búsqueda. En la actualidad se trabaja con volúmenes de datos cada vez más grandes, debido a esto, la escalabilidad de estos métodos se vuelve un factor cada vez más importante. Una posible solución es reducir el tamaño de las representaciones usadas por estos métodos.

El tema de reducción de dimensionalidad esta ganando popularidad en el ámbito de recuperación de imágenes, sin embargo, pocos trabajos han estudiado el caso particular de SBIR. En este trabajo se propone extender una arquitectura de SBIR [4] y explorar distintas estrategias para obtener vectores de características de menor tamaño sin afectar el desempeño del sistema de búsqueda.

1.2. Relevancia del problema

SBIR es un problema desafiante por múltiples razones, requiere acercar elementos de dominios distintos, como lo son las imágenes de la vida real y los dibujos, a un mismo espacio de características. Presenta una alta variabilidad entre dibujos de una misma clase, dado que cada persona tiene un estilo diferente de dibujo. Además, cuando uno hace un dibujo no tiene la imagen que busca como referencia, por lo que los dibujos tienden a presentar diferencias con respecto a las imágenes que se desea buscar.

Debido a esto se ha observado durante la última década un interés creciente en la comunidad de visión por computadora sobre recuperación de imágenes basada en dibujos. Al comienzo de este período, los investigadores se centraron en las características de bajo nivel, proponiendo variaciones del histograma de orientaciones para tratar imágenes como si fueran dibujos [15, 3, 7]. Otros investigadores también utilizaron características de mediano nivel [16, 18] para representar dibujos. Sin embargo, la explosión del Deep Learning, que mostró una eficacia superior en diversas tareas de visión por computadora redireccionó la investigación SBIR hacia estos modelos que aumentan significativamente la eficacia subyacente [20, 2]. Al final, las arquitecturas basadas en la redes siamesas con triplet loss, entrenado de manera incremental, mostraron el mejor rendimiento en diferentes datasets de SBIR [2].

La eficacia lograda por los métodos del estado del arte en la recuperación basada en dibujos

ha hecho posible pasar rápidamente del contexto científico a las aplicaciones industriales. Sin embargo, aplicaciones como un motor de búsqueda de comercio electrónico también imponen nuevos retos más allá de la eficacia en sí. En esta linea, el tiempo de búsqueda y el consumo de memoria son dos aspectos que los métodos SBIR deben considerar. Una forma de abordar este desafío consiste en proponer métodos que produzcan representaciones en un espacio de características de baja dimensión.

Por lo general, los métodos SBIR producen representaciones de punto flotante en un espacio de características de alta dimensión (p.ej. 1024). Este hecho tiene un gran impacto en los recursos requeridos por una empresa que ofrece un servicio de motor de búsqueda, ya que todos los vectores de características de un catálogo de tienda deben cargarse en la memoria para permitir la consulta en línea. Asumiendo que se tiene un catálogo con un millón de productos y la representación actual ocupa 2048 números flotantes, el catálogo ocuparía 8 Gigabytes en RAM. Si se lograra reducir el tamaño de los vectores a 512 valores binarios, el catálogo pasaría a ocupar sólo 61 Megabytes, lo cual es una disminución considerable, permitiendo tener más catálogos en la misma memoria o cargar catálogos de mayor tamaño, sin mencionar las mejoras en la velocidad de consulta.

Para hacer frente a estos problemas, algunas investigaciones han propuesto representaciones compactas [11, 1]. Sin embargo, estos métodos todavía muestran una baja efectividad en la recuperación. Es debido a esto que la obtención de representaciones en un espacio de baja dimensionalidad resulta una tarea relevante.

1.3. Objetivo General

Desarrollar y evaluar métodos para generar vectores de características compactos en el contexto de recuperación de imágenes basada en dibujos.

1.4. Objetivos Específicos

1. Desarrollar y entrenar una arquitectura base para extraer vectores de características de imágenes y dibujos en el contexto de SBIR
2. Explorar e implementar métodos de reducción de dimensionalidad para la tarea de SBIR que se basen en los vectores de característica obtenidos con la arquitectura base.
3. Diseñar e implementar un bloque de reducción simple, capaz de acoplarse a una red y aprender un espacio de características en el contexto de SBIR, como una exploración inicial del problema.
4. Diseñar e implementar bloques de reducción binaria, capaces de acoplarse a una red y aprender una representación binaria de las características en el contexto de SBIR.
5. Entrenar los bloques implementados y experimentar con distintos parámetros en busca de mejores resultados.
6. Evaluar el desempeño de los bloques en términos de efectividad (mAP), tiempo de procesamiento y memoria RAM necesaria.

1.5. Estructura del documento

El resto del documento se estructura como sigue: En el capítulo 2 se presenta el marco teórico, donde se explican los conceptos necesarios para entender el trabajo realizado. Además, se revisan los métodos del estado del arte explorados en la investigación. En el capítulo 3 se explica el trabajo realizado y las soluciones diseñadas para resolver el problema. En el capítulo 4 se muestran y analizan los resultados obtenidos respecto a distintas métricas. Finalmente, se expone la conclusión del trabajo realizado y se proponen posibles mejoras y caminos futuros.

Capítulo 2

Marco teórico y estado del arte

2.1. Redes Neuronales

Las redes neuronales corresponden a un modelo computacional ampliamente utilizado en la actualidad. Se inspiran en el funcionamiento de nuestro cerebro y consisten en un conjunto de unidades, llamadas neuronas, conectadas entre sí para transmitirse señales. La información de entrada atraviesa la red neuronal (donde se somete a diversas operaciones) produciendo valores de salida.

Cada neurona está conectada con otras a través de enlaces. En estos enlaces el valor de salida de la neurona anterior es multiplicado por un peso. Estos pesos en los enlaces pueden incrementar o inhibir el estado de activación de las neuronas adyacentes. Del mismo modo, a la salida de la neurona, puede existir una función que modifica el valor resultante o impone un límite que no se debe sobrepasar antes de propagarse a otra neurona. Esta función se conoce como función de activación.

2.1.1. Redes convolucionales

Las redes convolucionales son un tipo de red neuronal ampliamente utilizada en el contexto de visión por computadora. Estas se caracterizan por el uso de capas convolucionales, donde cada neurona centra su atención en un sector reducido de la capa anterior, imitando el comportamiento de las neuronas en la corteza visual del cerebro.

Las redes convolucionales pueden incluir capas de pooling local y/o global junto con capas convolucionales tradicionales. Las capas de pooling reducen las dimensiones de los datos al combinar las salidas de un grupo de neuronas en una capa en una sola neurona en la siguiente. El pooling local combina grupos pequeños, normalmente se agrupan las neuronas en cuadrados de 2 x 2, mientras que el pooling global actúa sobre todas las neuronas de la capa anterior. Existen dos tipos de pooling que se usan con frecuencia, max y average pooling. El max pooling utiliza el valor máximo de cada grupo local de neuronas de la capa anterior, mientras que el average pooling toma el valor promedio.

Una vez la red extrae características de la imagen, es común hacer un pooling global y luego utilizar capas Fully-Connected (FC), donde cada neurona está conectada a cada neurona de la capa

anterior, para clasificar la imagen.

Uno de los primeros trabajos utilizando redes convolucionales fue el de LeNet[9] en 1998, En el que se entrenó una red convolucional para la tarea de reconocer dígitos manuscritos. Sin embargo, no se observó mayor interés en el área hasta el año 2012, cuando AlexNet[8] ganó la competencia de ImageNet con una red convolucional de 8 capas como la que se muestra en la figura 2.1. Una de las innovaciones de este trabajo la adición de la función de activación llamada Rectified Linear Unit (ReLU), la cual es ampliamente utilizada en la actualidad. En la figura Y se muestra la curva de la función de activación ReLU.

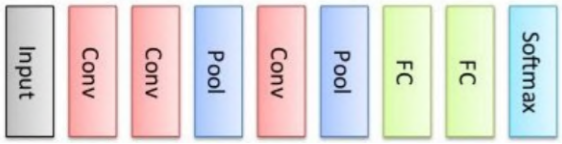

Figura 2.1: Diagrama de arquitectura Alexnet, Fuente: [17]

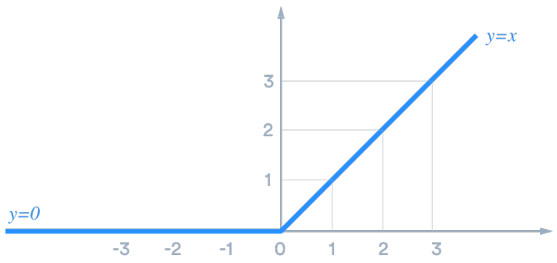

Figura 2.2: Función de activación ReLU.

Posteriormente VGG[21] obtuvo grandes resultados con una red mucho más profunda, utilizando un total de 16 capas. Sin embargo, esto venía acompañado de una cantidad de parámetros entrenables mucho mayor y un aumento considerable en la capacidad de cómputo y tiempo requerido para el entrenamiento. En la figura 2.3 se muestra la arquitectura de VGG 16.

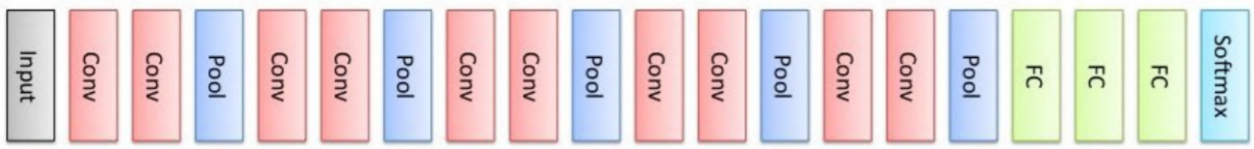

Figura 2.3: Diagrama de arquitectura VGG 16, Fuente: [17]

Llegado a este punto, los científicos se toparon con el problema de que agregar más capas a las redes ya no presentaba mejoras en los resultados y en algunas ocasiones incluso los empeoraba. Este problema se debía principalmente a que el algoritmo de back propagation, va propagando la pérdida desde el final de la red hasta el principio, pero se va desvaneciendo a medida que avanza por las capas, razón por la cual para redes muy profundas el aprendizaje de las primeras capas se vuelve cada vez más difícil, haciendo imposible que la red aprenda.

ResNet[5] logró resolver este problema con la adición de bloques residuales, los cuales evitan que el gradiente se desvanezca generando “atajos” en la red. Para generar estos atajos lo que se hace es sumar el input del bloque residual a la salida de este, como se muestra en la figura 2.4.

De esta manera es posible propagar la pérdida a través de estos atajos sin verse afectados por el desvanecimiento.

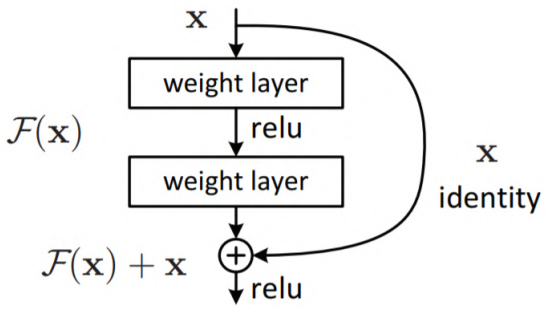

Figura 2.4: Diagrama de bloque residual, Fuente: [5]

En la figura 2.5 se observa la arquitectura ResNet-50, caracterizada por poseer cinco bloques de convolución, los cuales a su vez se dividen en sub-bloques residuales, sumando un total de 50 capas.

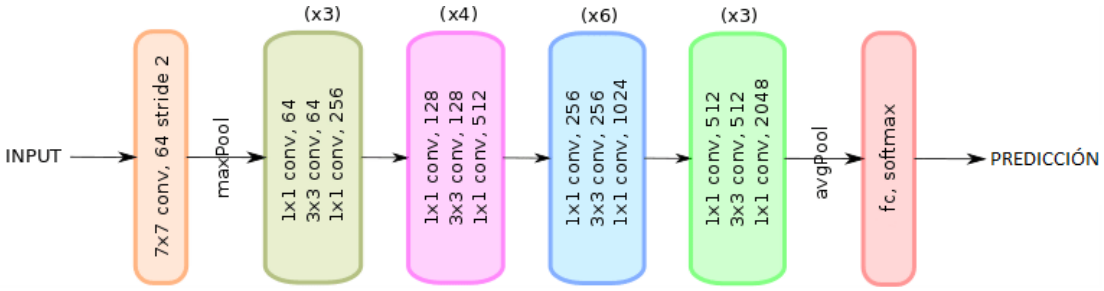

Figura 2.5: Diagrama de arquitectura ResNet-50

Una mejora posterior fueron las capas *Squeeze-and-Excitation* [6], que se agregan al final de cada bloque residual y buscan ponderar qué tanta importancia se le debe dar a cada canal del output. En la figura 2.6 se puede apreciar un diagrama de cómo se implementan.

2.1.2. Redes Siamesas

Las redes siamesas son un tipo de red neuronal que contienen dos o más redes idénticas, estas redes comparten la misma arquitectura y normalmente también comparten los pesos.

Estas redes son ampliamente utilizadas en problemas de búsqueda por similitud, ya que permiten entrenar el modelo con dos imágenes a la vez, generando dos embeddings distintos que pasan por una función de pérdida que los compara, permitiendo entrenar a la red para que acerque o aleje las representaciones de cada par de imágenes según corresponda. En la figura 2.7 se observa un ejemplo de una red siamesa, que posee dos arquitecturas idénticas y que comparten pesos. A ambas redes se les entrega una imagen, el *anchor* y el par que puede ser de la misma o distinta clase. En este caso se le entregan dos imágenes de la misma clase. Ambas redes generan embeddings de las imágenes recibidas, los cuales son comparados por la función de pérdida.

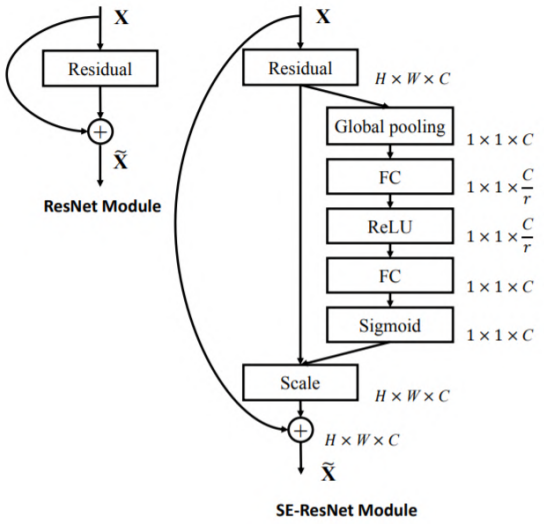

Figura 2.6: Diagrama de capas *Squeeze-and-Excitation* aplicadas en un bloque residual. Fuente: [6]

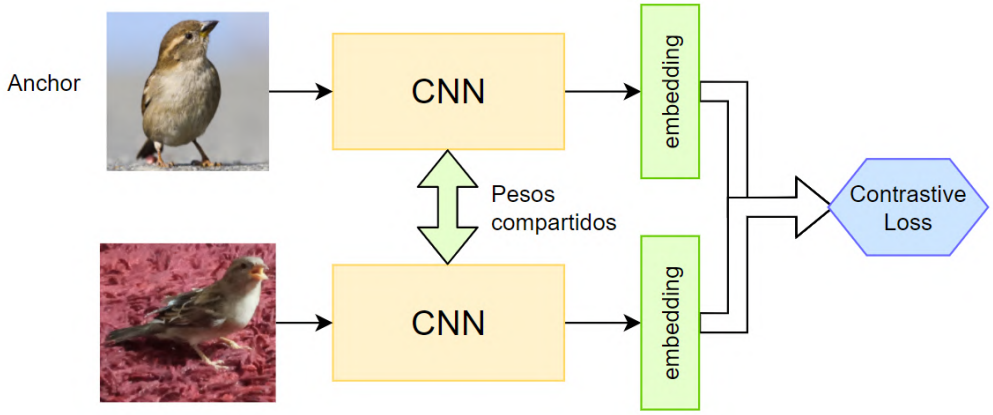

Figura 2.7: Ejemplo de red siamesa.

La función de pérdida más utilizada con redes siamesas es la *Contrastive Loss*, la que se define de la siguiente manera:

$$\mathcal{L}_c(u_\text{i}, u_j) = y \cdot D(u_\text{i}, u_j) + (1 - y) max(0, \lambda - D(u_\text{i}, u_j))$$

Donde u_i y u_j son los embeddings de las imágenes, y valdrá 1 si ambas imágenes pertenecen a la misma clase y 0 de lo contrario, $D(,)$ puede ser cualquier función de distancia, aunque normalmente se usa distancia L_2, y λ es un margen que indica qué tan alejadas deben estar las imágenes de distintas clases. Se puede observar que en aquellos casos en los cuales ambas imágenes son de la misma clase, sólo sobrevive el primer término, por lo que la red tratará de minimizar la distancia entre las representaciones de las imágenes. Por el contrario, si las imágenes pertenecen a clases distintas, el segundo término sobrevive por lo que la red intentará maximizar la distancia.

2.1.3. Redes Triples

Las redes triples son una variación de las redes siamesas en las que se usan tres redes idénticas en vez de dos. Posee la ventaja de permitir entrenar las redes con una imagen positiva (de la misma clase) y negativa (de distinta clase) a la vez, acelerando el entrenamiento y en ocasiones llevando a mejores resultados. En la figura 2.8 se presenta un ejemplo de una red triple.

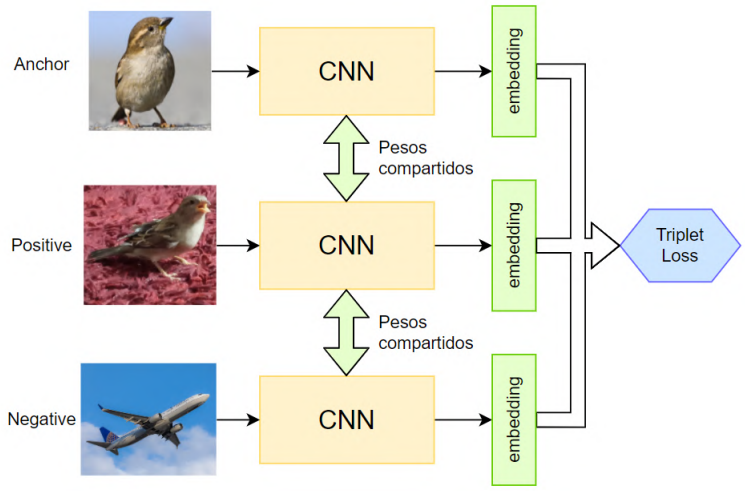

Figura 2.8: Ejemplo de red triple.

Para las redes triples se suele utilizar la *Triplet loss* que se calcula de la siguiente manera:

$$\mathcal{L}_t(u_\text{i}, u_j, u_k) = max(0, \lambda - D(u_\text{i}, u_k) + D(u_\text{i}, u_j))$$

Donde u_i, u_j y u_k son los embeddings del anchor, la imagen positiva y la imagen negativa respectivamente, $D(,)$ puede ser cualquier función de distancia, aunque normalmente se usa distancia L_2, y λ es un margen. Se puede observar que esta función es muy similar a la *Contrastive Loss* salvo que en este caso se calculan ambos términos a la vez.

2.2. Recuperación de imágenes

Los sistemas de SBIR se pueden dividir en dos grandes grupos, SBIR con características escogidas a mano y SBIR con *deep learning*. El primero se caracteriza por el uso de descriptores tradicionales como pueden ser descriptores basados en gradientes, texturas y color, entre otros; mientras que el segundo utiliza características extraídas a partir de redes convolucionales. Este trabajo utilizará el segundo enfoque.

Una de las grandes dificultades de SBIR es la necesidad de asignar vectores de características similares a elementos de dominios distintos, que son los dibujos y las imágenes. Una solución a este problema es acercar las imágenes al dominio de los dibujos [14], extrayendo los bordes de las imágenes con técnicas como Canny o Sketch-tokens. Sin embargo, esta solución muestra resultados inferiores al estado del arte, lo que se atribuye a que las imágenes generadas siguen perteneciendo

a un dominio distinto al de los dibujos, por lo que es difícil que una red pueda representar ambas de la misma forma.

Otra de forma de enfrentar este problema es entrenando dos redes por separado [20], una para los dibujos y otra para las imágenes. Posteriormente se entrenan ambas redes en conjunto usando una arquitectura de redes siamesas y *contrastive loss* con el fin de acercar ambas representaciones entre sí.

Finalmente, las arquitecturas que han mostrado mejores resultados en la búsqueda por similitud son las que combinan redes siamesas con triplet loss [23, 2, 4], especialmente cuando se desea una búsqueda detallada. Entre estos métodos, el trabajo de Bui et al. [2] destaca porque logra una alta precisión en conjuntos de datos públicos en la recuperación de imágenes basadas en dibujos. Este método propone una metodología incremental de cuatro etapas para entrenar una red capaz de producir un espacio de características en el que los dibujos y las fotos pueden coexistir. Las cuatro etapas están diseñadas de tal manera que pueden mejorar gradualmente su poder discriminatorio. Con este fin, utilizan también redes siamesas y tripletas conjuntamente con una pérdida de entropía cruzada, pero entrenadas desde una similitud de grano grueso al principio hasta una similitud de grano fino al final.

2.3. Reducción de dimensionalidad

Con respecto a este problema, algunos investigadores se han centrado en reducir el espacio de alta dimensión en la recuperación basada en dibujos y en la recuperación de imágenes. Sin embargo, los algoritmos de reducción propuestos degradan rápidamente la eficacia de la recuperación.

La forma más sencilla de obtener una representación binaria es agregando una función de activación al final de la red para que a la salida se le pueda aplicar un umbral y obtener un valor binario. Ejemplos de estas funciones son sigmoid y tanh. Podemos utilizar, por ejemplo, un umbral igual a 0,5 en el caso de la función sigmoid. Este método tiene la ventaja de ser fácil de incorporar a una red existente.

Deep Supervised Hashing (DSH) [10] y Deep Triplet Supervised Hashing (DTSH) [22] se enfocan en obtener un espacio de características binarias para la recuperación de imágenes. DSH utiliza una red con tres capas convolucionales y dos capas fully-connected. El método obliga a que la salida sea binaria mediante la siguiente función de pérdida por pares:

$$\begin{aligned}\mathcal{L}(b_1, b_2) = &\frac{1}{2}(1-y)||b_1 - b_2||_2^2 \\ &+ \frac{1}{2} y \operatorname{máx}(0, m - ||b_1 - b_2||_2^2) \\ &+ \lambda(||\,|b_1| - \mathbf{1}||_1 + ||\,|b_2| - \mathbf{1}||_1)\end{aligned}$$

donde los dos primeros términos funcionan como contrastive loss y el tercer término es el término de binarización que obliga a las salidas a ser binarias. Aquí, m es un parámetro de margen, $y = 1$ si $b1$, $b2$ son similares y 0 en caso contrario, y λ es un parámetro de ponderación que controla la fuerza del término de binarización. Una vez entrenados, los códigos binarios se pueden obtener

aplicando la función de signo.

Deep Triplet Supervised Hashing (DTSH) [22] es otro método para obtener representaciones binarias. La pérdida de DTSH proviene de la probabilidad de producir buenas tripletas, donde la similitud entre pares se calcula como el producto interno entre los vectores de características correspondientes. La pérdida se define de la siguiente manera:

$$\begin{aligned}\mathcal{L}(u_{\mathrm{i}}, u_j, u_k) = \log(1 + \mathrm{e}^{\Theta_{\mathrm{i},j} - \Theta_{\mathrm{i},k} - \alpha}) - (\Theta_{\mathrm{i},j} - \Theta_{\mathrm{i},k} - \alpha) \\ + \lambda(||b_{\mathrm{i}} - u_{\mathrm{i}}||_2^2 + ||b_j - u_j||_2^2 + ||b_k - u_k||_2^2)\end{aligned}$$

Donde $\Theta_{\mathrm{i},j} = \frac{1}{2} u_{\mathrm{i}}^T u_j$, $b_{\mathrm{i}} = sgn(u_{\mathrm{i}})$, α es un parámetro de margen, y λ es un parámetro de ponderación que controla la fuerza del término de binarización. De manera similar a DSH, el tercer término se encarga de imponer la binarización de la salida.

El uso de técnicas de reducción sobre las representaciones generadas por un modelo es otra alternativa para producir representaciones compactas. Por ejemplo, Bui et al. [1] utiliza Product Quantization y PCA para obtener vectores de características pequeños. Sin embargo, la precisión alcanzada por los métodos sigue siendo baja.

Las técnicas de reducción de dimensiones se pueden dividir en dos categorías: aquellas que intentan preservar la estructura global del espacio original y aquellas que están más enfocadas en mantener la estructura local del espacio. PCA es el método más habitual de la primera categoría; muestra resultados competitivos para ratios de reducción pequeños, pero su eficacia disminuye drásticamente cuando el ratio de reducción es alto. Por ejemplo, en el contexto de la recuperación basada en dibujos, una reducción de 2048 dimensiones a 8 produce una degradación de aproximadamente 400 % en mAP.

En la segunda categoría, t-SNE [12] y UMAP [13] son métodos que muestran resultados superiores. A diferencia de PCA, estos métodos tienen en cuenta la información local de cada punto en el espacio original. Este comportamiento permite que el método extraiga características relevantes para cada punto, lo que mejora la precisión de un método de recuperación de imágenes. Entre esta categoría, UMAP ha demostrado que supera a t-SNE, ya que conserva una mayor parte de la estructura global con un rendimiento superior en tiempo de ejecución. Además, UMAP puede escalar a datasets de tamaños más grandes.

2.4. Métricas

Una métrica de evaluación ampliamente utilizada en el contexto de *Image retrieval* es la de *Mean average precision* (mAP), que busca representar la cercanía de las respuestas correctas a la primera posición, con respecto al total de respuestas. Dado un conjunto de consultas Q, el mAP se calcula como el promedio de los AP (*Average precision*) tal como se muestra en la ecuación 2.1.

$$mAP = \frac{1}{|Q|} \cdot \sum_{\mathrm{i}=1}^{|Q|} AP_{\mathrm{i}} \tag{2.1}$$

A su vez, el AP_{i} se calcula como se muestra en la ecuación 2.2, donde GTP_{i} es la cantidad de imágenes correctas dentro de todo el conjunto para la consulta i, TP_k es la cantidad de imágenes correctas vistas hasta el instante k, y $\mathbb{1}_k$ es una función indicatriz que valdrá 1 si la imagen k es correcta y 0 de lo contrario. Se define como imagen correcta a aquella de la misma clase que la imagen consultada.

$$AP_{\text{i}} = \frac{1}{GTP_{\text{i}}} \cdot \sum_{k=1}^{|M|} \frac{\mathbb{1}_k \cdot TP_k}{k} \tag{2.2}$$

Otra métrica relevante en el contexto de Image Retrieval es el Mean Reciprocal Rank, que permite medir con qué precisión el método es capaz de recuperar la primera imagen relevante. Esta métrica es importante en aplicaciones reales como el Ecommerce ya que una persona sólo revisará las primeras imágenes recuperadas por un sistema, y el éxito de cualquier aplicación dependerá de que se encuentren imágenes relevantes en las primeras posiciones. En la ecuación 2.3 se muestra cómo se calcula el MRR.

$$MRR = \frac{1}{|Q|} \cdot \sum_{\text{i}=1}^{|Q|} \frac{1}{FTP_{\text{i}}} \tag{2.3}$$

Donde FTP_{i} corresponde a la posición del primer resultado relevante para la consulta i.

Otra métrica útil es la curva de precisión-recall, la que muestra cual es la mejor precisión alcanzada a partir de determinados valores de recall. La curva representa un gráfico monótonamente decreciente que permite ver qué tan bien se comporta el método para distintos valores de recall.

Capítulo 3

Desarrollo

3.1. Datasets

Para el entrenamiento y evaluación se usarán los datasets Flickr15K, Flickr25K, Sketchy, ImageNet y un dataset confeccionado en este trabajo, al cual se llamará eCommerce. En las siguientes secciones se explica en detalle cada uno de los datasets utilizados.

3.1.1. ImageNet

ImageNet es una gran base de datos de imágenes con más de 100.000 categorías y cerca de 500 fotos por categoría. El dataset que se utilizará está conformado por cerca de un millón de imágenes de ImageNet divididas en 1000 clases distintas. Este dataset se utilizará para las primeras etapas del entrenamiento.

3.1.2. Flickr25K

Flickr25K es un dataset que cuenta con 250 clases y 100 imágenes y 80 dibujos para cada clase. Estas clases se dividen en elementos de la vida cotidiana, como pueden ser vehículos, comida, animales, electrodomésticos, etc. En la figura 3.1 se presentan ejemplos de las imágenes y dibujos que contiene este dataset. Este dataset se utilizará para las primeras etapas del entrenamiento.

3.1.3. Sketchy

Sketchy cuenta con 12.500 imágenes repartidas entre 125 clases y para cada imagen posee alrededor de 5 dibujos hechos en base a esta, sumando 75.460 dibujos. El dataset consiste principalmente en elementos de la vida cotidiana y animales. La figura 3.2 presenta ejemplos de las imágenes y dibujos que contiene este dataset. Este dataset se utilizará para las últimas etapas del entrenamiento.

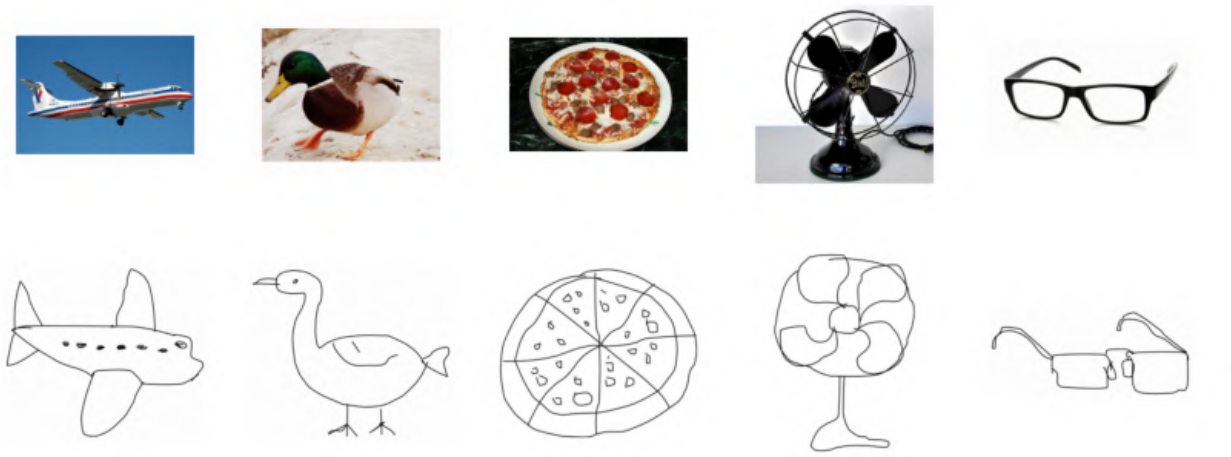

Figura 3.1: Muestra de imágenes del dataset Flickr25K.

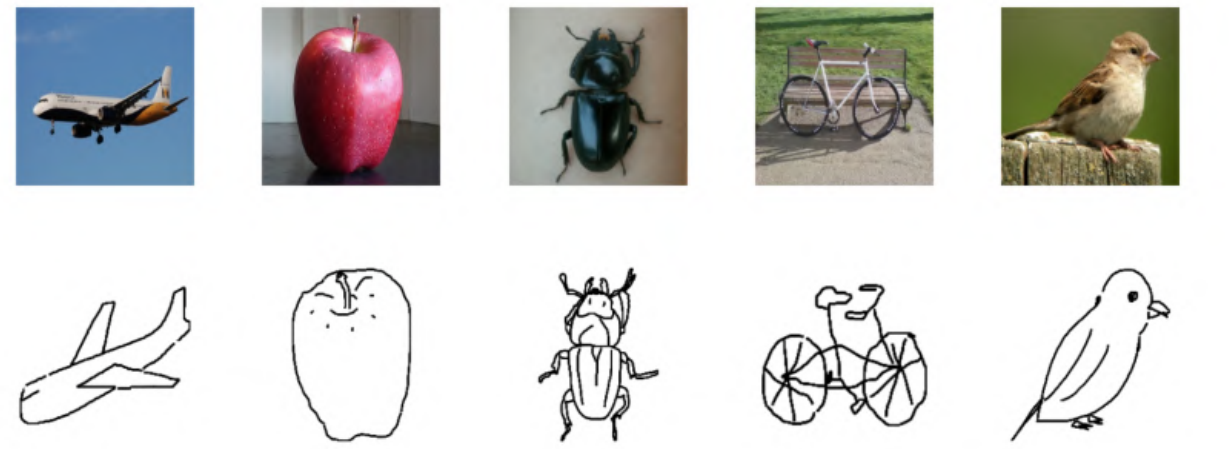

Figura 3.2: Muestra de imágenes del dataset Sketchy.

3.1.4. Flickr15K

Flickr15K es un dataset con 14.660 imágenes divididas en 33 clases, además, cuenta con 330 dibujos (10 por cada clase) que se pueden utilizar para hacer consultas en el dataset. Las clases se dividen principalmente en sitios historicos, animales, y algunos paisajes. En la figura 3.3 se presentan ejemplos de las imágenes y dibujos que contiene este dataset. Este dataset será utilizado para evaluar los resultados obtenidos.

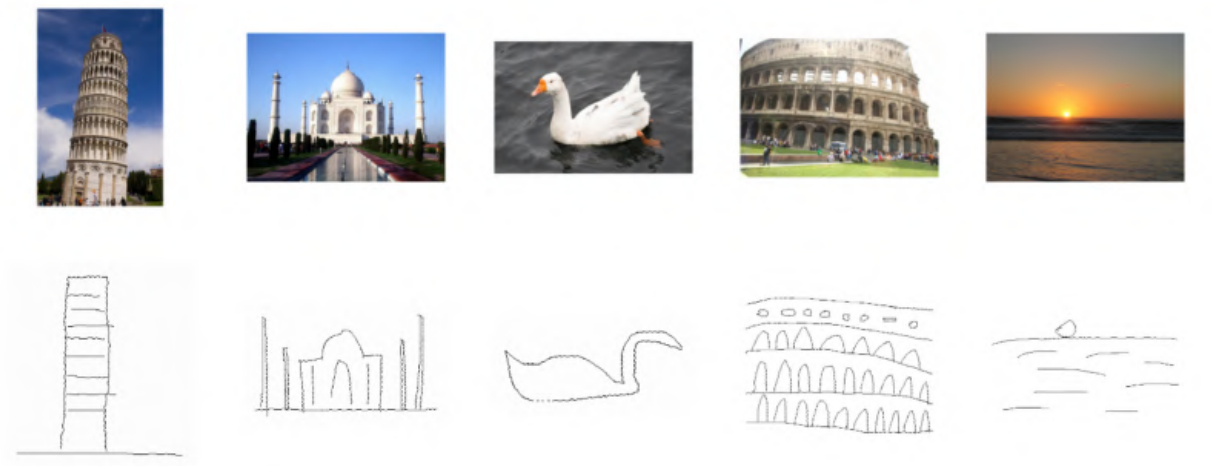

Figura 3.3: Muestra de imágenes del dataset Flickr15K.

3.1.5. eCommerce

Se confeccionó este dataset para comprender de mejor manera el comportamiento de los métodos SBIR en aplicaciones reales como el comercio electrónico. Se eligió este contexto porque representa una de las aplicaciones más modernas de recuperación basada en dibujos con un uso masivo.

Este dataset está compuesto por un catálogo con 10.600 fotos relacionadas con productos que se pueden encontrar en el comercio electrónico. Estas imágenes se distribuyen en 133 clases según su forma. El dataset tiene una variedad de productos como electrodomésticos, ropa, juguetes, etc. La figura 3.4 muestra un conjunto de fotos que se pueden encontrar en este dataset. Adicionalmente se recolectaron 665 dibujos para consultas, 5 por cada clase. La obtención de los dibujos se basó en imágenes del dataset. La figura 3.5 muestra una serie de consultas que se pueden encontrar en el conjunto de datos de comercio electrónico.

3.2. Modelo base

Basándose en el trabajo de Fuentes [4] se decidió usar una arquitectura SE-ResNet-50 para el modelo base (baseline) de este estudio.

La red se implementó en Python 3 usando la librería Tensorflow 2 y principalmente el módulo Keras. La implementación se dividió en distintas componentes (bloques convolucionales, bloques residuales, capas SE) con el fin de modularizar la red. Esto ayuda a que el código sea más legible y facilitó considerablemente el manejo de los pesos durante el entrenamiento.

Una vez terminada la implementación se probó que la red funcionara correctamente entrenándola para una tarea de clasificación. Para esto se utilizó un dataset reducido de dibujos donde la red mostró buenos resultados.

Figura 3.4: Ejemplos de imágenes del catálogo de eCommerce.

Figura 3.5: Ejemplos de consultas del catálogo de eCommerce.

El entrenamiento de la arquitectura se basó en el entrenamiento en múltiples etapas propuesto por Bui et al. [2], el cual consiste en cuatro etapas enfocadas a entrenar distintos aspectos de la red,

sin embargo, se decidió hacer una modificación. Tanto la segunda como la tercera etapa entrenan con el mismo dataset, con la única diferencia de que en la segunda etapa se usa una red siamesa y en la tercera una red triple. Sin embargo, en la literatura se ha observado que las redes triples logran el mismo objetivo que redes siamesas e incluso logran mejores resultados, debido a esto, se consideró que la segunda etapa era redundante y se omitió. También se realizó un entrenamiento previo de la red con el dataset de ImageNet con el fin de tener una buena base para el resto de las etapas. En la figura 3.6 se muestran las etapas de entrenamiento que se efectuaron, en amarillo se muestran los pesos independientes de la rama de dibujos, en rojo los pesos independientes de la rama de imágenes, en verde los pesos compartidos por ambas ramas y en azul las funciones de pérdida utilizadas. A continuación, se explica en detalle cada etapa.

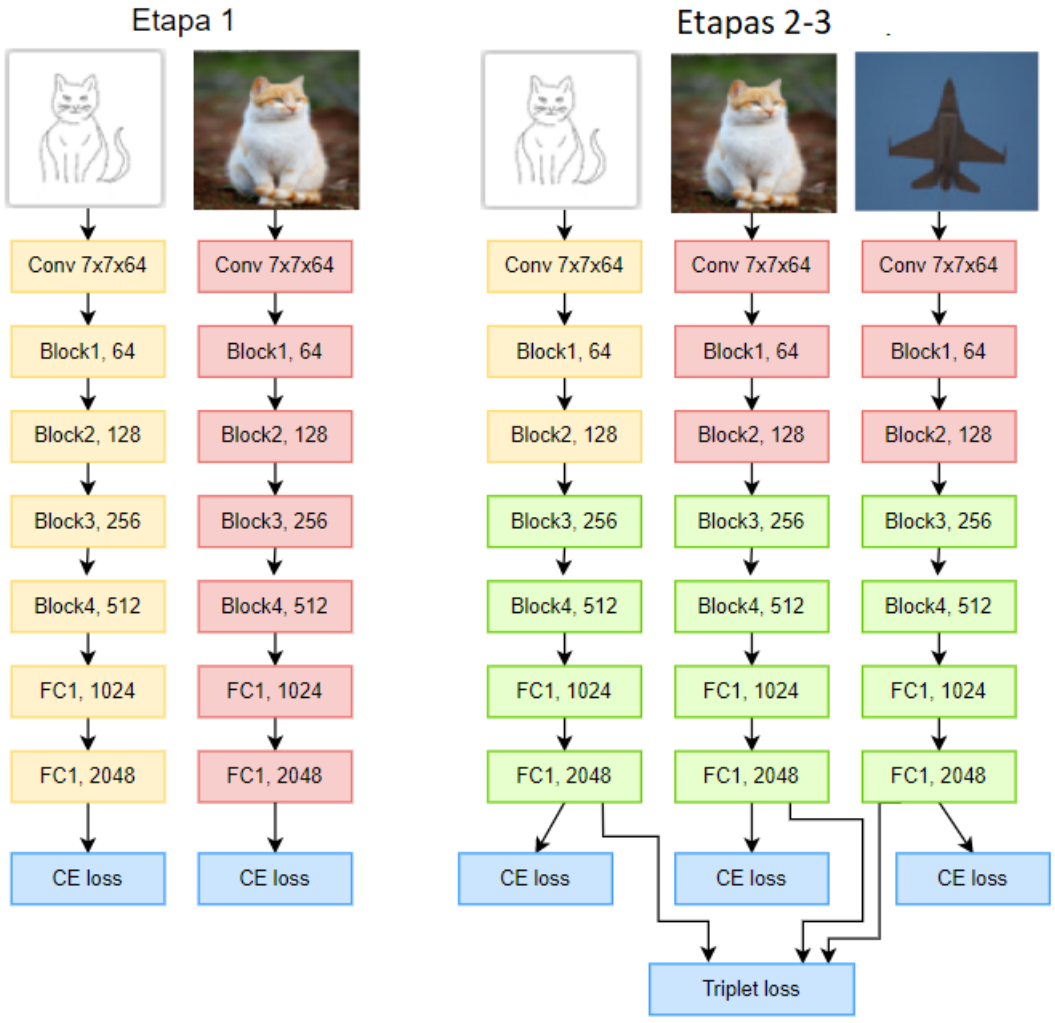

Figura 3.6: Etapas de entrenamiento.

3.2.1. Etapa 0: Entrenamiento preliminar

Esta etapa busca entrenar la red para una tarea de clasificación general con el fin de que aprenda una representación de las imágenes. Luego esta red puede ser reentrenada para una tarea de clasificación específica partiendo de una buena base. Para esto se usa ImageNet debido a su dificultad y la gran cantidad de ejemplos que posee (alrededor de 1000 por clase).

Debido al gran tamaño de este dataset no era posible cargar todo en memoria, por lo que se implementó un generador que fuera leyendo las imágenes a medida que fueran necesarias. Esto volvió el entrenamiento más lento, pero permitió aprovechar el dataset completo.

3.2.2. Etapa 1: Redes Separadas

Esta etapa consiste en entrenar una rama de la arquitectura con un dataset de dibujos y otra rama con un dataset de imágenes para la tarea de clasificación con el fin de entrenar los primeros pesos de la red (los pesos no compartidos) de forma independiente. Para esta etapa se utilizó el dataset Flickr25K.

El entrenamiento se realizó en dos etapas. Primero se entrenó una red con el dataset de dibujos y luego otra con el dataset de imágenes. En ambos casos se utilizaron los pesos entrenados en la etapa anterior y se cargaron los dataset en memoria, agilizando el entrenamiento.

3.2.3. Etapa 2: Redes Siamesas

Consiste en entrenar ambas ramas simultáneamente, utilizando los pesos de la etapa anterior para las primeras capas de cada red (los pesos no compartidos) y los pesos de la etapa 0 para las capas siguientes (pesos compartidos). Para esto se crea una rama de la arquitectura para dibujos y dos ramas para imágenes. Luego se les entrega una tripleta conformada por un dibujo, una imagen de la misma clase y una de una clase distinta y finalmente, se calcula una función de pérdida en base a las cross entropy loss de cada rama y la triplet loss de los embeddings como se muestra en la ecuación 3.1

$$\mathcal{L} = \alpha(CE_q + CE_p + CE_n) + (1 - \alpha)triplet\_loss \tag{3.1}$$

3.2.4. Etapa 3: Fine Tuning

Esta última etapa busca afinar los resultados obtenidos en la etapa anterior entrenando las redes con un dataset de grano fino como lo es Sketchy. Dado que cada dibujo de Sketchy cuenta con una imagen asociada en la cual está basado, esta se utilizará como par positivo. Para el par negativo se escogerá otra imagen de la misma clase o una imagen de una clase completamente distinta con $0{,}5$ de probabilidad cada una. Esta etapa permite obtener mejores resultados ya que es posible entrenar la red para casos más difíciles, como lo es comparar un dibujo con dos imágenes de la misma clase, pero con una más similar que la otra.

3.3. Estrategias propuestas

3.3.1. PCA

El análisis de componentes principales o PCA (por sus siglas en inglés) busca escoger un sistema de coordenadas donde las componentes estén ordenadas según la covarianza que presentan. De esta forma, se pueden eliminar las últimas dimensiones sin afectar mayormente el desempeño del sistema. Este es uno de los métodos más populares de reducción de dimensionalidad ya que permite reducir la cantidad de dimensiones de manera considerable sin afectar mayormente la cantidad de

información. En la figura 3.7 se puede observar el sistema de coordenadas obtenido producto de aplicar PCA a un conjunto de puntos en un espacio bidimensional.

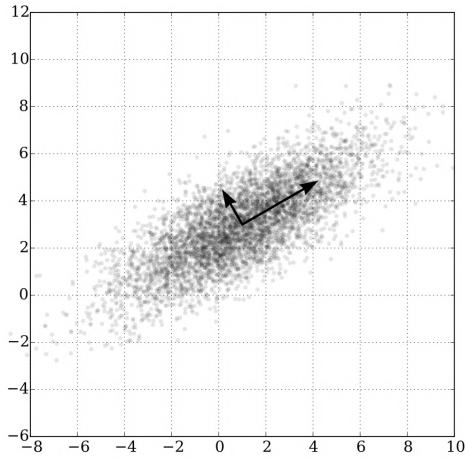

Figura 3.7: Ejemplo PCA.

Para esta estrategia se utilizó PCA sobre las representaciones generadas por la arquitectura base, con el fin de obtener representaciones de menor dimensión. En la figura 3.8 se puede observar la arquitectura final utilizando esta estrategia, donde D corresponde a la dimensión del embedding reducido.

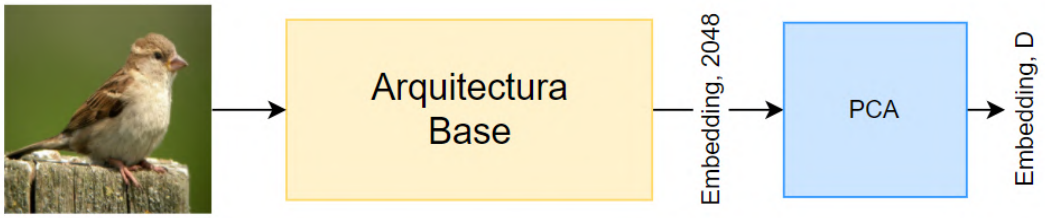

Figura 3.8: Arquitectura utilizando PCA.

3.3.2. Capa de Reducción Simple (CRS)

Esta estrategia consiste en agregar una capa de menor tamaño al final de la red, con el fin de obtener vectores de características a partir de esta. La red junto a esta nueva capa serán reentrenadas con el fin de aprender representaciones con un alto valor semántico e información suficiente que permita diferenciar imágenes de distintas clases.

Para la implementación de la reducción simple se extendió la implementación de la arquitectura base para que permitiera la adición de un bloque de reducción al final de la red. Esta modificación también será utilizada posteriormente por los métodos de reducción binaria. Para el bloque de reducción se utilizó una capa fully-connected con una cantidad de neuronas igual al nivel de reducción deseado. Esta capa se conecta a la capa de clasificación y las salidas de la red son los

embeddings otorgados por la capa de reducción y las clasificaciones realizadas por la capa siguiente. En la figura 3.9 se puede observar la arquitectura final utilizando esta estrategia, donde D corresponde a la dimensión del embedding reducido.

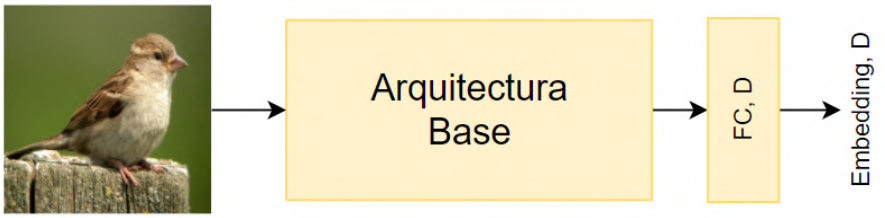

Figura 3.9: Arquitectura utilizando CRS.

3.3.3. Uniform Manifold Approximation and Projection (UMAP)

UMAP corresponde a una técnica no supervisada para la reducción de dimensiones, que aprovecha la estructura local de los puntos en el espacio original para aprender una representación compacta.

En términos generales lo que hace UMAP es armar un grafo en el espacio original de alta dimensión conectando cada elemento con los K vecinos más cercanos, para luego intentar ajustar ese grafo a un espacio de menor dimensión intentando preservar las distancias del espacio original. Para esto UMAP considera dos parámetros, la cantidad de vecinos a conectar durante la creación del grafo (K) y la distancia mínima (min_dist) que los puntos puedan tener en la representación de baja dimensión. El valor de K influye en qué tanto peso le atribuye el algoritmo a la estructura local vs la global. Altos valores de K intentarán preservar la estructura global mientras que bajos valores priorizarán la estructura local. El parametro min_dist controla qué tan cerca puede agrupar UMAP los puntos en el espacio. Valores más bajos permitirán que se formen concentraciones de puntos, mientras que valores altos intentarán esparcir los puntos a través del espacio. En la figura 3.10 se presenta un ejemplo de cómo UMAP representa un mammut de tres dimensiones en un espacio de dos dimensiones. Para dicha visualización se utilizaron los valores $K = 15$ y min_dist$= 0{,}1$

Para los experimentos se utilizará $K = 15$ y min _dist $= 0{,}1$. En la figura 3.11 se puede observar la arquitectura final utilizando esta estrategia, donde D corresponde a la dimensión del embedding reducido.

3.3.4. Capa binaria Sigmoidal (CBS)

Sigmoid es una función de activación popular descrita por la fórmula:

$$\sigma(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{3.2}$$

Como se puede observar en la figura 3.12 el efecto de esta función es acercar los valores negativos a 0 y los positivos a 1, lo cual resulta extremadamente útil si lo que se desea es entrenar una red *end to end* para que entregue valores binarios ya que es una función diferenciable.

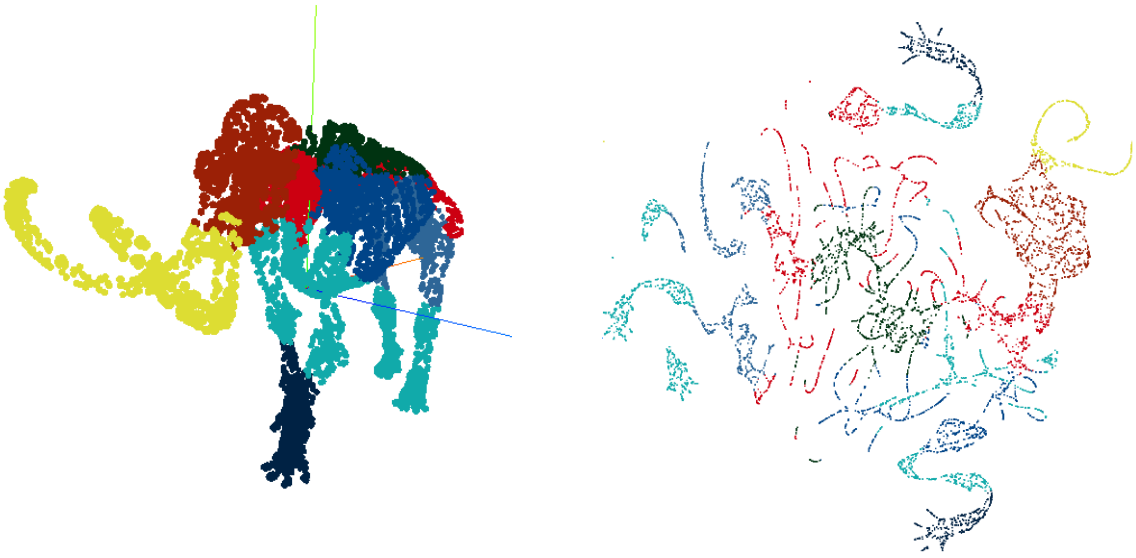

Figura 3.10: Ejemplo de UMAP

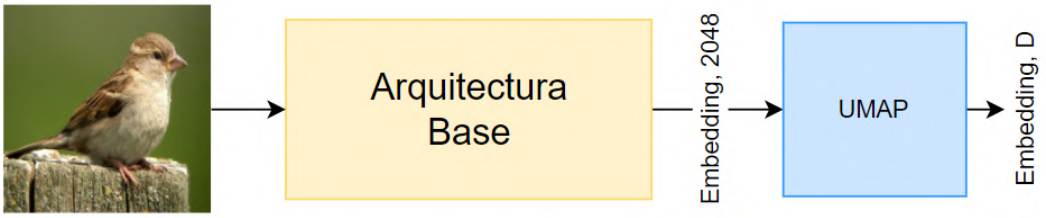

Figura 3.11: Arquitectura utilizando UMAP.

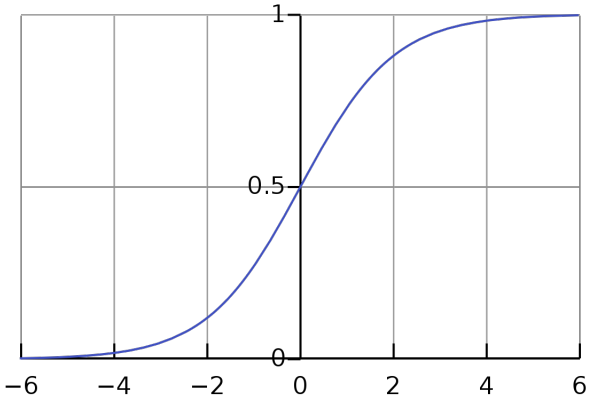

Figura 3.12: Curva de función sigmoid

Esta estrategia consiste en agregar una capa con función de activación sigmoidal al final de la red con el objetivo de obtener representaciones cercanas a valores binarios. Luego bastara con considerar todos los valores superiores a $0{,}5$ como 1s y los inferiores como 0s. La ventaja de este método es su fácil implementación, pero ha sido criticado en la literatura [10] ya que tiene la fama de sufrir de desvanecimiento de gradiente, ralentizando el entrenamiento. Se cree que este último efecto es irrelevante en nuestro caso ya solo se usará la función al final de la red. En la figura 3.13 se

puede observar la arquitectura final utilizando esta estrategia, donde D corresponde a la dimensión del embedding reducido.

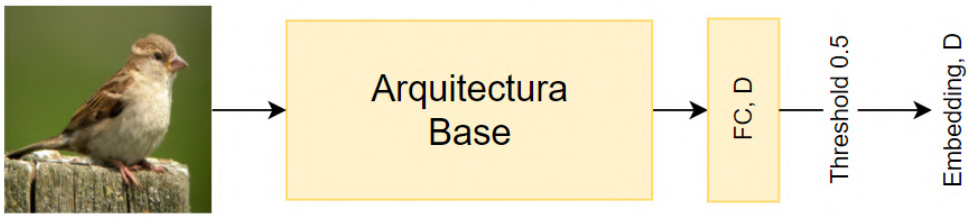

Figura 3.13: Arquitectura utilizando CBS.

3.3.5. DSH

DSH es una estrategia que busca entrenar una red que entregue salidas binarias, sin el uso de funciones de activación. Para esto se utiliza una función de pérdida que fuerza la binarización de la salida a la vez que busca un alto valor semántico en los embeddings obtenidos.

En el artículo original [10], DSH usa una red pequeña para aprender representaciones binarias. Se cree que, al utilizar una red más profunda como la arquitectura base de este trabajo, la precisión de recuperación debe mantenerse o incluso mejorar. Por lo tanto, se incorporó una capa fully-connected al final del backbone de tamaño igual a la dimensión objetivo. Finalmente, adaptó la pérdida de DSH definida en la ecuación 2.3 para trabajar con tripletas de la siguiente manera:

$$\begin{aligned}\mathcal{L}_{\mathcal{DSH}}(b_q, b_p, b_n) = \\ &\frac{1}{2}\,\text{máx}(0, m - ||b_q - b_n||_2^2 + ||b_q - b_p||_2^2) \\ &+ \lambda(||\,|b_q| - \mathbf{1}||_1 \\ &+ ||\,|b_p| - \mathbf{1}||_1 + ||\,|b_n| - \mathbf{1}||_1)\end{aligned}$$

donde b_q, b_p y b_n son los embeddings producidos por la red, donde el primero es el dibujo consultado y los otros son los embeddings de las imágenes positivas y negativas, respectivamente.

Además, se agregó una pérdida de cross-entropy para aprovechar las etiquetas de la imagen. Para ello, se agregó una capa de clasificación después de la capa de reducción. Para combinar la pérdida de DSH adaptada ($\mathcal{L}_{DSH}$) y la pérdida de cross-entropy ($\mathcal{L}_{CE}$), se utilizó un parámetro de ponderación $\alpha = 0{,}4$ y se dividió $\mathcal{L}_{DSH}$ por la dimensión objetivo D_o, ya que esta crece proporcional al tamaño de la última capa. La pérdida final se define de la siguiente manera:

$$\mathcal{L} = \alpha\frac{\mathcal{L}_{\mathcal{DSH}}}{D_o} + (1-\alpha)\mathcal{L}_{\mathcal{CE}}$$

En la figura 3.14 se observa la arquitectura final utilizando esta estrategia, donde D corresponde a la dimensión del embedding reducido. En este caso se aplica un threshold en 0 ya que la red entregará valores entre -1 y 1.

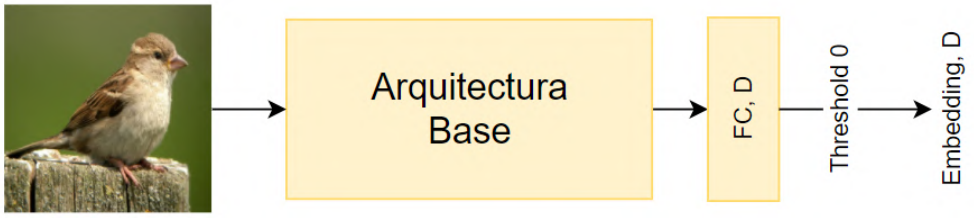

Figura 3.14: Arquitectura utilizando DSH.

3.3.6. DTSH

Similar al método anterior, DTSH busca entrenar una red que entregue representaciones binarias mediante el uso de una función de pérdida que fuerce la binarización de la salida.

Para esto se agrega una capa fully-connected al final de la arquitectura base para obtener la salida reducida. Como DTSH se propuso originalmente para trabajar con redes triples, no se necesita modificar la función de pérdida para usarla en la arquitectura. Sin embargo, de manera análoga al enfoque de DSH, la función de pérdida original se combina con la pérdida de entropía cruzada. Para ello, se agregó una capa de clasificación al final de la red produciendo una pérdida final como suma ponderada de ambas pérdidas. Para ponderar las pérdidas se usa un $\alpha = 0{,}02$ y se divide la $\mathcal{L}_{DTSH}$ por la dimensión objetivo D_o, resultando la siguiente función de pérdida:

$$\mathcal{L} = \alpha \frac{\mathcal{L}_{\mathcal{DTSH}}}{D_o} + (1 - \alpha)\mathcal{L}_{\mathcal{CE}}$$

En este caso se utilizó un valor mucho más bajo de α ya que $\mathcal{L}_{DTSH}$ resultó tener una magnitud mucho mayor que $\mathcal{L}_{\mathcal{CE}}$ incluso después de dividirla por D_o.

La arquitectura final de esta estrategia será idéntica a la indicada en la figura 3.14, debido a que ambas son estrategias binarias que entregarán valores entre -1 y 1.

3.4. Entrenamiento

Para entrenar los métodos de reducción basados en redes neuronales (RL, BSL, DSH, DTSH), se amplió la arquitectura base con las modificaciones discutidas anteriormente, cargando los pesos correspondientes y entrenándolos durante 10 epochs usando el conjunto de datos Sketchy [20]. Para la creación de pares se utilizó la misma estrategia mencionada en la etapa 3 del entrenamiento de la arquitectura base. El entrenamiento se llevó a cabo para cada dimensión objetivo.

Al utilizar las técnicas de reducción de dimensión (PCA y UMAP) se llevó a cabo una estrategia diferente. Dado que estos son métodos no supervisados, la transformación que pueden aprender debe inferirse con datos que sigan la misma distribución del conjunto de pruebas. Para este propósito, se aplicó cross-validation sobre los conjuntos de datos de Flickr15K y eCommerce. se dividió el conjunto de imágenes en tres partes, usando dos particiones para entrenar la transformación y la restante para evaluar. Es importante mencionar que sólo se utilizaron las imágenes de los conjuntos de datos para el proceso de entrenamiento, no los dibujos. Las métricas generadas por esta técnica se calculan como un promedio de las métricas obtenidas para cada partición.

Para las mediciones de tiempo se entrenó la transformación sobre todo el conjunto de imágenes (esto es particularmente importante en el caso de UMAP, ya que mientras más datos se le entreguen en el entrenamiento más pesada se vuelve la transformación) y a su vez se realizaron las consultas sobre todo el conjunto de imágenes, esto con el fin de ser consistentes con las mediciones obtenidas por los otros métodos.

Los métodos de representación de puntos flotantes fueron entrenados para dimensiones objetivo entre 1024 y 4. Para el caso de representaciones binarias, estas fueron entrenadas para dimensiones entre 2048 y 128.

3.5. Evaluación

Para realizar consultas de SBIR es necesario generar un ranking de las imágenes más cercanas a cada consulta. Para esto se utilizó un algoritmo de fuerza bruta, el cual consiste en calcular la distancia entre el embedding de la consulta y los embeddings del catálogo. Luego se ordenan las distancias de menor a mayor para generar un ranking de las imágenes más cercanas. Para el caso de las representaciones de punto flotante se usó la distancia euclidiana, mientras que para las representaciones binarias se utilizo la distancia de Hamming, que consiste en contar la cantidad de bits diferentes entre los dos embeddings a comparar.

Para la comparación entre métodos de representación binaria y métodos de representación de punto flotante se compararon las estrategias cuyas representaciones tuvieran el mismo peso. De esta manera una estrategia de representación binaria con 2048 dimensiones es comparada con las representaciones de punto flotante de 64 dimensiones, ya que las primeras pesan 32 veces menos que las segundas.

Capítulo 4

Resultados y análisis

4.1. Arquitectura base

A continuación, se presentan los resultados obtenidos en las distintas etapas de entrenamiento de la arquitectura base.

4.1.1. Etapa 0

En esta etapa se obtuvo una precisión de 49,87 % en el conjunto de validación de ImageNet. Estos resultados son inferiores a los obtenidos en el estado del arte (la mejor precisión alcanzada con SE-ResNet-50 es de 76,7 %, mientras que la mejor precisión obtenida con cualquier red llega a 90,2 %). Sin embargo, dado que la tarea requiere clasificar cada imagen entre 1000 clases distintas, se considera en este trabajo que un 50 % es un punto de partida aceptable para las siguientes etapas, dado que la red debió aprender una representación lo suficientemente buena para alcanzar esos resultados.

4.1.2. Etapa 1

Los mejores resultados obtenidos fueron una precisión de un 65,87 % para los dibujos y 77,87 % para las imágenes. En las figuras 4.1 y 4.2 se pueden observar ejemplos de la red clasificando dibujos e imágenes respectivamente.

4.1.3. Etapas 2 y 3

En la tabla 4.1 se muestra el mAP obtenido por la arquitectura base en el dataset de Flickr15K y se compara con los resultados obtenidos por Bui et al. [2] y Fuentes [4]. Se puede observar que el mAP obtenido por la arquitectura entrenada es menor al obtenido por trabajos del estado del arte. Esto se puede explicar debido a que los trabajos mencionados anteriormente incorporan múltiples optimizaciones con el fin de mejorar el rendimiento, mientras que para este estudio se optó por no incorporar demasiadas mejoras con el fin de mantener una arquitectura simple y facilitar el estudio de las estrategias posteriores. Si bien el mAP obtenido es inferior al estado del arte, no se estima que esto sea un problema, ya que el objetivo de este trabajo es estudiar el impacto que tendrán los

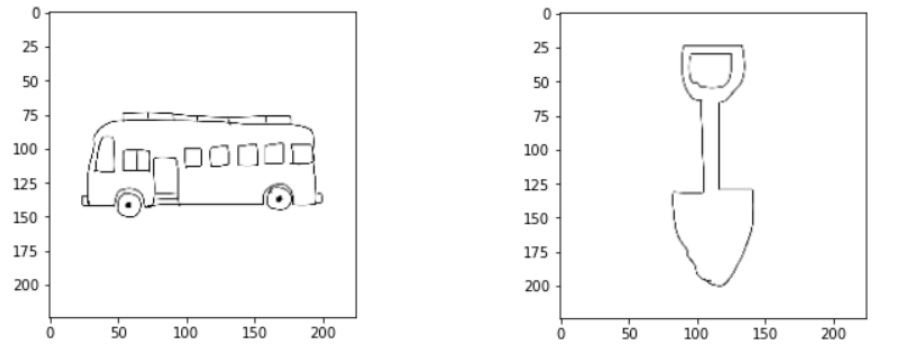

Figura 4.1: Ejemplo de clasificación de dibujos.

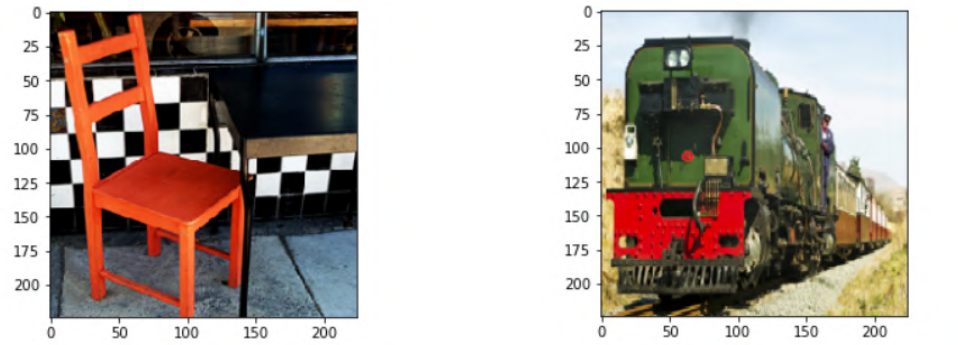

Figura 4.2: Ejemplo de clasificación de imágenes.

métodos de reducción sobre una arquitectura dada y el mAP obtenido es lo suficientemente bueno como para trabajar sobre él.

	Bui	**Fuentes**	**Implementación propia**
Etapa 2	0.35	0.52	0.39
Etapa 3	0.41	0.55	0.42
Etapa 4	0.51	-	-

Tabla 4.1: mAP alcanzado por la arquitectura base y trabajos del estado del arte.

En la tabla 4.2 se presentan los resultados obtenidos por la arquitectura base en los dataset de Flickr15K y eCommerce en términos de mAP y MRR. Se puede observar que, contrario a lo esperado, la tercera etapa de entrenamiento empeoró los resultados obtenidos en el dataset de eCommerce, se estima que esto se puede deber a la diferencia en la cantidad de clases de los datasets. Flickr15K cuenta con tan solo 33 clases, mientras que eCommerce cuenta con 133. A su vez, Flickr25K cuenta con 250 clases mientras que Sketchy tiene 125. Debido a lo anterior, se cree que los entrenamientos con Flickr25K estarán mejor preparados contra datasets con alta variabilidad

como lo es eCommerce mientras que entrenamientos con Sketchy estarán mejor preparados contra datasets con un enfoque acotado, como lo es Flickr15K.

	Flickr15K		eCommerce	
	mAP	MRR	mAP	MRR
Etapa 2	0.39	0.65	0.19	0.47
Etapa 3	0.42	0.71	0.14	0.40

Tabla 4.2: mAP y MRR alcanzados por la arquitectura base en datasets de eCommerce y Flickr15K.

Se decidió usar los resultados de la etapa 3 como modelo base para los experimentos de ambos datasets. Lo ideal hubiera sido entrenar los métodos de reducción de Flickr15K con Sketchy y los métodos de reducción de eCommerce con Flickr25K, pero por temas de tiempo esto no fue posible.

4.2. mAP

En las tablas 4.3 y 4.4 se muestra el mAP alcanzado por los distintos métodos de reducción para múltiples dimensiones objetivo en los dataset Flickr15K y eCommerce respectivamente. Además, en las figuras 4.3 y 4.4 se muestran gráficos con la misma información de las tablas con el fin de poder comparar los métodos visualmente.

Size	**CRS**	**CBS**	**DTSH**	**DSH**	**PCA**	**UMAP**
1024	0.40	-	-	-	0.39	0.56
512	0.37	-	-	-	0.39	0.54
256	0.38	-	-	-	0.40	0.56
128	0.37	-	-	-	0.40	0.56
64	0.37	0.39	0.19	0.38	0.41	0.55
32	0.32	0.38	0.22	0.34	0.40	0.57
16	0.26	0.38	0.28	0.30	0.37	0.55
8	0.19	0.35	0.31	0.26	0.30	0.55
4	0.18	0.33	0.33	0.25	0.20	**0.57**

Tabla 4.3: mAP alcanzado por los distintos métodos de reducción en el dataset de Flickr15K.

Size	**CRS**	**CBS**	**DTSH**	**DSH**	**PCA**	**UMAP**
1024	0.13	-	-	-	0.15	0.15
512	0.12	-	-	-	0.15	0.16
256	0.12	-	-	-	0.15	0.16
128	0.12	-	-	-	0.15	0.16
64	0.10	0.11	0.05	0.13	0.14	0.16
32	0.09	0.11	0.06	0.12	0.13	0.18
16	0.06	0.09	0.08	0.10	0.09	0.21
8	0.5	0.07	0.10	0.08	0.11	**0.21**
4	0.03	0.05	0.10	0.06	0.06	0.19

Tabla 4.4: mAP alcanzado por los distintos métodos de reducción en el dataset de eCommerce.

Se puede observar que en el caso de Flickr15K, UMAP logra resultados consistentemente superiores al resto de los métodos e incluso mejores que los obtenidos por el modelo base. Esto se

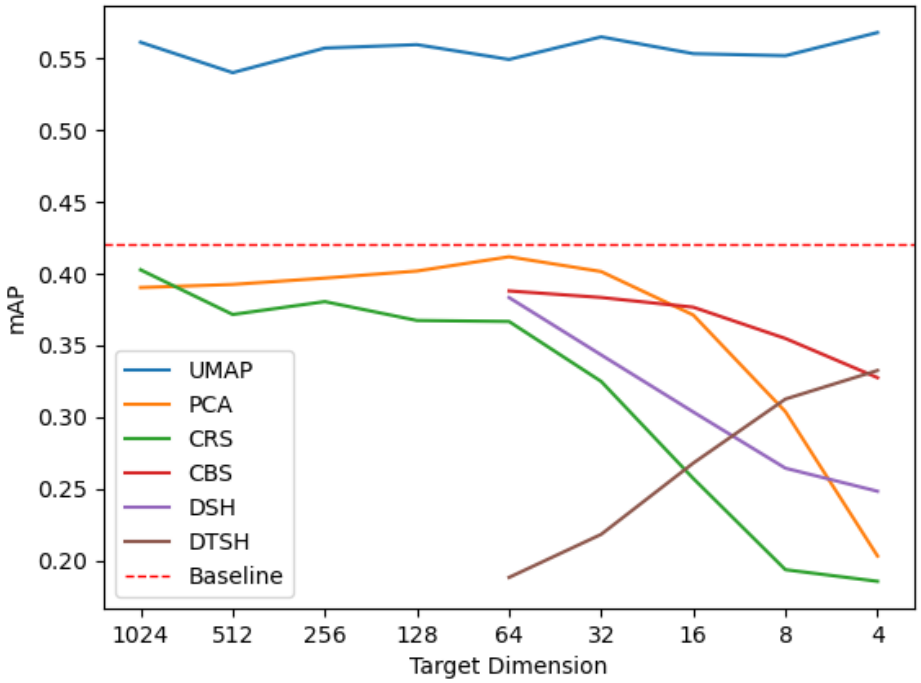

Figura 4.3: Gráfico de mAP para el dataset Flickr15K.

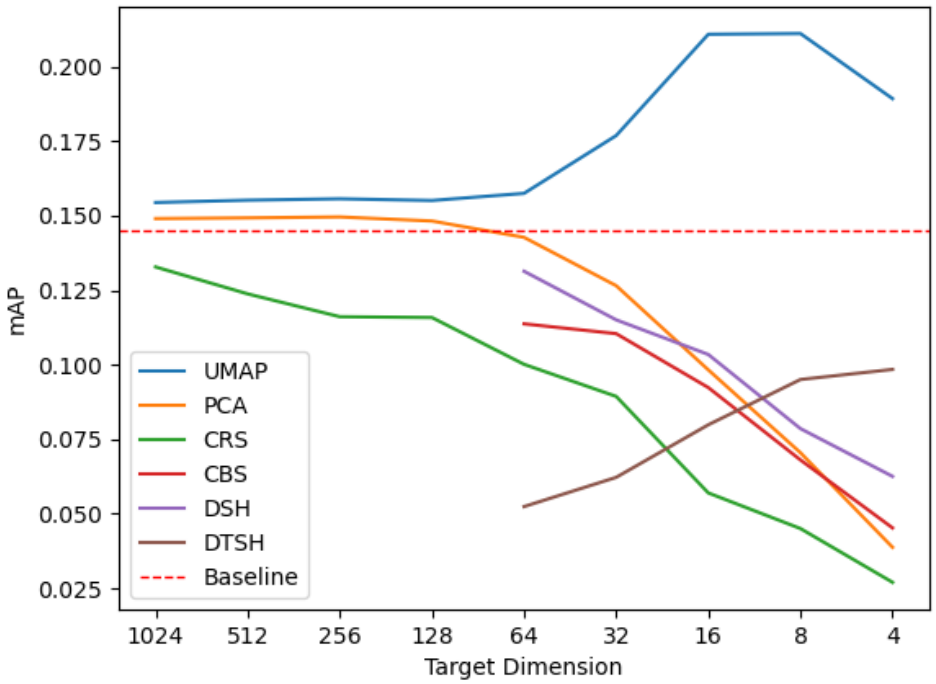

Figura 4.4: Gráfico de mAP para el dataset eCommerce.

explica dado que el algoritmo de UMAP acerca elementos según su vecindad formando clusters de elementos similares. Esta situación ayuda a que la recuperación sea mucho más consistente. También se observa que los resultados de UMAP se mantienen constantes independiente del nivel de reducción. Respecto al resto de los métodos, PCA muestra los mejores resultados para dimensiones objetivo más altas (1024-32), esto se atribuye a que PCA es capaz de reducir los embeddings manteniendo un nivel de información similar, sin embargo, para niveles de reducción extremos decae drásticamente, posiblemente debido a que no es capaz de mantener el mismo nivel de información para dimensiones tan reducidas.

CBS muestra resultados favorables en los niveles de reducción más extremos (16-4). Esto puede deberse a que al ser una estrategia binaria sigue siendo capaz de representar un gran número de características, pese al reducido tamaño de los embeddings, ya que cada bit puede representar una característica distinta. CRS muestra resultados similares a PCA para embeddings de tamaño alto (1024-256), sin embargo decae para dimensiones más bajas. Esto se atribuye a que, al entrenar la red con una cantidad tan baja de características, esta se ve obligada a centrarse en características especificas del dataset (Sketchy), razón por la cual sus embeddings no son generalizables para otros datasets con distintos tipos de imágenes, como lo es Flickr15K. DSH mostró resultados similares a CBS para dimensiones altas pero decae considerablemente para menores dimensiones.

DTSH presentó un comportamiento poco esperado, se puede observar que para mayores dimensiones objetivo presenta peores resultados y en reducciones de 8 ó 4 dimensiones presenta resultados similares a los demás métodos. Se estima que este comportamiento se debe a que no se logró balancear de forma correcta la pérdida de DTSH con la perdida de clasificación durante el entrenamiento. Originalmente la función de pérdida de DTSH fue diseñada para usarse con redes de baja dimensión de salida (12-48 salidas) mientras que en este trabajo se experimentó con redes de hasta 2048 salidas. Debido a esto los valores alcanzados por $\mathcal{L}_{DTSH}$ son considerablemente mayores y dividirlo por la dimensión objetivo demostró ser una solución poco efectiva.

En los resultados del dataset de eCommerce UMAP presenta un desempeño peor para dimensiones objetivo altas (1024-64) y mejora considerablemente para 16 y 8 dimensiones. También se puede observar que en este dataset DSH se comporta levemente mejor que CBS. El resto de los métodos se comportan de forma similar a la observada en el dataset de Flickr 15K. Un aspecto a destacar es que la mayoría de los métodos empeora considerablemente pasadas las 32 dimensiones.

4.3. Recall-Precision

En las figuras 4.5 y 4.6 se muestran los gráficos de Recall-Precision para los datasets de Flickr15K y eCommerce respectivamente.

En el gráfico de Flickr15K se puede observar que la mayoría de los métodos presentan una curva similar a la del modelo base, a excepción de UMAP y DTSH. Se puede observar que UMAP presenta la menor precisión en 0 (sin contar DTSH), pero se mantiene consistente para todos los niveles de recall. Esto sugiere que UMAP tiene mayor dificultad a la hora de entregar resultados relevantes, pero es sumamente consistente una vez lo logra. La curva de DTSH muestra que su desempeño es deficiente en comparación al resto de los métodos.

En el gráfico de eCommerce los métodos muestran mayores diferencias. PCA sigue de cerca a la

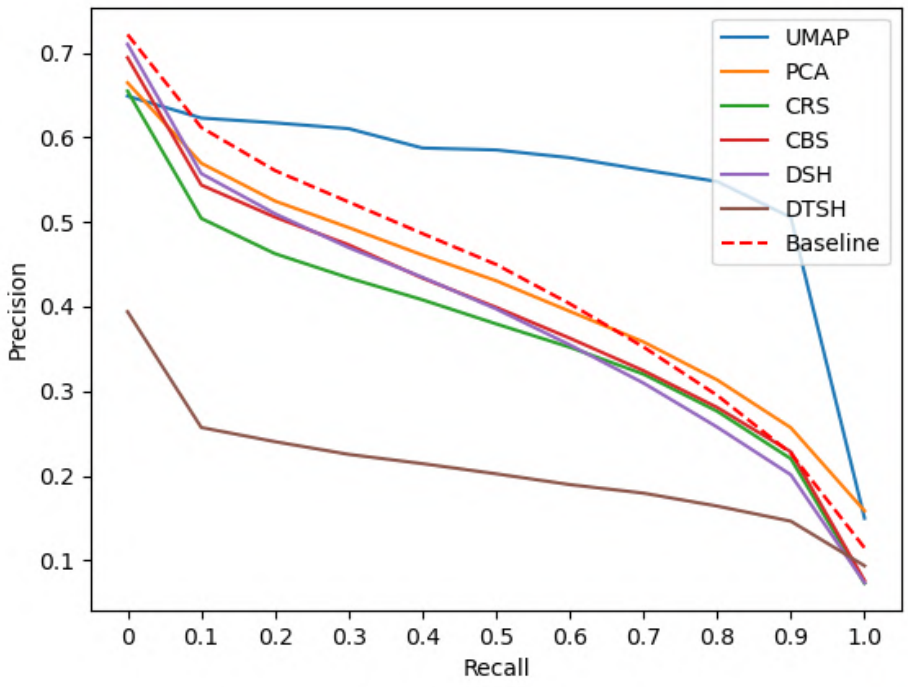

Figura 4.5: Gráfico de Recall-Precision para el dataset de Flickr15K.

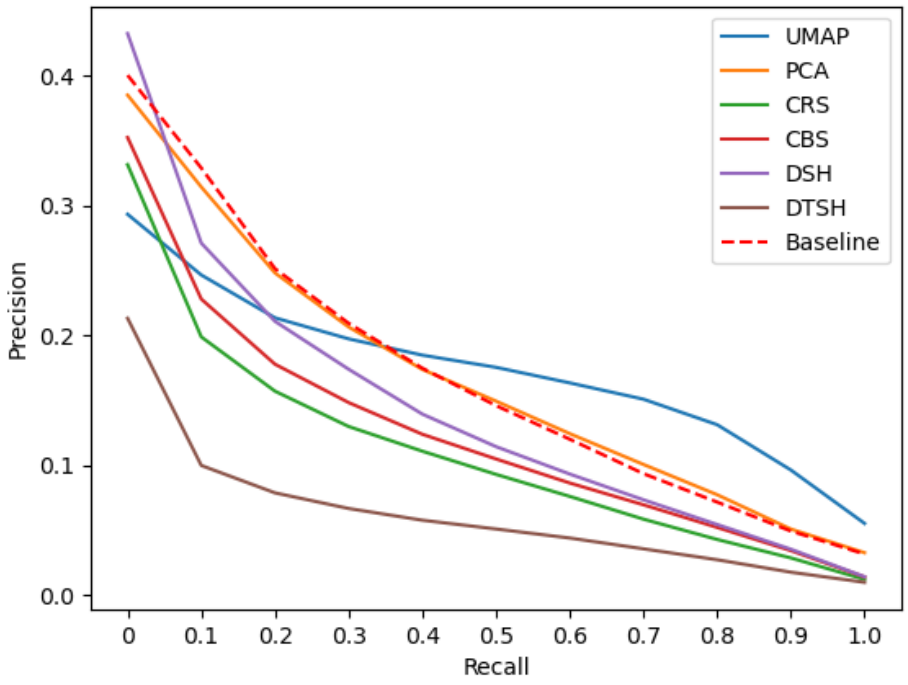

Figura 4.6: Gráfico de Recall-Precision para el dataset de eCommerce.

curva del modelo base, DSH parte con valores superiores al modelo base pero decae rápidamente, lo que sugiere que es efectivo a la hora de recuperar los primeros relevantes pero falla a la hora de ser consistente. UMAP parte mucho más bajo que los demás métodos, pero muestra mejores precisiones para valores altos de recall.

4.4. MRR

En las tablas 4.5 y 4.6 se muestra el MRR alcanzado por los distintos métodos de reducción para múltiples dimensiones objetivo en los dataset Flickr15K y eCommerce respectivamente. Además, en las figuras 4.7 y 4.8 se muestran gráficos con la misma información de las tablas con el fin de poder comparar los métodos visualmente.

	CRS	CBS	DTSH	DSH	PCA	UMAP
1024	0.68	-	-	-	0.62	0.59
512	0.62	-	-	-	0.62	0.60
256	0.63	-	-	-	0.62	0.62
128	0.60	-	-	-	0.61	0.60
64	0.62	0.66	0.34	0.67	0.61	0.60
32	0.56	0.64	0.39	063	0.59	0.61
16	0.45	0.62	0.45	0.60	0.54	0.59
8	0.33	0.59	0.47	0.55	0.45	0.59
4	0.30	0.55	0.45	0.53	0.29	0.60

Tabla 4.5: MRR alcanzado por los distintos métodos de reducción en el dataset de Flickr15K.

	CRS	CBS	DTSH	DSH	PCA	UMAP
1024	0.38	-	-	-	0.39	0.21
512	0.36	-	-	-	0.38	0.21
256	0.36	-	-	-	0.38	0.20
128	0.34	-	-	-	0.38	0.21
64	0.30	0.32	0.19	0.40	0.37	0.21
32	0.27	0.34	0.19	0.37	0.32	0.23
16	0.17	0.30	0.21	0.34	0.26	0.28
8	0.13	0.27	0.23	0.27	0.23	0.29
4	0.09	0.20	0.21	0.24	0.10	0.26

Tabla 4.6: MRR alcanzado por los distintos métodos de reducción en el dataset de eCommerce.

En las mediciones de Flickr 15K se puede observar que UMAP presenta un MRR comparable al resto de los métodos pese a haber obtenido un mAP considerablemente mayor en los experimentos anteriores. Esto se atribuye al hecho que UMAP es capaz de mejorar los casos en que ya funcionaba bien, volviéndolo un método más consistente y mejorando el mAP, pero no es capaz de mejorar los casos en que ya funcionaba mal, razón por la cual no se observaron mejoras en el MRR.

PCA y CRS obtienen resultados similares a UMAP para dimensiones altas, pero decrecen drásticamente bajo las 64 dimensiones. Se puede observar que los métodos binarios DSH y CBS obtienen un MRR cercano al modelo base para 64 dimensiones, aunque son superados por UMAP en los niveles más extremos de reducción. DTSH mostró resultados inferiores al resto de los métodos.

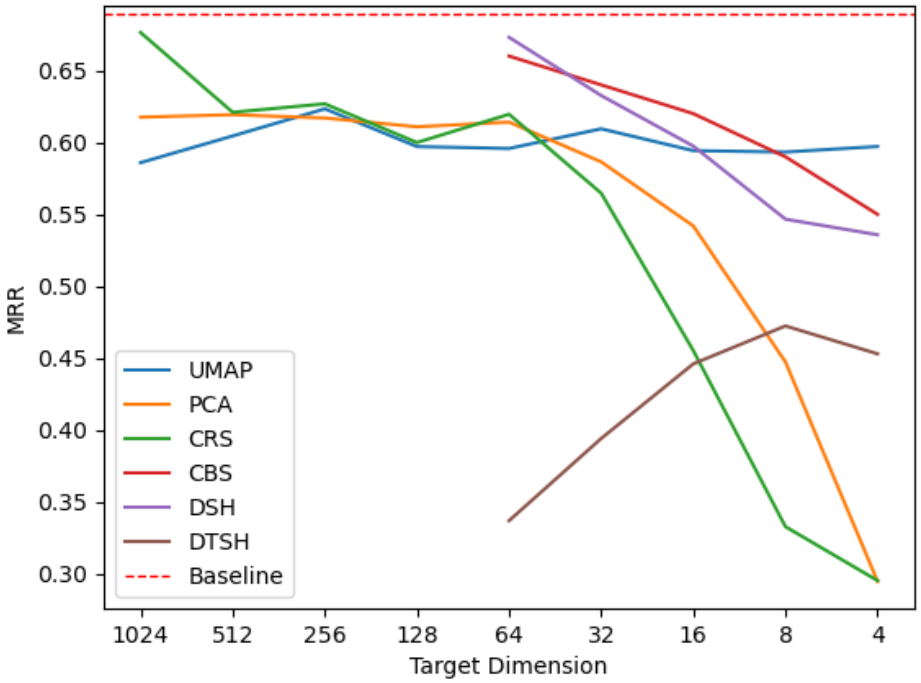

Figura 4.7: Gráfico de MRR para el dataset Flickr15K.

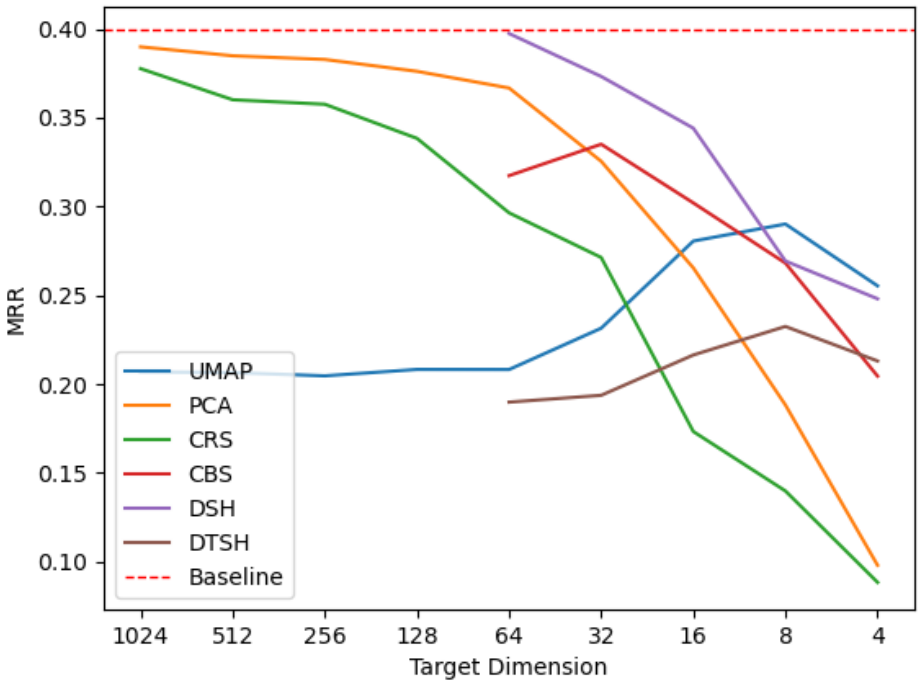

Figura 4.8: Gráfico de MRR para el dataset eCommerce.

En las mediciones de eCommerce se observa una gran caída en el MRR alcanzado por UMAP. Esto se atribuye a que el dataset de eCommerce es mucho más desafiante que Flickr15K lo que dificulta que UMAP aprenda del espacio de características original. Se puede observar que para altas dimensiones PCA obtiene mejores resultados, sin embargo, DSH lo supera para mayores niveles de reducción.

4.5. Tiempo

En la tabla 4.7 se muestran los tiempos de consulta para cada método de reducción y dimensión objetivo en el dataset de Flickr 15K. Para estas mediciones se consideró el tiempo requerido para transformar los embeddings, como es el caso de PCA y UMAP, y el tiempo que toma calcular la distancia entre los embeddings y su ordenamiento. No se consideró el tiempo que toma generar los embeddings con redes neuronales ya que es muy similar para todos los métodos y dimensiones objetivo. Se puede observar que PCA y CRS comparten los mismos tiempos ya que ambos son métodos que utilizan embeddings de punto flotante y el tiempo que toma la transformación de PCA es marginal. A su vez, DSH, DTSH y CBS comparten los mismos tiempos dado que los tres generan embeddings binarios y no requieren de una transformación. El tiempo de consulta del modelo base es de 120 milisegundos. En la figura 4.9 se presenta un gráfico con la información de la tabla con el fin de poder comparar los métodos visualmente.

	CRS	CBS	DTSH	DSH	PCA	UMAP
1024	63.6	-	-	-	63.6	603.6
512	21.4	-	-	-	21.4	291.6
256	14.5	-	-	-	14.5	151.3
128	11.4	-	-	-	11.4	87.5
64	9.4	20.2	20.2	20.2	9.4	52.5
32	8.8	13.6	13.6	13.6	8.8	36.0
16	8.2	10.3	10.3	10.3	8.2	28.5
8	7.8	8.4	8.4	8.4	7.8	24.1
4	6.9	7.4	7.4	7.4	6.9	21.3

Tabla 4.7: Comparación de tiempos de consulta en milisegundos.

Se puede observar que el tiempo de consultas de UMAP es considerablemente mayor al de los demás métodos, incluso superando al modelo base para las dimensiones más altas. Para las dimensiones más bajas se puede observar una mejora considerable con respecto al modelo base, pero sigue tomando alrededor del triple de tiempo que los demás métodos. Se puede observar que las demás estrategias superan al modelo base rápidamente y pasado cierto punto tienden hacia los 6 milisegundos. Esto se explica debido a que ese es el costo de ordenar el arreglo de 15.000 distancias con el fin de rankear las imágenes del dataset. Esa parte del algoritmo se efectuó con funciones nativas de Python y no con Numpy, lo cual resulta ineficiente, por lo que se estima que realizando los experimentos en un lenguaje de más bajo nivel como C++ se debieran obtener tiempos aun menores.

También se puede observar que los métodos con representaciones de punto flotante superan ligeramente a las binarias, esto es contraintuitivo ya que se esperaría que calcular la diferencia de bits entre dos arreglos debiera ser más rápido que la norma euclidiana. Esto se explica debido a que

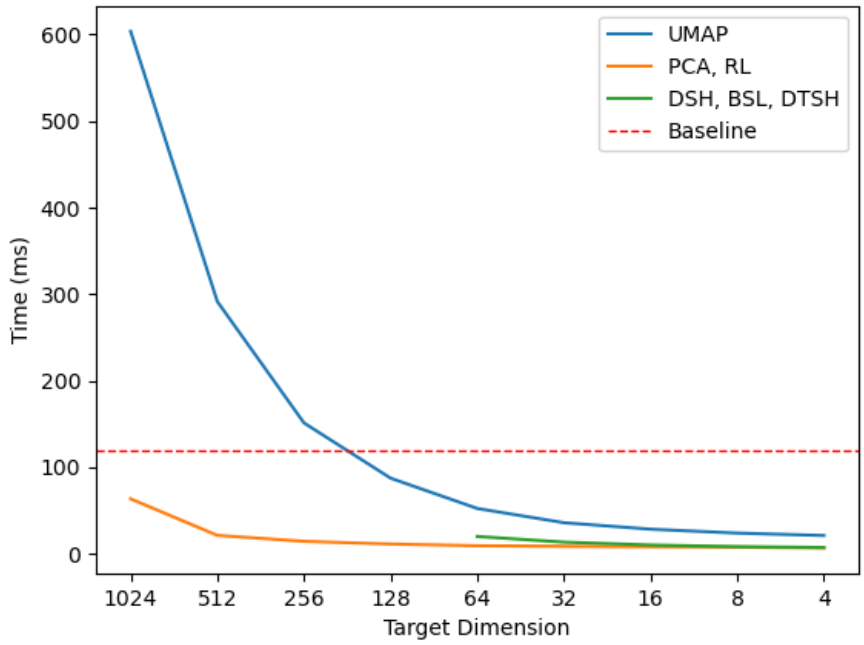

Figura 4.9: Comparación de tiempos de consulta en el dataset Flickr15K.

tanto Python como numpy no cuentan con formas directas de calcular la distancia de hamming entre múltiples arreglos binarios, por lo que fue necesario utilizar múltiples operaciones de numpy para obtener las distancias deseadas versus una única operación necesaria para calcular las distancias entre vectores flotantes. Se cree que en una implementación de bajo nivel estas diferencias de tiempo debieran verse invertidas.

4.6. Memoria

En la tabla 4.8 se muestra la memoria utilizada por los distintos métodos estudiados. Para este cálculo se consideró el peso de los embeddings del conjunto de imágenes y el peso de las transformaciones. No se consideró el peso de la red ya que este es constante e independiente del catálogo en que se use, por lo que una misma red se puede utilizar para calcular los embeddings de múltiples catálogos dentro de un mismo servidor o de catálogos arbitrariamente grandes sin aumentar el uso de memoria del sistema.

Se puede observar que UMAP utiliza una cantidad considerable de memoria para todos los niveles de reducción, utilizando incluso más que el método original. Esto se explica debido a que UMAP durante el entrenamiento guarda una cantidad considerable de puntos como *anchors* y genera una matriz de distancias entre estos con el fin de acelerar los tiempos de búsqueda. PCA usa mucha menor memoria y se acerca bastante al resto de los métodos. Las estrategias que no utilizan transformaciones muestran el mismo nivel de reducción y es proporcional a la cantidad de bytes de las representaciones.

	CRS	CBS	DTSH	DSH	PCA	UMAP
1024	59.4	-	-	-	76.2	569.4
512	29.7	-	-	-	38.1	511.7
256	14.8	-	-	-	19	481.8
128	7.4	-	-	-	9.5	467.4
64	3.7	3.7	3.7	3.7	4.8	459.7
32	1.8	1.8	1.8	1.8	2.3	456.8
16	0.9	0.9	0.9	0.9	1.2	454.9
8	0.4	0.4	0.4	0.4	0.5	453.4
4	0.2	0.2	0.2	0.2	0.3	453.2

Tabla 4.8: Comparación de memoria utilizada por las distintas estrategias.

4.7. Ejemplos de consultas

A continuación se muestran consultas realizadas a través de los distintos métodos estudiados en este trabajo. Las figuras 4.10 y 4.11 muestran consultas realizadas en el dataset Flickr15K, mientras que las figuras 4.12 y 4.13 muestran consultas realizadas en el dataset de eCommerce.

En estas pruebas se utilizaron representaciones de 64 valores para las estrategias de punto flotante y 2048 bits para las representaciones binarias. Se decidió utilizar este valor ya que es donde la mayoría de los métodos mostraron los mejores resultados.

En las figuras se puede observar lo señalado en las secciones anteriores. Se aprecia que UMAP es más consistente que los demás métodos, mejorando incluso los resultados del modelo base. Esto se atribuye a que es un método de reducción que se preocupa de mantener la estructura local, acercando elementos de la misma clase entre sí. La mejora en consistencia se ve reflejada en el alto mAP que UMAP obtiene en los experimentos anteriores.

Se observa que PCA presenta un comportamiento similar al del modelo base, recuperando una cantidad similar de imágenes relevantes y equivocándose en el mismo tipo de imágenes. Lo anterior se explica debido a que PCA no aporta información a las representaciones, si no que más bien se enfoca en perder la menor cantidad de información posible.

Tanto CBS como DSH logran recuperar imágenes relevantes rápidamente, pero fallan en ser consistentes. Esto se ve reflejado en el alto MRR y bajo mAP obtenido en los experimentos anteriores. CRS y DTSH presentan dificultades a la hora de recuperar imágenes relevantes, lo cual es consistente con los bajos valores obtenidos en las métricas anteriores.

En vista de los favorables resultados de UMAP, se decidió estudiar este método en mayor profundidad. Las figuras 4.14 y 4.15 muestran una serie de consultas comparando el modelo base con UMAP en los datasets de Flickr15K y eCommerce respectivamente.

Se observa que UMAP presenta mejoras considerables sobre el modelo base en la mayoría de los casos. Sin embargo, para algunas consultas UMAP falla y recupera imágenes de otra clase de manera consistente, como se puede observar en los casos del Taj Mahal y la flor en el dataset Flickr15K o el babero y la mamadera en el caso de eCommerce. Este comportamiento suele presentarse en aquellos casos en que el modelo base no logra obtener buenos resultados y se le atribuye

Figura 4.10: Comparación de imágenes recuperadas entre los distintos métodos en el dataset de Flickr15K.

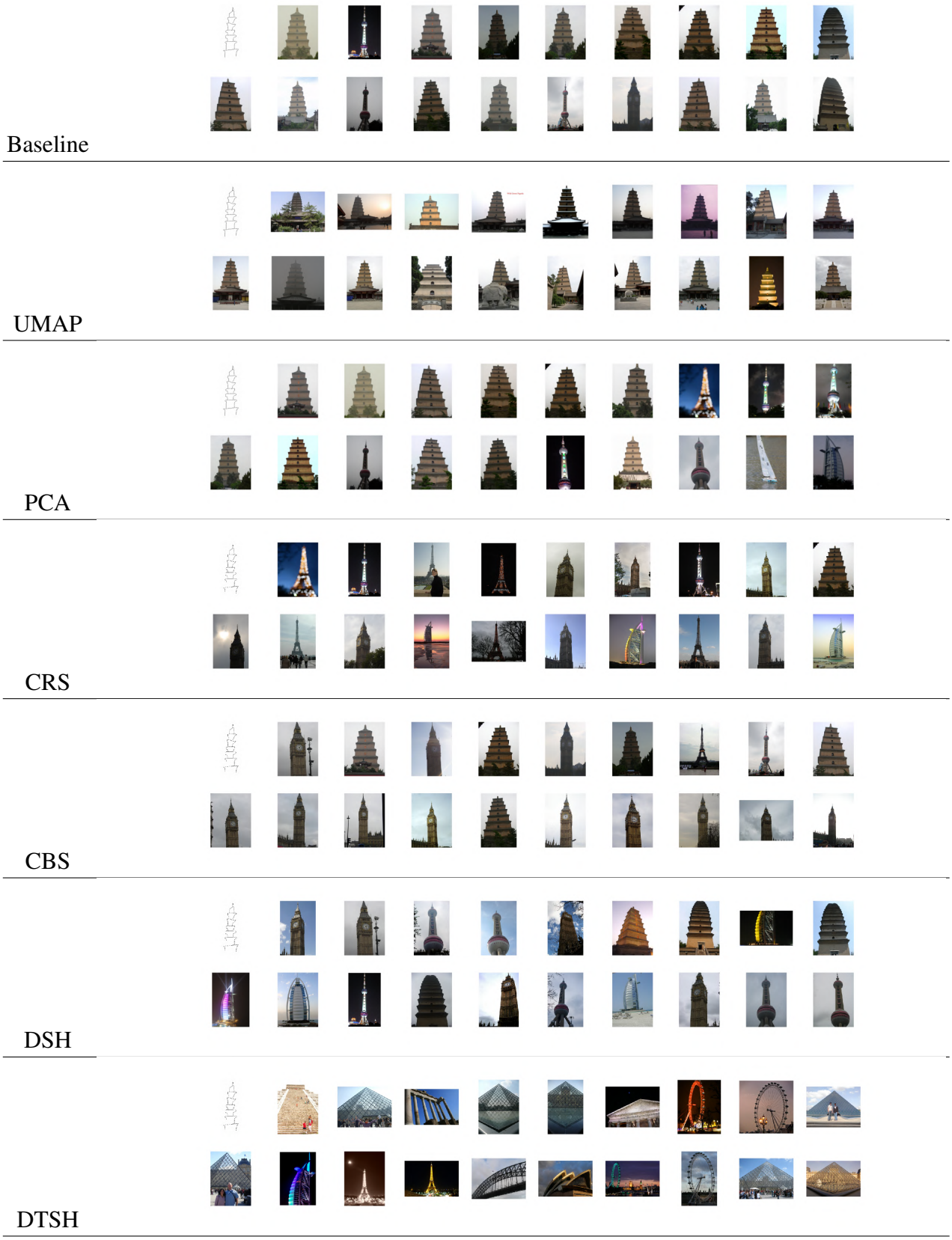

Figura 4.11: Comparación de imágenes recuperadas entre los distintos métodos en el dataset de Flickr15K.

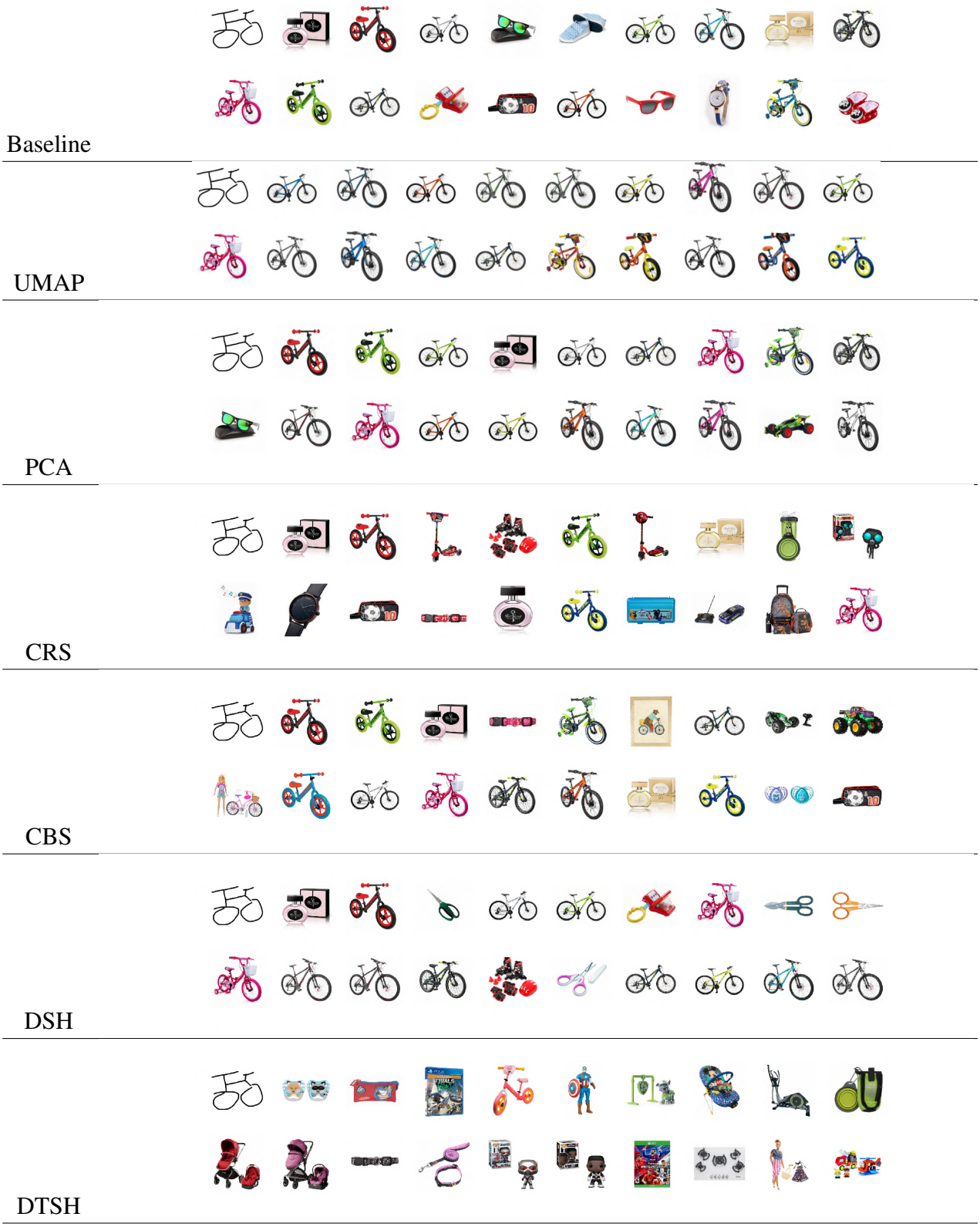

Figura 4.12: Comparación de imágenes recuperadas entre los distintos métodos en el dataset de eCommerce.

Figura 4.13: Comparación de imágenes recuperadas entre los distintos métodos en el dataset de eCommerce.

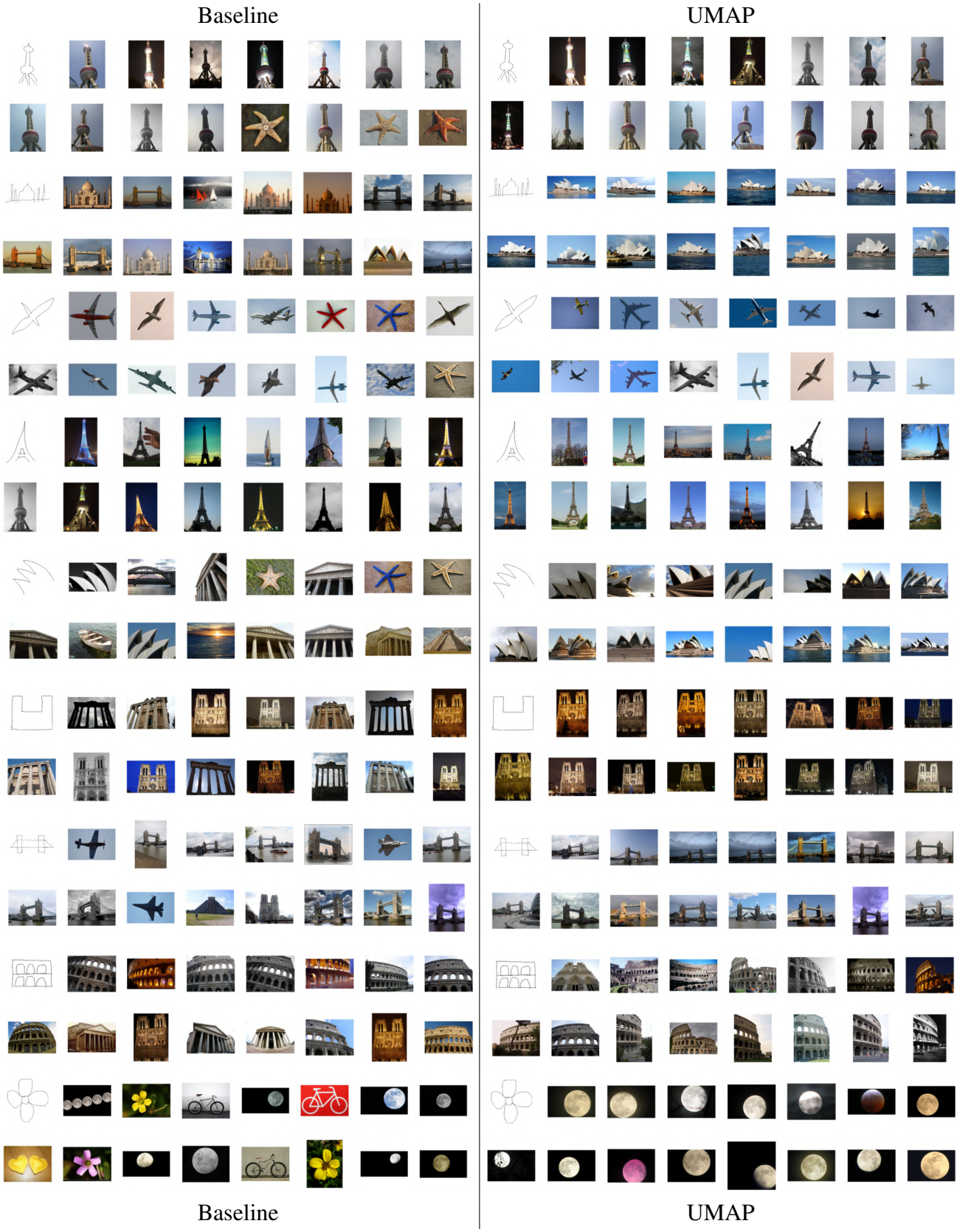

Figura 4.14: Comparación de las imágenes recuperadas entre el modelo base y UMAP en el dataset Flickr 15K.

Figura 4.15: Comparación de las imágenes recuperadas entre el modelo base y UMAP en el dataset de eCommerce.

a la forma en que funciona el método. UMAP intenta preservar la estructura local de los datos, por lo que tiende a acercar elementos similares formando clusters. Debido a esto, tiende a recuperar elementos de una única clase. Esto implica que si la representación del dibujo no es la mejor, se corre el riesgo de que UMAP ubique el dibujo lejos del cluster de la clase original y entregue resultados de una clase completamente distinta. De esta forma, se estima que UMAP logra mejores resultados en los casos en que las representaciones base tienen suficientes imágenes cercanas de la misma clase, pero empeora los resultados cuando las representaciones originales no cumplen esta condición.

4.8. Experimentos adicionales

En vista de los resultados obtenidos por UMAP se decidió estudiar el método en mayor profundidad. A continuación, se describen los experimentos realizados.

4.8.1. Parámetros de UMAP

Con el fin de observar cómo se ve afectado UMAP por los parámetros K y min_dist, se evaluó el método para distintos valores de dichos parámetros. Las figuras 4.16 y 4.17 muestran mapas de calor con el mAP alcanzado en los dataset de Flickr15K y eCommerce respectivamente. El eje x indica el valor de K utilizado y el eje y indica el valor de min_dist.

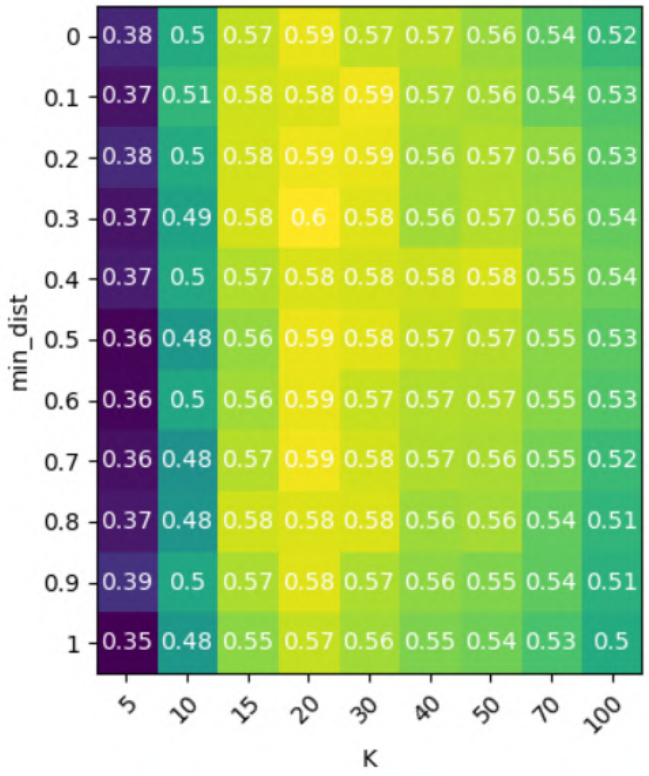

Figura 4.16: mAP alcanzado para distintos conjuntos de entrenamiento en el dataset Flickr15K.

En el caso de Flickr15K se observa que la cantidad de vecinos juega un rol importante en el desempeño del método, alcanzado mejores resultados para valores de K entre 15 y 30. Por otra parte, los distintos valores de min_dist no parecen tener un gran impacto en el desempeño.

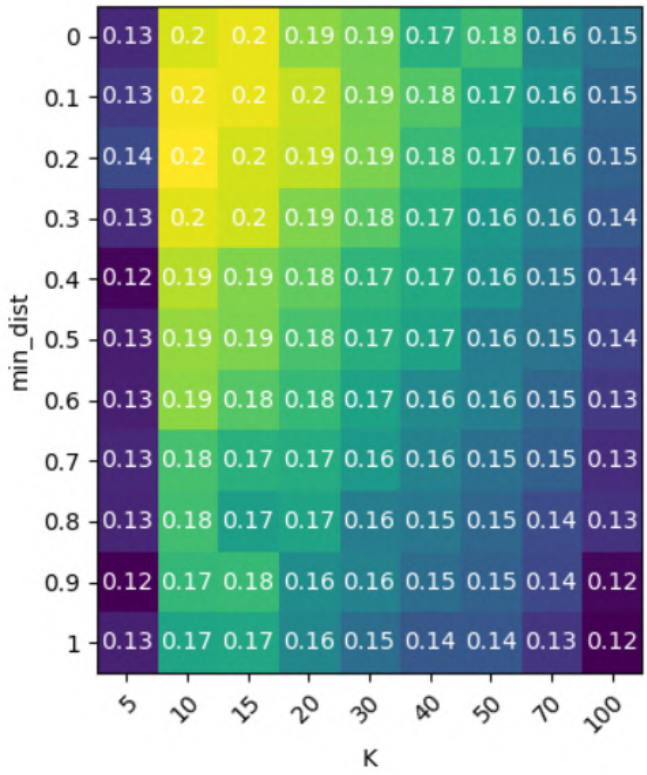

Figura 4.17: mAP alcanzado para distintos conjuntos de entrenamiento en el dataset eCommerce.

En el caso de eCommerce también se observa un alto impacto en la cantidad de vecinos ocupada, alcanzando mejores resultados para valores de K entre 10 y 15. Se estima que esta diferencia en el valor óptimo para Flickr15K y eCommerce se debe a la cantidad de imágenes por cada clase. Flickr15K cuenta con 14.600 imágenes y 33 clases, por lo que cada clase cuenta en promedio con cerca de 400 ejemplos, mientras que eCommerce cuenta con 10.600 imágenes repartidas entre 133 clases distintas, dejándolo con cerca de 80 ejemplos por clase. Debido a esta gran diferencia en la cantidad de ejemplos por clase, se plantea que UMAP se beneficia al considerar un mayor número de vecinos en el caso de Flickr15K, ya que es más probable que los vecinos sean elementos de la misma clase, mientras que en el caso de eCommerce considerar más vecinos tenderá a generar más ruido, ya que es menos probable que estos vecinos pertenezcan a la misma clase, debido a la baja cantidad de ejemplos por clase.

Se observa que en el dataset de eCommerce min_dist juega un rol importante, obteniendo mejores resultados para valores entre 0 y $0,3$, lo que indica que este dataset se ve beneficiado por representaciones que agrupen más los elementos de la mismas clases.

4.8.2. Entrenar UMAP con menos datos

Debido al incremento que genera UMAP en el consumo de memoria, se estudió el impacto que tendría entrenar UMAP con menos datos. Esto debido a que el peso de la transformación aprendida por UMAP es proporcional a la cantidad de datos utilizados en el entrenamiento. Para ello, se entrenó UMAP con subconjuntos de Flickr15K, partiendo con un 10 % del dataset y subiendo en rangos de 10 %. En la tabla 4.9 se presenta el mAP y el MRR alcanzados, además de la memoria

utilizada para cada tamaño del conjunto de entrenamiento.

	mAP	MRR	Memoria
10 %	0.48	0.57	12 MB
20 %	0.53	0.61	24 MB
30 %	0.52	0.58	134 MB
40 %	0.55	0.60	182 MB
50 %	0.56	0.60	224 MB
60 %	0.58	0.60	273 MB
70 %	0.56	0.59	319 MB
80 %	0.58	0.61	361 MB
90 %	0.57	0.58	408 MB
100 %	0.58	0.59	453 MB

Tabla 4.9: Memoria utilizada y métricas alcanzadas según el tamaño del conjunto de entrenamiento.

Con respecto al uso de memoria, se observa que disminuye considerablemente mientras menos datos se usen en el entrenamiento. Logrando como resultado menos del 3 % del consumo de memoria original original al utilizar el 10 % del dataset para el entrenamiento. Por otra parte, el mAP disminuye mientras menor sea el conjunto de entrenamiento, aunque incluso utilizando el 10 % del dataset, el método reporta mejores resultados que el modelo base. Se observa que a partir del 60 % se alcanzan resultados similares a los obtenidos utilizando el dataset completo.

Finalmente se observa que el MRR obtenido no se ve afectado por el tamaño del conjunto de entrenamiento.

4.8.3. mAP por clase

Bajo la hipótesis de que UMAP mejora su comportamiento mientras mejor sean las representaciones base, se estudió el mAP alcanzado por UMAP para cada clase de Flickr15K y se comparó con el obtenido por el modelo base. Con esto se esperaba que en aquellas clases en la que el modelo base presentaba un mal comportamiento, este empeorara con UMAP. En la figura 4.18 se presenta un gráfico comparando ambos métodos.

Como se puede observar, UMAP logra mejorar el mAP alcanzado para la mayoría de las clases del dataset, sin embargo, no se logra observar un patrón de qué clases mejoran en base al desempeño del modelo base. Se puede observar que tres de las cuatro clases en las que UMAP empeora, presentan un mAP bajo en el modelo base (clases 17, 19 y 31), pero además se pueden observar otras clases con mAPs similares que logran subir su desempeño, como es el caso de las clases 16, 28 y 30. Además, se observa que la otra clase que empeora sus resultados, que corresponde a la clase 0, es una de las que presenta mejores resultados en las mediciones originales.

4.8.4. UMAP paramétrico

Mientras se realizaba este estudio, Sainburg et al. [19] desarrolló una estrategia para obtener embeddings similares a los generados por UMAP, pero utilizando exclusivamente redes neuronales. Esta estrategia permitiría reducir el uso de memoria ya que remplazaría la transformación de UMAP

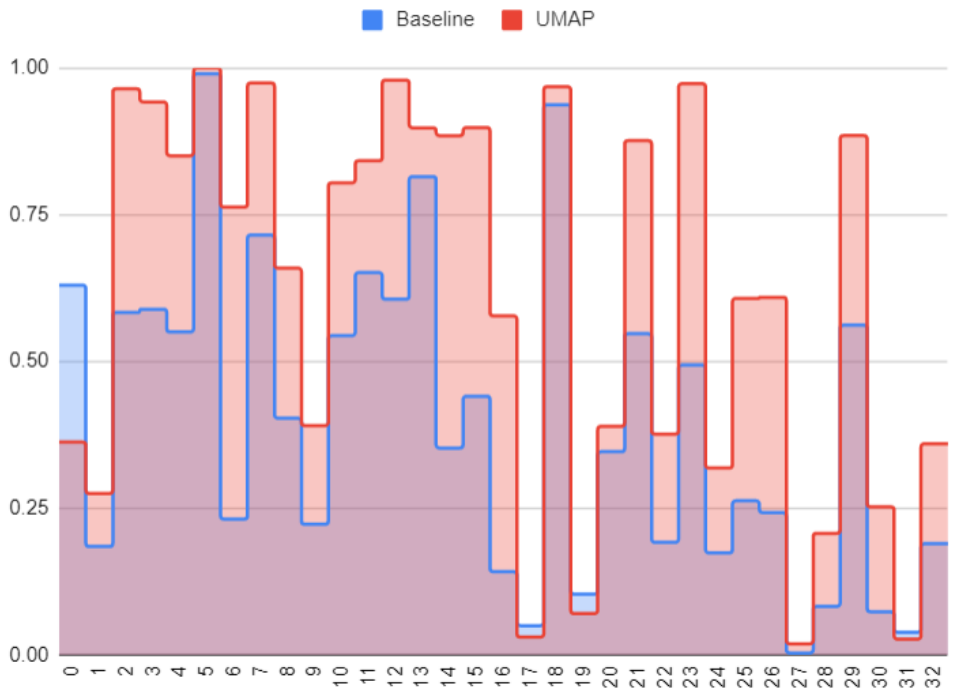

Figura 4.18: mAP por clases para el dataset de Flickr15K.

por una red neuronal de menor peso. Debido a esta ventaja, se decidió experimentar con la estrategia de UMAP paramétrico.

Para este experimento se entrenó UMAP paramétrico en el dataset de Flickr15K siguiendo la misma metodología utilizada con UMAP. Se utilizó para la implementación una librería de python desarrollada por los autores del trabajo, además se utilizó la arquitectura por defecto de la librería. En la tabla 4.10 se muestran el mAP y MRR alcanzados, además de la memoria utilizada por la red para distintas dimensiones objetivo.

	mAP	MRR	Memoria
1024	0.32	0.42	345 MB
512	0.31	0.42	325 MB
256	0.32	0.43	314 MB
128	0.31	0.41	309 MB
64	0.31	0.41	310 MB
32	0.28	0.36	307 MB
16	0.28	0.39	307 MB
8	0.25	0.36	306 MB
4	0.24	0.34	304 MB

Tabla 4.10: Memoria utilizada y métricas alcanzadas con UMAP paramétrico.

Como se puede observar los resultados obtenidos por UMAP paramétrico son considerablemente inferiores a los obtenidos por UMAP, tanto en términos de mAP como MRR. Se puede observar una mejora en el uso de memoria, pero aún sigue utilizando una cantidad de memoria considerable con respecto a las demás estrategias. Se espera que probando otras arquitecturas de UMAP paramé-

trico se debieran obtener mejores resultados, pero no se comprobó esta hipótesis dado que escapa del propósito de este trabajo.

4.9. Análisis general

Como se observa en los resultados presentados anteriormente, UMAP entrega mejoras considerables en términos de mAP, aunque estas mejoras se logran a cambio de un mayor consumo de memoria y un tiempo de consulta ligeramente mayor al de los demás métodos.

En base a los experimentos entrenando UMAP con subconjuntos del dataset, se estima que es posible aminorar el impacto que este método tiene sobre el consumo de memoria al menos en un 40 % sin afectar el desempeño. Es posible bajar el consumo aún más, aunque esto impacta levemente en el desempeño obtenido. Por otra parte, los experimentos realizados con UMAP paramétrico fallan en mantener la precisión del método original. En consecuencia, la versión paramétrica de UMAP requiere mayor estudio para lograr resultados comparables a la versión no paramétrica.

También se observó que UMAP mejora su comportamiento mientras mejores sean las representaciones originales, sin embargo, cuando las representaciones originales no encuentran suficientes vecinos de la misma clase, UMAP tiende a recuperar imágenes de clases completamente distintas. Debido a esto se pudieron observar considerables mejoras en el dataset de Flickr15K mientras que en el dataset de eCommerce las mejoras fueron moderadas.

PCA mostró buenos resultados en términos de mAP para dimensiones objetivo altas. Aunque se pudo apreciar un peor desempeño en términos de MRR. Se pudo observar que los métodos binarios de DSH y CBS obtuvieron los mejores resultados en términos de MRR, obteniendo valores comparables con el modelo base para 64 dimensiones. Tanto CRS como DTSH mostraron resultados inferiores al resto de los métodos en la mayoría de las métricas.

Capítulo 5

Conclusión

A partir de los resultados obtenidos en este trabajo se concluye que no existe una única y mejor estrategia de reducción, ya que los resultados varían de acuerdo a cada situación. UMAP presentó resultados considerablemente superiores al resto, incluso mejorando el modelo base en términos de mAP. Sin embargo, se observó que tiende a binarizar la calidad de los resultados, lo cual puede resultar negativo dependiendo de la aplicación que se le quiera dar al sistema SBIR. Además, se observó que UMAP incrementa la memoria requerida por el sistema. Se estima que este efecto se debe ver aminorado en el caso de datasets de mayor tamaño, sin embargo, en el caso de datasets más pequeños, como los estudiados en este trabajo, UMAP se vuelve prohibitivo si es que uno de los objetivos es reducir el uso de memoria.

Después de UMAP, PCA presentó los mejores resultados en términos de mAP para niveles de reducción más bajos (entre 1024 a 32 dimensiones), lo que lo vuelve útil si es que no se requiere una gran reducción en el tiempo de consulta o memoria utilizada y se quiere recuperar los elementos de una clase de manera consistente.

Si se requiere un nivel extremo de reducción, la reducción binaria usando una capa sigmoidal presenta los mejores resultados en términos de mAP.

Para los casos en que la recuperación de la primera imagen relevante es lo más importante, CBS y DSH mostraron los mejores resultados en términos de MRR con una dimensión objetivo de 64, aunque determinar cuál de estos resulte superior podría depender del dataset mismo, como se observó en los resultados.

Otro aporte del trabajo fue evaluar estrategias para aminorar el impacto que tiene UMAP en el uso de memoria. El entrenamiento con subconjuntos del dataset destaca por reducir la memoria utilizada en un 40 % sin afectar la efectividad del método. Por otra parte, UMAP paramétrico mostró resultados poco favorables.

Finalmente, se concluye que los métodos evaluados en este trabajo logran el objetivo de obtener representaciones compactas sin degradar la calidad de los resultados, logrando así reducir el tiempo y la memoria utilizada por los sistemas SBIR.

5.1. Trabajos futuros

A partir de los resultados obtenidos en este estudio se recomienda avanzar en el futuro en los siguientes trabajos:

- **Mejorar el dataset de eCommerce:** El conjunto de imágenes utilizado estaba pensado originalmente para la recuperación de imágenes basada en dibujos con color. Debido a esto, se pueden encontrar múltiples clases que sólo son diferenciables entre si por su esquema de color, lo cual vuelve sumamente difícil la recuperación de dichas clases para modelos que no se basan en color como los estudiados en este trabajo. Una posible mejora al dataset consistiría en revisar dichas clases con formas similares y quedarse con sólo una o fusionarlas en una nueva clase.
- **Optimizar las implementaciones:** A lo largo de este trabajo se presentaron dificultades a la hora de medir los tiempos de consulta debido a que python es ineficiente comparado a otros lenguajes y no era posible llevar a cabo todos los procesos con librerías como numpy. Se estima que se podrían obtener mejores mediciones realizando los experimentos en lenguajes más eficientes como C++.
- **Experimentar con un modelo base**: Sería valioso experimentar con un modelo base que presente mejores resultados, como los obtenidos por Bui o Fuentes, en especial en el caso de UMAP, que demostró un funcionamiento superior mientras mejores fueran las representaciones base.
- **Estudiar en profundidad optimizaciones de UMAP:** En el trabajo se evaluaron algunas optimizaciones de UMAP de manera superficial, con el objetivo de reducir el uso de memoria. Se recomienda profundizar más en estos experimentos, en especial en el caso de UMAP paramétrico.
- **Probar UMAP en otros contextos:** En este trabajo se utilizó UMAP en el contexto de SBIR, pero se estima que cualquier sistema de recuperación de imágenes se podría ver beneficiado con las mejoras de UMAP.

Bibliografía

[1] Bui, T., L. Ribeiro, M. Ponti y J. Collomosse: *Compact descriptors for sketch-based image retrieval using a triplet loss convolutional neural network*. Computer Vision and Image Understanding, 164:27–37, 2017.

[2] Bui, Tu, Leonardo Ribeiro, Moacir Ponti y John Collomosse: *Sketching out the Details: Sketch-based Image Retrieval using Convolutional Neural Networks with Multi-stage Regression*. Computers & Graphics, 71:77–87, 2018.

[3] Eitz, Mathias, James Hays y Marc Alexa: *How Do Humans Sketch Objects?* ACM Trans. Graph. (Proc. SIGGRAPH), 31(4):44:1–44:10, 2012.

[4] Fuentes, Aníbal: *Recuperación de imágenes basada en dibujos mediante redes convolucionales*. Memoria para optar al título de ingeniero civil eléctrico, Universidad de Chile, 2020.

[5] He, Kaiming, Xiangyu Zhang, Shaoqing Ren y Jian Sun: *Deep Residual Learning for Image Recognition*. CoRR, abs/1512.03385, 2015. http://arxiv.org/abs/1512.03385.

[6] Hu, Jie, Li Shen y Gang Sun: *Squeeze-and-Excitation Networks*. CoRR, abs/1709.01507, 2017. http://arxiv.org/abs/1709.01507.

[7] Hu, Rui y John Collomosse: *A performance evaluation of gradient field HOG descriptor for sketch based image retrieval*. Computer Vision and Image Understanding, 117(7):790–806, Julio 2013.

[8] Krizhevsky, Alex, Ilya Sutskever y Geoffrey E Hinton: *ImageNet Classification with Deep Convolutional Neural Networks*. En Pereira, F., C. J. C. Burges, L. Bottou y K. Q. Weinberger (editores): *Advances in Neural Information Processing Systems*, volumen 25. Curran Associates, Inc., 2012. https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[9] Lecun, Y., L. Bottou, Y. Bengio y P. Haffner: *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[10] Liu, H., R. Wang, S. Shan y X. Chen: *Deep Supervised Hashing for Fast Image Retrieval*. En *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 2064–2072, 2016.

[11] Liu, Li, Fumin Shen, Yuming Shen, Xianglong Liu y Ling Shao: *Deep Sketch Hashing: Fast*

Free-Hand Sketch-Based Image Retrieval. En *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, páginas 2298–2307. IEEE Computer Society, 2017.

[12] Maaten, Laurens van der y Geoffrey Hinton: *Visualizing Data using t-SNE*. Journal of Machine Learning Research, 9:2579–2605, 2008.

[13] McInnes, Leland, John Healy y James Melville: *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*, 2020.

[14] Qi, Yonggang, Yi Zhe Song, Honggang Zhang y Jun Liu: *Sketch-based image retrieval via siamese convolutional neural network*. En *IEEE International Conference on Image Processing (ICIP)*, páginas 2460–2464, 2016.

[15] Saavedra, J. M.: *Sketch based image retrieval using a soft computation of the histogram of edge local orientations (S-HELO)*. En *2014 IEEE International Conference on Image Processing (ICIP)*, páginas 2998–3002, 2014.

[16] Saavedra, Jose y Benjamin Bustos: *Sketch-based image retrieval using keyshapes*. Multimedia Tools and Applications, 73:2033–2062, Diciembre 2013.

[17] Saavedra, Jose M.: *Diapositivas del curso CC7221*, 2021. `https://www.u-cursos.cl/ingenieria/2021/1/CC7221/1/material_docente/detalle?id=3941954`.

[18] Saavedra, Jose M. y Juan Manuel Barrios: *Sketch based Image Retrieval using Learned KeyShapes (LKS)*. En *Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015*, páginas 164.1–164.11, 2015.

[19] Sainburg, Tim, Leland McInnes y Timothy Q. Gentner: *Parametric UMAP: learning embeddings with deep neural networks for representation and semi-supervised learning*. CoRR, abs/2009.12981, 2020. `https://arxiv.org/abs/2009.12981`.

[20] Sangkloy, Patsorn, Nathan Burnell, Cusuh Ham y James Hays: *The Sketchy Database: Learning to Retrieve Badly Drawn Bunnies*. ACM Transactions on Graphics (proceedings of SIGGRAPH), 2016.

[21] Simonyan, Karen y Andrew Zisserman: *Very Deep Convolutional Networks for Large-Scale Image Recognition*. En Bengio, Yoshua y Yann LeCun (editores): *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. `http://arxiv.org/abs/1409.1556`.

[22] Wang, Xiaofang, Yi Shi y Kris M. Kitani: *Deep Supervised Hashing with Triplet Labels*. CoRR, abs/1612.03900, 2016. `http://arxiv.org/abs/1612.03900`.

[23] Yu, Q., F. Liu, Y. Song, T. Xiang, T. M. Hospedales y C. C. Loy: *Sketch Me That Shoe*. En *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 799–807, 2016.