

UNIVERSIDAD EUROPEA DE MADRID
ESCUELA DE INGENIERÍA, ARQUITECTURA Y DISEÑO
MÁSTER EN BIG DATA



Detección de logos en el fútbol chileno utilizando YOLO

TRABAJO FINAL DE MÁSTER

Joaquín Vargas
22332530

Madrid - España
Octubre 2024

Índice general

1. Introducción y trabajos relacionados	1
1.1. Motivación	1
2. Estado del arte	4
2.1. YOLO	4
2.2. Evolución de YOLO	5
2.3. GoogLeNet	6
3. Pre-procesamiento de imágenes	10
3.1. Recopilación de imágenes	10
3.2. Creación de la base de datos	11
4. Metodologías	12
4.1. Modelo YOLOv8	12
4.2. Modelo YOLOv8 con mejoras	15
5. Experimentos y discusión	17
5.1. Matriz de confusión	19
5.2. F1 Curve y PR Curve	21
5.3. Análisis de muestra	24
5.4. Imágenes sin marcas por detectar	26
5.5. Aplicación	27
6. Conclusión	30
7. Anexo	32
7.1. Non-maximum Suppression	32
7.2. Función de pérdida CIoU	32
7.3. Función de pérdida DFL	34

Capítulo 1

Introducción y trabajos relacionados

1.1. Motivación

La identificación de logotipos comerciales en eventos deportivos, como los partidos de fútbol, ha aumentado su relevancia en áreas como el marketing y la publicidad en los últimos años. Dada la creciente difusión de eventos de este tipo, las marcas desean aumentar su visibilidad y evaluar con exactitud el retorno de sus inversiones publicitarias. En este escenario, la visión artificial se ha convertido en una herramienta fundamental para mejorar el reconocimiento y análisis de los logotipos que aparecen en las transmisiones deportivas. En este campo de la inteligencia artificial, los algoritmos de detección de objetos han mostrado ser efectivos (Bochkovskiy et al. (2020)).

Entre los algoritmos de visión artificial, los modelos YOLO (You Only Look Once) han destacado por su capacidad para realizar detección de objetos en tiempo real con alta precisión (Redmon et al. (2016)). Una de sus últimas versiones, YOLOv8, contempla mejoras significativas en términos de velocidad y exactitud con respecto a sus versiones previas (Terven y Cordova-Esparza (2023)), haciendo que sea especialmente adecuada para aplicaciones en entornos dinámicos y de alta demanda, como lo son los eventos deportivos en vivo. La capacidad de YOLOv8 para procesar imágenes en tiempo real permite a los analistas obtener información instantánea sobre la exposición de las marcas durante un partido, optimizando así las estrategias de marketing relacionadas con la exposición de determinada marca comercial.

La implementación de YOLOv8 en la detección de logos comerciales en el fútbol presenta varios desafíos, ya que los partidos de fútbol son eventos complejos con múltiples variables ambientales difíciles de predecir. La capacidad de identificar con precisión los logos en estas condiciones puede proporcionar datos valiosos para la

optimización de estrategias publicitarias y la cuantificación del retorno de inversión. En este sentido, YOLOv8 ofrece una herramienta veloz y precisa que mejora los resultados obtenidos con respecto a versiones anteriores (Jocher et al. (2023)).

El proceso de detección de logotipos comerciales con YOLOv8 comienza utilizando el modelo preentrenado proporcionado por YOLO, que se mejora con un conjunto de imágenes específicas de los logotipos que se desean detectar. Es crucial incluir en el conjunto de datos imágenes capturadas desde diferentes ángulos y posiciones para obtener un modelo más sólido. Una vez entrenado, este modelo puede aplicarse tanto a transmisiones en vivo como a grabaciones para identificar y contar la aparición de logotipos específicos en tiempo real.

Además de su aplicación en la publicidad, la detección de imágenes con YOLOv8 puede tener otras aplicaciones relevantes (Sahel et al. (2021)). Por ejemplo, puede ser utilizada para análisis de marca en redes sociales, monitoreo de patrocinios en otros deportes o eventos, y estudios de mercado (Cleofas et al. (2023)). La versatilidad y precisión de YOLOv8 lo convierten en una herramienta poderosa no solo para el fútbol, sino para cualquier industria donde la visibilidad de la marca es crucial (Bográn Ortiz y Martínez Hernández (2023)). Otras técnicas de detección de objetos, como SSD (Single Shot MultiBox Detector) y Mask R-CNN, también han mostrado potencial en este campo (Liu et al. (2015); He et al. (2017)), pero YOLOv8 ofrece ventajas específicas que lo hacen ser un modelo idóneo para lo que se busca en esta investigación.

Considerando la literatura actual, es posible encontrar bastantes investigaciones relacionadas con la detección de logos, por ejemplo en el artículo de Sahel et al. (2021) donde se entrena un modelo basado en redes neuronales convolucionales para la detección de logos comerciales y se llega a la conclusión de que modelos de Redes Neuronales Convolucionales (CNN) recurrentes obtienen los mejores resultados, pero con un mayor costo de GPU, en Shuo et al. (2018) también se trabaja la detección de logos utilizando un modelo YOLOv3 modificado, donde llegan a resultados bastante robustos para la detección de logos de automóviles. Además, de manera más reciente, algunos estudios como el de Alsheikhy et al. (2020) han propuesto modelos de aprendizaje por transferencia utilizando Densely Connected Convolutional Network (DenseNet), obteniendo resultados satisfactorios para modelos que usan la base de datos de Flicker-logos32. Como se ha expuesto, la literatura de detección de logos es bastante amplia, sin embargo, hasta donde sabemos, no hay casos donde se trabaje la detección de logos en fútbol.

En esta investigación se utiliza el modelo YOLOv8 y un modelo YOLOv8 con hiperparámetros óptimos para la detección de logos en el fútbol chileno, además se le dará una aplicación concreta a estos modelos con el fin de testear a los modelos de identificación como una herramienta para constatar la eficiencia del gasto en publicidad de las marcas evaluadas. Para lograr este objetivo se escogió un equipo de la liga chilena llamado Colo-Colo y dos marcas que se encuentran en su camiseta, las cuales son “Coolbet” y “Assist Card”.

El resto del artículo se organiza de la siguiente manera. La sección 2 comenta la metodología para pre-procesar las imágenes que se utilizarán, comentando detalles sobre la recopilación de imágenes y la creación de la base de datos. La sección 3

detalla el estado del arte actual relacionado con los modelos YOLO. La sección 4 introduce el modelo YOLOv8 y las características que posee el modelo a utilizar en este artículo. La sección 5 detalla los experimentos realizados, una discusión de los resultados y su aplicación con datos reales. Finalmente, en la sección 6 se concluye y se habla sobre proyectos futuros basándose en el desarrollo de estos modelos.

Capítulo 2

Estado del arte

Los modelos YOLO han sido objeto de gran atención desde sus primeras versiones, en este apartado se describe con mayor profundidad el modelo, sus avances a lo largo de los años, y la base de la arquitectura que posee actualmente.

2.1. YOLO

YOLO redefine el problema de la visión artificial, tratándolo como una regresión directa desde los píxeles de la imagen a las coordenadas de las cajas delimitadoras y las probabilidades de las clases, utilizando una única red neuronal. A diferencia de los métodos desarrollados previamente que reutilizaban clasificadores para la detección y empleaban complejos pipelines, YOLO simplifica el proceso, permitiendo la optimización de extremo a extremo directamente en el rendimiento de detección.

Como se define en Redmon et al. (2016), este método divide la imagen de entrada en una cuadrícula y cada celda de esta cuadrícula es responsable de predecir múltiples cajas delimitadoras y sus correspondientes puntajes de confianza. Estos puntajes reflejan tanto la probabilidad de que una caja contenga un objeto como la precisión de dicha predicción. Además, cada celda predice las probabilidades de clase condicionadas a la presencia de un objeto en ella. Este enfoque permite que YOLO realice predicciones globales sobre la imagen completa, capturando información contextual que mejora la precisión de la detección.

La arquitectura de YOLO, que se puede observar en la Figura 2.1, se inspira en el modelo GoogLeNet, utilizando 24 capas convolucionales seguidas de 2 capas completamente conectadas. La versión rápida de YOLO, denominada Fast YOLO, reduce el número de capas para aumentar la velocidad de procesamiento. Gracias a este diseño, YOLO puede procesar imágenes en tiempo real, logrando hasta 45 cuadros por segundo con el modelo base y 155 frames por segundo con la versión rápida. Esto lo convierte en una solución ideal para aplicaciones que requieren pro-

cesamiento de video en tiempo real con baja latencia.

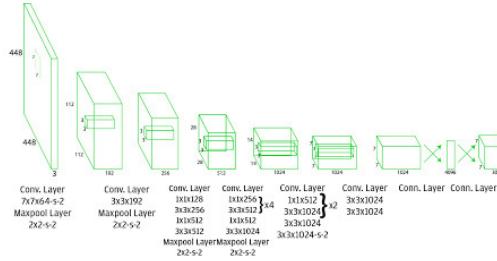


Figura 2.1: Arquitectura YOLO, Fuente: (Redmon et al. (2016)).

Comparado con otros sistemas de detección en tiempo real, YOLO sobresale en velocidad y precisión. En el conjunto de datos PASCAL VOC, Fast YOLO alcanza una precisión media del 52.7 %, mientras que la versión estándar alcanza una precisión media del 63.4 %. Estos resultados muestran que YOLO es más de dos veces más preciso que otros detectores en tiempo real, aunque puede cometer más errores de localización en comparación con métodos más avanzados con redes convolucionales. Sin embargo, su capacidad para generalizar y aprender representaciones robustas de objetos lo hace altamente versátil para diferentes dominios. (Redmon et al. (2016))

2.2. Evolución de YOLO

Si bien, YOLO es un modelo que desde sus comienzos ha mostrado buenos resultados en la detección de objetos, este destaca por sus modelos preentrenados que han ido evolucionando a lo largo de los años como se observa en la Figura 2.2. La primera versión (YOLO v1), lanzada en 2015, generó cambios en la forma en que se analizaba el problema de la visión artificial.

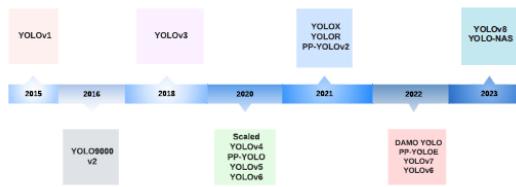


Figura 2.2: Linea de tiempo de versiones YOLO, Fuente: (Terven y Cordova-Esparza (2023)).

En base a la revisión bibliográfica realizada por Terven y Cordova-Esparza (2023), YOLOv1 introdujo el concepto de detección de objetos como un problema de regresión, prediciendo directamente las coordenadas de las cajas delimitadoras y las probabilidades de clase. Este enfoque permitió una velocidad sin precedentes, pero tuvo limitaciones en la precisión, especialmente con objetos pequeños y

agrupados. YOLOv2 mejoró significativamente al introducir las cajas de anclaje, permitiendo al modelo predecir más adecuadamente las dimensiones de los objetos. Además, incorporó Batch Normalization (que es la normalización de los inputs de las capas para que el entrenamiento sea más rápido) y una resolución de entrada de múltiples escalas, lo que mejoró tanto la precisión como la estabilidad del entrenamiento.

Con YOLOv3, se adoptó una arquitectura de red más profunda y compleja basada en Darknet-53, que permitió mejoras en la capacidad de detección a múltiples escalas mediante una detección en tres diferentes niveles de granularidad. Esto resultó en un rendimiento superior en términos de precisión, manteniendo una velocidad adecuada para aplicaciones en tiempo real. La transición de YOLO a PyTorch en esta versión también facilitó la integración y uso de los modelos en una amplia gama de aplicaciones.

Las versiones posteriores, como YOLOv4 y YOLOv5, continuaron refinando la arquitectura y las técnicas de entrenamiento. YOLOv4 introdujo CSPDarknet53 como su backbone, junto con optimizaciones como la convolución de mosaico, que mejoraron la generalización del modelo y su rendimiento en conjuntos de datos variados. YOLOv5, aunque no fue desarrollado por los creadores originales de YOLO, trajo mejoras en la facilidad de uso y eficiencia, utilizando un enfoque modular que simplificó la implementación y la adaptación del modelo a diferentes necesidades.

YOLOv6 y YOLOv7 introdujeron más innovaciones, como técnicas mejoradas en la asignación de etiquetas que permitieron mayor precisión en la detección en YOLOv6 y el uso de la estructura de NAS (Neural Architecture Search) en YOLO-NAS para optimizar automáticamente la arquitectura de la red. Estas versiones se enfocaron en mejorar tanto la precisión como la eficiencia computacional, permitiendo que YOLO siga siendo relevante en aplicaciones modernas que requieren procesamiento en tiempo real.

YOLOv8 y YOLO con transformers representan las iteraciones más avanzadas, integrando técnicas de vanguardia como la atención en el espacio de la imagen y la arquitectura de transformers. Estas innovaciones permiten una mejor captura de relaciones espaciales y contextuales entre los objetos, llevando la precisión de detección a nuevos niveles. (Terven y Cordova-Esparza (2023))

2.3. GoogLeNet

Como se mencionó en las secciones previas, los comienzos de YOLO se levantan en base a la arquitectura de GoogLeNet, es por esto que es importante definir detalles de este tipo de red neuronal convolucional para comprender los inicios de YOLO.

En el artículo Szegedy et al. (2014), se detalla GoogLeNet, también conocido

como Inception v1, es una arquitectura de red neuronal convolucional desarrollada por Google que destacó en el desafío de reconocimiento de imágenes de ImageNet en 2014. Una de las principales características de GoogLeNet es su eficiencia en el uso de recursos computacionales. Utiliza módulos de reducción de dimensión para minimizar el costo computacional sin sacrificar el rendimiento. A pesar de ser una red profunda con 22 capas (cuando se cuentan solo las capas con parámetros), mantiene un bajo presupuesto computacional, permitiendo su ejecución en dispositivos con recursos limitados.

La modularidad es una característica clave de GoogLeNet. Utiliza módulos Inception, que son bloques que combinan convoluciones de diferentes tamaños (1x1, 3x3 y 5x5) y max-pooling. Esto permite que la red capture características en múltiples escalas simultáneamente. Además, los módulos Inception incluyen capas de proyección 1x1¹ para reducir la dimensionalidad antes de aplicar convoluciones más costosas, lo que ayuda a mantener la eficiencia de la red.

Otra característica importante de GoogLeNet es la reducción significativa de parámetros en comparación con las redes anteriores, como AlexNet. Esto se logra gracias a la incorporación de convoluciones 1x1 para la reducción de dimensiones antes de aplicar convoluciones más grandes. Para manejar la gran profundidad de la red y asegurar una buena propagación del gradiente durante el entrenamiento, GoogLeNet incluye salidas de clasificación auxiliares conectadas a capas intermedias de la red. Estos clasificadores auxiliares no solo ayudan a regularizar el modelo, sino que también mejoran la propagación del gradiente.

La arquitectura de GoogLeNet se puede observar en la Figura 2.3, esta se compone de varios tipos de capas estructuradas de manera específica para maximizar la eficiencia y el rendimiento. Las capas iniciales de convolución incluyen una convolución 7x7 seguida de una capa de max-pooling 3x3, y una convolución 3x3 seguida de otra capa de max-pooling 3x3. A partir de aquí, la red se estructura en una serie de módulos Inception, que combinan múltiples convoluciones de diferentes tamaños y capas de max-pooling dentro del mismo módulo. Estos módulos permiten el procesamiento de información a diferentes escalas y la reducción de dimensionalidad previa a la convolución para optimizar el uso de recursos computacionales.

Antes de la capa de clasificación final, GoogLeNet utiliza una capa de average pooling global que reduce las características extraídas a una dimensión manejable para la clasificación. También incluye una capa de dropout para evitar el sobre ajuste, donde se descartan aleatoriamente algunas activaciones durante el entrenamiento. Los clasificadores auxiliares están conectados a los módulos Inception intermedios, específicamente en las salidas de los módulos 4a y 4d. Estos clasificadores consisten en una convolución 1x1 seguida de capas totalmente conectadas y softmax, y se utilizan solo durante el entrenamiento para mejorar la regularización.

La red finaliza con una capa totalmente conectada seguida de una capa softmax que produce las probabilidades finales para las clases objetivo. La arquitectura de GoogLeNet es un ejemplo de cómo la innovación en el diseño de redes neuronales

¹La capa de proyección 1x1 realiza una convolución en cada posición de la entrada abarcando solo 1 píxel, aunque se aplica a todas las profundidades de las características de entrada. Por ejemplo si se tiene una imagen de 32x32x256 (que sería de 32x32 píxeles y 256 mapas de características) al que se le aplica un filtro 1x1 con 128 filtros, se obtendría una imagen depurada de 32x32x128.

puede llevar a mejoras significativas en la eficiencia y el rendimiento sin aumentar drásticamente la complejidad computacional. (Szegedy et al. (2014))

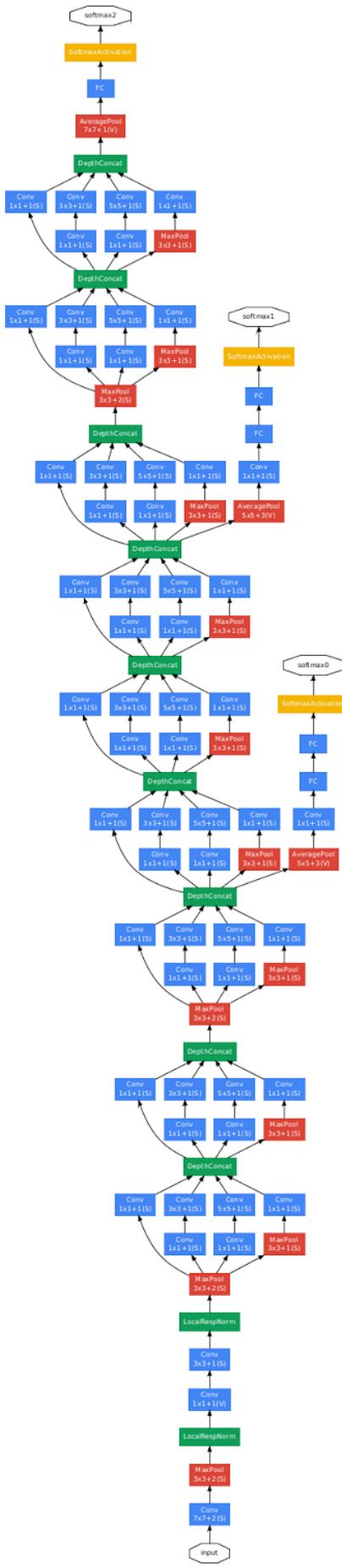


Figura 2.3: Arquitectura GoogleNet, Fuente: (Szegedy et al. (2014)).

Capítulo 3

Pre-procesamiento de imágenes

Pasando al desarrollo del modelo de detección de imágenes que se plantea en esta investigación, un punto crucial es el de la recolección y creación de la base de imágenes que se utilizarán para entrenar al modelo. En esta sección se detallan algunos aspectos de este paso.

3.1. Recopilación de imágenes

Las imágenes fueron capturadas manualmente a través de un programa, basándose en videos de partidos grabados. Estas muestran las marcas comerciales a detectar y se adaptaron para un tamaño de 640 x 640 píxeles en formato PNG.

Los videos que se utilizan como fuente de datos se obtuvieron desde la página web llamada footballia (<https://footballia.net/>), esta funciona como repositorio de partidos a nivel global de manera gratuita y contiene partidos de más de 80 países a lo largo del mundo.

La muestra comprende duelos entre el 16 de enero de 2024 y 6 de abril de 2024, esta muestra se compone de 3 partidos de fútbol seleccionados aleatoriamente entre las fechas mencionadas. Las imágenes capturadas muestran distintos escenarios bajo diversas condiciones climáticas y cambios de color en la camiseta del equipo estudiado, esto con el objetivo de crear un modelo más robusto ante distintas condiciones.

La muestra comprende 208 imágenes de las marcas mencionadas, siendo 106 de la marca “Coolbet” y 102 de “Assist Card”. De estas imágenes se seleccionaron de manera aleatoria y estratificada un 80 % como muestra de entrenamiento y un 20 % para probar el modelo. La selección estratificada asegura de igual manera una muestra equilibrada entre ambos logos comerciales para que no exista un desbalanceo en el modelo.

3.2. Creación de la base de datos

Una vez que se recopilan las imágenes, se crea la base de datos con los elementos descritos. A partir de esto, manualmente se etiquetan las marcas utilizadas en cada uno de los fotogramas recopilados utilizando la herramienta labelImg (<https://github.com/HumanSignal/labelImg>). Para esto, se crean recuadros dentro de cada imagen que contienen los logotipos comerciales investigados y se guardan con su nombre respectivo.

En el caso de tener imágenes que no mostraran claramente los elementos buscados, no se etiquetó para evitar etiquetas confusas que lleven a modelos erróneos. También se reafirma esta idea debido a que se busca un modelo que pueda reconocer marcas tal como lo haría un humano, por lo tanto, es importante mantener una muestra que me acerque lo más posible al reconocimiento de marcas “visibles al ojo humano” y no pequeños recuadros que computacionalmente pueden ser analizados, pero que un humano no identificaría correctamente.

Capítulo 4

Metodologías

Una vez que se tienen los datos, se pasa a la etapa de construcción del modelo.

Con las herramientas existentes actualmente, el entrenamiento para generar un modelo personalizado con la muestra que se posee, es un proceso que se puede desarrollar íntegramente con un lenguaje de programación como Python, y donde no es visible su construcción, sin embargo, es importante conocer como se procesan los datos recolectados. Es por esto, que se muestra la arquitectura del modelo YOLOv8 y como funciona internamente para obtener los resultados que se ven en la siguiente sección.

También, se detalla el proceso de obtención del modelo con hiperparámetros optimizados que se adopta en esta investigación.

4.1. Modelo YOLOv8

Yolov8 es una de las últimas actualizaciones del modelo que tiene su base en YOLO (Redmon et al. (2016)). Este modelo es utilizado para clasificación de imágenes, detección de objetos, etc. Tiene una estructura compuesta por dos componentes: Backbone y Head, como se puede observar en la Figura 4.1.

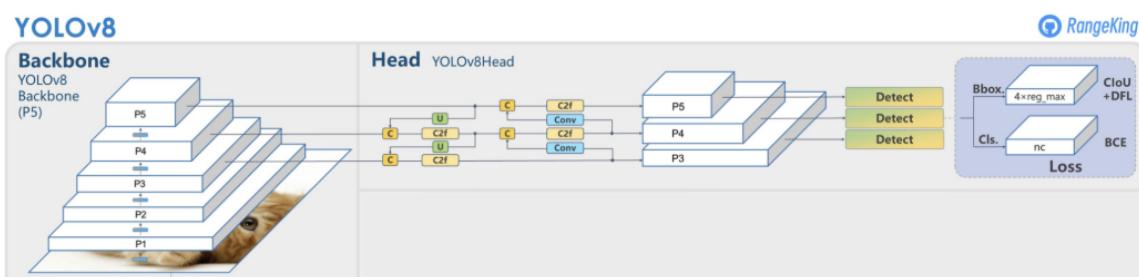


Figura 4.1: Arquitectura YOLOv8, Fuente: Rangeking (2024)

El Backbone es CSPDarknet53, una arquitectura de red que se enfoca en la extracción de características, compuesta por conexiones residuales y kernels pequeños.

Las mejoras fueron propuestas por (Wang et al. (2020)) y se basan en que las conexiones parciales entre las etapas del procesamiento fueran más ricas y disminuyeran la carga computacional. A diferencia de modelos anteriores, la columna tiene algunos cambios en el CSPLayer, ahora llamado módulo C2f¹.

Comúnmente en los modelos YOLO se tenía un componente que unía el Backbone y el Head llamado Neck, este tenía la misión de agregar y refinar lo extraído en la etapa de la columna, buscando mejorar información espacial y semántica a través de diferentes escalas. Se componía de un módulo de pooling de pirámide espacial (He et al. (2015)) que eliminaba la necesidad de recortar, eliminar o deformar imágenes, y de un módulo CSP-Path Aggregation Network (Terven y Cordova-Esparza (2023)) que hacía más directo el traspaso entre las capas inferiores y superiores, incorporando las características aprendidas en el paso previo. Sin embargo, para esta última entrega se eliminó esta parte por la inclusión del módulo C2f.

El head consta de 3 ramas que predice una escala de características diferentes. El modelo original utiliza 3 tamaños de cuadrícula, 13x13, 26x26 y 52x52. Cada rama produce cuadros delimitadores, probabilidades de clase y puntuaciones de confianza. Además, la red utiliza Non-maximum Suppression (NMS)² (Hosang et al. (2017)) para filtrar los cuadros delimitadores superpuestos. En este modelo YOLOv8 se utiliza una cabeza desacoplada para procesar las tareas de objetividad, clasificación y regresión de manera independiente. El diseño de esta manera, permite que cada rama se centre en su tarea y mejore la precisión general del modelo.

La arquitectura desglosada a máximo detalle tiene un aspecto como el que se muestra en la Figura 4.2, en la que se pueden observar la cantidad de neuronas que la componen, el procesamiento de como entran las imágenes, y otros detalles como el C2F reemplazando al cuello que antes existía en las arquitecturas YOLO.

El modelo finalmente predice cuadros delimitadores para predecir la ubicación y tamaño de un objeto dentro de una imagen. Para lograr predecir los cuadros, se predicen las coordenadas de las cajas con respecto a escalas y ratios predeterminados, que ajusta para que se adecuen al aspecto del objeto. Es relevante mencionar que YOLOv8 utiliza funciones de pérdida CIoU³ (Zheng et al. (2020)) y DFL⁴ (Li et al. (2020)) para la pérdida de los cuadros delimitadores y entropía cruzada binaria para la pérdida de clasificación. Estas pérdidas han mejorado el rendimiento de la detección de objetos, principalmente cuando se trata de objetos más pequeños.

Esta arquitectura fue evaluada en el conjunto de datos de prueba MS COCO test-dev 2017, YOLOv8x logró una precisión promedio (AP) del 53.9 % con un tamaño de imagen de 640 píxeles (en comparación con el 50.7 % de YOLOv5 con el

¹Este módulo mejora la eficiencia y capacidad de la red para detectar objetos, ya que a diferencia del CSPLayer que simplemente dividía los flujos de características para mejorar el aprendizaje y reducir la carga computacional, mantiene esta separación parcial de características e incorpora un proceso de fusión que permite que la red capture tanto características de bajo como de alto nivel simultáneamente. Esta combinación de características hace que el C2f sea más efectivo en la detección de objetos, proporcionando una representación más rica y detallada, como también mejorando la precisión sin aumentar significativamente el costo computacional.

²Ver anexo

³Ver anexo

⁴Ver anexo

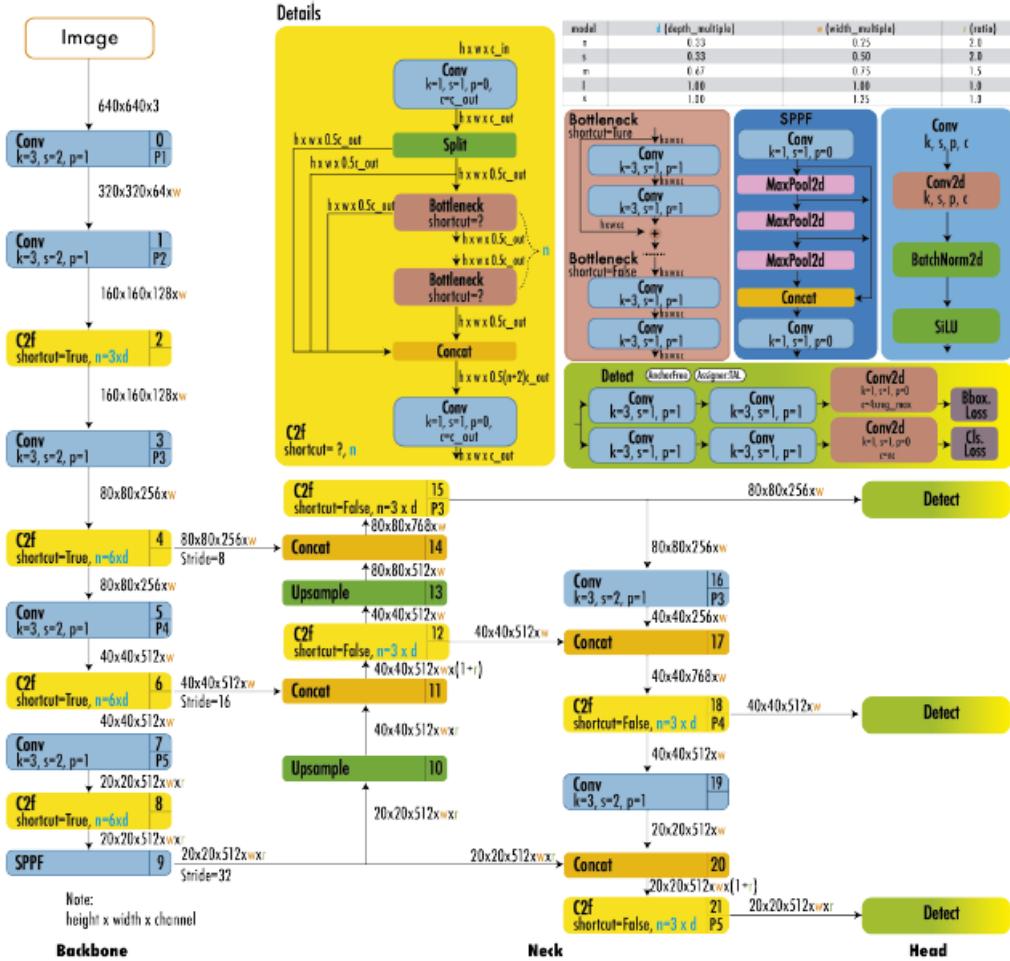


Figura 4.2: Arquitectura YOLOv8 detallada, Fuente: Terven y Cordova-Esparza (2023)

mismo tamaño de entrada) con una velocidad de 280 FPS en una NVIDIA A100 y TensorRT (Terven y Cordova-Esparza (2023)).

Para efectos de esta investigación se utiliza entonces el modelo descrito, en términos simples, el proceso que lleva a cabo la estructura comentada se puede ver representada gráficamente con la Figura 4.3.

En la Figura se puede observar como el modelo toma una foto seleccionada, a la cual le incorpora rejillas de 13 x 13. Cuando el objeto a detectar cae en una cuadrícula de la rejilla, entonces este la debe identificar. Por lo tanto, cada cuadrilla predice cuadros, delimitadores y probabilidades asociadas al objeto que se busca predecir. Cuando más de un cuadro detecta cuadros delimitadores, YOLO utiliza NMS para quedarse con la mejor opción.

Considerando lo expuesto y con el detalle de la arquitectura de YOLOv8, es que entonces se tendrá un modelo principal que nace desde esta arquitectura, y que se entrena con el dataset de imágenes armado que se mencionó en la sección anterior.

Para esta construcción del modelo, se consideran los parámetros por default que nos entrega Ultralytics (plataforma que documenta y desarrolla YOLOv8). El

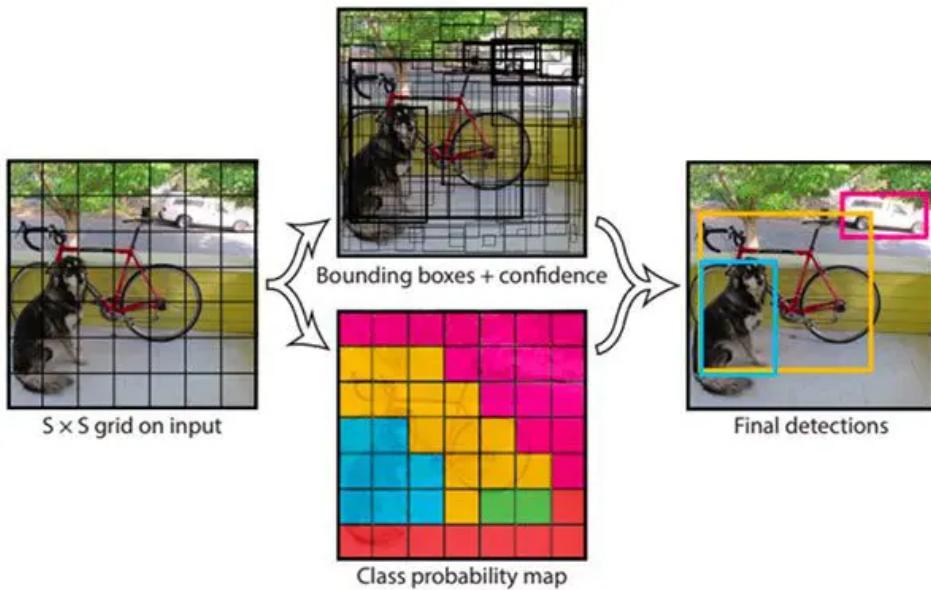


Figura 4.3: Detección YOLOv8, Fuente: Redmon et al. (2016)

proceso se puede llevar a cabo computacionalmente, en este caso como se mencionó utilizando Google Colab.

4.2. Modelo YOLOv8 con mejoras

Para una mayor robustez de la investigación, se plantea de igual manera un modelo YOLOv8 con hiperparámetros optimizados.

A continuación se detalla que es cada hiperparámetro⁵:

- lr0 : Tasa de aprendizaje inicial. Ajustar este valor es crucial para el proceso de optimización, ya que influye en la rapidez con la que se actualizan los pesos del modelo.
- lrf: Tasa de aprendizaje final como una fracción de la tasa inicial = $(lr0 * lrf)$, utilizada junto con planificadores para ajustar la tasa de aprendizaje a lo largo del tiempo.
- momentum: Factor de momento para optimizadores SGD o beta1 para optimizadores Adam, que influye en la incorporación de gradientes pasados en la actualización actual.
- weight_decay: Término de regularización L2, que penaliza los pesos grandes para prevenir el sobreajuste (overfitting).
- warmup_epochs: Número de épocas para el calentamiento de la tasa de aprendizaje, aumentando gradualmente la tasa de un valor bajo hasta la tasa de aprendizaje inicial para estabilizar el entrenamiento en las primeras etapas.

⁵Información extraída desde la documentación de Ultralytics

- warmup_momentum: Momento inicial para la fase de calentamiento, ajustándose gradualmente al momento establecido durante el período de calentamiento.
- warmup_bias_lr: Tasa de aprendizaje para los parámetros de sesgo (bias) durante la fase de calentamiento, ayudando a estabilizar el entrenamiento del modelo en las primeras épocas.
- box: Peso del componente de pérdida de la caja (box loss) en la función de pérdida, que influye en la importancia de predecir con precisión las coordenadas de los cuadros delimitadores.
- cls: Peso de la pérdida de clasificación en la función de pérdida total, afectando la importancia de la predicción correcta de la clase en relación con otros componentes.
- dfl: Peso de la pérdida focal de distribución (distribution focal loss), utilizada en ciertas versiones de YOLO para una clasificación más detallada.

Entonces, a grandes rasgos el modelo planteado en esta subsección posee la misma arquitectura presentada previamente, lo que nos llevará a tener un modelo YOLOv8 con parámetros por defecto definido por la construcción de YOLOv8 y otro modelo con parámetros optimizados en base a una búsqueda que optimice los resultados.

En la práctica, la búsqueda de parámetros óptimos se realiza utilizando las herramientas de tuning proporcionadas por Ultralytics, lo que se lleva a cabo computacionalmente utilizando Google Colab. Es importante destacar que se parte desde la base del modelo principal expuesto en la subsección anterior y en base a este se van modificando levemente ciertos componentes del modelo para buscar mejores resultados. En la siguiente sección se detallan la configuración de cada modelo luego de ser entrenados.

Capítulo 5

Experimentos y discusión

Una vez que ya se tiene la base de imágenes preparada y el marco general desde el cual se sustentan los modelos a utilizar, es que entonces se puede pasar al entrenamiento y revisión del desempeño de los modelos obtenidos.

La base de esta investigación es la detección de marcas comerciales utilizando YOLOv8, para comprobar su efectividad se entrena con una muestra propia de los logos a detectar que fue descrita en el capítulo 3, por lo tanto, el modelo base a utilizar es YOLOv8 con una muestra propia, mientras que existe un modelo mejorado que se genera desde la misma base del modelo principal pero con hiperparámetros optimizados.

Ambos modelos se entrenaron y testearon utilizando los recursos GPU por defecto que ofrece Google Colab, este es NVIDIA Tesla K80 con 12GB de VRAM (Video Random-Access Memory). Los parámetros principales de los modelos están en la Tabla 5.1.

Parámetro	Modelo Base	Modelo Optimizado
lr0	0.01	0.00907
lrf	0.01	0.00968
momentum	0.937	0.95452
weight_decay	0.0005	0.0005
warmup_epochs	3.0	3.02427
warmup_momentum	0.8	0.8
box	7.5	6.65033
cls	0.5	0.54817
dfl	1.5	1.5

Cuadro 5.1: Parámetros en el modelo

Para entrenar los modelos, se crearon sets de imágenes como los que se mencionaban en el capítulo 3, un ejemplo de los fotogramas utilizados es como los que se pueden ver en las Figuras 5.1 ,5.2, 5.3 y 5.4 .



Figura 5.1: Detección Assist Card. Fuente: Elaboración propia



Figura 5.2: Detección Assist Card. Fuente: Elaboración propia

Para los modelos se utilizaron 210 imágenes de entrenamiento repartidas entre las marcas Coolbet y Assist Card. Las métricas para medir el rendimiento del modelo son F1 score y Precision Recall Curve, también se utiliza la Matriz de Confusión.



Figura 5.3: Detección Coolbet. Fuente: Elaboración propia



Figura 5.4: Detección Coolbet. Fuente: Elaboración propia

5.1. Matriz de confusión

La matriz de confusión es una herramienta que permite evaluar el rendimiento de un modelo en base a sus resultados. Esta pseudo-métrica con base en los Verdaderos Positivos (TP), Verdaderos Negativos (TN), Falsos Positivos (FP) y Falsos Negativos (FN), nos entrega una noción de que tan correcta es la clasificación que está llevando a cabo el modelo. En la Figura 5.5 se muestra la matriz de confusión normalizada para el modelo YOLOv8 principal.

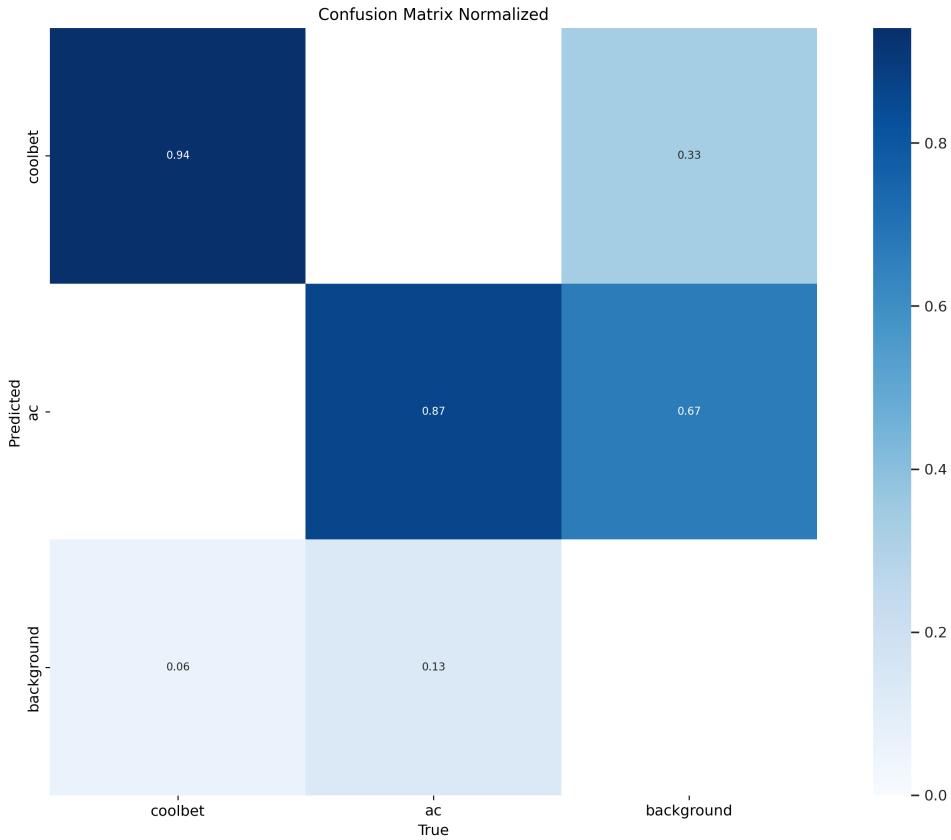


Figura 5.5: Matriz de confusión normalizada, Fuente: Elaboración propia

Basándose en los resultados que se pueden observar en la figura, el modelo predice correctamente un 94 % de las imágenes a las que se enfrenta el modelo para la marca Coolbet, mientras que para la marca Assist card este acierta en un 87 % de los casos.

Estos resultados son buenos considerando que solo entre un 6 % y un 13 % de las imágenes se están clasificando de manera errónea, sin embargo, podrían mejorarse sus resultados.

Para el caso del modelo optimizado, los resultados se pueden ver en la Figura 5.6. Este modelo en base a como se construye, debería mostrar mejores resultados.

Considerando lo que se puede observar en la figura, es que se obtienen mejores resultados que en el modelo base, esto en primer lugar reafirma el hecho de que efectivamente se obtiene un "mejor" modelo. Con respecto a la matriz, este modelo predice correctamente un 97 % de las imágenes referentes a la marca Coolbet, mientras que en el caso de Assist card lo hace en un 90 % de la muestra referida a ese logotipo comercial.

Considerando ambos casos, en términos generales vemos un buen desempeño, donde destaca más el modelo optimizado que tiene solo un 3 % de espacio de mejora en el caso de la marca Coolbet y un 10 % en el caso de Assist card.

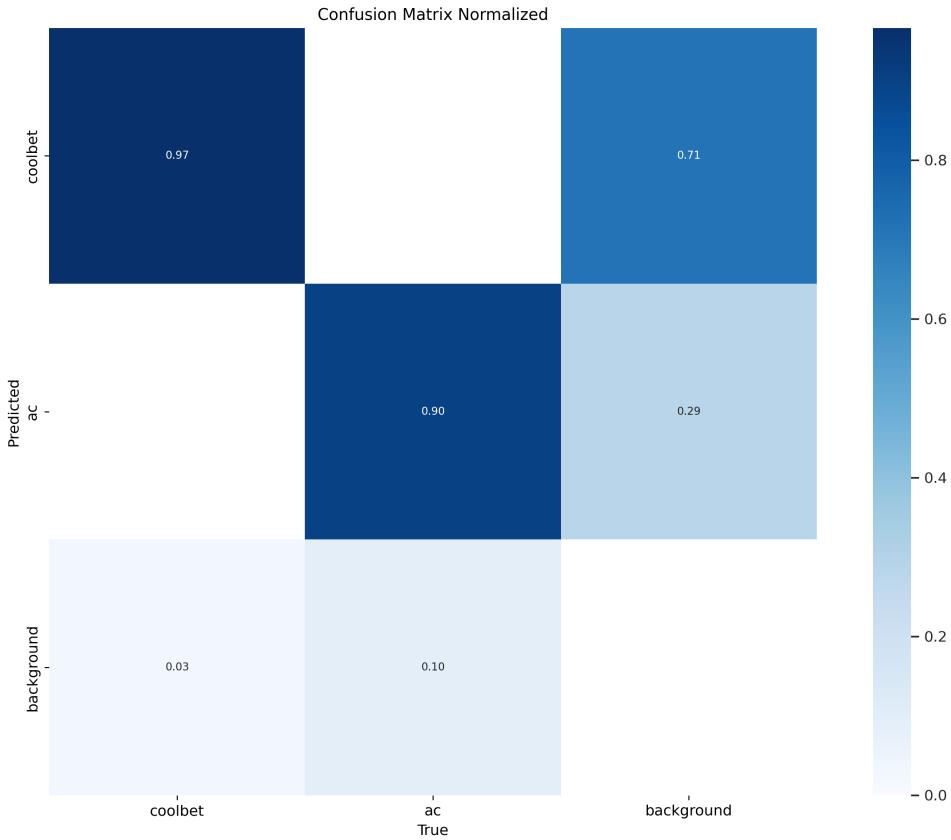


Figura 5.6: Matriz de confusión normalizada, Fuente: Elaboración propia

5.2. F1 Curve y PR Curve

Antes de ver los resultados de estas métricas para los modelos utilizados, es necesario definir cada uno de estos conceptos. La F1 Curve muestra la media ponderada entre Precision y Recall, lo que termina entregando una evaluación equilibrada del rendimiento considerando FP y FN.

En el caso de PR Curve, La precisión cuantifica la proporción de verdaderos positivos entre todas las predicciones positivas, evaluando la capacidad del modelo para evitar falsos positivos. Por otro lado, el Recall calcula la proporción de verdaderos positivos entre todos los positivos reales, midiendo la capacidad del modelo para detectar todos los casos de una clase (Jocher et al. (2023)).

Se incorporan los gráficos de la F1 curve. Al ver los resultados para el modelo principal en la Figura 5.7 se observa que la performance del modelo es de 0.93 para todas las clases, mientras que el umbral óptimo para hacer predicciones es en 0.647. Importante notar que en este modelo la marca Coolbet tuvo un mejor desempeño en su detección que la marca Assist card en bajos niveles de confianza, pero en niveles de confianza más altos Assist card es el que tiene un mejor desempeño.

Para el caso del modelo optimizado, se puede observar en la Figura 5.8 que se obtiene un resultado del 0.95, cuyo umbral óptimo se encuentra en 0.642. Si bien no existen grandes diferencias con respecto al modelo principal, se obtiene un mejor F1 score, pero se baja un poco la confianza en la cual se obtiene el peak de rendimiento.

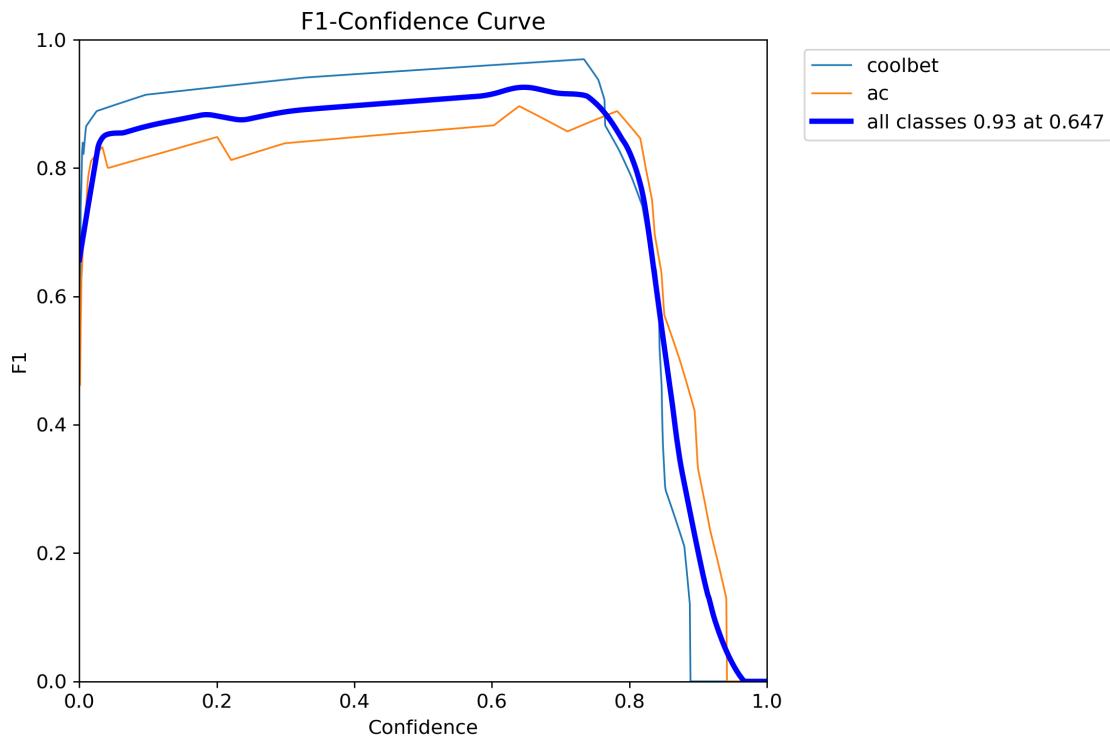


Figura 5.7: F1 Curve, Fuente: Elaboración propia

Al igual que en el otro modelo, Coolbet tiene mejores resultados en umbrales bajos de confianza y Assist card lo sobrepasa para mayores niveles de confianza.

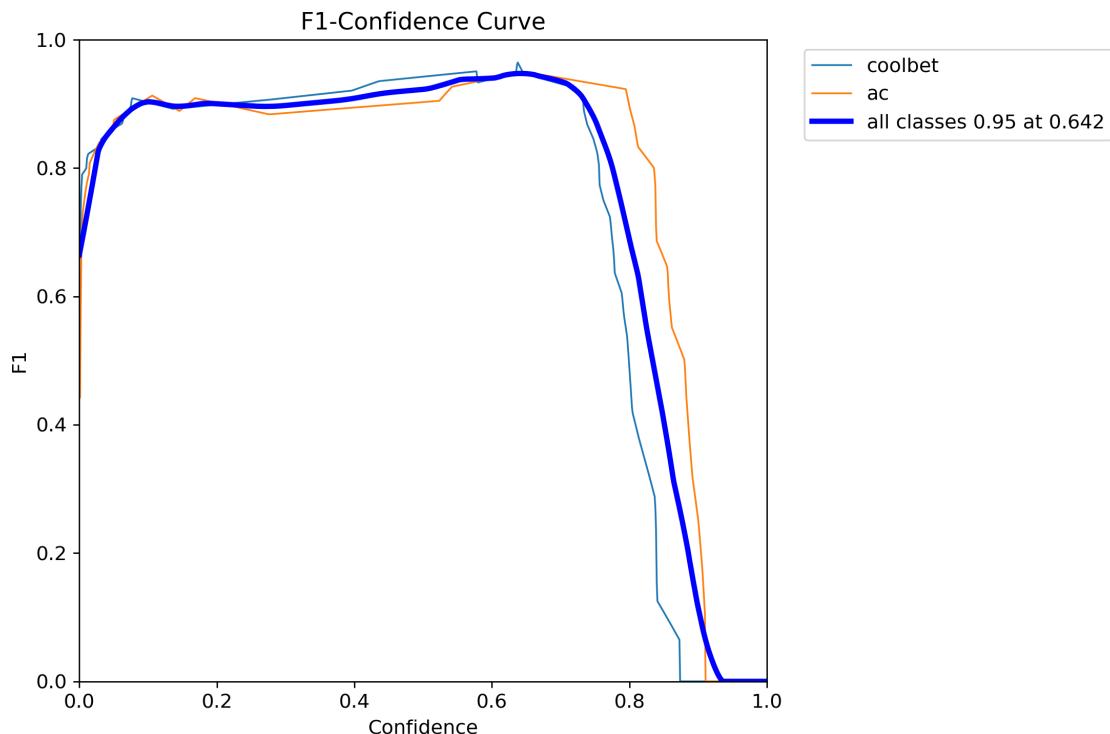


Figura 5.8: F1 Curve, Fuente: Elaboración propia

Para el caso de la Precision-Recall curve, los resultados que se observan en la Figura 5.9 para el caso del modelo principal de YOLOv8 nos muestra que el modelo tiene una precisión promedio del 97% para todas las clases, mientras que destaca Coolbet con un 0.982 por sobre Assist card que tiene un 0.957. Esta gráfica también se evalúa basándose en su AUC, que visualmente se asocia a una curva más cercana al eje derecho y arriba, en este caso está bastante cercana, por lo que se podría asociar a que es un modelo con buenos resultados.

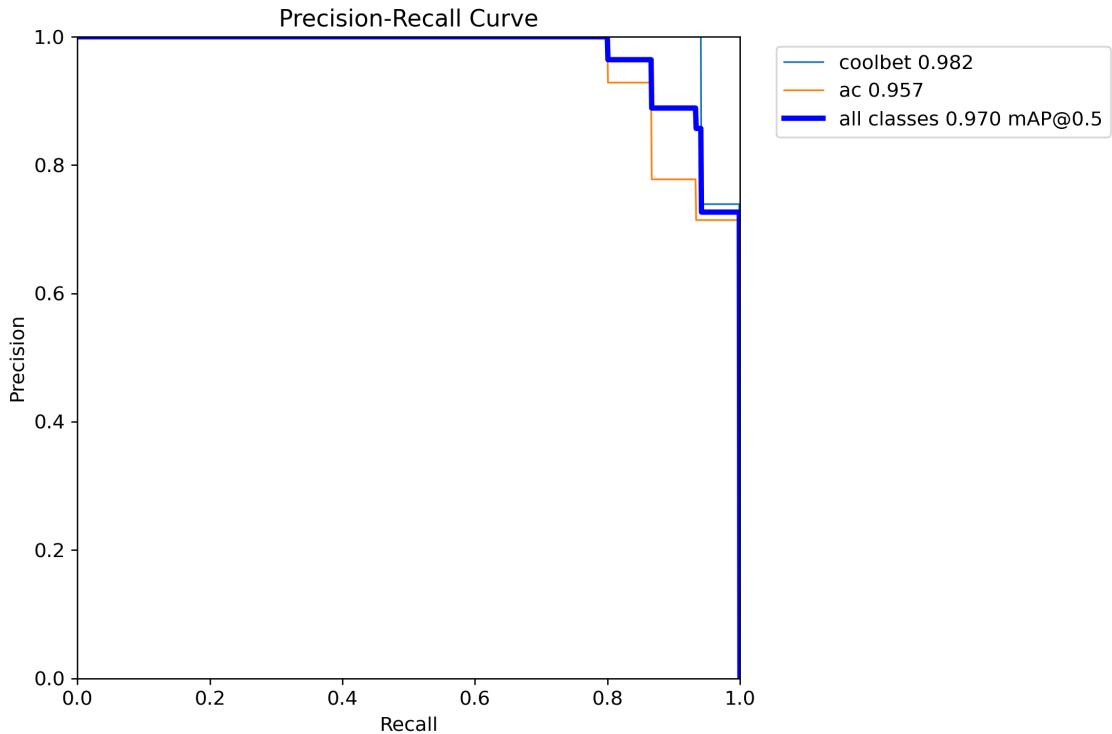


Figura 5.9: Precision-Recall Curve, Fuente: Elaboración propia

Para el modelo optimizado, se obtuvieron los resultados de la Figura 5.10. Acá es interesante notar que se obtiene un mejor resultado que el modelo base, debido a que se tiene una precisión de 99 % aproximadamente para el conjunto de ambas clases. Para el caso de Coolbet nuevamente se tienen mejores resultados, ya que se llega al 99 % y lo sigue de cerca Assist card con un 98 %.

En general, ambas métricas apuntan a buenos resultados de ambos modelos, donde destaca principalmente el modelo optimizado. Esto último tiene sentido teóricamente considerando lo que se mencionaba previamente sobre la construcción de este modelo mejorado.

Con respecto a los resultados relativos a las marcas, Coolbet tiene mejores resultados que el promedio de ambas marcas. A partir de esto nace la duda del motivo de esta situación, una hipótesis es que el hecho de tener levemente más observaciones de Coolbet, genera una mejor identificación de la marca, sin embargo, es una diferencia muy leve que no debería tener gran efecto.

Un segundo motivo que podría explicar esto, es la facilidad de identificar el logotipo de Coolbet, esto debido a que es un logo central y más grande que el de

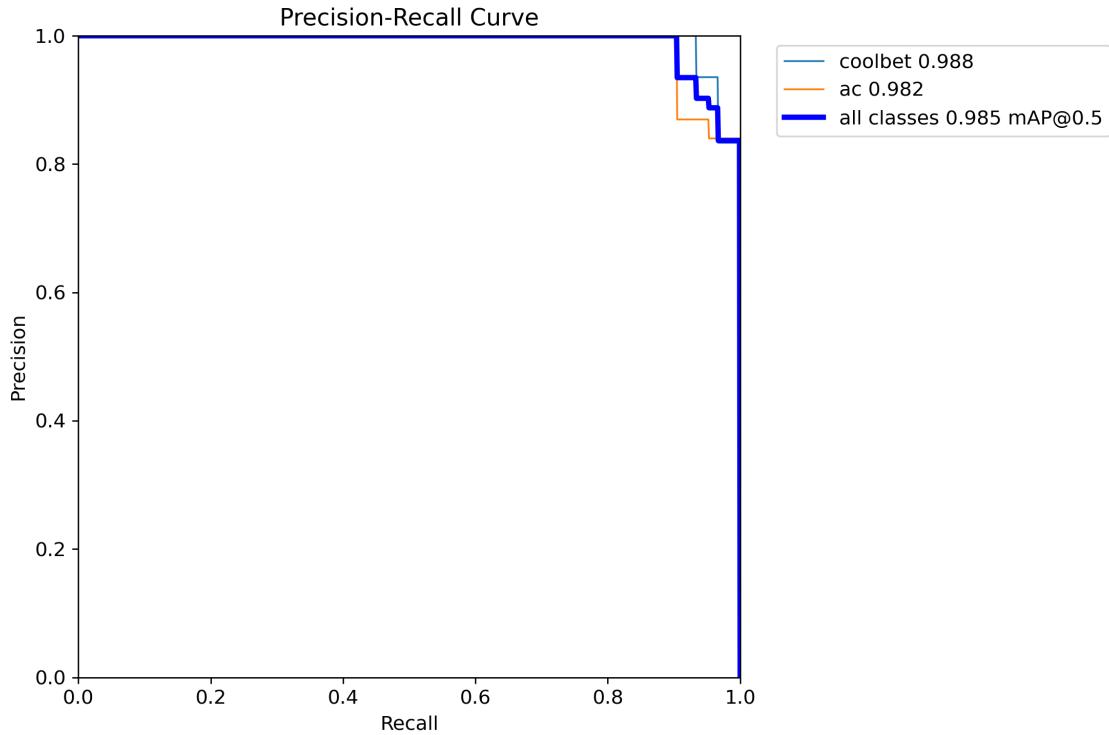


Figura 5.10: Precision-Recall Curve, Fuente: Elaboración propia

Assist card, que es un logo que suele aparecer en la parte superior de la espalda, por lo que quizá el pelo de los jugadores pueda interferir o los planos televisivos no lo muestren tan claramente como si muestran a Coolbet, que aparece en el centro de la camiseta con un tamaño mayor.

Tomando esto en cuenta, posiblemente sea uno de los grandes motivos de la mejor identificación de la primera marca mencionada por sobre la segunda, ya que la habilidad de detección de los modelos utilizados, depende de gran manera de que tan clara sea la exposición del objeto visual que se esté identificando.

5.3. Análisis de muestra

La robustez del modelo utilizado en esta investigación es un punto importante de constatar, con este objetivo en mente es que entonces se demuestra la robustez mediante los resultados de los modelos con distintas cantidades de imágenes de entrenamiento.

En este caso, se testeó la robustez con el modelo base que permite una mayor facilidad técnica para ser probada bajo distintas muestras de información. Además, considerando que los resultados del modelo base y optimizado son cercanos, sería posible asumir que los resultados de esta subsección podrían ser extensibles al modelo optimizado.

En específico, se realizaron 3 distintas especificaciones del modelo, la primera utilizando 56 imágenes de entrenamiento (Figura 5.11), la segunda con 120 imágenes

(Figura 5.12) y por último, la tercera con 200 imágenes (Figura 5.13).

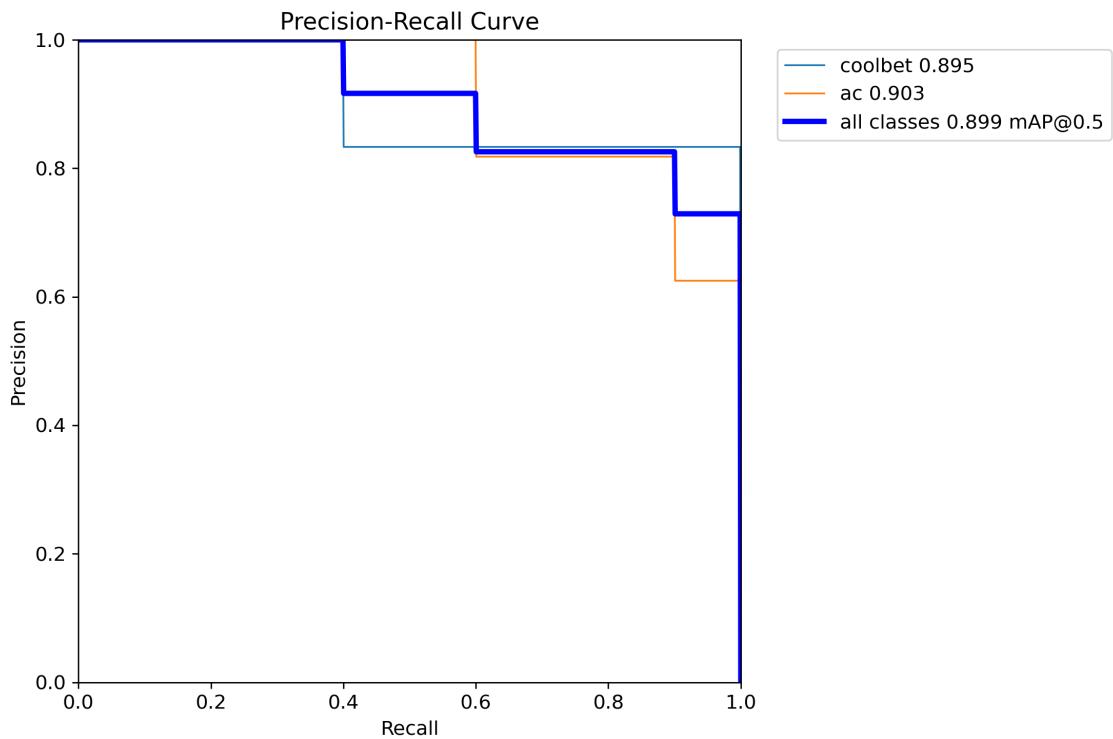


Figura 5.11: PR Curve especificación 1, Fuente: Elaboración propia

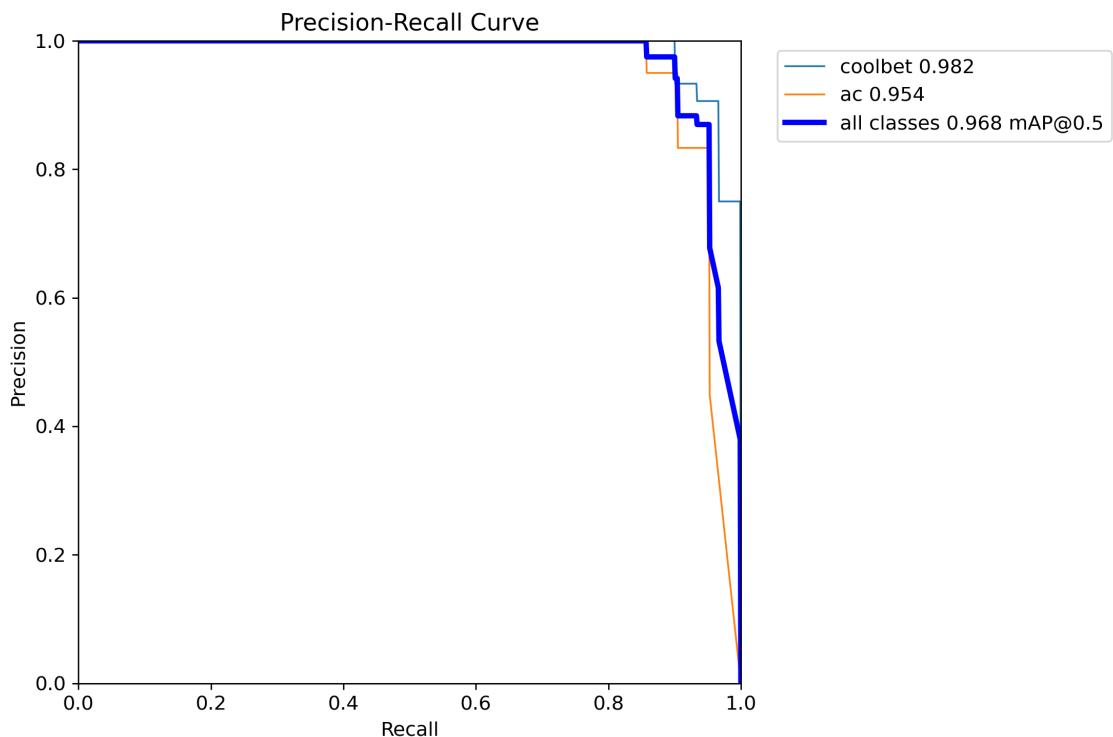


Figura 5.12: PR Curve especificación 2, Fuente: Elaboración propia

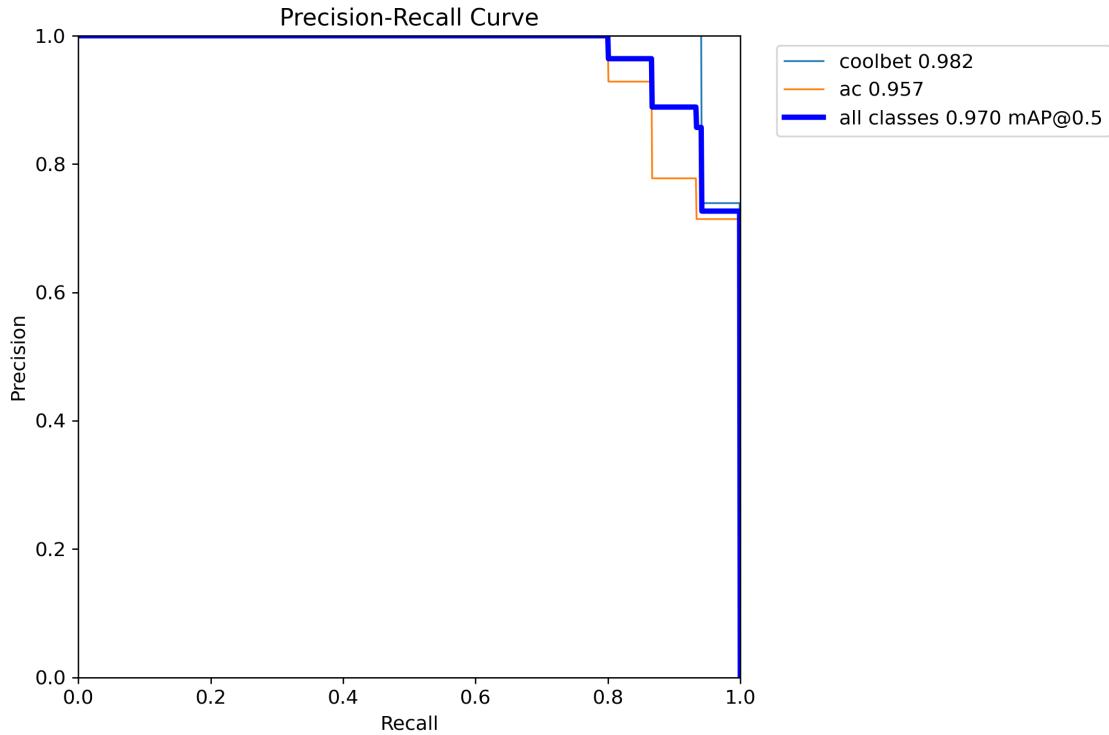


Figura 5.13: PR Curve especificación 3, Fuente: Elaboración propia

En la Tabla 5.2, se pueden ver los promedios de PR curve para las 4 distintas opciones de especificaciones, que sintetizan la información mostrada en las figuras recientemente nombradas.

Especificación	PR Curve Promedio
Especificación 1	0.899
Especificación 2	0.968
Especificación 3	0.970

Cuadro 5.2: Resultados Precision-Recall para cada modelo con diferentes muestras.

Tomando en consideración la información expuesta, es posible concluir que los modelos mejoran al aumentar la cantidad de datos que se les entregan.

El hecho de que los resultados en el modelo mejore con una mayor cantidad de imágenes disponibles, se puede entender por la disponibilidad de situaciones con las que el modelo práctica, por lo tanto, al tener mayor espacio de práctica, mejores resultados genera.

5.4. Imágenes sin marcas por detectar

Un último análisis del modelo se realizó con el fin de probar la calidad del modelo, en este caso se aplicó el modelo de detección de marcas comerciales a 30 imágenes que no contenían ninguna de las marcas a identificar, pero si contenían

otras marcas o simplemente planos que no tenían objetos que fueran relevantes para esta investigación. De estas, en 2 casos se detectó erróneamente, esto permite pensar que en ambos casos los modelos son robustos ante imágenes de fondo neutro con información irrelevante para el artículo.

5.5. Aplicación

Considerando las métricas expuestas en las subsecciones previas, es posible concluir que el modelo optimizado tiene mejores resultados que el modelo base. Es por esto, que se utiliza el modelo con mejores resultados para la aplicación sobre un caso de uso real en el fútbol.

El objetivo de plantear el uso de lo que se construyó a lo largo de esta investigación, es probar el desempeño de lo creado con datos reales no etiquetados, y principalmente, poner en exposición la utilidad de este uso de la inteligencia artificial en el ámbito comercial del fútbol.

El set de imágenes a utilizar se obtiene a través de videos de partidos que no hayan sido utilizados para entrenar ni validar el modelo, en específico se seleccionó 1 partido al azar del último año. En específico el duelo sobre el cual se evalúa el modelo es el encuentro entre Colo-Colo y O'Higgins disputado el día 3 de junio de 2024.

Algunas especificaciones relevantes en cuanto al manejo de los datos, los videos se separan en frames de 0.5 segundos cada uno para poder capturar de mejor manera los detalles. Una vez que se separan los frames, se le aplica el modelo optimizado de detección. Para este caso, una demostración de como se ve la detección de logos es lo que se muestra en la Figura 5.14 para el caso de Coolbet, y en la Figura 5.15 para el caso de Assist card.

Es importante notar que el modelo identifica los logos comerciales a detectar y le asigna una probabilidad de estar en lo correcto, esto es lo que se observa en el recuadro azul de las imágenes, donde contiene una etiqueta con la palabra “ac” en el caso de la Figura 5.15 que hace alusión a Assist Card, mientras que con el número que se nota en la imagen, se refiere a la probabilidad de seguridad de la detección para este frame en específico, en este caso un 55 %.

Basándose en los resultados, se obtienen apariciones de logos detectados de las marcas Coolbet y Assist Card utilizando el modelo con hiperparámetros optimizados, el cual, además de identificar me indica la probabilidad de seguridad en cada evento, para efectos de la aplicación que se hace en esta investigación solo se toman las predicciones con un 50 % de seguridad o más.

Los resultados obtenidos se pueden separar en distintos tramos del partido debido al output que se obtiene desde la predicción, para un análisis que tiene mayor riqueza en la estimación, se muestra en la Tabla 5.3.

Para asociarlo a unidades de medidas más intuitivas, en el caso de la marca Coolbet se tiene una media de 257 segundos en promedio por partido, esto considerando las 514 veces de aparición promedio por partido, para los 0.5 segundos por



Figura 5.14: Detección modelo optimizado para marca Coolbet, Fuente: Elaboración propia



Figura 5.15: Detección modelo optimizado, Fuente: Elaboración propia

frame que se mencionaba previamente.

Esta información puede ser utilizada con diversos fines, y cobra mayor relevancia si es que se complementa con otras informaciones relevantes.

Tomando un caso de uso de esta información, se podrían obtener conclusiones significativas, por ejemplo, el valor aproximado de inversión de Coolbet para posicionar su marca en la camiseta de Colo-Colo es de 225 millones de pesos chilenos mensuales (aproximadamente 225 mil dólares), esto es considerando que la oferta

Tramo del partido	Apariciones Coolbet	Apariciones AC
Minuto 1-15	60	36
Minuto 15-30	43	26
Minuto 30-45	69	65
Minuto 45-60	165	177
Minuto 60-75	54	42
Minuto 75-90	123	126
Total	514	472

Cuadro 5.3: Cantidad de apariciones de cada marca por partido.

de Coolbet asciende a aproximadamente \$2.700.000.000 pesos chilenos anuales, que corresponden a \$225.000.000 mensuales (Tercera (2024)).

Por otro lado, se tiene el antecedente de que Coolbet tenga una aparición promedio de 257 segundos por partido y que se juegan alrededor de 4 partidos por mes, llegando a una exposición de 1028 segundos mensuales.

Tomando en cuenta la información recopilada, esto quiere decir que la marca está invirtiendo cerca de 218.871 de pesos chilenos (aproximadamente 218 dólares mensuales) por segundo de aparición en promedio por partido.

Esto sin duda es una herramienta que podría ser clave en la evaluación de las inversiones no solo en el ámbito de la publicidad en el fútbol, sino que a nivel de todos los deportes e incluso en otros ámbitos de la vida. Lo que si se debe tener en cuenta, es que los valores invertidos por las empresas muchas veces consideran más cosas que solo la exposición durante el partido, entre ellas posición de marca, reconocimiento, etc.

Capítulo 6

Conclusión

La investigación ha demostrado que el modelo YOLOv8 es una herramienta efectiva y precisa para la detección de logotipos comerciales en eventos deportivos, específicamente en partidos de fútbol. A través del uso de YOLOv8 y su versión mejorada con hiperparámetros óptimos, se logró una alta tasa de precisión en la identificación de las marcas “Coolbet” y “Assist Card” en las camisetas del equipo Colo-Colo. La metodología implementada, que incluyó la recolección y etiquetado de imágenes, permitió entrenar y probar los modelos con una muestra representativa y equilibrada, obteniendo resultados robustos y confiables.

Los resultados indicaron que YOLOv8, tanto en su versión estándar como mejorada, ofrece una alta precisión y capacidad de detección en tiempo real, con una precisión promedio del 97 % y un F1 score significativo para ambas marcas. Además, la implementación de técnicas de mejora de hiperparámetros mostró un aumento en el rendimiento del modelo, aunque las diferencias no fueron sustanciales, sugiriendo que YOLOv8 ya posee una configuración eficiente para este tipo de tareas.

El análisis de la matriz de confusión y las curvas F1 y PR confirmó la robustez del modelo, destacando su capacidad para diferenciar correctamente entre las distintas marcas y minimizar los errores de clasificación. El hecho de que ambos modelos mejoren con una mayor cantidad de datos de entrenamiento resalta la importancia de disponer de un amplio y variado conjunto de datos para optimizar el rendimiento del sistema de detección.

Además de su aplicación en la evaluación de la exposición de marcas en eventos deportivos, la tecnología utilizada en esta investigación tiene el potencial de extenderse a otros ámbitos, como el análisis de marcas en redes sociales, monitoreo de patrocinios en diferentes deportes y estudios de mercado. La versatilidad y precisión de YOLOv8 lo convierten en una herramienta valiosa para cualquier industria donde la visibilidad de la marca es crucial.

En conclusión, el modelo YOLOv8 ha demostrado ser una solución efectiva para la detección de logotipos en transmisiones deportivas, proporcionando datos valiosos para la optimización de estrategias publicitarias y la cuantificación del retorno de inversión. Futuras investigaciones podrían enfocarse en la aplicación de este modelo

en otros deportes o eventos, así como en la integración de técnicas avanzadas de aprendizaje automático para mejorar aún más la precisión y eficiencia de la detección de logotipos.

Capítulo 7

Anexo

7.1. Non-maximum Suppression

El Non-maximum Suppression (NMS) de acuerdo a Hosang et al. (2017), es una técnica utilizada en el procesamiento de imágenes, particularmente en tareas de detección de objetos, para eliminar redundancias y mejorar la precisión de los resultados. NMS es fundamental para la detección de objetos porque, durante este proceso, es común que múltiples cuadros delimitadores (bounding boxes) se predigan para el mismo objeto.

La técnica del NMS consta de 4 etapas, la primera asigna puntajes de confianza (dados por el modelo de detección utilizado) a los distintos cuadros delimitadores predichos, esto se realiza con el fin de reflejar una probabilidad de que el cuadro contenga un objeto relevante. La segunda etapa está dada por el ordenamiento de los puntajes basándose en la probabilidad determinada. En tercer lugar, considerando el cuadro de mayor puntaje se comienzan a eliminar aquellos cuadros que tienen un menor puntaje y que se superponen con el cuadro mejor evaluado (determinación que está definida sobre la base del índice de intersección sobre la unión). Finalmente, el proceso se repite para los siguientes cuadros en base a su puntaje hasta que no queden cuadros o no exista la necesidad de eliminar cuadros por superposición.

De acuerdo con Hosang et al. (2017), al eliminar cuadros redundantes y conservar solo el cuadro delimitador más confiable para cada objeto detectado, NMS ayuda a consolidar las detecciones, reduciendo la confusión causada por múltiples detecciones del mismo objeto. El funcionamiento del NMS se puede observar en la Figura 7.1.

7.2. Función de pérdida CIoU

La función de pérdida CIoU (Complete Intersection over Union) fue introducida por Zheng et al. (2020) como una mejora significativa sobre las métricas de pérdida

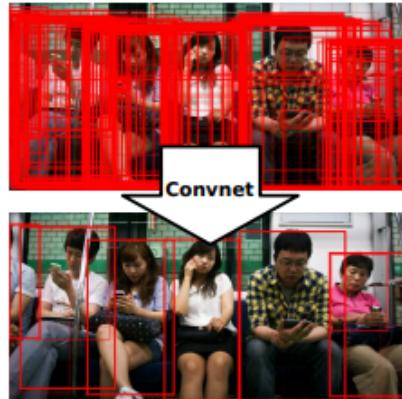


Figura 7.1: Funcionamiento NMS, Fuente: (Hosang et al. (2017)).

anteriores utilizadas en la regresión de cuadros delimitadores para la detección de objetos. Esta función de pérdida está diseñada para optimizar simultáneamente la superposición, la distancia central y la relación de aspecto de los cuadros delimitadores predichos, lo que permite una mejor convergencia y mayor precisión en la detección de objetos.

A diferencia de las pérdidas basadas únicamente en el IoU (Intersection over Union), que solo miden la superposición entre los cuadros delimitadores predichos y los cuadros delimitadores reales, la pérdida CIoU incorpora un término adicional que penaliza la distancia entre los centros de estos cuadros. Esta penalización adicional ajusta mejor la ubicación del cuadro delimitador predicho hacia el cuadro real, mejorando la precisión espacial de la detección.

Además, la función de pérdida CIoU considera la diferencia en la relación de aspecto entre el cuadro delimitador predicho y el cuadro real. Este aspecto es fundamental para ajustar correctamente el tamaño y la forma del cuadro delimitador predicho, asegurando que se alinee más adecuadamente con el objeto real que se intenta detectar. Este enfoque holístico proporciona una métrica más completa para la regresión de cuadros delimitadores.

La fórmula de la pérdida CIoU es la siguiente:

$$CIoU = IoU - \frac{\rho^2(b, b^*)}{c^2} - \alpha v \quad (7.1)$$

En esta fórmula:

- IoU es la Intersección sobre la Unión.
- $\rho^2(b, b^*)$ es la distancia euclídea al cuadrado entre los centros de los cuadros de delimitación predicho b y verdadero b^* .
- c^2 es la distancia euclídea al cuadrado entre los centros de los cuadros de delimitación más externo que envuelve a ambos cuadros.
- α es un peso que balancea la contribución del término de aspecto en el cálculo

de CIoU.

- v es un término que penaliza las diferencias de aspecto.

La pérdida CIoU presenta varias ventajas sobre las funciones de pérdida tradicionales, como la pérdida IoU y la pérdida GIoU (Generalized Intersection over Union). Al tener en cuenta múltiples factores que afectan la precisión de los cuadros delimitadores, la pérdida CIoU ayuda a que los modelos converjan más rápido durante el entrenamiento. Además, mejora la precisión final de la detección de objetos, especialmente en situaciones en las que los cuadros delimitadores deben ajustarse con mayor precisión a objetos de diferentes formas y tamaños.

7.3. Función de pérdida DFL

En el campo de la detección de objetos, la función de pérdida Differentiable Focal Loss (DFL), propuesta por Li et al. (2020), ha demostrado ser una mejora significativa sobre la Focal Loss estándar. Esta última fue desarrollada para mitigar el problema del desequilibrio de clases en tareas de detección de objetos, ajustando el peso de los ejemplos según su dificultad. A pesar de su eficacia, la Focal Loss puede enfrentar problemas de diferenciabilidad que limitan la convergencia durante el entrenamiento del modelo. Para resolver estos problemas, Li et al. introdujeron la DFL, que mejora el proceso de optimización al hacer que la función de pérdida sea más diferenciable.

La formulación de la Focal Loss se expresa como:

$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t)$$

donde:

- p_t es la probabilidad predicha del ejemplo correcto,
- α_t es un parámetro de balanceo para contrarrestar el desequilibrio de clases,
- γ es el parámetro de modulación que reduce el peso de los ejemplos bien clasificados y aumenta el peso de los ejemplos difíciles.

En la DFL, se introduce una refinada estructura de diferenciabilidad al modificar esta función para ajustarse mejor a la naturaleza continua de los gradientes. La fórmula generalizada de la DFL puede expresarse como:

$$DFL(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t) + \beta_t \cdot L_{smooth}(p_t)$$

donde:

- β_t es un factor de suavización que ajusta la magnitud de la penalización adicional,

- $L_{smooth}(p_t)$ es un término de suavización (smooth loss) que mejora la continuidad de la función y, en consecuencia, su diferenciabilidad.

Este término adicional $L_{smooth}(p_t)$ se encarga de suavizar las transiciones abruptas en la pérdida, mejorando la capacidad del modelo para aprender de manera más eficaz al proporcionar gradientes más suaves y mejor controlados. En comparación con la Focal Loss estándar, la DFL facilita una optimización más estable, especialmente en casos donde los datos presentan un fuerte desequilibrio de clases o los ejemplos difíciles son prevalentes.

El objetivo principal de la DFL es mejorar la capacidad de los modelos de detección de objetos para aprender de ejemplos difíciles y desequilibrados, proporcionando una función de pérdida que responde de manera más efectiva a los cambios en los gradientes durante el entrenamiento. Esto resulta en un mejor rendimiento general del modelo en la clasificación y detección de objetos, particularmente en escenarios desafiantes.

Bibliografía

- Alsheikhy, A., Said, Y., & Barr, M. (2020). Logo Recognition with the Use of Deep Convolutional Neural Networks. *Engineering, Technology Applied Science Research, 10*, 6191-6194. <https://doi.org/10.48084/etasr.3734>
- Bochkovskiy, A., Wang, C.-Y., & Liao, H.-y. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection.
- Bográn Ortiz, L. Y., & Martínez Hernández, J. J. (2023). Comparativa de modelos de detección de objetos y personas en espacios cerrados de acceso público. *Revista Universidad y Sociedad, 15*, 661-672. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2218-36202023000400661&nrm=iso
- Cleofas, L., Posadas-Durán, J., Martínez-Ortiz, P., Loyo-Desiderio, G., Ruvalcaba-Hernández, E., & Brito, O. (2023). Automatic Detection of Vehicular Traffic Elements based on Deep Learning for Advanced Driving Assistance Systems. *Computación y Sistemas, 27*. <https://doi.org/10.13053/cys-27-3-4508>
- He, K., Gkioxari, G., Dollár, P., & Girshick, R. B. (2017). Mask R-CNN. *CoRR, abs/1703.06870*. <http://arxiv.org/abs/1703.06870>
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 37*(9), 1904-1916. <https://doi.org/10.1109/TPAMI.2015.2389824>
- Hosang, J. H., Benenson, R., & Schiele, B. (2017). Learning non-maximum suppression. *CoRR, abs/1705.02950*. <http://arxiv.org/abs/1705.02950>
- Jocher, G., Chaurasia, A., & Qiu, J. (2023). *Ultralytics YOLOv8* (Ver. 8.0.0). <https://github.com/ultralytics/ultralytics>

- Li, X., Wang, W., Wu, L., Chen, S., Hu, X., Li, J., Tang, J., & Yang, J. (2020). Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection. *CoRR*, *abs/2006.04388*. <https://arxiv.org/abs/2006.04388>
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., & Berg, A. C. (2015). SSD: Single Shot MultiBox Detector. *CoRR*, *abs/1512.02325*. <http://arxiv.org/abs/1512.02325>
- Rangeking. (2024).
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection, 779-788. <https://doi.org/10.1109/CVPR.2016.91>
- Sahel, S., Alsahafi, M., Alghamdi, M., & Alsubait, T. (2021). Logo Detection Using Deep Learning with Pretrained CNN Models. *Engineering, Technology and Applied Science Research*, *11*(1), 6724-6729. <https://doi.org/10.48084/etasr.3919>
- Shuo, Y., Zhang, J., Bo, C., Wang, M., & Chen, L. (2018). Fast vehicle logo detection in complex scenes. *Optics Laser Technology*, *110*. <https://doi.org/10.1016/j.optlastec.2018.08.007>
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going Deeper with Convolutions. *CoRR*, *abs/1409.4842*. <http://arxiv.org/abs/1409.4842>
- Tercera, L. (2024). *El contrato récord que negocia Colo Colo para su nuevo main sponsor* (Ver. 8.0.0). <https://www.latercera.com/el-deportivo/noticia/el-contrato-record-que-negocia-colo-colo-para-su-nuevo-main-sponsor/EK47OVZVJZGXDJVTIMZD3GMHVE/>
- Terven, J., & Cordova-Esparza, D.-M. (2023). A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond.
- Wang, C.-Y., Liao, H.-y., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W., & Yeh, I.-H. (2020). CSPNet: A New Backbone that can Enhance Learning Capability of CNN, 1571-1580. <https://doi.org/10.1109/CVPRW50498.2020.00203>
- Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2020). Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. *Proceedings of*

the AAAI Conference on Artificial Intelligence, 34(07), 12993-13000. <https://doi.org/10.1609/aaai.v34i07.6999>