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

RECONOCIMIENTO DE OBJETOS PARTICULARES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

MATÍAS IGNACIO CORNEJO ALARCÓN

PROFESOR GUÍA:
JAVIER RUIZ DEL SOLAR SAN MARTÍN

PROFESOR CO-GUÍA:
PATRICIO LONCOMILLA ZAMBRANA

COMISIÓN:
FELIPE TOBAR HENRÍQUEZ

SANTIAGO DE CHILE
2024

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: **MATÍAS IGNACIO CORNEJO ALARCÓN**
FECHA: 2024
PROF. GUÍA: JAVIER RUIZ DEL SOLAR
PROF. CO-GUÍA: PATRICIO LONCOMILLA

RECONOCIMIENTO DE OBJETOS PARTICULARES

El ámbito de la detección de objetos ha experimentado un notable avance en términos de precisión y velocidad gracias al progreso en la capacidad computacional. Sin embargo, un aspecto menos explorado es la detección de objetos particulares, conocida como detección de instancias de objetos. Este enfoque implica desarrollar metodologías de entrenamiento para modelos computacionales con bases de datos limitadas. Para ilustrar este punto, mientras que hay numerosas bases de datos que permiten detectar categorías generales, como perros o animales, el desafío aumenta al intentar detectar instancias específicas, como un perro particular.

Por esta razón, en este trabajo se propone una nueva arquitectura, basada en un modelo que representa la detección de objetos como mapas de calor, CenterNet, capaz de explorar este problema. Esta propuesta de modelo se entrena con la base de datos COCO, igual que CenterNet. Además, entrega como resultado un mapa de calor único, que contiene la detección de los objetos, logrando incluso, detecciones de objetos que no se presentan en la base de datos en la que se entrenó. Una vez entrenado, se extraen las características del objeto detectado para luego ser comparado con una pequeña base de datos de objetos particulares. En una evaluación preliminar, la precisión lograda en términos de clasificación para esta base de datos reducida de 26 objetos fue del 80.77 %.

En conclusión, en cuanto a la detección de objetos, se logra un F-score del 55.92 % para la configuración de 6 objetos por imagen con un umbral de IoU de 0.75, y un F-score del 82.28 % con un umbral de IoU de 0.50. Es relevante señalar que, aunque el modelo muestra buen rendimiento en detección y clasificación por separado, se identifican desafíos al intentar integrar eficazmente ambos aspectos. Este hallazgo destaca la complejidad de lograr una armoniosa combinación de detección y clasificación de objetos en un único modelo.

A todos los que piensan que soy mejor,
de lo que yo mismo pienso.

Gracias

Agradecimientos

Es paradójico pensar que estos agradecimientos nunca habrían existido sin la valiosa ayuda y respaldo de las personas que me rodean, por lo tanto;

Agradezco profundamente a mis padres, Jeannette Alarcón y Freddy Cornejo, por todo el apoyo brindado a lo largo de estos, aún cortos, 27 años. Gracias por proporcionarme las herramientas necesarias para mi desarrollo académico y, lo que es más importante, por apoyarme en mi crecimiento personal. Agradezco su comprensión y aceptación de mis decisiones, como el cambio de colegio y la decisión de abandonar la idea de ser médico en el último momento. También, agradezco su respaldo durante mi experiencia de intercambio, incluso cuando eso implicó sacrificios a nivel económico. Por todo esto y más, les estoy sinceramente agradecido.

No quiero dejar fuera a nadie, así que agradezco a todos mis familiares, hermanas, tíos, tías, primos y primas que siempre han preguntado *cómo iba la u* y han intentado apoyarme en todo momento.

A mis amigos, que veo o hablo a diario y me soportan en el día a día. Julian y Eduardo, con los cuales hablo dia y noche, acompañandonos siempre en esta aventura llamada estudio.

Mi gratitud también se extiende a todos aquellos que conocí durante mi prolongada etapa universitaria y con quienes compartí momentos inolvidables.

A aquellos que siempre me han tenido en un pedestal, aún sin entender completamente por qué, les expreso mi agradecimiento sincero.

Agradezco a quienes mostraron interés, de una forma u otra, en mis charlas apasionadas sobre inteligencia computacional y temas relacionados.

Finalmente, pero no menos importante, agradezco a todos los profesores con los que he interactuado a lo largo de mi vida, cada uno de los cuales ha aportado una pieza valiosa a este rompecabezas que conforma mi persona.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación

El desarrollo de los modelos de inteligencia computacional avanza cada vez más rápido. Con el avance paralelo de la computación, el área de Machine Learning se vuelve cada vez más eficiente y otorga más herramientas para el desarrollo dentro de esta área [1]. Dentro de este campo existen muchos tópicos, tanto para texto, audio, imágenes, videos, etc.

Existen dos temas bastante estudiados, pero que están lejos de ser perfeccionados; el de reconocimiento de objetos y reconocimiento de objetos particulares, donde el primero lleva años mejorando y el segundo aun no está tan explotado. Con respecto al reconocimiento de objetos particulares, importante para el desarrollo de esta memoria, Oriol Vinyals et al. 2016 fue uno de los primeros en explorar el área de *aprender con pocos ejemplos* [2].

Tal como se mencionó, el reconocimiento de objetos lleva años mejorando, en parte, debido a que las bases de datos utilizadas cada vez son más grandes dado el gran flujo de información que mueve la población mundial actualmente, que pone en mesa conceptos como 'Aprendizaje profundo' y 'Big Data'. Sin embargo, ¿qué ocurre cuando no hay tantos datos disponibles?, en este caso, los grandes algoritmos empiezan a fallar, en especial cuando se habla de ambientes domésticos, donde se quiere buscar la 'instancia de un objeto' como lo es *mi taza de café, mi billetera o mi llave de casa*. Hay limitados ejemplos (imágenes) disponibles para estas instancias, lo que presenta un desafío para estas arquitecturas.

1.2. Objetivos

1.2.1. Objetivos Generales

El objetivo fundamental de esta memoria es evaluar una propuesta de modelo que, basándose en una red neuronal convolucional, busca abordar el desafío de las detecciones de instancias particulares. Para ello, se tomará como modelo base a CenterNet, que representa la detección de objetos mediante mapas de calor, donde los puntos de máxima intensidad indican la posición de un objeto detectado. Este modelo será modificado siguiendo la metodología de YoloSPoC, otro enfoque que aborda esta problemática. Es importante destacar que se empleará un clasificador de distancia coseno para mejorar la generación del mapa de

características.

1.2.2. Objetivos Específicos

Para lograr el desafío anterior se requiere cumplir diferentes objetivos específicos, los cuales se listan a continuación:

- Modificar CenterNet para transformarlo en un generador de descriptores adecuado a la tarea de detección de instancias específicas.
- Lograr que este modelo modificado genere buenas propuestas de detecciones de objetos.
- Dadas las propuestas de objetos, lograr que el modelo modificado pueda clasificar imágenes.
- Evaluar los resultados del modelo modificado.

Capítulo 2

Antecedentes Generales

2.1. CNN-RNN

2.1.1. CNN: Redes neuronales convolucionales

En 1998, *Yann Lecun y otros* [3] dan a conocer al mundo el entrenamiento de una *Red Neuronal Multicapa* a la cual nombran *LeNet-5*, que se convertiría en el estado del arte de las *Redes Neuronales Convolucionales*. Este modelo utiliza una serie de *capas* convolucionales de dos dimensiones en imágenes, junto a técnicas de reducción de dimensiones (*Sub-Sampling*), que ayudan a predecir, al final de esta cadena, la imagen que se le está entregando.

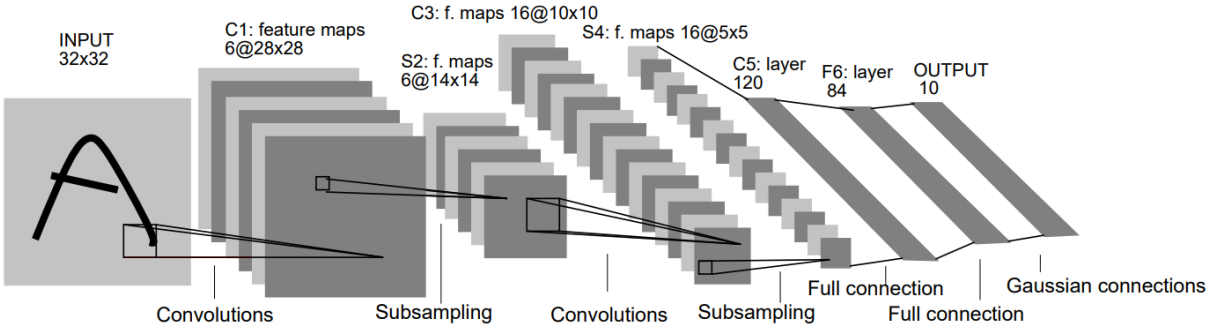


Figura 2.1: Arquitectura CNN LeNet-5 para reconocimiento de dígitos. Figura obtenida de [3].

En el modelo de la figura 2.1, se suministra una imagen de entrada de 32x32 píxeles, donde la primera capa de convolución (C1: mapas de características) produce 6 imágenes o tensores de dimensiones 28x28 píxeles, como se indica en el texto "6@28x28". Desde un punto de vista matemático, una convolución se define como la integral del producto de una función x con una función h, desplazando una de estas en una distancia t:

$$(x * h)(t) = \int_{-\infty}^{\infty} x(x)h(t-x)\,dx$$

En vista de que se trabaja con un dominio de tiempo discreto, en este caso se resume a:

$$x[n] * h[n] = \sum_{-\infty}^{\infty} x[k]h[n-k]$$

donde $x[n]$ es una señal de entrada y $h[n]$ un *filtro o kernel*.

Dicho brevemente, para una convolución en 2 dimensiones se extiende el pensamiento anterior como una convolución conjunta de la dirección horizontal y vertical, dentro de un espacio bidimensional, quedando:

$$x[m,n] * h[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j]h[m-i,n-j]$$

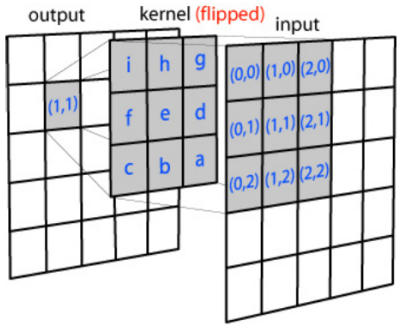


Figura 2.2: Convolución 2D.

En el contexto del procesamiento de imágenes, x representa a una imagen de entrada, $x(i,j)$ el píxel en la posición [i,j] y h una matriz (generalmente de tamaño impar) a la cual se llamará *filtro* o *kernel*. En definitiva, y de manera gráfica, una convolución en 2D es la suma de la multiplicación punto a punto de un píxel con el kernel invertido en ambas dimensiones.

En la figura 2.2 se presenta un kernel de tamaño 3x3, que se invierte (desplaza) para recorrer la imagen. Sea I(x,y) el píxel de la imagen de entrada en la posición (x,y), el resultado del primer paso realizado por el kernel sería;

$$O(1,1) = i * I(0,0) + h * I(1,0) + g * I(2,0) + ... + b * I(1,2) + a * I(2,2)$$

y así para cada paso que dé el kernel en la imagen de entrada.

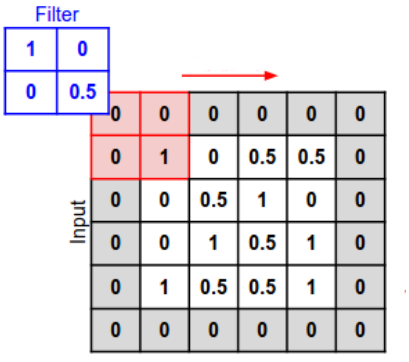


Figura 2.3: Padding.

En la imagen de salida, el primer resultado no se guarda en la posición $O(0,0)$ debido a que, al realizar una convolución, el kernel no puede moverse la misma cantidad de píxeles que hay en la imagen original. Esto provoca una reducción en el tamaño de salida, generando una pérdida de información en los bordes de la imagen. Para solucionar eso se utiliza una técnica llamada *Padding*, lo que permite generar una imagen de salida del mismo tamaño que la imagen de entrada. La forma más común de realizar un padding es agregando un borde en la imagen original con el valor de 0, tal como se muestra en la figura 2.3.

2.1.2. RNN: Redes neuronales residuales

Con el avance del tiempo y junto a las mejoras en hardware de las GPU [4], los resultados al utilizar Redes Neuronales Convolucionales en temas de procesamiento de imágenes iba mejorando notablemente, incluso para *Datasets* más grandes [5].

Junto al avance tecnológico en hardware, se podría asumir que mientras más grande la red, es decir, mientras más capas convolucionales existan en ésta, mejores resultados se pueden obtener, lo cual resultó ser contraproducente, ya que una red compleja implica un mayor costo computacional y consume más tiempo, pero en una aplicación del mundo real como escenarios industriales y comerciales, los ingenieros y desarrolladores a menudo se enfrentan al requisito de un presupuesto de tiempo limitado [6][7]. Por tanto, en 2016 se explora una nueva arquitectura [8], la cual utilizaría *Redes Residuales*, derivado de *'shortcut connections'*.

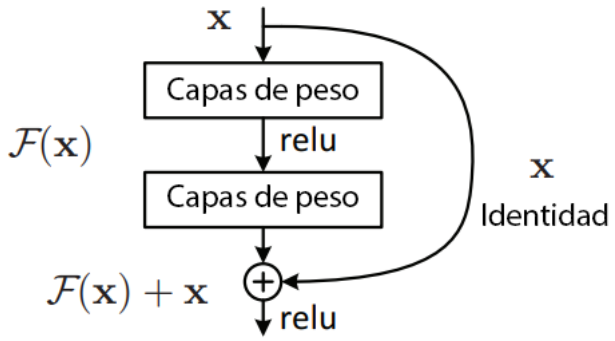


Figura 2.4: Bloque de construcción para aprendizaje residual. Figura obtenida de [8].

Tal como se muestra en la figura 2.4, un bloque residual procesa la entrada x con distintas capas/funciones (F), resultando en $F(x)$. En una CNN básica, $F(x)$ sería la salida habitual, pero en este caso a la salida se le suma la identidad de la entrada x, obteniendo una salida $F(x) + x$.

Esta arquitectura utiliza estos atajos con el fin de saltar ciertas capas que terminan siendo poco útiles para el problema. La incorporación de estos atajos es esencial, ya que en arquitecturas de redes muy profundas puede surgir un problema conocido como 'Vanishing Gradient' o 'Desvanecimiento de gradiente'. Este fenómeno se manifiesta cuando los gradientes se vuelven tan pequeños que dificultan que la red se entrene de manera efectiva. La presencia de atajos permite mitigar este problema al proporcionar rutas más directas para la retropropagación del gradiente, facilitando así el entrenamiento de la red.

Si se considera nuevamente la figura 2.4 y se extrapola, cuando en mitad del modelo de una arquitectura grande existe una capa llamada x, que representa el resultado ideal que el modelo preferiría utilizar como salida, este resultado óptimo pasaría nuevamente por las capas (F). En este caso, el modelo tendría que aprender nuevamente a generar x. Por lo que, al agregar la identidad x a $F(x)$, se permite al modelo simplemente ignorar $F(x)$, lo que resulta en una reducción en el tiempo de entrenamiento, ya que el modelo no tiene que aprender nuevamente a generar la misma salida deseada.

2.2. Detección de objetos

Figura 2.5: Ejemplo Detección de objetos. Figura obtenida de [9].

Junto al avance de las CNN, un área dentro del procesamiento de imágenes fue avanzando; la detección de objetos. En primer lugar, es necesario entender la diferencia entre la detección de objetos y la clasificación de imágenes. La detección, importante para el desarrollo de este memoria, se basa en detectar un objetivo de interés, dibujando algo conocido como *bounding box*, que se representa como una caja alrededor del objeto encontrado, tal como se ilustra en la figura 2.5, donde una persona montando a caballo es encerrada en un rectángulo rojo, este último representando el bounding box o *bbx*. Por otra parte, la clasificación de imágenes, tal como su nombre lo dice, clasifica el objeto dentro de una imagen con una *etiqueta o label* que representa el titulo o *clase* del objeto, en el caso de la figura sería 'persona'.

2.2.1. RoI y Extracción de características

Para entender las siguientes arquitecturas, que servirán de base para la arquitectura final de esta memoria, es necesario tener claro el concepto de *Región de interés o RoI*, del inglés *Region of Interest* y Extracción de características, del inglés *Feature Extraction*.

Figura 2.6: Ejemplo de regiones de interés.

En la detección de objetos, una región de interés es un área de la imagen que posiblemente contiene un objeto a identificar. De manera gráfica y utilizando la figura 2.6, una región de interés puede representarse mediante un rectángulo. Inicialmente, un objeto puede tener más de una región de interés, y el objetivo de un modelo de aprendizaje automático es reducir el número de regiones para generar finalmente una región más amplia que representa al objeto.

En el ámbito del aprendizaje automático, la extracción de características se refiere al proceso de selección y filtrado de características en una base de datos. Este proceso tiene un triple propósito, según Guyon y Elisseeff (2003)[10]: mejorar el rendimiento predictivo de los clasificadores, proporcionar clasificadores más eficientes y rentables, y ofrecer una comprensión mejorada de los datos. De igual modo, llámese *mapa de características* a la salida de una capa dentro de una CNN que tiene la información o características relevante de una imagen, que facilitan la detección o clasificación de un objeto.

2.2.2. R-CNN y mejoras

Una red neuronal se puede pensar como una caja negra a la que se le entrega una entrada y devuelve una salida con ciertos datos que se interpretan dependiendo del problema a resolver. En el caso de la detección de objetos, utilizando CNN, la entrada es una imagen y la salida posee una cantidad finita, pero variable, de objetos detectados. El problema es que la salida de una red neuronal debe tener un número fijo de resultados, lo que no es compatible con el dinamismo de la detección de objeto. Por lo tanto, no se puede resolver el problema de detección de objetos con un modelo CNN tradicional, como lo sería la arquitectura mostrada en la figura 2.1.

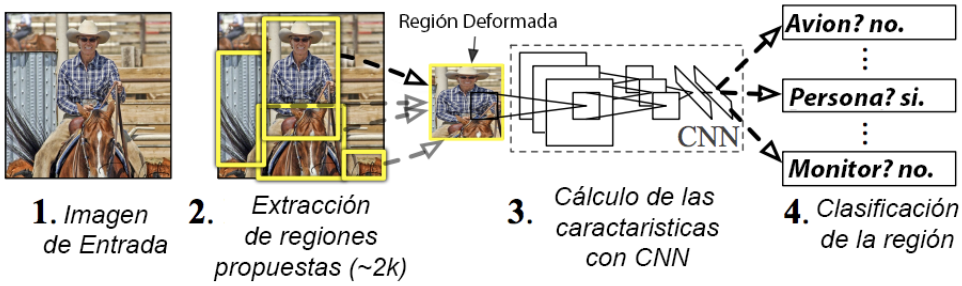


Figura 2.7: Red RCNN. Figura obtenida de [9].

Para resolver este problema, y tal como se observa en la figura 2.7, Ross Girshick et al.[9] proponen extraer una cantidad grande de regiones de interés, que dependiendo del escenario puede ser sobre 2000, cada región con distintos tamaños y relaciones de aspecto. Luego, en el tercer paso de la figura, se calculan las características de cada región utilizando una CNN, que a su vez se utilizan para clasificar la región en el cuarto paso, mediante un modelo de machine learning SVM (*Support Vector Machine*).

Para generar las aproximadas 2000 regiones de interés, en el paso dos, se utilizó un algoritmo de búsqueda selectiva [9], que combina de forma recursiva regiones en regiones más grandes, midiendo la similitud de color, textura, tamaño y forma entre regiones.

Esta solución presenta al menos dos grandes problemas. El primero es que, tal como se mencionó, cada imagen genera 2000 regiones, lo que provoca un gran tiempo de entrenamiento, ya que si por cada región se calcula unas características mediante la red CNN, por cada imagen son 2000 veces el tiempo de lo que tarda una extracción. El segundo problema es que el algoritmo de búsqueda es un algoritmo fijo, es decir, la detección de regiones no va mejorando con cada ejemplo, no aprende.

2.2.2.1. Fast R-CNN

En el contexto de la mejora de la velocidad de entrenamiento, el mismo autor de la solución anterior propone ajustes adicionales en una versión posterior, conocida como Fast R-CNN [11]. En esta variante, la generación de características mediante la Convolutional Neural Network (CNN), que corresponde al paso 3 en la figura 2.7, se realiza directamente sobre la imagen de entrada en lugar de hacerlo respecto a las 2000 propuestas de regiones,

dando como resultado un mapa de características de la imagen de entrada.

Paralelamente, a la imagen de entrada se le aplica una extracción de regiones propuestas, tal como en el paso 2 de la figura 2.7. Una vez teniendo las coordenadas de todas las regiones propuestas, se extraen estas posiciones en el mapa de características generado anteriormente, volviendo al resultado de R-CNN original.

La gran diferencia, a nivel de costo computacional, es que ahora se calcula solo una vez el mapa de características por imagen, a diferencia de R-CNN original, que calculaba 2000 por imagen, reduciendo drásticamente el tiempo de entrenamiento.

2.2.2.2. Faster R-CNN

A pesar de que Fast R-CNN es 9 veces más rápido que R-CNN al momento de entrenar[11], solucionando en parte uno de los grandes problemas de la detección de objetos, aun quedaba por mejorar la detección de los *bounding boxes*, ya que tal como se dijo anteriormente, las búsquedas selectivas son algoritmos fijos que son lentos y pueden tener muchos problemas al momento de generar regiones de interés, ya que calculan un número elevado de propuestas que no se adapta a distintas situaciones, en el caso de R-CNN, aproximadamente 2000 regiones.

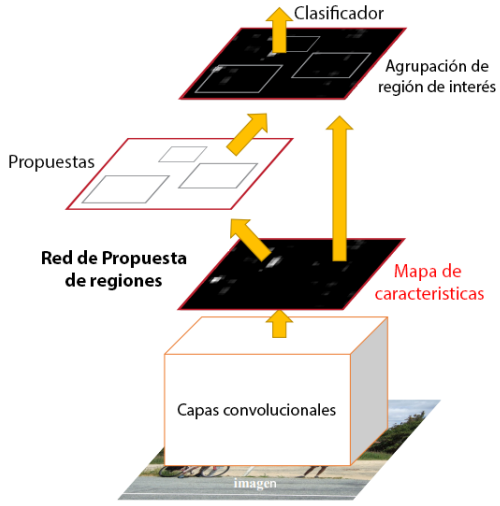


Figura 2.8: Arquitectura Faster R-CNN. Figura obtenida de [12].

Observando la figura 2.8, similar a Fast R-CNN, Faster R-CNN[12] calcula un mapa de características único a la imagen de entrada utilizando una CNN, pero en este caso se extraen las propuestas de regiones utilizando otra CNN, la cual nombraron (*Region Proposal Network*). Esta nueva red aprende a generar regiones, remplazando la búsqueda selectiva. Junto a cada región, se obtiene un puntaje de confianza en su detección, al que llaman *score*. Finalmente, y de manera análoga a sus versiones anteriores, se realiza una clasificación sobre las características extraídas en estas regiones.

2.2.3. YOLO

Los algoritmos previamente presentados comparten una estructura similar en su funcionamiento, ya que siempre emplean regiones de la imagen para detectar un objeto, limitándose a 'mirar' solo una parte de la imagen en lugar de considerar el contexto completo. En este contexto, Joseph Redmon et al. presentan un enfoque innovador en la detección de objetos llamado *You Only Look Once* (YOLO) [13].

YOLO, que significa *'Solo Miras Una Vez'* en inglés, busca examinar una imagen de manera más global, teniendo en cuenta el contexto completo en lugar de dividir la tarea en regiones, como se hacía en modelos anteriores.

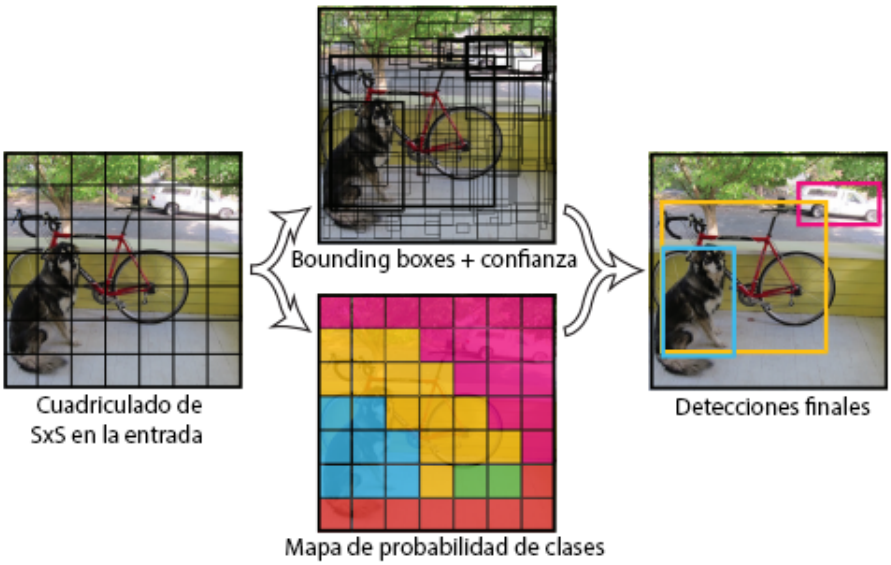


Figura 2.9: Arquitectura YOLO. Figura obtenida de [12].

Observando la figura 2.9, el algoritmo primero divide la imagen en SxS cuadrículas (Grid), a cada uno de los cuadrados generados se le calcula un número B de *Bounding Boxes*, por defecto YOLO utiliza dos bounding boxes por cuadrado. Junto a los bounding boxes, YOLO calcula un puntaje de confianza para cada uno de estos; esto se observa en la parte superior de la figura 2.9, donde el grosor de la linea en el bounding box representa la confianza del bounding box generado. Paralelamente, YOLO calcula un mapa de probabilidad para las clases, esto para determinar la clase del objeto detectado, en la parte inferior de la figura 2.9, el color cian, amarillo y rosado corresponde a perro, bicicleta y auto respectivamente.

Es importante destacar que para cada cuadro delimitador (bounding box), se realizan cinco predicciones distintas, que incluyen las coordenadas del centro (x, y), el ancho, la altura y un puntaje de confianza asociado al cuadro delimitador. Como resultado, el modelo combina estas predicciones para generar un tensor de tamaño $S \times S \times (B \times 5 + C)$, donde $S \times S$ representa el número de celdas generadas por la cuadrícula, B es el número de cuadros delimitadores por celda, 5 es el tamaño de cada cuadro delimitador, y C es el número de clases en la base de datos utilizada.

2.3. CornerNet y CenterNet

2.3.1. CornerNet

Avanzando en el razonamiento, el estado de arte de la detección de objetos se resume en la predicción de *Bounding Boxes*. En los modelos mencionados anteriormente, el modelo siempre indicaba las coordenadas del bounding box de manera numérica; tomando de ejemplo YOLO, el resultado final era un tensor de tamaño $S \times S \times (B \times 5 + C)$, donde $S \times S$, entregando un bounding box con los valores (x, y, w, h).

Razones por las cuales, en 2018, Hei Law y Jia Deng intentan otra manera para detectar un objeto, a la cual llaman *'Detectar objetos como puntos clave emparejados'*[14] o *CornerNet*, que detecta un objeto como un par de esquinas del bounding box agrupadas, pero de una manera gráfica. CornerNet crea, simultáneamente, dos *imágenes binarias* a las cual llamaremos mapas de calor (del inglés *heatmap*); el primero posee la información de la esquina superior izquierda, mientras que el segundo la esquina inferior derecha.

El mapa de calor se representa como una matriz binaria $\hat{Y} \in [0,1]^{WxHxC}$, con W, H y C; el ancho, alto y número de puntos claves de un objeto, donde $\hat{Y}_{x,y,c} = 1$ cuando existe una esquina y $\hat{Y}_{x,y,c} = 0$ cuando es el fondo. Un punto clave, esquina en este caso, está representado en la capa *mapas de calor* de la figura 2.10 con el color rojo, que representa un máximo en el mapa de calor.

CornerNet calcula características en cada punto máximo del mapa de calor, características que, por simplicidad, llamaremos por su nombre en inglés, *embeddings*. Estos *embeddings* ayudan a detectar a qué objeto pertenece cada esquina, representados por los colores verde y naranja en la capa *embeddings* de la figura 2.10, los cuales a su vez indican el grupo al que pertenecen. En una etapa de postprocesamiento, estas esquinas se agrupan para formar los bounding boxes finales, representados por los rectángulos segmentados en la figura 2.10.

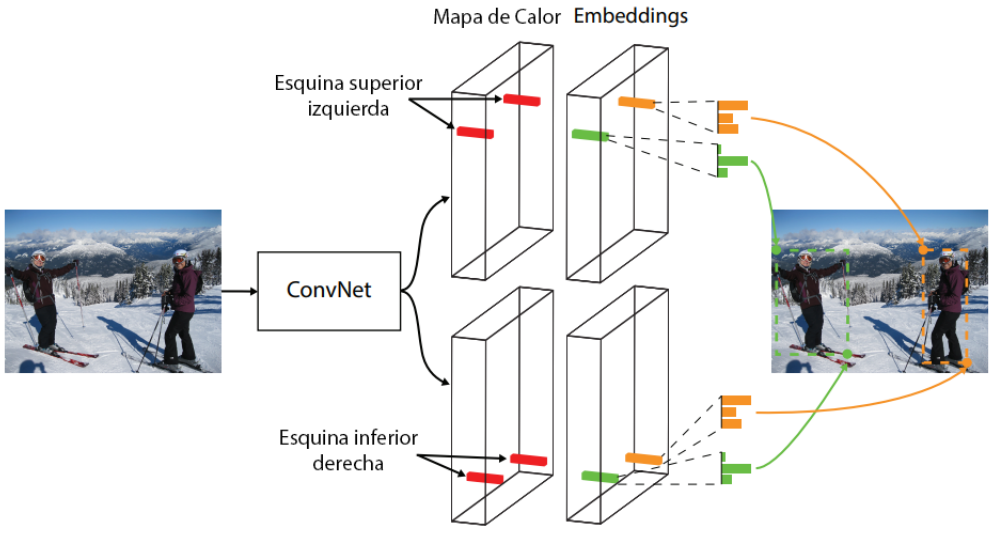


Figura 2.10: Arquitectura CornerNet. Figura obtenida de [14].

2.3.2. CenterNet

Siguiendo la idea de CornerNet, CenterNet[15] simplifica la complejidad del modelo a un solo punto de interés en el mapa de calor, aprovechando el centro de masa de un objeto. Junto a este punto central, extraído del mapa de calor, se calcula el ancho y alto del bounding box.

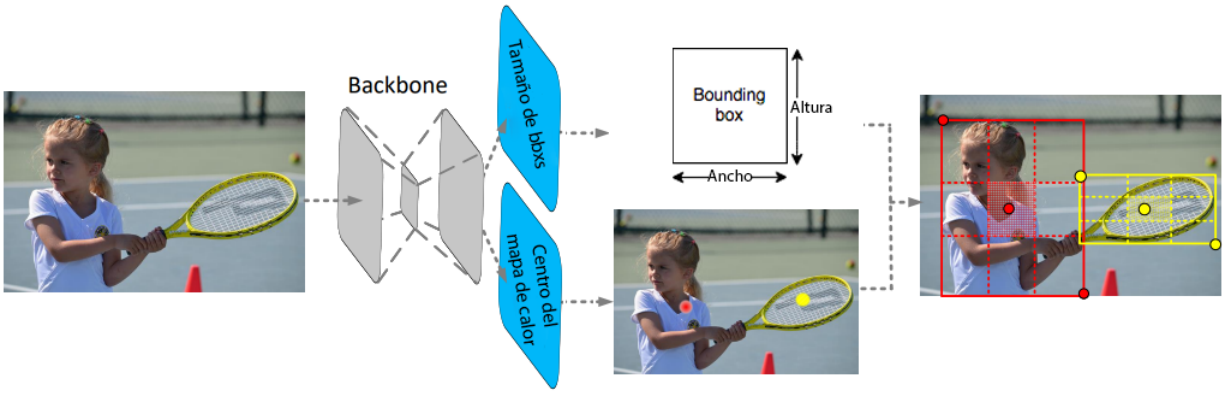


Figura 2.11: Arquitectura Centernet. Figura obtenida de [15].

Sirva de ejemplo la figura 2.11, los puntos rojos y amarillos en la imagen ya procesada representan al centro de masa de cada objeto, en este caso, las clases rojo y amarillo representan; *persona y raqueta* respectivamente. En la parte superior, y de manera paralela, se calculas las dimensiones del bounding box final. Finalmente se unen estos resultados para generar los rectángulos rojos y amarillos de cada objeto. En la figura 2.11 se observa un modelo nombrado como *Backbone*. Este es el nombre que se le entrega a un modelo, generalmente una red neuronal convolucional, que funciona como un extractor de características, de igual manera que los modelos de detección de objetos presentados anteriormente.

2.4. YoloSPoC

Con cada algoritmo nuevo, los resultados de las detecciones de objetos va mejorando a un punto que se comparan con la detección que puede hacer un humano, pero para tener estos resultados necesitan una gran cantidad de imágenes de entrenamiento para cada clase. Como ejemplo se tiene a COCO, abreviado de (*Common Objects in Context*) [16], que es una base de datos con 118.000 imágenes de entrenamiento, que también posee un número de 80 clases o categoría de objetos, donde una imagen puede tener más de un objeto, ergo, puede haber más de una clase por imagen, lo que implica un número de 1.5M de objetos, que se conocen como *instancia o vista.*

Este gran número de instancias no es posible cuando se encuentran en un ambiente doméstico en el que solo tienen un número limitado de *vistas* por objeto, ya que no se desea predecir la categoría del objeto, sino el objeto per se. De manera puntual, si se quisiera detectar a una persona llamada *pedro*, un modelo de detección como CenterNet, entregaría la clase *persona* como resultado, mientras que un modelo de detección particular quiere devolver como resultado *pedro*, pero a diferencia de CenterNet que trabaja con miles de clases con la etiqueta *persona*, un modelo de detección particular cuenta con un número limitado de imágenes de *pedro*, que deben ser distintas, hablando desde un punto de vista de correlación.

Por esta razón, los profesores de nuestra facultad, Patricio Loncomilla y Javier Ruiz-del-Solar, implementan YoloSPoC [17] que intenta resolver la tarea de reconocimiento de instancia de objeto particular utilizando métodos basados en CNN y clasificación de vecinos cercanos.

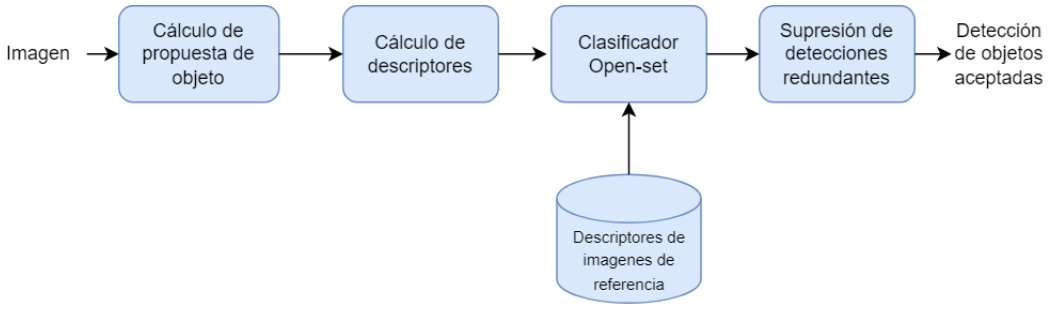


Figura 2.12: Arquitectura YoloSPoC. Figura obtenida de [17].

Tal como se observa en la figura 2.12, la metodología de este trabajo se puede dividir principalmente en cuatro bloques; (i) se calculan propuestas de objetos utilizando YOLOv3, (ii) se calculan descriptores o características de las propuestas, (iii) se reconoce la instancia de objeto usando un clasificador de vecinos cercanos *open-set*, que puede rechazar la detecciones generada por objetos fuera de la base de datos de entrenamiento, y (iv) se eliminan las detecciones redundantes.

2.5. *Backbone* DLA: Deep Layer Aggregation

Tal como se mencionó anteriormente, *Backbone* es el nombre que se le entrega a un modelo, generalmente una red neuronal convolucional, que funciona como un extractor de características. Por lo tanto, se puede interpretar una arquitectura *Backbone* como una caja negra dentro de un modelo complejo como CenterNet. Es necesario explicar uno de estos modelos para el completo entendimiento, no solo de CenterNet, sino de cualquier modelo que utilice algún modelo *Backbone*. En concreto, en el desarrollo de este proyecto, se utiliza CenterNet con un *Backbone* DLA.

Dicho lo anterior, *Deep Layer Aggregation*, abreviado *DLA*, es una red neuronal convolucional que explora la mejor manera de agregar capas en una red. Experimentalmente, esta técnica muestra mejoras en el uso de la memoria y el rendimiento con respecto a modelos base como ResNet en tareas de clasificación [18].

En el artículo *deep layer aggregation* [18] se define *agregación* como la combinación de diferentes capas a lo largo de una red. Se llama profundo, a un grupo de agregaciones, si es composicional, no lineal y la primera capa agregada pasa a través de múltiples agregaciones. Como las redes pueden contener muchas capas y conexiones, el diseño modular ayuda a contrarrestar la complejidad mediante la agrupación y la repetición. Las capas se agrupan en bloques, que luego se agrupan en etapas por su resolución característica. DLA es un término que *cubre* dos estructuras diferentes: *Iterative Deep Aggregation (IDA)* y *Hierarchical Deep*

Aggregation (HDA).

2.5.1. IDA: Iterative Deep Aggregation

Para ilustrar mejor, observando la figura 2.13, la agregación comienza en la escala más pequeña y superficial y luego se fusiona iterativamente en escalas más profundas y más grandes. De esta manera, las características poco profundas se refinan a medida que se propagan a través de diferentes etapas de agregación.

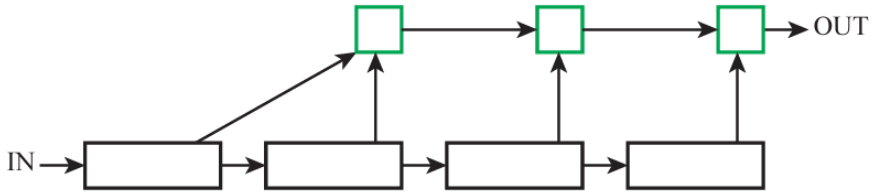


Figura 2.13: Arquitectura IDA. Figura obtenida de [15].

2.5.2. HDA: Hierarchical Deep Aggregation

HDA fusiona bloques y etapas en un árbol para preservar y combinar canales de características. Con HDA, las capas más superficiales y más profundas se combinan para aprender combinaciones más ricas que abarcan una mayor parte de la jerarquía de características. La siguiente figura sirve para explicar de manera gráfica HDA:

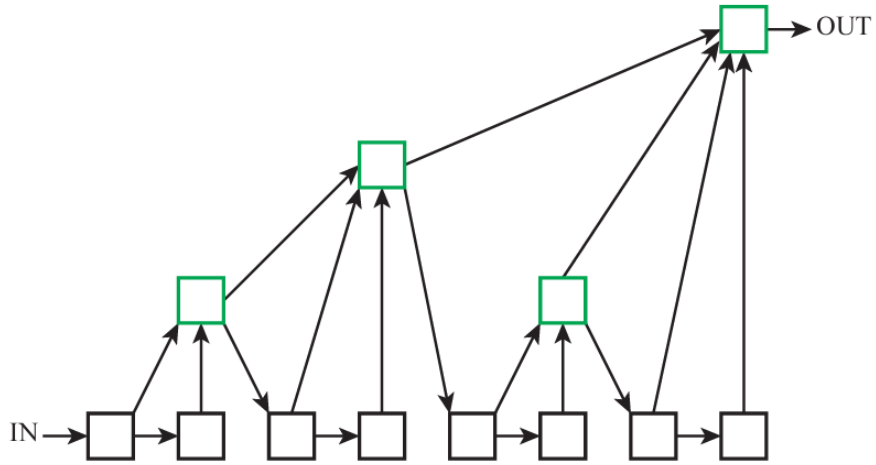


Figura 2.14: Arquitectura HDA. Figura obtenida de [15].

Notar cómo la salida de un nodo de agregación alimenta la entrada del siguiente bloque; esto conserva las características de las capas anteriores.

2.5.2.1. Combinación de IDA y HDA

Consideremos ahora una red neuronal simple, la combinación de IDA y HDA sobre este, se ve ilustrado en la figura 2.15;

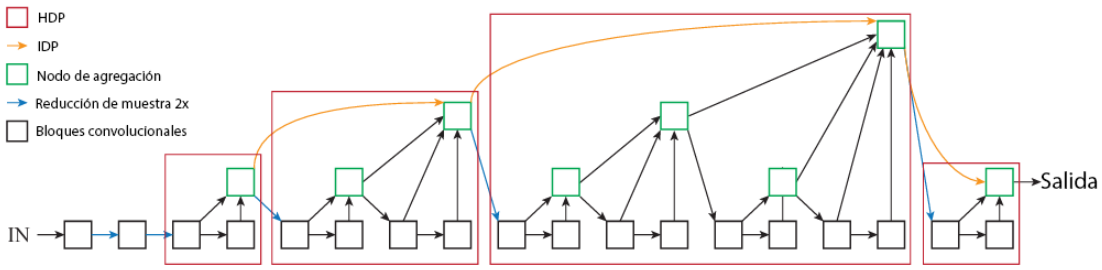


Figura 2.15: Arquitectura DLA. Figura obtenida de [15].

Donde la agregación de capas profundas aprende a extraer mejor el espectro completo de información semántica y espacial de una red. Las conexiones iterativas unen etapas vecinas para profundizar progresivamente y refinar espacialmente la representación. Las conexiones jerárquicas cruzan etapas con árboles que abarcan el espectro de capas para propagar mejor las características y los gradientes de la red.

Capítulo 3

Metodología

En este capítulo se presenta la metodología empleada para abordar el problema de detección de instancias de objetos. En términos generales, el objetivo principal de este proyecto es lograr la detección y clasificación de instancias específicas de objetos, lo que implica que el modelo debe ser capaz de identificar un objeto entre otros de la misma clase. Un ejemplo de esto sería encontrar ***mi taza*** entre 5 tazas similares. En este caso, el desafío va más allá de la simple detección de clases. Para abordar este problema, se adopta la idea de YoloS-PoC, que se basa en la comparación de descriptores para evaluar la similitud entre instancias particulares de objetos. Este enfoque se explorará más detalladamente al explicar el nuevo algoritmo desarrollado.

Finalmente, la separación de las tareas de detección y clasificación, junto con la generación de un mapa de calor único, facilita la identificación de puntos de interés y la generación de bounding boxes alrededor de los objetos detectados. Se utiliza un clasificador basado en la distancia coseno para comparar descriptores y evaluar la similitud entre instancias detectadas y aquellas en la base de datos de entrenamiento. El rendimiento se analiza mediante métricas como precisión, recall, F-score e IoU.

3.1. Entendimiento de CenterNet

Esta propuesta de modelo se basa fuertemente en CenterNet. A pesar que se quiere lograr un objetivo distinto al que logra CenterNet, la manera en que este genera sus embeddings es digna de utilizar y adaptar en un nuevo problema. Para esto se utilizará CenterNet para generar descriptores (características) de los objetos para poder realizar una identificación en estos.

Dada la explicación anterior de CenterNet, se sabe que este genera paralelamente, 1 mapa de calor para cada clase y las dimensiones del bounding box para cada objeto detectado. Sin embargo, este algoritmo es aun más complejo.

3.1.1. Metodología de CenterNet

3.1.1.1. Mapa de Calor

Sea $I \in \mathbb{R}^{WxHx3}$ una imagen de ancho W y altura H, con 3 canales (RGB). Para calcular el centro de un objeto, CenterNet produce un mapa de calor de puntos de interés $\hat{Y} \in [0,1]^{\frac{W}{R}x\frac{H}{R}xC}$, donde R es el *Stride* de salida, que es el paso que toma un filtro dentro

del cálculo de la convolución para recorrer la imagen al momento de calcular esta misma. CenterNet usa $R = 4$ como valor por defecto en sus pruebas.

Como se afirmó arriba, una predicción con $\hat{Y}_{x,y,c} = 1$ corresponde a un punto de interés detectado, mientras que $\hat{Y}_{x,y,c} = 0$ es el fondo. En CenterNet se utilizan, y comparan, diversas redes *Backbones* para el cálculo de la predicción, pero en este caso particular, solo se mencionará a *DLA*.

Para entrenar el modelo, con respecto al cálculo del centro de un objeto, se compara la predicción hecha por CenterNet y lo que se conoce como *Ground Truth*, que representa la salida ideal del problema. En este caso, el *Ground Truth* es un mapa de calor generado de forma manual que posee los valores correctos para los puntos claves en las imágenes de entrenamiento.

Para cada punto clave de *ground truth* $p \in \mathbb{R}^2$ de la clase $c \in C$, se calcula un equivalente de baja resolución $\tilde{p} = \lfloor \frac{p}{R} \rfloor$. Finalmente, se generan los mapas de calor para los puntos claves utilizando una función gaussiana, tal que $Y_{xyc} = exp(-\frac{(x-\tilde{p}_x)^2+(y-\tilde{p}_y)^2}{2\sigma_p^2})$, donde σ es la desviación estándar adaptativa al tamaño del objeto detectado [15].

La forma en que un modelo de Machine Learning mejora, no dista de la optimización de una función matemática; de hecho, la idea básica del entrenamiento de un modelo de inteligencia computacional consta en la optimización de múltiples funciones lineales y no lineales, a las cuales se le llama neuronas[3]. Y la forma de indicar que tan *correcta* es la predicción de un modelo, o dicho de otra manera, la función objetivo que se quiere optimizar, se le llama *función de pérdida* del inglés *loss function*. Por lo tanto, mientras más bajo sea el *loss*, mejor funciona el modelo.

Existen distintos tipos de funciones de pérdida; uno muy común es el llamado *Cross Entropy Loss*, que calcula la diferencia entre dos distribuciones de probabilidad, tal que:

$$CEL = -\sum_i P(i) log \hat{P}(i)$$

En el contexto de la inteligencia computacional, P representa la distribución de probabilidad de nuestro *Ground Truth* y $\hat{P}$ la distribución de probabilidad de la predicción generada por el modelo.

En CenterNet, para evaluar la eficacia del modelo y su posterior optimización, se contrasta el mapa de calor del *Ground Truth* con el mapa de calor generado por el modelo. Sin embargo, al emplear una función de pérdida como la Entropía Cruzada (CEL), se otorga un peso significativo a los píxeles que corresponden al fondo de la imagen, donde no hay puntos de interés ($Y_{x,y,c} = 0$). Dado que la cantidad de píxeles que representan el fondo es considerablemente mayor que aquellos que representan un objeto, esto genera una distribución de probabilidades desequilibrada hacia el fondo.

Por esta razón en CenterNet, utilizando la misma idea de CornerNet, se usa una función de pérdida llamada *FocalLoss*, que penaliza la influencia de los ejemplos fáciles, como sería la

detección del fondo, ya que casi todos los puntos dentro de una imagen pertenecen a este, y beneficia a los ejemplos difíciles, como la detección de objetos, implementado de la siguiente forma:

$$FL = \frac{-1}{N} \sum_{xyc} \begin{cases} (1 - \hat{Y}_{xyc})^{\alpha} log(\hat{Y}_{xyc}) & si\ Y_{xyc} = 1 \\ \\ (1 - Y_{xyc})^{\beta} (\hat{Y}_{xyc})^{\alpha} log(1 - \hat{Y}_{xyc}) & de\ lo\ contrario \end{cases}$$

Donde α y β son hyper-parametros del *FocalLoss*, N el número de puntos de interés en una imagen I. En este proyecto, se utilizó $\alpha = 2$ y $\beta = 4$, ya que fueron los valores utilizados en CenterNet para todos sus experimentos.

3.1.1.2. Tamaño de bounding boxes

Sea $(x_1^{(k)}, y_1^{(k)}, x_2^{(k)}, y_2^{(k)})$ el bounding box de un objeto k de la clase c_k, con $(x_1^{(k)}, y_1^{(k)})$ y $(x_2^{(k)}, y_2^{(k)})$ pares de coordenadas que representan el punto superior izquierdo y punto inferior derecho del bounding box respectivamente. Donde su centro se aloja en $p_k = (\frac{x_1^{(k)} + x_2^{(k)}}{2}, \frac{y_1^{(k)} + y_2^{(k)}}{2})$. CenterNet, para cada punto de interés k, intenta predecir el tamaño del objeto $s_k = (x_2^{(k)} - x_1^{(k)}, y_2^{(k)} - y_1^{(k)})$. Para indicar que tan bien el modelo predice este par s_k, se utiliza una función de pérdida que mide el error absoluto medio entre la predicción $(\hat{S})$ y el *Ground Truth* s_k de la siguiente forma:

$$L = \frac{1}{N} \sum_{k=1}^{N} |\hat{S}_{pk} - s_k|$$

3.1.1.3. Error de discretización Offset

Durante el cálculo del mapa de calor, se menciona que el mapa de calor generado posee dimensiones $(\frac{W}{R} x \frac{H}{R} x C)$, con *Stride* $R = 4$. Esta disminución en el tamaño de la imagen genera un error de discretización. Así, por ejemplo, sea $(\hat{x}, \hat{y}) = (11, 44)$ la posición central de un objeto. Dada la reducción del *Stride* $R = 4$, el modelo puede predecir $(x, y) = (2, 11)$ o $(x, y) = (3, 11)$, que re-escalados quedan; $(x, y) = (8, 44)$ o $(x, y) = (12, 44)$ respectivamente. Esta diferencia debido al cambio de tamaño de la imagen se recupera con una predicción paralela generada por CenterNet para cada punto central, llamada *offset local* $\hat{O} \in \mathbb{R}^{\frac{W}{R} x \frac{H}{R} x 2}$.

Para su optimización, se utiliza una función de pérdida *L1 Loss*, similar al tamaño de bounding boxes, de la siguiente forma:

$$L_{off} = \frac{1}{N} \sum_{p} |\hat{O}_{\tilde{p}} - (\frac{p}{R} - \tilde{p})|$$

Con $\hat{O}$ la predicción realizada por el modelo. Recordando que p es un punto clave de *ground truth* de la clase c, y $\tilde{p}$ su equivalente de baja resolución.

3.1.1.4. Estimación Final

Finalmente, CenterNet usa una red única que predice de manera simultánea, los puntos claves $\hat{Y}$, tamaño de bounding box $\hat{S}$ y offset $\hat{O}$.

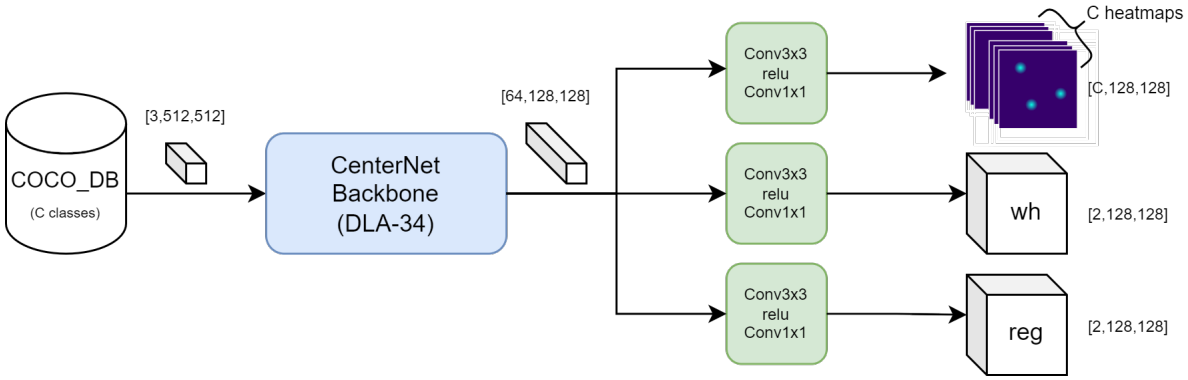


Figura 3.1: Arquitectura CenterNet más detallada.

Con respecto al entrenamiento, y observando la figura 3.1, CenterNet utiliza la base de datos de COCO, con C = 80 Clases, donde todas sus imágenes están escaladas a 512x512 píxeles, estas imágenes son ingresadas al backbone de CenterNet (DLA) que devuelve un tensor con dimensiones (64x128x128), representado por los pequeños paralelepípedos en la figura 3.1; 64 el número de *máscaras* generadas por el *backbone* y 128x128 la nueva dimensión de la imagen. Esta salida es común para cada *rama* observada en la figura, que se pasa a través de una convolución, una función ReLU y otra convolución para generar la predicción de Mapas de calor, tamaño de bounding boxes (wh en la figura) y offset (reg en la figura) respectivamente. La función ReLU es una función rectificadora que elimina los valores negativos de la siguiente forma:

$$ReLU(x) = \begin{cases} x & si\ x \geq 0 \\ \\ 0 & de\ lo\ contrario \end{cases}$$

3.1.2. Nuevo algoritmo

Se debe recordar que el objetivo de esta nueva arquitectura es poder generar descriptores para un objeto particular, tal como se hace en YoloSPoC, con el fin de poder discernir y reconocer un objeto comparando su descriptor con los objetos de referencia.

Un modelo como CenterNet, que está entrenado en la base de datos COCO, funciona bien para las 80 clases de COCO, pero no es capaz de detectar objetos fuera de estas 80 categorías. Por lo tanto, el primer objetivo es separar, de alguna forma, la detección del objeto de su clasificación.

La hipótesis es que si al modelo se le entrena de manera separada la forma en que se detecta y clasifica un objeto, el modelo va a *aprender* a como detectar un objeto independientemente de su clase, detectando así objetos fuera de las categorías de la base de datos, COCO en este caso. Sin embargo, aunque el modelo sea capaz de detectar un objeto fuera de la base de datos, no clasificará este objeto de manera correcta, ya que CenterNet está limitado para las 80 categorías de COCO, pero la clasificación del objeto no es realmente importante para el nuevo algoritmo, que tiene de objetivo generar buenos descriptores.

Notar que CenterNet hace este proceso de detectar y clasificar de manera simultánea,

ya que cada clase tiene su propio mapa de calor con sus puntos de interés, que representan objetos, tal como se observa en la figura 3.1, donde hay C mapas de calor, cada uno de dimensiones 128x128 píxeles.

3.1.2.1. Separación de detección y clasificación

Para lograr este nuevo algoritmo, se crea una *rama* nueva, separando la detección y la clasificación de la siguiente manera:

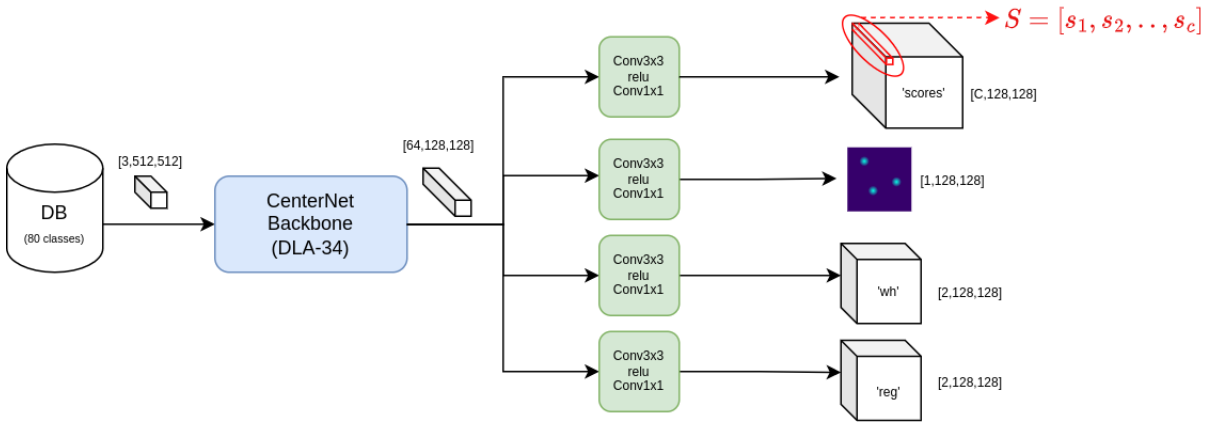


Figura 3.2: Separación de *rama* mapas de calor.

En la figura 3.2, se observan dos *ramas* nuevas, una que representa a los *scores* o puntuación, y otra que representa a un mapa de calor único, al cual llamaremos *Objectness*. El nuevo tensor de salida *scores*, entrega una distribución de probabilidades para cada píxel en la dimensión 128x128. Esta distribución se observa en la figura como un Vector S de tamaño C, representando las C clases de la base de datos. El máximo valor $s_c \in S$ representaría la clase con mayor puntaje, indicando la categoría del objeto en la posición $(x, y) \in \mathbb{R}^{128x128}$.

De manera paralela, el modelo genera solo un mapa de calor, *Objectness*, para todos los objetos detectados, sin importar su clase. Uniendo estas 4 ramas, se vuelve al resultado generado por CenterNet. En teoría, esta nueva arquitectura tiene sentido, pero no necesariamente va a tener resultados competentes o similares a los obtenidos por CenterNet.

En particular, la *rama Scores* es la más importante para este proyecto, ya que los *embeddings* generados por esta rama para clasificar un objeto son los que se utilizarán como descriptores. Por esta razón, se debe forzar a este modelo a generar embeddings para un cierto píxel $p \in \mathbb{R}^{128x128}$.

Para lograr lo anterior, antes de generar el tensor *scores* final, se coloca una clasificador en medio de la siguiente forma:

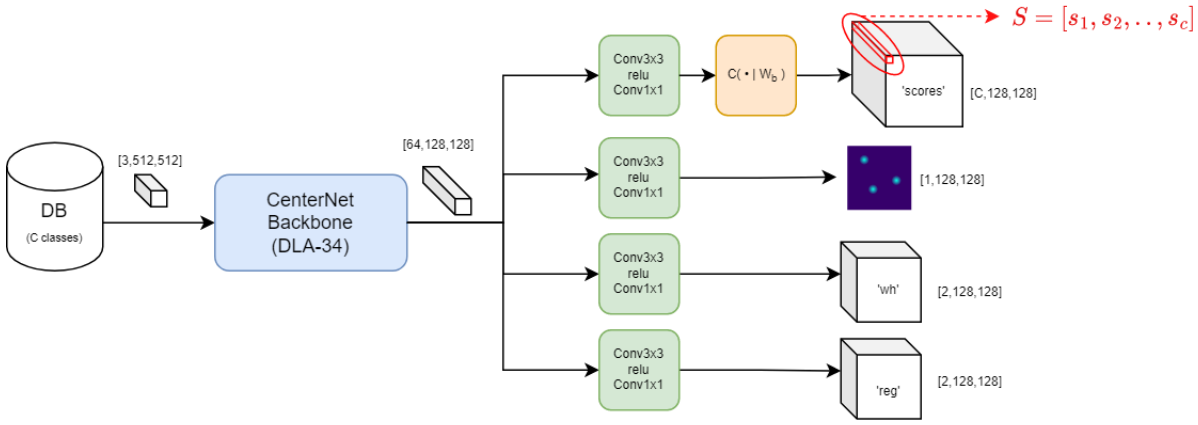


Figura 3.3: Implementación de clasificador.

El clasificador implementado se refleja, de manera modular, en al figura 3.3 como un cuadrado de color naranjo. La idea de este clasificador es poder separar los embeddings y lograr hacerlos independientes píxel a píxel, es decir, que cada píxel de la imagen tenga una predicción de una clase particular.

3.1.2.2. Clasificador Distancia Coseno

Se ha mencionado reiteradamente que la idea del proyecto es generar descriptores o embeddings que destaquen las características de un objeto para luego poder comparar los descriptores de distintos objetos y ver la distancia de estos. De esta forma, si existiera una base de datos con descriptores de imágenes de referencia representando objetos particulares como ***mi taza*** o ***mi billetera***, se podría comparar una imagen de entrada con este clasificador para ver a cual de todas las imágenes de referencia tiene más *cercanía*, igual que YoloSPoC en 2.12.

En este proyecto se utiliza un clasificador distancia coseno, que mide la similitud entre dos vectores mediante una función trigonométrica.

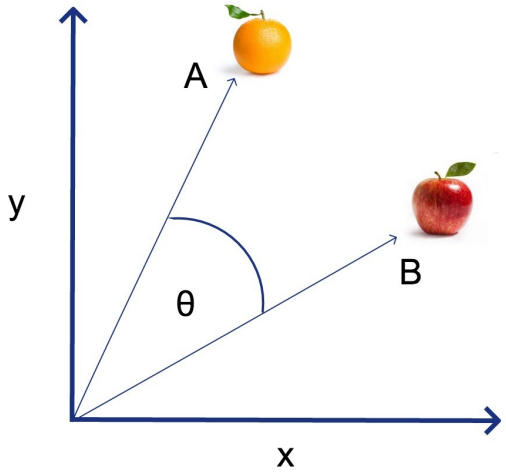


Figura 3.4: Ejemplo Distancia coseno.

Sean dos vectores, A y B, que contienen ciertas características de una naranja y una manzana, acompañándose de 3.4, la forma de medir su similitud sería tal que:

$$C(A,B) = cos(\theta) = \frac{A * B}{||A||||B||} \in [0,1]$$

. Donde $C(A,B) = 1$ indicaría que los objetos A y B son iguales.

En el contexto de este proyecto; Sea d el tamaño de un embedding, $F_S \in \mathbb{R}^{dx128x128}$ el tensor de salida del cuadrado verde de la *rama scores* en la figura 3.3 y $W \in \mathbb{R}^{dxC}$ un tensor entrenable que contiene las características necesarias para las C clases de la base de datos a entrenar (COCO en este caso), un clasificador coseno de la forma:

$$C(\cdot|W) = \frac{W^T F_s}{||W||||F_S||} \in [0,1]^{Cx128x128}$$

calcula la distancia coseno de dos embeddings, de tamaño d, en W y F_S para un píxel $p \in \mathbb{R}^{128x128}$ y una clase $c \in C$.

Para lograr el tensor *scores* de la figura 3.3, que contiene la distribución de probabilidad de una clase C, se aplica una función *softmax* o función exponencial normalizada, que comprime un vector K-dimensional, $\mathbf{z}$, de valores reales arbitrarios en un vector K-dimensional $\sigma(\mathbf{z}) \in [0,1]^K$, dada por:

$$\sigma(\mathbf{z})_j = \frac{e^{\mathbf{z}_j}}{\sum_{k=1}^{K} e^{\mathbf{z}_k}} \; para \; j = 1, ..., K$$

Esta función *softmax* se aplica sobre la dimensión de las clases del clasificador coseno, obteniendo así el vector S de la figura 3.3.

3.1.2.3. Recuperación de estimación final

Una vez establecida la forma de calcular las nuevas *ramas* de este nuevo algoritmo, se debe establecer como proceder a entrenar el modelo. Dado que solo la *rama* de los mapas de calor de CenterNet fue modificada, es la única en la que se debe cambiar la manera en calcular el *loss*. Esto tiene al menos dos soluciones posibles; La primera, mantener el cálculo del *FocalLoss* sobre el mapa de calor único, pero agregando un *loss* sobre el tensor *scores*, utilizando *CrossEntropyLoss*. La segunda, es combinar la salida de *scores* con la salida del mapa de calor único, para retornar la misma salida que CenterNet original, es decir, un mapa de calor para cada una de las clases. Ambas opciones fueron implementadas, pero solo la última tuvo resultados destacables (ver capítulo 4).

Por lo tanto, la manera elegida para recuperar la estimación, manteniendo la salida del modelo y entrenamiento de CenterNet original, es la siguiente:

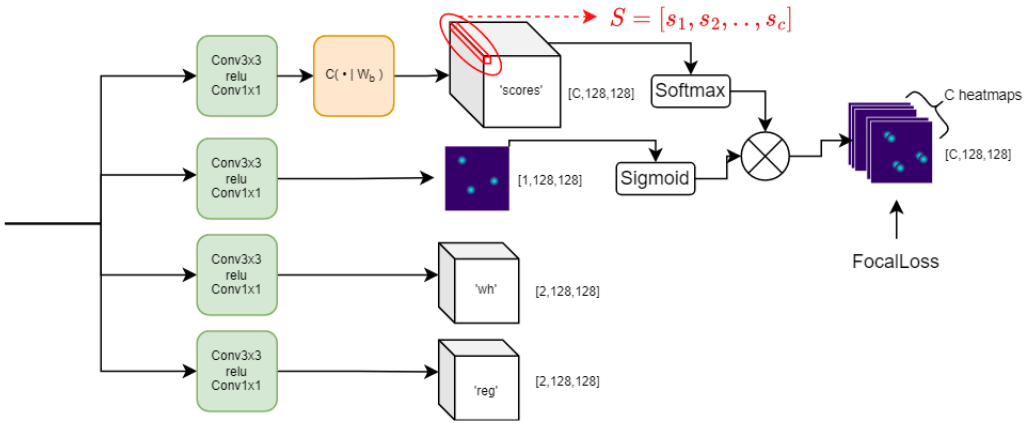


Figura 3.5: Implementación final del nuevo modelo.

Si se observa la figura 3.5, al mapa de calor único objetivo se le aplica una función sigmoide o *sigmoid*, de esta manera se asegura que el mapa de calor está entre los rangos $[0, 1]$. La función sigmoide está definida por $P(t) = \frac{1}{1+e^{-t}} \in [0, 1]$. Finalmente, se realiza una multiplicación píxel a píxel, volviendo a un tensor con un mapa de calor para cada clase.

Condensando lo dicho hasta aquí, la hipótesis de este nuevo modelo es que al forzar mediante un clasificador coseno el modelo aprenda a distinguir las 80 clases de COCO, utilizando la metodología de CenterNet, la *rama* de *scores* pueda generar embeddings que tengan características relevantes de la clase de un objeto, de esta manera al comparar embeddings entre objetos se podría medir la distancia entre estos.

Finalmente, utilizando la figura 3.6 para explicar la nueva red, el cálculo de descriptores viene dado por esta nueva arquitectura basada en CenterNet para luego ser comparados con descriptores de imágenes de referencia. Estas imágenes de referencia también son obtenidas utilizando la nueva arquitectura.

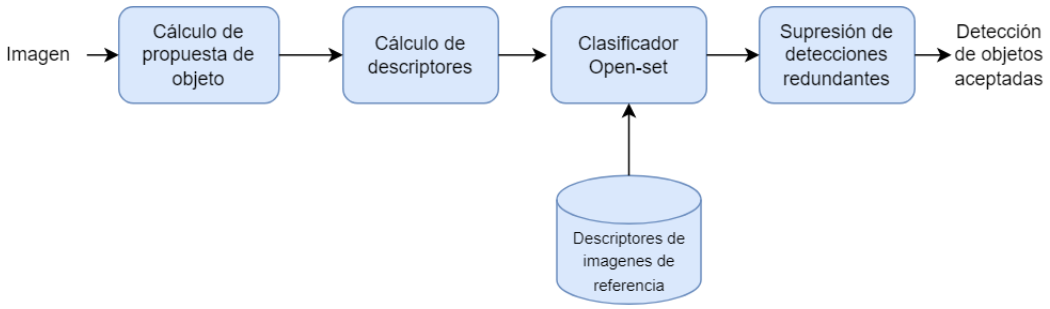


Figura 3.6: Arquitectura YoloSPoC adaptada a CenterNet. Figura obtenida de [17].

3.1.2.4. Detección de instancias de objetos

Hasta el momento, se ha abordado el proceso de recuperación de la estimación final para entrenar el modelo siguiendo la metodología de CenterNet con COCO. No obstante, el objetivo central de esta memoria se centra en la detección de instancias específicas de objetos, en lugar de realizar la detección y clasificación basándose en la base de datos de COCO.

Por lo tanto, para detectar instancias de objetos mediante el nuevo modelo, se introduce una imagen de prueba que contiene objetos no presentes en la base de datos de COCO, tal como se observa en la figura 3.7. El modelo utiliza el mapa de calor generado en la segunda rama de la arquitectura para calcular la posición de los objetos propuestos. De manera paralela, se extraen los embeddings de las posiciones con mayor calor respectivamente, siendo el modulo $C(\cdot|W_b)$ el responsable de medir la distancia entre los embeddings generados por el modulo verde y los embeddings de referencia W_b. Este enfoque sigue un mecanismo similar al utilizado por YoloSPoC, lo que posibilita la identificación y clasificación de instancias específicas de objetos en lugar de clases genéricas presentes en COCO.

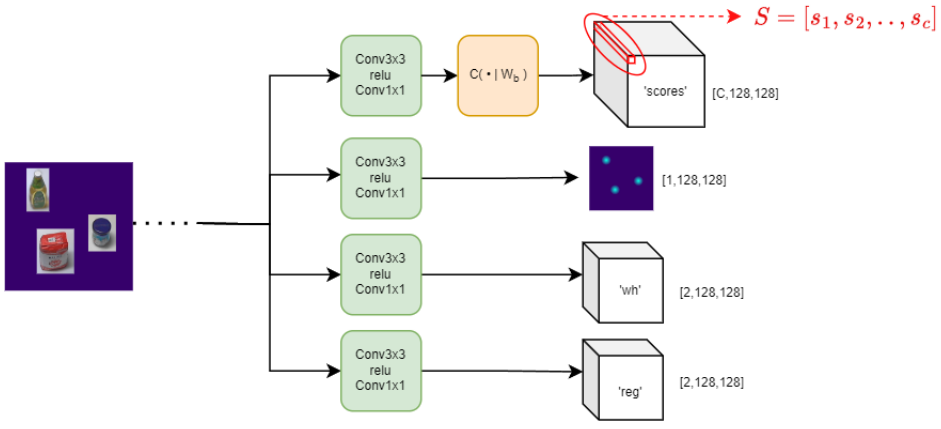


Figura 3.7: Esquema de modelo nuevo basado en CenterNet.

3.1.3. Métrica AP: Average Precision

Para ver el alcance de este modelo con respecto a CenterNet en la detección y clasificación sobre COCO se utiliza la métrica de detección de objeto AP.

En primer lugar es importante tener claro los términos de Verdadero Positivo (TP), Falso Positivo (FP) y Falso Negativo (FN):

1. Verdadero Positivo (TP): Detección correcta por parte del modelo.
2. Falso Positivo (FP): Detección incorrecta por parte del modelo.
3. Falso Negativo (FN): Un *Ground Truth* no detectado por parte del modelo.
4. Verdadero Negativo (VN): Detección negativa correctamente asignada.

Además se necesita una métrica auxiliar llamada Intersección sobre unión, del ingles *Intersection over Union (IoU)*. En la detección de objetos, la métrica IoU evalúa el grado de superposición entre el *Ground Truth* (gt) y la predicción realizada por un modelo (pd). En el contexto de esta memoria, el *Ground Truth* representa al bounding box del objeto a detectar. El IoU se calcula de la siguiente manera:

$$IoU = \frac{area(gt \cap pd)}{area(gt \cup pd)}$$

Visualmente, la Intersección sobre Unión se define de la siguiente manera:

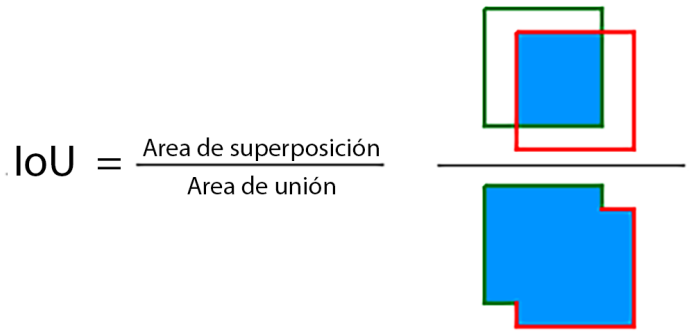


Figura 3.8: Diagrama IoU.

Donde el cuadrado verde de la figura 3.8 representa el *Ground Truth* y el cuadrado rojo la predicción realizada. IoU varía entre 0 y 1, con 0 la no superposición y 1 la superposición perfecta entre *Ground Truth* y la predicción. Cuando IoU vale 1, se considera como Verdadero Positivo.

Debido a que una superposición perfecta es muy complicada para cualquier modelo, se utiliza un umbral o *threshold* para definir desde que valor se considera Verdadero Positivo (VP). Por lo tanto, sea α un umbral, si $IoU(gt, pd) \geq \alpha$ la predicción se considera como Verdadero Positivo y si $IoU(gt, pd) < \alpha$ se considera como Falso Positivo. Así, por ejemplo, sea $\alpha = 0.5$, los casos de Verdadero Positivo, Falso Positivo y Falso Negativo se observan en la figura 3.9 abajo:

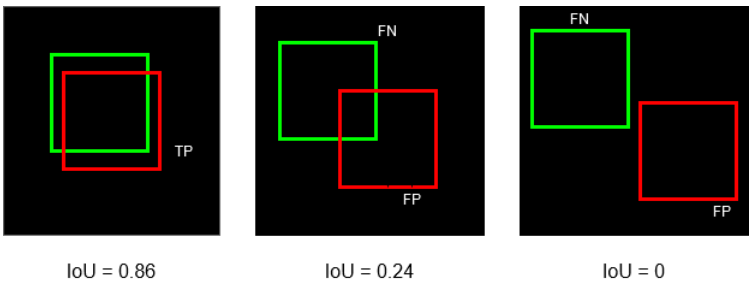


Figura 3.9: Ejemplo de casos en IoU.

Cabe señalar que si el umbral fuese $\alpha > 0.86$, el primer ejemplo marcado como Verdadero Positivo cambiaría a Falso Positivo.

De manera que se define AP de la siguiente manera:

$$AP_{@IoU=\alpha} = \frac{TP}{TP + FP} * 100$$

Donde α el umbral para el cálculo de TP.

Capítulo 4

Resultados y Análisis

El propósito de este capítulo es mostrar los distintos resultados obtenidos en este proyecto con su respectivo análisis. En primer lugar se presenta el entrenamiento realizado sobre la base de datos de COCO, con la metodología ya explicada. Una vez entrenado el modelo se realiza la comparación entre embeddings.

4.1. Entrenamiento

Es esencial recalcar que el objetivo de este nuevo modelo es la generación de propuestas de objetos y descriptores eficaces para objetos que no están necesariamente presentes en la base de datos utilizada para entrenar CenterNet. Por consiguiente, la propuesta de re-entrenar CenterNet con las modificaciones planteadas es para que el modelo se concentre en realizar la detección y clasificación de forma independiente. No obstante, es esencial lograr resultados competentes en la clasificación de COCO para evaluar, en una primera instancia, la validez de la modificación realizada en CenterNet.

Para evaluar la eficacia de este nuevo modelo con respecto a la base de datos de COCO, se empleará la misma métrica utilizada por CenterNet, denominada *Average Precision (AP)*, visto en 3.1.3. Así mismo, se medirá el *loss* generado por las distintas salidas del modelo. Sin embargo no se mostrará los resultados de la salida *off_loss* debido a su bajo valor, el cual no suma un análisis relevante. La salida *off_loss* ayuda a corregir errores en la predicción del ancho y alto del rectángulo que enmarca un objeto detectado.

Además, es relevante señalar que el tiempo necesario para completar una época de entrenamiento es aproximadamente de 100 minutos. Este cálculo se basa en el uso de una unidad de procesamiento gráfico (GPU) en una estación de trabajo *Nvidia DGX* [19].

4.1.1. Decisión de entrenamiento

Como se explicó en la metodología, se contemplaron dos enfoques para llevar a cabo el entrenamiento del nuevo algoritmo: el primero consiste en generar una función de pérdida (*loss*) para cada una de las 4 'ramas' o salidas, mientras que el segundo implica modificar las capas internas pero manteniendo 3 salidas y generar únicamente 3 *losses*, siguiendo el enfoque de CenterNet. En los resultados que se presentarán a continuación, se expondrán las razones que respaldan la elección adoptada.

4.1.1.1. Un *loss* por salida

En esta versión de aplicó una función de pérdida *CLE* para la salida de *scores* y función de pérdida *FocalLoss* para el mapa de calor único. Estos resultados se muestran en la figura 4.1:

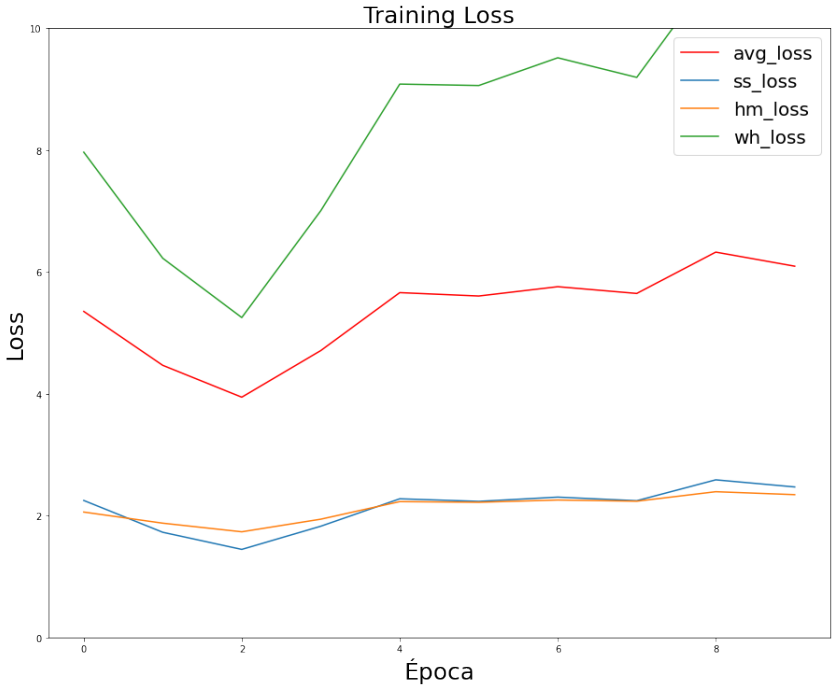


Figura 4.1: Resultado de modelo con heatmap único y scores.

A pesar de las modificaciones realizadas en CenterNet para crear el nuevo modelo, se encuentra un problema significativo durante el entrenamiento. Al analizar la Figura 4.1, se observa que la función de pérdida (*loss*) no disminuye en ninguna época, sino que aumenta progresivamente. Este comportamiento indica que el modelo no está aprendiendo eficazmente la tarea de detección y clasificación de objetos, a pesar de la implementación de cambios. En resumen, la modificación actual del modelo parece no contribuir positivamente a su capacidad de aprendizaje, sugiriendo la necesidad de una revisión exhaustiva de las adaptaciones introducidas.

En particular, la función de pérdida de *FocalLoss* presenta incompatibilidades con un mapa de calor único, que a diferencia de CenterNet, la cantidad de píxeles que representan el fondo es significativamente menor. Para ilustrar esta discrepancia, sea una imagen $I \in R^{512x512}$ en la que hay 6 objetos a detectar, el mapa de calor *Ground truth* generado por CenterNet original sería un tensor de tamaño $80x128x128$, lo que equivale a 1.310.720 píxeles en total. De estos, solo 6 representan un objeto, mientras que 1.310.714 representan el fondo. En contraste, la modificación introduce un mapa de calor único *Ground truth* con un tensor de tamaño $128x128$, con 16.384 píxeles en total, donde solo 6 representan un objeto y 16.378 representan el fondo. Si se compara con el algoritmo modificado, CenterNet tiene 80 veces más la cantidad de píxeles que representan al fondo. Esta drástica reducción en la representación del fondo impacta en la capacidad de *FocalLoss* para abordar efectivamente la desigualdad de clase. La función de pérdida se basa en asignar mayor peso a los ejemplos difíciles, aquellos con menor probabilidad de clasificación correcta. Sin embargo, la marcada disminución en la

representación del fondo en la modificación podría resultar insuficiente para que *FocalLoss* capture adecuadamente la dificultad inherente en la detección de objetos. Esta discrepancia estructural podría ser una causa central detrás de la observación de un *loss* bajo, incluso en las primeras etapas del entrenamiento, como se evidencia en la Figura 4.1.

4.1.1.2. Multiplicación de mapa de calor único con *scores*

Luego de no lograr los resultados esperados en la primera iteración, se opta por modificar la estrategia de entrenamiento al implementar la multiplicación del mapa de calor único. Este ajuste se realiza con la intención de seguir un enfoque más cercano al utilizado por CenterNet. A continuación, en las figuras 4.2 y 4.3, se presentan los resultados obtenidos tras esta adaptación.:

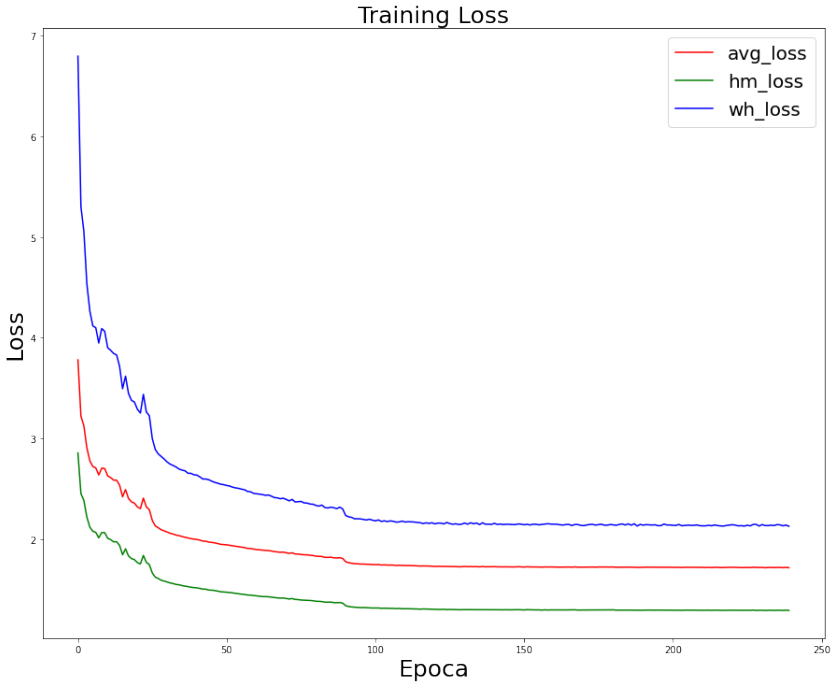


Figura 4.2: Resultado de training loss de modelo nuevo.

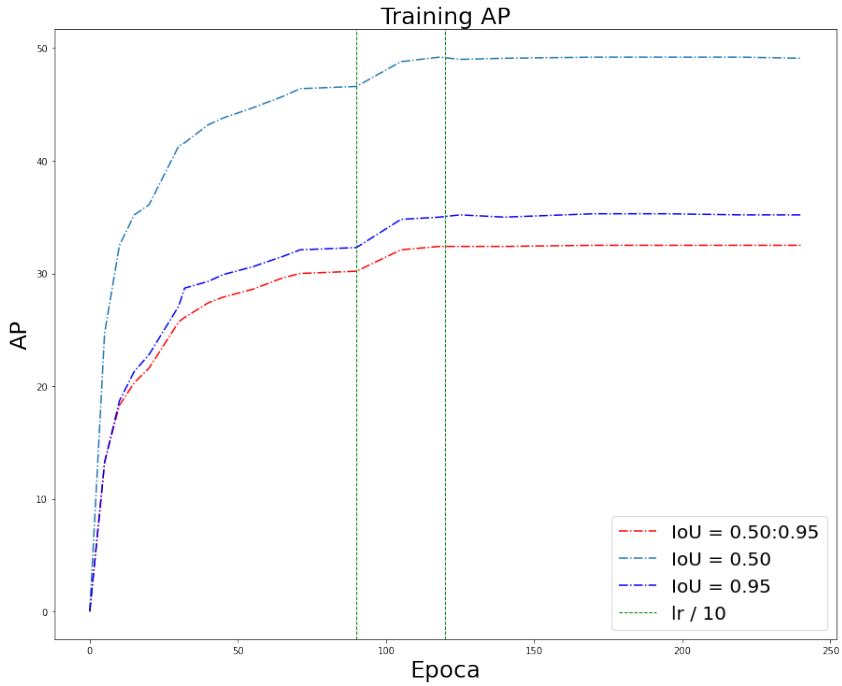


Figura 4.3: Resultado de AP de modelo nuevo.

En la Figura 4.2, se aprecia la disminución progresiva del *loss* a medida que avanzan las épocas, un comportamiento típico en el entrenamiento de modelos. Sin embargo, es importante destacar que esta tendencia descendente del *loss* no garantiza un rendimiento equivalente en una base de datos de validación, como se evidencia en la Figura 4.3, donde se presenta el Average Precision (AP) sobre datos de validación.

A pesar de la inferencia intuitiva que se podría hacer al examinar ambas figuras, indicando que un mayor número de épocas de entrenamiento podría resultar en mejores resultados, esta suposición no se materializó. Después de la época 120, el AP oscila en su máximo sin mostrar mejoras significativas. Adicionalmente, en la Figura 4.3 se destacan líneas verticales verdes en las épocas 90 y 120, que indican la disminución del learning rate en un factor de 10. En la época 90, se observa un aumento significativo en el AP, sugiriendo una mejora en el rendimiento del modelo. Sin embargo, en la época 120, a pesar de que el AP mejora levemente, ya no se registran mejoras notables, indicando una posible estabilización del aprendizaje en esa fase del entrenamiento, logrando el máximo en la época 220, indicado en la tabla 4.1. Este comportamiento refuerza la complejidad de optimizar el rendimiento del modelo a medida que avanza el entrenamiento.

A continuación, se muestran las tablas 4.2 y 4.1, que muestran los resultados obtenidos tras el entrenamiento del modelo nuevo y CenterNet original:

Tabla 4.1: AP Modelo Nuevo.

Épocas	**IoU=0.50:0.95**	**IoU=0.50**	**IoU=0.75**
220	32.5 AP	49.2 AP	35.2 AP

Si se contrastan los valores de Average Precision (AP) logrados por el modelo nuevo, como

Tabla 4.2: AP CenterNet Original.

Epocas	**IoU=0.50:0.95**	**IoU=0.50**	**IoU=0.75**
230	37.4 AP	55.1 AP	40.8 AP

se presenta en la Tabla 4.1, con los obtenidos por el CenterNet original, detallados en la Tabla 4.2, se observa que el modelo nuevo alcanza aproximadamente el 90 % del AP logrado por el CenterNet original. Este enfoque en la generación de embeddings resulta fundamental para entender la utilidad y eficacia del modelo modificado. Aunque la comparación directa de los valores de Average Precision (AP) sugiere una diferencia del 10 % con respecto al CenterNet original, este factor por sí solo no refleja completamente el rendimiento del modelo en la tarea de detección de objetos particulares.

La verdadera prueba de la efectividad del modelo radica en su capacidad para generar embeddings significativos que capturen las características distintivas de los objetos en la imagen. Dado que la tarea principal del proyecto se centra en este aspecto, el hecho de que el modelo nuevo alcance aproximadamente el 90 % del AP de CenterNet sugiere que la generación de embeddings podría estar siendo gestionada de manera eficiente, incluso si los resultados en términos de AP no son idénticos.

4.2. Resultados y análisis de nuevo modelo

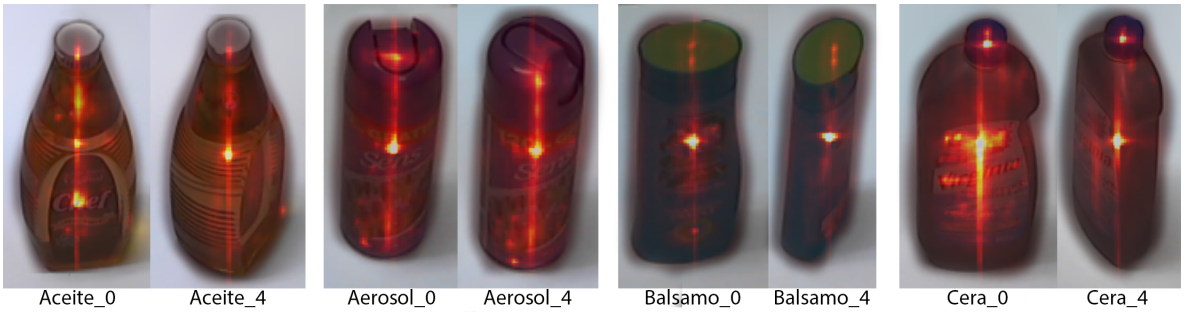
Después de obtener resultados competentes en la detección y clasificación en la base de datos de COCO, el análisis se enfoca en examinar los mapas de calor y mapas de características generados por el nuevo modelo.

Se seleccionan imágenes que contienen objetos no presentes en la base de datos de COCO. Específicamente, se extraen un par de vistas de cada objeto nuevo. En la figura 4.4, se presentan las imágenes utilizadas junto con sus nombres:

Figura 4.4: Imágenes de referencia.

4.2.1. Mapa de calor único

Lo primero que se analiza después de procesar cada una de estas imágenes con el nuevo modelo es la generación del mapa de calor único para cada objeto. Con fines de visualización, se superpondrá el mapa de calor en la imagen original, tal como se muestra en la figura 4.5:



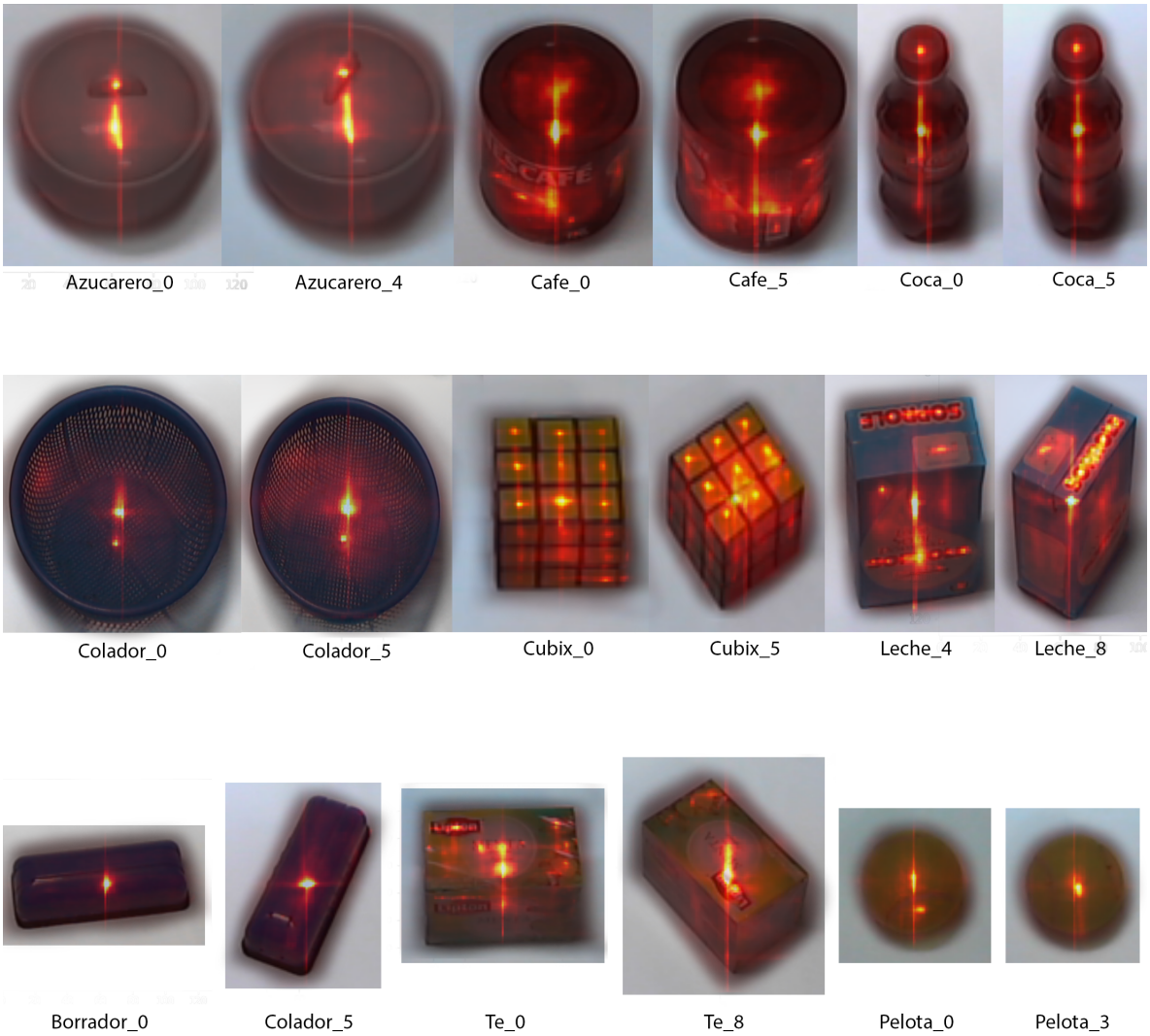


Figura 4.5: Imágenes de referencia con mapa de calor superpuesto.

Es destacable que ninguno de estos objetos se encuentra en la base de datos de COCO, sin embargo, el modelo logra identificar el centro de cada objeto de manera efectiva. Al observar algunos objetos, como *Aceite_0*, se sugiere que el modelo genera puntos de interés en áreas de la imagen con cambios de color. Esta observación es más evidente en objetos como *Cubix_0 o Cubix_5*, donde cada cuadrado del Cubix es señalado como punto de interés, aunque el punto más brillante sigue siendo el centro. Este patrón también se aprecia en los objetos *Colador o Borrador*, donde el punto de interés máximo es más fácil de identificar, dado que estos objetos tienen un color más uniforme.

Además, es digno de destacar que en los mapas de calor generados para los objetos de *Leche*, el modelo identifica las letras de *SOPROLE* como puntos de interés. Esta capacidad de detección de letras también se observa en objetos como *Cera_0, Te_0 o Te_8*.

4.2.2. Mapa de características

Es importante destacar que esta sección se enfocará principalmente en la clasificación de objetos, dejando de lado la detección de estos. Esto se debe a que el modelo nuevo se basa en

la separación de la detección y clasificación, permitiendo así un análisis más detallado de la capacidad del modelo para generar descriptores efectivos a través de la clasificación de objetos.

Para la extracción de descriptores o *embeddings* se utilizará el mapa de características generado por el modulo verde en la figura 3.5. Este módulo fue entrenado por el clasificador coseno, como se detalló en la metodología.

Una vez calculado el mapa de calor, se identifica el punto máximo, que corresponde al centro del objeto. Posteriormente, se extrae el *embedding* en esa posición. Este proceso se repite para cada uno de los 26 objetos presentados anteriormente. Finalmente, cada una de las imágenes se compara con las otras 25 utilizando la distancia coseno. Los resultados se presentan mediante una matriz de distancias coseno:

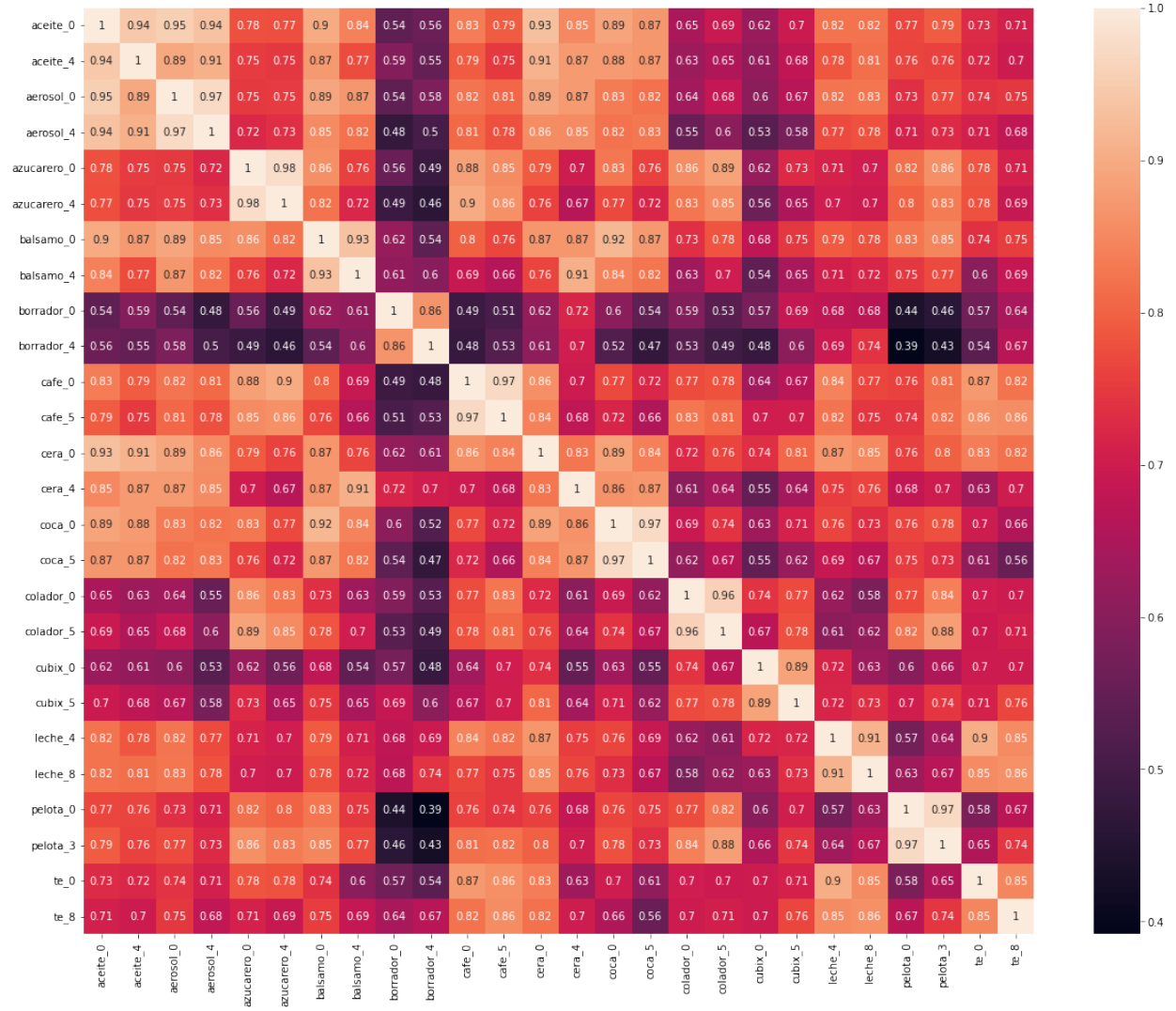


Figura 4.6: Matriz de distancias coseno para 26 objetos.

La diagonal de la matriz representa la comparación de un objeto consigo mismo, lo que justifica el valor máximo de 1. Idealmente, un objeto en la parte izquierda de la matriz debería tener una puntuación más alta con su par que con el resto de los objetos.

Para facilitar la visualización de estos resultados, se extraen los objetos, indicando la posición del píxel con el valor máximo en el mapa de calor, el valor del píxel en el mapa de calor, el puntaje máximo de la distancia coseno con los otros objetos y el nombre del objeto. Esto simula, de manera ideal, la comparación de una imagen de entrada con una base de datos. Los resultados son los siguientes:

N Objeto	Ys	Xs	Hm_max	Predicted
Aceite_0	63	64	0.9129	Aerosol_0
Aceite_4	66	63	0.9636	Aceite_0
Aerosol_0	60	64	0.9983	Aerosol_4
Aerosol_4	62	66	0.9975	Aerosol_0
Azucarero_0	59	61	0.9514	Azucarero_4
Azucarero_4	62	63	0.9061	Azucarero_0
Balsamo_0	62	62	0.9997	Balsamo_4
Balsamo_4	61	64	0.9988	Balsamo_0
Borrador_0	63	64	0.9999	Borrador_4
Borrador_4	63	64	0.9999	Borrador_0
Café_0	63	63	0.9307	Café_5
Café_5	63	63	0.9319	Café_0
Cera_0	19	67	0.9981	Aceite_0
Cera_4	62	63	0.9999	Balsamo_4
Coca_0	63	59	0.9364	Coca_5
Coca_5	61	65	0.9325	Coca_0
Colador_0	64	61	0.998	Colador_5
Colador_5	66	63	0.9977	Colador_0
Cubix_0	64	64	0.9302	Cubix_5
Cubix_5	58	41	0.9983	Cubix_0
Leche_4	90	65	0.9896	Leche_8
Leche_8	63	65	0.9886	Leche_4
Pelota_0	61	63	0.9786	Pelota_3
Pelota_3	65	66	0.9805	Pelota_0
Te_0	65	63	0.971	Leche_4
Te_8	64	64	0.9999	Leche_8

Figura 4.7: Tabla de resultados máximos por objeto.

En relación con la posición de los objetos, en general, ambos ejes coinciden entre los objetos, ya que todas las imágenes utilizadas están bastante centradas. Dado que la imagen final del mapa de calor tiene dimensiones $\mathbb{R}^{128x128}$, los valores de 64 son centrales, coincidiendo en términos generales. No obstante, esto no sucede para los objetos *Cera_0, Cubix_5, Leche_4*, resaltado en amarillo en la figura 4.7. A pesar de estas desviaciones en la posición, no se observa una correlación clara entre una posición desviada y la predicción de clase.

La parte más crucial de estos resultados es analizar cuántas veces la comparación identificó correctamente su par. Si se calcula los casos correctos en relación con el número total de comparaciones, la precisión obtenida en estas imágenes es del 80.77 %. Este valor es significativo para validar de manera preliminar el funcionamiento del algoritmo, ya que cada imagen

se compara con otras 25. Desde un punto de vista probabilístico, la probabilidad de acertar correctamente el par de un objeto dentro de esta base de datos es del 4 %, lo que descarta la *casualidad probabilística*.

A pesar de observar resultados positivos que respaldan, de manera preliminar, la hipótesis inicial en el desarrollo de este nuevo algoritmo, es esencial tener en cuenta las limitaciones establecidas en la configuración de las pruebas hasta el momento. La comparación actual se realiza entre pares de imágenes, cada una con dos vistas del mismo objeto y centradas. Sin embargo, se reconoce que esta configuración representa un escenario idealizado, donde las variaciones en la posición y orientación de los objetos son limitadas. La evaluación completa del algoritmo deberá abordar desafíos más diversos al expandir las pruebas hacia escenarios que reflejen situaciones del mundo real, donde los objetos pueden presentarse en diversas posiciones y orientaciones. Este enfoque será considerado en las secciones posteriores del análisis.

4.3. Comparación con CenterNet

Esta sección tiene como objetivo realizar una comparación de resultados con respecto a CenterNet Original para confirmar o rechazar la hipótesis propuesta en la metodología.

Para comprender adecuadamente los resultados, es necesario recordar la hipótesis formulada en esta memoria. El proyecto se originó a partir de la idea de que al separar la salida de mapas de calor de CenterNet en dos salidas (una para la clasificación de objetos mediante la distancia coseno y otra para la detección de objetos mediante un mapa de calor único, también llamado 'Objectness'), los embeddings generados por la salida de clasificación probablemente serían adecuados para su uso como descriptores. Esto permitiría realizar comparaciones entre embeddings y determinar si los descriptores pertenecen al mismo objeto. Al mismo tiempo, se plantea que el modelo nuevo generará un mapa de calor único que probablemente detecte objetos fuera de la base de datos de entrenamiento.

Para llevar a cabo esta comparación, se realizarán las mismas pruebas que al modelo nuevo, pero en el modelo CenterNet sin modificar. Es por esto que, para la generación de *embeddings*, se extraen las características generadas por la convolución de kernel 3x3 en la salida de mapas de calor. En la figura 3.1, se observa la convolución 3x3 en el cuadrado verde en la rama superior de la arquitectura del modelo.

Ahora bien, CenterNet no detecta ni clasifica objetos que no estén en la base de datos de COCO. Recordando que en el modelo propuesto se extrae el punto de calor máximo que representa el centro de un objeto, pero CenterNet genera 80 mapas de calor, donde cada uno representa una clase y dado que los objetos de prueba no pertenecen a COCO, estos mapas de calor no sirven para detectar los objetos. Por lo tanto, por simplicidad, y dado que los objetos de prueba están centrados y recortados, se utilizan los puntos $(x, y) = (64, 62)$ para la extracción de embeddings, puntos que representan el centro promedio de los objetos, calculado de la tabla 4.7.

Una vez extraídos los embeddings en la posición establecida, se comparan los embeddings de todos los objetos obteniendo la matriz de distancias coseno ilustrada en la siguiente figura:

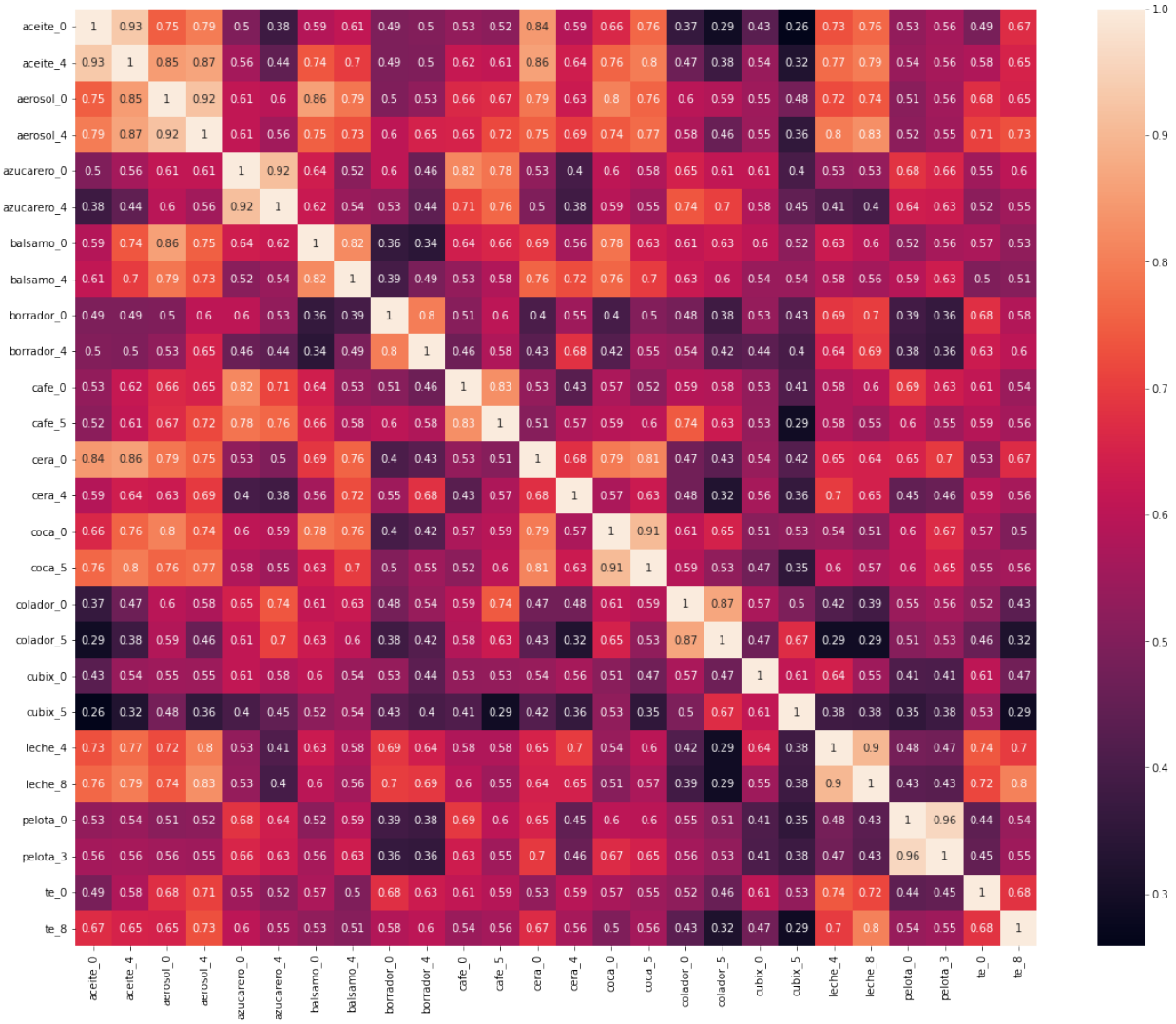


Figura 4.8: Matriz de distancias coseno para 26 objetos, CenterNet Original.

Además, se extraen los datos para un análisis más detallado, los cuales se presentan en la siguiente tabla:

N Objeto	Predicted
Aceite_0	Aceite_4
Aceite_4	Aceite_0
Aerosol_0	Aerosol_4
Aerosol_4	Aerosol_0
Azucarero_0	Azucarero_4
Azucarero_4	Azucarero_0
Balsamo_0	Aerosol_0
Balsamo_4	Balsamo_0
Borrador_0	Borrador_4
Borrador_4	Borrador_0
Café_0	Café_5
Café_5	Café_0
Cera_0	Aceite_4
Cera_4	Balsamo_4
Coca_0	Coca_5
Coca_5	Coca_0
Colador_0	Colador_5
Colador_5	Colador_0
Cubix_0	Leche_4
Cubix_5	Colador_5
Leche_4	Leche_8
Leche_8	Leche_4
Pelota_0	Pelota_3
Pelota_3	Pelota_0
Te_0	Leche_4
Te_8	Leche_8

Figura 4.9: Tabla de resultados máximos por objeto, CenterNet Original.

En este caso, la precisión proporcionada por la extracción y comparación de embeddings es del 73.07 %.

Un primer análisis de estos datos revela que los embeddings generados por CenterNet Original tienen un rendimiento ligeramente inferior al modelo propuesto. Sin embargo, al observar la matriz de distancias coseno, se nota que la diferencia entre la distancia del par del objeto y la segunda mejor distancia es mayor que en el modelo propuesto. Por ejemplo, el objeto *Aceite_0* tiene una distancia de 0.93 con su par *Aceite_4*, mientras que el segundo mejor puntaje, correspondiente al objeto *Cera_0*, tiene una distancia de 0.84. En comparación, en el modelo propuesto, el objeto *Aceite_4* tiene una distancia de 0.94 con su par y una distancia de 0.91 con su segundo mejor puntaje.

Al profundizar en este análisis comparativo, se observa que varios de los errores en la predicción de pares de objetos se replican tanto en CenterNet Original como en el modelo propuesto. Específicamente, se encuentran dificultades notables con los objetos Te y Cera en ambos modelos. Ambos objetos no se emparejan de manera correcta en ninguna de las implementaciones, e incluso se emparejan erróneamente con el mismo objeto, lo que sugiere un desafío persistente en la detección y comparación de estos elementos en ambas configura-

ciones.

En contraste, en el caso de objetos como Cubix, el modelo propuesto logra prever correctamente su par, aunque con una distancia coseno inferior a 0.90. Este resultado podría indicar que CenterNet, ya sea en su forma original o modificada, experimenta dificultades específicas al detectar y clasificar objetos con las características particulares de Cubix. La diferencia en las distancias coseno entre los modelos revela que, aunque ambos enfrentan desafíos similares, el modelo propuesto logra una mejor discriminación entre objetos cercanos, incluso en situaciones donde la clasificación exacta puede ser difícil.

4.4. Detección de objetos mediante *objectness*

En esta sección, se explorará la detección de instancias de objetos mediante el *objectness* creado por el modelo nuevo. El *objectness* hace referencia al mapa de calor único entregado por la modificación del modelo. En teoría, y como se mostró anteriormente, un objeto no presente en la base de datos de COCO tiene un mapa de calor asociado bastante preciso. Sumando este mapa de calor con la predicción que realiza CenterNet con respecto al tamaño de un bounding box, se puede recuperar el bounding box final, logrando así una detección. Cabe mencionar que esta sección se centrará en la detección de objetos y en la recuperación de los bounding boxes finales.

Como se mencionó anteriormente en la sección 3.3, esta versión modificada entrega una matriz de puntaje, el mapa de calor único (*objectness*), la matriz *wh* y la matriz *reg*. La matriz *wh* predice el ancho y alto de un bounding box para un píxel en particular, mientras que la matriz *reg* predice un error para el centro (*x, y*) que se extrae del píxel más luminoso del mapa de calor. Por lo tanto, para recuperar el bounding box, se puede realizar la siguiente operación:

$$\text{bbox} = \left[(\overline{x} + \overline{reg_x}) - \frac{\overline{wh}_x}{2}, (\overline{y} + \overline{reg_y}) - \frac{\overline{wh}_y}{2}, (\overline{x} + \overline{reg_x}) + \frac{\overline{wh}_x}{2}, (\overline{y} + \overline{reg_y}) + \frac{\overline{wh}_y}{2}\right]$$

Donde $(\overline{x}, \overline{y})$ representan el centro del bounding box, $(\overline{reg_x}, \overline{reg_y})$ la corrección del centro, $\overline{wh}_x$ el ancho del bounding box y $\overline{wh}_y$ el alto.

Dado que en una imagen puede haber más de un objeto, la información proporcionada por un solo punto de calor en el mapa de calor no es suficiente. Por lo tanto, se seleccionan los K puntos más destacados en el mapa de calor, los cuales representarían K objetos detectados en la imagen.

A modo de prueba, se obtienen los K=8 objetos más destacados en la siguiente imagen, donde los bounding boxes están etiquetados del 0 al 7, siendo 0 el objeto más destacado:

Figura 4.10: Detección de objetos con K=8.

Al observar la figura 4.10, en la cual hay solo 6 objetos, se logra detectar con éxito todos los objetos. Sin embargo, dado que $K = 8$, hay 2 bounding boxes adicionales que no corresponden a objetos presentes en la imagen. Además, si se limitara $K = 6$, un objeto, en este caso, el desodorante, quedaría excluido de la detección, ya que se observa una detección errónea con una prioridad mayor a la derecha de la imagen.

En estas pruebas, es crucial tener en cuenta que no se ha aplicado un filtro para las propuestas de bounding boxes. Generalmente, en un modelo de detección de objetos, esta tarea recae en la clasificación, donde se establece un umbral para cada propuesta y se descartan aquellas que no superan dicho umbral.

Para obtener resultados más sólidos, se calcularán diversas métricas, como IoU, Recall, Precisión y F-Score, para diferentes valores de K e IoU_{th}. La base de datos utilizada para este análisis está compuesta por 160 imágenes, cada una con 6 objetos. Estos objetos consisten en 40 elementos que no pertenecen a la base de datos COCO, misma base de datos utilizada en [17]. Estas métricas se ven reflejadas en la tabla 4.3:

Tabla 4.3: Métricas de detección para diferentes valores de K e IoUth.

	K	IoU	Recall	Precision	Fscore
IoUth = 0.75	5	0.8336	0.5414	0.645	0.5887
	6	0.833	0.6224	0.6177	0.62
	7	0.833	0.659	0.5607	0.6059
	8	0.8324	0.6663	0.4961	0.5687
	9	0.8205	0.6705	0.4438	0.5341
IoUth = 0.50	5	0.7877	0.7859	0.9363	0.8545
	6	0.7898	0.9119	0.9052	0.9085
	7	0.787	0.9622	0.8188	0.8847
	8	0.7861	0.977	0.7273	0.8339
	9	0.7861	0.9821	0.65	0.7823

Observando la tabla 4.3, se observa que la configuración con $K = 6$ presenta el valor de F-Score más equilibrado. Este resultado tiene sentido, dado que las imágenes utilizadas contienen precisamente 6 objetos.

Es notable también que, a medida que se incrementa el valor de K, el recall mejora, mientras que la precisión disminuye. Esto se debe a que al aumentar K, el modelo tiene más oportunidades de detectar los objetos correctos, lo que incrementa el recall. Sin embargo, tener más oportunidades de detección también implica una mayor probabilidad de errores, lo que disminuye la precisión. Este fenómeno destaca el compromiso entre recall y precisión al ajustar el valor de K.

Finalmente, en cuanto a los umbrales de IoU, es evidente que al tener un umbral menor (0.50), los valores de recall y precisión mejorarán. Esto se debe a que permitir una mayor libertad en el solapamiento entre el bounding box predicho y el ground truth aumenta la probabilidad de que una predicción se considere correcta. Sin embargo, reducir el umbral de IoU a 0.5 impacta directamente en la precisión de los bounding boxes, como se refleja en los valores de IoU en la tabla 4.3, los cuales disminuyen para un umbral menor. Una propuesta de bounding box menos precisa puede causar problemas al intentar clasificar el objeto posteriormente. Este trade-off entre recall y precisión resalta la importancia de elegir cuidadosamente el umbral de IoU según los requisitos específicos de la aplicación.

4.5. Unión de Detección y Clasificación

En esta sección, se integrará la detección y clasificación del modelo para realizar los últimos análisis y conclusiones. En particular, se ha observado que el modelo es capaz de generar propuestas de bounding boxes para los objetos. Además, se confirmó que en un entorno ideal,

el modelo es competente en la clasificación de los objetos, como se detalló previamente. Sin embargo, para una evaluación completa, es crucial analizar cómo se comporta el modelo cuando se combinan ambas tareas, es decir, cuando se realiza la clasificación de los objetos detectados mediante los bounding boxes propuestos. Este enfoque integrado permitirá evaluar la capacidad del modelo para proporcionar una solución completa de detección y clasificación de objetos en un entorno más desafiante.

Para llevar a cabo esta integración, se calculará la distancia coseno entre cada propuesta de objeto y una base de datos de descriptores de imágenes de referencia, de igual manera a la utilizada en YoloSPoC en la figura 2.12. Para la construcción de esta base de datos de descriptores, se realizó un promedio de las vistas de todas las imágenes de referencia.

De manera preliminar, se realiza una prueba con la misma imagen presentada en la figura 4.10. En este caso, se aplica un $K = 10$, pero se filtran todas las propuestas de bounding boxes que tengan una distancia coseno menor a un umbral de 0.80. El resultado obtenido se muestra en la figura 4.11:

Figura 4.11: Prueba combinada con K=10.

El primer resultado notable es que se eliminan correctamente las detecciones erróneas presentadas en la figura 4.10, detectando únicamente los objetos de interés. Sin embargo, al momento de realizar la clasificación, a pesar de tener una distancia coseno umbral elevada, la clasificación lograda es incorrecta en todos los casos.

Para realizar un análisis más exhaustivo, se repetirá el procedimiento de calcular las métricas de IoU, Recall, Precisión y F-Score para $K = 6$ y $K = 10$ con un umbral de IoU de 0.50 y 0.75. Esto permitirá evaluar el rendimiento integral del modelo al combinar la detección de objetos y la clasificación basada en la distancia coseno con la base de datos de descriptores de referencia. Estas métricas se ven reflejadas en la siguiente tabla:

Tabla 4.4: Métricas de detección para diferentes valores de K e IoUth y umbral aplicado.

	K	IoU	Recall	Precision	Fscore
IoUth = 0.75	6	0.8341	0.5099	0.6191	0.5592
	10	0.8330	0.5498	0.4165	0.4740
IoUth = 0.50	6	0.7880	0.7502	0.9108	0.8228
	10	0.7870	0.8091	0.6129	0.6974

La tabla 4.4 considera superar un umbral donde la distancia coseno es 0.8. No obstante, estos datos no reflejan la clasificación precisa de las propuestas de bounding boxes.

Cuando se restringe un caso correcto (TP) definiéndolo como una propuesta que tiene una distancia coseno cercana a la del ground truth, las métricas disminuyen significativamente:

Tabla 4.5: Métricas de detección y clasificación para diferentes valores de K e IoUth.

	K	Recall	Precision	Fscore
IoUth = 0.50	7	0.0399	0.0420	0.0409

Observando los resultados de las tablas 4.4 y 4.5, se infiere que el principal desafío del modelo radica en la clasificación de los objetos. Aunque el modelo logró resultados destacables en el escenario ideal de una propuesta bien recortada y centrada, experimenta fallos sustanciales al enfrentarse a situaciones más realistas, donde la detección y clasificación de objetos fallan de manera significativa.

En el enfoque convencional de clasificación de objetos, el modelo se entrena para reconocer y asignar imágenes a clases específicas. Sin embargo, en el caso de la detección y clasificación de objetos a nivel de instancias, donde se busca identificar y clasificar objetos individuales en una imagen, surge un desafío adicional. Los modelos de detección como CenterNet se entrenan para prever la presencia de clases generales y no necesariamente para distinguir entre instancias específicas de una clase.

En otras palabras, CenterNet se centra en aprender patrones generales asociados con clases, pero no necesariamente aprende representaciones específicas para cada instancia única de una clase. Esto puede resultar en dificultades cuando se trata de clasificar objetos específicos en situaciones del mundo real, donde las variaciones en la apariencia y posición de los objetos son comunes.

Una aproximación más efectiva podría ser entrenar modelos que aprendan representacio-

nes específicas de objetos, lo que implica capturar características y detalles únicos de cada instancia. Sin embargo, este enfoque también podría requerir conjuntos de datos más grandes y variados, así como técnicas avanzadas de entrenamiento.

En resumen, el desafío principal radica en adaptar el modelo para aprender representaciones más detalladas y específicas de objetos individuales, lo que podría mejorar significativamente la precisión en la detección y clasificación en entornos del mundo real.

4.5.1. Cálculo de descriptores con modelo externo

Observando los resultados de las secciones 4.4 y 4.5 se nota que el modelo es capaz de generar buenas propuestas de objetos, pero falla al momento de clasificar mediante la comparación de descriptores (embeddings). Para confirmar esto, y mejorar estos resultados, se utiliza un modelo externo para el cálculo de descriptores. En este caso, se utiliza el modelo propuesto en *Fine-tuning CNN Image Retrieval with No Human Annotation* [20], que utiliza una combinación de los resultados de resnet para generar un descriptor óptimo para una imagen.

De igual manera, una vez obtenidos las propuestas de objetos, estos son ingresados a este nuevo modelo. Los resultados para la imagen de prueba se observan en la figura 4.12:

Figura 4.12: Prueba con modelo externo.

Con esto se confirma que los descriptores generados por CenterNet no son lo suficientemente buenos. En el caso del modelo externo falla solo en uno de los objetos.

4.6. Mejora del modelo con SAM: Segment Anything Model

Se consideró la posibilidad de mejorar el modelo propuesto mediante SAM (Semgent Anything Model), una técnica que permite segmentar cualquier imagen, sin necesidad de haber visto casos previamente. A modo de ejemplo, se muestra la figura 4.13, donde se utilizó SAM:

Figura 4.13: Prueba de SAM sin entrada.

Cuando se aplica SAM a una imagen sin ningún 'prompt' o entrada adicional, se genera una segmentación completa de la imagen. Sin embargo, las instancias que se desean detectar no presentan una segmentación uniforme. Por lo tanto, para obtener una segmentación precisa, se proporciona a SAM una imagen de entrada junto con una entrada que indica el bounding box, en este caso, extraído de las detecciones del modelo propuesto en esta memoria. Este resultado se muestra en la figura 4.14:

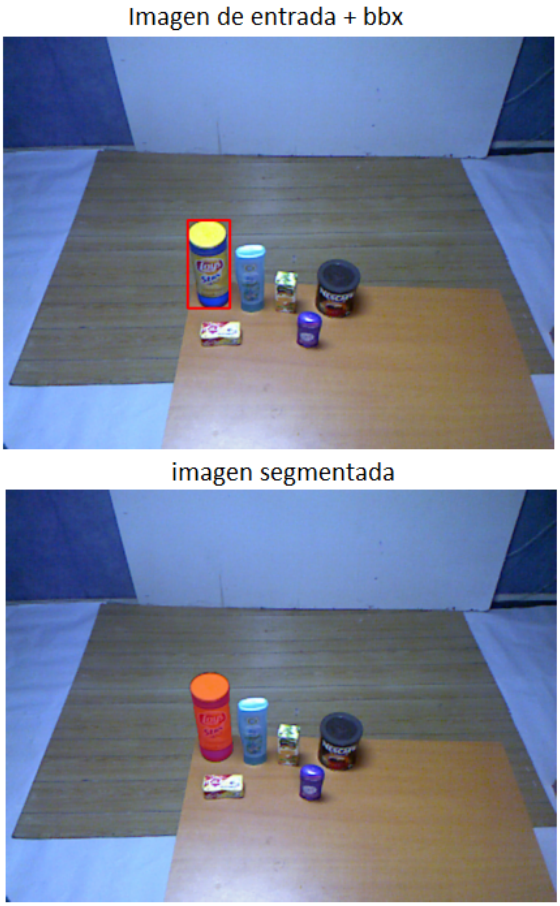


Figura 4.14: Prueba de SAM con bounding box.

Ahora se observa una segmentación uniforme y completa para el objeto seleccionado.

Una vez segmentado el objeto, se realiza un promedio de los embeddings de los píxeles dentro de la segmentación. La hipótesis subyacente es que si, en lugar de considerar solo utilizar el embedding central, se utilizan todos los embeddings de la instancia de objeto, este conjunto podría contener más información del objeto y así se lograría una mejor comparación utilizando la distancia coseno.

Sin embargo, a pesar de tener más información, los resultados empeoran, lo cual sugiere que la forma en que CenterNet trabaja con los mapas de calor no es compatible con una segmentación. Esto se debe a que toda la información relevante del objeto se concentra en el punto de calor máximo, y la segmentación no aporta beneficios significativos en este contexto.

Capítulo 5

Conclusión y posible extensión de trabajo

El propósito principal de este capítulo es consolidar la información presentada hasta ahora y reflexionar sobre los resultados y análisis llevados a cabo. Es crucial subrayar que esta propuesta de modelo surge de una idea experimental de investigación que, en última instancia, podría ser correcta o incorrecta.

El primer desafío en este proyecto fue poder replicar CenterNet, para luego hacer un modelo modificado que pudiese entrenarse de una manera correcta. Reproducir los resultados de CenterNet no fue trivial, dado que en primer lugar, CenterNet fue publicado en 2019, por lo que está implementado en un entorno de trabajo desactualizado. En particular, originalmente CenterNet se implementó con *Pytorch 0.4.1* y para este proyecto se adaptó a una versión más reciente, *Pytorch 1.4*. Este proceso de adaptación lo considero muy importante ya que trabajar con códigos de alta complejidad no es sencillo y requiere un basto aprendizaje autónomo.

Al alcanzar los objetivos mencionados anteriormente, otro desafío significativo fue modificar CenterNet para desarrollar el modelo propuesto y llevar a cabo su entrenamiento. Al examinar los datos y realizar el análisis correspondiente en el capítulo respectivo, se concluye que la hipótesis fue confirmada solo de manera parcial. Específicamente, se hace referencia a la posibilidad dentro de la hipótesis de que el modelo propuesto pudiera detectar objetos sin necesidad de que estos estuvieran en la base de datos de entrenamiento. Como se demostró en los resultados, este aspecto de la hipótesis se logró con éxito, aunque se reconoce la necesidad de una mejora en el proceso de extracción si se desea probar esta detección en imágenes más complejas de *interpretar*, lo cual no era el objetivo principal de esta memoria.

Dicho lo anterior, a pesar de los buenos resultados en la detección de objetos, el enfoque principal de esta memoria es la detección de objetos particulares. Por lo tanto, además de la detección del objeto, es crucial poder distinguir entre objetos de manera precisa. Aunque los resultados en este aspecto no fueron completamente desalentadores, la comparación entre los embeddings generados por el modelo propuesto y los embeddings extraídos directamente de CenterNet mostró que los embeddings del modelo propuesto eran solo ligeramente mejores. Esto refuta la hipótesis que sugiere que los embeddings generados por un modelo basado en mapas de calor, que separa la detección y la clasificación, son competentes para realizar

comparaciones entre objetos.

Cuando se aborda una idea experimental de investigación, es fundamental reconocer que los resultados pueden variar y, en este caso, los resultados preliminares no proporcionan un estímulo convincente para continuar investigando sobre este modelo. La posible razón del fallo en la clasificación radica en que el sistema (CenterNet) se entrena para clasificar según clases, no para clasificar objetos particulares. Para lograr la clasificación efectiva de objetos específicos, probablemente sería necesario incorporar características que representen vistas específicas de objetos en lugar de depender de clases de objetos, ya que estas últimas utilizan información de nivel superior.

De todas formas, para lograr resolver el objetivo propuesto de lograr un modelo basado en CenterNet que pueda detectar y clasificar imágenes, se utilizó un modelo externo para generar los descriptores. Esto sirve para resolver el problema propuesto, pero no se desarrolla más esta idea ya que el objetivo principal era generar descriptores utilizando la modificación implementada en CenterNet.

Además, se puede estudiar que tanto influye el alcance de AP que tuvo el entrenamiento del modelo, en este caso se logró un 90 % el AP logrado por CenterNet. Por lo que es posible una mejora si se puede lograr o superar el AP de CenterNet.

En definitiva, los supuestos presentados en esta memoria se lograron parcialmente, pero hay posibles extensiones que podrían conducir a la completa realización de la hipótesis. Estas extensiones, aunque escapan de lo factible en un trabajo de pregrado, se proponen en la siguiente lista:

1. Se puede revisitar la idea fallida de tener 4 salidas, con 4 funciones de perdida respectivamente, que puedan, siendo optimista, mejorar los resultados de CenterNet.

2. Probar lo que se conoce como *Fine-Tuning*, es decir, re-entrenar el modelo con una base de datos adaptada para el reconocimiento de objetos particulares. Este re-entrenamiento se realizaría utilizando el modelo previamente entrenado en esta memoria, inicialmente con la base de datos de COCO. Sin embargo, esto podría implicar perder la búsqueda de un modelo entrenable de principio a fin.

En conclusión, este proyecto fue significativo para mi desarrollo como estudiante de Ingeniería Civil Eléctrica con especialización en inteligencia artificial. En particular, a pesar de no detallarse explícitamente en este ensayo, se emplearon diversas herramientas que representan conocimientos importantes para mi futuro profesional, tales como *Docker, Nvidia DGX, Github, SSH, Linux*, entre otros. Estas experiencias prácticas han enriquecido mi conjunto de habilidades y contribuirán significativamente a mi desarrollo como profesional en el campo de la inteligencia artificial.

Bibliografía

[1] Thompson, N. C., Greenewald, K., Lee, K., y Manso, G. F., "The computational limits of deep learning," 2020, doi:10.48550/ARXIV.2007.05558.

[2] Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., y Wierstra, D., "Matching networks for one shot learning," 2016, doi:10.48550/ARXIV.1606.04080.

[3] Yann LeCun, León Bottou, Y. B. y Haffner, P., "Gradient-based learning applied to document recognition," 1998, http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf.

[4] Dan C. Cires¸an, Ueli Meier, J. M. L. M. G. J. S., "Flexible, high performance convolutional neural networks for image classification," 2011, https://people.idsia.ch//~juergen/ijcai2011.pdf.

[5] Alex Krizhevsky, I. S. y Hinton, G. E., "Imagenet classification with deep convolutional neural networks," 2012, https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[6] He, K. y Sun, J., "Convolutional neural networks at constrained time cost," en Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5353–5360, 2015.

[7] Srivastava, R. K., Greff, K., y Schmidhuber, J., "Highway networks," arXiv preprint arXiv:1505.00387, 2015.

[8] He, K., Zhang, X., Ren, S., y Sun, J., "Deep residual learning for image recognition," en Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

[9] Girshick, R., Donahue, J., Darrell, T., y Malik, J., "Rich feature hierarchies for accurate object detection and semantic segmentation," en Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580–587, 2014.

[10] Guyon, I. y Elisseeff, A., "An introduction to variable and feature selection," Journal of machine learning research, vol. 3, no. Mar, pp. 1157–1182, 2003.

[11] Girshick, R., "Fast r-cnn," en Proceedings of the IEEE international conference on computer vision, pp. 1440–1448, 2015.

[12] Ren, S., He, K., Girshick, R., y Sun, J., "Faster r-cnn: towards real-time object detection with region proposal networks," IEEE transactions on pattern analysis and machine intelligence, vol. 39, no. 6, pp. 1137–1149, 2016.

[13] Redmon, J., Divvala, S., Girshick, R., y Farhadi, A., "You only look once: Unified, real-time object detection," en Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788, 2016.

[14] Law, H. y Deng, J., "Cornernet: Detecting objects as paired keypoints," en Proceedings of the European conference on computer vision (ECCV), pp. 734–750, 2018.

[15] Zhou, X., Wang, D., y Krähenbühl, P., "Objects as points," CoRR, vol. abs/1904.07850, 2019, http://arxiv.org/abs/1904.07850.

[16] Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., y Zitnick, C. L., "Microsoft COCO: common objects in context," CoRR, vol. abs/1405.0312, 2014, http://arxiv.org/abs/1405.0312.

[17] Loncomilla, P. y Ruiz-del Solar, J., "Yolospoc: Recognition of multiple object instances by using yolo-based proposals and deep spoc-based descriptors," en Robot World Cup, pp. 154–165, Springer, 2019.

[18] Yu, F., Wang, D., Shelhamer, E., y Darrell, T., "Deep layer aggregation," 2017, doi:10.48550/ARXIV.1707.06484.

[19] NVIDIA, "Nvidia dgx platform," 2023, https://www.nvidia.com/en-us/data-center/dgx-platform/.

[20] Radenović, F., Tolias, G., y Chum, O., "Fine-tuning cnn image retrieval with no human annotation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 7, pp. 1655–1668, 2019, doi:10.1109/TPAMI.2018.2846566.