UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

APRENDIZAJE AUTO-SUPERVISADO PARA DETECCIÓN ONE-SHOT EN DOCUMENTOS HISTÓRICOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

MARCELO ANDRÉS PIZARRO TAPIA

PROFESOR GUÍA:
JOSÉ SAAVEDRA RONDO

MIEMBROS DE LA COMISIÓN:
IVÁN SIPIRÁN MENDOZA
JORGE SILVA SÁNCHEZ

SANTIAGO DE CHILE
2024

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: MARCELO ANDRÉS PIZARRO TAPIA
FECHA: 2024
PROF. GUÍA: JOSÉ SAAVEDRA RONDO

APRENDIZAJE AUTO-SUPERVISADO PARA DETECCIÓN ONE-SHOT EN DOCUMENTOS HISTÓRICOS

La digitalización continua de documentos históricos ha facilitado el acceso a nuestro patrimonio cultural. Sin embargo, la detección de objetos en estos documentos sigue siendo un desafío complejo debido a la falta de datos etiquetados y a su diversidad y complejidad. En este contexto, la utilización de modelos entrenados de forma auto-supervisada puede ser particularmente útil para extraer características únicas de símbolos y objetos, gracias a su gran capacidad de generalización. Por esta razón, este trabajo se centra en desarrollar un *framework* basado en la extracción de características usando el modelo auto-supervisado DINOv2 para la detección *one-shot* en documentos históricos.

El desarrollo del framework se divide en varias etapas. Primero, se analiza el dataset DocExplore, compuesto por 1500 imágenes de documentos históricos y 1447 *queries*. Luego, se realizan pruebas con el *encoder* de DINOv2 para analizar los *embeddings* resultantes. Posteriormente, se desarrolla un método para eliminar los bordes sin información de los documentos del *dataset* y se experimenta con la correlación 2D de las características extraídas. Se evalúa la tarea de recuperación de imágenes, probando diferentes configuraciones y métodos de comparación de características. Por último, se experimenta con métodos de *re-ranking* y de reducción de dimensionalidad para mejorar el desempeño del modelo.

Los resultados experimentales muestran que el *framework* desarrollado se acerca significativamente al estado del arte en la tarea de recuperación de imágenes, superando los resultados de algunos métodos anteriores en *queries* grandes. Además, el método para eliminar bordes consiguió una mejora en el desempeño del modelo, logrando mantener la información útil en la mayoría de los casos. Sin embargo, se observa un bajo desempeño en las *queries* pequeñas no cuadradas y una disminución general de las métricas al utilizar UMAP para reducir la dimensionalidad de los *embeddings*. Además, los métodos de *re-ranking* desarrollados no ayudaron a mejorar las métricas en las clases con peor desempeño.

Los desafíos a futuro incluyen un mejor ajuste de los parámetros de UMAP y una selección adecuada de los conjuntos de entrada. Además, se sugiere realizar un entrenamiento auto-supervisado de DINOv2 con los documentos del *dataset* DocExplore y evaluar el *framework* en otros conjuntos de datos históricos para validar su eficacia y capacidad de generalización.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas aquellas personas que, de una u otra manera, contribuyeron al desarrollo y finalización de este trabajo de título.

En primer lugar, agradezco a mi familia, que con su amor, comprensión y apoyo incondicional me han permitido llegar hasta aquí. A mis padres, por enseñarme el valor del esfuerzo y la perseverancia; a mi abuelita Nancy, por acogerme y apoyarme durante mis estudios; a mi hermano Panchito y mi polola Paty, por su compañía y respaldo en los momentos más difíciles.

También quiero agradecer a mis amigos, quienes con su apoyo y compañerismo hicieron de este camino una experiencia más enriquecedora y llevadera. Sus gestos y palabras de aliento han sido esenciales para mantenerme enfocado y motivado.

Finalmente, quiero agradecer a mi profesor José Saavedra y a mis compañeros Cristóbal, Christopher y Luis, quienes me guiaron y ayudaron en todo el desarrollo de este trabajo, resolviendo cualquier duda que tenía.

A todos ustedes, mi más sincero agradecimiento.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación y formulación del problema

En las últimas décadas, los avances tecnológicos han revolucionado el acceso a cantidades significativas de datos digitales gracias a sistemas de almacenamiento más asequibles. En este contexto, la digitalización continua de documentos históricos ha permitido un acceso más amplio a nuestro patrimonio cultural. Sin embargo, debido a la falta de datos etiquetados en estos documentos, es importante abordar la búsqueda y detección de símbolos u objetos relevantes por medio del contenido visual de las imágenes más que solo los metadatos de estas. En este sentido, la detección de objetos en documentos históricos sigue siendo un desafío complejo debido a la diversidad y complejidad de estos materiales, que pueden incluir una amplia gama de ilustraciones y formatos diversos.

La mayoría de los métodos existentes para la detección de objetos, requieren una gran cantidad de datos etiquetados para entrenar modelos precisos, lo que puede resultar costoso y consumir una cantidad significativa de tiempo. Una alternativa prometedora es la utilización de modelos auto-supervisados, que pueden reducir significativamente la cantidad de datos etiquetados necesarios para lograr un entrenamiento preciso. Estos modelos se caracterizan por su capacidad de aprender de manera autónoma a partir de los datos disponibles, sin depender exclusivamente de anotaciones humanas. Además, los modelos auto-supervisados pueden mejorar la generalización del modelo a diferentes conjuntos de datos y escenarios, lo que aumenta su versatilidad.

Dentro del amplio espectro de modelos auto-supervisados, los *Vision Transformers* (ViT) han demostrado ser particularmente efectivos en tareas de procesamiento de imágenes. Los ViT son una clase de modelos basados en la arquitectura de los *transformers*, originalmente diseñados para tareas de procesamiento de lenguaje natural. Estos modelos procesan las imágenes como una secuencia de parches, aplicando mecanismos de atención que permiten al modelo centrarse en partes específicas de la imagen que son más relevantes para la tarea en cuestión. Esta capacidad los hace ideales para la detección de objetos en documentos históricos, donde es crucial identificar y diferenciar entre detalles finos y patrones complejos en las ilustraciones.

Uno de los desarrollos recientes en este campo es DINO (*Self-DIstillation with NO labels*), una técnica de entrenamiento auto-supervisado para *Vision Transformers* que ha sido adap-

tada en la versión DINOv2. Esta nueva versión de DINO permite que los modelos entrenen de manera más eficiente mediante un proceso de destilación del conocimiento, donde un modelo *teacher* guía el aprendizaje de un modelo *student* a través de la comparación de representaciones de características. Este enfoque no solo mejora la eficiencia del entrenamiento al reducir la necesidad de etiquetas detalladas, sino que también potencia la capacidad del modelo para extraer características distintivas sin supervisión directa.

En el contexto de detección de objetos en documentos históricos, DINOv2 puede ser particularmente útil para extraer y discernir características únicas de símbolos y objetos en variados estilos de ilustración y escritura. Además, la habilidad de este modelo para adaptarse a diferentes estilos visuales y complejidades es fundamental para aplicaciones en patrimonio cultural, donde cada documento puede representar un conjunto único de desafíos en términos de contenido visual.

1.2. Objetivos

1.2.1. Objetivo general

El objetivo general de este trabajo es desarrollar un *framework* basado en la extracción de características usando el modelo auto-supervisado DINOv2 destinado a la detección *one-shot* de imágenes en documentos históricos.

1.2.2. Objetivos específicos

- Definir estrategias de procesamiento para facilitar la detección *one-shot* en documentos históricos.
- Diseñar un modelo basado en *visual encoders* para representar los documentos y consultas.
- Evaluar el modelo en el dataset DocExplore.
- Realizar comparaciones con el estado del arte.

Capítulo 2

Marco Teórico

Este marco teórico explorará las bases y desarrollos en la detección de objetos, el paradigma del aprendizaje auto-supervisado y la detección *one-shot*, así como una revisión del funcionamiento básico de algunos modelos auto-supervisados relevantes para la investigación. En particular, se detallará la arquitectura base del modelo DINOv2, el *vision transformer*, y la forma en que logra extraer características representativas de una imagen. Por último, se definirán las tareas y métricas para evaluar el desempeño del modelo.

2.1. Detección de Objetos en Imágenes

La detección de objetos en imágenes es una tarea esencial en el campo de la visión por computadora, la cual implica no solo la identificación de la presencia de un objeto en una imagen, sino también la localización precisa de su posición. Por esta razón, es común utilizar los llamados *bounding boxes* para representar la detección de un objeto. Esta tarea es fundamental para una variedad de aplicaciones, desde sistemas de seguridad hasta aplicaciones médicas y análisis de patrimonio cultural.

La detección de objetos se basa en la identificación de patrones visuales específicos que representan un objeto o una clase de objetos en una imagen. Estos patrones pueden basarse en características como bordes, texturas, colores y formas. La complejidad de esta tarea radica en la variabilidad de las apariencias de los objetos debido a factores como cambios en la iluminación, la perspectiva, la oclusión y el fondo.

2.1.1. Backbones

Antes de adentrarse en los métodos actuales de detección de objetos usando *deep learning*, es importante mencionar el uso común de arquitecturas de red pre-entrenadas como punto de partida o mejor conocidos como *backbones*. Estos *backbones* son redes neuronales convolucionales que han sido previamente entrenadas en grandes conjuntos de datos, como ImageNet [1], y han demostrado un alto rendimiento en tareas de clasificación de imágenes. Al utilizar estas redes como base, se puede aprovechar la representación de características aprendida y adaptarla a tareas específicas, mejorando así la eficiencia y precisión del modelo final. Esta técnica de entrenar *backbones* para tareas y *datasets* específicos es llamado *fine-tuning*. A continuación, se detallan dos arquitecturas de red ampliamente utilizadas en tareas de detección y segmentación de objetos.

2.1.1.1. VGG

La red VGG, desarrollada por el *Visual Geometry Group* de la Universidad de Oxford [2], es conocida por su arquitectura simple pero efectiva. Se caracteriza por tener capas convolucionales con pequeños filtros de 3x3 seguidos de capas de *pooling*. Aunque hay varias variantes de la red VGG, como VGG-16 y VGG-19 (donde los números indican la cantidad de capas con pesos), todas comparten esta estructura básica. Como ejemplo, en la figura 2.1 se puede observar la arquitectura de la red VGG-16.

Como *backbone*, VGG proporciona una representación densa de características en diferentes niveles de la red, lo que la hace adecuada para tareas que requieren una comprensión detallada del contenido de la imagen, como la segmentación semántica.

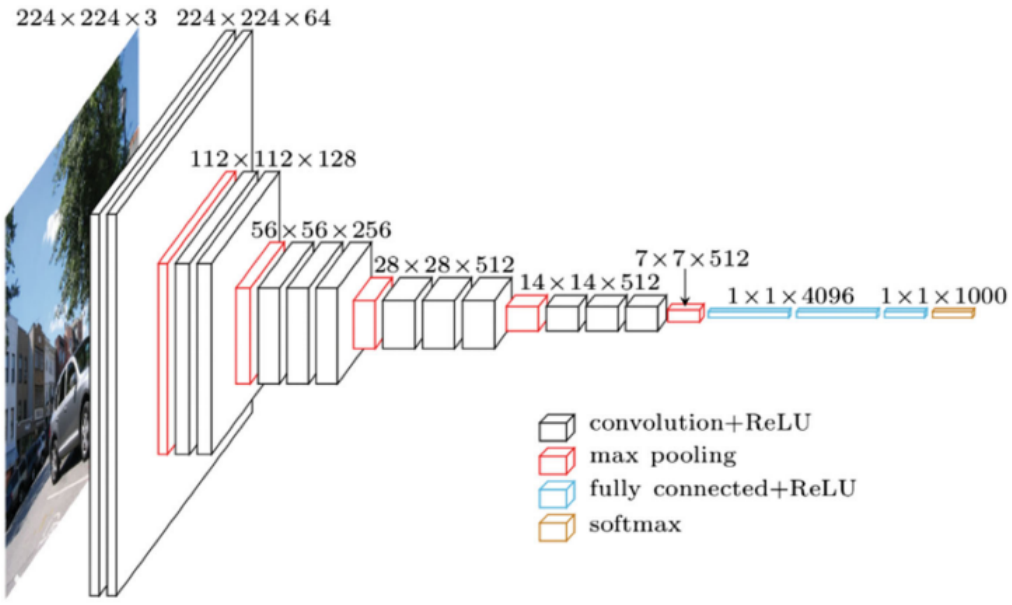

Figura 2.1: Esquema de la arquitectura de VGG-16 [3].

2.1.1.2. ResNet

ResNet (*Residual Networks*) [4], introdujo el concepto de "conexiones residuales" o "atajos" para abordar el problema de la degradación del entrenamiento en redes profundas. Estas conexiones permiten que la señal se transmita directamente de una capa a otra, saltándose algunas capas en el medio. Esto facilita el entrenamiento de redes mucho más profundas, con arquitecturas que van desde ResNet-18 hasta ResNet-152 y más allá. En la figura 2.2 se aprecia un esquema un bloque unitario usado en ResNet, el cual puede ser concatenado para crear redes más profundas.

El uso de ResNet como *backbone* es popular en muchas tareas de visión por computadora debido a su capacidad para aprender representaciones ricas y variadas a diferentes niveles de profundidad. Su diseño residual también ayuda a mantener la información de las capas anteriores, lo que es beneficioso para tareas como la detección de objetos, donde la información contextual es crucial.

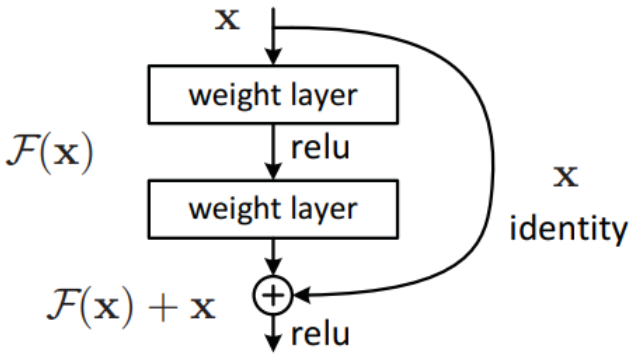

Figura 2.2: Conexión residual usada en ResNet [4].

2.1.2. Métodos de detección basados en Deep Learning

Existen diversos métodos y técnicas para la detección de objetos en imágenes. Los enfoques tradicionales se basan en características manuales y técnicas de aprendizaje automático. Sin embargo, con el auge de las redes neuronales convolucionales (CNN), la detección de objetos ha experimentado avances significativos en términos de precisión y velocidad [5]. En este sentido, los detectores de objetos basados en *deep learning* pueden ser divididos en dos categorías: *Two-Stage Detectors* y *One-Stage Detectors*.

2.1.2.1. Two-Stage Detectors

Los detectores de dos etapas, como su nombre indica, realizan la detección en dos pasos principales. En la primera etapa, se proponen regiones candidatas o *region proposals* que podrían contener objetos. En la segunda etapa, estas regiones propuestas se clasifican y refinan utilizando una red neuronal convolucional. A continuación, se presentarán algunos detectores de este tipo.

R-CNN (Regions with CNN features): R-CNN [6] fue uno de los primeros enfoques en utilizar CNN para la detección de objetos. Utiliza un método para proponer regiones candidatas basado en la agrupación jerárquica de segmentos similares en una imagen. Luego, extrae características de cada región propuesta utilizando una CNN. Estas características se pasan a través de SVM (*Support Vector Machines*) para clasificar los objetos y ajustar los *bounding boxes*. El funcionamiento de este modelo se muestra en la figura 2.3.

A pesar de su eficacia, R-CNN tiene algunas limitaciones. Es computacionalmente costoso, ya que requiere pasar cada propuesta de región a través de la CNN. Además, el proceso de entrenamiento es de múltiples etapas, lo que lo hace menos eficiente en comparación con los enfoques posteriores. Sin embargo, R-CNN sentó las bases para los avances subsiguientes en la detección de objetos, como Fast R-CNN y Faster R-CNN.

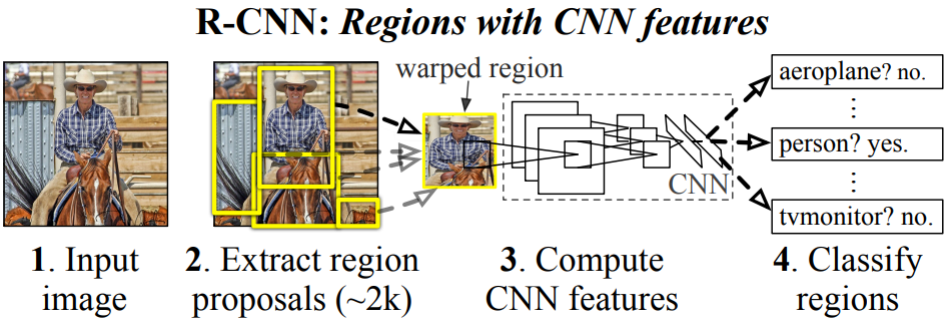

Figura 2.3: Esquema de funcionamiento de R-CNN [6].

Fast R-CNN: Una mejora del R-CNN original, Fast R-CNN [7] introduce una técnica que permite extraer características de las regiones propuestas directamente del mapa de características, eliminando la necesidad de extraer características para cada región propuesta por separado, lo cual mejora significativamente la velocidad de detección. La arquitectura de este modelo se muestra en la figura 2.4. Esta red toma como entrada una imagen y un conjunto de regiones de interés (RoI), y procesa la imagen con una red convolucional para producir un mapa de características. Luego, para cada RoI, una capa de *pooling* extrae un vector de características del mapa creado, el cual pasa por una red *fully connected*, produciendo dos vectores de salida: uno que representa las probabilidades de la función *softmax* para estimar la clase del objeto, y un vector de regresión del *offset* del *bounding box* por cada clase.

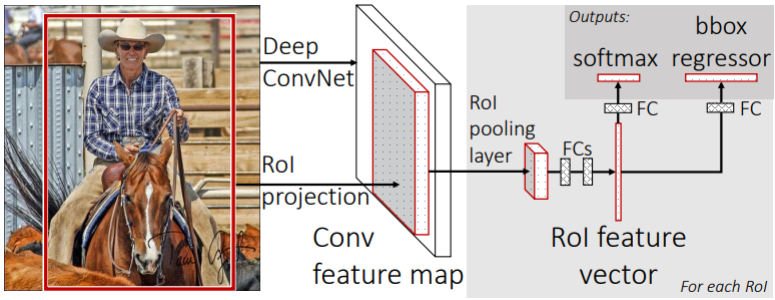

Figura 2.4: Esquema de la arquitectura de Fast R-CNN [7].

Faster R-CNN: Faster R-CNN [8] introduce una red denominada *Region Proposal Network* (RPN) que comparte convoluciones con la red de detección, permitiendo la generación de *region proposals* en tiempo real. Esto elimina la necesidad de utilizar métodos externos para proponer regiones, haciendo que la detección sea aún más rápida. En la figura 2.5 se muestra la arquitectura de este modelo integrando la RPN.

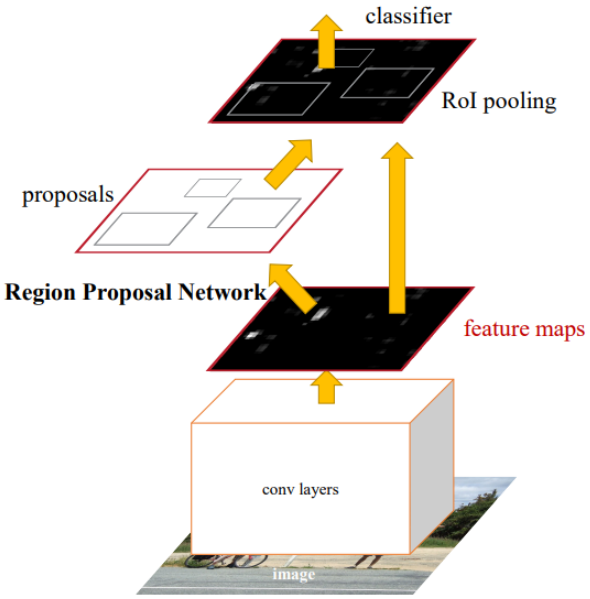

Figura 2.5: Esquema de la arquitectura de Faster R-CNN [8].

2.1.2.2. One-Stage Detectors

A diferencia de los detectores de dos etapas, los detectores de una etapa realizan la clasificación y la localización de objetos en un solo paso, considerando todas las posiciones en la imagen como potenciales objetos y tratando de clasificar cada región de interés como fondo u objeto. Esto generalmente los hace más rápidos pero a veces menos precisos que los detectores de dos etapas [9].

YOLO (You Only Look Once): YOLO [10] considera la detección de objetos como un problema de regresión, dividiendo la imagen en una cuadrícula y prediciendo los *bounding boxes* y confianzas para cada celda de la cuadrícula. Es extremadamente rápido y puede detectar objetos en tiempo real, pero en ocasiones es menos preciso en comparación con los detectores de dos etapas. En la figura 2.6 se muestra el funcionamiento de este modelo. Una red convolucional predice simultáneamente los *bounding boxes* y la probabilidad de su pertenencia a cada clase. Luego, se entrena con imágenes completas y se optimiza directamente el desempeño de la detección.

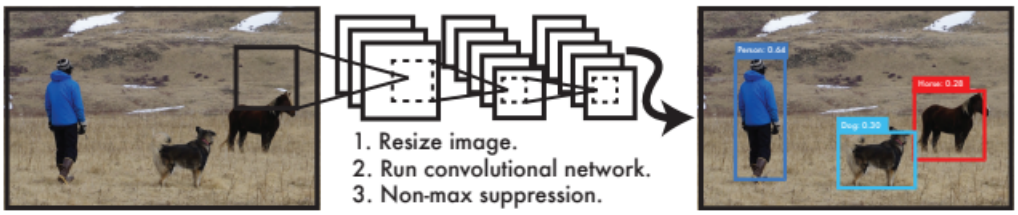

Figura 2.6: Esquema de funcionamiento de YOLO [10].

RetinaNet: RetinaNet [9] introduce una función de pérdida llamada *Focal Loss* diseñada para abordar el desequilibrio entre los fondos y los objetos en la detección. Utiliza una red de características piramidal llamada *Feature Pyramid Network* (FPN) para detectar objetos

en diferentes escalas. A pesar de ser un detector de una etapa, RetinaNet logra una precisión comparable a los detectores de dos etapas. La figura 2.7 ilustra la arquitectura de este modelo. Usando una red ResNet en conjunto con FPN como *backbone*, se calculan características a distintas resoluciones y se generan *anchors* (o regiones donde se propone la existencia de objetos) en cada nivel de la FPN. Luego, se utilizan dos subredes: una para clasificar si un objeto está presente en el *anchor* y otra para realizar la regresión que define los *bounding boxes* definitivos.

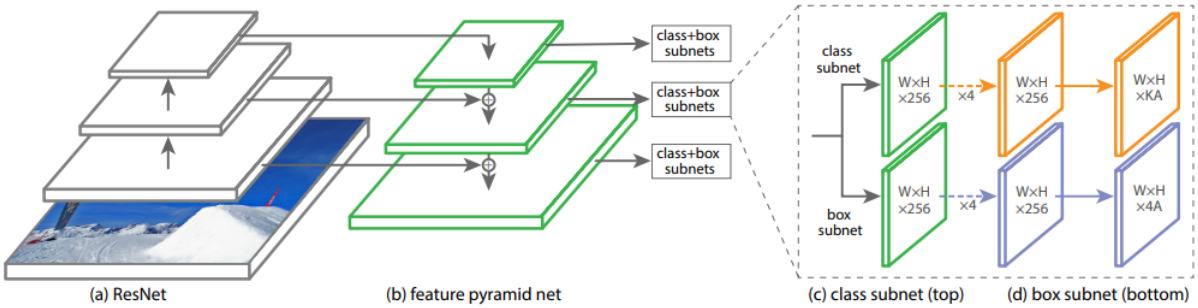

Figura 2.7: Esquema de la arquitectura de RetinaNet [9].

2.2. Transformers

Los *transformers* son una clase de modelos de *deep learning* que se basan en mecanismos de atención para procesar datos secuenciales de manera paralela y capturar dependencias a largo plazo. Introducidos en [11], han establecido el estado del arte en el procesamiento del lenguaje natural (NLP) y han sido adaptados con éxito para tareas de visión por computadora.

2.2.1. Arquitectura del Transformer

La arquitectura completa del *transformer* es altamente modular y se caracteriza por su capacidad para procesar todos los elementos de la entrada en paralelo, lo que la hace muy eficiente en comparación con los modelos secuenciales anteriores que procesaban los elementos uno por uno. Como se ilustra en la figura 2.8, el *transformer* tiene dos componentes principales: el codificador (mitad izquierda) y el decodificador (mitad derecha). Cada uno de estos componentes está compuesto por una serie de capas que se apilan verticalmente.

Codificador:

En la parte del codificador o *encoder*, cada capa tiene dos subcapas principales. La primera es una subcapa de *Multi-Head Attention*, que permite al modelo atender a diferentes partes de la secuencia de entrada simultáneamente. La segunda subcapa es una red *fully connected feed-forward* que se aplica a cada posición de manera independiente y en paralelo.
Cada una de estas subcapas está seguida por una operación de *skip connection* y normalización de capa. Esto significa que la salida de cada subcapa es la suma de su entrada y su salida, seguida de una normalización. Esta estructura ayuda a mitigar el problema del desvanecimiento del gradiente y permite que el modelo entrene más eficientemente.

Decodificador:

El decodificador o *decoder* también está compuesto por una serie de capas idénticas. Además de las dos subcapas presentes en el codificador, cada capa del decodificador tiene una

tercera subcapa que realiza *Multi-Head Attention* sobre la salida del codificador. Esto permite que cada posición en el decodificador atienda a todas las posiciones en el codificador, lo que facilita la captura de relaciones entre la entrada y la salida.
En la parte superior del decodificador, después de la última capa, se encuentra una capa lineal seguida de una función *softmax*, que genera la probabilidad de cada token (o palabra) en el vocabulario.

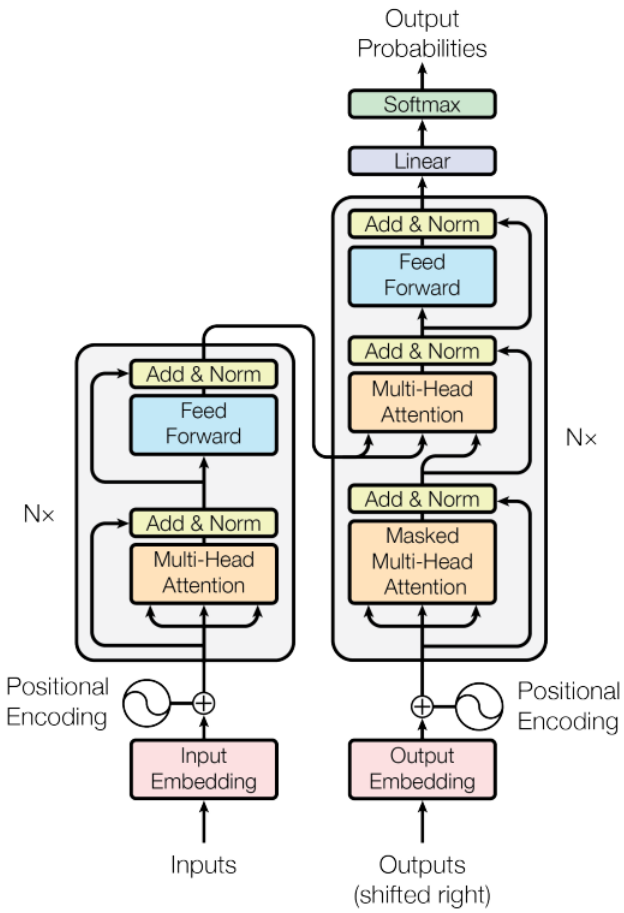

Figura 2.8: Esquema de la arquitectura de un *transformer*.

2.2.2. Mecanismos de Atención

En el centro de la arquitectura de un *transformer* se encuentra el mecanismo de atención, que permite al modelo ponderar la importancia relativa de diferentes partes de la entrada. La atención se calcula usando un conjunto de consultas (Q), claves (K) y valores (V), todos derivados de la entrada. En la figura 2.9, a la izquierda, se aprecia cómo las consultas interactúan con las claves para crear un mapa de atención que luego se aplica a los valores para crear la salida ponderada. Este cálculo se muestra en la ecuación 2.1.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

Este proceso se repite en múltiples capas de atención, permitiendo al modelo construir una representación compleja y contextual de la entrada. Esto es lo que forma la *Multi-Head*

Attention, como se ve en la figura 2.9 a la derecha, donde múltiples "cabezas" de atención permiten al modelo enfocarse en diferentes partes de la secuencia de entrada, simultáneamente. Cada cabeza calcula puntuaciones de atención, que determinan cuánto de cada elemento de la secuencia debe ser considerado al producir una representación de un *token* dado.

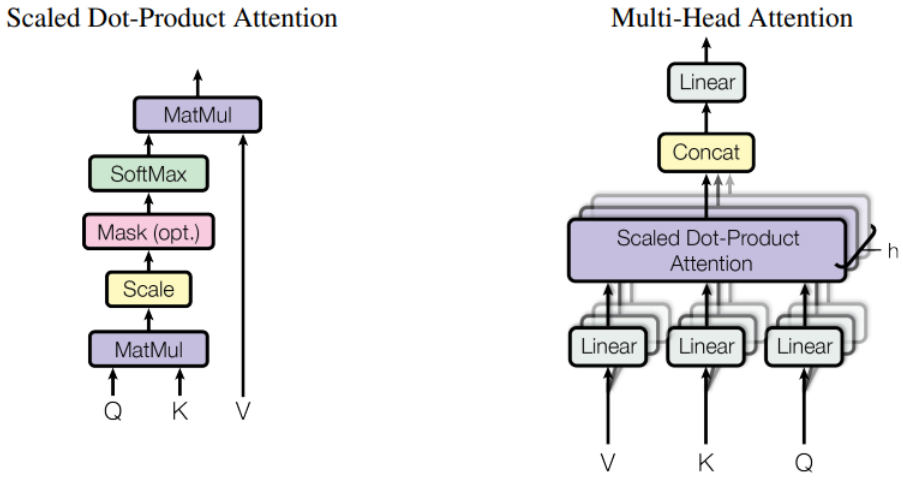

Figura 2.9: (Izquierda) Esquema del mecanismo de atención usado en [11]. (Derecha) Esquema de *Multi-Head Attention* [11].

2.2.3. Vision Transformers (ViT)

Al adaptar la arquitectura de los *transformers* al dominio de la visión computacional, los *Vision Transformers* (ViT) [12] tratan las imágenes no como una cuadrícula de píxeles, sino como una secuencia de parches. En la figura 2.10 se observa la arquitectura de este modelo, el cual toma como entrada una imagen dividida en numerosos parches cuadrados. Cada parche se aplana y se proyecta en un espacio de características más alto, similar a cómo las palabras se incrustan en NLP. Estos parches incrustados, ahora equivalentes a *tokens* en NLP, se pasan a través de una serie de capas de un *Transformer Encoder* que utilizan el mecanismo de atención para integrar la información a lo largo de toda la imagen. A la salida de este *encoder* se anexa un MLP como "cabeza" para realizar la clasificación.

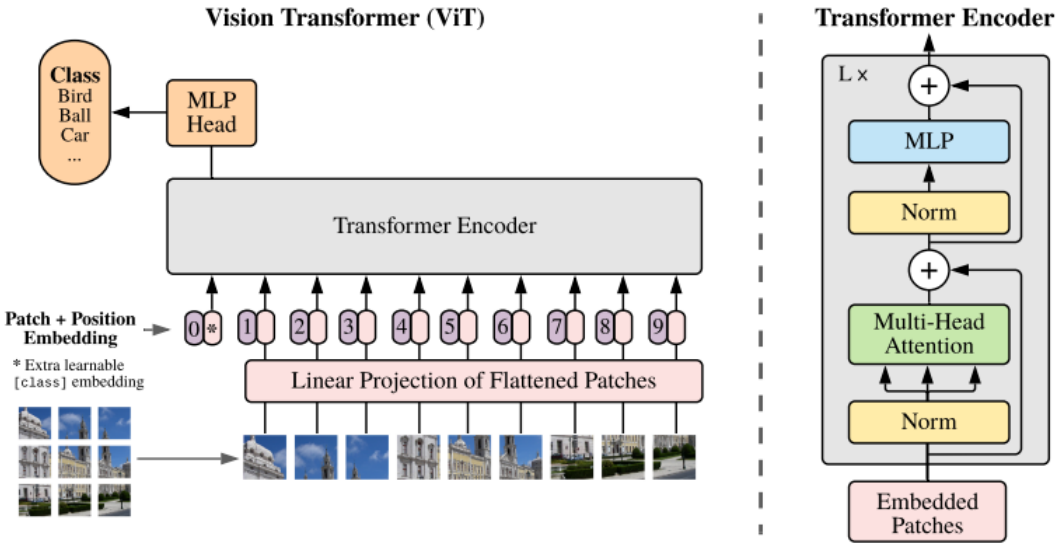

Figura 2.10: Esquema de la arquitectura de un ViT [12].

Los ViT han demostrado ser efectivos en una variedad de tareas de visión por computadora, incluyendo clasificación de imágenes, detección de objetos y segmentación semántica. Una de las ventajas de los ViT es su capacidad para capturar relaciones a largo plazo entre parches de imagen, lo que es beneficioso para comprender el contexto global de una imagen [13]. Además, los ViT pueden beneficiarse del preentrenamiento en grandes conjuntos de datos, lo que les permite aprender representaciones visuales ricas que pueden transferirse a otras tareas de visión por computadora [14].

2.3. Aprendizaje auto-supervisado

El aprendizaje auto-supervisado es una técnica de *Machine Learning* que se sitúa entre el aprendizaje supervisado y el no supervisado. En lugar de depender de datos etiquetados por humanos, el aprendizaje auto-supervisado genera sus propias etiquetas a partir de los datos de entrada. Esto se logra definiendo una tarea auxiliar, como predecir una parte de los datos de entrada a partir de otra parte, lo que permite que el modelo aprenda representaciones ricas y útiles de los datos sin la necesidad de anotaciones manuales [15].

El aprendizaje auto-supervisado se caracteriza por su capacidad para aprovechar grandes cantidades de datos no etiquetados. Al definir tareas predictivas donde la entrada sirve como su propio supervisor, los modelos pueden aprender características generales que son útiles para una variedad de tareas de *machine learning*. Estas tareas pueden incluir la predicción de la siguiente palabra en una secuencia de texto o la predicción de una parte de una imagen basada en otra [16].

2.3.1. Aplicaciones en la Detección de Objetos

En la detección de objetos, el aprendizaje auto-supervisado se utiliza para aprender representaciones de imágenes que pueden ser útiles para identificar y localizar objetos dentro de ellas. Por ejemplo, un modelo podría aprender a predecir la orientación de un objeto distorsionado o completar una imagen parcialmente oculta. Estas tareas auxiliares permiten que

el modelo desarrolle una comprensión intuitiva de la estructura y la forma de los objetos en las imágenes [17].

Además, el aprendizaje auto-supervisado puede ser particularmente útil en escenarios donde los datos etiquetados son escasos o difíciles de obtener, como es el caso de los documentos históricos. Al utilizar representaciones aprendidas a través de tareas auto-supervisadas, los modelos de detección de objetos pueden ser preentrenados en grandes conjuntos de datos no etiquetados y luego afinados con un conjunto más pequeño de datos etiquetados, mejorando así su rendimiento y eficiencia.

2.4. One-Shot Detection

La detección *one-shot* es una tarea de reconocimiento visual que desafía a los modelos a identificar y localizar objetos en imágenes después de ver ejemplos de esos objetos solo una vez. Esta tarea es particularmente desafiante debido a la escasez de datos de referencia para cada clase de objeto, lo que requiere que el modelo generalice a partir de muy poca información.

En la detección *one-shot*, los modelos deben ser capaces de aprender representaciones ricas y discriminativas de los objetos con un solo ejemplo de entrenamiento. Esto es desafiante debido a la variabilidad en la apariencia, la iluminación, la escala y la perspectiva que puede presentar un objeto en diferentes imágenes. La capacidad de generalizar a partir de un solo ejemplo y realizar inferencias precisas es crucial en aplicaciones donde la recopilación de grandes conjuntos de datos etiquetados es impracticable.

2.4.1. Recuperación de Imágenes

La recuperación de imágenes es una tarea estrechamente relacionada con la detección *one-shot*, ya que ambas requieren que los modelos identifiquen y reconozcan patrones visuales con información limitada. En la recuperación de imágenes, el objetivo es encontrar y recuperar imágenes relevantes de una base de datos grande, utilizando una imagen de consulta o *query* como referencia. Este proceso implica la extracción de características visuales discriminativas que pueden ser comparadas de manera eficiente para determinar la similitud entre la imagen de consulta y las imágenes en la base de datos.

Al igual que en la detección *one-shot*, la recuperación de imágenes enfrenta desafíos como la variabilidad en la apariencia, la iluminación y la perspectiva de los objetos en las imágenes. Las técnicas avanzadas de *machine learning*, incluyendo redes neuronales convolucionales y métodos de aprendizaje auto-supervisado, han demostrado ser efectivas para mejorar la precisión de la recuperación de imágenes. Estas técnicas permiten a los modelos aprender representaciones ricas de las características visuales, facilitando la comparación y la búsqueda eficiente en grandes conjuntos de datos.

2.4.2. Aprendizaje auto-supervisado para detección one-shot

A continuación, se detallarán tres modelos auto-supervisados que representan un avance significativo en el aprendizaje de representaciones visuales, proporcionando herramientas poderosas para abordar la detección *one-shot* y otras tareas de reconocimiento visual con

limitaciones de datos.

2.4.2.1. SimSiam (*Simple Siamese*)

SimSiam es un enfoque de aprendizaje auto-supervisado que se basa en la idea de las redes siamesas. Una red siamesa consta de dos redes neuronales idénticas, cada una procesando una de las entradas. Estas redes gemelas están conectadas por su salida final a una capa que calcula una métrica de similitud entre las representaciones de alto nivel aprendidas por las redes. Durante el entrenamiento, la red ajusta sus pesos no solo para identificar patrones en los datos de entrada, sino también para maximizar esta métrica de similitud.

SimSiam trabaja con dos "vistas" o transformaciones de una imagen, las cuales son procesadas por la misma red *encoder* (un *backbone* para extraer características). Luego, una de las representaciones se pasa por una red de predicción, la cual intenta predecir la otra representación. Finalmente, se aplica una operación de *stop-gradient* a la otra rama, lo que significa que los gradientes de la retropropagación no fluyen a través de esa rama. Esto ayuda a mantener ambas redes como funciones asimétricas. La arquitectura de este modelo se aprecia en la figura 2.11.

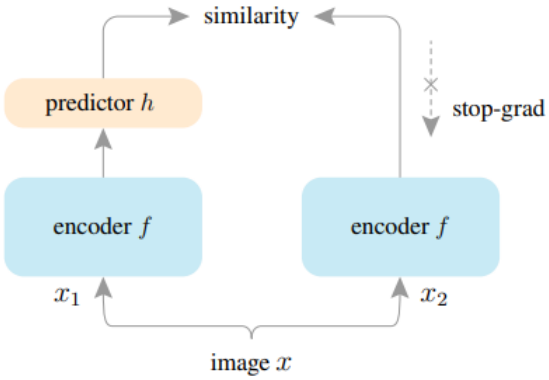

Figura 2.11: Esquema de la arquitectura de SimSiam [18].

2.4.2.2. BYOL (*Bootstrap Your Own Latent*)

BYOL [19], al igual que SimSiam, se basa en redes siamesas: una red *online* y una red *target*, que se actualizan de manera asimétrica. La red *online* transforma los datos de entrada en representaciones, luego en una proyección y por último realiza una predicción para calcular la pérdida. Por otro lado, la red *target*, quitando el predictor final, posee una estructura similar a la red *online*; sin embargo, a diferencia de ésta que se actualiza mediante retropropagación, la red *target* se actualiza como un promedio móvil exponencial de los parámetros de la red *online*. Esta característica hace que la red *target* actualice sus pesos a un ritmo más lento que la red *online*, lo que es llamado *momentum encoder* [20]. En la figura 2.12 se muestra la arquitectura de este modelo.

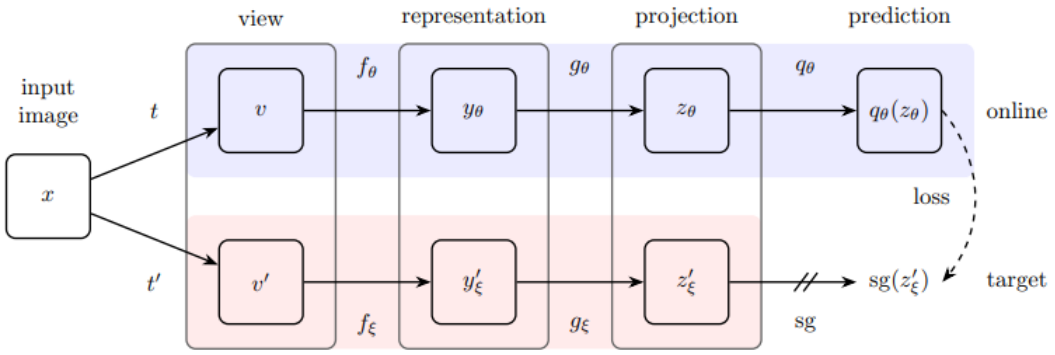

Figura 2.12: Esquema de la arquitectura de BYOL [19].

2.4.2.3. DINO (*Self-DIstillation with NO labels*)

DINO está específicamente diseñado para entrenar redes ViT sin utilizar etiquetas anotadas [21]. DINO se basa en el concepto de "destilación de conocimiento" y la idea de "enseñar a sí mismo", donde una red *student* aprende a replicar la salida de una red *teacher*. En la figura 2.13, se ilustra la arquitectura de este modelo. Como se puede observar, el modelo tiene como entrada dos transformaciones de una imagen, la cual se pasa por dos redes con la misma arquitectura pero diferentes parámetros. La salida de la red *teacher* es centrada con un promedio calculado sobre el *batch*. Luego la salida de cada red se pasa por una función *softmax* para luego aplicar la medida de similitud. Durante el entrenamiento, los parámetros de la red *teacher* se actualizan con un método similar a un *momentum encoder*, lo que ayuda a producir representaciones más consistentes y estables.

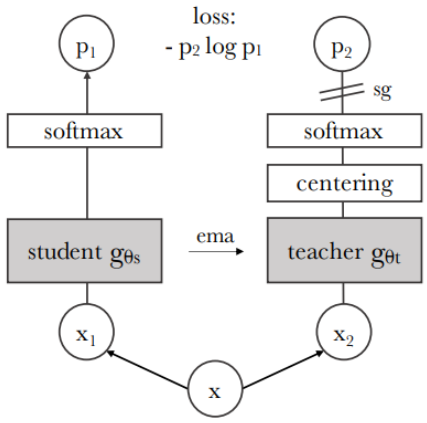

Figura 2.13: Esquema de la arquitectura de DINO [21].

2.5. DINOv2

DINOv2 representa una evolución significativa del modelo DINO original, introduciendo mejoras claves que optimizan la capacidad de los ViT para aprender de manera autosupervisada. Estas mejoras están diseñadas para incrementar la eficiencia del aprendizaje y la calidad de las representaciones características generadas por el modelo [22].

2.5.1. Conjunto de entrenamiento

DINOv2 se entrena con un conjunto de datos mucho más grande y "curado", es decir, que ha sido previamente analizado, eliminando imágenes innecesarias y balanceando la cantidad de datos según su concepto, resultando en un *dataset* de 142 millones de imágenes. Este enfoque mejora la diversidad y la representatividad de los datos de entrenamiento, lo que a su vez mejora la capacidad del modelo para generalizar a través de diferentes tareas y distribuciones de imágenes.

2.5.2. Mejoras en la arquitectura y el entrenamiento

Una de las principales innovaciones en DINOv2 es la modificación en la dinámica de actualización de los parámetros del modelo *teacher*. Mientras que DINO utiliza un método de actualización basado en *momentum*, DINOv2 introduce un enfoque adaptativo que ajusta la tasa de *momentum* en función del aprendizaje del modelo *student*. Este ajuste permite que el modelo profesor se actualice de manera más flexible, adaptándose mejor a las complejidades de los datos procesados y mejorando la estabilidad de las representaciones aprendidas.

DINOv2 también optimiza el proceso de "destilación de conocimiento" mediante la introducción de técnicas de regularización más sofisticadas. Estas técnicas incluyen la regularización de temperatura en la función *softmax*, que ayuda a suavizar las probabilidades de salida y facilita que el modelo estudiante aprenda representaciones más generales y menos propensas a sobreajustarse a las particularidades del profesor. Además, se implementa un enfoque más riguroso en el manejo de los pesos durante la destilación, equilibrando mejor la influencia de las características aprendidas.

2.5.3. Impacto en la extracción de características

Gracias a estas mejoras, DINOv2 no solo aprende de forma más eficiente, sino que también produce características visuales más discriminativas y útiles para tareas subsiguientes, como la detección y clasificación de objetos. Esto es crucial para aplicaciones en áreas donde los datos etiquetados son escasos o donde la precisión en la identificación y clasificación de elementos visuales es fundamental.

2.6. Métricas de evaluación

Para los sistemas de recuperación de imágenes es esencial medir cuán efectivos son a la hora de identificar y proporcionar imágenes relevantes basadas en consultas específicas. Una de las métricas más importantes es la precisión media promedio o *mean average precision* (mAP), pero para explicar esta métrica hace falta primero entender los conceptos de *precision* y *recall*.

2.6.1. Precision

La *precision* en sistemas de recuperación de imágenes se define como la cantidad de imágenes relevantes (bien recuperadas para la consulta hecha) dividida entre la cantidad total de imágenes recuperadas, tal como se muestra en la ecuación 2.2. Esta métrica es esencial para determinar la calidad de los resultados obtenidos, asegurando que las imágenes mostradas al usuario sean pertinentes.

$$precision = \frac{|\{\text{Imágenes relevantes}\} \cap \{\text{Imágenes recuperadas}\}|}{|\{\text{Imágenes recuperadas}\}|} \tag{2.2}$$

Por definición, la *precision* toma en cuenta todas las imágenes recuperadas, sin embargo, también puede ser evaluada considerando las primeras k imágenes recuperadas. Esta métrica es llamada *precision at* k o $P@k$.

2.6.2. Recall

El *recall* en sistemas de recuperación de imágenes evalúa la capacidad del sistema para recuperar todas las imágenes relevantes disponibles en la base de datos para una consulta específica. Se calcula según la ecuación 2.3. Esta métrica es crucial para asegurar que el sistema no omita imágenes importantes en sus resultados.

$$recall = \frac{|\{\text{Imágenes relevantes}\} \cap \{\text{Imágenes recuperadas}\}|}{|\{\text{Imágenes relevantes}\}|} \tag{2.3}$$

2.6.3. Precisión media promedio (mAP)

La precisión media promedio (mAP) combina la evaluación de la *precision* y el *recall* a lo largo de diferentes umbrales, ofreciendo una visión global sobre el rendimiento del sistema de recuperación de imágenes. La mAP se calcula promediando las precisiones obtenidas a cada punto de recuperación donde se añade una imagen relevante al conjunto de resultados, proporcionando una medida de la *precision* en diferentes niveles de *recall*. La ecuación 2.4 muestra el cálculo de esta métrica:

$$mAP = \frac{1}{Q} \sum_{q=1}^{Q} \left(\frac{1}{|Rel_q|} \sum_{k=1}^{n} P@k \times rel@k \right) \tag{2.4}$$

Donde:

- Q es el número total de consultas de prueba.
- Rel_q es el conjunto de imágenes relevantes para la consulta q.
- n es el número total de imágenes recuperadas para la consulta q.
- $P@k$ es la precisión calculada en el corte de las primeras k imágenes.
- $rel@k$ es la función de relevancia, la cual es 1 si la imagen en la posición k es relevante y 0 en caso contrario.

La mAP, por tanto, refleja no solo la capacidad de un sistema para identificar imágenes relevantes, sino también su habilidad para clasificarlas en un orden que prioriza a las más pertinentes primero.

Capítulo 3

Estado del Arte

A continuación, se presentarán cuatro *papers* con diferentes soluciones y resultados al problema de detección *one-shot* (también llamado *pattern spotting*) en el *dataset* DocExplore. En la tabla 3.1, se muestra una comparativa de los resultados de precisión media promedio de cada *paper* en la tarea de recuperación de imágenes.

Por otro lado, gracias al protocolo de evaluación presentado en [23], los resultados pueden ser separados dependiendo de la escala y forma de las *queries*. En la tabla 3.2 se muestran los resultados usando este protocolo de evaluación para los dos primeros papers.

3.1. A scalable pattern spotting system for historical documents

En este trabajo [24] se propone un sistema dividido en dos etapas: *offline* y *online*. En la etapa *offline*, se extraen regiones de interés en la imagen usando un método de separación de texto-imagen y luego se ocupa un enfoque de análisis de "ventana deslizante" para extraer características de varias subventanas. Estas representaciones se calculan usando dos descriptores compactos llamados *Vector of Locally Aggregated Descriptors* (VLAD) y *Fisher Vectors*, para finalmente proyectarse a un espacio de menor dimensión. En la etapa *online* se calcula el vector de características de la imagen de consulta y, dados los vectores de características del *dataset* completo calculados en al etapa *offline*, se utiliza una medida de similitud para realizar la detección.

Los resultados obtenidos en el *dataset* DocExplore muestran que su enfoque logra mejores resultados de recuperación, con una mayor eficiencia en términos de tiempo/memoria, en comparación con enfoques estándares. Sin embargo, el *paper* también reconoce ciertas limitaciones y desafíos. Por ejemplo, la tarea de *pattern spotting* puede ser más compleja que la detección de palabras o símbolos debido a la menor estructura y mayor variabilidad en color o textura. Además, los métodos clásicos utilizados para preprocesar documentos para la segmentación de palabras y símbolos, no se pueden aplicar en el contexto del *pattern spotting*. El sistema también debe buscar en cada ubicación posible en la imagen, lo que plantea desafíos en términos de escalabilidad y eficiencia.

3.2. Improving pattern spotting in historical documents using feature pyramid networks

En este *paper* [23] se propone el uso de una CNN, específicamente RetinaNet, para extraer características multi-escala de las regiones de los documentos y de las consultas. Al igual que en el *paper* de la sección anterior, este sistema se divide en dos etapas: una, *offline*, que se centra en el procesamiento de los documentos históricos, incluyendo la extracción de características y la indexación y otra, *online*, que se centra en el procesamiento de la consulta de entrada. En ambas etapas, se utiliza RetinaNet para extraer representaciones de las imágenes.

Los experimentos realizados en el *dataset* DocExplore muestran que su propuesta es mejor para localizar patrones y requiere menos almacenamiento para la indexación de imágenes que el sistema nombrado en la sección anterior, aunque falla en recuperar múltiples páginas que contienen instancias de la consulta. Además, ambos sistemas tienen un mal desempeño al trabajar con imágenes de consulta muy pequeñas.

3.3. Deep Learning Approaches for Image Retrieval and Pattern Spotting in Ancient Documents

Este *paper* [25] explora dos enfoques basados en *deep learning* para la tarea de recuperación y detección de imágenes en documentos históricos. El primer enfoque utiliza un CNN preentrenado, que se ajusta por medio de *fine-tuning* para lograr una representación compacta de las consultas y las regiones candidatas. Este método se basa en el aprendizaje por transferencia, aprovechando un *dataset* limitado para la afinación del modelo. El segundo enfoque emplea una Red Neuronal Convolucional Siamesa (SCNN) entrenada previamente en un subconjunto de pares de imágenes del conjunto de datos ImageNet para proporcionar mapas de características basados en la similitud. Ambos métodos consideran mapas de características de diferentes escalas y múltiples combinaciones para medir similitud.

Los autores muestran que sus enfoques basados en *deep learning* alcanzan resultados competitivos en comparación con el estado del arte. Sin embargo, una importante desventaja de estos enfoques es el excesivo número de regiones candidatas, lo que genera un tiempo de preprocesamiento mucho más largo tanto para el *dataset* como para el retorno de la consulta.

3.4. Pattern Spotting and Image Retrieval in Historical Documents using Deep Hashing

Al igual que en los trabajos anteriores, este trabajo [26] utiliza un enfoque de *deep learning* para la detección y recuperación de imágenes en documentos históricos. El proceso comienza con un algoritmo de *region proposals* que detecta candidatos a objetos en las imágenes de las páginas de documentos. Luego, se utilizan CNN para la extracción de características, considerando dos variantes distintas que proporcionan representaciones en valores reales o en código binario. Finalmente, las imágenes candidatas se clasifican calculando la similitud de características con una consulta dada. Los experimentos evalúan el enfoque propuesto considerando cada esquema de representación (valores reales y código binario) en la base de

datos de imágenes DocExplore.

Los resultados experimentales muestran que los modelos propuestos son comparables o superiores a los enfoques del estado del arte, superando a otros modelos en un 2.56 % usando las mismas técnicas para la detección de imágenes. Además, el enfoque propuesto también reduce el tiempo de búsqueda hasta en 200 veces y el costo de almacenamiento hasta en 6000 veces en comparación con trabajos basados en representaciones de valores reales.

Tabla 3.1: Resultados del estado del arte para el *dataset* DocExplore.

Método	mAP Recuperación
En et Al. [24]	**0.580**
Úbeda et Al. [23]	0.505
Wiggers et Al. [25]	0.386
Dias et Al. [26]	0.532

Tabla 3.2: Resultados del estado del arte según protocolo de evaluación presentado en [23].

Tamaño	Forma	mAP Recuperación	
		En et Al.	Úbeda et Al.
Grande	Cuadrada	**0.881**	0.749
	No cuadrada	**0.701**	0.660
Pequeña	Cuadrada	**0.801**	0.742
	No cuadrada	**0.535**	0.459

Capítulo 4

Desarrollo

El desarrollo del *framework* basado en la extracción de características usando DINOv2 se puede dividir en varias etapas. En primer lugar, se realizó un análisis del *dataset* a ocupar, para luego realizar pruebas con el *encoder* de DINOv2 y analizar los *embeddings* a su salida. Posteriormente, se trabajó en el desarrollo de un método para eliminar los bordes sin información de las documentos del *dataset* y se realizaron experimentos usando correlación 2D de las características extraídas. Luego, se evaluó la tarea de recuperación de imágenes, experimentando con diferentes configuraciones y métodos de comparación de características. Finalmente, se experimentó con una estrategia de *re-ranking* y el uso de UMAP [27] para mejorar el desempeño del modelo.

4.1. Dataset DocExplore

DocExplore es un proyecto que busca acercar a las personas a su patrimonio cultural por medio de la digitalización de documentos históricos en un formato más accesible, usando herramientas interactivas basadas en computadora [28]. Dentro de este proyecto, se generó un *dataset* con manuscritos medievales con el objetivo de resolver la tarea de *pattern spotting*, es decir, encontrar todas las ocurrencias de un objeto de interés dentro de un documento. De este modo, el *dataset* DocExplore está formado por 1500 imágenes de documentos históricos y 1447 *queries*, o imágenes de objetos de interés a buscar dentro del *dataset*. En la figura 4.1, se muestran ejemplos de las imágenes que contiene el *dataset* junto a algunas *queries* encerradas en amarillo.

Figura 4.1: Ejemplos de patrones de interés en *dataset* DocExplore (rectángulos amarillos) [28].

Dentro del *dataset*, se define un conjunto de objetos de interés para historiadores, dividido en 35 categorías de objetos, desde letras ornamentadas hasta rostros humanos y objetos decorativos. En la figura 4.2 se muestran todas las categorías de las *queries* de DocExplore y su correspondiente número de ocurrencias.

Cada *query* se diferencia de sus correspondientes ocurrencias en su misma categoría por ligeras diferencias de forma, color y tamaño, debido a los diferentes estilos de dibujo, la distorsión durante el escaneo y la degradación por el paso de los años. En la figura 4.3 se muestra la variabilidad que existe en algunas de las *queries* del *dataset*. Ejemplos de cambios de color se pueden encontrar en las categorías *"Cross"*, *"Letter S"* o *"Letter T"*, mientras que *"Double separator"* y *"Simple separator"* muestran cambios en su *aspect ratio* o relación de aspecto, y *"Statue"* y *"Ship"* muestran posibles variaciones de forma en el objeto mismo.

Por último, tomando en cuenta el protocolo de evaluación presentado en [23], se puede hacer un recuento y caracterización de las *queries* por su tamaño y forma, tal como se muestra en la tabla 4.1.

Tabla 4.1: Cantidad de *queries* según su tamaño y forma.

Tamaño	Forma	Cantidad
Grande	Cuadrada	37
	No cuadrada	21
Pequeña	Cuadrada	183
	No cuadrada	1206

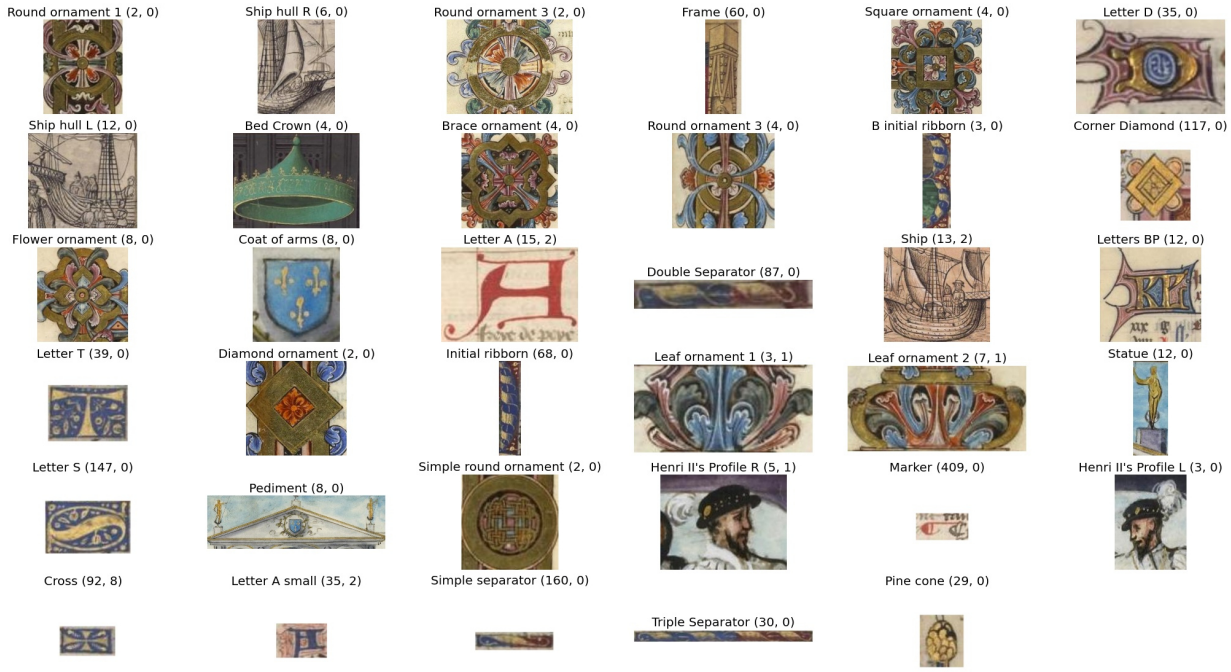

Figura 4.2: Las 35 categorías de *queries* anotadas en el *dataset* DocExplore y, en paréntesis, el número de ocurrencias y el número de objetos basura (eliminados del *dataset* final) [28].

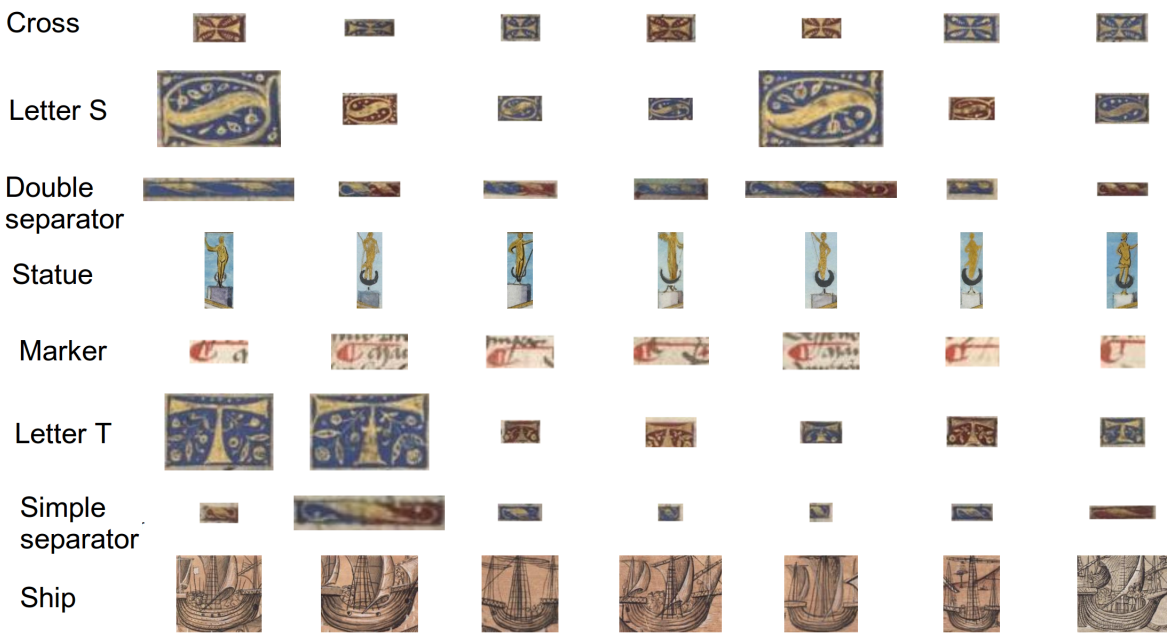

Figura 4.3: Ilustraciones de la variabilidad entre categorías de las *queries* del *dataset* DocExplore [28].

4.2. Extracción de características con DINOv2

DINOv2 cuenta con un *encoder* con una arquitectura de ViT que usa un tamaño de parche de 14 pixeles y en su modelo base tiene una dimensión de *embedding* de 768. Esto quiere decir, por ejemplo, que la salida del *encoder* para una imagen de entrada de 224x224 pixeles es un vector de largo 768 que representa a toda la imagen, llamado *class token*, y 256 vectores de largo 768 que representan cada parche de la imagen, llamados *patch tokens*. A partir de aquí, se referirá a estos *patch tokens* solo como *embeddings*, de modo que el *embedding* de una imagen serán todos los *patch tokens* que representan las características de los parches de la imagen en cuestión. En la figura 4.4 se muestra un ejemplo del *embedding* obtenido usando el *encoder* de DINOv2.

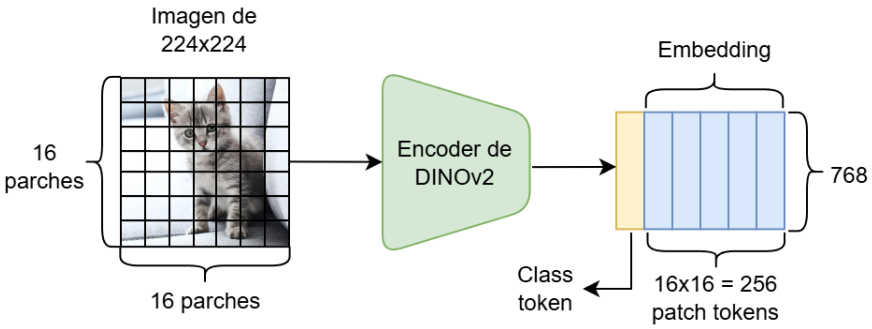

Figura 4.4: Ejemplo de *embedding* obtenido a partir del *encoder* de DINOv2.

4.3. Eliminación de bordes

Dado que las páginas que conforman el *dataset* DocExplore se componen principalmente de una zona central con información útil y un gran borde blanco en sus extremos, se hace necesaria la implementación de un método para eliminar estos bordes que no ofrecen mayor información y que sólo agregan ruido a la imagen al realizar una detección o recuperación.

En primer lugar, se tomaron cinco ejemplos de páginas en blanco sacadas directamente del *dataset* y se calcularon sus características o *embeddings* usando el *encoder* de DINOv2. Utilizando el algoritmo de *clustering k-means* con diez *clusters* a formar sobre los *embeddings* de estas páginas, se extrajeron los diez centroides para usarlos como vectores representativos del fondo blanco. Posteriormente, se extrajo las características de cada página del *dataset* y se compararon usando similitud coseno con los centroides calculados, tomando el máximo valor para cada parche de la imagen. Así, se obtiene una imagen en blanco y negro cuyos valores más altos (cercanos al blanco) corresponden a los más parecidos al fondo blanco de las páginas y los valores más bajos (cercanos al negro) corresponden a los menos parecidos.

A continuación, se toma esta imagen, se normaliza para que su valor máximo y mínimo sean 1 y 0, respectivamente, y se binariza usando un umbral de 0,4 calculado experimentalmente. Luego se le aplica una operación morfológica de dilatación y se etiqueta cada región conectada. Finalmente, se toma la región conectada más grande junto a las regiones con al menos la mitad del área de la más grande y se crea una máscara rectangular de la imagen original.

De este modo, ampliando esta zona rectangular por el tamaño del parche, se logra recortar la página para eliminar los bordes innecesarios. En la figura 4.5 se muestra un esquema del método desarrollado.

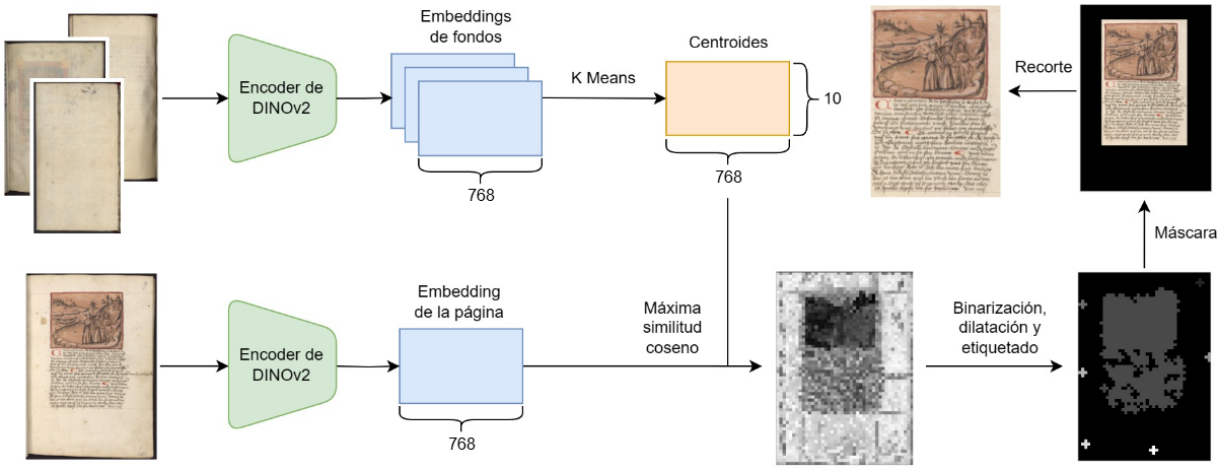

Figura 4.5: Esquema del método desarrollado para eliminación de bordes.

4.4. Correlación de embeddings

Como primera opción para comparar *embeddings* del *target* y la *query* fue la correlación en dos dimensiones. Así, extrayendo los *embeddings* con el *encoder* de DINOv2 y luego cambiando su forma, se realiza la correlación usando el *embedding* de la *query* como filtro obteniendo un mapa de calor. Buscando el valor máximo en este mapa de calor es posible encontrar la localización de la *query* sobre el *target*. En la figura 4.6 se muestra este proceso.

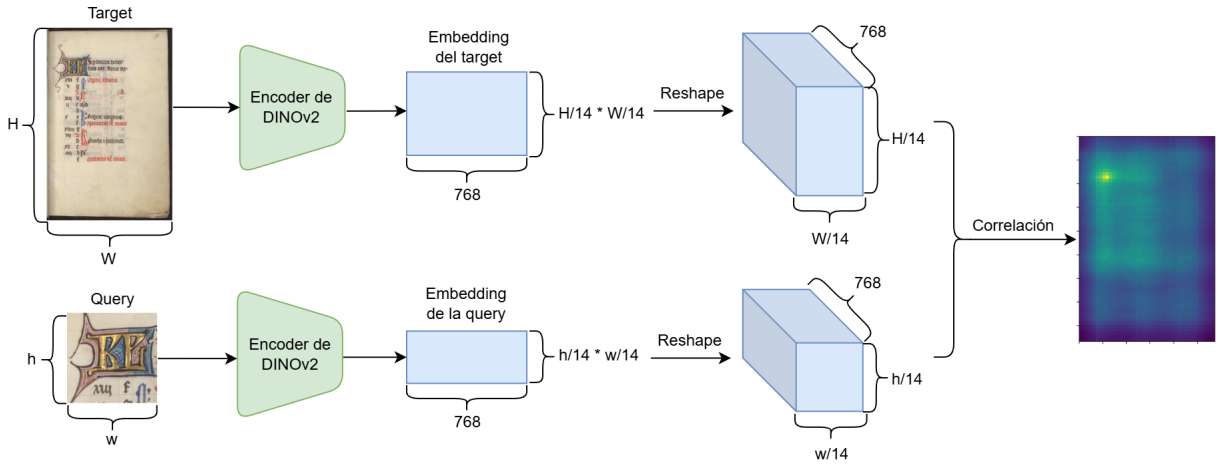

Figura 4.6: Esquema del método de correlación de *embeddings*.

4.5. Recuperación de imágenes

Una de las tareas importantes a evaluar usando el *dataset* DocExplore es la recuperación de imágenes, por lo que se comenzó realizando experimentos y evaluando los resultados en esta tarea. En primer lugar, se tomaron los *embeddings* extraídos de cada imagen *target* y de cada imagen *query*, para luego comparar cada *embedding* de las *queries* con cada *embedding* de los *targets* y así obtener un *ranking* por *query*. Este proceso se muestra en la figura 4.7.

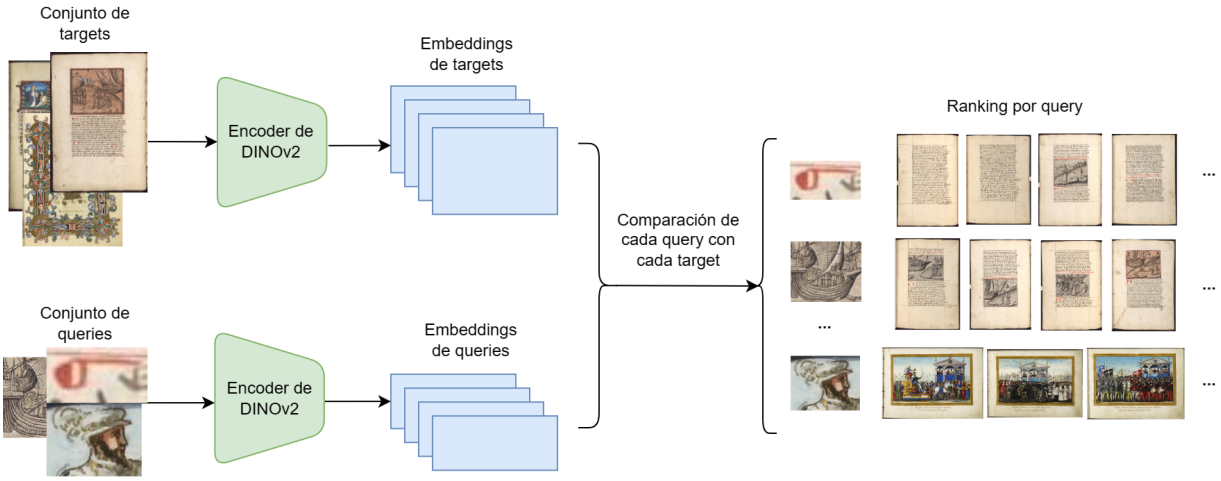

Figura 4.7: Esquema del proceso de recuperación de imágenes sobre el *dataset* DocExplore.

Se realizaron diferentes experimentos para comparar características y realizar el *ranking*. Además, se realizaron pruebas extrayendo características diferentes a partir del *encoder* de DINOv2 y agregando procesamientos adicionales a los *embeddings* extraídos. A continuación se explican los experimentos realizados:

- **Correlación:** Para realizar el *ranking* de las páginas *target*, se realizó la correlación 2D y se tomó el máximo valor por página como *score* o puntaje.
- **Correlación con *targets* sin bordes:** Utilizando el método de eliminación de bordes sobre las imágenes *target* y posteriormente calculando sus *embeddings*, se realizó el mismo proceso de correlación y *ranking*.
- ***Class Token* de la *query***: Gracias a la arquitectura de *vision transformer* del *encoder* de DINOv2, se extrajo el *class token* de la *query* y se comparó con cada *embedding* de los *targets* usando similitud coseno y distancia euclidiana, obteniendo un puntaje que es ordenado de mayor a menor en el primer caso y de menor a mayor en el segundo caso.
- ***Global Average Pooling* de la *query*:** Otra opción evaluada fue usar GAP sobre el *embedding* de la *query*, obteniendo un vector unidimensional que se compara con los *embeddings* de los *targets* usando similitud coseno.
- ***Average Pooling* a los *targets*:** Al usar GAP sobre la *query*, se volvió necesario experimentar realizando una operación de *average pooling* sobre los *embeddings* de los *targets*. Para no perder el tamaño del *embedding*, se utilizó un *stride* de 1, mientras que, dado que la relación de aspecto de la mayoría de las *queries* es no cuadrada, se utilizaron tres diferentes tamaños: 3x3, 1x3 y 3x1.

4.6. Re-ranking

Con el objetivo de mejorar los resultados en las clases del *dataset* DocExplore con menor desempeño, se evaluó una estrategia de *re-ranking* para estas clases, haciendo uso del *class token* de DINOv2 y métodos como SIFT [29] y SuperPoint [30] para el cálculo de puntos de interés. De este modo, se consideraron los primeros 100 documentos del *ranking* para cada *query*. En cada uno de ellos, se tomó un recorte de 1.5 veces el tamaño de la *query* centrado en la posición del máximo valor del mapa de calor del documento y se calculó el *class token* de cada recorte, para así compararlos con el *class token* de la *query* y obtener un nuevo *ranking* de similitud.

Por otro lado, usando los recortes mencionados, se calcularon los puntos de interés de éstos y su respectivo descriptor usando el modelo SuperPoint. Luego, se obtuvieron los calces más cercanos usando similitud coseno entre los descriptores de la *query* y de los recortes, y se utilizó el algoritmo RANSAC para encontrar los *inliers* dentro de todos los calces que coincidieran con una transformación homográfica. Finalmente, se construyó el nuevo *ranking* ordenando el promedio de los residuos de los calces según el modelo estimado por RANSAC, es decir, ordenando de menor a mayor según la distancia euclidiana entre los calces y el modelo de homografía.

Por su parte, también se utilizó el algoritmo SIFT para calcular puntos de interés y obtener calces, esta vez usando tres métodos para generar el nuevo *ranking*: ordenando los recortes de mayor a menor según la cantidad de calces entre la *query* y el recorte; ordenando según el mayor valor de similitud de un calce; y ordenando según la similitud promedio entre todos los calces de un recorte.

Por último, se utilizó como puntos de interés los parches obtenidos de DINOv2 y se realizó el mismo procedimiento para encontrar calces, esta vez usando los *embeddings* calculados con DINOv2 y generando el *ranking* de dos formas: ordenando según el mayor valor de similitud de un calce y ordenando según la similitud promedio entre todos los calces de un recorte.

4.7. UMAP

Otra idea para intentar mejorar el desempeño del modelo fue utilizar UMAP (*Uniform Manifold Approximation and Projection*) [27], un método de reducción de dimensionalidad y *clustering* de datos. La idea era mapear los *embeddings* de alta dimensión extraídos usando DINOv2 en *embeddings* de menor dimensión, tratando de preservar las características más útiles. De este modo, se utilizaron distintos subconjuntos de *embeddings* para ajustar el modelo UMAP y diferentes tamaños de salida para mapearlos.

El primer experimento se centró en analizar el desempeño de UMAP en las *queries* grandes, por lo que se extrajeron subconjuntos de *embeddings* de los *targets* en los cuales la similitud coseno con los *embeddings* de las *queries* grandes eran mayores a un umbral de 0.5. Así, se ajustó el modelo UMAP usando alrededor de 50 mil *embeddings* y tamaños de salida de 256, 128, 64 y 32. Luego, se redujo la dimensión de los *embeddings* de las *queries* y *targets* y se compararon usando similitud coseno para calcular las nuevas métricas.

Por otro lado, dado que el objetivo está centrado principalmente en encontrar las *queries* dentro de los *targets*, también se utilizaron los *embeddings* de las *queries* para ajustar el modelo UMAP, reduciendo su tamaño a 256.

Capítulo 5

Resultados y Discusión

5.1. Eliminación de bordes

En la figura 5.1 se muestran de forma gráfica algunos de los mejores y peores resultados del método de eliminación de bordes desarrollado. Tal como se aprecia en la figura 5.1.a, el método logra eliminar gran parte de los bordes de la página, manteniendo toda la información útil; mientras que en la figura 5.1.b no se logra eliminar todos los bordes debido a líneas en la página e imágenes que se traslucen del otro lado de la página. Sin embargo, aún en los peores casos, el método desarrollado no elimina partes importantes en la página, lo cual no debería implicar una disminución del desempeño a la hora de detectar o recuperar las *queries*.

(a) Mejores casos (b) Peores casos

Figura 5.1: Ejemplos de eliminación de bordes en imágenes *target*. A la izquierda, la página original. A la derecha, la página recortada.

5.2. Correlación de embeddings

En la figura 5.2 se muestran ejemplos mapas de calor obtenidos a partir de la correlación entre una *query* y una página *target*. Como se puede ver, los mapas de calor se activan o tienen un mayor valor en las zonas donde la *query* se encuentra. Sin embargo, pueden existir problemas en casos donde la *query* tiene parte del fondo, como es el caso de la piña arriba a la izquierda. En este caso, el mapa de calor se activa en varias zonas que corresponden a la hoja en blanco debido a la aparición de fondo dentro de la *query*.

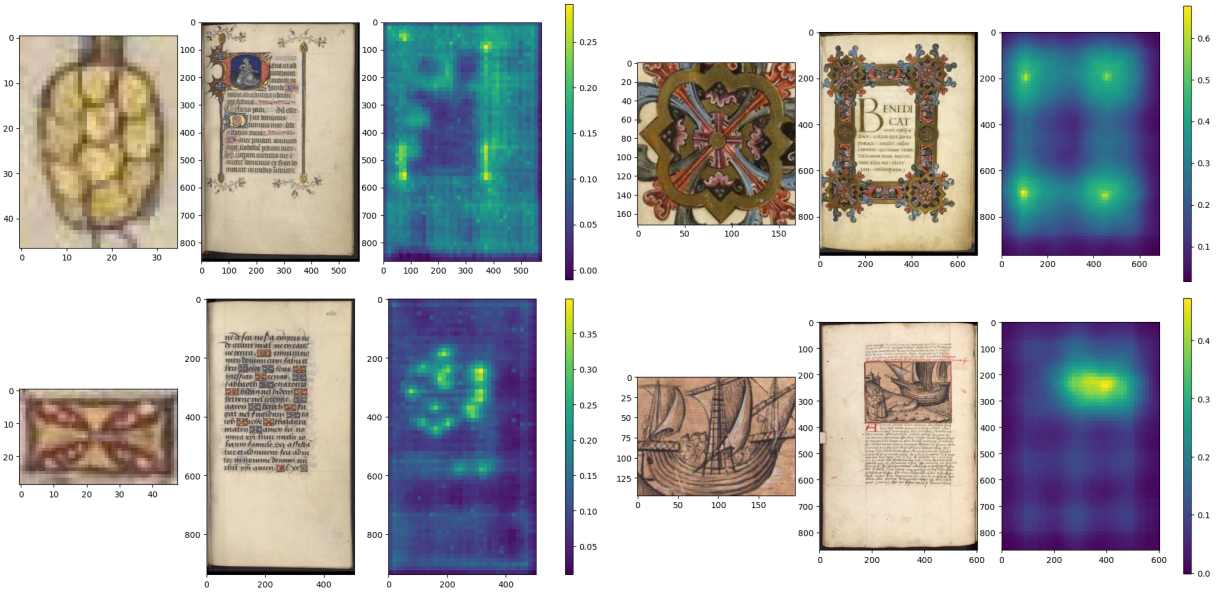

Figura 5.2: Ejemplos mapas de calor a partir de la correlación de *embeddings* entre una *query* y un *target*.

5.3. Recuperación de imágenes

Los resultados de mAP en la tarea de recuperación de imágenes separados según el tamaño y forma de la *query* y usando diferentes modelos y configuraciones, se muestra en la tabla 5.1. La columna Correl. hace referencia a la correlación 2D; CLS-T L2 se refiere al uso del *class token* de la *query* y a la distancia euclidiana como método de comparación, mientras que COS se refiere al uso de la similitud coseno; GAP hace referencia al uso de *Global Average Pooling* sobre el *embedding* de la *query*; y por último, Pool se refiere a la operación de *pooling* sobre los *targets* con su respectivo tamaño.

Tabla 5.1: Resultados de mAP en recuperación de imágenes para diferentes modelos según protocolo de evaluación presentado en [23].

Tamaño	Forma	Correl.	Correl. sin bordes	CLS-T L2	CLS-T COS	GAP	Pool 3x3	Pool 3x1	Pool 1x3
Grande	Cuadrada	0.914	**0.930**	0.771	0.809	0.848	0.832	0.855	0.815
	No cuadrada	0.779	0.792	0.796	0.799	0.733	**0.835**	0.820	0.818
Pequeña	Cuadrada	0.751	**0.779**	0.508	0.580	0.721	0.730	0.745	0.739
	No cuadrada	0.161	0.214	0.186	0.231	0.304	0.311	0.318	**0.341**

Como se puede observar de la tabla 5.1, al comparar el método de correlación usando las páginas *target* originales y las páginas sin bordes, se ve que en este último se mejora el desempeño en todas las *queries*, especialmente en las pequeñas no cuadradas. Gracias a esto, para los demás experimentos se mantuvo el uso de los *targets* sin bordes. Por otro lado, al comparar el uso del *class token* comparando con distancia L2 y similitud coseno, se aprecia que en todos los casos la similitud coseno obtiene mejores resultados, por lo que se mantuvo este método de comparación para los demás experimentos.

Dado que la mayoría de las *queries* se concentran en las pequeñas no cuadradas, como se vió en la tabla 4.1, los siguientes experimentos se centraron en mejorar el desempeño en este tipo de *queries*. De este modo, a pesar de que el uso de GAP disminuye el desempeño en las *queries* grandes no cuadradas, al aumentar en especial el desempeño de las pequeñas no cuadradas, se mantuvo para los siguientes experimentos. Por último, al comparar las operaciones de *pooling* con diferentes tamaños, se aprecia que el que tiene mejor desempeño en las *queries* pequeñas no cuadradas es el de tamaño 1x3, por lo tanto, dentro de los experimentos realizados, este modelo se consolida como el mejor.

5.3.1. Clases con peor desempeño

En la figura 5.3 se muestra un gráfico de la *precision* promedio de cada *query* de las clases en las que se obtuvo un mAP menor a 0.6, usando el mejor modelo obtenido. Se puede notar el desbalance en la cantidad de *queries* por clase y la variabilidad de resultados de una misma clase.

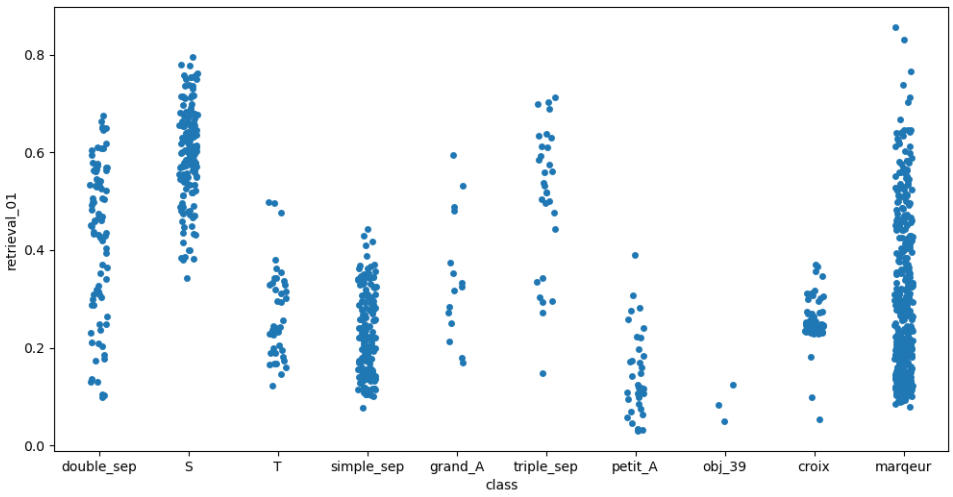

Figura 5.3: *Precision* promedio de las clases con peor desempeño.

Realizando un análisis cualitativo de las recuperaciones en estas clases, se logra observar que las *queries* más pequeñas se confunden con documentos en los que hay muchos adornos, como es el caso, por ejemplo, de la figura 5.4. En este caso, la variedad de colores y formas en los adornos de las páginas 1 y 5 de la figura aportan ruido a la recuperación, activando zonas en las que no aparece la *query*.

Otro caso recurrente en este tipo de *queries* fue que el modelo le daba una alta ponderación al fondo blanco de los documentos debido a la aparición de éste dentro de la *query*,

como se aprecia en la figura 5.5. Se puede notar en los mapas de calor cómo se activa el fondo dentro de los documentos recuperados, lo que produce fallos en la recuperación.

En otros casos, la forma de la *query* puede crear sesgos en la recuperación, como en la figura 5.6, en la cual la *query* tiene un alto contraste del centro con los lados, lo que hace activar los documentos en zonas donde ocurre este fenómeno, como es el caso de los marcos de pinturas.

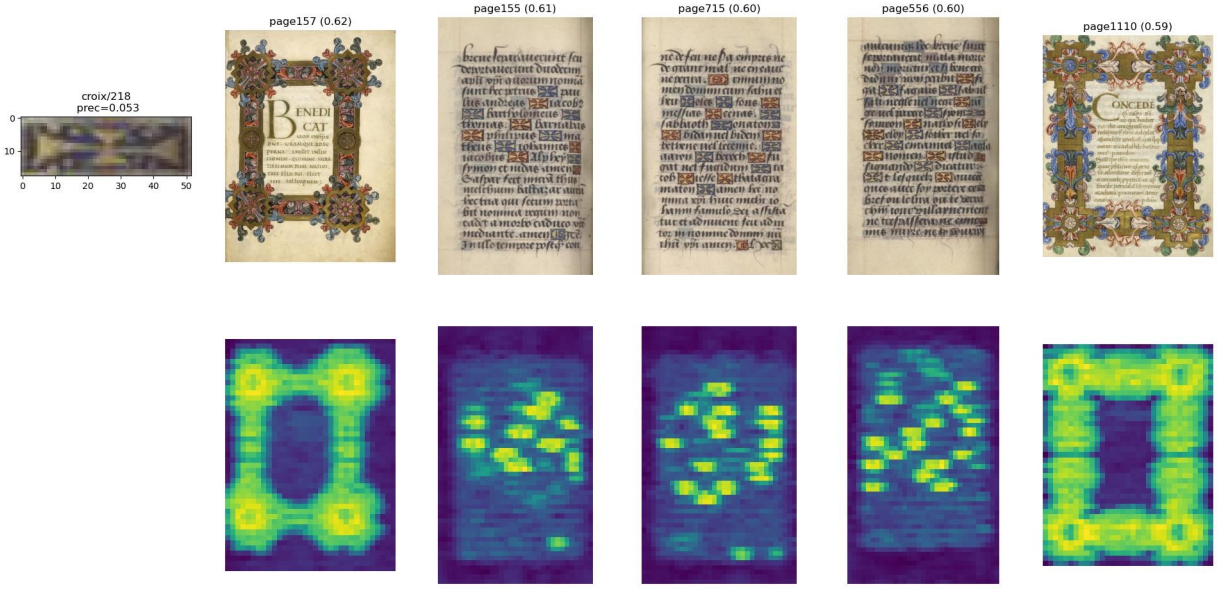

Figura 5.4: Primeros cinco documentos recuperados con su respectivo mapa de calor para un ejemplo de la clase *croix*.

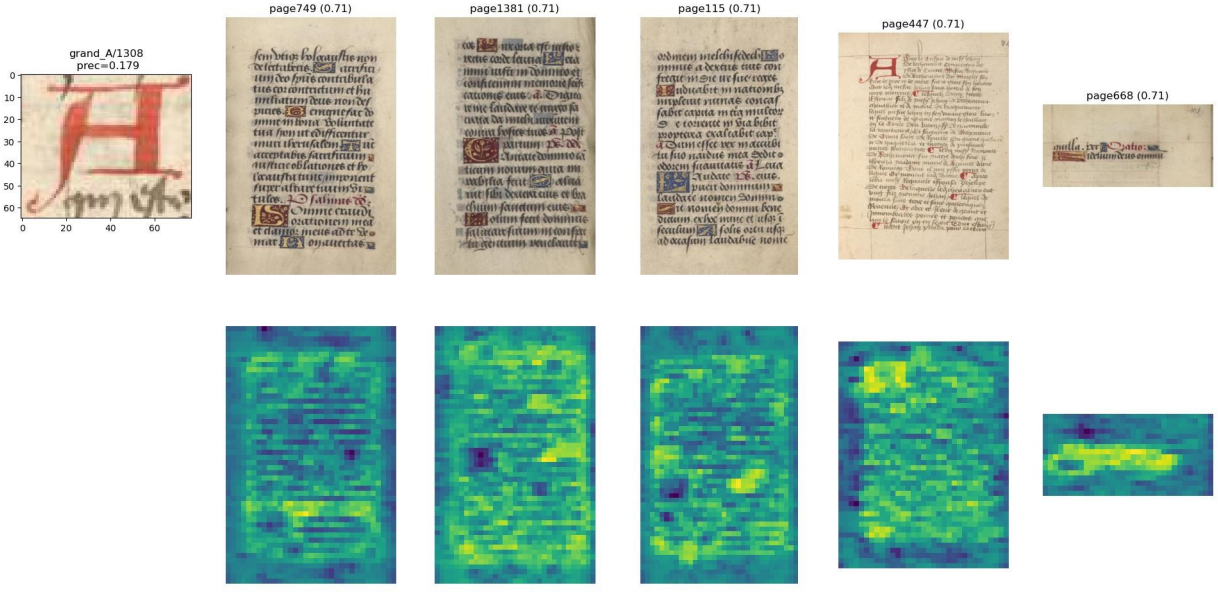

Figura 5.5: Primeros cinco documentos recuperados con su respectivo mapa de calor para un ejemplo de la clase *grand_A*.

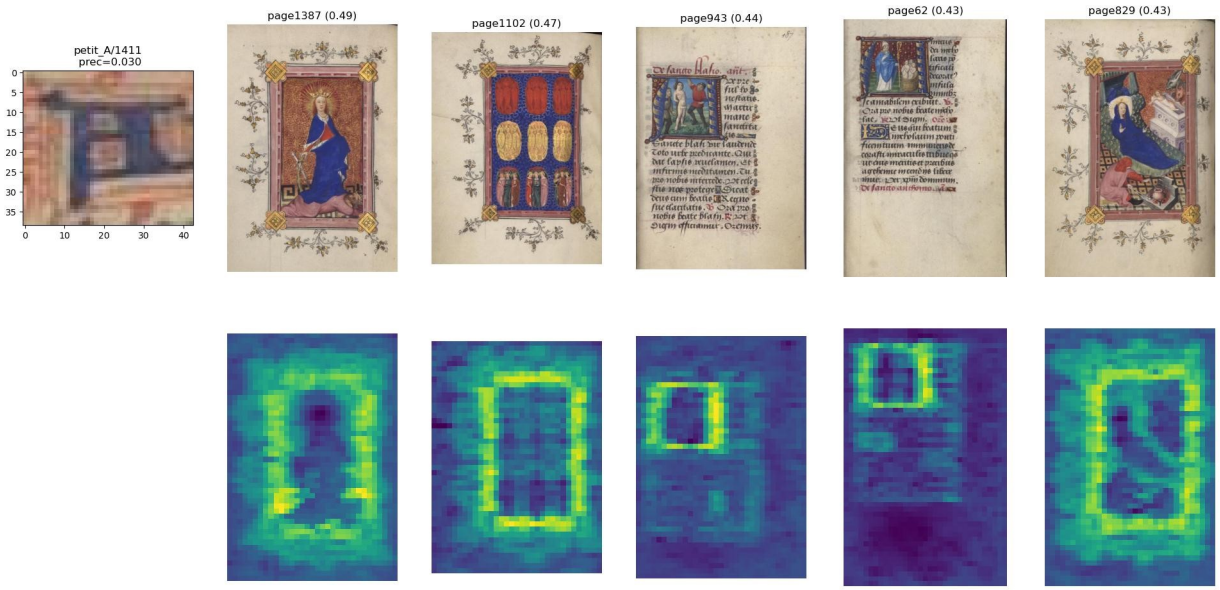

Figura 5.6: Primeros cinco documentos recuperados con su respectivo mapa de calor para un ejemplo de la clase *petit_A*.

Por otro lado, si bien la elección de una arquitectura de ViT del *encoder* de DINOv2 permite manejar campos receptivos bastante amplios, permitiendo capturar relaciones globales en los datos, esta misma característica puede limitar la capacidad del modelo para identificar patrones locales o detalles específicos, que son cruciales en la detección de ciertas características finas en imágenes. Este comportamiento podría explicar, al menos en parte, el bajo desempeño en ciertas clases donde la identificación de detalles locales es fundamental. Así, la elección de un *encoder* de *transformer* debe balancearse con técnicas que fortalezcan la detección de patrones locales, quizás a través de la integración de arquitecturas complementarias o ajustes en los mecanismos de atención para mejorar la granularidad de las características aprendidas.

5.4. Re-ranking

En la tabla 5.2, se muestran los resultados de mAP las clases con peor desempeño usando el mejor modelo, comparándolos con los resultados de las diferentes estrategias de *re-ranking* implementadas. La columna CLS-T hace referencia al uso del *class token* de DINOv2; SuperPoint se refiere al uso del descriptor extraído con el modelo del mismo nombre; SIFT n calces, max sim y mean sim se refiere al uso del descriptor SIFT ordenando los recortes según la cantidad de calces, según el máximo valor de similitud entre calces y según la similitud promedio entre calces, respectivamente; y por último, DINOv2 max sim y mean sim hace referencia al uso de los *embeddings* extraidos con el modelo DINOv2 ordenando los recortes según el máximo valor de similitud entre calces y según la similitud promedio entre calces, respectivamente.

Tabla 5.2: Resultados de mAP en recuperación de imágenes para algunas clases.

Clase	Mejor modelo	CLS-T	SuperPoint	SIFT n calces	SIFT max sim	SIFT mean sim	DINOv2 max sim	DINOv2 mean sim
S	**0.579**	0.367	0.375	0.426	0.417	0.428	0.364	0.353
T	0.279	0.166	0.159	**0.28**	0.274	0.242	0.149	0.149
croix	0.248	0.089	0.03	**0.281**	0.234	0.233	0.109	0.096
double_sep	**0.412**	0.323	0.216	0.305	0.341	0.315	0.367	0.358
grand_A	0.364	0.324	0.153	**0.491**	0.29	0.351	0.324	0.344
marqeur	**0.284**	0.154	0.097	0.12	0.118	0.121	0.12	0.102
obj_39	0.083	**0.208**	0.051	0.038	0.01	0.015	0.236	0.125
petit_A	0.165	0.159	0.102	**0.218**	0.178	0.17	0.146	0.147
simple_sep	**0.225**	0.086	0.078	0.082	0.089	0.085	0.085	0.083
triple_sep	**0.506**	0.401	0.189	0.31	0.368	0.391	0.415	0.394

Se puede apreciar de la tabla 5.2 que la mayoría de las estrategias de *re-ranking* disminuyen su mAP en comparación con el mejor modelo en las clases con peor desempeño, siendo únicamente en la clase *obj_39* donde el uso de *class token* tiene mejor desempeño y en las clases *T*, *S*, *grand_A* y *petit_A* el uso del descriptor SIFT ordenando según la cantidad de calces logra aumentar el valor de la mAP. Dado que las mejoras en estas clases fueron mínimas en comparación con el empeoramiento del desempeño en las demás clases, se descartó el uso de *re-ranking* para el *framework* final.

En la figura 5.7 se muestra un ejemplo de recuperación usando *re-ranking* con el modelo SuperPoint. Como se puede observar, este método logra recuperar en algunas ocasiones objetos que coinciden con la *query*, sin embargo, no lo hace en los primeros lugares del *ranking*, en los cuales se equivoca con otros objetos parecidos. Esto se debe a que los puntos de interés que encuentra el modelo SuperPoint se confunden con puntos muy similares en los documentos, haciendo que el calce posterior sea muy ruidoso y los *inliers* obtenidos tengan bajo error en objetos que no coinciden con la *query*. Además, debido a la variabilidad entre las *queries* de una misma clase, al obtener los *inliers* se puede tener un error considerable en la homografía calculada entre la *query* y un recorte donde aparezca. Por esta razón, para los experimentos con SIFT y DINOv2, no se utilizó RANSAC para encontrar *inliers*.

Figura 5.7: Primeros diez recortes recuperados usando el modelo SuperPoint para un ejemplo de la clase *T*.

Por otro lado, es necesario mencionar que la estrategia de *re-ranking* está condicionada a los recortes que se obtienen usando los *embeddings* de DINOv2, por lo cual, se necesita

una representación y diferenciación aún más fina entre imágenes que tienen mucha similitud entre sí. Esto significa una complejidad adicional a la tarea de *re-ranking*, la cual no se pudo resolver de forma general para todas las clases con los métodos desarrollados.

5.5. UMAP

Los resultados del uso de UMAP en comparación con el mejor modelo se pueden ver en la tabla 5.3. Las columnas UMAP Big hacen referencia al uso de UMAP ajustado a los *embeddings* de los *targets* con mayor similitud a las *queries* grandes; y UMAP queries se refiere al uso de UMAP ajustado con los *embeddings* de las *queries* con tamaño de salida de 256.

Tabla 5.3: Resultados de mAP en recuperación de imágenes usando UMAP.

Tamaño	Forma	Mejor modelo	UMAP Big 256	UMAP Big 128	UMAP Big 64	UMAP Big 32	UMAP queries
Grande	Cuadrada	**0.815**	0.346	0.369	0.345	0.355	0.650
	No cuadrada	**0.818**	0.327	0.243	0.199	0.277	0.458
Pequeña	Cuadrada	**0.739**	0.329	0.299	0.490	0.277	0.552
	No cuadrada	**0.341**	0.198	0.194	0.195	0.184	0.245

Los resultados de la tabla 5.3 muestran que los experimentos realizados con UMAP disminuyeron su mAP en todos los casos, lo que indica una pérdida de información discriminativa importante en los *embeddings* extraídos con DINOv2 al disminuir su dimensión. Sin embargo, esto puede intentar mejorarse realizando un mejor ajuste de parámetros y usando un conjunto de entrada más representativo de todas las *queries*.

5.6. Comparación con el estado del arte

Los resultados de mAP en la tarea de recuperación de imágenes del mejor modelo desarrollado en comparación con el estado del arte se muestra en la tabla 5.4. Además, se muestra en la tabla 5.5 los resultados del mejor modelo y el modelo de correlación sin bordes comparados con el estado del arte, siguiendo el protocolo de evaluación presentado en [23].

De la tabla 5.4, se aprecia que el modelo desarrollado, si bien no logra superar el trabajo de En [24] en la tarea de recuperación de imágenes, logra acercarse bastante, sobrepasando los resultados del trabajo de Wiggers [25] en esta tarea. Además, la tabla 5.5 muestra que el modelo de correlación quitando los bordes supera al estado del arte en las *queries* grandes cuadradas, llegando a un mAP de 0.93. Además, el mejor modelo también logra superar los resultados de En [24] y Úbeda [23] en las *queries* grandes no cuadradas, logrando un mAP de 0.818.

Esto demuestra que DINOv2 logra producir características visuales discriminativas y útiles en documentos de patrimonio cultural, especialmente en objetos en los que se tiene más información, como serían las *queries* grandes, sin necesidad de ser entrenado con este tipo de imágenes.

Tabla 5.4: Comparación de resultados del mejor modelo con estado del arte.

Método	mAP Recuperación
Este modelo	0.4114
En et Al. [24]	**0.580**
Úbeda et Al. [23]	0.505
Wiggers et Al. [25]	0.386
Dias et Al. [26]	0.532

Tabla 5.5: Comparación de resultados con el estado del arte según protocolo de evaluación presentado en [23].

Tamaño	Forma	mAP Recuperación			
		En et Al. [24]	Úbeda et Al. [23]	**Correl. sin bordes**	**Mejor modelo**
Grande	Cuadrada	0.881	0.749	**0.930**	0.815
	No cuadrada	0.701	0.660	0.792	**0.818**
Pequeña	Cuadrada	**0.801**	0.742	0.779	0.739
	No cuadrada	**0.535**	0.459	0.214	0.341

Capítulo 6

Conclusiones y Trabajo futuro

El presente trabajo ha abordado el desafío de la detección *one-shot* en documentos históricos mediante la implementación de un *framework* basado en la extracción de características y comparación de *embeddings* usando el modelo auto-supervisado DINOv2. A lo largo de este estudio, se han logrado cumplir los objetivos planteados inicialmente. Primero, se desarrolló un método eficaz para eliminar los bordes blancos de los documentos, logrando mantener la información útil en la mayoría de los casos. Aunque en algunos escenarios no se logró eliminar todos los bordes debido a las líneas en la página o imágenes traslúcidas, el método no eliminó partes importantes, lo que sugiere que no debería afectar negativamente el desempeño en la detección o recuperación de imágenes.

En cuanto a la extracción de características y la comparación de *embeddings*, utilizando DINOv2 se logró una extracción de características efectiva. Los mapas de calor generados a partir de la comparación de *embeddings* demostraron la capacidad del modelo para identificar zonas relevantes en las imágenes, aunque se observaron desafíos con las *queries* pequeñas y las que incluían fondos. Los experimentos realizados en la tarea de recuperación de imágenes mostraron que el *framework* desarrollado se acercó significativamente al estado del arte, logrando superar los resultados de algunos métodos anteriores en *queries* grandes. Sin embargo, sigue teniendo un peor desempeño en las *queries* pequeñas no cuadradas, dada la poca información que se puede extraer de éstas.

Por otro lado, los métodos de *re-ranking* desarrollados no ayudaron a mejorar las métricas en las clases con peor desempeño, debido a la complejidad de diferenciar los recortes extraídos de los documentos. Además, se observó una disminución en el desempeño cuando se utilizó UMAP para reducir la dimensionalidad de los *embeddings*. Por esta razón, futuros trabajos deberían centrarse en ajustar mejor los parámetros de UMAP y seleccionar conjuntos de entrada más representativos que puedan preservar mejor la información discriminativa de los *embeddings*.

Otra posible mejora, dado el bajo rendimiento en las *queries* pequeñas, sería el *fine-tuning* de DINOv2, entrenándolo de forma auto-supervisada con los documentos del *dataset* DocExplore. De esta forma, se podrían extraer *embeddings* más representativos de este tipo de *queries* y mejorar el desempeño en la tarea de recuperación. Además, se podría investigar el impacto de utilizar otros modelos auto-supervisados y comparar sus resultados con DINOv2.

Finalmente, en un futuro se debería evaluar el *framework* propuesto en otros conjuntos de datos históricos, lo cual podría proporcionar una validación más amplia de su eficacia y su capacidad para generalizar, a la vez que podría revelar nuevas áreas de mejora específicas para diferentes tipos de documentos.

Bibliografía

[1] Krizhevsky, A., Sutskever, I., y Hinton, G. E., "Imagenet classification with deep convolutional neural networks", en Advances in Neural Information Processing Systems (Pereira, F., Burges, C., Bottou, L., y Weinberger, K., eds.), vol. 25, Curran Associates, Inc., 2012, https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[2] Simonyan, K. y Zisserman, A., "Very deep convolutional networks for large-scale image recognition", en 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (Bengio, Y. y LeCun, Y., eds.), 2015, http://arxiv.org/abs/1409.1556.

[3] Nash, W. T., Drummond, T., y Birbilis, N., "A review of deep learning in the study of materials degradation", npj Materials Degradation, vol. 2, pp. 1–12, 2018, https://api.semanticscholar.org/CorpusID:70275328.

[4] He, K., Zhang, X., Ren, S., y Sun, J., "Deep residual learning for image recognition", CoRR, vol. abs/1512.03385, 2015, http://arxiv.org/abs/1512.03385.

[5] Wu, X., Sahoo, D., y Hoi, S. C. H., "Recent advances in deep learning for object detection", CoRR, vol. abs/1908.03673, 2019, http://arxiv.org/abs/1908.03673.

[6] Girshick, R. B., Donahue, J., Darrell, T., y Malik, J., "Rich feature hierarchies for accurate object detection and semantic segmentation", CoRR, vol. abs/1311.2524, 2013, http://arxiv.org/abs/1311.2524.

[7] Girshick, R. B., "Fast R-CNN", CoRR, vol. abs/1504.08083, 2015, http://arxiv.org/abs/1504.08083.

[8] Ren, S., He, K., Girshick, R. B., y Sun, J., "Faster R-CNN: towards real-time object detection with region proposal networks", CoRR, vol. abs/1506.01497, 2015, http://arxiv.org/abs/1506.01497.

[9] Lin, T., Goyal, P., Girshick, R. B., He, K., y Dollár, P., "Focal loss for dense object detection", CoRR, vol. abs/1708.02002, 2017, http://arxiv.org/abs/1708.02002.

[10] Redmon, J., Divvala, S. K., Girshick, R. B., y Farhadi, A., "You only look once: Unified, real-time object detection", CoRR, vol. abs/1506.02640, 2015, http://arxiv.org/abs/1506.02640.

[11] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., y Polosukhin, I., "Attention is all you need", CoRR, vol. abs/1706.03762, 2017, http://arxiv.org/abs/1706.03762.

[12] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., y Houlsby, N., "An

image is worth 16x16 words: Transformers for image recognition at scale", CoRR, vol. abs/2010.11929, 2020, https://arxiv.org/abs/2010.11929.

[13] Zhai, X., Kolesnikov, A., Houlsby, N., y Beyer, L., "Scaling vision transformers", CoRR, vol. abs/2106.04560, 2021, https://arxiv.org/abs/2106.04560.

[14] Bao, H., Dong, L., y Wei, F., "Beit: BERT pre-training of image transformers", CoRR, vol. abs/2106.08254, 2021, https://arxiv.org/abs/2106.08254.

[15] Jing, L. y Tian, Y., "Self-supervised visual feature learning with deep neural networks: A survey", CoRR, vol. abs/1902.06162, 2019, http://arxiv.org/abs/1902.06162.

[16] Doersch, C., Gupta, A., y Efros, A. A., "Unsupervised visual representation learning by context prediction", CoRR, vol. abs/1505.05192, 2015, http://arxiv.org/abs/1505.05192.

[17] Gidaris, S., Singh, P., y Komodakis, N., "Unsupervised representation learning by predicting image rotations", CoRR, vol. abs/1803.07728, 2018, http://arxiv.org/abs/1803.07728.

[18] Chen, X. y He, K., "Exploring simple siamese representation learning", CoRR, vol. abs/2011.10566, 2020, https://arxiv.org/abs/2011.10566.

[19] Grill, J., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. Á., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., y Valko, M., "Bootstrap your own latent: A new approach to self-supervised learning", CoRR, vol. abs/2006.07733, 2020, https://arxiv.org/abs/2006.07733.

[20] He, K., Fan, H., Wu, Y., Xie, S., y Girshick, R. B., "Momentum contrast for unsupervised visual representation learning", CoRR, vol. abs/1911.05722, 2019, http://arxiv.org/abs/1911.05722.

[21] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., y Joulin, A., "Emerging properties in self-supervised vision transformers", CoRR, vol. abs/2104.14294, 2021, https://arxiv.org/abs/2104.14294.

[22] Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.-Y., Li, S.-W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., y Bojanowski, P., "Dinov2: Learning robust visual features without supervision", 2024.

[23] Úbeda, I., Saavedra, J. M., Nicolas, S., Petitjean, C., y Heutte, L., "Improving pattern spotting in historical documents using feature pyramid networks", Pattern Recognition Letters, vol. 131, pp. 398–404, 2020, doi:https://doi.org/10.1016/j.patrec.2020.02.002.

[24] En, S., Petitjean, C., Nicolas, S., y Heutte, L., "A scalable pattern spotting system for historical documents", Pattern Recognition, vol. 54, pp. 149–161, 2016, doi:https://doi.org/10.1016/j.patcog.2016.01.014.

[25] Wiggers, K. L., de Souza Britto Junior, A., Koerich, A. L., Heutte, L., y de Oliveira, L. E. S., "Deep learning approaches for image retrieval and pattern spotting in ancient documents", CoRR, vol. abs/1907.09404, 2019, http://arxiv.org/abs/1907.09404.

[26] Da S. Dias, C., De S. Britto, A., Barddal, J. P., Heutte, L., y Koerich, A. L., "Pattern spotting and image retrieval in historical documents using deep hashing", en 2022 IEEE

International Conference on Systems, Man, and Cybernetics (SMC), pp. 2869–2875, 2022, doi:10.1109/SMC53654.2022.9945070.

[27] McInnes, L., Healy, J., y Melville, J., "Umap: Uniform manifold approximation and projection for dimension reduction", 2020.

[28] En, S., Nicolas, S., Petitjean, C., Jurie, F., y Heutte, L., "New public dataset for spotting patterns in medieval document images", Journal of Electronic Imaging, vol. 26, no. 1, p. 011010, 2016, doi:10.1117/1.JEI.26.1.011010.

[29] Lowe, D. G., "Distinctive image features from scale-invariant keypoints", International Journal of Computer Vision, vol. 60, pp. 91–110, 2004, doi:10.1023/B:VISI.0000029664.99615.94.

[30] DeTone, D., Malisiewicz, T., y Rabinovich, A., "Superpoint: Self-supervised interest point detection and description", 2018.