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

DETECCIÓN ONE-SHOT DE PATRONES BIDIMENSIONALES EN IMÁGENES MEDIANTE CORRELACIÓN DE MAPAS DE CARACTERÍSTICAS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

JEAN FRANCO CHERUBINI FOUILLOUX

PROFESOR GUÍA:
JOSÉ SAAVEDRA RONDO

MIEMBROS DE LA COMISIÓN:
FRANCISCO RIVERA SERRANO
IVÁN SIPIRAN MENDOZA

SANTIAGO DE CHILE
2021

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: JEAN FRANCO CHERUBINI FOUILLOUX
FECHA: 2021
PROF. GUÍA: JOSÉ SAAVEDRA RONDO

DETECCIÓN ONE-SHOT DE PATRONES BIDIMENSIONALES EN IMÁGENES MEDIANTE CORRELACIÓN DE MAPAS DE CARACTERÍSTICAS

En el contexto de la detección de imágenes con *Deep Learning* actual, el paradigma de entrenamiento está cambiando. Esto se debe a que los logros recientes en detección, si bien impresionantes, han mantenido el requerimiento de grandes conjuntos de imágenes etiquetadas para lograr su cometido. Esto implica un gasto de tiempo y recursos humanos. Es por este motivo que se ha empezado a desarrollar el campo del *Few-Shot Learning* y, en particular, el *One-Shot Learning*. En ellos se busca encontrar mecanismos de aprendizaje utilizando muy pocos datos.

A su vez, la detección de patrones bidimensionales en imágenes es un problema interesante debido a que corresponde a una situación intermedia entre el *2-D* y el *3-D*, debido a la naturaleza de dos dimensiones de este tipo de patrón, pero que pueden también encontrar distorsiones propias del mundo tridimensional, debido a la forma del plano en que se encuentran.

El problema a abordar en el presente documento busca tomar inspiración en el campo del *One-Shot Learning* para investigar la eficacia de utilizar características a partir de modelos de *deep learning* en la detección de patrones bidimensionales.

Para esto se implementó una metodología basada en la extracción de características de modelos pre-entrenados en datos no relacionados con los patrones a detectar, la búsqueda de *queries* en imágenes a través de la correlación entre características. Esta metodología se basó en lo estudiado por un grupo investigador paralelo y se añadieron cambios y mejoras.

En primera instancia, se realizó una investigación en un problema simplificado de detección de patrones bidimensionales, que corresponde a símbolos e imágenes en documentos históricos, en un set llamado *DocExplore*. En este, se consiguieron resultados muy positivos, superando el estado del arte y acercándose a los resultados de un grupo del investigación con el que se trabaja en paralelo en esta investigación.

En segunda instancia, se probó la eficacia de la estrategia utilizada en la tarea inicial sobre el dataset *Flickrlogos 47*. Esta tarea es más compleja que el problema anterior debido a que en lugar de páginas y documentos, los logos se encuentran en ambientes realistas. Con estas pruebas se llegó a conclusiones interesantes, sin embargo, la falta de estado del arte para esta tarea en específico no permite la comparación directa con otros resultados. Se concluye el documento con resultados prometedores para el crecimiento de la investigación en esta tarea de detección.

A mi nonno Vittorio.

Agradecimientos

Quiero agradecer primero que nada a mi familia; mis padres Leonardo y Jeannete que me han apoyado en absolutamente todo y han hecho de pilar en cualquier cosa que me he propuesto. Y a quienes agradezco mi motivación por el aprendizaje y el sentido de responsabilidad que me inculcaron desde pequeño, por fomentar mi curiosidad y tenerme paciencia en todo lo que hago.

Quisiera darle las gracias a mi hermano Stefano, por ser mi socio y mi amigo, por las buenas conversaciones y los momentos de distensión que tanto uno necesita a veces. A mi abuela Idilia que me ha regaloneado desde que tengo memoria, con un postre rico y su inconmensurable preocupación por mi.

A mis nonnos Vittorio y Aurora, a quienes extraño cada día más pero que me enseñaron que con sacrificio, astucia y sobre todo cariño se pueden alcanzar grandes cosas, pero que por sobre todo eso está la familia.

A mi tía regalona, Giovanna que me ha consentido desde el día en que nací, y a mis demás tíos, tías y primos, con quienes he tenido la suerte de crecer de forma cercana y que han confiado siempre en mí y con quienes tantas veces me he reído en las juntas y reuniones familiares.

A mis amigos de la infancia, Francisca e Ignacio, con quienes ya he formado vínculos de más de 20 años. Gracias por siempre estar ahí, por las risas, la preocupación, los consejos y el apoyo. Son fundamentales en mi vida y siempre lo serán.

A mis amigos del colegio, Alonso, Cristóbal y Bastián, personas de una calidad enorme, y con quienes me reí tantas veces por sus shows en el colegio. Por los partidos en los recreos y las tardes de estudio, no se imaginan lo agradecido que estoy.

A mis amigas de la vida, a quienes no imagino fuera de ella, Coté, Consuelo y Paola. Gracias por las risas y los buenos ratos. Las quiero muchísimo.

A mis ratitas y electrobigotitos, Claudio, Anibal, Diego C., Diego B., Valentina, Tomás, Malú, Fernanda, Camila, Gabriel e Iván. Con ustedes viví mi el proceso universitario, el cambio más fuerte de mi vida y con quienes crecí y aprendí tanto como persona. No pude haber pedido una mejor experiencia y mejores personas para acompañarme. Gracias por los almuerzos, los carretes, los paseos, las pizzas, los partidos y las tardes de estudio. Tienen y tendrán un lugar en mi corazón por el resto de mi vida. Quiero especialmente agradecer a

Claudio, quien fue mi compañero en tantos cursos y trabajos, gracias por las conversaciones, la paciencia, el apoyo y los memes vistos y creados en el proceso, hiciste todo proyecto en el que trabajamos fuese algo entretenido e interesante.

Gracias al profesor José que me dio la oportunidad de participar en este trabajo, y me brindó orientación y guía.

Finalmente, gracias a los jueguitos de computadora, son lo mejor que me pasó.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación y formulación del problema

Uno de los objetivos mas antiguos del campo de visión computacional es el reconocimiento de objetos. Mucha investigación y recursos han sido dedicados a resolver esta tarea en general, y a partir de ello se han explorado múltiples subproblemas que, sin embargo, aun no se encuentran resueltas totalmente. En este contexto, el reconocimiento automático de patrones bidimensionales (o planos) en el mundo real se transforma en una tarea deseable, pues si bien corresponde a un tipo determinado de objeto (tiene solo una cara frontal y no posee volumen, como por ejemplo, el logo de una marca), para su detección en imágenes se debe lidiar con posibles transformaciones afines, como de rotación o perspectiva. A su vez su detección es menos compleja que la de objetos que poseen múltiples caras debido a su falta de profundidad. Por este motivo su detección resulta un punto medio de investigación interesante, pues existen problemas que pueden resolverse bajo este contexto. Se debe considerar también que al realizar la detección de patrones bidimensionales en ambientes realistas ("in the wild"), se debe lidiar con los problemas que los diferentes fondos de la imagen le traen a la detección.

En el reconocimiento de objetos tradicional se han utilizado técnicas que ya son consideradas como clásicas, cuya formulación consiste comúnmente en dos etapas. En una primera etapa, se calcula algún tipo de descriptor de la imagen, basándose en elementos de su composición como bordes y/o colores, y luego, se realiza una segunda etapa en la cual se realizan las detecciones utilizando los descriptores. Comúnmente, estas detecciones involucran técnicas de *machine learning* al momento de utilizar los descriptores. Ejemplos de estos son los histogramas de gradientes o color, deformable part models (DPM) y "bag of visual-words" de descriptores de bordes, entre otros. Sin embargo, y el motivo por el cual estos métodos son considerados clásicos, es que el paradigma cambió con la aparición del *Deep Learning*.

El *Deep Learning* es una herramienta utilizada en el aprendizaje de maquinas, la cual se centra en el entrenamiento de muchas capas convolucionales bidimensionales. En otras palabras, múltiples filtros de imágenes que son aprendidos mediante datos e imágenes disponibles. Este tipo de sistema es capaz de aprender de forma jerarquizada y, a través de sus filtros, otorga la capacidad de obtener características desde las imágenes de forma automática, eli-

minando la necesidad de hacerlo manualmente. Esto es relevante pues los sistemas de *deep learning* destacan al momento de detectar patrones, poder representarlos numéricamente y hacerlos comparables.

La extracción de características hace innecesario el cálculo previo de descriptores que se utiliza en los métodos clásicos y mejora considerablemente el desempeño, pero esto hace necesaria la existencia de grandes cantidades de ejemplos para ajustar un modelo en comparación con los métodos clásicos, anteriormente mencionados. Por ejemplo, si se quisiera detectar un objeto particular con detectores *deep* pertenecientes al estado del arte (como YOLO o R-CNN), se necesitarían del orden de cientos de fotos para entrenar un modelo y lograr detectar el objeto. Esta es la principal ventaja que tienen actualmente los métodos más tradicionales sobre los métodos *deep*, en conjunto con la mayor rapidez de detección que estos suelen tener.

Debido a que los sistemas *deep* necesitan una gran cantidad de ejemplos para entrenar la detección de un nuevo objeto, que no ha sido anteriormente visto, se motivan formas alternativas de manejo de esta situación. Entre ellas, aparece el campo del *few-shot learning*, o en otras palabras, el aprendizaje en redes de *deep learning* que busca generalizar y lograr detecciones mediante pocas muestras de consulta, las cuales nunca han sido antes vistas por el sistema.Por ejemplo, si ya se sabe reconocer animales en general, si se consulta por un animal en particular y se tienen pocas fotos de éste, dicho animal debiese poder ser reconocido entre otros objetos, en particular entre otros animales, y además, en diferentes entornos [2].

El enfoque *few shot* permite ahorrar recursos y tiempo al intentar detectar un nuevo objeto perteneciente a una clase ya generalizada, pues se necesitarían pocas imágenes para caracterizarlo. En particular, cuando el enfoque esta orientado a la realización de detecciones con una sola imagen de consulta, a esto se le llama *one-shot learning*. Es posible entonces imaginar como la tarea de detectar patrones bidimensionales bajo este enfoque es bastante posible, pues un objeto de una cara es un tanto más simple de describir que otro con múltiples, pues sus características son más sencillas en general. Adicionalmente a esto, las rotaciones y deformaciones no le afectarán a un patrón plano de la misma forma que a un objeto tridimensional, debido a su falta de profundidad.

Cuando se mira, además, el desempeño, es claro que los sistemas de detección *deep* superan en precisión a los métodos clásicos, pero también es cierto que su velocidad de funcionamiento en tiempo real es en general menor. Es por este motivo que, constantemente, se buscan formas de agilizar el procesamiento de las consultas para poder utilizar los sistemas *deep* en aplicaciones que requieren respuestas mas instantáneas. Para lograrlo, es importante estudiar las posibles utilidades de las características extraídas por los modelos *deep*, pues puede ser viable mejorar su velocidad de funcionamiento mediante la correcta utilización de estas.

Un sistema que logre una buena detección y reconocimiento de objetos en ambientes reales, que incorpore un enfoque de *one shot learning* para objetos en general, es un problema bastante complejo, pero es posible acotarlo y resolver un subproblema. Como ya se mencionó antes, la detección de patrones bidimensionales en ambientes reales es un problema relevante, por lo que el enfoque de este trabajo se concentrará en la detección de este tipo de patrones.

La resolución del problema de detección de este tipo de patrones es útil pues tiene muchas

aplicaciones inmediatas como por ejemplo el caso particular de la búsqueda de logos: medición de *product placement* en programas de televisión y películas, la identificación en tiempo real de productos en estantes de tiendas, chequeo de compras de usuarios en cajas de autoservicio de supermercados con el propósito de evitar robo por intercambio de etiqueta, uso no autorizado de marcas y en particular, la búsqueda automática de elementos y productos en planogramas[1], ya que en general en este tipo de vista los objetos pueden interpretarse como patrones bidimensionales, en la mayoría de los casos. A su vez, la detección de patrones planos en documentos es una clara utilidad de este tipo de detección al buscar, por ejemplo, timbres, símbolos o firmas. Además de todo esto, es posible en teoría extrapolar este tipo de sistema a otras categorías de objetos que compartan características con este problema, algo que se conoce como *transfer learning*.

1.2. Objetivos

1.2.1. Objetivo general

El objetivo general de este trabajo es desarrollar un *framework* capaz de detectar e identificar patrones bidimensionales en imágenes mediante características adquiridas desde arquitecturas de *deep learning* utilizando una única imagen de consulta.

1.2.2. Objetivos específicos

1. Comprobar la eficacia del uso de la correlación entre características de imagen y *query* para detección.
2. Hacer pruebas en diversas redes para encontrar la de mejor desempeño en detección de patrones bidimensionales.
3. Definir una arquitectura basada en la correlación para realizar detección de patrones bidimensionales.
4. Evaluar el desempeño de detección en set de datos de Logos.
5. Evaluar el desempeño de detección en DocExoplore.

[1]Plano frontal de una góndola de tienda o supermercado

Capítulo 2

Marco teórico

Este trabajo se encuadra en el campo de la visión computacional, específicamente, en detección de objetos. Para comprender lo realizado, es necesario entender algunos conceptos relacionados con la detección. Además, se revisarán tanto métodos clásicos como métodos actuales de detección, debido a que la propuesta de este trabajo involucra conceptos de ambas tendencias.

2.1. Detección en imágenes

La detección de objetos en el campo de la visión computacional corresponde a identificar un objeto en específico en una imagen o video, encontrando su posición en el espacio de la imagen. Comúnmente, se utilizan los llamados *bounding boxes* para representar una detección en una imagen, describiendo en éste, además, a que tipo de objeto corresponde. Ejemplos de detección de objetos se pueden ver en la Figura 2.1. Además, existe otro enfoque de detección que consiste en clasificar de forma individual la pertenencia de los pixeles que componen una imagen a un tipo de objeto determinado. Este enfoque se denomina segmentación, sin embargo, no será el foco de este trabajo, pues se centrará en detección a través de *bounding boxes*.

Como comúnmente existe confusión, es necesario diferenciar la tarea de clasificación de objetos de la de detección. La primera consiste en que, dado un objeto de una clase determinada, asignarle una etiqueta de clase. En cambio, la segunda consiste en que dada una imagen, se determine si una cierta clase de objeto se encuentra en la imagen o no y, además, si esta existe, se indica su posición en dicha imagen. Es por esto que en comparación, la detección de objetos es una tarea más desafiante que requiere de métodos más complejos para ser resuelta [4].

2.2. Métodos clásicos de detección

Los métodos más clásicos, como los presentados a continuación, consisten en calcular algún tipo de descriptor para las imágenes.

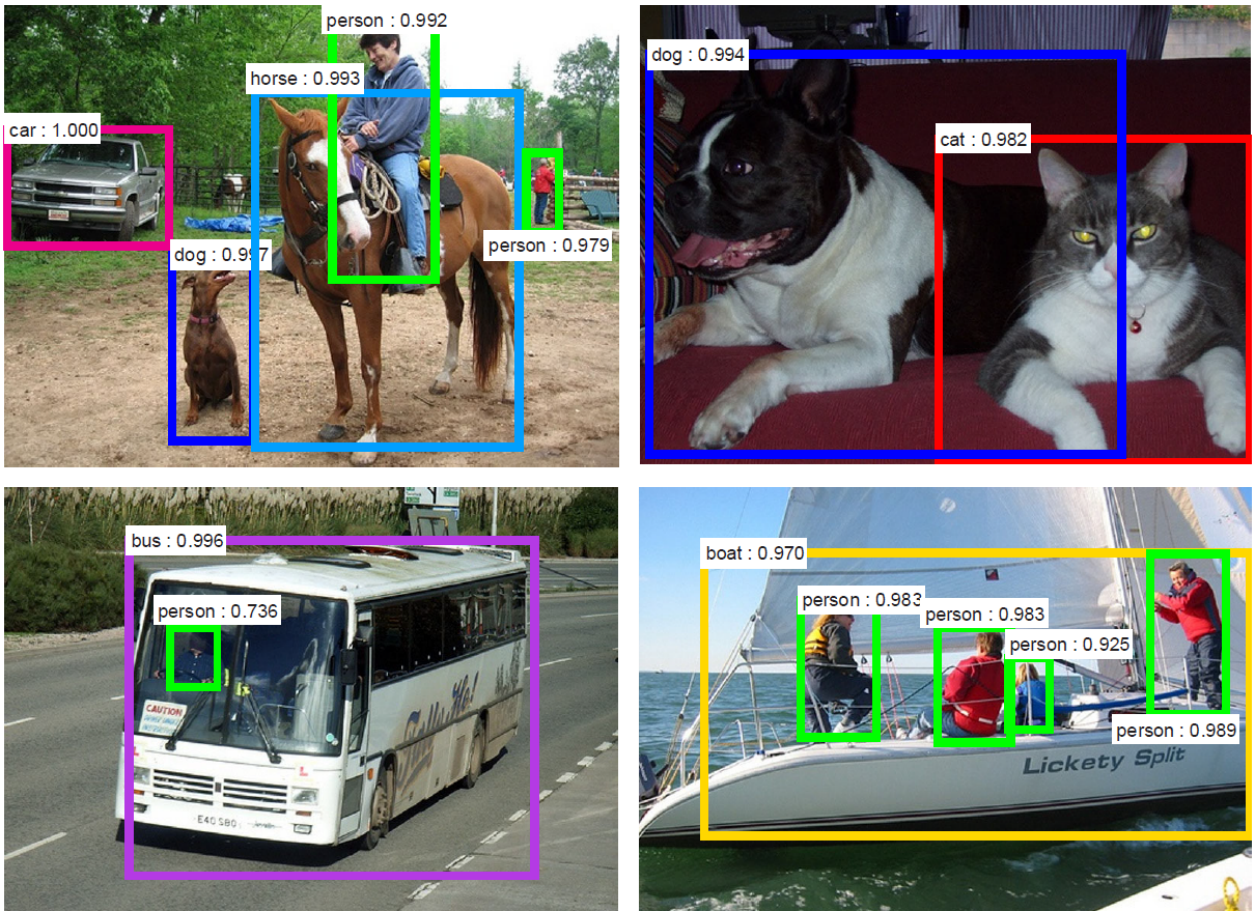


Figura 2.1: Detecciones en imágenes [5].

2.2.1. Sliding window (Ventana deslizante)

Una aproximación popular a la detección de objetos, involucra el problema de la clasificación binaria. El ejemplo más simple de este tipo de enfoque es la ventana deslizante o *sliding window*[8]. Este consiste en tomar las características que representan el objeto que se busca detectar y encuadrarlas en una ventana. Con ella, que tiene las características del objeto, se hace una clasificación sobre todas las otras ventanas pertenecientes a la imagen en la cual se quiere encontrar el objeto y se determina si corresponde o no a lo buscado. De aquí el termino clasificación binaria.

Si se encuentra una ventana en la imagen que sea lo suficientemente similar al objeto entonces esta corresponde a una detección. Es pertinente mencionar que esta clasificación se realiza en todas las posiciones, escalas y orientaciones posibles en la imagen, lo que es bastante costoso computacionalmente si se realiza directamente. Es por este motivo que se realizan ciertas modificaciones a este tipo de sistemas, como el utilizado en [6] en el cual se emplean clasificadores más complejos en cascada de forma incremental, de manera que las primeras (y más rápidas clasificaciones) puedan determinar y descartar rápidamente las ventanas que corresponden al fondo de la imagen y no a objetos, haciendo la tarea mucho mas eficiente dado que los recursos computacionales se gastan solo en clasificar regiones en donde se sospecha que hay un objeto.

2.2.2. Deformable part models (modelo de partes deformables)

Son modelos que consisten en encontrar objetos para los cuales se supone una cierta configuración de partes, lo cual lleva a un llamado *espacio de hipótesis*, el cual es bastante grande. Este espacio de hipótesis corresponde a las formas en las cuales se pueden distribuir las partes del objeto en el espacio.

Una detección con modelos de partes deformables se puede realizar considerando todas las posibles ubicaciones en la imagen donde exista una parte *raíz* y, para cada una de estas *raíces*, encontrar la mejor configuración de las partes faltantes [8]. La mejor configuración de piezas faltantes se obtiene mediante un puntaje, el cual es asignado de distintas formas, pero se relaciona con el desplazamiento, rotaciones y deformaciones de las partes que componen al objeto a detectar.

2.2.3. Histograma de gradientes

Para detectar objetos en imágenes es necesario describir los elementos dentro de la imagen, idealmente con descriptores que sean robustos ante muchos aspectos de ella, por ejemplo: iluminación, contraste y diferentes fondos. En este contexto, se descubrió que la gradiente genera un buen descriptor para muchos tipos de objetos. En particular, para detección de humanos [7].

La gradiente en el contexto de imágenes se refiere a un vector que indica en que dirección se produce un cambio en la imagen. Cada gradiente tiene una magnitud y un ángulo producido gracias a sus componentes (2 componentes en el caso de imágenes 2d).

Con las orientaciones de las gradientes, se generan los descriptores zonales denominados

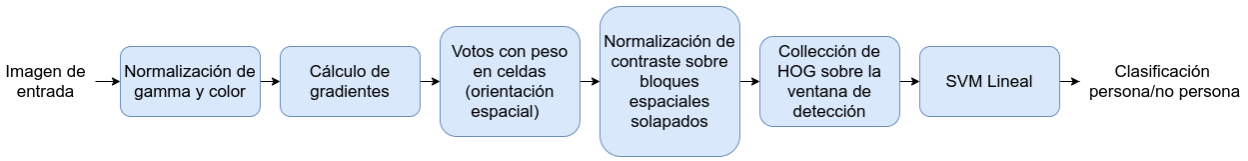


Figura 2.2: Esquema de cadena de extracción de características y detección de personas. Extrapido y traducido desde [7].

histogramas de gradiente, que consisten en generar histogramas con bines que se distribuyen según el ángulo de orientación de cada gradiente. Esto permite contabilizar qué cantidad de pixeles de la imagen tienen las orientaciones definidas por los bines. Esto se normaliza y se genera un descriptor local o general de la imagen, y es comparable con histogramas de gradiente de otra imagen.

Utilizando la comparación y a través de métricas de similitud entre el histograma de gradiente del objeto buscado con histogramas de gradiente de localidades de la imagen de búsqueda, es posible encontrar similitudes y, finalmente, detectar el objeto si es que este existe. También se pueden utilizar clasificadores como *SVM* o *Redes Neuronales* para determinar si el objeto es o no el buscado. Un esquema que resume el funcionamiento de este tipo de sistemas para el caso particular de detección de personas se puede observar en la Figura 2.2.

2.3. Métodos actuales de clasificación y reconocimiento de imágenes

2.3.1. VGG

Con la popularización de modelos convolucionales, se desarrollaron diversos sistemas capaces de clasificar imágenes mediante aprendizaje. Dentro de ellos, las redes VGG fueron pioneras en la investigación del efecto de la profundidad y densidad de la red. Esto se realizó experimentando con la adición de una mayor cantidad de filtros convolucionales. Este tipo de red utiliza filtros pequeños de 3x3 en bloques y *max pooling*. En particular, este modelo tiene diferentes versiones dependiendo de la profundidad de la experimentación. Estos modelos se pueden apreciar en la Figura 2.3.[23]. Como ejemplo, una representación de la red VGG-D (también llamada *VGG16*) se puede ver en la Figura 2.5.

Es importante mencionar que este tipo de redes tienen una cantidad considerable de parámetros entrenables, como se puede apreciar en la Figura 2.4, lo que la hace requerir muchas muestras para su entrenamiento.

2.3.2. ResNet

Este modelo consiste en una serie de bloques convolucionales que permite facilitar el entrenamiento de redes convolucionales profundas[27][28] y un paso final de clasificación. Cada bloque contiene una cierta cantidad de filtros convolucionales de forma interna. Tiene la característica de que es capaz de realizar un entrenamiento eficiente gracias a su capacidad de mantener "residuos" a lo largo de la red. Esto se realiza mediante la integración de "corto-

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figura 2.3: Estructura convolucional de los diversos modelos de la red VGG [23].

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Figura 2.4: Parámetros VGG [23].

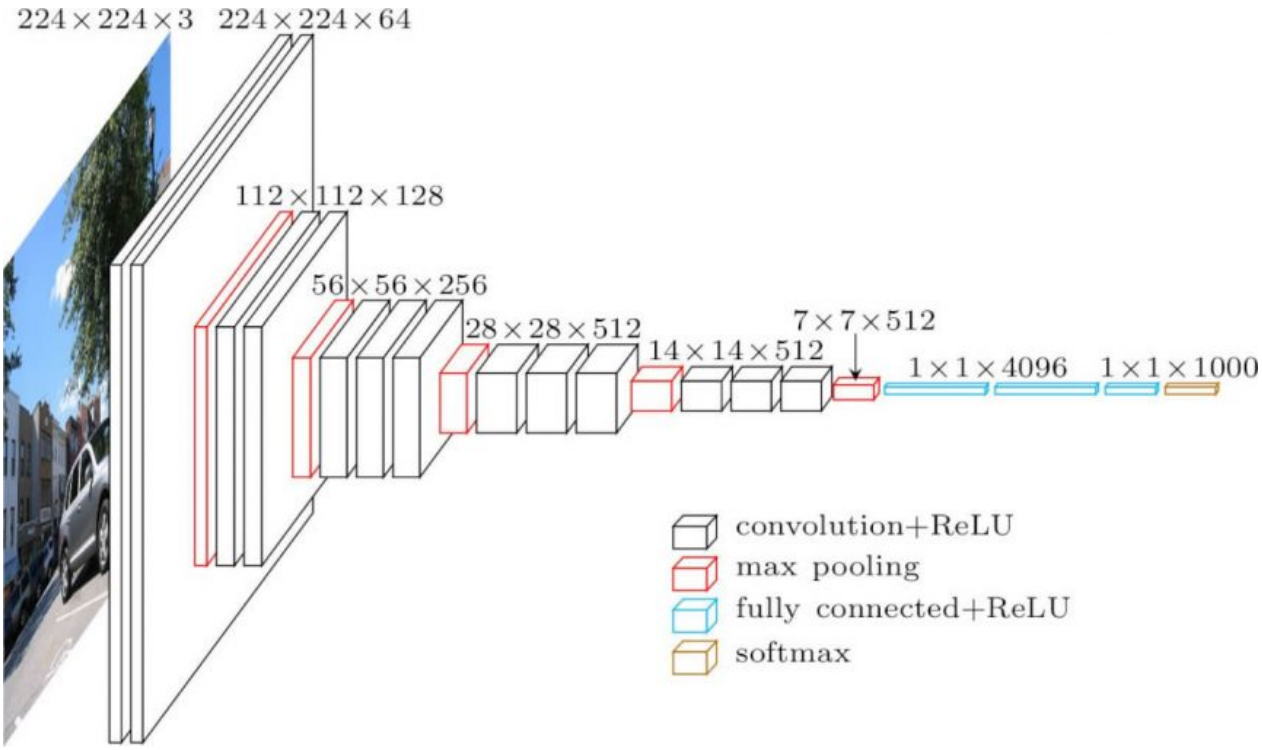


Figura 2.5: Estructura de la red VGG16 [24].

circuitos" o *atajos*, permitiendo el avance de los residuos mediante la mezcla de estos con capas que se encuentran más adelante en la red. Esto se describe en la Figura 2.6, donde se representa uno de los bloques unitarios de este modelo, los cuales tienen la posibilidad de ser concatenados uno tras otro.

Finalmente, en lo que se denomina la "cabeza" de la red se incorporan clasificadores *fully connected*, que determinarán si las características generadas por los bloques convolucionales corresponden o no a una determinada clase. Es importante mencionar que esta red es entrenada con un tamaño de imagen fijo, pues las redes *fully connected* tienen este requerimiento. A pesar de esto último, es posible eliminar dicha cabeza de la red y tomar los bloques convolucionales ya entrenados para emplearlos en otros usos, como la realización de *fine tuning* (tomar un modelo pre-entrenado en un set de imágenes, y tomarlo como punto de partida para adaptarlo a un set más específico, generalmente más pequeño), como extractores de características, como es el caso de este trabajo.

2.4. Métodos actuales de detección en imágenes

Los métodos actuales, consistentes mayoritariamente de redes convolucionales entrenadas mediante ejemplificación, fueron rechazados en su época pues si bien tenían buenos resultados, estos eran poco intuitivos y explicables. De hecho, en el año 2008 *"Un paper (...) fue rechazado por la principal conferencia de visión computacional debido a que utilizaba redes neuronales y por lo tanto, no proveía de ninguna intuición sobre como diseñar un sistema de visión. En esa época, ya mayoría de los investigadores del área de visión computacional creía que un sistema decisión necesitaba ser cuidadosamente diseñado a mano utilizando un*

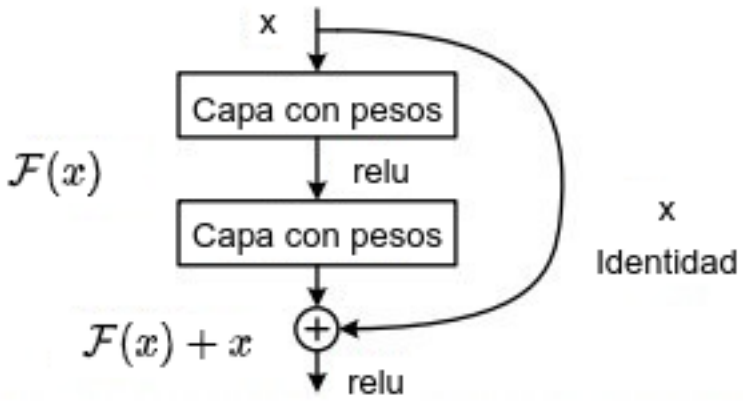


Figura 2.6: Corto circuitos en una ResNet. Extraída y traducida desde[27].

detallado entendimiento de la naturaleza de la tarea. (...) Lo que muchos investigadores de la comunidad fallaron en apreciar fue que métodos que requieren la ingeniería manual realizada por un programador que entiende el dominio no escala tan bien como métodos que reemplazan al programador con un poderoso procedimiento de aprendizaje de propósito general [9]." Esto generó un intenso debate y la revitalización de este tipo de sistemas que utilizan redes convolucionales para las tareas de clasificación y detección.

Hoy en día, este tipo de sistemas son los que presentan mejores resultados en la tarea de detección de objetos, y se centra principalmente en dos corrientes; Detección de dos etapas (*Two-stage detection*) y de una etapa (*One-stage detection*).

2.4.1. Two-stage detection

Este tipo de sistema centra su funcionamiento en dos etapas fundamentales, el cual fue popularizado por el framework de R-CNN [4, 3]. La primera es la de proposición de regiones (*region proposal*). Esta consiste en el entrenamiento de redes para lograr que estas propongan regiones candidatas a ser un objeto cualquiera de una imagen. Habitualmente, esta tarea es realizada por *region proposal networks* o RPN. La segunda etapa corresponde a clasificación. En ella, se toman las regiones propuestas por la primera etapa y se determina a que tipo de objeto pertenece la región propuesta, si es que corresponde a alguna clase o simplemente al fondo de la imagen. Utilizando esta información, se generan las detecciones de los objetos en cada imagen, tanto con su ubicación como con su clase correspondiente.

A continuación se explican algunas implementaciones de este tipo de estrategia.

R-CNN

Este modelo consta de tres módulos, el primero genera proposiciones de regiones que son independientes de la categoría del objeto. Cada proposición define un set de candidatos a detección disponibles para el detector. El segundo módulo es una gran red convolucional que extrae un vector de largo fijo de cada región de la imagen. El tercer módulo es un set de clasificadores de tipo SVM que clasifica este vector de características.[3] En este caso, los primeros dos módulos conforman la etapa de proposición de regiones y el tercer módulo la

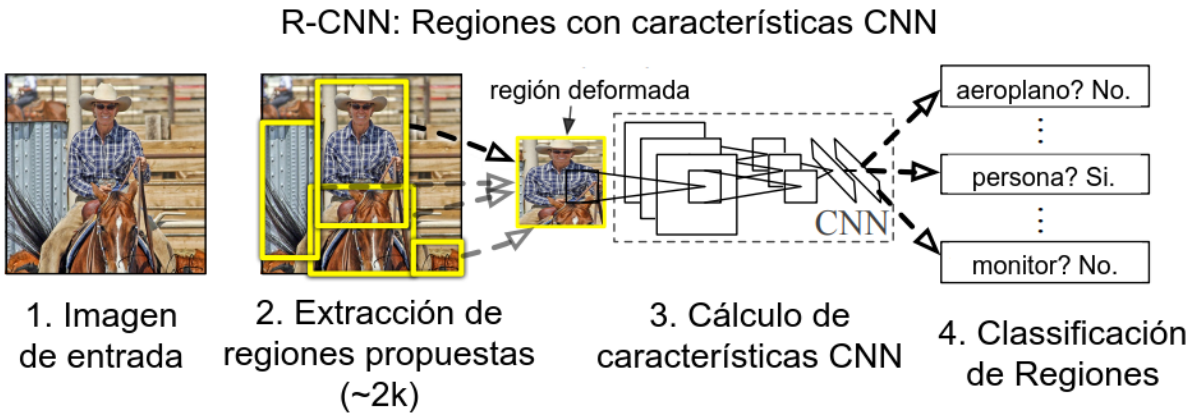


Figura 2.7: Esquema de funcionamiento de R-CNN [3]

etapa de clasificación. Este funcionamiento se expresa en el esquema de la Figura 2.7

Fast R-CNN

Consiste en una modificación del modelo revisado en la Sección 2.4.1. Las principales mejoras de las que dispone este modelo son en el entrenamiento y en la velocidad de testeo, además de mayor precisión. Esto se debe a que el modelo de R-CNN, anteriormente explicado, tiene tres inconvenientes notables [4]:

1. El entrenamiento es una *multi-stage pipeline* (Es lineal y no paralelizado).
2. El entrenamiento es costoso en memoria y tiempo.
3. La detección de objetos es lenta.

Por estos motivos, en el modelo *Fast R-CNN* se propone un nuevo algoritmo de entrenamiento que resulta en [4]:

1. Mayor calidad detección
2. Entrenamiento de una etapa, utilizando una función de pérdida multitarea
3. El entrenamiento puede actualizar todas las capas de la red
4. No se requiere espacio de disco para almacenar las características en memoria caché

La arquitectura utilizada se puede ver en la Figura 2.8. Consiste en que una imagen de entrada y múltiples regiones de interés (RoI) son la entrada para una red *fully convolutional*. Cada región de interés pasa por un proceso de *pooling* el cual es mapeado a un vector de características por una red *fully connected*. La red tiene dos vectores de salida, los cuales representan las probabilidades de la función *softmax* (que establece a que clase pertenece el objeto) y un vector de regresión del offset de *bounding box* por cada clase. La arquitectura se entrena con una función de pérdida multi-tarea.

Faster R-CNN

Si bien el modelo de *Fast R-CNN* es más rápido y tiene mejor funcionamiento que *R-CNN* por si solo, evidenció un cuello de botella en la etapa de proposición de regiones. Es por este

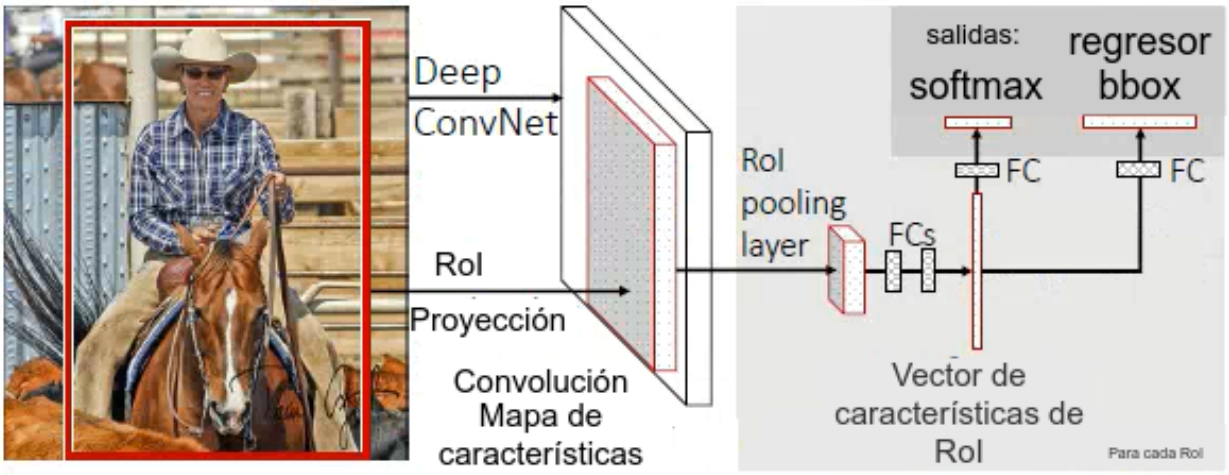


Figura 2.8: Esquema de funcionamiento Fast R-CNN. Traducida desde [4].

motivo que *Faster R-CNN* introduce una red denominada *Region Proposal Network*. Estas comparten características convolucionales con la red de detección, habilitando así proposiciones de regiones sin costo computacional extra. Esta red es una red *fully convolutional* que simultáneamente predice las delimitaciones espaciales de los objetos, y además, entrega un score en cada posición [5]. Un esquema de funcionamiento con la integración de la RPN se puede observar en la Figura 2.9

Retinanet

Es una red de detección que incorpora una función de pérdida residual (*focal loss*) para su entrenamiento y que incorpora una red de características piramidal (*Feature Pyramid Network*), que es capaz de crear una pirámide de características multiescala. Cada nivel de ésta puede ser usado para la detección de objetos de diferente escala.[11].

En primer lugar, la imagen tiene un paso por 4 bloques de capas convolucionales pertenecientes al *backbone*. Esta estructura se complementas tomando una red de características piramidal (FPN) que se encarga de extraer las características multiescala. Las características se calculan en 5 niveles, $P3$, $P4$, $P5$, $P6$ y $P7$, donde cada una reduce las dimensiones de las características en ancho y en alto en un factor de 2^{i}, con i el sufijo del nivel correspondiente p_{i}. Luego, se generan *anchors*(regiones donde se propone la existencia de objetos) en cada una de los niveles de la FPN y luego de predicen las probabilidades de la presencia de objetos en dichos *anchors*. Finalmente, se genera una regresión para definir los *bounding boxes* definitivos para los objetos encontrados. Una vista de la arquitectura puede ser encontrada en la Figura 2.10.

DeepLab

Es un modelo de segmentación que también es capaz de calcular características en diferentes escalas a través de convolución en formato diluido (*atrous convolution*), las que permiten que exista una menor reducción de dimensiones al extraer características. Este modelo también utiliza un backbone además de las convoluciones ya mencionadas, pero este se ubica

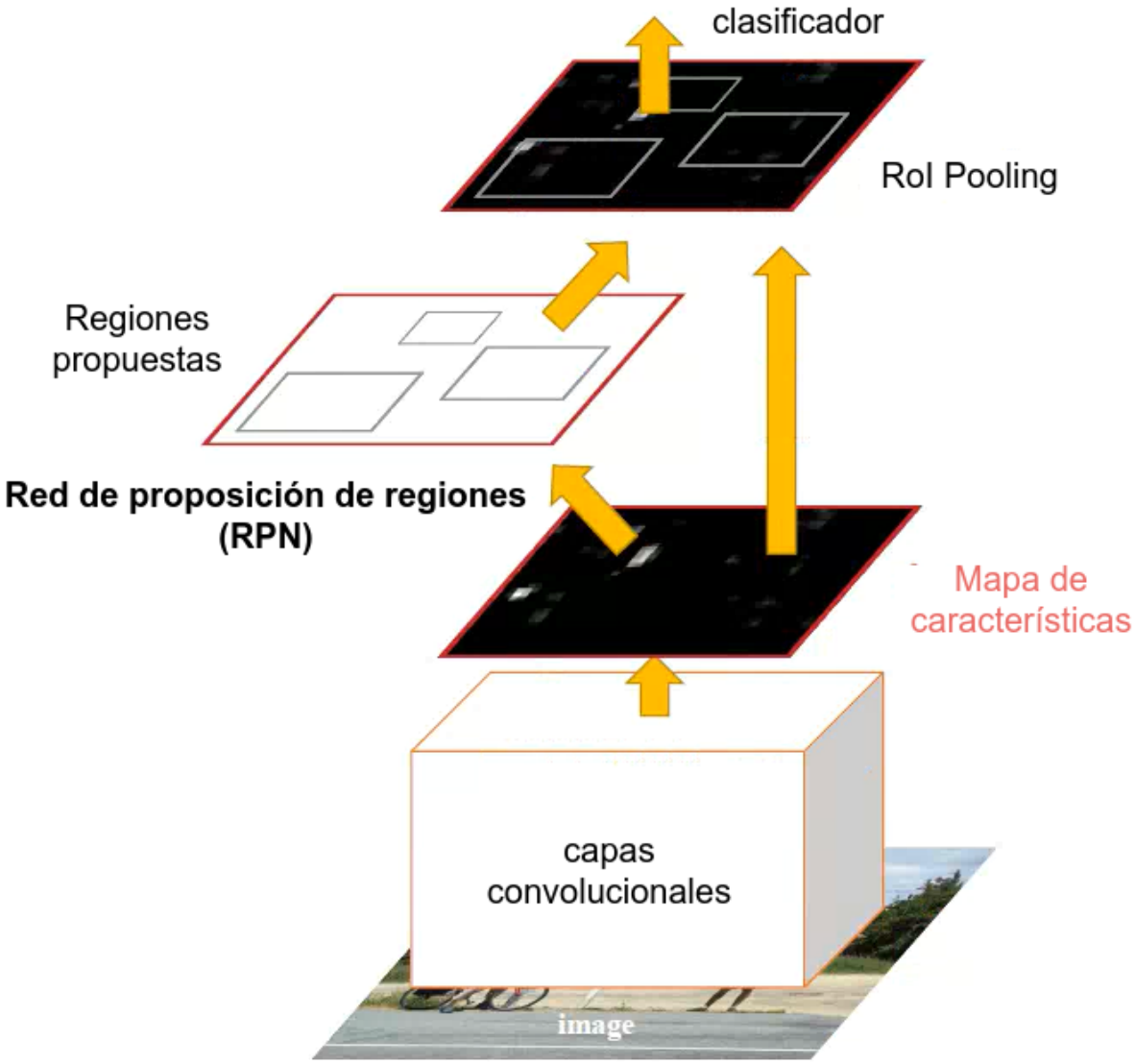


Figura 2.9: Esquema de funcionamiento de red Faster R-CNN. Traducido desde [5].

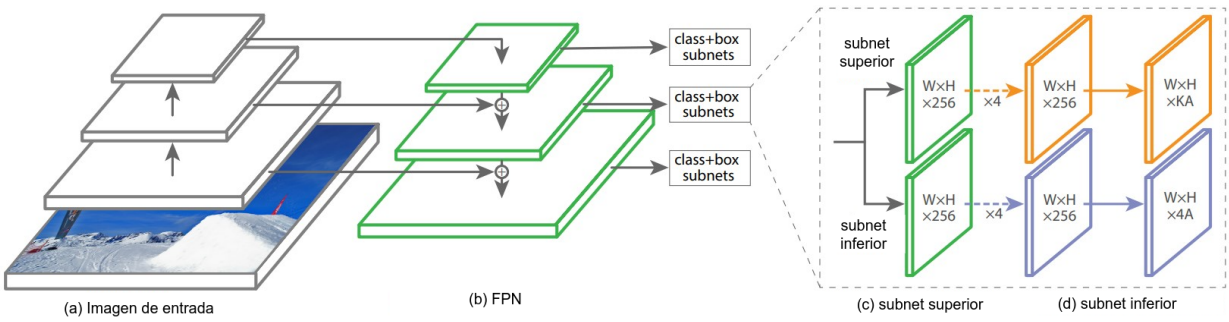


Figura 2.10: Estructura de RetinaNet Detector [11]

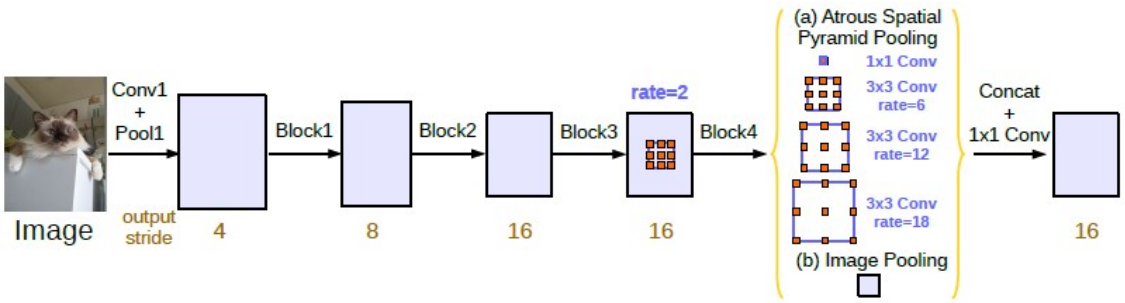


Figura 2.11: Estructura DeepLab [30]

de forma previa a las características y no en paralelo, por lo que primero se produce una reducción de características en el backbone y luego la extracción, sin embargo, a causa de la naturaleza de las convoluciones diluidas, todas las escalas mantienen la misma dimensión no obstante, la escala que examinan cambia debido a la modificación en su *rate* o cantidad de dilución[30]. La estructura descrita se puede observar en la Figura 2.11, en la cual se muestra la forma en que los diferentes filtros diluidos son utilizados en paralelo para añadir información multi-escala a las características finales.

2.4.2. One-Stage detection

A diferencia de la aproximación a la detección en dos etapas, que es aplicada a un conjunto *sparse* (poco denso) de ubicaciones candidatas a objeto, este tipo de sistema es aplicado sobre un muestreo denso y regular de posibles ubicaciones, escalas y ratios de aspecto de objetos. Tiene el potencial de ser mas rápido y simple que el enfoque de dos etapas, pero tradicionalmente ha quedado rezagado en términos de precisión de sus detecciones en comparación.[11]

You only look once (YOLO)

Este sistema fue, en su momento, un nuevo enfoque a la detección de objetos. En este caso, se plantea la detección como un problema para *bounding boxes* separados y sus probabilidades de clase asociadas [10]. Es decir, se hace una regresión sobre los espacios de la imagen determinando la probabilidad de que dicho espacio pertenezca a una determinada clase. Una red neuronal predice *bounding boxes* y las probabilidades de clase desde las imágenes completas en una sola evaluación.

Para entender el proceso que este sistema realiza, se puede observar la Figura 2.12. En esta, se describe como en la primera etapa, una red convolucional predice múltiples *bounding boxes* simultáneamente y la probabilidad de su pertenencia a cada clase. Luego, se entrena con imágenes completas y se optimiza directamente el desempeño de la detección.[10]

2.5. One-Shot Detection

El campo del *one shot learning* busca imitar la capacidad que tiene el sistema visual humano de aprender características escenciales de un objeto, a través de muy pocas muestras y sin conocimiento previo de un objeto nunca antes visto y, luego, ser capaz de identificarlo. El objetivo es generar la capacidad de generalizar, de forma satisfactoria, las características

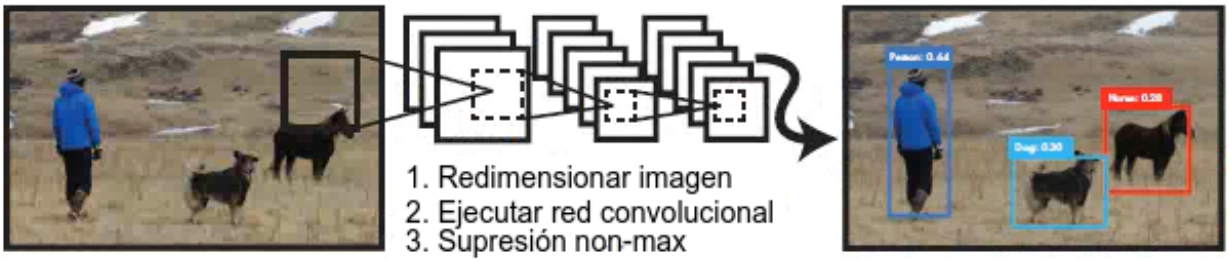


Figura 2.12: Esquema de funcionamiento de YOLO. Traducida desde [10].

de un objeto a través de una mínima cantidad de ejemplos y, posteriormente, identificar otras instancias del mismo sin mayores dificultades. Una vez que se ha visto un caballo, se debería tener la capacidad de identificar otros caballos sin importar su tamaño, su color o la distancia a la que se encuentra.

La idea detrás de esto es poder generar sistemas en los que no se necesite de una gran cantidad de muestras pre-etiquetadas para poder detectar correctamente, al contrario de la tendencia que se ha producido en los últimos años en el campo de la detección de objetos en imágenes.

Un trabajo en este campo destacable y muy relevante para el trabajo que se presentará a continuación es el de Úbeda [26]. En este, se plantea una metodología en la cual se detectan patrones (*queries*) en documentos históricos utilizando *Feature Pyramid Networks (FPN)* pre-entrenadas como extractores de características.

El modelo planteado utiliza los mapas de características generadas para representar un a zona de la imagen original, a la cual se le asigna el término *campo receptivo*. Esto se realiza además en un estilo multiescala, aprovechando la capacidad de la *FPN* de producir características de esa forma. Por este motivo, se realiza una selección de escala dependiendo de la *query* a detectar, utilizando los campos receptivos asociados a ella. En la Figura 2.13 se observan puntos que representan una zona de la imagen original en dos escalas diferentes. Estos campos receptivos actúan como *neuronas*, y cuando un patrón similar al de la *query* buscada, se detecta una activación zonal de estas, por lo que se asigna una detección en dicha zona.

El método expuesto obtuvo muy buenos resultados en el set de datos *DocExplore*, logrando superar el estado del arte en detección al momento de la publicación.

2.6. Conceptos Complementarios

2.6.1. Definición de característica

Para el propósito de este trabajo, y como se ha mencionado anteriormente, es necesario definir que se entenderá por característica. Como es sabido, una imagen a color común se compone de pixeles. Esta cantidad de pixeles está determinada por su tamaño (alto y ancho). El color de cada uno de ellos está dado por los valores en los tres canales de cada uno, definidos por el espacio *RGB*, definiendo un canal para rojo, uno para verde y uno para

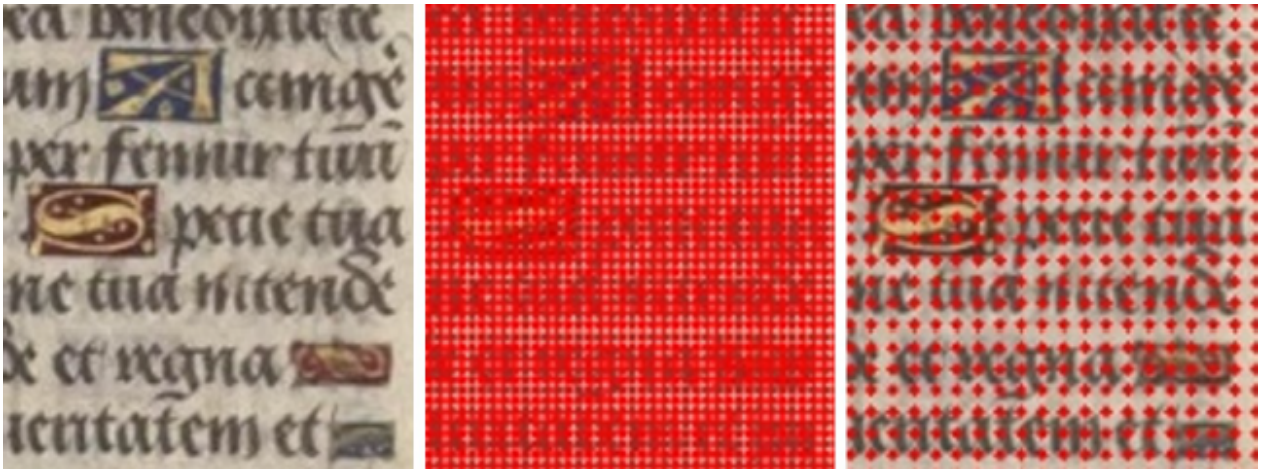


Figura 2.13: Visualización de centros de cada campo receptivo (puntos rojos) para todas las neuronas de la capa P_2 (imagen central) y P_3 (derecha).

azul. Las características de las que se hablará en este documento siguen el mismo concepto de pixel, pero tiene una mayor cantidad de canales. La cantidad de canales y el valor que estos obtienen estará dado por la capa de extracción que se esté utilizando y los filtros que la definen. Estas capas se describen en mayor detalle en la sección 4.

Se acuñará entonces el término característica como cada punto de la imagen producida por la capa de extracción que tiene una profundidad en canales. Pareciéndose de esta forma a un pixel pero con una distinta profundidad y forma de generación. y una explicación gráfica y simple de qué es lo que se entiende por característica en este documento se puede ver en la Figura 2.14

2.6.2. Patrón Plano

Se hablará de patrones planos en este documento para denominar el tipo de imágenes que serán utilizadas principalmente para este trabajo como imágenes de búsqueda (*queries*). Estos patrones planos cumplen algunos simples requisitos, por lo que pueden ser agrupadas bajo este concepto. Se definirán aquellos elementos que no poseen volumen y que cuentan con sólo una vista, siendo esta la frontal. Un ejemplo de esto es por ejemplo un logo, pues se definen o dibujan en dos dimensiones, sin que estos posean profundidad. Esto muestra una clara diferenciación de un objeto común que puede o no caber en esta descripción, pensando por ejemplo en un animal, este tiene varias vistas además de la frontal, por lo que quedaría excluido del grupo de imágenes de interés para este trabajo.

Es importante mencionar que esta vista frontal puede sufrir deformaciones, tanto de tamaño como de escala y rotación, por lo que sigue siendo un problema interesante, en particular para problemas de tipo planograma.

2.6.3. Componentes Principales

En estadística, el análisis de componentes principales corresponde a una técnica que permite reducir la dimensionalidad en sets de datos demasiado grandes. Este método realiza en

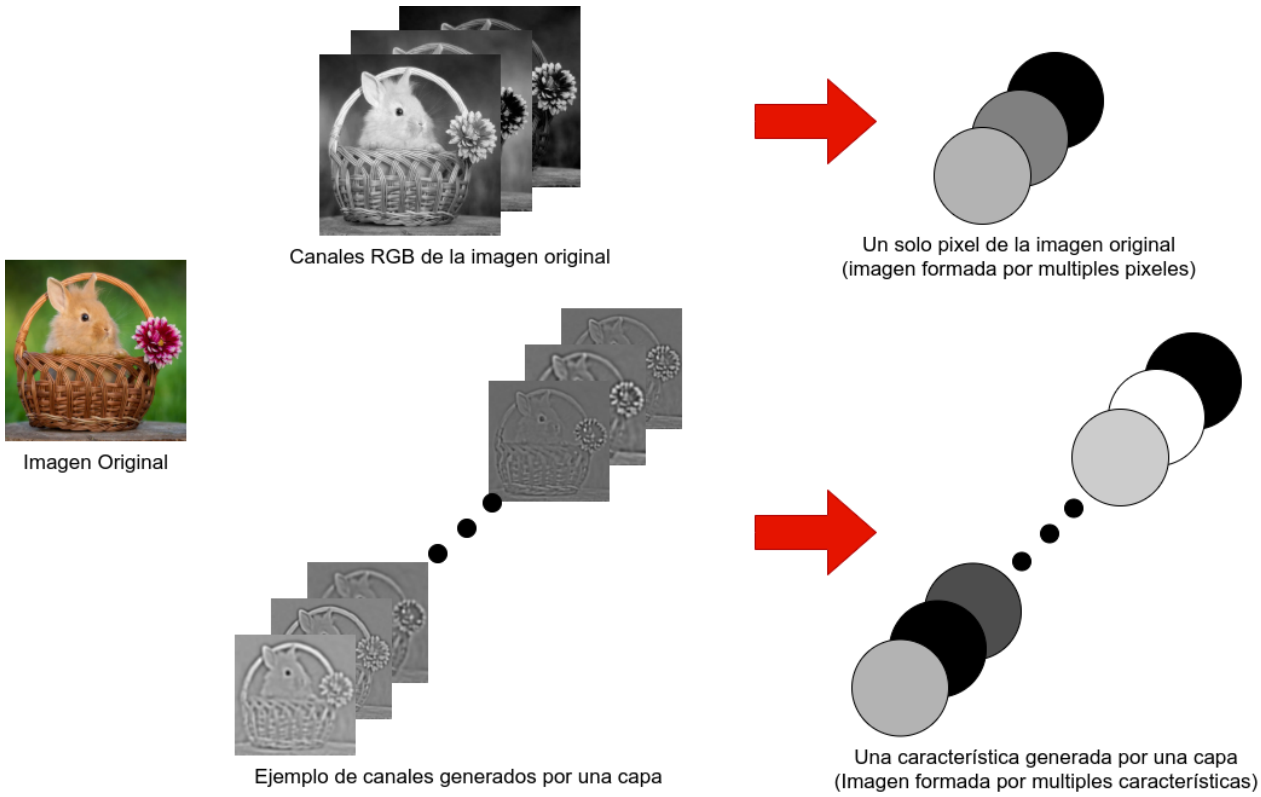


Figura 2.14: Esquema explicativo de lo que se entiende por característica. Se realiza la comparación con un pixel.

cierta medida un intercambio entre precisión y simplicidad, pues selecciona la información que "representa" de mejor manera los datos y elimina cierta información que considera poco importante. Esto tiene la utilidad de que, si bien se pierde información, el trabajo con los datos se simplifica en gran medida, pues es más fácil visualizar datos y también trabajarlos más eficientemente debido a la reducción del poder computacional necesario.

El proceso de selección de componentes principales para un set de datos vectoriales es relativamente simple; el proceso puede resumirse en 4 pasos:

1. Se calculan tanto el promedio como la desviación estándar para cada variable de los vectores y posteriormente todos ellos son normalizados, siguiendo lo mostrado por la Ecuación 2.1.

$$\hat{x} = \frac{x - \overline{x}}{\sigma} \tag{2.1}$$

2. Luego, para entender qué tanto varían los valores de los vectores del promedio y entre ellos mismos (es decir, si se parecen entre ellas). Se calcula para esto la matriz de covarianza (como ejemplo, se muestra la matriz de covarianza para vectores de 3 dimensiones en la ecuación 2.2). Si dos variables de los vectores se relacionan en gran medida, uno de ellos se conservará mientras que el otro será eliminado.

$$\begin{bmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{bmatrix} \tag{2.2}$$

3. Posteriormente se utiliza álgebra lineal para calcular los vectores propios y valores propios de la matriz de covarianza.
4. Utilizando estos valores y vectores, mediante transformaciones lineales, se construyen nuevos vectores diferentes a los de los datos originales y con menor dimensionalidad. Esto ayuda a conservar una gran cantidad de información sin la necesidad de tener la gran cantidad de variables de los vectores iniciales, generando otros menos "profundos".

2.6.4. Correlación Cruzada

La correlación cruzada es similar en principio a la convolución. La diferencia principal, es que la correlación no involucra la inversión de la segunda función en la operación. La operación para la correlación cruzada se puede apreciar en la Ecuación 2.3 para el caso discreto y 2.4 para el continuo. En particular, la correlación es útil para comprobar si dos funciones (continuas o discretas) son parecidas entre ellas.

$$(f * g)_{\mathrm{i}} = \sum_{j} f_j^* \cdot g_{\mathrm{i}+j} \tag{2.3}$$

$$(f * g)(x) = \int f^*(t) \cdot g(x+t)\ \mathrm{d}t \tag{2.4}$$

En el caso bidimensional (matrices bidimensionales) la correlación entre dos matrices X y H de dimensiones $M \times N$ y $P \times Q$ respectivamente, se genera una nueva matriz C de dimensiones $M + P - 1 \times N + Q - 1$. Cada punto de esta matriz C será representado por la fórmula mostrada en la Ecuación 2.5.

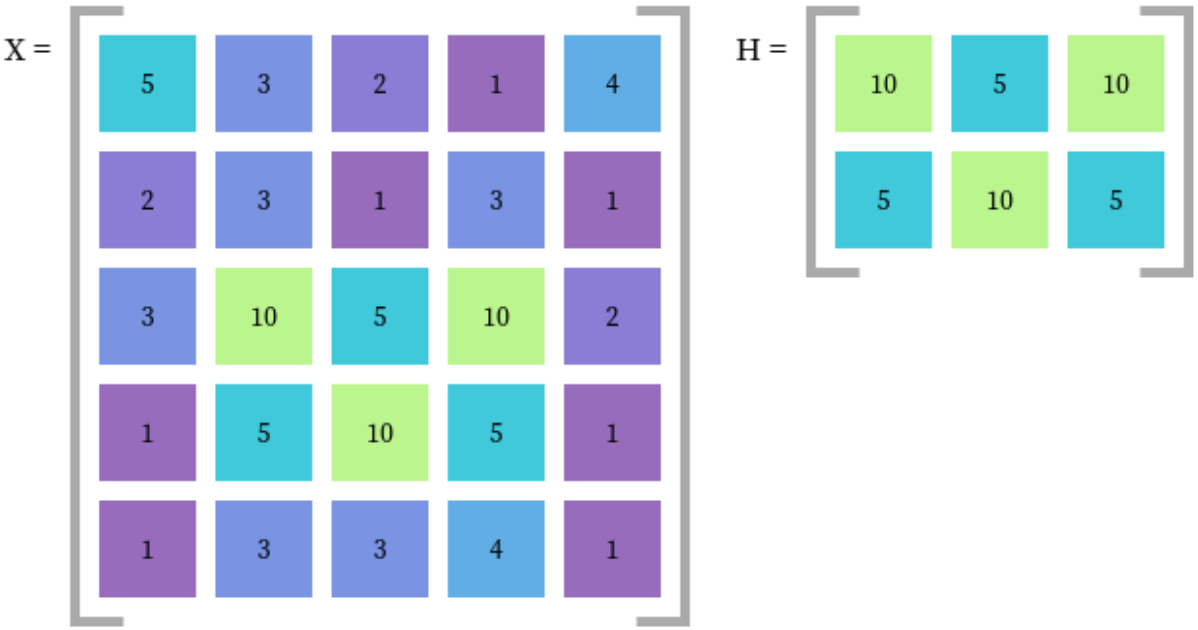
Se adjunta en la Figura 2.15 un ejemplo de correlación discreta entre dos matrices, en la cual se manifiesta el parecido entre dos matrices bidimensionales[15]. Es claro que el resultado numérico de *C(j,k)* es mayor mientras más se parezca la sección de la matriz X que está siendo operada en el momento con la matriz *H*. Es por este motivo que se puede utilizar como una búsqueda de patrones.

En el caso de una matriz tridimensional, se considera la correlación cruzada como la suma individual de todas las correlaciones entre matrices bidimensionales.

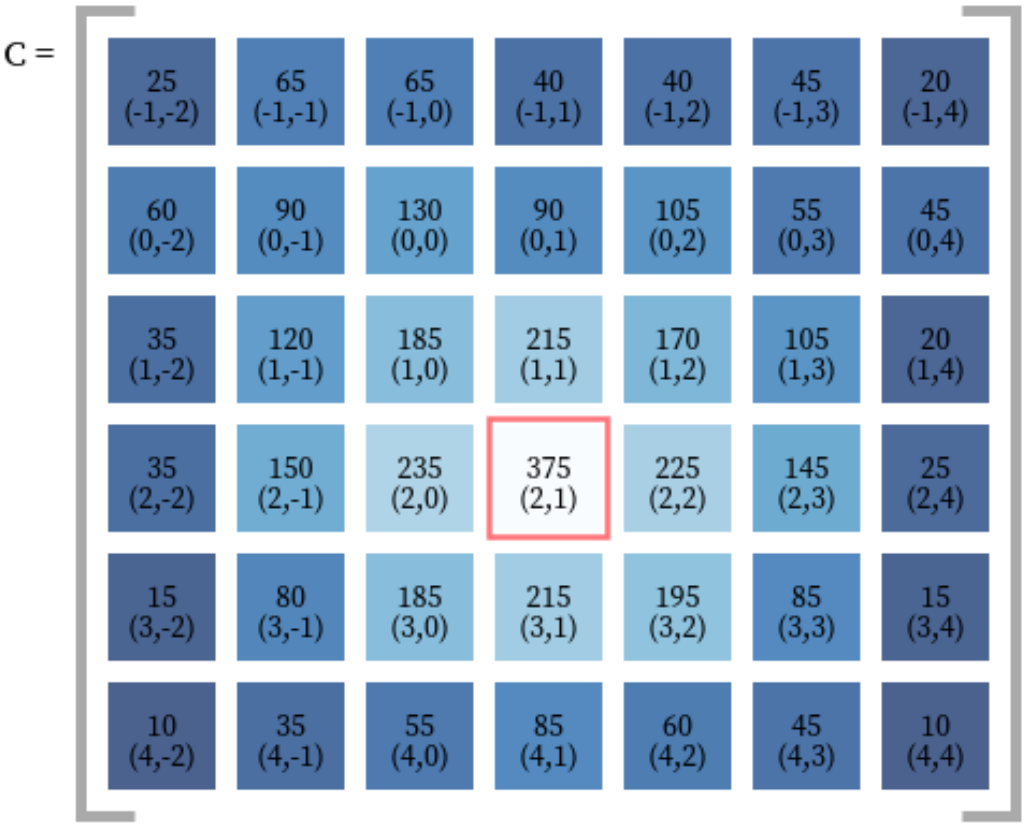
$$C(k,l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} X(m,n) \cdot \bar{H}(m-k, n-l),\ \textit{con } \bar{H} \textit{ conjugacin compleja de } H \tag{2.5}$$

2.6.5. Normalización L2

La normalización L2 corresponde a una forma de normalización que opera según la ecuación 2.6. Corresponde a la división de cada valor individual de un vector por la raíz cuadrada de la suma de los valores cuadrados del mismo.

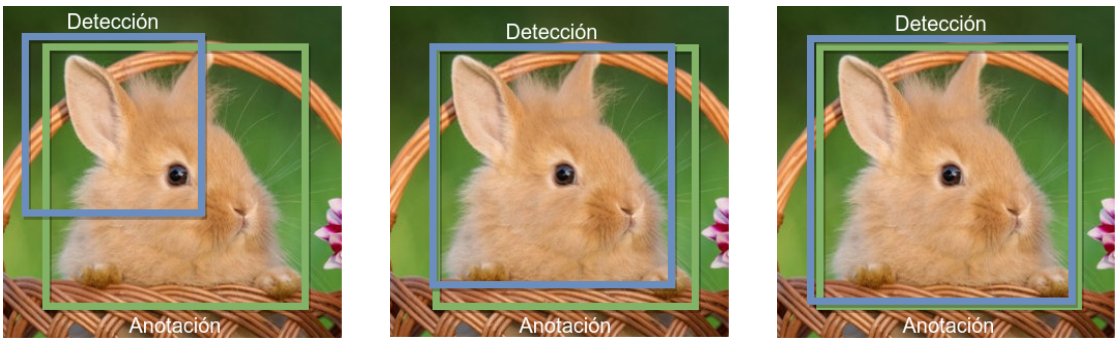


(a) Matrices a operar mediante correlación cruzada.



(b) Matriz resultante de correlación bidimensional.

Figura 2.15: Correlación cruzada en matrices bidimensionales.[15]



(a) IoU = 0.4034 (b) IoU = 0.733 (c) IoU = 0.9264

Figura 2.16: Comparación de valores IoU

$$\hat{V} = \frac{V}{\sqrt{\sum_{i=0}^{n} V_i^2}} \tag{2.6}$$

Lo que hace esta norma es calcular la distancia que existe entre el la coordenada del vector y el punto de origen del espacio vectorial, y normalizar por ella.

2.6.6. Intersección sobre la unión (*IoU*)

Es una unidad de medida de detección que sirve para determinar la precisión de la detección realizada por un modelo en comparación con el verdadero elemento existente en la imagen. Esta medida se obtiene tomando el *bounding box* detectado y el *bounding box* real. Se calcula el área de intersección entre ellos y el área total formada por ambos, y se consigue la intersección sobre la unión como se muestra en la Ecuación 2.7. Este tipo de medición pues es bastante buena para calcular la similitud entre los *bounding boxes* encontrados y, con esto, establecer el nivel de precisión de las detecciones encontradas para cada caso. Esto se ilustra claramente en la Figura 2.16, donde se representa la detección como el *bounding box* azul y la anotación correcta con un *bounding box* verde.

$$IoU = \frac{Area_{Interseccion}}{Area_{Union}} \tag{2.7}$$

2.6.7. Precisión media promedio

La presición media promedio, también llamada *mean average presicion (mAP)* es una medida que permite estimar qué tan bueno es el desempeño de las detecciones de una determinada metodología, tomando valores entre 0 y 1, con 1 como máximo. En particular, el *mAP* se puede calcular a partir de la curva *precisión/recall* dada por un set de datos. Tanto la *precisión* como el *recall* se pueden calcular según las ecuaciones 2.8 y 2.9, fijándose en la cantidad de *verdaderos positivos (TP)*, *falsos positivos (FP)*, *verdaderos positivos (TP)* y *verdaderos negativos*, dado un criterio de aceptación (por ejemplo, intersección sobre la unión mayor a 0.5 en detección).

$$precision = \frac{TP}{TP + FP} \tag{2.8}$$

$$recall = \frac{TP}{TP + FN} \tag{2.9}$$

Si se genera una curva de *precisión/recall* a lo largo del set de datos que se quiere estimar, es posible obtener el mAP como el cálculo del área bajo la curva.

El caso de la evaluación en la tarea de recuperación de imágenes es un tanto más simple. La diferencia radica en que en este caso se ordenan de mayor a menor índice de confianza las imágenes en las que se detectó la *query* (sin repetición de imágenes), como un *ranking*. Luego se considera *verdadero positivo (TP)* si la imagen contiene la *query* o *falso positivo (FP)* si no la contiene. Los *falsos negativos (FN)* corresponden a todas las imágenes que quedaron fuera del *ranking* en le punto de la evaluación de la curva. Con esto se genera también la curvas de *precisión/recall* para obtener la precisión promedio como el área bajo la curva.

2.7. Estado del arte en detección de logos

A continuación, se presentarán algunos métodos utilizados de forma reciente para la resolución del problema de detección de logos en imágenes.

2.7.1. Open set logo detection and retrieval

Como primer acercamiento a la detección de logos, se explicará una estrategia que utiliza *deep learning* para realizar la detección. En particular, esta estrategia tiene un enfoque de detección *open set*, lo cual implica que se busca distinguir los logos en espacios abiertos o naturales. Es por esto que además de la metodología se recolecta y se comparte al público el set de datos *Logos in the Wild*[12].

El funcionamiento de esta metodología consiste en entrenar dos redes convolucionales. Una red es entrenada para reconocimiento de logos en general, es decir, sin identificar qué logo es sino que sólo proponer regiones donde existen logos en la imagen y la otra red es entrenada, para clasificar en las regiones detectadas por la primera si es que alguna corresponde a una consulta realizada de un logo que puede o no haber visto durante su entrenamiento. El funcionamiento de esta red se describe más claramente en la Figura 2.17. Para la red de detección de logos en general, se utilizó una red *Faster R-CNN*, como la mencionada en la sección 2.4.1 entrenada para esta tarea, pero se establece que también sería posible el uso de una red YOLO como la de la sección 2.4.2 para detección mas rápida a costo de desempeño en la detección. Para la red de clasificación, se utilizan *CNN* del estado del arte entrenadas en el set de datos Image-Net [13], las cuales son ajustadas al dominio de los logos mediante un proceso de *fine-tuning*.

El enfoque de extracción de características mediante estas dos redes es ventajosa en términos de la calidad de estas características debido a que la red de extracción (segunda red

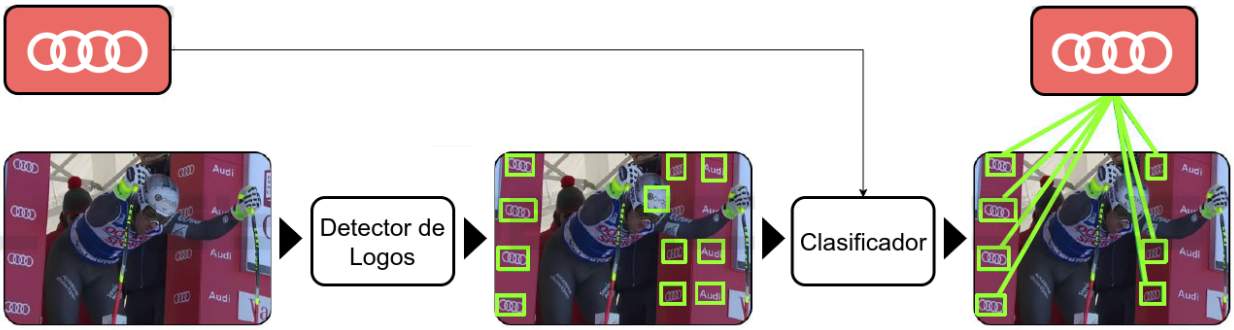


Figura 2.17: Estrategia de detección de logos implementada en [12].

mencionada) sólo debe concentrarse en características regiones específicas que contienen logos, mientras que un enfoque que incluya tanto detección como comparación en la misma red (por ejemplo una misma Faster R-CNN), pues esta carece de generalización para clases nunca antes vistas.[12]. En particular, esto es muy importante para el paradigma de detección de *open set*, pues la capacidad de generalizar, para clases no vistas anteriormente, es fundamental. La comparación entre los enfoques mencionados entre una y dos redes se puede observar en la Figura 2.18.

2.7.2. Scalable logo detection and recognition with minimal labeling

Esta propuesta [14] aborda un enfoque de detección de logos en una imagen a través de la utilización de *deep learning* y data de entrenamiento sintética (es decir, imágenes generadas de forma artificial) como la mostrada en la Figura 2.19. En particular, en esta estrategia se utilizan las imágenes sintéticas para entrenar una red CNN en detección de logos, y luego es utilizada para detectar y localizar logos desde imágenes extraídas desde la web. Finalmente, estas imágenes son utilizadas para entrenar un clasificador de logos. Este enfoque de dos etapas permite clasificar múltiples logos en una misma escena. La ventaja de esta metodología es que no se necesita una gran cantidad de imágenes etiquetadas de forma manual, pues el entrenamiento es realizado con imágenes obtenidas de una manera automática, con supervisión humana mínima.

La primera red entrenada es una red de tipo *Faster R-CNN*, cuyo set de entrenamiento consiste en imágenes sintetizadas, y la segunda red, que permite clasificar las imágenes, corresponde a una *DenseNet* [16]. Un resumen de la propuesta se puede observar en la Figure 2.20.

2.7.3. Deep learning logo detection with data expansion by synthesising context

En la estrategia definida en [17], se describe un modelo síntesis de imágenes de entrenamiento capaz de mejorar significativamente la detección de logos mediante una cantidad

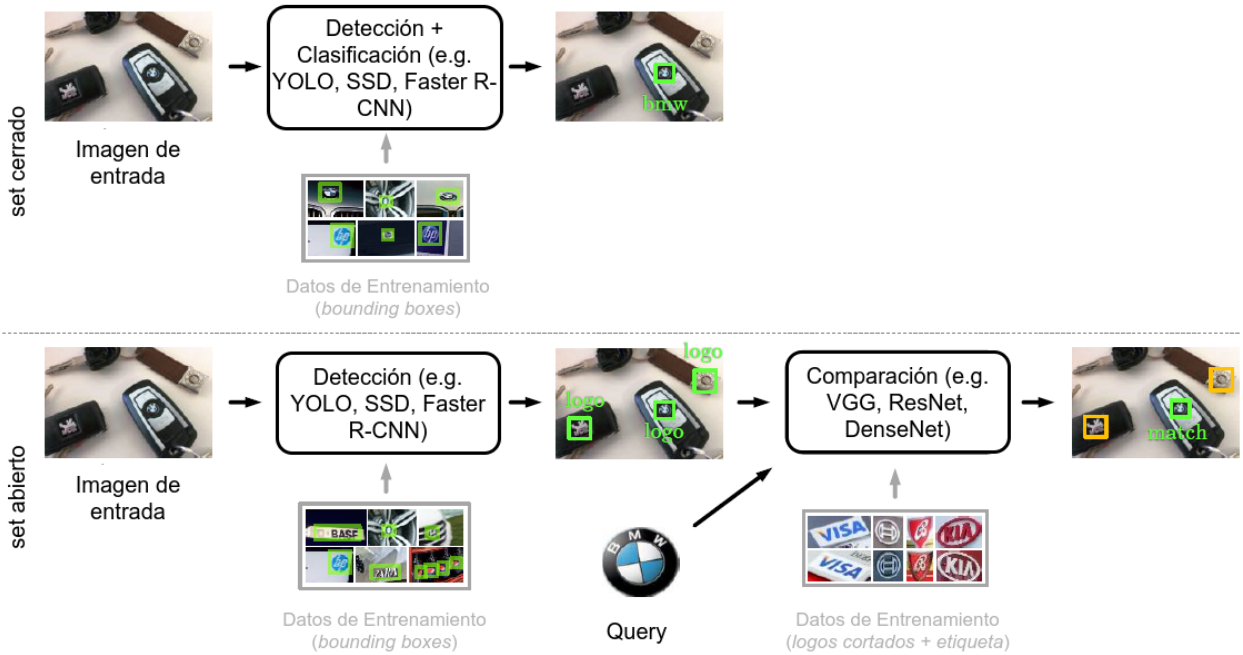


Figura 2.18: Comparación entre estrategia con una y dos redes.[12].

Figura 2.19: Ejemplos de imágenes sintetizadas [14].

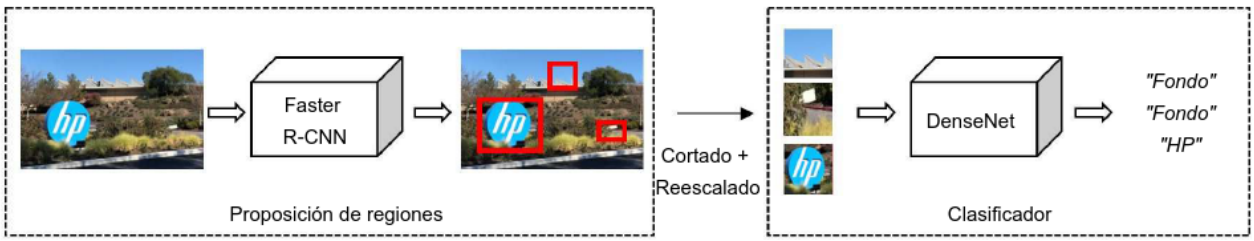


Figura 2.20: Resumen de estructura de detección de logos propuesta en [14].

pequeña (e.g, 10) de imágenes etiquetadas capturadas en un contexto realista, evadiendo el costo de etiquetado manual. Específicamente, se desarrolla un nuevo algoritmo diseñado para generar imágenes de entrenamiento de tipo *Logos de Contexto Sintético* (SCL) para incrementar la robustez de los algoritmos de detección de logos en fondos no conocidos, generando una expansión considerable de la cantidad de data disponible. Además de esto, este trabajo proporciona un set de datos de imágenes sintetizadas de logos con sus *bounding boxes* anotadas, consistente de 463 logos.

El algoritmo consiste en la utilización de imágenes de logos e imágenes de contexto aleatorio que no corresponde a logos. Además, se considera la existencia de variaciones en los logos distintas de las del contexto, por ejemplo, la geometría/perspectiva del logo y los cambios de iluminación. Para simular estos efectos, se aplican fundamentalmente dos transformaciones:

- Transformación geométrica (*Geometric Transform*): Se realiza una transformación aleatoria de primero de escala, luego de cizallamiento y finalmente de rotaciones.
- Transformación de coloración (*Colouring Transform*): Se realiza la coloración mediante la utilización de variaciones de los valores de los pixeles mediante lo mostrado en la Ecuación 2.10, con c representando el valor del pixel y τ representando un valor aleatorio sampleado uniformemente entre 0 y 2.

$$c^* = \tau c \tag{2.10}$$

Posteriormente, se genera un número aleatorio de variaciones para cada logo y se utilizan para sintetizar imágenes colocando los logos transformados sobre las imágenes de contexto.

Con el procedimiento antes descrito, se obtienen imágenes sintéticas como las mostradas en la Figura 2.21, en donde se observa la inserción de logos deformados en diferentes fondos donde no se encontrarían habitualmente.

Con este algoritmo, se muestran resultados en la detección de logos que muestran mejorías al entrenar arquitecturas de detección con las imágenes sintéticas en comparación con un entrenamiento con una menor cantidad de imágenes etiquetadas ya disponibles. Esto se demuestra en una mejoría de más del 10 % para una red *Faster R-CNN* del estado del arte sobre el set *Flickr-Logo32* y de más de 40 % sobre el set de datos *TopLogo-10*.[17]

2.7.4. Scalable Logo Recognition using Proxies

En este articulo[18], se formula el reconocimiento de logos como un problema de detección *few-shot*. De forma similar a lo descrito en la Sección 2.7.1, se utiliza un flujo que consta de dos etapas. Primero existe un detector de logos universal o general y posteriormente se utiliza (y entrena) un reconocedor de logos *few-shot*, que corresponde a un espacio embedido.

El detector universal consiste en estructuras de *deep learning* como las ya antes mencionadas que se encargan de entender qué es un logo y entregar las regiones donde estos se encuentran en las imágenes. En particular, las estructuras probadas en esta estrategia fueron *SSD*, *Faster R-CNN* y *YOLOv3*.

Por otro lado, el reconocedor de logos *few-shot* utiliza el algoritmo de búsqueda *nearest neighbours* entrenado por *triplet loss* utilizando proxies[19]. Este entrenamiento se realizó así

Figura 2.21: Ejemplos de imágenes de logos de contexto sintético [17].

debido a que la utilización de *proxies* optimiza el *embedding space* de forma que no se utilice simplemente una medida de distancia y también a que se ha demostrado que el desempeño de las funciones de pérdida basadas en la medida de distancia, depende fuertemente de la forma en la que los pares o tripletas de imágenes son seleccionadas. Es así como el detector universal entrega candidatos a logo y, en el reconocedor de logos embedido, toma estas propuestas y las proyecta junto con la *query* en el espacio. Aquellas propuestas más cercanas a la *query* bajo un criterio son clasificadas como detecciones de dicha *query*.

Una visualización t-SNE[20] del *embeded space* generado para un subconjunto de de clases de test se puede observar en la Figura 2.22. En este caso, la arquitectura utilizada fue una Resnet50[27] modificada.

Esta formulación es relevante debido a que en su experimentación utiliza como testeo el dataset Flickrlogos 32, que es un subconjunto del dataset utilizado en este documento. Los resultados presentados para este set corresponden a los mostrados en la Tabla 2.1.

Modelo	mAP Detección FlickrLogos 32
Faster R-CNN	0.42
SSD	0.38
YOLOv3	0.22
Faster R-CNN + Few Shot	**0.56**
SSD + Few Shot	0.44
YOLOv3 + Few Shot	0.3

Tabla 2.1: Resultados estado del arte en detección para Flickrlogos 32

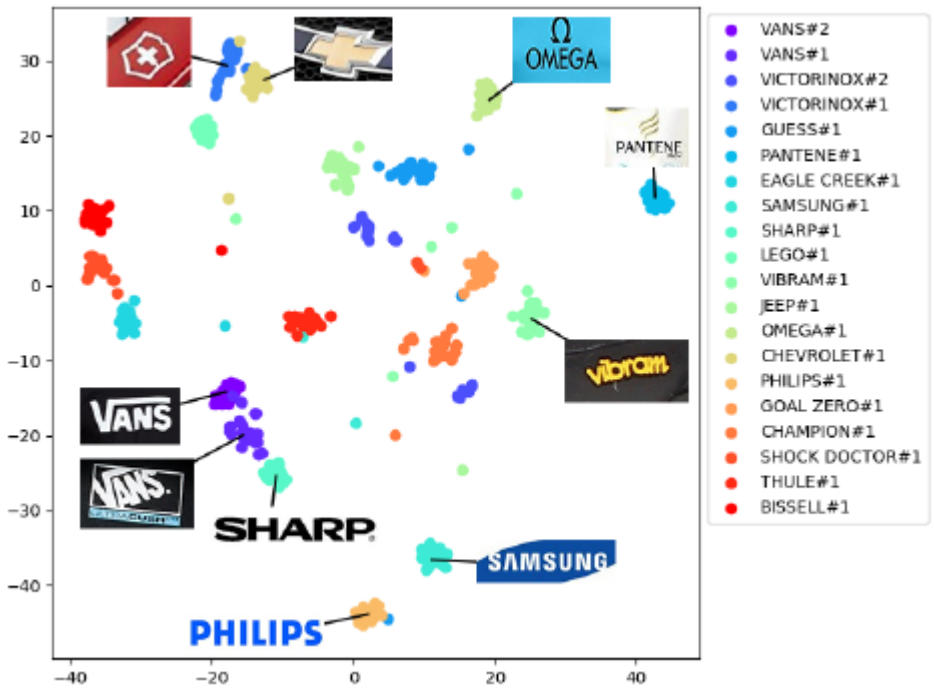


Figura 2.22: Gráfico t-SNE de elementos de test en el *embedding space* conseguido [18].

2.8. Estado del arte en detección One-Shot de logos

En esta sección se explorarán trabajos dedicados al enfoque *one-shot* en detección de logos. En particular, destacan los trabajos de Úbeda [26], Wiggers [21][22] y Curi et al.[31]. Todos abordan las tareas de detección y búsqueda de imágenes en documentos históricos en el dataset DocExplore y, Wiggers y Curi en particular, analizan la utilización de sus estrategias en búsqueda de logos. Además de esto, todos ellos utilizan estrategia de extracción de *features* desde modelos convolucionales y posteriormente experimentan con diferentes medidas de similitud. En particular, este trabajo se basa en la estrategia planteada por Curi[31], debido a que corresponde a una investigación conjunta que está en curso en estos momentos, en la cuál el autor de este documento participa.

2.8.1. Resultados Dataset DocExplore

A modo de resumen y con el propósito de realizar una comparación posterior, se presentan los resultados presentados por los dos trabajos expuestos anteriormente en el dataset DocExplore en la Tabla 2.2.

Metodo	mAP Detección	mAP Recuperación
En et Al.[25]	0.1569	0.5801
Wiggers et Al.[22]	0.1740	0.3860
Úbeda et Al.[26]	0.2720	0.5770
Curi et Al.[31]	0.6442	0.8131

Tabla 2.2: Resultados de estado del arte para el *dataset* DocExplore.

Gracias al protocolo de evaluación presentado en [26] y al trabajo realizado por [31], es posible separar los resultados en una clasificación de escala y forma de las *queries*; estos se presentan en la Tabla 2.3.

Forma	Tamaño	mAP Detección			mAP Recuperación		
		En et Al.	Ubeda et Al.	**Curi et al.**	En et Al.	Ubeda et Al.	**Curi et Al.**
Cuadrada	Grande	0.546	0.681	**0.880**	0.881	0.749	**0.937**
	Pequeña	0.102	0.546	**0.858**	0.801	0.742	**0.927**
No cuadrada	Grande	0.405	0.509	**0.752**	0.701	0.660	**0.826**
	Pequeña	0.149	0.214	**0.603**	0.535	0.459	**0.792**

Tabla 2.3: Resultados de estado del arte para el *dataset* DocExplore de acuerdo a protocolo de evaluación planteado en [26]. Origen de tabla en [31].

Capítulo 3

Metodología

3.1. Resumen

En este trabajo se realizó experimentación para encontrar características descriptivas a través de modelos de *deep learning* para luego utilizarlas en la detección *one shot* de logos en imágenes en contextos reales. El proceso consistió en primer lugar en la búsqueda de *datasets* de logos que hubieran sido utilizados anteriormente para estudios de este tipo. Posteriormente, se procedió a adquirir modelos de *deep learning* ya entrenados capaces de extraer las características necesarias, siendo estos VGG16 y ResNet, pre entrenados en ImageNet, y obtenidos desde https://keras.io/api/applications/.

Se obtuvieron las características a partir de los modelos y se dio paso a la experimentación. Como primera parte experimental, se recurrió a operaciones entre las características extraídas de imágenes que contuvieran una *query* y dicha *query*, una para generar mapas de calor y apreciar si existe o no similaridad.

Luego de esto, se formuló una metodología a través de la cual se realizaron las detecciones en las imágenes para ambos modelos, formulado principalmente en dos etapas, *online* y *offline*.

Finalmente se comprobó la precisión de las mediciones de acuerdo con el criterio *intersección sobre la unión*, que se explica en la sección 2.6.6 y se calcularon métricas de precisión.

3.2. Recopilación de Datasets

Este paso de la metodología fue dedicado a la búsqueda de sets de datos de patrones bidimensionales, con el propósito de desarrollar el modelo. Estos sets han sido utilizados en diversas investigaciones tanto en clasificación como en detección. Corresponden a DocExplore, *Flickr-Logos 47* y por consiguiente, *Flickr-Logos 32* (subconjunto de *Flickr-logos 47*), los cuales serán descritos a continuación.

Figura 3.1: Variabilidad de *queries* en DocExplore *dataset*

3.2.1. DocExplore

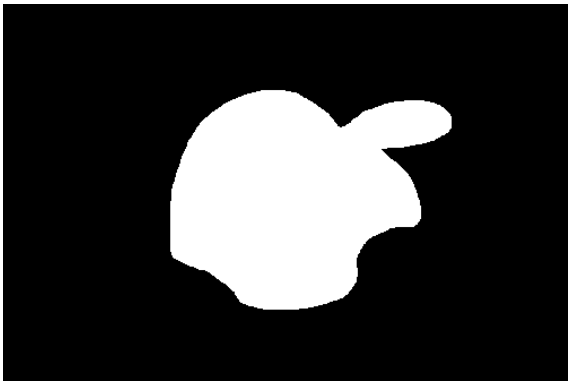
DocExplore es un proyecto que busca resolver un problema de detección de patrones que corresponden a símbolos que aparecen en documentos históricos. Este proyecto puede ser encontrado en `http://www.docexplore.eu/`. Existe además de esto, un repositorio de imágenes y *queries* extraído directamente de este proyecto[25], que consta de 1500 imágenes de páginas de documentos y 1447 *queries*. Todas las *queries* difieren un poco en cuanto a tamaño, color, distorsión debido a escaneos y degradados. Una ilustración de algunas posibles *queries* existentes se presenta en la Figura 3.1.
Este set cuenta con una herramienta propia de cálculo de mAP, y no cuenta con anotaciones desde su origen. Por lo tanto, se utilizará esta herramienta como *benchmark* y cálculos realizados por el autor del presente documento a con un set de anotaciones manuales complementario realizado por Úbeda.[26]

3.2.2. Flickr-Logos 32

Este set de datos consiste de imágenes reales extraídas desde el sitio `www.flickr.com` que muestran logos de compañías en varias circunstancias. Fue diseñado para recuperación de logos (búsqueda) y para reconocimiento y detección multi-clase de logos. Sin embargo, a veces las anotaciones de este set se encuentran incompletas. Es por esto que existe el set de datos Flickr-Logos 47, del cual se hablará posteriormente. Fue utilizado en la investigación [1]. Este set contiene, de todas maneras, fotos que muestran 32 marcas en total, distribuidas en:

- Conjunto de entrenamiento: Consta de 320 imágenes, en las cuales se incluyen algunas elegidas a mano (10 por clase).

(a) 339296129.jpg



(b) 339296129_mask.jpg

Figura 3.2: Imagenes encontradas en Flick-Logos 32

- Conjunto de validación: Consta de 3960 imágenes. De estas, 30 por clase incluyen al menos una instancia de un logo dentro, y las otras 300 no contienen logos de ningún tipo.
- Conjunto de test: Consta de 3960 imágenes. De estas, 30 por clase incluyen al menos una instancia de un logo dentro, y las otras 300 no contienen logos de ningún tipo.

Además de las anotaciones de *bonding boxes*, se incluyen también anotaciones a nivel de pixel en forma de mascaras binarias, permitiendo marcar de forma mas precisa la posición de los logos en las imágenes. Un ejemplo de imagen encontrada en el set se puede observar en la Figura 3.2

3.2.3. Flickr-Logos 47

Es un set complementario al mostrado en la Sección 3.2.2, pero este contiene correcciones en las anotaciones y se han agregado 15 clases adicionales. La diferencia radica en que algunas clases se separan entre símbolo y texto, por ejemplo, la clase "Adidas" fue dividida en "adidas_symbol" y "adidas_text". Este set ha sido preparado de forma que exista en el formato de anotaciones de COCO[29]. Este set no cuenta con imágenes de consulta (queries) en sí mismo, pero sí con anotaciones, por este motivo, se decidió utilizar como *queries* los recortes de las anotaciones desde de las imágenes de consulta para ser posteriormente utilizadas. Es así como desde este set de datos queda con 1935 *queries* que se encuentran repartidas entre 833 imágenes.

3.3. Recopilación de estructuras capaces de extraer características

En esta sección, primero corresponde mencionar que el framework de trabajo será en *python*, en particular utilizando las librerías *pytorch*,, además de *tensorflow* en su versión para gpu y *keras*. Estas son fundamentales pues son la base para implementar las estructuras de *deep learning* que se utilizarán. Además de esto, todos los experimentos finales realizados fueron sobre una máquina equipada con una tarjeta gráfica *Nvidia GeForce GTX X* de $12GB$ de memoria GDDR5X. Procesador $Intel(R)Core(TM)$i75960X y $64GB$ RAM DDR4.

3.3.1. VGG

Como se explica en la sección 2.3.1, es una red convolucional profunda utilizada para reconocimiento de imágenes. Esta es una de las estructuras que se utilizarán para extraer características, considerándola como pre-entrenada en el conjunto de imágenes ImageNet [9], pero sin la utilización de las capas de clasificación *fully connected*, para tener la posibilidad de variar el tamaño de entrada de las imágenes y conservar solo las capas con filtros convolucionales.
Considerando que una imagen tiene 3 canales, las capas a utilizar en este modelo son aquellas que tienen profundidad de 256 y 512, y a esto es a lo que se refiere como característica. Cada vector de $1 \times 1 \times 256$ y $1 \times 1 \times 512$ generado por una imagen será considerado como un vector de características. Los modelos a utilizar en esta sección son los pre-entrenados encontrados en la librería *keras*, específicamente en la sección de aplicaciones ubicada en `https://keras.io/api/applications/`, el cual se carga sin las capas *fully connected*.

3.3.2. ResNet

Como se explica en la sección 2.3.2, es una red que presenta una mejora considerable con respecto a otras redes convolucionales profundas dada su forma de mejorar el entrenamiento para redes muy profundas (muchas capas), gracias a su forma de *aprendizaje residual*. El modelo pre-entrenado corresponde también a uno de los encontrados en *Keras*, nuevamente cargado sin las capas *fully connected*.

Esta la otra estructura que se utilizarán para extraer características, también pre-entrenada en el conjunto de imágenes ImageNet [9], también dejando de lado las capas de clasificación. En este caso también se utilizarán en este modelo aquellas capas producen vectores con una profundidad de 256 y 512. Cada vector de $1 \times 1 \times 256$ y $1 \times 1 \times 512$ generado por una imagen será considerado como un vector de características.

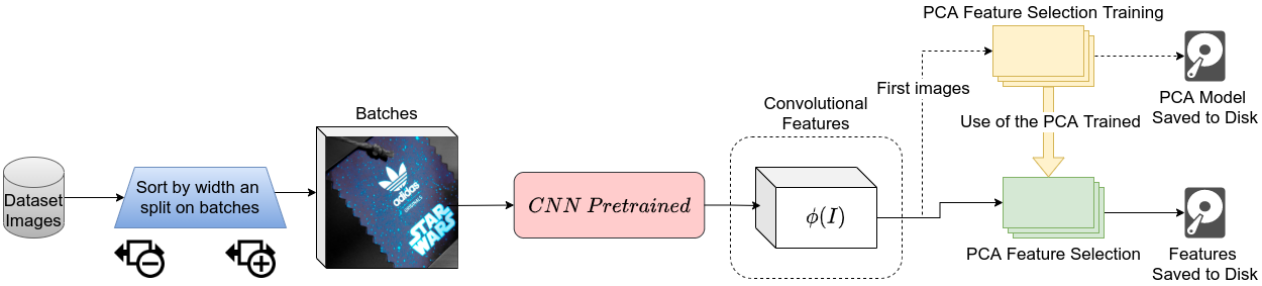


Figura 3.3: Fase Offline

3.4. Pruebas de mezclado de características

Para poder encontrar similaridad entre las características, para primero hacer retrieval (comprobar que la consulta se encuentra en la imagen) y posteriormente encontrar la detección de la imagen, marcando su bounding box, es necesario procesar las características de diferentes maneras y encontrar una que describa de buena forma las coincidencias entre imagen y consulta. Para esto, se utilizan las características de la query para operarlas con las características de la imagen a través de una medida de distancia o similitud, con el objetivo de encontrar coincidencia o poca distancia cuando la query si se encuentra en la imagen. Esto se almacena en una imagen y se reajusta al tamaño de la imagen original para generar mapas de calor entre cada cruce de características.

Se realizaron experimentos con las siguientes medidas para generar mapas de calor entre imagen y consulta:

- Convolución
- Correlación
- Distancia L1
- Distancia L2
- Coeficiente de correlación de Pearson
- Similitud Coseno

Para esta sección, no fue necesario un cálculo de mediciones de IoU, sino mera observación visual. Esto debido a que luego de mucha experimentación se observó como las mediciones de similitud entre características obtenidas presentaron cierta utilidad, especialmente la correlación. Aunque también se observó que existe una cierta dependencia de la orientación y tamaño de la query para la efectividad de la medida de similitud.

3.5. Procedimiento para la detección

El proceso de detección de imágenes se basa en el descrito por un trabajo que se encuentra en estos momentos en vías de publicación [31]. Consiste básicamente en una fase *offline* y una fase *online*, las cuales se presentan de manera resumida en las Figuras 3.3 y 3.4, y serán descritas en detalle a continuación.

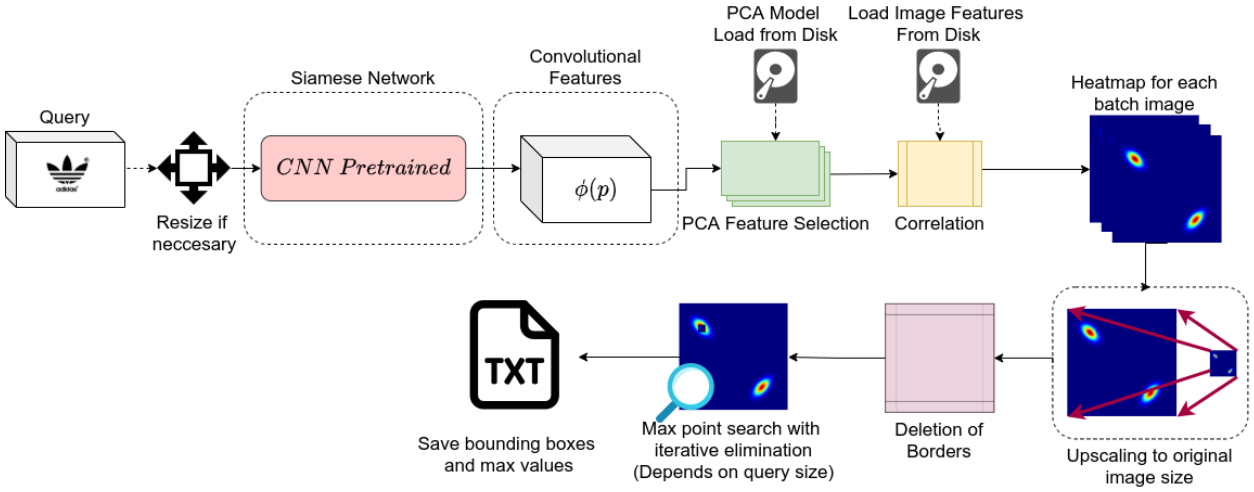


Figura 3.4: Fase Online

3.5.1. Fase Offline

Carga de imágenes

En particular, las imágenes no tienen un mayor tratamiento, salvo la utilización de *padding* cuando se carga más de una imagen en un *batch*. En tal caso, se ajusta un *padding* que incluya las dimensiones máximas de alto y largo que existan entre las imágenes que incluirá dicho *batch*. Un Ejemplo de esto se puede observar en la Figura 3.5, donde se muestra un ejemplo de *batch* de imágenes con *padding*.

Carga de modelo

En primer lugar, se realiza la carga del modelo convolucional, pudiendo ser VGG16 o ResNet, y dejando la posibilidad de utilizar otros modelos. También en esta etapa se determina que capa será la utilizada para extraer las características tanto de las imágenes en las que se pretende realizar la detección como de la *query*. En otras palabras, se selecciona la dimensión de los vectores que representarán los elementos de cada imagen.

Entrenamiento PCA

Una vez materializado el modelo, se realiza un entrenamiento *PCA* con 100 imágenes del *pool* con lo que se presume serían las características más significativas en cuanto a variabilidad. Es decir, si por ejemplo se seleccionó una capa con una profundidad de 256 características, es posible realizar un entrenamiento *PCA* para seleccionar las 64 características con mayor variabilidad en estas imágenes de entrenamiento. El modelo *PCA* es almacenado en el disco duro para su uso posterior.

Extracción de Características

Una vez realizado el entrenamiento *PCA*, se realiza mediante *batches* de tamaño 5 la extracción de características. Cada vector producido es filtrado por la selección de caracterís-

Figura 3.5: Ejemplo de un *batch* de tamaño 5 con *padding* aplicado.

ticas PCA y posteriormente una normalización L2 es aplicada, como se explica en la Sección 2.6.5. Esta normalización se utiliza con el propósito de hacer los vectores comparables entre imágenes.

Una vez se procesa cada *batch* de imágenes, las características extraídas son almacenadas en el disco duro para su utilización posterior. La idea detrás de este paso de almacenamiento es que elimina la necesidad de calcular todas las características nuevamente en caso de querer realizar búsqueda de más de un elemento.

Además de todo lo anterior, se implementó una funcionalidad para que, en caso de no tener memoria suficiente para procesar un *batch*, este redujera la cantidad de imágenes que contiene. En caso de que incluso un *batch* de una imagen muy grande no quepa en memoria, se integró la posibilidad de dividir esta imagen en dos y procesarla en dos partes.

3.5.2. Fase Online

Carga de modelo

Al igual que en la *fase offline*, se seleccionan tanto el modelo como la capa del modelo que se utilizarán. Teniendo esto, se cargan en memoria tanto este modelo convolucional como el modelo PCA seleccionador de características.

Carga y procesamiento de la *query*

Se indica cuál es la *query* cuya detección se busca en las imágenes del *pool* de imágenes. Esta pasa por el modelo convolucional seleccionado, la selección de características realizada con *PCA* y la normalización L2 de cada vector.

Generación de heatmaps

Al disponer tanto de las características de la imagen como de la *query*, se realiza la operación de correlación cruzada descrita en la Sección 2.6.4. Esto se hace en forma de *batch*, por lo que múltiples imágenes se operan de forma simultanea para generar tantos mapas de calor como imágenes en el *batch*. Un comentario con respecto a este proceso es que mientras más imágenes tenga un *batch*, el proceso de búsqueda será más rápido, pero se debe tomar en cuenta la capacidad de memoria de video del dispositivo.

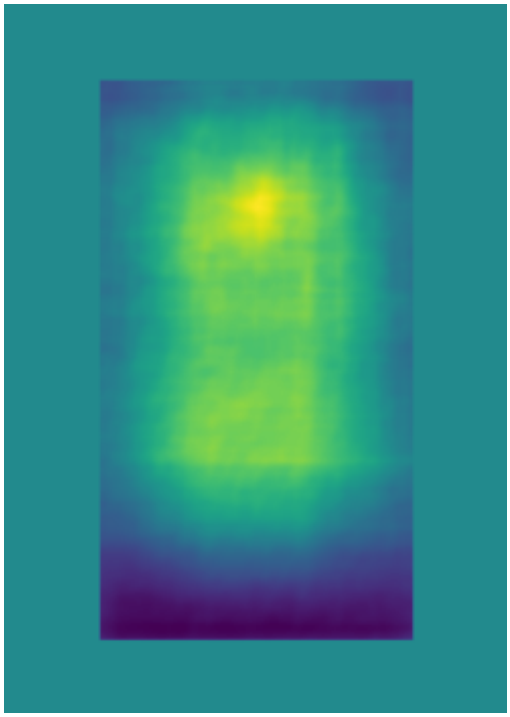
Un ejemplo de mapa de calor formado por una *query* y una imagen que la contiene en la Figura 3.6. En esta, se puede observar en color amarillo lo que sería el resultado máximo de la correlación entre ambas, y se es fácil advertir que tiene una posición similar al lugar donde se ubica la instancia a encontrar, sin ser necesariamente idéntica a la *query*.

Eliminación de bordes

Luego de obtener el heatmap de cada imagen de un *batch*, se realiza un redimensionamiento de este al tamaño original del *batch*, y posteriormente se aplica una eliminación de bordes del tamaño de la mitad de la query tanto a lo largo como a lo ancho. Esta eliminación se hace debido a los efectos que puede producir el *padding* utilizado al momento de la convolución. Es posible observar esta eliminación en la Figura 3.6c.

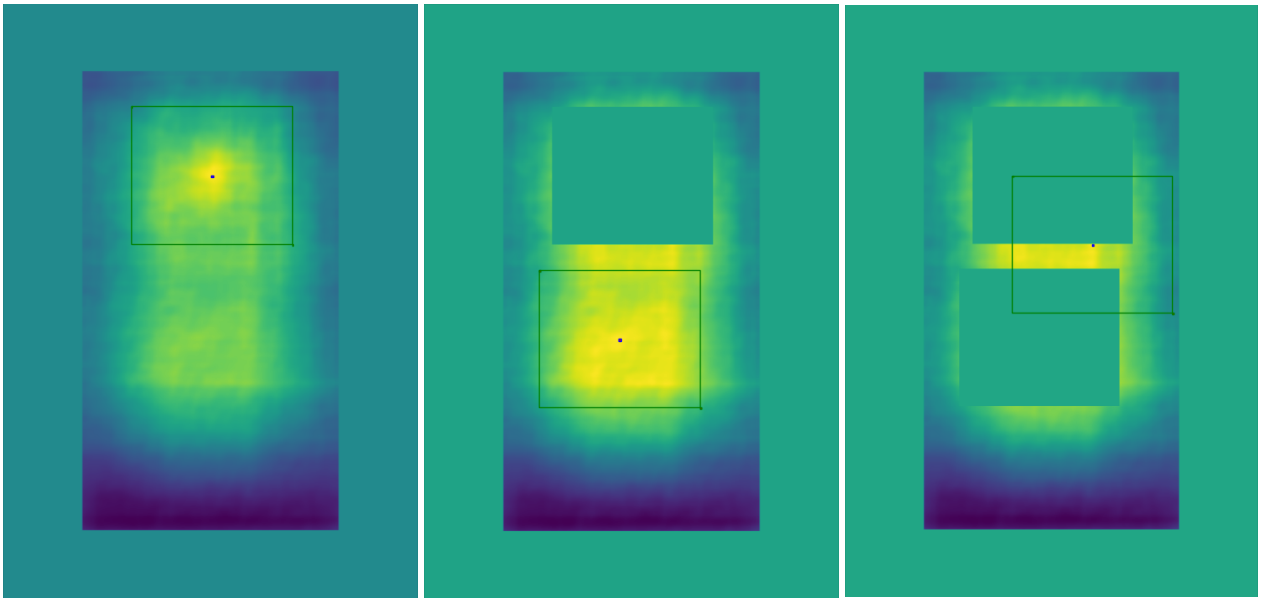
(a) *Query* de clase *bateau*.

(b) Imagen que contiene una *query* de la clase *bateau*.



(c) *Heatmap* generado por la correlación entre imagen y *query*.

Figura 3.6: Generación de *heatmap* de una imagen.



(a) Punto máximo. (b) 2° punto máximo. (c) 3° punto máximo.

Figura 3.7: Proceso de búsqueda de puntos máximos en *heatmap*.

Búsqueda Iterativa

Una vez obtenido un heatmap, se realiza una búsqueda iterativa que consiste en rescatar los puntos máximos de la *query*, eliminando con valor cero los alrededores de este punto para la siguiente búsqueda. En particular, en este método se rescatan 15 puntos. El punto máximo se considera como el centro de una detección y se considera un *bounding box* del tamaño de la *query* como una detección. Además se considera el valor de este punto como el valor de "confianza". Este proceso se puede observar en la Figura 3.7

3.6. Mejoras y variaciones

Para incorporar algunas mejoras, se pensó en las posibles debilidades de las que sufre este método. Por este motivo, las siguientes secciones explican algunas de las mejoras incorporadas para compensar estas debilidades.

3.6.1. Selección de escala

Debido a la forma en la que funcionan las redes convolucionales y al *pooling* que se realiza en ellas, cuando se procesa una imagen y a medida que avanza por estas capas, existe una reducción de tamaño en las características producidas. A modo de ejemplo, la red VGG16 aplica *max pooling* luego de cada bloque, por lo tanto, la dimensión se reduce a la mitad luego de cada bloque, como se puede ver en la Figura 2.5.

Por este motivo, se puede intuir que las características de una capa determinada representan una cierta zona de la imagen original. Para esto se acuñará el término zona de atención. Se puede ver en la Figura 3.8 como cada celda (lo que sería un vector de características) representa lo que una zona de la imagen original, dependiendo de la capa convolucional de

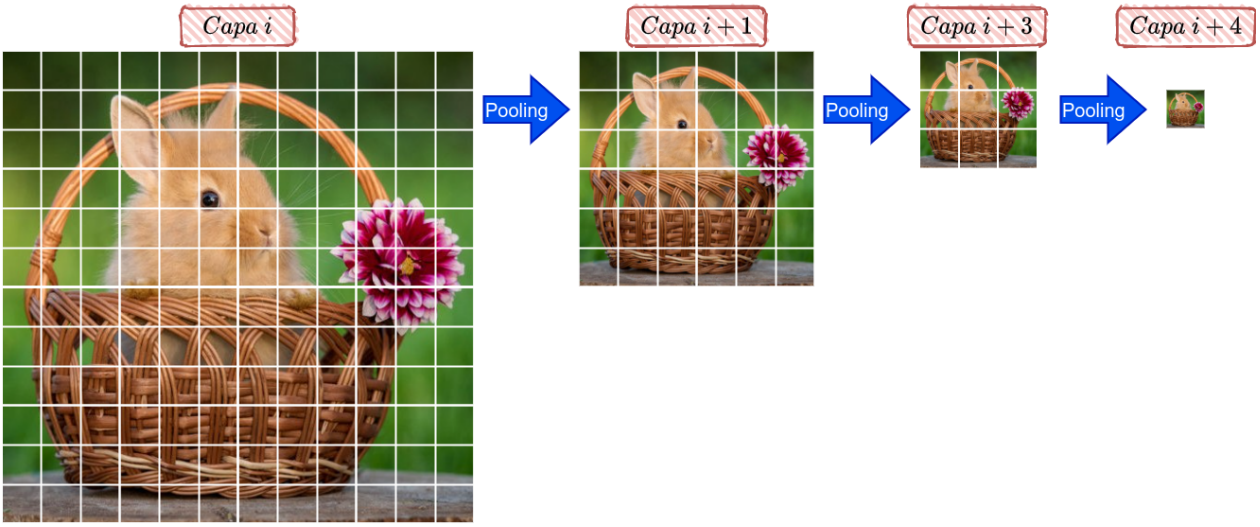


Figura 3.8: Representación del efecto de *pooling* y zona de atención

la que se extraigan las características.

Debido a esto, se decidió luego de cierta cantidad de experimentación que las *queries* que no superaran los 100 pixeles en alguno de sus lados utilizarían *features* que reducen la dimensión en 4, mientras que *queries* más grandes utilizarían *layers* que reducen en una escala de 8.

La selección se hace con el propósito de representar de mejor manera las *queries* pequeñas y grandes en la imagen, además de agilizar el proceso. Una *query* pequeña con una reducción muy grande en escala puede llegar a perder gran parte de la información de la imagen, por lo que se dificultará su búsqueda. Mientras tanto, una *query* de gran tamaño y con una reducción de escala más pequeña será mucho más costosa (computacionalmente) al realizar la operación de convolución.

3.6.2. Transformaciones

Además de la dificultad de la reducción de escala que se intenta solucionar con la selección, también se encuentra el problema de las distintas posiciones de rotación y diferentes tamaños en los que se puede encontrar la *query* deseada. Por este motivo, se decidió incorporar a la búsqueda de las siguientes transformaciones lineales a la *query*, generando *features* para cada una de ellas:

- Flip horizontal
- Rotación en 90°
- Rotación en 180°
- Rotación en 270°
- Corte central: Se toma un corte central de la imagen y se re-escala al tamaño de la *query* original. Con ella se busca encontrar instancias más grandes de la *query*.
- Escalado a la mitad: Generar la misma *query* pero con la mitad de su tamaño (sólo para *queries* ambos lados con largo mayor a 300). Con ella se buscan encontrar instancias

más pequeñas de la *query*. Luego de obtener este escalado, también se incorporan las rotaciones antes mencionadas a esta imagen mas pequeña.

- Escalado a un cuarto: Generar la misma *query* pero con la un cuarto de su tamaño (sólo si ambos lados son mayores a 500 pixeles). Con ella se buscan encontrar instancias aún más pequeñas de la *query*. A esta reducción también se le aplican las rotaciones mencionadas y se realiza la búsqueda con ellas.

Ejemplos de cada una de estas transformaciones se pueden encontrar en la Figura 3.9.

Cada una de estas transformaciones genera un *heatmap* similar a lo mostrado por la Figura 3.6c. Luego, en cada una se realiza la búsqueda de puntos máximos almacenando tanto el *bounding box* como el valor del punto. Finalmente se comparan todos los *bounding box* para comprobar los cruces que existen, y si existe una intersección grande entre dos *bounding boxes* (intersección sobre la unión mayor a 0.5), se conserva aquel que tenga un mayor valor máximo.

Es importante mencionar que el incorporar todas estas convoluciones incrementa la profundidad (en canales), y por lo tanto la cantidad de tiempo empleado en realizar la búsqueda.

3.7. Procedimiento para Recuperación de imágenes

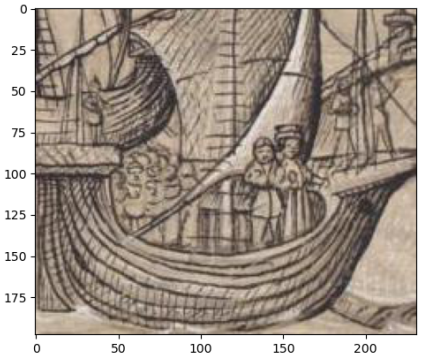
El procedimiento para recuperación de imágenes en lugar de detección es idéntico al descrito en la sección 3.5. Esta tarea simplemente se deriva de las detecciones obtenidas ordenadas por índice de confianza. Desde estas se extraen las imágenes en las que se encontró una detección sin repetición.

El procedimiento simplificado para la obtención de resultados en recuperación se debe a que el planteamiento de esta metodología es principalmente para detección, pero comprobar su efectividad en recuperación no dificulta mayormente el proceso.

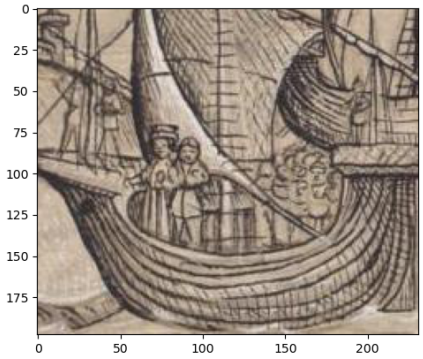
3.8. Medición de resultados

3.8.1. Detección

Para medir los resultados de la metodología propuesta se utilizó la precisión promedio, explicada en la sección 2.6.7. El criterio para considerar una detección como correcta es calcular la intersección sobre la unión con las anotaciones correctas de cada *dataset*. Si el IoU es igual o mayor a 0.5 la detección se considera como correcta. Además, si una anotación ya ha sido marcada como detectada y se detecta nuevamente con un IoU mayor a 0.5, es marcada como falso positivo (debido a que se debería a una repetición de una misma instancia, lo que no es correcto). Para efectos de simplicidad del cálculo, se consideran las primeras mil detecciones ordenadas por su valor en la correlación (a lo que se llama "confianza") para cada *query* para el cálculo del *mAP*. Con esto se genera la curva de *precisión/recall* y se realiza el cálculo del área bajo esta curva para obtener la precisión de la búsqueda.

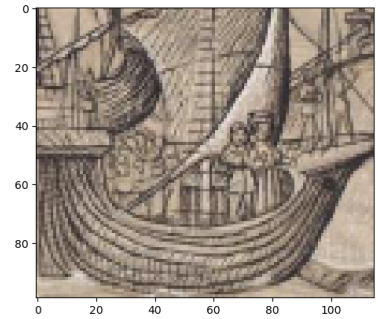


(a) *Query* original.

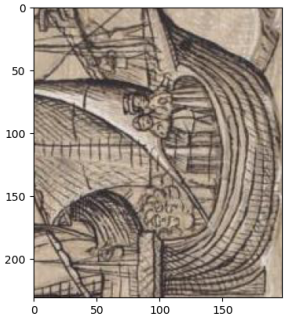


(b) *Query* con *flip* horizontal.

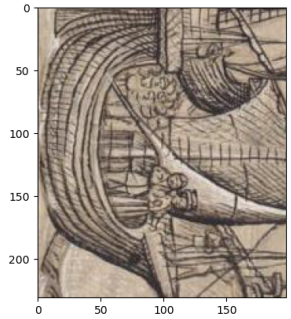
(c) Corte central de la *query* original.



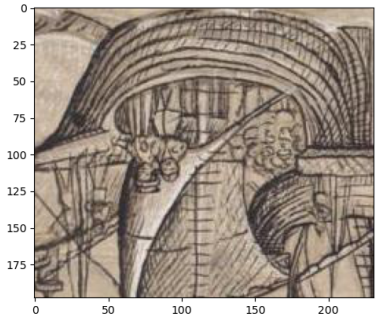
(d) *Query* con redimensionamiento a la mitad.



(e) *Query* original rotada en 90°.



(f) *Query* original rotada en 270°.



(g) *Query* original rotada en 180°.

Figura 3.9: Transformaciones aplicadas a la *query*

Figura 3.10: Ejemplo de tipos de *queries*. Los grupos son: Grande y no cuadrada (superior izquierda), grande y cuadrada (superior derecha), pequeña y no cuadrada (inferior izquierda) y pequeña y cuadrada (inferior derecha) [26]

3.8.2. Recuperación

En esta tarea también se utilizó la precisión promedio para juzgar el desempeño. En este caso, realizar la medición es más simple que en detección. En este caso, se ordenan para cada *query* las imágenes detectadas por el nivel de "confianza" de sus detecciones. Luego, en orden, se consideran como verdaderas positivas las imágenes que contienen realmente a la *query* buscada, mientras que si la imagen no contiene a la query, se marca como falso positivo. En este caso, se consideran para el cálculo las primeras mil imágenes detectadas, se calcula la curva de *precisión/recall* y se calcula también el área bajo la curva.

3.8.3. Agrupación de Queries

Para extender el análisis, se siguió el protocolo de agrupación de *queries* mostrado en [26], en el cual se clasifican por tamaño y por forma. Esto consideró el alto h y el ancho (w).

Para realizar la separación de las *queries* en pequeñas y grandes, se utilizó el logaritmo del tamaño h*w y se calculó un punto intermedio (Ecuación 3.1). De acuerdo con este, se clasificaron las *queries* como pequeñas y grandes.

$$\begin{aligned} Size_{cut} &= middle((10^{(log_{10}(h_{middle} \cdot w_{middle}))}/10)) \\ Grande\ si\ log(size) &>= Size_{cut} \\ Pequea\ si\ log(size) &>= Size_{cut} \end{aligned} \tag{3.1}$$

Para clasificar una *query* como cuadrada o no cuadrada se utilizó el valor absoluto del ratio entre lados. Se utilizó una tolerancia de 0.2 de forma que si se considera el ratio entre lados de una *query* q como ar_q, corresponde a una *query* cuadrada si la $|ar_q - 1| <= 0{,}2$. Un ejemplo que muestra las cuatro categorías se puede observar en la Figura 3.10.

Capítulo 4

Resultados y Discusión

En este capítulo se presentaran los resultados obtenidos junto al análisis de los mismos. Para obtener una muestra de los resultados, en primer lugar se realizaron pruebas con el set de datos DocExplore debido a que es una tarea más simple que *Flickrlogos*. Esto se debe a que existen menos variaciones de escala, distorsión, rotaciones y fondo. Los primeros experimentos correspondieron a la prueba con un subconjunto de *queries* (una de cada clase) para comprobar la calidad de las detecciones en distintas capas y la capacidad de recuperar imágenes correctamente, con una cantidad diferente de componentes principales.

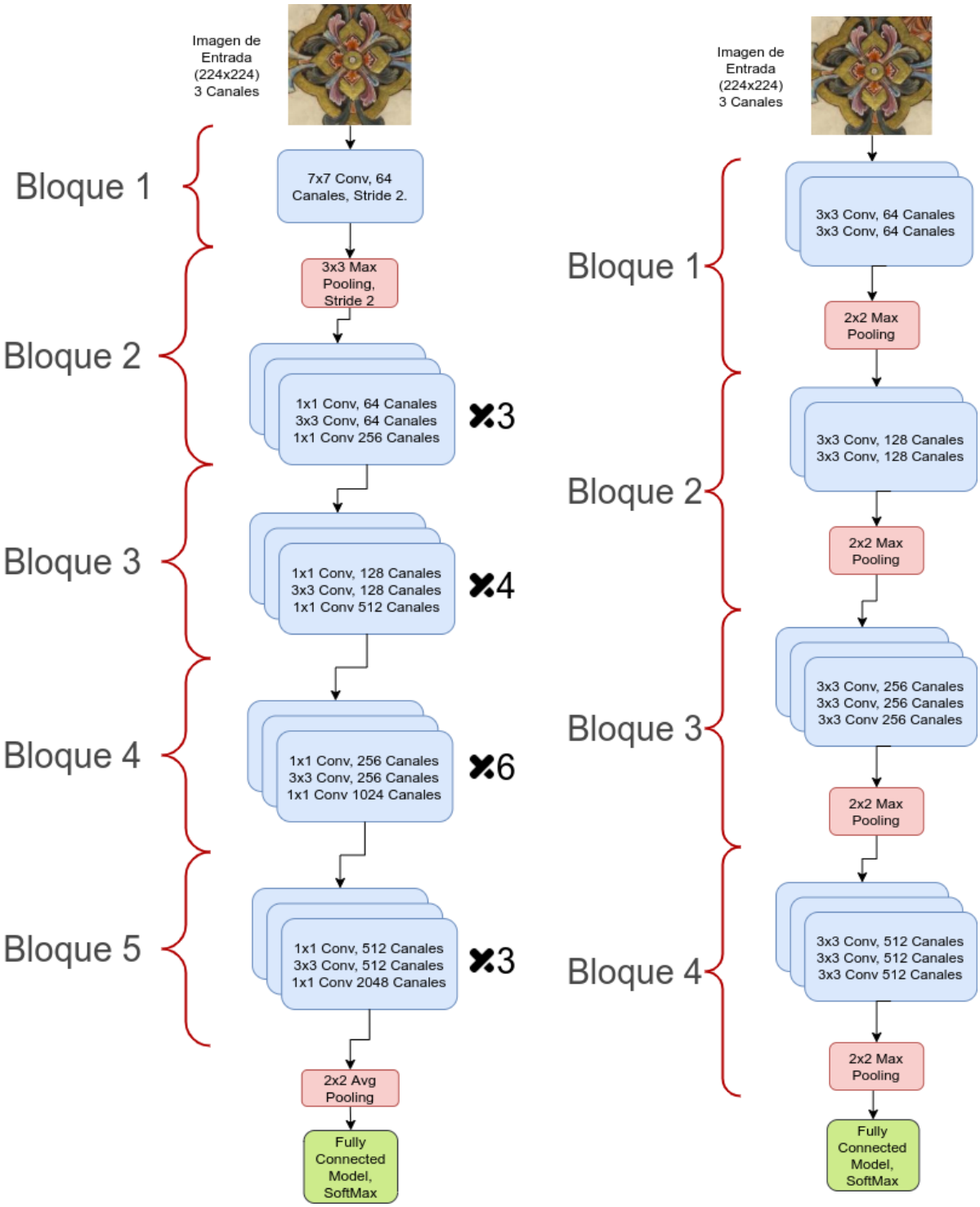
Una vez realizadas las pruebas iniciales y la obtención de sus resultados, se escogieron 3 modelos en base a los resultados para experimentar sobre los *dataset* completos. Esta elección se justifica tanto en las limitaciones de tiempo como de hardware, pues la estas pruebas toman una cantidad considerable de tiempo debido al número de *queries* por *dataset*.

A continuación, en la Tabla 4.1 se adjuntan las capas utilizadas, de donde son extraídas y la profuyndidad de las características que produce cada una. Además, se adjunta en la Figura 4.1 un resumen de las estructuras utilizadas, con el fin de esclarecer la notación.

Modelo	Capa	Can. características	Nombre en Keras Applications
ResNet50	Bloque 3	128	conv2_block3_out
ResNet50	Bloque 4	256	conv3_block4_out
VGG16	Bloque 3	256	block3_conv3
VGG16	Bloque 4	512	block4_conv3

Tabla 4.1: Capas utilizadas y profundidad de las características producidas por cada una de ellas.

Finalmente, una vez se dispuso de las detecciones para todas las *queries*, se continuó con el cálculo de precisión promedio para detección y recuperación de imágenes según las métricas descritas en 2.6.7.



(a) Resnet

(b) VGG16

Figura 4.1: Esquemas resumen para ambas estructuras utilizadas.

4.1. Resultados DocExplore

4.1.1. Resultados de una *query* por clase

En este set de datos, se utilizaron 35 *queries* en total, una por cada clase. Estas *queries* son las mismas para todos y cada uno de los experimentos.

De acuerdo a lo mostrado en la Tabla 4.2, que expresa los resultados promedio para cada configuración, es claro que el modelo que utiliza simplemente ResNet como extractor de características no realizó detecciones de buena manera. En comparación, la red VGG16 demostró resultados prometedores debido a que en la mayoría de sus variaciones obtuvo una precisión promedio elevada. En general el rendimiento de VGG16 en detección es superior a ResNet, destacando la capa correspondiente al *Bloque 3*, que consiguió los mejores resultados de las capas utilizadas directamente.

Se observó que la selección de escala mejoró marginalmente los resultados en comparación con sólo utilizar la el *Bloque 3*, incorporando la capa asociada al *Bloque 4* para *queries* de mayor tamaño. También se establece que su implementación vale la pena hasta cierto punto debido a que al utilizar una capa con una reducción mayor de tamaño de los vectores, el tiempo de procesamiento disminuye considerablemente.

Se destaca además que la aplicación de transformaciones (que incluye selección de escala) disminuye la precisión promedio y aumenta el tiempo de procesamiento, por lo que resulta ser poco útil en esta tarea.

Modelo	Capa	Comp. Principales	mAP Detección	mAP Recuperación	Tiempo Total [s]	Imágenes por Segundo
ResNet	Bloque 3	32	0.363	0.437	385.2	3.89
	Bloque 4	32	0.078	0.140	**170.7**	8.78
	Sel. de Escala	32	0.188	0.273	199.3	7.52
VGG16	Bloque 3	16	0.668	0.749	269.3	5.56
		32	0.724	0.803	380.3	3.94
		64	**0.735**	**0.819**	600.8	2.49
	Bloque 4	16	0.519	0.571	169.6	8.84
		32	0.564	0.636	172.9	8.67
		64	0.602	0.685	181.2	8.27
	Sel. de Escala	16	0.679	0.752	183.3	8.18
		32	0.729	0.801	200.8	7.47
		64	**0.741**	**0.815**	236.9	6.33
	Transf.	32	0.664	0.739	397.1	3.77

Tabla 4.2: Resultados promedio en DocExplore para una query por clase.

Los resultados obtenidos en detección y recuperación para estas 35 *queries* en todas las configuraciones testeadas se pueden observar en las Figuras 4.2 y 4.3. Es posible apreciar que existen ciertas *queries* que se detectan de buena forma en muchas configuraciones, por ejemplo *obj_3* y *obj_42*. Por el contrario otras *queries* tienen resultados muy malos a lo largo de todos los modelos, como los separadores *simple_sep*, *double_sep* y *triple_sep*.

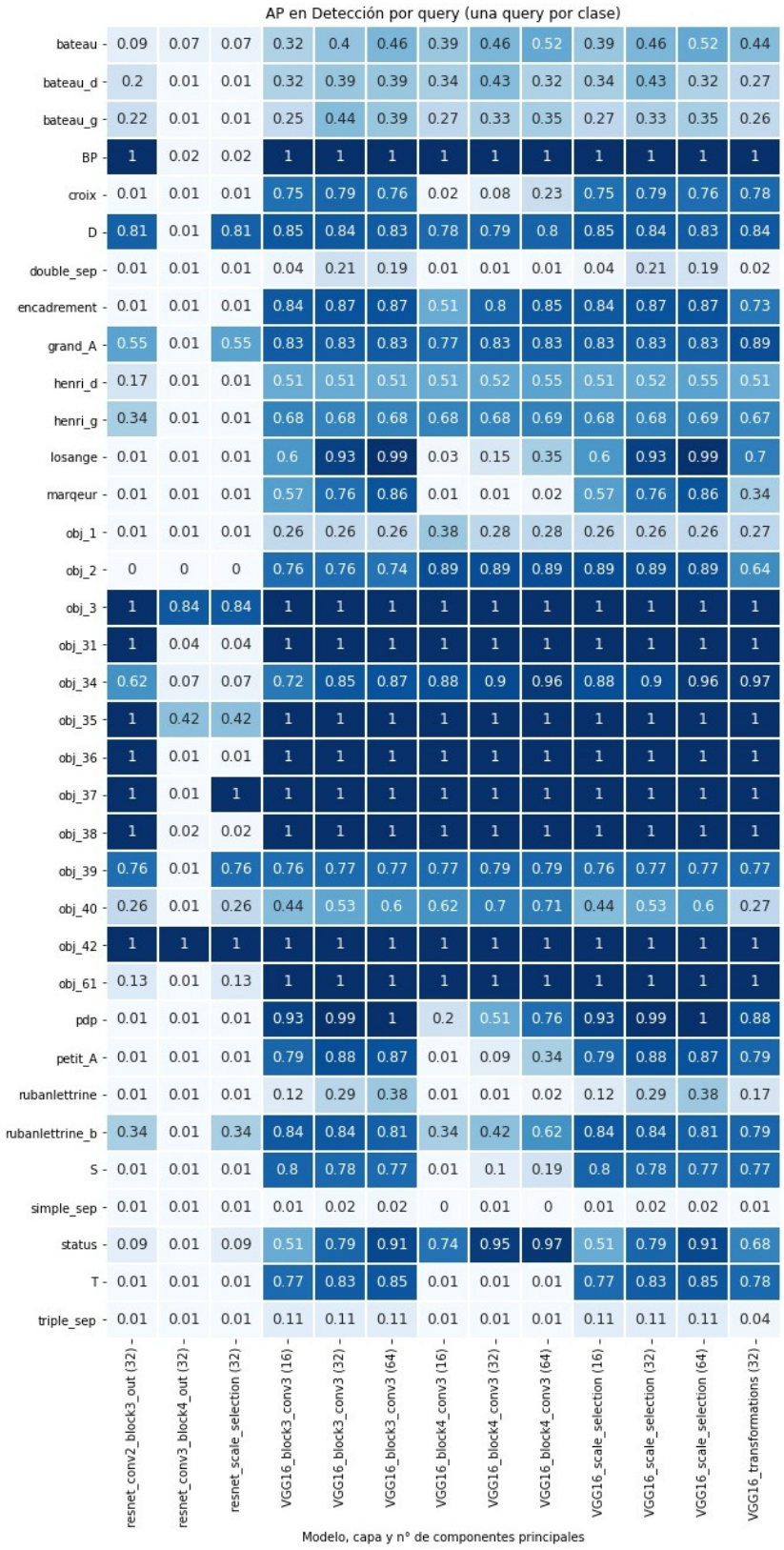


Figura 4.2: Resultados en detección para DocExplore por clase, para una *query* por clase.

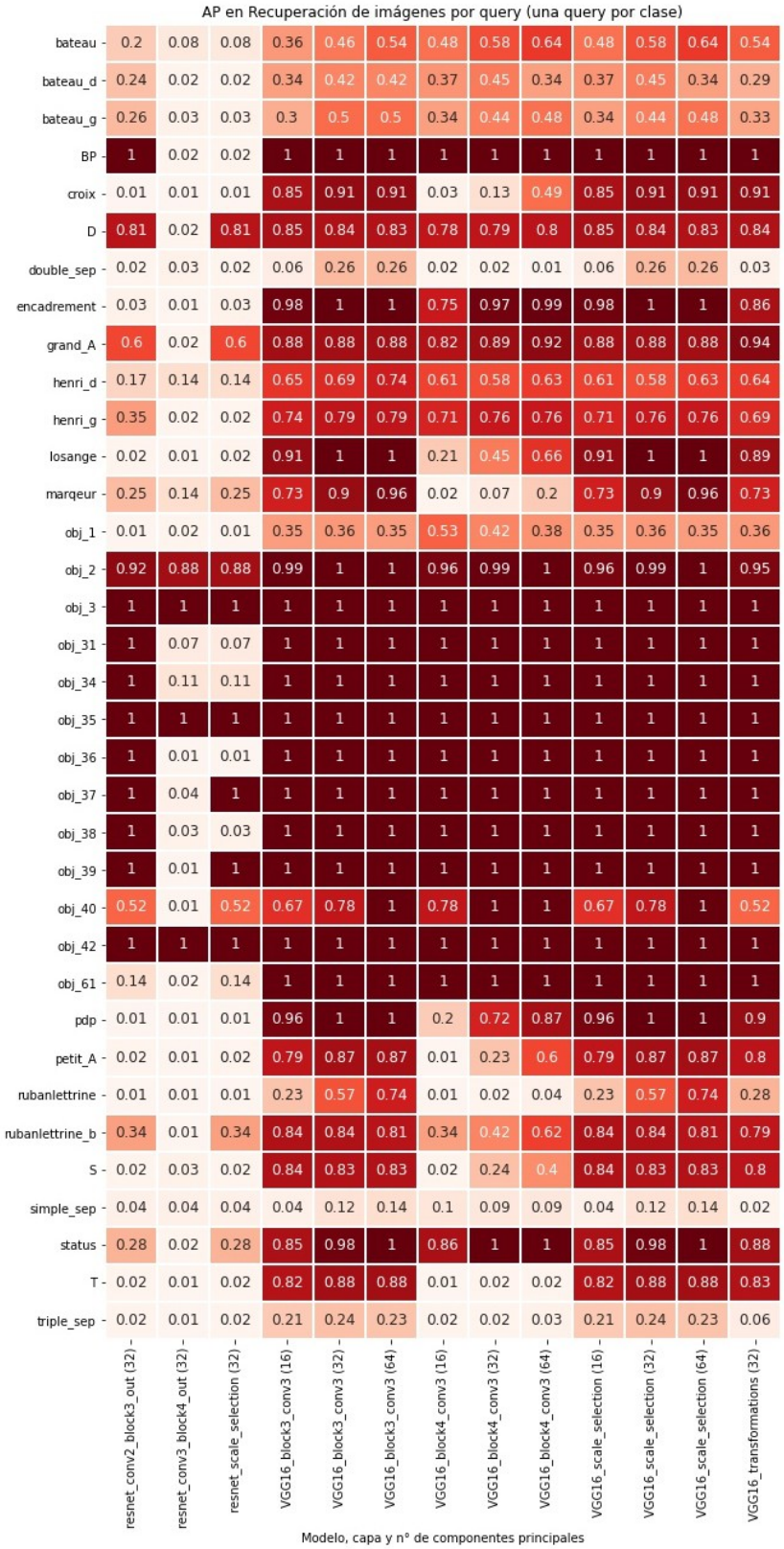


Figura 4.3: Resultados en recuperación para DocExplore por clase, para una *query* por clase.

Figura 4.4: Query *163.jpg* de la clase *obj_43*.

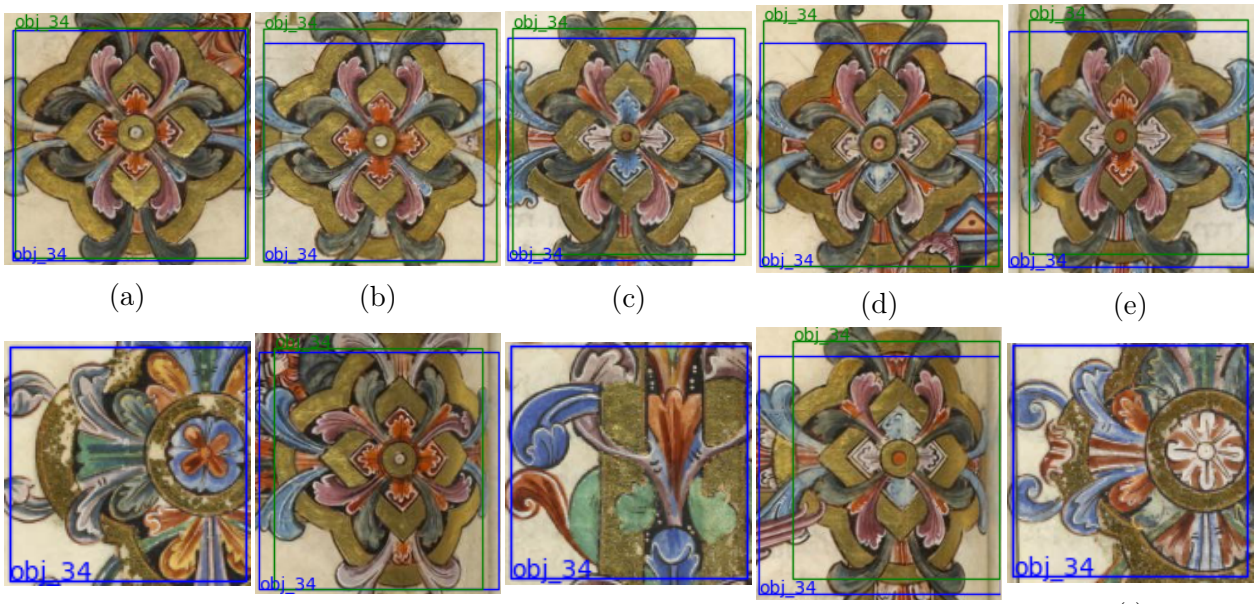


Figura 4.5: Resultado de las primeras 10 detecciones para la *query 163.jpg* de clase *obj_34* obtenido por el modelo *VGG16* utilizando la capa asociada a la salida del *Bloque 3* con 64 componentes principales.

Un ejemplo de detecciones, la Figura 4.5 muestran con un *bounding box* azul las 10 detecciones con mayor índice de confianza para la *query 163.jpg* (mostrada en la Figura 4.4) perteneciente a la clase *obj_34* en una de las configuraciones. Además, con un *bounding box* verde se representan las anotaciones que corresponden a la misma clase de la *query* (lo que se llamaría *ground truth*). Si no existe un *bounding box* de color verde, esto implica que no existe una anotación en dicha zona.

La precisión promedio que corresponde a la búsqueda de la *query* se puede observar en la Figura 4.2 y corresponde a *0.87*, por lo que es una detección bastante buena pero cuenta con errores. Si se observan detenidamente las detecciones que corresponden a errores, se puede comprobar a pesar de no ser correctas existe cierto parecido visual entre la *query* y las detecciones realizadas, por lo que se puede inferir que las anotaciones manuales no son necesariamente perfectas.

Debido a que esta la detección es una tarea un poco más simple que la detección de imágenes, puesto que basta con determinar si el elemento de búsqueda se encuentra en la imagen o no, los resultados reflejan en general mayor precisión que en detección, como se puede comprobar en la Tabla 4.2. Por este motivo, algunas *queries* presentan una mayor precisión individualmente también, por ejemplo, la *query 163.jpg* tiene en este caso una precisión de 1 (sin errores), como se puede ver en la Figura 4.3 y se puede comprobar en el ejemplo de *ranking* mostrado en la Figura 4.6, donde se observan las primeras 9 imágenes recuperadas.

4.1.2. Resultados de todas las queries

Luego de realizar la experimentación de una *query* por clase, se decidió realizar el experimento de búsqueda sobre el *dataset* completo con el propósito de comparar el método con el estado del arte. Para esto, se seleccionaron configuraciones en base a los resultados para una *query* tomando en consideración tanto la precisión obtenida el tiempo de procesamiento.

Para efectos de ahorro de tiempo, no se eligieron las configuraciones con los mejores resultados en precisión promedio. En su lugar, se seleccionaron las opciones con una segunda mejor precisión y menor profundidad de las características debido a que en general, según lo mostrado en la tabla 4.2, existe una pérdida no mayor de desempeño pero un gran ahorro de tiempo por búsqueda. Además de esto, se decidió incorporar a las pruebas la estrategia de transformaciones para analizar su rendimiento.

En la tabla 4.3 se presentan los resultados generales obtenidos para las tres configuraciones seleccionadas.

Modelo	Capa	mAP		Tiempo Total	Imágenes por segundo
		Detección	Recuperación		
VGG16	Bloque 3	**0.571**	**0.705**	230.9	6.49
	Selección de Escala	**0.571**	**0.705**	**202.1**	**7.42**
	Transformaciones	0.455	0.5718	361.3	4

Tabla 4.3: Resultados de DocExplore para todas las *queries*.

Con el fin de profundizar el análisis, se presentan los resultados obtenidos según la agrupación planteada en 3.8.3 en la Tabla 4.4. Es claro que los modelos varían en su precisión, pero que para esta tarea, los modelos predominantes son el que utiliza la capa del *Bloque 3* y la selección de escala. El modelo que utiliza transformaciones queda detrás de ellos en prácticamente todas las tareas testeadas.

Finalmente, se presentan la precisión promedio para ambas tareas para todas las queries, agrupadas por clase. Esto se puede ver en las Figuras 4.7 y 4.8. Se reitera en estas figuras que existieron clases particulares que tuvieron notables dificultades en su detección, haciendo notar *simple_sep*, *doubl_sep* y *triple_sep*. Esto puede deberse a su que su forma es no cuadrada y pequeña, que es coincidentemente la categoría de peor desempeño mostrada en la Tabla 4.4.

Figura 4.6: Resultado de las 10 primeras imágenes recuperadas para la *query 163.jpg* de clase *obj_ 34* obtenido por el modelo *VGG16* utilizando la capa asociada a la salida del *Bloque 3* con 64 componentes principales.

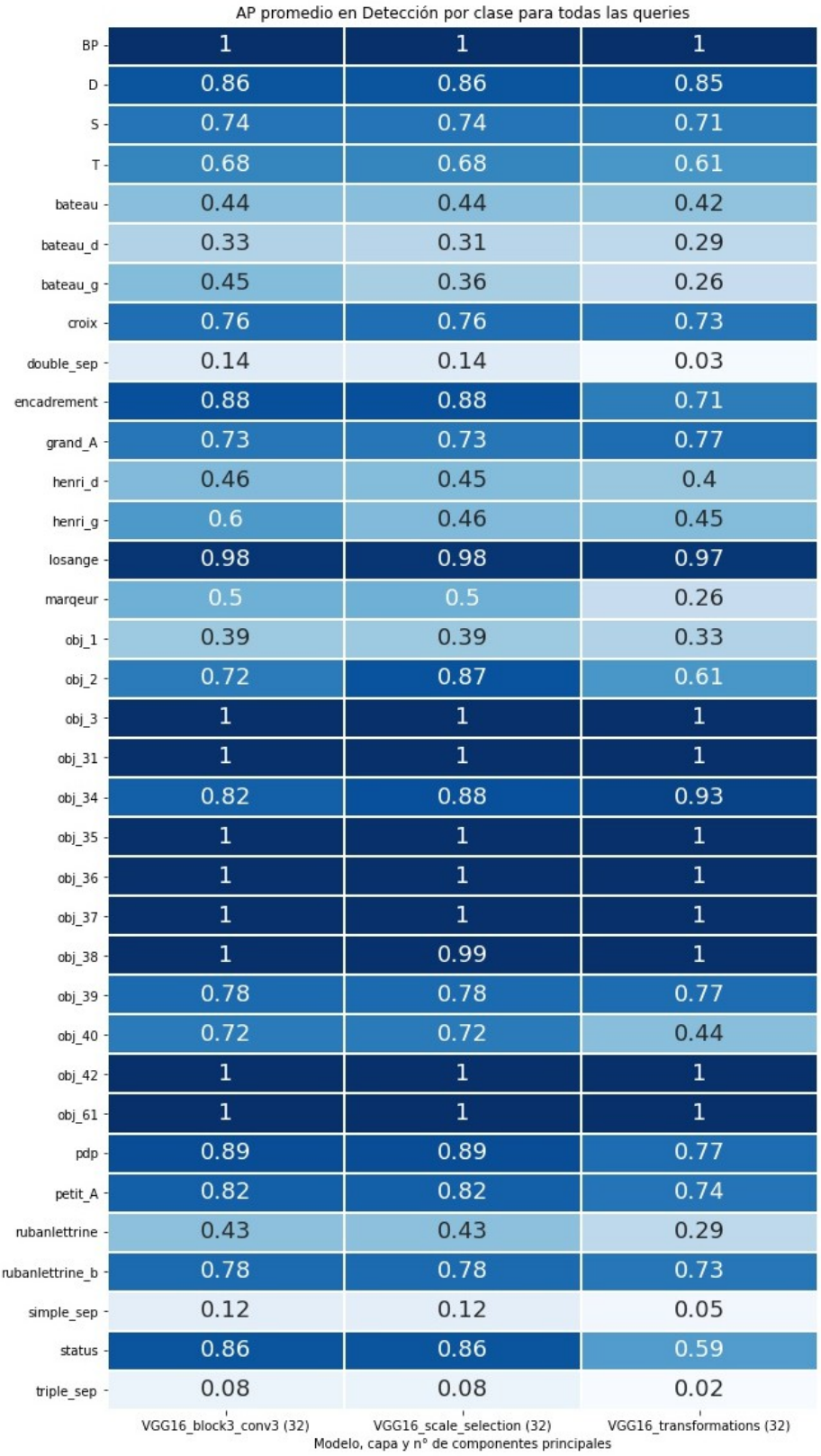


Figura 4.7: Resultados promedio en Detección para DocExplore por clases. Búsqueda de todas las *queries*.

Figura 4.8: Resultados promedio en Recuperación para DocExplore por clases. Búsqueda de todas las *queries*.

Forma	Tamaño	Prof. Queries	mAP Deteccion			mAP Recuperacion		
			Bloque 3	Sel. de Escala	Transf.	Bloque 3	Sel. de Escala	Transf
Cuadrada	Grande	33	0.84	0.849	**0.851**	0.895	**0.901**	0.885
	Pequeña	187	**0.852**	0.849	0.817	**0.910**	0.908	0.871
No cuadrada	Grande	21	0.648	**0.688**	0.533	0.770	**0.782**	0.698
	Pequeña	1206	**0.519**	0.518	0.387	**0.667**	**0.667**	0.515

Tabla 4.4: Resultados en DocExplore por tipo de query de acuerdo a protocolo de evaluación planteado en [26]. Todas corresponden al modelo VGG16 con 32 componentes principales.

Comparación estado del arte

A continuación, se presenta una comparación entre los resultados disponibles del estado del arte para este set de datos con el modelo propuesto en esta investigación que tuvo el mejor desempeño, el cual corresponde a selección de escala. Este se puede ver en la Tabla 4.5. Se observa que si bien el modelo obtenido no supera el máximo, si se coloca como segundo en rendimiento tanto en detección como en recuperación.

Modelo	mAP	
	Detección	Recuperación
Este modelo (Selección de Escala)	**0.571**	**0.705**
En et Al.[25]	0.1569	0.5801
Wiggers et Al.[22]	0.1740	0.3860
Úbeda et Al.[26]	0.2720	0.5770
Curi et Al.[31]	0.6442	0.8131

Tabla 4.5: Comparación de resultados entre mejor modelo obtenido y el estado del arte.

Además, se presenta la comparación de los resultados en la configuración de grupos de las queries en las Tabla 4.6. Se puede observar que si bien globalmente los resultados del modelo implementado no superan el trabajo de Curi[31], se llega a un punto muy similar tanto en detección como en recuperación, haciendo hincapié en que los resultados en recuperación para queries no cuadradas fueron superados por este. Esto hace sentido pues la implementación a la que se llegó finalmente se basa y apoya mucho en este trabajo.

4.2. Resultados Flickrlogos 47

Luego de realizar los experimentos en DocExplore, se pretendió abordar la tarea más compleja de detección de logos en imágenes más complejas y cercanas a la realidad. Los experimentos realizados sobre este *dataset* fueron hechos con las configuraciones que tuvieron mejor desempeño en la tarea de DocExplore, y corresponden las mencionadas en la Sección 4.1.2.

Los resultados para este *dataset* se presentan en la Tabla 4.3. Como es posible observar,

Forma	Tamaño	mAP Detección				mAP Recuperación			
		En et Al. [25]	Ubeda [26]	Curi [31]	Sel. de Escala	En et Al. [25]	Ubeda [26]	Curi [31]	Sel. de Escala
Cuadrada	Grande	0.546	0.681	**0.880**	0.865	0.881	0.749	**0.937**	0.782
	Pequeña	0.102	0.546	**0.858**	0.845	0.801	0.742	**0.927**	0.667
No cuadrada	Grande	0.405	0.509	**0.752**	0.688	0.701	0.660	0.826	**0.911**
	Pequeña	0.149	0.214	**0.603**	0.518	0.535	0.459	0.792	**0.905**

Tabla 4.6: Comparación con estado del arte en DocExplore con el mejor modelo obtenido de acuerdo a protocolo de evaluación planteado en [26].

los resultados son bastante inferiores a los obtenidos en DocExplore. Esto se debe a que la detección de logos en ambientes realistas es una tarea mucho más compleja.

Modelo	Capa	mAP		Tiempo Total [s]	Imágenes por segundo
		Detección	Recuperación		
VGG16	Bloque 3	0.129	0.229	188.5	4.41
	Selección de Escala	0.138	0.245	**118**	**7.05**
	Transformaciones	**0.165**	**0.289**	227.4	3.66

Tabla 4.7: Resultados de Flickrlogos 47 para todas las *queries*. Todas corresponden al modelo VGG16 con 32 componentes principales.

Nuevamente, se realiza el análisis tanto de detección como de recuperación por tipo de query. Se pueden observar estos resultados en la Tabla 4.8.

Forma	Tamaño	Prof. Queries	mAP Detección			mAP Recuperación		
			Bloque 3	Sel. de Escala	Transf.	Bloque 3	Sel. de Escala	Transf
Cuadrada	Grande	146	0.129	**0.151**	0.13	0.227	0.263	**0.297**
	Pequeña	446	0.151	0.159	**0.20**	0.257	0.271	**0.327**
No cuadrada	Grande	302	0.116	**0.143**	0.129	0.212	0.263	**0.291**
	Pequeña	1042	0.123	0.123	**0.164**	0.232	0.237	**0.283**

Tabla 4.8: Resultados en FlickLogos 47 por tipo de query de acuerdo a protocolo de evaluación planteado en [26]. Todas corresponden al modelo VGG16 con 32 componentes principales.

Se presentan además la precisión promedio para ambas tareas para todas las *queries*, agrupadas por clase. Esto se puede ver en las Figuras 4.9 y 4.10 .

Se decidió analizar el las *queries* con mejor y peor desempeño de cada clase en el mejor modelo obtenido; VGG16 con 32 componentes principales y aplicando transformaciones. Esto se realizó debido a que dado el planteamiento de la metodología existe una clara dependencia de la *query* ingresada al modelo. Para esto, se decidió comparar tres escenarios:

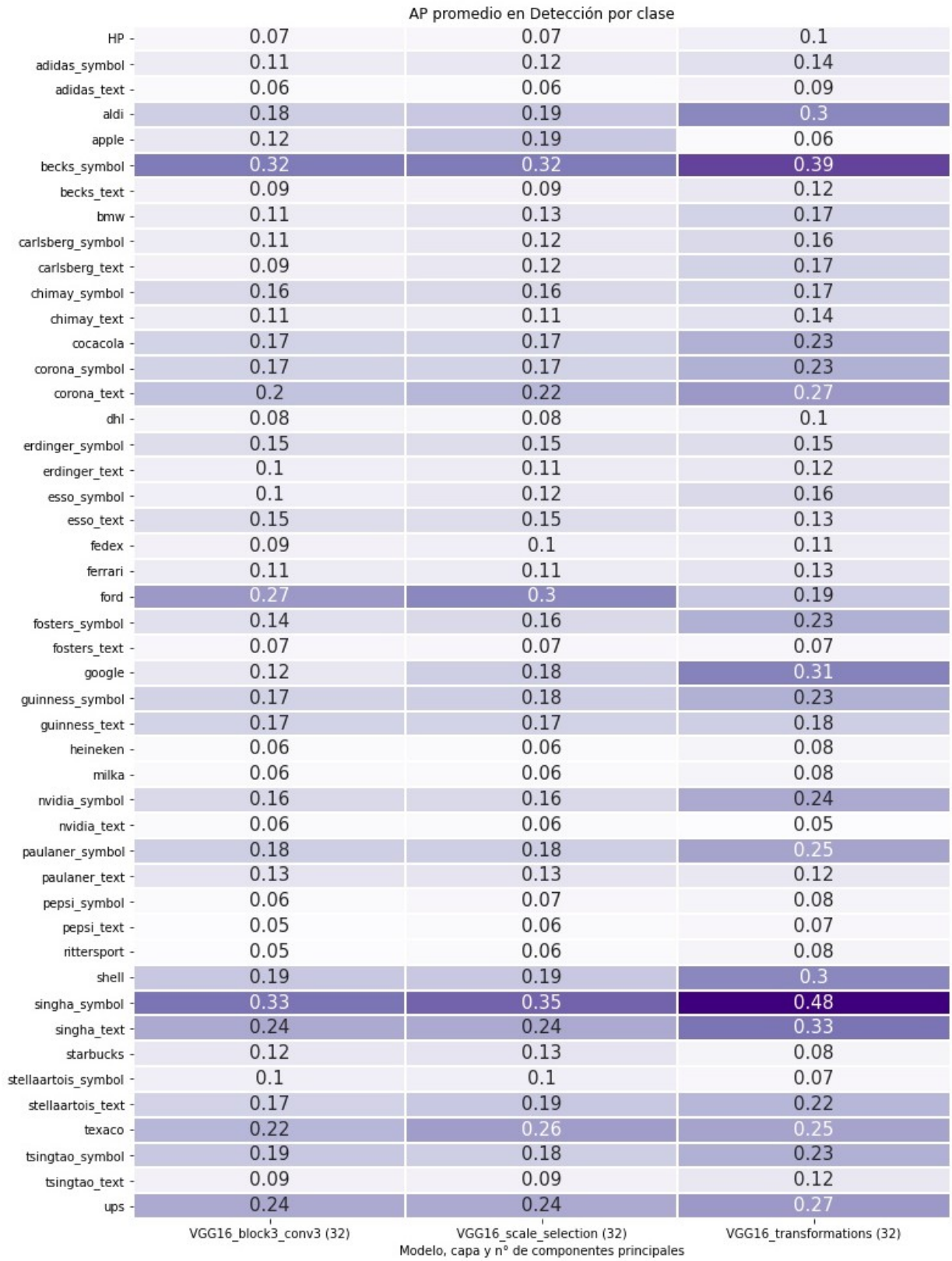


Figura 4.9: Resultados para detección en Flickrlgos 47 por clase, para todas las *queries*.

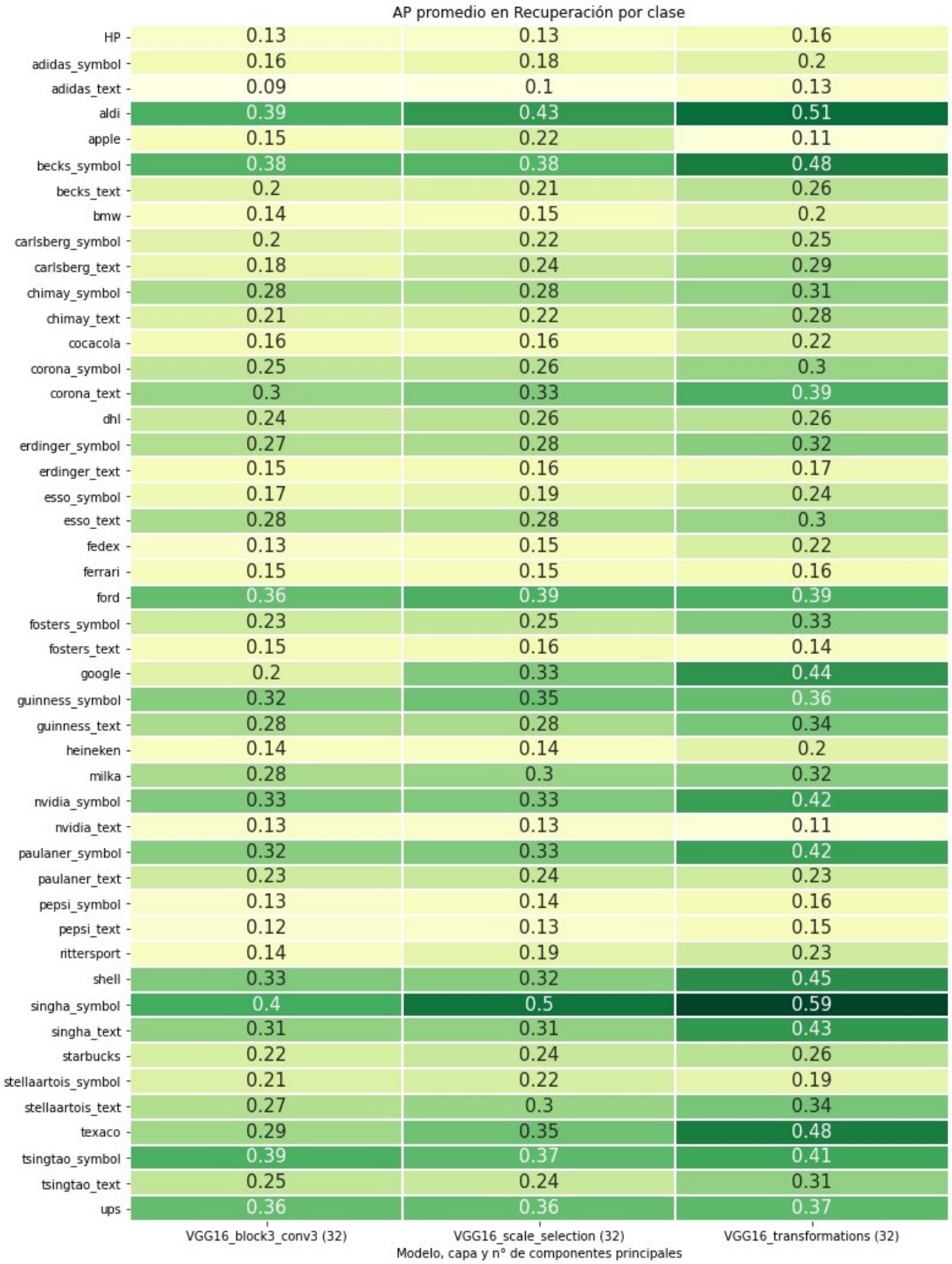


Figura 4.10: Resultados para recuperación en Flickrlgos 47 por clase, para todas las *queries*.

- Caso Pesimista: Se recolectaron las peores *queries* de cada clase, y luego se promedió su precisión. Se calcularon además características asociadas a ellas.
- Caso Optimista: Se recolectaron en esta instancia las *queries* que tuvieron un mejor desempeño en precisión y también se calcularon estadísticas.

Los resultados para este estudio se presentan en la Tabla 4.9. Es claro que la diferencia entre ambos casos es abismal, por lo que se interpreta que la metodología depende considerablemente de la *query* que se utilice.

Caso	mAP Promedio		Promedio [px]			Mediana [px]		
	Detección	Recuperac.	Tamaño	Lado Mayor	Lado Menor	Tamaño	Lado Mayor	Lado Menor
Optimista	**0.404**	**0.610**	10177.9	125	70.82	6097	107	59
Pesimista	**0.033**	**0.033**	19455.4	117.5	57.3	1088	56	20

Tabla 4.9: Resultados promedio para búsqueda en FlickrLogos para la mejor y peor *query* por clase.

Todo esto con el propósito de comprobar qué tipo de *queries* se asocian a cada caso. Es decir, ver si influye su tamaño, su forma o su calidad. Se presentan en las Figuras 4.12 y 4.11.

Desde un punto de vista cualitativo, se puede observar que las *queries* asociadas al caso pesimista corresponden en su mayoría a *queries* muy pequeñas (que se ven borrosas), *queries* con deformaciones muy pronunciadas, otras donde el logo no se ve completamente. También se observó que aquellas *queries* muy grandes tampoco fueron correctamente detectadas. En contraste, las *queries* asociadas al caso optimista tienden a mostrar una representación clara y frontal de los logos asociados, sin mayores deformaciones. Tampoco se encuentran en este caso *queries* excesivamente pequeñas en comparación.

El análisis cualitativo y visual es relevante pues mediante lo observado en la Tabla 4.9, es difícil determinar directamente si una *query* será buena o no para la detección basándose sólo en su tamaño. Sin embargo, se pueden hacer ciertas observaciones, como por ejemplo que la mediana de tamaño en el caso pesimista es mucho menor, por lo que este escenario concentra una buena cantidad de *queries* pequeñas en comparación.

Observar el tamaño de las *queries* puede resultar útil, tomando en cuenta lo mostrado en la Figura 4.13, se puede ver cómo los peores casos acumulan *queries* excesivamente pequeñas, independiente de su forma. También sucede para el caso de las *queries* excesivamente grandes. En cambio, las *queries* con mejores resultados se tienen mayoritariamente en un tamaño intermedio.

Figura 4.11: Queries asociadas al caso pesimista.

Figura 4.12: *Queries* asociadas al caso optimista.

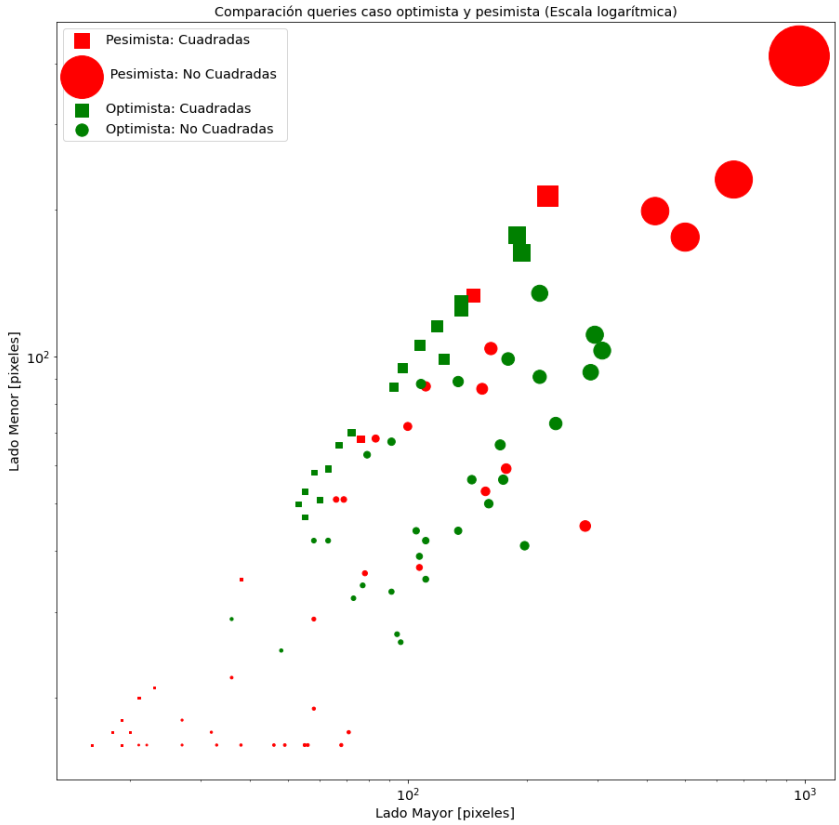


Figura 4.13: Comparación de *queries* para caso optimista y pesimista. Tamaño de punto basado en el tamaño ($Alto \times Largo$) de cada *query*.

Capítulo 5

Conclusiones y Trabajo futuro

El trabajo realizado pudo abordar la problemática de detección de patrones bidimensionales en imágenes. Se logró desarrollar un framework capaz de realizar detecciones a través de características generadas por modelos pre-entrenados, dejando una metodología definida. Esta formulación permitirá seguir expandiendo la experimentación con modelos adicionales a los utilizados en este documento, y por otra parte facilitar mucho los ensayos con otros sets de datos.

La investigación de comparación de características multiescala a través de la correlación tuvo resultados realmente positivos, demostrando su viabilidad en el ámbito de la detección. Además, una de las grandes ventajas de esta investigación es que no necesita una gran cantidad de muestras para ser empleada, y, por otra parte, reutiliza directamente modelos ya entrenados en otras tareas, lo cual implica un gran ahorro en recursos de todo tipo.

Si se estudian los objetivos planteados en el inicio de este escrito, se puede observar que:

- Se comprobó que la correlación permite encontrar similitud entre partes de una imagen y una *query* que corresponde a un patrón bidimensional.
- Se realizaron pruebas en distintas redes y capas para encontrar las de mejor desempeño, descubriendo que la calidad de las características a comparar si dependen del modelo de extracción, y abriendo la interrogante del por qué se da este fenómeno.
- Se definió una arquitectura basada en correlación para realizar la detección de patrones bidimensionales.
- Se realizaron las pruebas correspondientes sobre el set de datos *DocExplore*, obteniendo resultados razonablemente buenos, consiguiendo estadísticas y mediciones descriptivas de los mismos.
- Se evaluó el desempeño en detección y recuperación sobre el dataset de Logos *FlickrLogos47*, obteniendo también mediciones que describen su desempeño.

Se probó además que utilizar características de múltiples escalas dependiendo del tamaño de la query buscada puede producir una mejora en tiempo de búsqueda, sin afectar gravemente la precisión de las detecciones.

Respecto al estado del arte correspondiente a DocExplore, se lograron superar notoriamente todos los resultados obtenidos en investigaciones anteriores, a excepción de un estudio en desarrollo y aun no publicado[31] en el que se está cooperando. Esto se puede comprobar en la Tabla resumen 4.5.

Por lo demás, se comprobó que no es confiable aplicar este método directamente sobre ambientes complejos (flickrlogos 47) para detección, pero si se aprendió que es útil para la tarea recuperación de imágenes hasta cierto punto, y que con modificaciones a la forma en la que se obtienen los *bounding boxes* de cada detección la precisión en detección podría aumentar considerablemente.

La búsqueda de *logos in the wild* corresponde a un caso de estudio de la utilización de un método que funciona de forma excelente en documentos. Se puede tender a pensar en que la complejidad de la tarea es mucha para la metodología planteada, sin embargo, se ha mostrado que bajo condiciones ideales, la detección y recuperación son viables (considerando mejores casos de detección y recuperación, en donde las queries se definen como buenas). Esto abre una oportunidad para seguir investigando, realizando pruebas y alteraciones al método. A modo de ejemplo, se habló de utilizar otro tipo de selección de características además de PCA, como UMAP, que tiene la capacidad de conservar contexto espacial, cuestión que PCA es incapaz de realizar, pero por plazos no pudo realizarse en este documento.

Como no se encontró un estudio de comparación para el *dataset* FlickrLogos 47 en detección a través de *queries*, se decidió observar los resultados para las clases del *dataset* FlickrLogos-32, pues existe un estudio en detección que corresponde a *Few-Shot Learning* y que realiza detección en este *dataset*. Sin embargo, este estudio[18] involucra entrenamiento específico tanto en logos generales como en logos pertenecientes al mismo *dataset*, mientras que los modelos propuestos en este documento utilizan modelos pre-entrenados en imágenes no relacionadas (*ImageNet* [9]). Por este motivo, los resultados no son directamente comparables.

Es razonable entonces encontrar que los modelos planteados en 2.7.4 tengan un mucho mejor desempeño para logos que los modelos planteados en este documento, pues fueron entrenados para esto. Al punto de que debió generarse, con el esfuerzo de múltiples personas, un set de datos etiquetados suficiente para realizar sus entrenamientos. En cambio este documento corresponde la experimentación con el uso de modelos no entrenados para la tarea aplicados a la resolución del problema de detección de logos.

La metodología que se plantea en este documento para búsqueda de patrones bidimenssionales *in the wild* deja espacio para el crecimiento, pues dada la implementación, puede ser fácilmente expandida. Vale la pena estudiar cuál sería el comportamiento y desempeño utilizando modelos como los mostrados en 2.7.4, que están entrenados en reconocer logos en general, como extractores de características y luego aplicar la búsqueda por similaridad con correlación como se hace en la metodología descrita en el presente informe.

Un punto muy positivo de esta implementación es que no se limita a la detección y recuperación de patrones bidimensionales, pues la manera en la que se formuló permite la experimentación con cualquier tipo de *query*, en cualquier tipo de dataset que esté anotado en formato COCO. Esto permitiría en teoría solucionar problemas que pueden ser reducidos a

algo similar la búsqueda de patrones bidimensionales, por ejemplo la ya mencionada búsqueda de productos en planogramas, dado que si bien los productos no son habitualmente planos, la vista de ellos es generalmente frontal.

En el futuro se experimentará con el *dataset* Tobacco 800 [32], que también es abordado por Ubeda [26], pues este representa un desafío "intermedio" entre DocExplore y Flickrlogos, ya que corresponde a logos en documentos, teniendo una mayor variabilidad que *DocExplore* pero una menor complejidad de ambiente que *Flickrlogos*.

Además de esto, se está trabajando junto al profesor guía y Cristóbal Loyola en una estructura de detección *one-shot*, incorporando correlación, extracción de características de modelos pre-entrenados y otros conceptos mencionados en este documento. Esto se realiza con el propósito de encontrar un método más general que los patrones bidimensionales, y buscando mejorar también la obtención de *bounding boxes* de forma más inteligente, utilizando redes como CenterNet[33].

Bibliografía

[1] Romberg S., Pueyo L., Lienhart R., Van Zwol R. *Scalable Logo Recognition in Real-World Images*. Proceedings of the 1st ACM International Conference on Multimedia Retrieval, ICMR'11, Trento, Abril 2011.

[2] Karlinsky L., Shtok J., Harary S et al. *RepMet: Representative-based Metric Learning for Classification and One-shot Object Detection*. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2018.

[3] Girshick R., Donahue J., Darrell T. et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2014.

[4] Girshick, R. *Fast R-CNN*. In Proceedings of the IEEE International Conference on Computer Visions (Vol. 2015 International Conference on Computer Vision, ICCV 2015, pp. 1440–1448). Institute of Electrical and Electronics Engineers Inc. `https://doi.org/10.1109/ICCV.2015.169`. 2015

[5] Ren S., He K., Girshick R. et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137–1149. `https://doi.org/10.1109/TPAMI.2016.2577031`. 2017.

[6] Viola P. & Jones, M. *Rapid Object Detection using a Boosted Cascade of Simple Features*. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR (Vol. 1). IEEE. 2001.

[7] Dalal N., Triggs B. *Histograms of Oriented Gradients for Human Detection*. International Conference on Computer Vision & Pattern Recognition (CVPR '05), San Diego, United States. pp.886–893. Junio 2005.

[8] Felzenszwalb P., Girshick R., McAllester D. *Cascade object detection with deformable part models*. Computer Society Conference on Computer Vision and Pattern Recognition. IEEE. 2010.

[9] Krizhevsky A., Sutskever I. & Hinton G. *ImageNet classification with deep convolutional neural networks*. Communications of the ACM, 60(6), 84-90. 2017.

[10] Redmon J., Divvala S., Girshick R., & Farhadi, A. *You Only Look Once: Unified, Real-Time Object Detection*. Conference on Computer Vision and Pattern Recognition (CVPR). IEEE. 2016

[11] Lin T., Goyal P., Girshick R. *Focal loss for dense object detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, PP(99), p.1. 2018.

[12] Tüzko A., Herrmann C., Manger D. et al. *Open Set Logo Detection and Retrieval*. Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. 2017.

[13] Deng J., Dong W., Socher R. et al. *Imagenet: A large-scale hierarchical image database*. In Conference on Computer Vision and Pattern Recognition, pages 248–255. IEEE. 2009.

[14] Mas Montserrat D., Lin Q., Allebach J., Delp E. *Scalable Logo Detection and Recognition with Minimal Labeling*. In Proceedings - IEEE 1st Conference on Multimedia Information Processing and Retrieval, MIPR 2018 (pp. 152–157). Institute of Electrical and Electronics Engineers Inc. `https://doi.org/10.1109/MIPR.2018.00034`. 2018

[15] Lauren N. *Cross-correlation of 2 matrices*. `https://observablehq.com/@lemonnish/cross-correlation-of-2-matrices`. Septiembre 2019.

[16] Huang G., Liu Z., Van Der Maaten L. et al. *Densely connected convolutional networks*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708, Honolulu, Junio 2017.

[17] Hang S., Xiatian Zhu., Shaogang G. *Deep Learning Logo Detection with Data Expansion by Synthesising Context*. IEEE Winter Conference on Applications of Computer Science (WACV), Santa Rosa, USA. Marzo 2017.

[18] Fehérvári I., & Appalaraju S. *Scalable logo recognition using proxies*. In Proceedings - 2019 IEEE Winter Conference on Applications of Computer Vision, WACV 2019 (pp. 715–725). Institute of Electrical and Electronics Engineers Inc. `https://doi.org/10.1109/WACV.2019.00081`. 2019.

[19] Movshovitz-Attias Y., Toshev A., Leung T. et al. *No fuss distance metric learning using proxies*. In ICCV, pages 360–368. IEEE Computer Society, 2017.

[20] Van Der Maaten L., Hinton G. *Visualizing data using t-SNE*. Journal of machine learning research, 9(Nov):2579–2605, 2008.

[21] Wiggers K., Britto A., Heutte L., Koerich A., Oliveira L. *Image Retrieval and Pattern Spotting using Siamese Neural Network*. Proceedings of the International Joint Conference on Neural Networks. 2019.

[22] Wiggers K., Britto A., Koerich A., Heutte L., Oliveira L. *Deep learning approaches for image retrieval and pattern spotting in ancient documents*. 2019.

[23] Simonyan K., Zisserman A. *Very deep convolutional networks for large-scale image recognition.* 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 2015.

[24] Nash W., Drummond T., Birbilis N. *A review of deep learning in the study of materials degradation.* npj Materials Degradation. 2018.

[25] En. Sovann, Nicolas Stéphane, Petitjean Caroline, Jurie Frédéric, Heutte, Laurent. *New public dataset for spotting patterns in medieval document images.* 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. 2015.

[26] Ubeda, I., Saavedra, J.M., Nicolas, S., Petitjean, C., Heutte, L. *Pattern spotting in historical documents using convolutional models.* arXivpreprint arXiv:1906.08580. 2019.

[27] Xie S., Girshick R., Dollár P. et al. *Aggregated residual transformations for deep neural networks.* Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, vol. 2017 - January (2017) pp. 5987-5995 Published by Institute of Electrical and Electronics Engineers Inc. Enero 2017.

[28] He K., Zhang X., Ren S. et al. *Deep residual learning for image recognition.* Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2016-December pp. 770-778 Published by IEEE Computer Society. Diciembre 2016.

[29] Lin T., Maire M., Belongie S. et al. *Microsoft COCO: Common objects in context*, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition pp. 3686-3693. 2015.

[30] Chen L., Zhu Y., Papandreou G. et al. *Rethinking Atrous Convolution for Semantic Image Segmentation*, Google Inc. arXiv preprint arXiv:1706.05587. 2018.

[31] Zacarias C., Stéphane N., Pierrick T., Alceu de Souza B., José M.S., Laurent H. *Combining fully-convolutional activation features and cross-correlation for image retrieval and pattern spotting.* Preprint submitted to Pattern Recognition. Noviembre 2020

[32] Zhu G., Doermann D. *Automatic document logo detection.* Proceedings of the International Conference on Document Analysis and Recognition, ICDAR. 2007.

[33] Duan KBai SXie L et al. *CenterNet: Object Detection with Keypoint Triplets.* Proceedings of the IEEE International Conference on Computer Vision. 2019.