



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Segmentación y detección de objetos en imágenes
deportivas mediante aprendizaje profundo

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Blasco Abril, Félix

Tutor/a: Juan Císcar, Alfonso

CURSO ACADÉMICO: 2024/2025



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Escola Tècnica
Superior d'Enginyeria
Informàtica

Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Valencia

Segmentación y detección de objetos en imágenes deportivas mediante aprendizaje profundo

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Felix Blasco Abril

Tutor: Alfons Juan Ciscar

Curso 2024-2025

Resum

L'automatització de la segmentació visual en imatges esportives està prenent cada vegada més rellevància en el món de l'aprenentatge profund. Aquest enfocament naix de la necessitat d'analitzar de manera efectiva com es desenvolupen els atletes en diferents disciplines esportives. Aquests mètodes permeten la identificació precisa i el seguiment automàtic d'elements clau com els jugadors i el baló en un entorn esportiu específic, per a facilitar una anàlisi objectiva i automatitzada d'esdeveniments esportius. En aquest context d'estudi es proposa realitzar una anàlisi detallada i comparativa de diversos enfocaments avançats de segmentació visual basats en models d'aprenentatge profund. L'objectiu és examinar propostes pròximes a l'estat actual més avançat en aquest camp per a seleccionar un model efectiu i precís que pugui adaptar-se sense complicacions a diverses situacions esportives. Finalment, es planteja posar a prova de manera experimental la tècnica triada mitjançant la seua aplicació en imatges esportives reals, per a avaluar el rendiment de forma automàtica i generalitzada; la qual cosa donaria lloc a eines pràctiques per a l'anàlisi automatitzada en aquest àmbit.

Paraules clau: Aprenentatge automàtic; aprenentatge profund; segmentació d'imatges; imatges esportives

Resumen

La automatización de la segmentación visual en imágenes deportivas está tomando cada vez más relevancia en el mundo del aprendizaje profundo. Este enfoque nace de la necesidad de analizar de forma efectiva cómo se desempeñan los atletas en diferentes disciplinas deportivas. Estos métodos permiten la identificación precisa y el seguimiento automático de elementos clave como jugadores y balones en un entorno deportivo específico para facilitar un análisis objetivo y automatizado de eventos deportivos. En este marco de investigación se propone realizar un análisis detallado y comparativo de distintos enfoques avanzados de segmentación visual basados en modelos de aprendizaje profundo. El objetivo es examinar propuestas próximas al estado del arte para seleccionar un modelo efectivo y preciso que pueda adaptarse sin complicaciones a diversas situaciones deportivas. Como paso final, se plantea someter a prueba experimental la técnica seleccionada aplicándola a imágenes deportivas reales, con el fin de evaluar su rendimiento de manera automática y generalizable, lo que permitiría desarrollar herramientas prácticas para el análisis automatizado en este ámbito.

Palabras clave: Aprendizaje automático; aprendizaje profundo; segmentación de imágenes; imágenes deportivas

Abstract

Automating visual segmentation in sports images is becoming increasingly relevant within deep learning. This approach arises from the need to effectively analyze how athletes perform across different sports disciplines. These methods enable precise identification and automatic tracking of key elements such as players and the ball within a specific sports environment, facilitating an objective and automated analysis of sporting events. In this study context, a detailed, comparative analysis of advanced visual segmentation approaches based on deep-learning models is proposed. The goal is to examine methods close to the current state of the art in order to select an effective and accurate model that can be readily adapted to diverse sports scenarios. Finally, we propose experimentally

testing the chosen technique on real sports images to evaluate performance in an automatic and generalizable way, resulting in practical tools for automated analysis in this field.

Key words: Machine learning; deep learning; image segmentation; sports images

Índice general

Índice general	V
Índice de figuras	IX
Índice de tablas	X
1 Introducción	1
1.1 Motivación	1
1.2 Objetivos	1
1.3 Alcance y aplicabilidad	2
1.4 Estructura de la memoria	2
2 Preliminares	5
2.1 Visión por computador	5
2.1.1 Introducción histórica	5
2.1.2 Aspectos técnicos	6
2.1.3 Aplicaciones prácticas	6
2.2 Aprendizaje profundo	8
2.2.1 Historia del aprendizaje profundo	8
2.2.2 Principales arquitecturas	9
2.2.3 Aplicaciones generales y en visión por computador	9
2.2.4 Aplicaciones en el ámbito deportivo	10
2.3 Aprendizaje automático	10
2.3.1 Fundamentos técnicos y algoritmos clásicos	11
2.3.2 Aplicaciones generales del aprendizaje automático	12
2.3.3 Aplicaciones en el deporte	13
3 Datasets	17
3.1 Introducción	17
3.2 Soccer-Panoptic (Kaggle)	17
3.2.1 Motivación de la elección	17
3.2.2 Descripción y composición	18
3.2.3 Estructura y preparación para SAM	18
3.2.4 Particiones y <i>batches</i> de evaluación	18
3.2.5 Esquema de clases y uso en evaluación	19
3.2.6 Ejemplos visuales	19
3.2.7 Buenas prácticas y control de calidad	19
3.2.8 Limitaciones y sesgos	20
3.3 Vinalesa FC (dataset propio)	22
3.3.1 Descripción y propósito	22
3.3.2 Proceso de creación	22
3.3.3 Versionado y <i>splits</i>	22
3.3.4 Ejemplos	22
3.3.5 Notas prácticas	23
3.4 Conclusiones del capítulo	24
4 SAM	25

4.1	Introducción	25
4.2	SAM	25
4.3	Experimento 1 – Evaluación base de SAM en fútbol	25
4.3.1	Introducción	25
4.3.2	Tecnologías empleadas	26
4.3.3	Diseño experimental y casos de prompting	26
4.3.4	Métrica	26
4.3.5	Resultados cuantitativos	27
4.3.6	Análisis cualitativo (Batch 1)	28
4.3.7	Análisis cualitativo (Batch 2)	29
4.3.8	Análisis cualitativo (Batch 3: caso difícil)	30
4.3.9	Análisis y discusión	31
4.3.10	Código ilustrativo (implementación)	31
4.3.11	Limitaciones y trabajo futuro	32
4.3.12	Resumen	32
4.4	Experimento 2 – Grid/AMG vs BBox en fútbol	33
4.4.1	Introducción	33
4.4.2	Metodología	33
4.4.3	Resultados	34
4.4.4	Análisis cualitativo (Batch 1)	37
4.4.5	Análisis cualitativo (Batch 2)	38
4.4.6	Análisis cualitativo (Batch 3: caso difícil)	39
4.4.7	Código ilustrativo (implementación)	39
4.4.8	Discusión	41
4.4.9	Limitaciones y trabajo futuro	41
4.4.10	Conclusiones	41
4.5	Conclusiones — SAM (Experimentos 1 y 2)	41
5	Fine-tuning de SAM	43
5.1	Introducción	43
5.2	Fine-tuning	43
5.3	Experimentos	44
5.4	Conclusiones — Fine-tuning de SAM	48
6	Conclusiones	49
6.1	Resumen del trabajo realizado	49
6.2	Objetivos conseguidos	50
6.3	Trabajo futuro	51
	Bibliografía	53
A	Anexo de Listings	57
A.1	Estructura práctica del dataset	57
A.2	Derivación de máscaras y <i>prompts</i> desde panoptic	57
A.3	Funciones auxiliares de evaluación y visualización	57
A.4	Comparación justa de métodos (BBox)	58
A.5	Orquestación por lotes	58
A.6	Descubrimiento y partición por batch	59
A.7	Fine-tuning decoder-only	59
A.8	Comparación de métodos con distintos BBoxes	59
A.9	Collate para DataLoader	60
A.10	Overlay cualitativo	60
B	Configuración del sistema	61
B.1	Entorno hardware y software	61
B.2	Fase de inicialización	62

B.3	Identificación de dispositivos	63
B.4	Notas y limitaciones	64

Índice de figuras

2.1	Aplicaciones de visión por computador en deportes: seguimiento de jugadores y eventos.	14
3.1	Vista general de Soccer-Panoptic (frame y máscara panóptica).	19
3.2	Derivación de máscaras por clase desde la anotación panóptica.	20
3.3	Overlay GT (verde/rojo) vs. predicción SAM (azul/amarillo) en un frame de ejemplo.	20
3.4	Mosaico de ejemplos (Vinalesa FC).	23
3.5	Overlay GT (verde/rojo) vs. predicción de SAM (azul/amarillo) en imágenes de Vinalesa FC.	24
4.1	mIoU por lote (Jugadores, ViT-B) para los tres enfoques: <i>Bbox específicos</i> , <i>Bbox global style</i> y <i>Imagen completa</i>	27
4.2	mIoU por lote (Balón, ViT-B). En <i>Batch 1</i> no hay balón anotado.	27
4.3	Promedio final mIoU (Jugadores, ViT-B) por enfoque.	28
4.4	Promedio final mIoU (Balón, ViT-B) por enfoque.	28
4.5	Experimento 1 — Ejemplo cualitativo (Batch 1). GT en verde/rojo; SAM en azul/amarillo, superpuesto a la GT.	29
4.6	Experimento 1 — Ejemplo cualitativo (Batch 2). GT: verde/rojo; SAM: azul/amarillo, superpuesto a la GT.	30
4.7	Experimento 1 — Ejemplo cualitativo (Batch 3). Escena difícil con múltiples instancias y líneas de campo que inducen errores; GT en verde/rojo y SAM en azul/amarillo.	30
4.8	mIoU por lote (Jugadores) – Grid/AMG vs BBox (ViT-B).	34
4.9	mIoU por lote (Balón) – Grid/AMG vs BBox (ViT-B). En <i>Batch 1</i> no hay balón anotado.	35
4.10	Promedio final mIoU (Jugadores) por método (ViT-B).	35
4.11	Promedio final mIoU (Balón) por método (ViT-B).	36
4.12	Experimento 2 — Ejemplo cualitativo (Batch 1). GT en verde/rojo; SAM en azul/amarillo, superpuesto a la GT.	37
4.13	Experimento 2 — Ejemplo cualitativo (Batch 2). GT: verde/rojo; SAM: azul/amarillo, superpuesto a la GT.	38
4.14	Experimento 2 — Ejemplo cualitativo (Batch 3). Ningún método capta bien el frame: GT en verde/rojo y SAM en azul/amarillo.	39
5.1	Split 80-20 — IoU y Dice por categoría (Global, Balón, Jugador).	45
5.2	Split 80-20 — Precisión y Recall por categoría.	45
5.3	Fine-tuning SAM (ViT-B) — mIoU por lote en Soccer Dataset.	46
5.4	Comparación por lote: SAM base (bbox específicos) vs SAM <i>fine-tuned</i>	46

Índice de tablas

3.1	Soccer-Panoptic: resumen operativo para esta memoria (clases relevantes, formato, uso).	18
3.2	Clases del dataset y papel en esta memoria.	19
3.3	Resumen de Vinalesa FC.	22
4.1	Promedio final mIoU (ViT-B) por enfoque y clase.	27
4.2	Resumen cualitativo por enfoque (ViT-B).	33
4.3	Promedio final mIoU (Jugadores/Balón) por método (ViT-B, 720 frames). .	34
5.1	Evolución de pérdidas (entrenamiento/validación) en Vinalesa v1 (13) y v2 (27).	44
5.2	Métricas en test (split 80/20).	44
5.3	Rendimiento por lote tras fine-tuning (mIoU en test).	45

CAPÍTULO 1

Introducción

1.1 Motivación

Este trabajo surge de mi experiencia personal con el fútbol, tanto como jugador desde niño como ahora como entrenador. Esta doble perspectiva me motiva a buscar métodos para identificar de forma más rápida problemas tácticos y técnicos en mis equipos, sin depender tanto de procesos manuales y subjetivos.

Por otro lado, surge un interés social: si estas soluciones resultan accesibles, podrían simplificar el análisis en clubes de menor envergadura, potenciar la formación y contribuir a identificar riesgos para fomentar una práctica deportiva más segura, como la prevención de sobrecargas. Desde el punto de vista de mi desarrollo profesional, quiero probar una tecnología reciente para observar su funcionamiento: entender sus aciertos, errores y las razones detrás de ellos. Busco aprender a evaluarla con criterio y comunicar sus limitaciones con transparencia.

En resumen, este trabajo aspira a integrar la experiencia práctica con la innovación tecnológica, aprovechando las ventajas actuales y destacando, desde la visión del terreno de juego, los aspectos que requieren aún mayor desarrollo. Partiendo de esta motivación, procedemos a definir los objetivos.

1.2 Objetivos

El propósito principal de este estudio es analizar una tecnología de segmentación avanzada y aplicarla al fútbol, de forma clara, repetible y práctica. No solo me enfoco en las estadísticas, sino que busco comprender su funcionamiento en partidos, sus requisitos para un buen desempeño y sus limitaciones en situaciones complejas.

Inicialmente, me concentro en los conjuntos de datos. Elijo y preparo un conjunto público con anotaciones apropiadas para obtener indicaciones comparables y, al mismo tiempo, creo mi propio conjunto (Vinalesa FC) con imágenes reales y segmentaciones manuales. El ideal es tener datos que permitan pruebas equitativas entre métodos, con y sin ayuda, y que reflejen las expectativas de un cuerpo técnico.

En segundo lugar, evalúo SAM sin ejemplos previos, usando tres estrategias basadas en recuadros (imagen completa, recuadro general por clase y recuadro específico por instancia) y una opción sin ayuda basada en rejilla/AMG. Mantengo el mismo criterio de evaluación (mIoU por clase y por lote) y un enfoque justo: comparar principalmente los métodos sin ayuda y usar las variantes asistidas como referencia máxima. El fin es establecer puntos de partida claros, con análisis cuantitativos y cualitativos que expli-

quen los éxitos y errores comunes, especialmente en el balón, debido a su tamaño y a las obstrucciones.

El tercer objetivo clave es ajustar SAM para un campo específico. Se explora un ajuste supervisado del decodificador de máscaras con pocos datos, controlando la separación de la validación y las limitaciones de cálculo. Se busca mostrar cuándo el ajuste mejora los resultados en comparación con los métodos básicos sin ayuda y registrar las concesiones requeridas (tiempo, equipo, estabilidad), para que el resultado se pueda aplicar y entender bien.

En conjunto, este trabajo busca una evaluación actual y práctica: datos correctos, pruebas equilibradas y un ajuste responsable que ayude a saber cuál es el límite actual de esta tecnología y qué pasos serían necesarios para usarla en los análisis reales de juegos. A continuación, delimitaremos el alcance y su aplicabilidad.

1.3 Alcance y aplicabilidad

Este proyecto se plantea como **herramienta de aprendizaje y prototipo funcional**, no como producto final. El público objetivo son **cuerpos técnicos y academias/clubes de fútbol base** que deseen explorar segmentación y análisis visual con recursos limitados. La aplicabilidad inmediata es: (i) *estudio y formación* (comprender limitaciones/fortalezas de SAM en fútbol) y (ii) *prototipado* de flujos de anotación y evaluación sobre conjuntos propios. El caso natural de prueba es un club local (p. ej., Vinales FC) como entorno para un piloto controlado sin compromiso de producción.

1.4 Estructura de la memoria

La estructura de este documento comprende seis capítulos principales, complementados por la bibliografía y los apéndices.

El Capítulo 1 (Introducción) explica por qué se hizo este trabajo, cuál era el objetivo y qué temas abarca. Termina explicando cómo está organizado el documento.

El Capítulo 2 (Preliminares) aborda las ideas y definiciones que se emplearán posteriormente. Engloba conceptos fundamentales de visión artificial, aprendizaje profundo y aprendizaje automático, junto con la terminología que se utilizará a lo largo del documento.

El Capítulo 3 (Datasets) cuenta cómo se usaron los datos y cómo se prepararon para las pruebas. Explica en detalle Soccer-Panoptic (cómo está hecho, qué anotaciones tiene y cómo está organizado) y el conjunto de datos propio Vinales FC (cómo se creó y anotó), con ejemplos y detalles prácticos.

El Capítulo 4 (SAM) presenta el modelo y los dos primeros experimentos. El Experimento 1 establece una prueba inicial en fútbol usando indicaciones de cajas (imagen completa, caja global por clase y caja específica por instancia). El Experimento 2 usa Grid/AMG como una opción no asistida y la compara con la imagen completa, manteniendo la misma forma de comparar. En ambos casos, se muestran números y análisis de calidad, junto con ejemplos de código.

El Capítulo 5 (Fine-tuning de SAM) examina cómo ajustar el mask decoder con datos propios, usando la misma forma de evaluar y comparando con las opciones no asistidas. Se explica la configuración, los resultados y los problemas.

El Capítulo 6 (Conclusiones) resume los resultados principales, revisa si se cumplieron los objetivos y sugiere posibles trabajos futuros.

Finalmente, la Bibliografía recopila todas las fuentes referenciadas en el documento, mientras que los Apéndices ofrecen material complementario para facilitar la replicación del trabajo (entorno, parámetros y herramientas), además de resultados adicionales y contenidos suplementarios.

CAPÍTULO 2

Preliminares

2.1 Visión por computador

2.1.1. Introducción histórica

La visión por computador, también denominada visión artificial, estudia métodos para extraer información de imágenes y vídeo y tomar decisiones a partir de ella. En esencia, se trata de dotar a sistemas automáticos de la capacidad de analizar entradas visuales de forma estructurada, con aplicaciones en múltiples dominios donde se requiere interpretar el entorno.

El inicio de la visión por computador se sitúa en los años 50, con el desarrollo de las primeras computadoras. Aunque estas máquinas eran rudimentarias y los esfuerzos iniciales para procesar imágenes resultaron limitados por restricciones técnicas, ya se gestaban las primeras nociones de conectar la visión artificial con la inteligencia artificial y la percepción humana.

Desde esos días, la visión por computador ha recorrido un largo camino y ha ido avanzando a pasos agigantados. A lo largo de las décadas, se han alcanzado varios hitos importantes que han permitido que esta disciplina sea lo que conocemos hoy en día.

Línea del tiempo de la visión por computador:

- **Década de 1950:** Surge el primer interés en usar computadoras para procesar imágenes. Se introduce la idea de convertir señales de luz en datos que las computadoras pueden analizar.
- **Años 1960:** Se mejora la capacidad de procesamiento y las computadoras comienzan a analizar imágenes digitales y describir objetos de manera matemática. También se desarrollan los primeros filtros para detectar bordes, como los de Roberts, Sobel y Prewitt.
- **Años 1970:** Con circuitos más potentes, se realizan más investigaciones. Se introduce la transformada de Hough para detectar formas, lo que abre el camino para el reconocimiento de estos patrones.
- **Años 1980:** Se establece un proceso organizado de flujo, creando algoritmos que ayudan a filtrar, segmentar y extraer características de las imágenes.
- **Años 1990:** Comienzan a aparecer aplicaciones prácticas, como el OCR (reconocimiento óptico de caracteres), y hay mejoras en la calibración de cámaras y reconstrucción 3D.

- **Finales del siglo XX y siglo XXI:** Se siguen desarrollando descriptores como SIFT y SURF, mejorando el emparejamiento de puntos clave. A medida que las GPU y las grandes bases de datos se vuelven más comunes, la visión por computador continúa utilizando modelos matemáticos tradicionales.

2.1.2. Aspectos técnicos

La visión por computador no solo permite que las máquinas vean imágenes, sino que también les ayuda a comprender lo que hay en ellas. Generalmente, un sistema típico de visión por computador funciona en varias etapas, cada una crucial para extraer y analizar la información visual.

Adquisición de imágenes

Existen diferentes tipos de cámaras, como las RGB, que capturan imágenes en color, y cámaras multiespectrales, que recogen datos en varias bandas del espectro. También están los sensores de profundidad y escáneres LIDAR, que ayudan a medir distancias.

Preprocesamiento

Se aplican filtros y correcciones geométricas para preparar las imágenes. Esto incluye reducir el ruido e igualar el contraste.

Segmentación

Separa la imagen en diferentes partes usando métodos que se basan en colores, texturas o límites para identificar objetos.

Extracción de características

Métodos como el de Harris o SIFT ayudan a identificar esquinas y cambios en la intensidad de las imágenes.

Emparejamiento y reconstrucción 3D

Se utilizan técnicas como la visión estéreo para calcular el espacio tridimensional, lo que permite crear modelos espaciales del entorno.

Interpretación y reconocimiento

Se pueden utilizar modelos geométricos y heurísticas para clasificar objetos, además de sistemas de análisis basados en plantillas.

2.1.3. Aplicaciones prácticas

La visión por computador se ha convertido en una herramienta clave en muchos sectores, ayudando a automatizar procesos y optimizar recursos. Algunas de las aplicaciones más relevantes en varios campos incluyen:

Industria

En el sector industrial, la visión por computador se utiliza para:

- **Inspección de calidad:** Detectar automáticamente defectos en productos, como grietas o deformaciones, asegurando que se mantengan estándares de calidad consistentes.
- **Automatización de procesos:** Integrarse en las líneas de producción para tareas como el ensamblaje robótico y la clasificación de productos.
- **Lectura y verificación de códigos:** Reconocer códigos de barras, QR y caracteres impresos para facilitar la trazabilidad y el control de inventario.
- **Medición de precisión:** Evaluar las dimensiones de los componentes para garantizar que cumplan con las especificaciones técnicas.

Medicina

En el ámbito médico, la visión por computador contribuye a:

- **Diagnóstico por imágenes:** Analizar radiografías, resonancias magnéticas y tomografías para identificar anomalías como tumores o fracturas.
- **Seguimiento de pacientes:** Monitorear signos vitales y movimientos para evaluar la evolución de enfermedades o la efectividad de tratamientos.
- **Asistencia en cirugías:** Proporcionar información visual en tiempo real durante procedimientos quirúrgicos, lo que mejora la precisión y la seguridad.

Automoción y transporte

La industria automotriz y de transporte aplica la visión por computador en:

- **Sistemas ADAS:** Se están implementando funciones como el frenado automático, el mantenimiento de carril y la detección de peatones, todo con el objetivo de hacer nuestras carreteras más seguras.
- **Vehículos autónomos:** Estos vehículos utilizan cámaras y sensores para entender su entorno, permitiendo que naveguen y tomen decisiones sin necesidad de intervención humana.
- **Gestión del tráfico:** Gracias a cámaras inteligentes, se monitorean los flujos de tráfico y se detectan infracciones, ayudando a mantener el orden en las calles.

Seguridad

En el campo de la seguridad, la visión por computador se emplea para:

- **Videovigilancia inteligente:** Se utilizan para detectar comportamientos inusuales o situaciones de riesgo en tiempo real, lo que mejora la seguridad pública.
- **Reconocimiento facial:** Esta tecnología permite identificar personas para controlar accesos o localizar individuos en espacios públicos.
- **Análisis forense:** Se revisan y analizan grabaciones para ayudar en investigaciones criminales o en la reconstrucción de eventos.

Agricultura

En la agricultura, la visión por computador facilita:

- **Monitoreo de cultivos:** Permite evaluar la salud de las plantas a través de imágenes tomadas por drones o satélites.
- **Detección de plagas y enfermedades:** Ayuda a identificar problemas fitosanitarios de manera temprana, lo que permite una intervención rápida.
- **Automatización de cosechas:** Utiliza robots con visión para recoger selectivamente los frutos maduros.

Ciencia y medioambiente

En investigaciones científicas y medioambientales, la visión por computador se aplica en:

- **Estudios astronómicos:** Se encarga del análisis de imágenes del espacio para detectar cuerpos celestes o fenómenos astronómicos.
- **Monitoreo ambiental:** Permite seguir los cambios en ecosistemas, la deforestación o la contaminación mediante imágenes satelitales.
- **Investigación animal:** Facilita la observación y el análisis del comportamiento de especies en su hábitat natural a través de grabaciones de videos.

Estas aplicaciones demuestran la versatilidad y el impacto significativo de la visión por computador en diversos sectores, contribuyendo al avance tecnológico y a la mejora de procesos en múltiples disciplinas.

En todos los casos, la conjunción entre sensores avanzados (RGB, infrarrojos, LIDAR), software optimizado y hardware especializado (como GPUs o tarjetas de visión) ha permitido abordar tareas complejas que van desde la navegación autónoma hasta la inspección industrial a alta velocidad. La visión por computador continúa siendo una tecnología clave en el desarrollo de sistemas inteligentes en diversos entornos.

2.2 Aprendizaje profundo

2.2.1. Historia del aprendizaje profundo

Para entender el aprendizaje profundo, hay que retroceder un poco en el tiempo. Todo comenzó con algunas ideas sobre redes neuronales artificiales. En 1943, dos investigadores, McCulloch y Pitts, propusieron un modelo bastante básico que mostraba cómo funcionaba una neurona. Luego, en 1958, Frank Rosenblatt creó el perceptrón, una red neuronal simple que podía aprender algunas reglas simples.

No obstante, la situación se tornó más compleja en 1969. Minsky y Papert evidenciaron las limitaciones del perceptrón, particularmente en la resolución de problemas más intrincados, como el XOR. Este hallazgo provocó un declive temporal en el interés por las redes neuronales. Sin embargo, en la década de 1980 se produjo un resurgimiento, gracias al descubrimiento del algoritmo de retropropagación por parte de Rumelhart, Hinton y Williams. Esta técnica permitió entrenar redes más sofisticadas con múltiples capas.

En 1989, un investigador llamado LeCun llevó las cosas un paso más allá al aplicar este algoritmo en redes convolucionales, dándole vida a la LeNet, que se usaba para reconocer dígitos manuscritos. Luego, en 1997, Hochreiter y Schmidhuber presentaron las famosas LSTM, que son una clase de redes recurrentes capaces de aprender de forma más efectiva al recordar información a largo plazo.

La verdadera revolución del aprendizaje profundo comenzó alrededor de 2010, gracias a un aumento en la potencia de cómputo disponible y el acceso a grandes volúmenes de datos. En 2012, un modelo conocido como AlexNet, creado por Krizhevsky y su equipo, ganó un concurso llamado ImageNet, lo que marcó un gran avance en el uso de redes profundas en el campo de la visión por computador [23].

2.2.2. Principales arquitecturas

Cuando hablamos de las arquitecturas de aprendizaje profundo, hay varias que son clave.

Las Multilayer Perceptrons (MLP) son redes neuronales formadas por capas que están completamente conectadas entre sí. Son muy flexibles, pero no son la mejor opción para manejar datos que tienen una estructura espacial, como las imágenes, ya que necesitan una gran cantidad de parámetros para funcionar.

Entre las arquitecturas destacadas, las Redes Convolucionales (CNN) representan una evolución significativa. Gracias a técnicas como las convoluciones locales y el pooling, estas redes pueden capturar patrones visuales de forma jerárquica. Esta capacidad las posiciona como herramienta predilecta para el procesamiento de imágenes. Generalmente, una CNN incorpora capas de convolución, seguidas de activaciones ReLU y max-pooling, culminando en capas densas para la toma de decisiones.

Asimismo, las Redes Recurrentes (RNN) se han desarrollado para manejar secuencias temporales. Estas estructuras permiten retener información previa. No obstante, las RNN enfrentan un desafío conocido como el desvanecimiento del gradiente. En este punto, las LSTM emergen como solución. Estas variantes logran superar dicho obstáculo mediante mecanismos de puertas, lo que las hace especialmente valiosas en tareas de procesamiento del lenguaje.

Los Transformers son otra gran innovación que apareció en 2017 gracias a Vaswani y su equipo [24]. Estos modelos utilizan una técnica de autoatención que les permite captar relaciones entre diferentes partes de la información en su entrada. Han demostrado tener un rendimiento superior sobre las RNN en muchas tareas secuenciales, y también han encontrado su camino en el ámbito de la visión gracias a los Vision Transformers (ViT) [25], que son especialmente buenos cuando se trata de manejar grandes cantidades de datos. Sobre estas bases arquitectónicas, exploramos aplicaciones paradigmáticas.

2.2.3. Aplicaciones generales y en visión por computador

El aprendizaje profundo ha hecho una gran diferencia en muchas áreas. Por ejemplo, en el procesamiento de lenguaje natural, modelos como BERT y GPT son ahora los reyes en tareas como la traducción, el análisis de sentimientos y la creación de texto.

En el campo de la medicina, las CNN son utilizadas para segmentar órganos en imágenes médicas, detectar tumores y también para ayudar en diagnósticos automáticos. Esto, sin duda, está transformando la forma en que se aborda la salud.

En el mundo automotriz, las tecnologías de aprendizaje profundo se están usando en coches autónomos para detectar peatones, señales de tráfico y para segmentar la carretera en tiempo real, lo que es esencial para la seguridad y la navegación.

En el ámbito de la visión por computador, las aplicaciones incluyen todo, desde clasificar imágenes, hasta detectar objetos utilizando modelos como YOLO, Faster R-CNN, Mask R-CNN y DETR [26, 27, 36, 37]. También se utiliza para la segmentación semántica [28, 29, 30] y el seguimiento en videos, lo que permite realizar análisis más elaborados. A partir de estos fundamentos en aprendizaje profundo, concluimos el marco general y avanzamos hacia los principios y algoritmos tradicionales de aprendizaje automático, que servirán como punto de referencia.

2.2.4. Aplicaciones en el ámbito deportivo

El aprendizaje profundo también está cambiando las reglas del juego en el mundo del deporte, ayudando a automatizar el análisis visual.

Por ejemplo, hay algoritmos como Deep SORT que pueden detectar y seguir a los jugadores en el campo [31], permitiendo analizar sus movimientos y rendimiento físico. Esto es útil para los entrenadores que quieren optimizar el entrenamiento de sus jugadores.

También se usan técnicas de segmentación para dividir el campo en diferentes áreas como césped, líneas y porterías. Esto hace que sea más fácil interpretar tácticamente el juego y mejora la capacidad de la visión por computador para entender el contexto.

Otra aplicación interesante es en la detección de eventos. Hay sistemas automáticos que pueden identificar goles, saques de esquina o faltas simplemente analizando las secuencias de video. Esto permite catalogar eventos clave sin que alguien tenga que hacerlo manualmente, lo que ahorra mucho tiempo y esfuerzo.

De igual manera, el análisis táctico ha experimentado un notable progreso. Permite discernir formaciones y patrones de juego, facilitando el análisis estratégico mediante datos más objetivos.

Todo esto se combina con flujos de video en tiempo real, ofreciendo métricas precisas que complementan lo que los humanos pueden observar. Equipos deportivos profesionales están aprovechando esto tanto en sus entrenamientos como en las retransmisiones de partidos, buscando siempre mejorar el rendimiento de sus jugadores y prevenir lesiones.

2.3 Aprendizaje automático

El aprendizaje automático es un subcampo de la inteligencia artificial que estudia cómo desarrollar modelos que permitan a las computadoras aprender de los datos sin programación explícita. En términos de definición clásica, Arthur Samuel (IBM, años 1950) lo describió como “el campo de estudio que da a las computadoras la habilidad de aprender sin ser programadas explícitamente”.

Desde sus inicios en la década de 1950, el aprendizaje automático ha evolucionado notablemente, impulsado por innovaciones en algoritmos, disponibilidad de datos y potencia de cómputo:

- 1943: McCulloch y Pitts proponen el primer modelo matemático de una neurona artificial.
- 1950: Alan Turing plantea el Test de Turing.

- 1952: Arthur Samuel desarrolla un programa que aprende a jugar a las damas mediante aprendizaje por refuerzo, acuñando el término “aprendizaje automático”.
- 1956: Taller de Dartmouth, donde John McCarthy y otros definen formalmente los campos de investigación en IA y ML.
- 1986: Resurge el interés en redes neuronales tras la difusión del algoritmo de retro-propagación.
- Década de 1990: Surgen Support Vector Machines (Vapnik) y algoritmos de kernel.
- 2012: AlexNet revoluciona el reconocimiento de imágenes, marcando el inicio del aprendizaje profundo a gran escala.
- 2016: AlphaGo vence al campeón mundial de Go, demostrando el poder del aprendizaje automático en tareas complejas.
- 2020 en adelante: El ML se vuelve ubicuo; el 67 % de las empresas lo usan, y el 97 % lo usa o planea usar próximamente.

A lo largo de esta trayectoria, el campo se ha ampliado a tareas como clasificación, regresión, agrupamiento, visión por computador, y procesamiento de lenguaje natural, convirtiéndose en una herramienta esencial para transformar datos en conocimiento. Con este repaso, presentamos los fundamentos técnicos del área.

2.3.1. Fundamentos técnicos y algoritmos clásicos

Antes de adentrarse en algoritmos específicos, es fundamental comprender que el aprendizaje automático se basa en la construcción de modelos matemáticos capaces de identificar patrones en los datos y hacer predicciones o tomar decisiones sin intervención explícita. Estos modelos aprenden a partir de ejemplos previos, y su rendimiento depende tanto de la calidad de los datos como del algoritmo utilizado.

Los algoritmos clásicos del aprendizaje automático forman la base sobre la cual se han desarrollado técnicas más avanzadas. Aunque algunos han sido superados en rendimiento por modelos más complejos como las redes neuronales profundas, su simplicidad, interpretabilidad y eficiencia computacional los hacen todavía muy relevantes, especialmente en problemas donde los datos son limitados o donde se requiere una alta capacidad de explicación del modelo.

A continuación, se presentan algunos de los algoritmos más emblemáticos:

- **Modelos lineales: Regresión lineal y logística.**

La regresión lineal, por su parte, modela la relación entre una variable dependiente continua y una o más variables independientes mediante una combinación lineal. Resulta efectiva en tareas predictivas donde se presume una relación aproximadamente lineal entre las variables. Entre sus virtudes principales figuran su bajo costo computacional, su interpretabilidad y su sencillez de entrenamiento. Sin embargo, no puede capturar relaciones no lineales y es susceptible a la influencia de valores atípicos.

En contraste, la regresión logística se emplea cuando la variable objetivo es categórica, habitualmente binaria. Calcula la probabilidad de pertenencia a una clase aplicando una función sigmoide sobre una combinación lineal de las variables de entrada. Se utiliza extensamente en clasificación binaria debido a su efectividad, particularmente en contextos donde la interpretabilidad resulta esencial.

- **Máquinas de vectores de soporte (SVM).**

Por su parte, las SVM constituyen algoritmos supervisados de clasificación que persiguen el hiperplano óptimo para separar dos clases, maximizando el margen entre ellas. Esta estrategia ofrece una sólida capacidad de generalización, incluso en conjuntos de datos de alta dimensionalidad. Asimismo, mediante funciones kernel, como el kernel Gaussiano, las SVM pueden enfrentar problemas no lineales proyectando los datos a espacios de mayor dimensión. No obstante, su entrenamiento puede resultar oneroso, y la elección apropiada del kernel y los parámetros resulta crucial para su desempeño.

- **Algoritmo de los k vecinos más cercanos (k-NN).**

El algoritmo k-NN es un método basado en instancias que clasifica una muestra según la mayoría de las clases de sus k vecinos más cercanos, determinados por una métrica de distancia (usualmente la euclídea). No requiere un proceso de entrenamiento explícito y es fácil de implementar, aunque su rendimiento puede degradarse con grandes volúmenes de datos o en espacios de alta dimensión, debido a la maldición de la dimensionalidad.

- **Árboles de decisión.**

Los árboles de decisión son modelos jerárquicos que dividen el espacio de características en regiones mediante preguntas binarias basadas en los valores de los atributos. Cada nodo interno representa una decisión sobre una característica, mientras que las hojas representan las clases o valores predichos. Son altamente interpretables y eficaces en problemas con datos mixtos (categóricos y numéricos). Sin embargo, tienden a sobreajustar si no se controla su profundidad, lo que se puede mitigar mediante técnicas de poda o mediante el uso de métodos ensemble.

- **Métodos ensemble: Random Forest y Gradient Boosting.**

Los métodos ensemble combinan múltiples modelos débiles (como árboles de decisión) para obtener un modelo más robusto y preciso.

Random Forest genera un conjunto de árboles de decisión entrenados sobre subconjuntos aleatorios del conjunto de datos y de características, promediando sus resultados. Esto reduce la varianza y mejora la capacidad de generalización.

Gradient Boosting construye modelos secuenciales donde cada nuevo árbol intenta corregir los errores cometidos por los anteriores. Modelos como AdaBoost, Gradient Boosted Trees o XGBoost se basan en esta idea. Aunque son muy potentes, requieren mayor tiempo de entrenamiento y son más sensibles al sobreajuste si no se ajustan correctamente los hiperparámetros.

Con estos fundamentos, pasamos a aplicaciones generales.

2.3.2. Aplicaciones generales del aprendizaje automático

El aprendizaje automático ha transformado múltiples sectores mediante la automatización de procesos, el descubrimiento de patrones complejos y la capacidad predictiva. A continuación se describen algunas de sus aplicaciones más destacadas en diversos ámbitos:

- **Medicina:** El aprendizaje automático se utiliza para el diagnóstico por imagen, donde modelos entrenados con miles de radiografías o resonancias pueden detectar anomalías como tumores pulmonares o hemorragias cerebrales con precisión similar a la de expertos humanos. También se aplica en la predicción de enfermedades crónicas, como la diabetes o enfermedades cardiovasculares, a partir de historiales

clínicos y datos biométricos. Además, permite el monitoreo continuo de pacientes mediante dispositivos wearables, anticipando situaciones de riesgo en tiempo real.

- **Manufactura:** En la industria, se emplea para el mantenimiento predictivo de maquinaria mediante sensores IoT que recogen datos como vibración, temperatura o consumo energético, detectando patrones que preceden a fallos. En control de calidad, las cámaras inspeccionan productos en tiempo real, y algoritmos clasifican defectos o irregularidades. Asimismo, se optimizan procesos de producción mediante modelos que ajustan parámetros dinámicamente para maximizar eficiencia y minimizar desperdicios.
- **Finanzas:** Los modelos de aprendizaje automático son clave en la detección de fraudes en tiempo real, analizando miles de transacciones en busca de comportamientos anómalos. En scoring crediticio, permiten evaluar la solvencia de los clientes incorporando fuentes no tradicionales de datos, como historial de pagos digitales o actividad en redes sociales. También se usan en el análisis de riesgo para carteras de inversión o gestión de activos financieros.
- **Energía:** En el sector energético, se emplean modelos predictivos para estimar la demanda eléctrica en distintas franjas horarias, mejorando la eficiencia de la red. En energías renovables, como la solar o eólica, se predice la generación en función de datos meteorológicos, lo que permite integrar mejor estas fuentes al sistema eléctrico. También se optimiza el rendimiento de instalaciones mediante el análisis continuo de datos operativos.
- **Retail y marketing:** En comercio minorista, se utiliza para la segmentación de clientes mediante análisis de comportamiento de compra, lo que permite campañas de marketing personalizadas. La predicción de demanda ayuda a gestionar inventarios y evitar rupturas de stock. También se aplican modelos dinámicos de fijación de precios (dynamic pricing), ajustando precios en tiempo real según demanda, competencia y otros factores.
- **Otros sectores:** En el ámbito de los seguros, se automatiza la evaluación de siniestros y el cálculo de primas personalizadas. En comercio electrónico, se implementan motores de recomendación que sugieren productos basados en preferencias previas. En robótica, se entrena a robots para realizar tareas complejas como manipulación de objetos o navegación autónoma. En agricultura de precisión, sensores y modelos predicen el momento óptimo para sembrar, regar o aplicar fertilizantes, mejorando el rendimiento y reduciendo el impacto ambiental.

Por último en este bloque, vemos aplicaciones en el deporte.

2.3.3. Aplicaciones en el deporte

El aprendizaje automático se aplica en el ámbito deportivo mediante el análisis de grandes volúmenes de datos provenientes de múltiples fuentes: posiciones y trayectorias de jugadores, patrones de movimiento, métricas fisiológicas, datos de rendimiento, grabaciones de vídeo, entre otros. Su objetivo principal es extraer conocimiento útil para optimizar el rendimiento, prevenir lesiones, mejorar la táctica y enriquecer la experiencia tanto de jugadores como de entrenadores y aficionados.

Entre sus aplicaciones más relevantes se incluyen:

- **Seguimiento de rendimiento y entrenamiento personalizado:** mediante sensores GPS, dispositivos portátiles y cámaras se recopilan datos en tiempo real durante

entrenamientos y competiciones. Algoritmos supervisados y no supervisados permiten identificar patrones de fatiga, cargas de trabajo excesivas y proponer planes de entrenamiento adaptados a las necesidades individuales del deportista.

- **Prevención de lesiones:** el análisis de datos históricos y biométricos permite construir modelos predictivos capaces de identificar factores de riesgo asociados a lesiones musculares o articulares. Estos modelos permiten tomar decisiones preventivas y gestionar de manera más eficiente los periodos de descanso y recuperación.
- **Análisis táctico y estratégico:** la visión por computador y los algoritmos de agrupamiento pueden analizar la disposición táctica de los jugadores en el campo, identificar formaciones, reconocer patrones de juego y evaluar el impacto de ciertas estrategias en tiempo real. Estos análisis son útiles en múltiples disciplinas como fútbol, baloncesto o hockey.
- **Evaluación de talento y scouting:** los equipos profesionales utilizan modelos de aprendizaje automático para analizar automáticamente el rendimiento de posibles incorporaciones. Se tienen en cuenta métricas técnicas (pases, disparos, robos), físicas (velocidad, resistencia) y contextuales (posición, edad, rendimiento relativo) para apoyar decisiones de fichaje.
- **Visión por computador y análisis de vídeo:** tecnologías como la detección de objetos, el seguimiento de movimiento y el reconocimiento de acciones permiten analizar automáticamente grabaciones de partidos. Algoritmos como HOG+SVM, k-NN, YOLO o redes neuronales convolucionales se utilizan para detectar jugadores, balón, zonas de juego y eventos clave como goles o faltas.
- **Modelado y simulación de partidos:** mediante técnicas de aprendizaje por refuerzo, se pueden modelar comportamientos tácticos y simular escenarios complejos para optimizar la toma de decisiones bajo distintas condiciones, como transiciones ofensivas, presión rival o situaciones de balón parado.
- **Interacción con el aficionado y análisis de medios:** el aprendizaje automático también se emplea para generar resúmenes automáticos de partidos, clasificar jugadas según su importancia, detectar momentos destacados y ofrecer estadísticas enriquecidas en tiempo real, mejorando así la experiencia de los espectadores.



Figura 2.1: Aplicaciones de visión por computador en deportes: seguimiento de jugadores y eventos.

En resumen, el aprendizaje automático en el deporte abarca desde modelos predictivos y sistemas personalizados de entrenamiento hasta análisis automatizados de vídeo y simulaciones tácticas. Su integración con tecnologías como sensores, visión artificial y plataformas digitales está transformando radicalmente la forma en que se entrena, se compete y se vive el deporte.

Con estas bases, el siguiente capítulo define los datasets (público y propio) que harán posible una evaluación justa y reproducible del desempeño de SAM en fútbol.

CAPÍTULO 3

Datasets

3.1 Introducción

En el campo de la visión artificial y el aprendizaje profundo, contar con datos etiquetados es algo importante. La calidad y el formato de estas etiquetas son fundamentales para que los modelos funcionen bien y se puedan interpretar correctamente. Bases de datos como COCO (Common Objects in Context) o ImageNet han ayudado mucho al desarrollo de este campo gracias a su gran tamaño y a que han establecido tareas, etiquetas y formas de medir el rendimiento [32, 33]. Para este estudio, requerimos un formato compatible con SAM que permita derivar *prompts* comparables (imagen completa, *bbox* por clase e instancias). Por esta razón, cambiamos DeepSport por Soccer-Panoptic, un conjunto con anotaciones panópticas a nivel de píxel sobre secuencias reales de fútbol [35, 10], lo que habilita una derivación uniforme de *prompts* que compararemos en los Experimentos 4.3–4.4. Este capítulo explica Soccer-Panoptic en detalle y luego presenta el conjunto de datos propio, Vinalesa FC, que usamos para examinar el ajuste con pocos datos y control de la variabilidad. A continuación, justificamos la elección de Soccer-Panoptic y su encaje con SAM. En primer lugar, se presentará brevemente SAM y, posteriormente, se desarrollarán los experimentos.

3.2 Soccer-Panoptic (Kaggle)

3.2.1. Motivación de la elección

Elegimos **Soccer-Panoptic** por tres razones prácticas y alineadas con los objetivos del trabajo. A continuación se describe su composición y qué partes utilizamos en este trabajo:

1. **Anotación panóptica:** cada píxel está asociado a *clase* e *id de instancia*, lo que simplifica derivar tanto máscaras *semánticas* (por clase) como *de instancia* (una por jugador/objeto) desde una única fuente.
2. **Preparación directa de *prompts* SAM:** de la máscara panóptica se extraen de forma sistemática (i) la *bbox* de imagen completa (no asistida), (ii) la *bbox* global por clase (asistida) y (iii) *bboxes* específicas por instancia (asistidas), manteniendo constante el resto del flujo.
3. **Curva de entrada baja y modificación sencilla:** el volumen es manejable para análisis cuadro a cuadro; las secuencias son de *broadcast* y las anotaciones se distribuyen con metadatos claros [10].

3.2.2. Descripción y composición

extbfSoccer–Panoptic reúne **secuencias de fútbol** con **anotación panóptica** generada sobre *frames* muestreados uniformemente. Cada fotograma cuenta con una máscara en la que los valores de píxel codifican *identificadores de instancia* asociados a *categorías* como *players* (jugadores), *ball* (balón), *referees* (árbitros), *field* (césped), *pitch lines* (líneas de campo), *goal lines* y *background*. En esta memoria nos centramos en **jugadores** y **balón** para alinear métricas y comparativas con los experimentos de SAM [10].

Tabla 3.1: Soccer–Panoptic: resumen operativo para esta memoria (clases relevantes, formato, uso).

oprule Origen Conjunto público (Kaggle) [10]

3.2.3. Estructura y preparación para SAM

Para facilitar la reproducibilidad, reorganizamos el dataset con la estructura mostrada en el Anexo A.1.

De panoptic a prompts SAM. Desde la máscara panóptica generamos:

- **Máscaras por clase** (binarias): p. ej., *players* y *ball*.
- **Máscaras por instancia**: una por cada jugador y por el balón (cuando aparece).
- **BBoxes**: (i) imagen completa, (ii) envolvente global por clase y (iii) envolventes por instancia.

El procedimiento detallado para derivar estas máscaras y *bboxes* se recoge en el Anexo A.2.

Extracción de frames (si parte de vídeo). Cuando el conjunto incluye vídeo, extraemos frames con *ffmpeg*, manteniendo el *timestamp* original [11, 12]. Con la preparación lista, definimos las particiones y lotes de evaluación:

```
1 ffmpeg -i partido.mp4 -r 30 -qscale:v 2 images/frame_%05d.jpg
```

Listing 3.1: Extracción de frames con *ffmpeg*.

Con la preparación lista, definimos las particiones y lotes de evaluación.

3.2.4. Particiones y *batches* de evaluación

Para un análisis equilibrado, seleccionamos **720 frames** distribuidos en **tres lotes** (**Batch 1–3**) según la **geometría y dificultad** de la escena:

- **Batch 1** (escena compacta): pocos jugadores, escala grande (más píxeles por jugador), fondo relativamente limpio.
- **Batch 2** (bordes finos y poses complejas): contornos desafiantes (brazos, manos), suelos de alto brillo y líneas visibles.
- **Batch 3** (escena amplia y dispersa): muchos jugadores, solapes, balón pequeño/ocluído y líneas de campo que confunden los bordes.

Estas tres condiciones son las que empleamos en los **Experimentos 1 y 2** (cap. SAM): fijan una lectura robusta por caso de *prompting* y reflejan caídas esperadas de mIoU cuando la caja se aproxima a la imagen completa o el balón es minúsculo.

Para cerrar la especificación, indicamos qué clases se miden y con qué propósito.

3.2.5. Esquema de clases y uso en evaluación

Aunque el dataset ofrece varias categorías, en este trabajo reportamos métricas sobre:

- **Jugadores** (*players*): clase dominante y principal objetivo de segmentación.
- **Balón** (*ball*): objeto pequeño, con alta variabilidad y frecuentes oclusiones.

El resto de clases (árbitros, campo, líneas) actúan como **contexto** y ayudan a interpretar fallos típicos (p. ej., confusión con líneas de área en Batch 3), pero no computamos métricas sobre ellas.

Tabla 3.2: Clases del dataset y papel en esta memoria.

Clase	Tipo	Uso aquí
Players	Instancia	Métrica (mIoU, por batch y global)
Ball	Instancia	Métrica (mIoU, por batch y global)
Referees	Instancia	Contexto/visualización cualitativa
Field	Semántica	Contexto (evitar fugas al césped)
Pitch/Goal lines	Semántica	Contexto (bordes de alto contraste)
Background	Semántica	Contexto

3.2.6. Ejemplos visuales

La paleta utilizada en toda la memoria es: **GT** (jugadores: verde translúcido; balón: rojo translúcido) y **SAM** (jugadores: azul translúcido; balón: amarillo translúcido), siempre *por encima* de la GT.

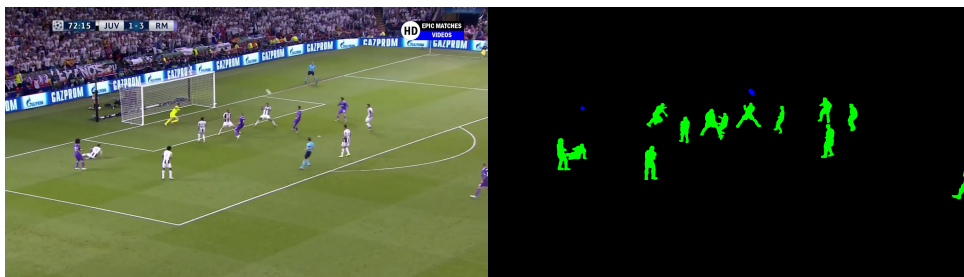
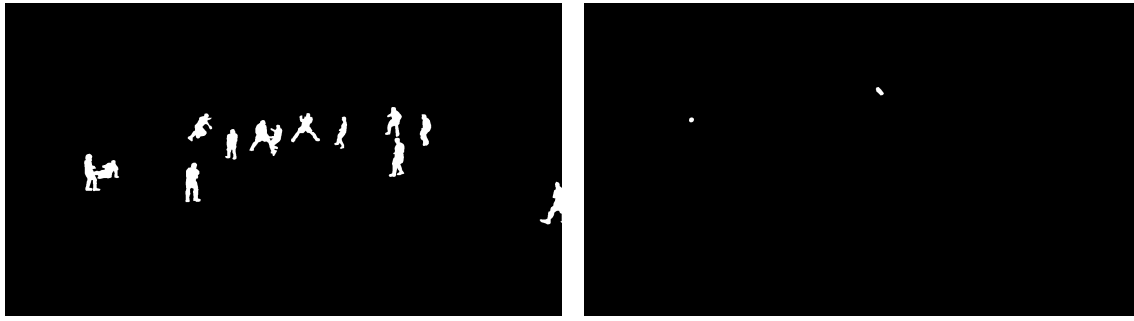


Figura 3.1: Vista general de Soccer–Panoptic (frame y máscara panóptica).

Con estos ejemplos en mente, recogemos buenas prácticas y controles de calidad aplicados.

3.2.7. Buenas prácticas y control de calidad

El ciclo típico de creación/curado de datasets (recolección, limpieza, anotación, validación, estructura) sigue siendo válido y recomendable [22, 39]. En Soccer–Panoptic aplicamos:



(a) Máscara por clase: *players*.

(b) Máscara por clase: *ball*.

Figura 3.2: Derivación de máscaras por clase desde la anotación panóptica.



Figura 3.3: Overlay GT (verde/rojo) vs. predicción SAM (azul/amarillo) en un frame de ejemplo.

- **Chequeo de consistencia:** detección de máscaras vacías o ids huérfanos; verificación de que *ball* no invade regiones imposibles (portería, gradas).
- **Normalización de resolución:** evitar escalados excesivos para no introducir aliasing en objetos pequeños (balón).
- **Trazabilidad:** guardar el *mapping* panoptic $\rightarrow$ clase/instancia y las *bboxes* derivadas por versión.

Finalmente, explicitamos las limitaciones y sesgos relevantes del dominio.

3.2.8. Limitaciones y sesgos

La distribución natural del dominio introduce retos:

- **Desbalance por clase:** *players* domina frente a *ball*; los píxeles del balón son pocos y sensibles a *blur*/oclusiones.
- **Líneas de campo y alto contraste:** inducen *fugas* en bordes, sobre todo con *prompts* poco informativos (imagen completa, Grid/AMG no asistida).
- **Variabilidad de la escena amplia** (Batch 3): dispersión de instancias y solapes reducen el mIoU incluso con cajas globales por clase.

Estas limitaciones motivan tanto la incorporación de **Grid/AMG** como la exploración de *fine-tuning* específico, documentados en los capítulos correspondientes.

Consideradas las limitaciones del conjunto público, presentamos ahora Vinalesa FC para explorar el ajuste con datos escasos y un mayor dominio del contexto.

3.3 Vinalesa FC (dataset propio)

3.3.1. Descripción y propósito

Para estudiar el *fine-tuning* con datos **escasos** y bajo control, construimos un **dataset propio** con imágenes reales del **Vinalesa FC**. El objetivo es doble: (i) analizar si el ajuste del *mask decoder* (encoder congelado) aporta *ganancias consistentes* frente a las líneas base no asistidas, y (ii) entender la **sensibilidad al tamaño del conjunto** y a la diversidad de escenas. A continuación se resume el proceso de creación y anotación.

3.3.2. Proceso de creación

1. **Captura:** fotografías originales de jugadores del club en distintos contextos (entrenamiento/partido), variando pose, escala, fondo e iluminación.
2. **Selección:** 30 imágenes finales (diversidad y calidad), con presencia/ausencia de balón.
3. **Anotación:** segmentación **manual** en **Roboflow** para *jugadores* y *balón*; revisión de contornos finos (brazos, manos).
4. **Exportación:** máscaras binarias por clase (PNG) y/o COCO JSON para integrarlas en el *pipeline*. Posteriormente, establecemos las versiones y particiones que emplearemos en los experimentos.

Definimos después las versiones y particiones que usaremos en los experimentos.

3.3.3. Versionado y splits

Definimos dos versiones con distinta cobertura:

- **Vinalesa v1** (13 imágenes): valida convergencia con muy pocos datos.
- **Vinalesa v2** (27 imágenes): mayor diversidad; **split 80–20** (entrenamiento/validación) para estimar generalización.

Tabla 3.3: Resumen de Vinalesa FC.

Atributo	Valor
Imágenes totales	30 (v1: 13; v2: 27)
Clases	2 (jugadores, balón)
Anotación	Segmentación manual (Roboflow)
Uso en esta memoria	<i>Fine-tuning</i> decoder-only (ViT-B), splits por tamaño

3.3.4. Ejemplos

Cerramos con notas operativas para mantener la calidad de anotación y exportación.

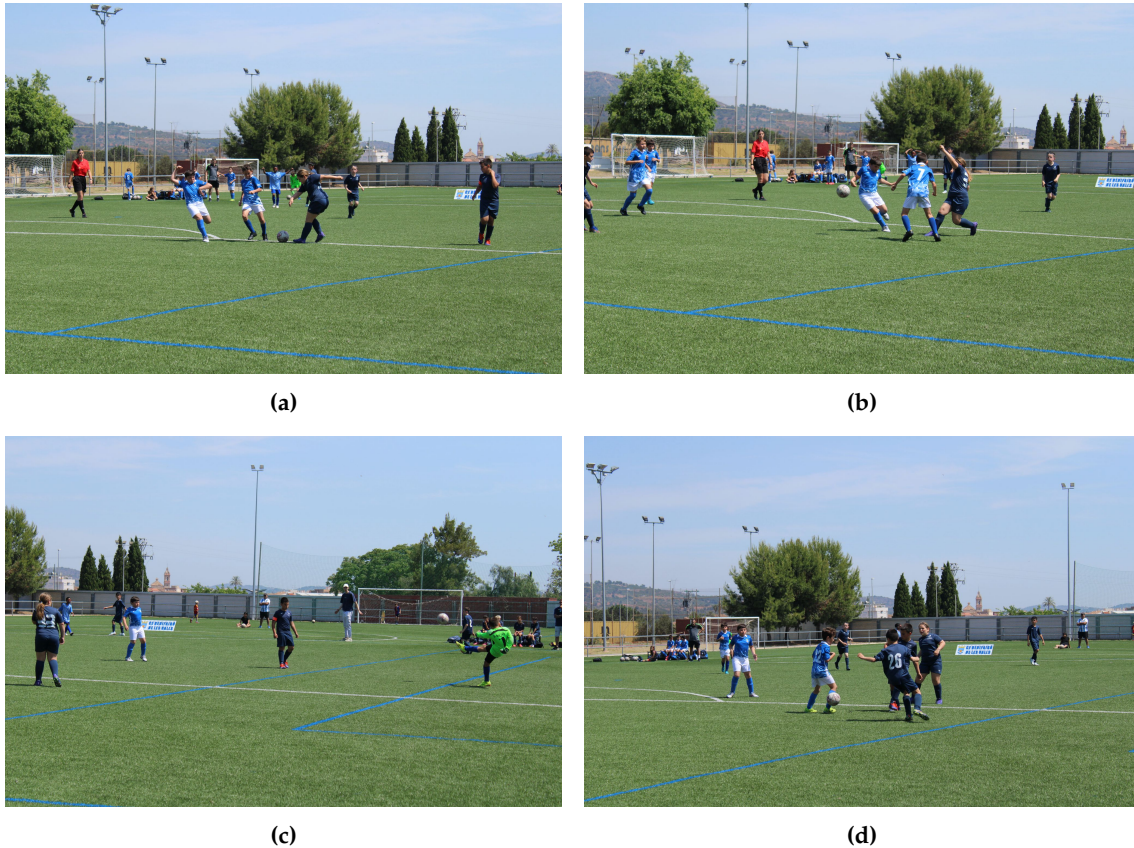


Figura 3.4: Mosaico de ejemplos (Vinalesa FC).

3.3.5. Notas prácticas

- **Consistencia de contornos:** revisar bordes finos a ≥ 200 % de zoom para evitar *dientes de sierra*.
- **Aumento de datos:** rotaciones leves, *color jitter*, recortes y *motion blur* controlado; mantener la escala del balón.
- **Exportación** en Roboflow: preferible conservar ambas salidas (PNG por clase y COCO JSON) para trazabilidad.



Figura 3.5: Overlay GT (verde/rojo) vs. predicción de SAM (azul/amarillo) en imágenes de Vinalesa FC.

3.4 Conclusiones del capítulo

El objetivo del bloque de Datasets era fundamentar una base de datos adecuada para evaluar SAM en fútbol y, en paralelo, disponer de un conjunto propio que nos permitiera controlar la variabilidad y validar ajustes finos. Con ese propósito, Soccer-Panoptic se adopta como eje principal porque ya viene en un formato COCO/panoptic muy compatible con el flujo de máscaras y bboxes requerido por SAM, con escenas variadas y anotaciones que facilitan tanto la evaluación por clase como por lote; en otras palabras, cumple la función de banco de pruebas amplio y reproducible que pedía el objetivo. Paralelamente, iniciativas como SoccerNet proporcionan tareas y protocolos estandarizados para visión en fútbol (detección, seguimiento, eventos, segmentación), ofreciendo un marco de comparación complementario [38]. Por su parte, Vinalesa aporta el complemento controlado: imágenes estáticas del mismo dominio, con anotaciones manuales cuidadas, idóneas para estudiar estabilidad, riesgo de sobreajuste y sensibilidad con tamaños de datos pequeños (v1: 13 imágenes; v2: 27 imágenes). Juntas, ambas fuentes satisfacen la doble meta del capítulo: un conjunto grande, directo para SAM (Soccer-Panoptic), y un conjunto propio y compacto para experimentos de validación fina (Vinalesa).

Con los datos y protocolos ya fijados, el siguiente capítulo presenta SAM y ejecuta dos experimentos comparables (cajas y Grid/AMG) que establecen la línea base sobre la que se valorará el *fine-tuning*.

CAPÍTULO 4

SAM

4.1 Introducción

4.2 SAM

Segment Anything Model 2 (SAM2) es una evolución de Segment Anything Model (SAM), un modelo de segmentación desarrollado por Meta AI que permite identificar y delimitar objetos en imágenes de manera flexible y adaptable.

La arquitectura de este sistema se basa en un enfoque modular, compuesto por tres componentes clave: un codificador de imagen, un codificador de prompts y un decodificador de máscaras. Con este contexto, pasamos al Experimento 1.

4.3 Experimento 1 – Evaluación base de SAM en fútbol

4.3.1. Introducción

Este primer experimento establece la línea base de precisión de *Segment Anything Model* (SAM) en el dominio del fútbol, en configuración cero-disparos y sobre las dos clases de interés del trabajo: **jugadores** y **balón**. El objetivo es medir de forma aislada el impacto del *prompting* basado en cajas sobre la calidad de la segmentación y fijar un punto de comparación para los experimentos posteriores. En concreto, contrastamos tres niveles de información contextual en la entrada: (i) *bbox global de imagen* (“imagen completa”), una única caja coincidente con todo el frame; (ii) *bbox parcial por clase* (“bbox global por clase”), una caja que envuelve la región ocupada por la clase en cada frame; y (iii) *bbox específica por instancia*, una caja por cada componente conectado (cada jugador y el balón). Manteniendo constante el flujo de inferencia y evaluación, probamos SAM con **ViT-B** y **ViT-H** como *backbones* de referencia, habilitando salida multimáscara y seleccionando por *IoU* la hipótesis más adecuada cuando corresponde. La métrica principal es **mIoU**, promediada por clase y frame válido. Para estudiar robustez frente a la distribución espacial de objetos, el conjunto de evaluación agrupa escenas en lotes con distinta densidad y dispersión (desde pocos jugadores hasta múltiples jugadores con solapes), lo que permite analizar cómo se degrada el rendimiento cuando la caja cubre regiones más amplias y heterogéneas. Partimos de la hipótesis de que la *bbox específica por instancia* proporcionará la mayor precisión al aportar el contexto más informativo por objeto, que la *bbox parcial por clase* funcionará razonablemente bien en escenas compactas pero perderá eficacia con alta dispersión, y que la *bbox global de imagen* tenderá a fallar al carecer de guía espacial suficiente—especialmente con múltiples jugadores o con un balón pequeño u ocluido.

Con este diseño, la sección sienta una referencia cuantitativa clara sobre la que comparar enfoques más avanzados (p. ej., rejillas/AMG, agrupación o *fine-tuning* específico del dominio).

4.3.2. Tecnologías empleadas

SAM (Segment Anything). Utilizamos el predictor oficial de SAM, en configuración de inferencia con el *backbone* ViT-B (Vision Transformer). El predictor acepta *prompts* (puntos, cajas, máscaras) y devuelve una o varias hipótesis de máscara por objeto (vía `multimask_output`).

Entorno y librerías. El experimento se ejecutó en Linux (WSL) con Python, PyTorch para inferencia, OpenCV/NumPy para pre/post-procesado y ffmpeg para la extracción de frames. La evaluación y agregación de métricas se implementó en scripts propios siguiendo formato COCO-like para máscaras binarias por clase.

4.3.3. Diseño experimental y casos de prompting

Se evaluaron tres *casos* (estrategias) con SAM (ViT-B), aplicados sobre todos los frames anotados de tres lotes (*Batch 1–3*) de secuencias en HD:

1. **Bboxes específicos por componente conectado (“Bbox específicos”):** para cada componente conectado en la máscara *ground truth* (e.g., cada jugador individual y el balón), se deriva un *bounding box* específico y se invoca el predictor una vez por cada caja. Es la estrategia más informada (desagregada) por objeto.
2. **BBox global por clase :** se calcula una única caja global envolvente de toda la máscara de clase (jugadores o balón) y se llama a SAM con `multimask_output=True`; se escoge la hipótesis con mayor IoU por clase y frame. Es un compromiso entre simplicidad y especificidad.
3. **BBox de imagen completa (“Imagen completa”):** se usa la caja de toda la imagen como único *prompt*. Es la configuración más simple y, a priori, la menos informativa.

Adicionalmente se empleó selección de hipótesis por IoU cuando `multimask_output=True`. Todas las métricas por clase y lote se promediaron sobre los frames con presencia efectiva del objeto de interés.

Con este diseño fijado, pasamos a definir la métrica utilizada.

4.3.4. Métrica

La métrica principal es **mIoU**, definida a partir del *IoU* (*índice de Jaccard*) entre la predicción P y la máscara de referencia G :

$$\text{IoU}(P, G) = \frac{|P \cap G|}{|P \cup G|}.$$

El *mIoU* reportado se obtiene promediando por frames válidos y por clase (jugadores, balón).

Con la métrica fijada, presentamos los resultados cuantitativos.

4.3.5. Resultados cuantitativos

En la Tabla 4.1 se resume el *promedio final* por enfoque para ViT-B (todos los frames), desglosado por clase. Los detalles por lote se visualizan en las Figuras 4.1 y 4.2. Los valores provienen del fichero de evaluación consolidada.

Tabla 4.1: Promedio final mIoU (ViT-B) por enfoque y clase.

oprulr extbfMétodo	Jugadores (mIoU)	Balón (mIoU)
Bbox específicos (asistido)	0,800	0,805
Bbox global (asistido)	0,651	0,562
Bbox imagen completa (no asistido)	0,000	0,000

Para mayor claridad, también incluimos sendas figuras comparativas de los promedios globales (Figuras 4.3 y 4.4).

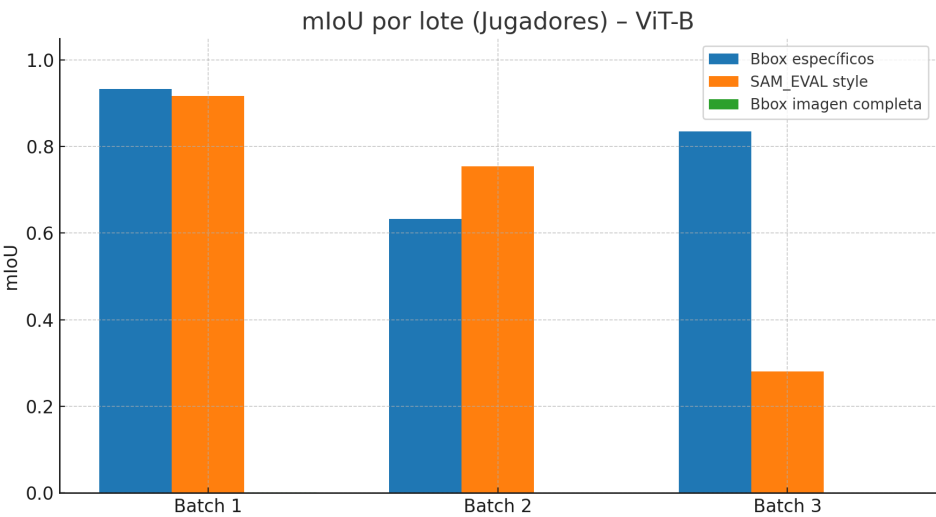


Figura 4.1: mIoU por lote (Jugadores, ViT-B) para los tres enfoques: *Bbox específicos*, *Bbox global* y *Imagen completa*.

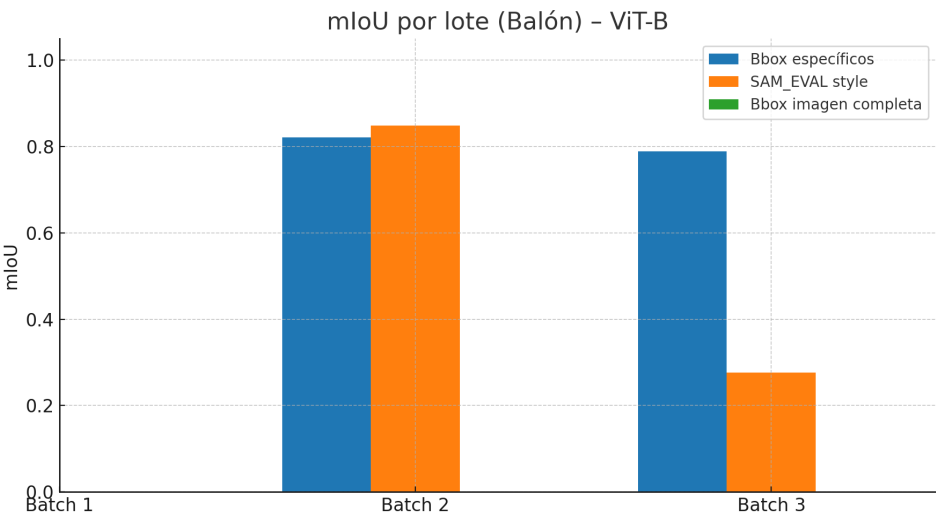


Figura 4.2: mIoU por lote (Balón, ViT-B). En *Batch 1* no hay balón anotado.

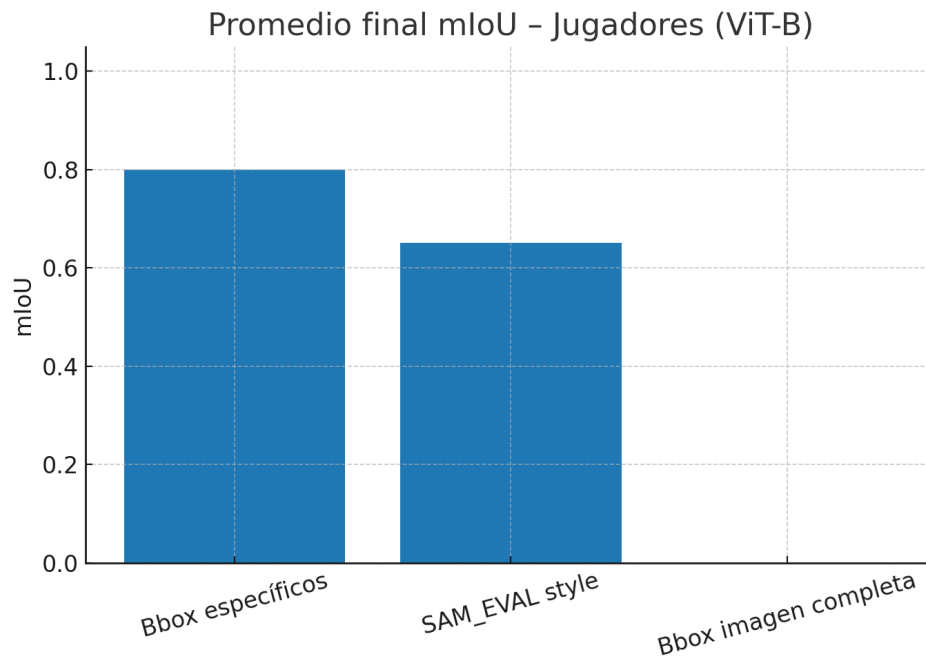


Figura 4.3: Promedio final mIoU (Jugadores, ViT-B) por enfoque.

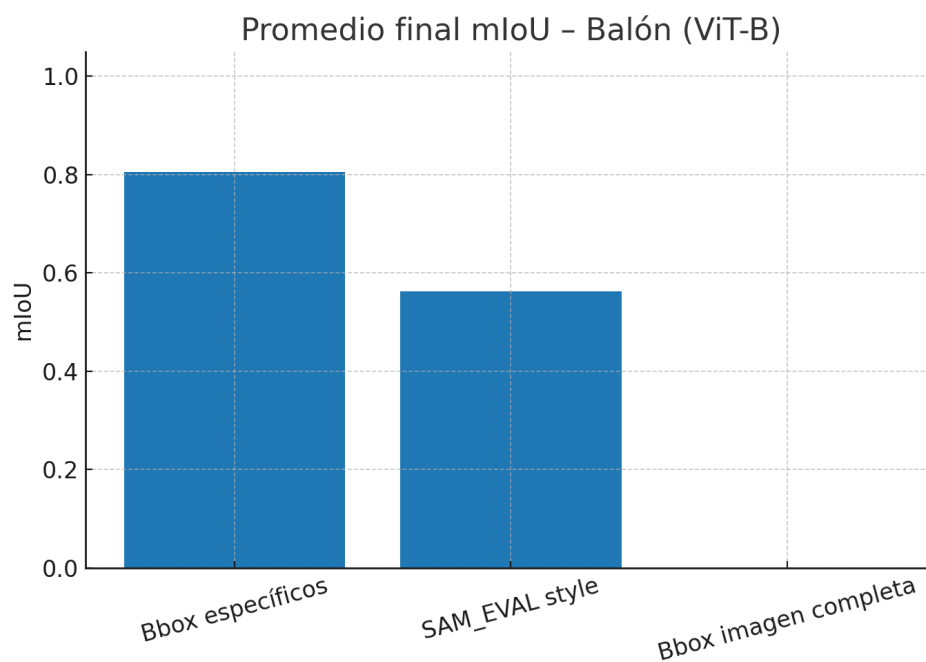


Figura 4.4: Promedio final mIoU (Balón, ViT-B) por enfoque.

Para interpretar mejor estos valores, pasamos a ejemplos cualitativos por lote.

4.3.6. Análisis cualitativo (Batch 1)

En la Figura 4.5 se ilustran los tres enfoques de *bbox* del Experimento 1 sobre un frame representativo del *Batch 1* (jugador centrado y bien definido). La paleta es consistente: **GT** usa **verde** translúcido para *jugadores* y **rojo** translúcido para *balón*; las **predicciones de SAM** usan **azul** translúcido para *jugadores* y **amarillo** translúcido para *balón*, siempre *por encima* de la GT para facilitar la inspección visual.



Figura 4.5: Experimento 1 — Ejemplo cualitativo (Batch 1). GT en verde/rojo; SAM en azul/amarillo, superpuesto a la GT.

Lectura rápida. BBoxes específicos por instancia. La máscara azul sigue con precisión el contorno del jugador (cabeza y hombros), con desajustes mínimos en bordes finos. Es el comportamiento esperado al aportar una caja estrecha por objeto.

BBox global por clase. Con un jugador dominante y compacto, la caja única por clase continúa siendo informativa: la predicción azul se alinea bien con la GT verde, con un leve halo en el perfil de la cabeza. Este caso explica los buenos promedios de *Batch 1*.

BBox de imagen completa. Sin guía de localización, el modelo no discrimina al jugador: la superposición azul útil desaparece o queda desestructurada, coherente con el mIoU cercano a cero de este enfoque.

Tras este caso, continuamos con el siguiente lote.

4.3.7. Análisis cualitativo (Batch 2)

La Figura 4.6 muestra los tres enfoques basados en *bbox* sobre un fotograma del *Batch 2* con dos jugadores en pose arrodillada y bordes finos (brazos, manos). En este frame no hay balón anotado/visible, por lo que la lectura se centra en **jugadores**. La leyenda cromática se mantiene: **GT en verde** translúcido (jugadores) y **rojo** (balón); **SAM en azul** (jugadores) y **amarillo** (balón), superpuesto sobre la GT.

Lectura rápida. BBoxes específicos por instancia. Buena adherencia de la máscara azul a contornos complejos (rodillas, antebrazos, manos), con ligeras pérdidas en estructuras muy finas y en zonas de brillo sobre la línea blanca. Mantiene la separación entre instancias.

BBox global por clase. La caja única por clase sigue siendo informativa con sujetos dominantes; las máscaras azules cubren bien ambos jugadores, pero aparecen pequeñas fugas y, en ocasiones, tendencia a fusionar zonas de ambas instancias cuando están cercanas.

BBox de imagen completa. Falta de guía: el método tiende a segmentar el fondo (césped



(a) BBoxes específicos por instancia



(b) BBox global por clase (Bbox global)



(c) BBox de imagen completa

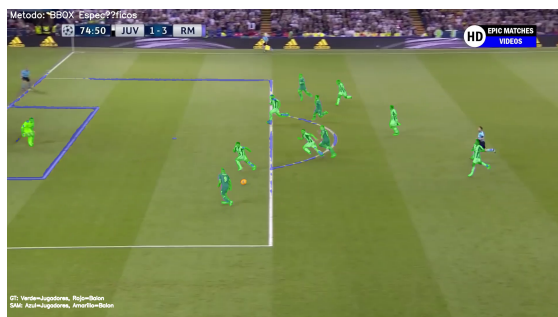
Figura 4.6: Experimento 1 — Ejemplo cualitativo (Batch 2). GT: verde/rojo; SAM: azul/amarillo, superpuesto a la GT.

o graderío) en grandes regiones azules, mientras pierde detalle en los jugadores. Esto explica el mIoU próximo a cero observado para esta configuración.

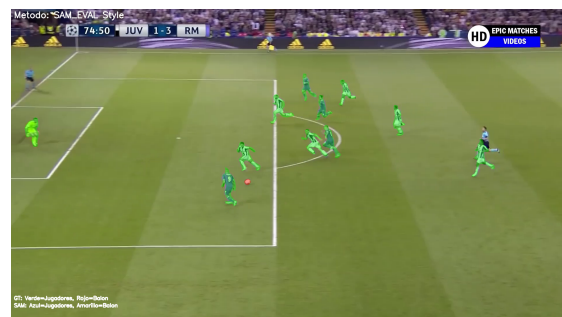
Tras este caso, continuamos con el siguiente lote.

4.3.8. Análisis cualitativo (Batch 3: caso difícil)

La Figura 4.7 ilustra un ejemplo del *Batch 3* donde **ninguno de los métodos** ofrece una segmentación satisfactoria. Es una escena amplia con múltiples jugadores, solapes y líneas blancas fuertes que introducen ambigüedad. La paleta se mantiene: **GT** en **verde** (jugadores) y **rojo** (balón); **SAM** en **azul** (jugadores) y **amarillo** (balón), dibujado *por encima* de la GT.



(a) BBox de imagen completa



(b) BBox global por clase (Bbox global)

Figura 4.7: Experimento 1 — Ejemplo cualitativo (Batch 3). Escena difícil con múltiples instancias y líneas de campo que inducen errores; GT en verde/rojo y SAM en azul/amarillo.

- **BBoxes específicos por instancia.** A pesar de disponer de cajas por objeto, aparecen *huecos* y *fugas* en contornos (especialmente en jugadores solapados); algunas extremidades quedan fuera y hay confusión con las líneas del área.
- **BBox global por clase.** La caja envolvente de la clase reúne demasiada variabilidad espacial: la predicción tiende a *fusionar* jugadores cercanos y a invadir el césped; pierde separación de instancias.
- **BBox de imagen completa.** Sin guía espacial, el modelo colorea grandes regiones del fondo y apenas captura estructura útil de los jugadores.

Cerramos así la parte cualitativa y pasamos al análisis y discusión.

4.3.9. Análisis y discusión

Ranking y conclusiones cuantitativas. El enfoque **Bbox específicos** obtiene la mayor precisión de forma consistente (*Jugadores* 0,800; *Balón* 0,805), seguido de **Bbox global** (*Jugadores* 0,651; *Balón* 0,562). La estrategia de **Imagen completa** resulta inefectiva (mIoU ≈ 0) en este dominio.

Efecto de la dispersión de los objetos. En escenas con *jugadores concentrados* (p.ej., *Batch 1*), la caja global por clase funciona razonablemente bien (*Jugadores* $\approx 0,917$). Sin embargo, cuando los objetos están *muy dispersos* (p.ej., *Batch 3*), el *bbox* global se aproxima a la imagen completa, degradando la segmentación (*Jugadores* $\approx 0,281$, *Balón* $\approx 0,276$ con *Bbox global*). En esos casos, *Bbox específicos* mantiene resultados altos al descomponer la escena por objeto (*Jugadores* 0,835; *Balón* 0,789).

Selección multimáscara y robustez. La opción `multimask_output` y la selección por IoU aportan robustez cuando el *bbox* contextual es suficientemente informativo (p.ej., cajas compactas por clase u objeto). Con cajas *excesivamente grandes*, el predictor tiende a segmentar fondo o elegir subconjuntos poco representativos.

Lecciones prácticas. (i) En fútbol, la *unidad de segmentación* adecuada es el *objeto individual* (jugador/ balón), no la clase completa en una sola caja; (ii) la *dispersión* espacial penaliza las cajas globales; (iii) la *simplicidad extrema* (imagen completa) no es una opción viable con SAM en este dominio; (iv) como línea base sólida sin entrenamiento, *Bbox específicos* es la referencia a batir para mecanismos más avanzados (rejillas/ AMG, agrupación posterior o *fine-tuning*).

Este razonamiento se verá más claro en el código del siguiente apartado.

4.3.10. Código ilustrativo (implementación)

BBoxes por componente conectado (*Bbox específicos*).

```

1 def bboxes_from_mask(mask, min_area=30):
2     m = (mask > 0).astype(np.uint8)
3     num, labels = cv2.connectedComponents(m)
4     boxes = []
5     for lab in range(1, num):
6         ys, xs = np.where(labels == lab)
7         if ys.size < min_area:
8             continue
9         x0, y0, x1, y1 = xs.min(), ys.min(), xs.max(), ys.max()
10        boxes.append((int(x0), int(y0), int(x1), int(y1)))
11    return boxes

```

```

12
13 predictor.set_image(img_rgb)
14 for (x0,y0,x1,y1) in bboxes_from_mask(gt_players):
15     masks, scores, _ = predictor.predict(box=np.array([x0,y0,x1,y1]),
16         multimask_output=True)
17     best = max(masks, key=lambda m: iou_bool(m, gt_players))

```

Listing 4.1: Extracción de bboxes por componente y llamada a SAM por objeto

bbox global por clase con selección por IoU.

```

1 bbox_players = get_mask_bbox(gt_players.astype(np.uint8))
2 if bbox_players is not None:
3     masks_pred, scores, _ = predictor.predict(box=bbox_players, multimask_output=
4         True)
5     best_mask = max(masks_pred, key=lambda m: iou_bool(m.astype(bool), gt_players
6         ))

```

Listing 4.2: Caja global de clase + multimasca + mejor IoU

Comparación justa: cambiar solo el bbox (imagen completa vs. bbox global). El código que implementa esta comparación se recoge en el Anexo [A.3](#).

Visualización GT vs. predicción (overlay por clase). La función utilizada para superponer la GT (verde/rojo) y las predicciones de SAM (azul/amarillo) se detalla en el Anexo [A.4](#).

Con estos fragmentos como referencia, cerramos el experimento con sus limitaciones y trabajo futuro.

4.3.11. Limitaciones y trabajo futuro

No se exploran en este experimento estrategias de *aumento de prompts* (puntos guiados, rejillas adaptativas por regiones ni *ensembles* de hipótesis), ni *post-procesado morfológico* para refinar bordes. El siguiente paso lógico es estudiar métodos que reduzcan la dependencia del *bbox* (p.ej., *Automatic Mask Generation* o agrupación) y, en última instancia, *fine-tuning* específico del dominio para robustecer las clases difíciles (balón pequeño/ocluído, solapes múltiples).

Finalmente, resumimos los hallazgos clave del experimento.

4.3.12. Resumen

Bbox específicos ofrece la *mejor precisión y consistencia* sin entrenamiento adicional. **Bbox global** es una alternativa simple y razonable en escenas compactas, pero cae en escenas dispersas. **Imagen completa** es descartable en fútbol. Estas conclusiones establecen una referencia cuantitativa clara para las mejoras que se abordan en experimentos posteriores.

Debido a algunos **resultados inesperados** con el enfoque basado en *bbox* —en particular, la **ineficacia de la imagen completa** ($mIoU \approx 0$) y la **caída de la caja global por clase** en escenas dispersas— se plantea evaluar una alternativa **no asistida** que no dependa de pistas espaciales: **Grid/AMG**. Tras constatar los límites con *bboxes*, comparamos ahora un enfoque no asistido (Grid/AMG) bajo el mismo protocolo.

Tabla 4.2: Resumen cualitativo por enfoque (ViT-B).

Ranking	Enfoque	Ventajas	Desventajas
1°	Bbox específicos	Máxima precisión y estabilidad en escenas densas y dispersas; selección multimáscara efectiva.	Requiere derivar bboxes por componente; mayor coste de inferencia por frame.
2°	Bbox global	Simplicidad; buen rendimiento en escenas compactas; aprovecha multimáscara.	Sensible a dispersión; con bboxes muy grandes tiende a segmentar fondo/-subconjuntos.
3°	Imagen completa	Implementación trivial.	Ineficaz en este dominio ($mIoU \approx 0$).

4.4 Experimento 2 – Grid/AMG vs BBox en fútbol

4.4.1. Introducción

Este experimento evalúa el desempeño de *Segment Anything Model* (SAM) en modo **Grid/Automatic Mask Generation (AMG)** frente a variantes basadas en **cajas** (BBox) en el dominio del fútbol, sobre el conjunto de Soccer_Panoptic. La idea principal es distinguir entre métodos *no asistidos* (sin pistas de localización) y métodos *asistidos* (que llevan información sobre la solución). En particular, hay que fijarse que la comparación **justa** de referencia para un método no asistido como Grid/AMG es el **BBox de imagen completa** (ambos sin ayuda); los enfoques de *BBox global por clase* y *BBox específico por instancia* incluyen *pistas* de la solución (filtración de información) y deben interpretarse como *metodos asistidos*.

4.4.2. Metodología

Escenarios comparados. Partiendo de las **tres configuraciones** evaluadas en el **Experimento 1** (Sección 4.3) —(i) *BBox de imagen completa (no asistido)*, (ii) *BBox global por clase (asistido)* y (iii) *BBox específico por instancia (asistido)*— en este experimento, hemos añadido una cuarta configuración: (iv) Grid / Generación Automática de Máscaras (AMG) como un método no asistido. Para garantizar que sea estrictamente comparable con el Experimento 1, mantenemos el backbone ViT-B, la métrica mIoU (promediada por clase —jugadores, balón— y por lote) y el mismo flujo de evaluación.

Fairness de la comparación. Siguiendo el diseño y la lectura del Experimento 1, la comparación principal debe enfocarse en Grid/AMG frente a BBox de imagen completa, ya que ambos son métodos no asistidos (sin pistas de localización). Las variantes con bboxes que derivan de la referencia —BBox global por clase y BBox específico por instancia— incluyen pistas parciales de la solución y, por lo tanto, solo se utilizan como un techo asistido; no son directamente comparables cuando el objetivo es validar una solución sin ayuda.

Limitaciones prácticas (tiempo/hardware) y efectos en Grid. A diferencia del Experimento 1, donde las variantes BBox no necesitan una exploración densa del plano de imagen, Grid/AMG requiere una generación masiva de propuestas. Esto, en una máquina con recursos limitados, nos llevó a implementar ciertas restricciones operativas: un número moderado de `points_per_side`, un `downscale` de la imagen y un límite de máscaras por frame. En nuestras pruebas, los valores típicos fueron `points_per_side=8`,

downscale=0.75 y un máximo de aproximadamente 20 máscaras por imagen. Es importante entender que estos ajustes son compromisos de ingeniería diseñados para reducir los tiempos y evitar bloqueos, no mejoras intrínsecas del método. Cualquier ganancia que observemos se debe al equilibrio entre coste y tiempo bajo recursos limitados, y no a un cambio conceptual en Grid.

Con la metodología definida, presentamos los resultados.

4.4.3. Resultados

Para contextualizar los números, pasamos a los ejemplos cualitativos. La Tabla 4.3 resume los promedios globales (ViT-B) y las Figuras 4.8–4.11 muestran los desgloses por lote y por clase.

Tabla 4.3: Promedio final mIoU (Jugadores/Balón) por método (ViT-B, 720 frames).

oprulr extbfMétodo	Jugadores (mIoU)	Balón (mIoU)
Bbox específicos (asistido)	0,800	0,805
Bbox global (asistido)	0,651	0,562
Grid/AMG (no asistido)	0,476	0,071
Bbox imagen completa (no asistido)	0,000	0,000

Lectura justa (no asistidos): **Grid/AMG** supera claramente a **BBox imagen completa**, estableciendo que el uso de rejilla sin pistas sí aporta segmentaciones útiles aun sin localización previa (aunque lejos del techo asistido).

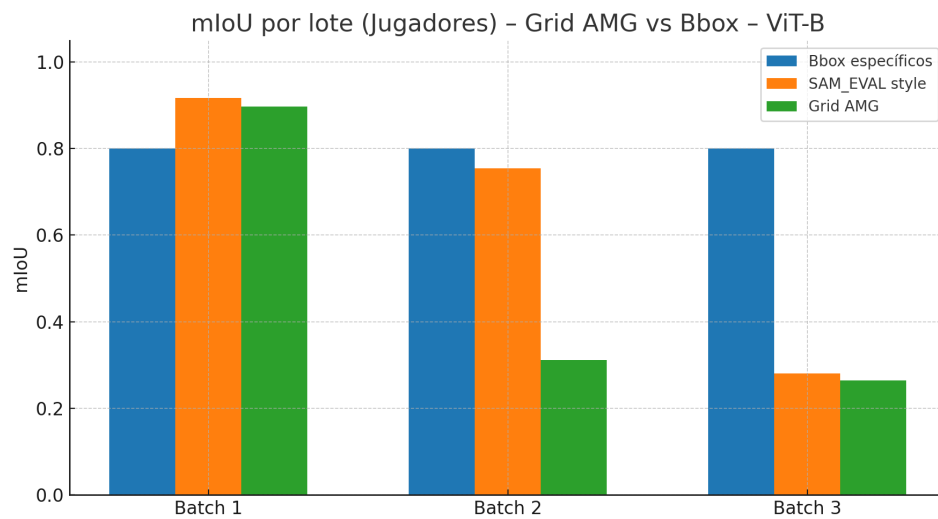


Figura 4.8: mIoU por lote (Jugadores) – Grid/AMG vs BBox (ViT-B).

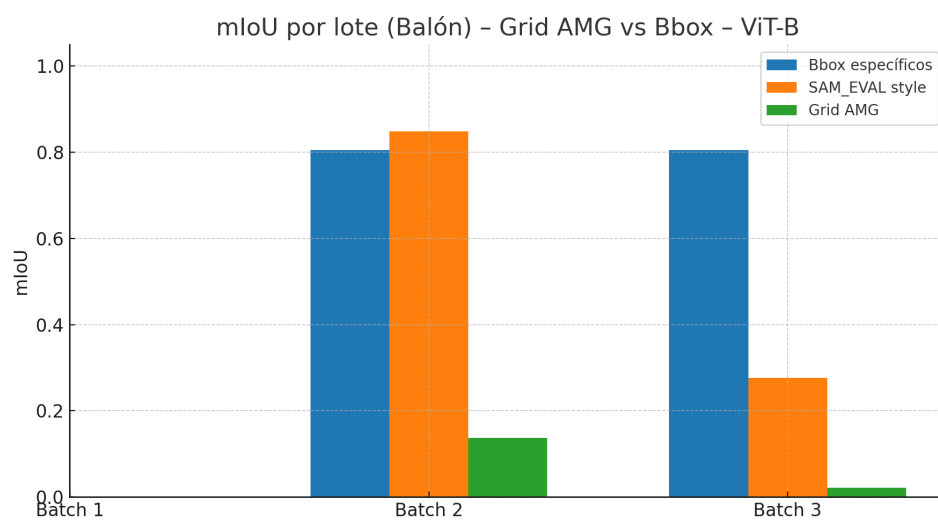


Figura 4.9: mIoU por lote (Balón) – Grid / AMG vs BBox (ViT-B). En *Batch 1* no hay balón anotado.

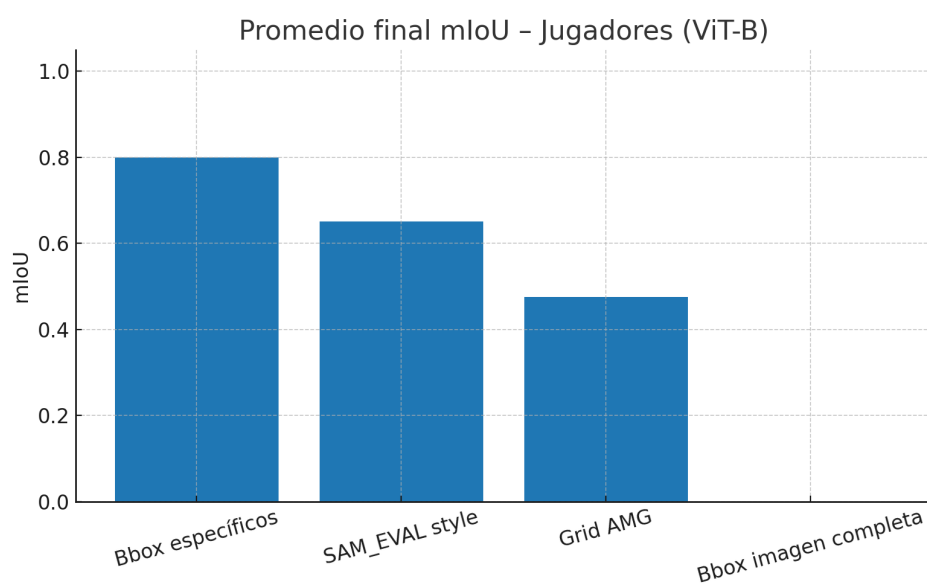


Figura 4.10: Promedio final mIoU (Jugadores) por método (ViT-B).

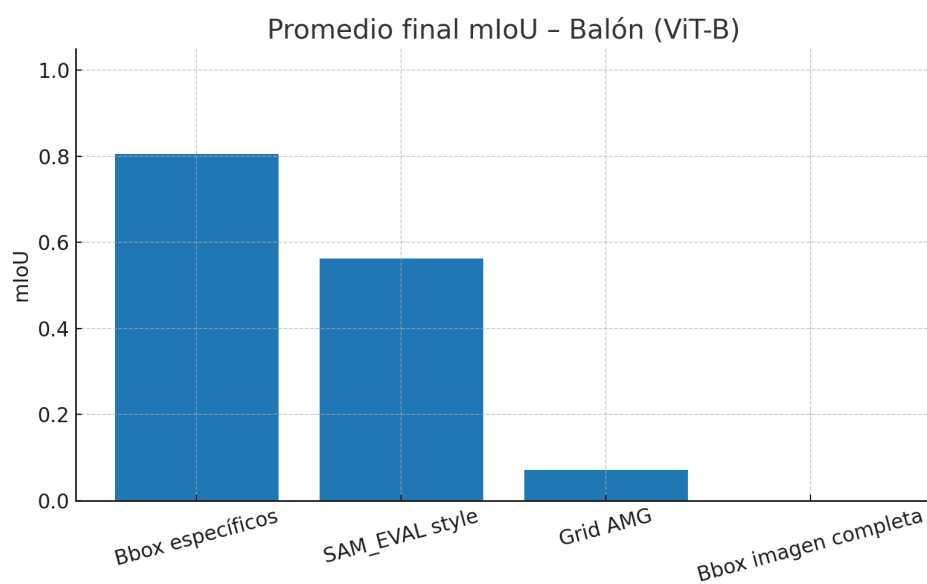


Figura 4.11: Promedio final mIoU (Balón) por método (ViT-B).

4.4.4. Análisis cualitativo (Batch 1)

La Figura 4.12 compara **Grid/AMG** con las variantes basadas en *bbox*. Recordamos que la comparación *justa* entre métodos *no asistidos* es **Grid/AMG** frente a **BBox de imagen completa**; las opciones con *bboxes* derivadas de la referencia son *asistidas* y se interpretan como techos.



Figura 4.12: Experimento 2 — Ejemplo cualitativo (Batch 1). GT en verde/rojo; SAM en azul/amarillo, superpuesto a la GT.

Lectura rápida. Grid/AMG (no asistido). La rejilla genera varias propuestas; la de mayor IoU cubre correctamente al jugador (azul sobre verde), con contornos razonables. En escenas compactas como este *Batch 1*, el mejor candidato suele coincidir con el jugador.

BBox de imagen completa (no asistido). Confirma la comparación justa: sin pistas, la segmentación de jugador falla o se diluye.

BBox global por clase (asistido). La guía espacial moderada produce una máscara azul de buena calidad; se observan pequeñas discrepancias en bordes finos.

BBoxes específicos por instancia (asistido). Caja precisa por objeto → la cobertura azul es la más ajustada, evidenciando que parte de la solución proviene de la propia caja.

Continuamos con el siguiente lote.

Lectura rápida. BBoxes específicos por instancia. Buena adherencia de la máscara azul a contornos complejos (rodillas, antebrazos, manos), con ligeras pérdidas en estructuras muy finas y en zonas de brillo sobre la línea blanca. Mantiene la separación entre instancias.

BBox global por clase. La caja única por clase sigue siendo informativa con sujetos dominantes; las máscaras azules cubren bien ambos jugadores, pero aparecen pequeñas fugas y, en ocasiones, tendencia a fusionar zonas de ambas instancias cuando están cercanas.

BBox de imagen completa. Falta de guía: el método tiende a segmentar el fondo (césped

o graderío) en grandes regiones azules, mientras pierde detalle en los jugadores. Esto explica el mIoU próximo a cero observado para esta configuración.

4.4.5. Análisis cualitativo (Batch 2)

La Figura 4.13 compara **Grid/AMG** con las variantes basadas en *bbox* sobre el mismo frame del *Batch 2*. Para una lectura justa entre *no asistidos*, se contrasta **Grid/AMG** frente a **BBox de imagen completa**; las configuraciones con *bboxes* derivadas de la referencia actúan como *techos asistidos*.

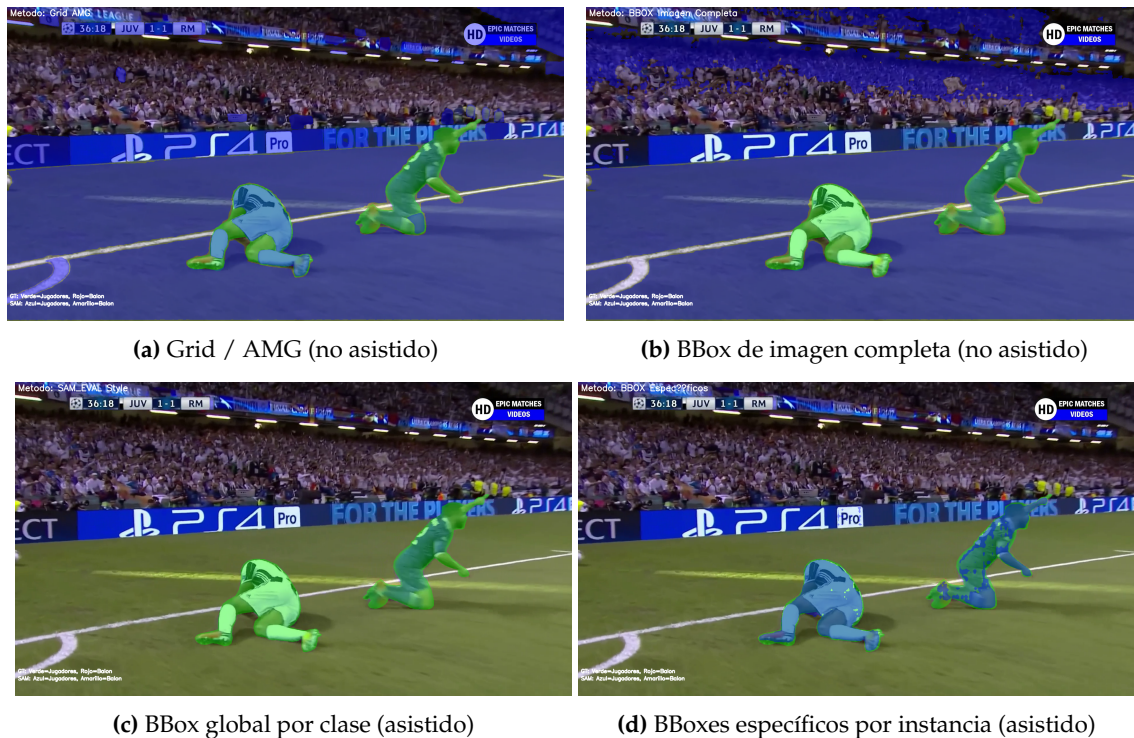


Figura 4.13: Experimento 2 — Ejemplo cualitativo (Batch 2). GT: verde/rojo; SAM: azul/amarillo, superpuesto a la GT.

Lectura rápida. Grid/AMG (no asistido). La rejilla propone múltiples máscaras; la mejor cubre razonablemente al jugador en primer plano (azul sobre verde), pero aparecen *blobs* espurios en el graderío y zonas amplias del césped coloreadas, síntoma de sobrecobertura cuando falta guía espacial.

BBox de imagen completa (no asistido). Confirma el patrón: el método tiende a alfombrar el fondo y pierde al sujeto principal.

BBox global por clase (asistido). Con la guía de clase, las máscaras azules capturan ambos jugadores con calidad, aunque persisten pequeños errores en bordes finos y zonas de alto brillo.

BBoxes específicos por instancia (asistido). Es la configuración más precisa: mantiene separación entre jugadores y ajusta bien contornos complejos, sirviendo de *techo asistido* para esta escena.

Continuamos con el siguiente lote.

4.4.6. Análisis cualitativo (Batch 3: caso difícil)

En la Figura 4.14 se compara **Grid/AMG** con las variantes basadas en *bbox* sobre el mismo frame complicado del *Batch 3*. Subrayamos que es un **ejemplo límite** donde *ningún método* alcanza una segmentación adecuada.

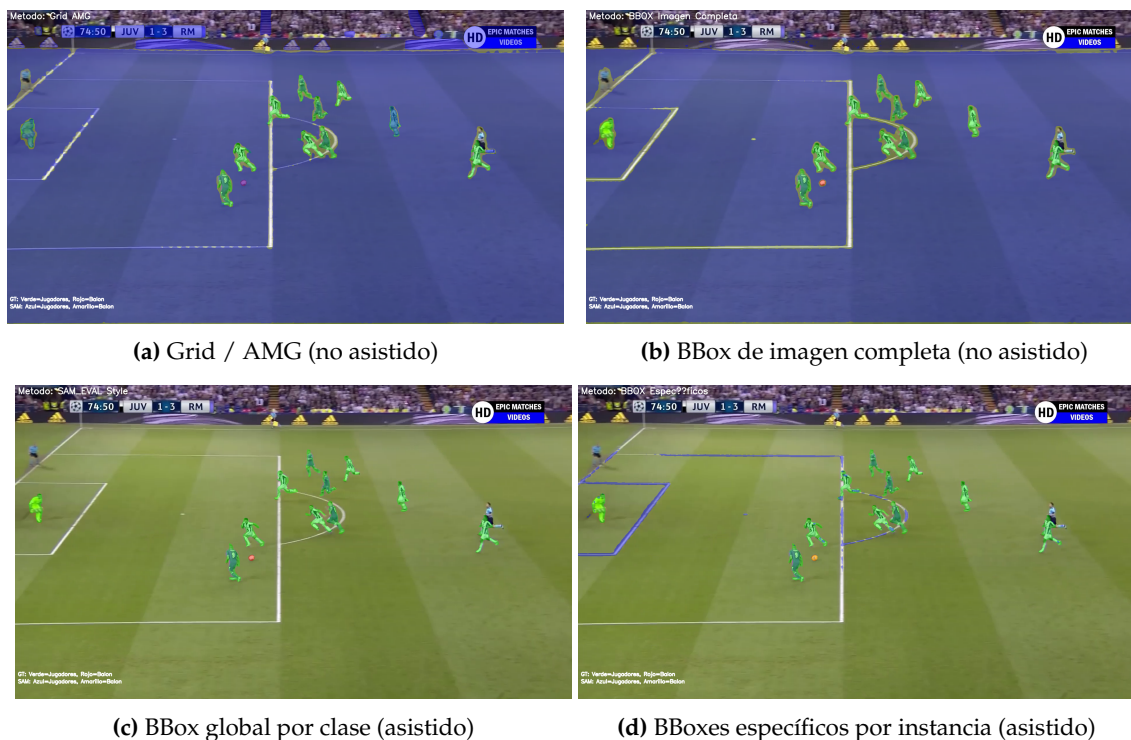


Figura 4.14: Experimento 2 — Ejemplo cualitativo (Batch 3). Ningún método capta bien el frame: GT en verde/rojo y SAM en azul/amarillo.

Lectura rápida. Grid/AMG (no asistido). La rejilla genera *blobs* espurios en grada y césped; aunque alguna propuesta toca a ciertos jugadores, la cobertura es inconsistente y ruidosa.

BBox de imagen completa (no asistido). Resultado dominado por fondo: grandes áreas del terreno aparecen coloreadas y los jugadores se diluyen.

BBox global por clase (asistido). La guía de clase mejora respecto a los no asistidos, pero la variabilidad espacial y los solapes provocan *fusión de instancias* y errores distribuidos por la zona del área.

BBoxes específicos por instancia (asistido). Incluso con cajas por objeto, los solapes y las líneas de campo inducen *fugas* y *recortes* en contornos; la precisión cae notablemente frente a escenas compactas.

Este caso ilustra el **peor escenario** del dominio: planos amplios con muchos jugadores, oclusiones y texturas de alto contraste (líneas y sombras). Sirve para contextualizar el descenso de mIoU del *Batch 3* y motiva estrategias adicionales (rejillas adaptativas, fusión de propuestas, o *fine-tuning* específico).

Cerramos la parte cualitativa y pasamos a los fragmentos de código.

4.4.7. Código ilustrativo (implementación)

AMG básico con SamAutomaticMaskGenerator (no asistido).

```
1 from segment_anything import sam_model_registry, SamAutomaticMaskGenerator
```

```

2 sam = sam_model_registry["vit_b"](checkpoint="sam_vit_b.pth").to(device).eval()
3 mask_generator = SamAutomaticMaskGenerator(
4     model=sam,
5     points_per_side=8,
6     pred_iou_thresh=0.70,
7     stability_score_thresh=0.50,
8     crop_n_layers=1,
9 )
10 masks = mask_generator.generate(img_rgb) # propuestas sin pistas
11 best = max(masks, key=lambda d: iou_bool(d["segmentation"], gt_mask))

```

Listing 4.3: Generacion automatica con grid y umbrales base

Grid alternativo con SamPredictor (batcheando puntos).

```

1 def create_dense_grid_points(h, w, pps=8):
2     xs = np.linspace(0, w-1, pps, dtype=int)
3     ys = np.linspace(0, h-1, pps, dtype=int)
4     xx, yy = np.meshgrid(xs, ys)
5     return np.column_stack([xx.ravel(), yy.ravel()])
6
7 predictor.set_image(img_rgb)
8 points = create_dense_grid_points(*img_rgb.shape[:2], pps=8)
9 best_iou, best_mask = 0.0, None
10 for i in range(0, len(points), 64):
11     masks, scores, _ = predictor.predict(
12         point_coors=points[i:i+64],
13         point_labels=np.ones(min(64, len(points)-i), dtype=int),
14         multimask_output=True
15     )
16     for m in masks:
17         iou = compute_iou_bin(m.astype(np.uint8), gt_mask)
18         if iou > best_iou:
19             best_iou, best_mask = iou, m

```

Listing 4.4: Rejilla densa manual, procesada por lotes, con seleccion por IoU

Compromisos de ingenieria (limitacion de mascaras y downscale).

```

1 mask_generator = SamAutomaticMaskGenerator(
2     model=sam,
3     points_per_side=8,
4     pred_iou_thresh=0.75,
5     stability_score_thresh=0.60,
6     crop_n_layers=0,
7     min_mask_region_area=200,
8     output_mode='binary_mask',
9 )
10 # Evaluar solo las ~20 mejores mascaras por frame y usar downscale=0.75

```

Listing 4.5: Configuracion usada para acotar tiempo/memoria en Grid

Comparación justa (recordatorio): mismo método, distinto bbox. El código correspondiente a esta comparación se encuentra en el Anexo [A.5](#).

Con esta base implementada, discutimos las implicaciones y límites del experimento.

4.4.8. Discusión

extbfFair comparison y jerarquía de métodos. Considerando **sólo métodos no asistidos**, Grid/AMG *mejora* a BBox de imagen completa (0.476 vs 0.000 en jugadores; 0.071 vs 0.000 en balón), confirmando que una cobertura densa de la imagen con una rejilla de propuestas es preferible a carecer de guía. Al comparar con métodos **asistidos**, el *techo* lo marcan las variantes con BBox (0.800/0.805), en especial el BBox específico por instancia, que, por construcción, *contiene parte de la solución* al heredar localización precisa (no es *zero-shot* real).

Efecto de la geometría. En escenas compactas (Batch 1), *SAM-EVAL style* y Grid alcanzan cifras altas en jugadores (0.917 y 0.897), pero con dispersión (Batches 2–3) Grid cae (0.312; 0.265) y la caja global por clase también (0.754; 0.281), mientras BBox específico mantiene consistencia. Esto refuerza que, sin pistas, la rejilla sufre con objetos pequeños o muy dispersos (balón: 0.071 de media).

Coste computacional y compromisos. Aunque *conceptualmente* Grid parece más simple, en la práctica su generación masiva de máscaras implica más tiempo que variantes BBox; parte de estas cifras dependen de toques de máscaras, *downscale* y *points_per_side*, fijados para adaptarse al hardware disponible. No interpretamos esos ajustes como “mejoras del método”, sino como **límites prácticos** para poder completar la evaluación.

4.4.9. Limitaciones y trabajo futuro

(i) Evaluación sujeta a **restricciones de recursos** (número de frames y densidad de rejilla limitados), (ii) sensibilidad de Grid a **parámetros** (*points_per_side*, umbrales de IoU/estabilidad, tope de máscaras, *downscale*); (iii) balón pequeño/ocluído sigue siendo difícil sin ayuda. Futuros pasos: rejillas adaptativas por regiones, heurísticas de fusión de propuestas, o introducir pistas *no derivadas de la ground truth* (p.ej., *priors* de detección genéricos) que preserven la comparación justa.

4.4.10. Conclusiones

Para el objetivo de una solución **no asistida**, Grid/AMG es claramente superior a **BBox de imagen completa** y establece una base razonable sin filtración de información. Aun así, el **gap** frente a los métodos asistidos justifica explorar rejillas adaptativas, fusión de máscaras y, llegado el caso, mecanismos de localización *no leakage*. Las “mejoras” observadas bajo ajustes de cómputo deben leerse como **compromisos por tiempo/hardware**, no como ventajas intrínsecas del algoritmo.

4.5 Conclusiones — SAM (Experimentos 1 y 2)

En coherencia con el objetivo del bloque SAM —evaluar el modelo en *zero-shot* bajo comparaciones justas y medir el impacto del *prompting*— los dos experimentos trazan con claridad qué puede y qué no puede ofrecer SAM en fútbol.

1. **Línea base con BBoxes (Experimento 4.3).** Cuando se proporciona la pista espacial mínima y correcta por instancia, SAM ofrece segmentaciones fiables: el *bbox* específico por objeto (una caja por cada jugador/ balón) fija el mejor rendimiento observado (**mIoU jugadores: 0,800; balón: 0,805**). La caja global por clase (*Bbox global*) es competitiva en escenas compactas, pero se degrada con alta dispersión

(mIoU menores, fusiones de instancias y fugas al fondo). La caja de imagen completa resulta ineficaz ($mIoU \approx 0$) al carecer de guía.

Conexión con el objetivo: esta jerarquía ancla la **línea base** y delimita el **techo asistido** frente al cual juzgar mejoras posteriores.

2. **No asistidos: Grid/AMG (Experimento 4.4).** Para una comparación *justa* entre no asistidos, **Grid/AMG supera** con claridad a la **imagen completa (jugadores: 0,476 vs 0,000; balón: 0,071 vs 0,000)**. No obstante, queda lejos del techo asistido que marcan los *bboxes* específicos.

Conexión con el objetivo: se cumple el propósito de disponer de una **línea base no asistida defendible**, sin filtración de información.

3. **Lectura cualitativa por lotes.** La evidencia visual respalda la lectura numérica: (i) en *Batch 1* (jugadores bien delimitados), la *caja global* y *Grid/AMG* funcionan razonablemente; (ii) en *Batch 2* (poses complejas y bordes finos), el *bbox* específico mantiene la mejor adherencia a contornos; (iii) en *Batch 3* (escenas amplias con solapes y líneas de campo), ningún método ofrece segmentación plenamente satisfactoria.

Conexión con el objetivo: la inspección cualitativa explica el *porqué* de las diferencias de mIoU y acota los **casos límite** del dominio.

4. **Coste y sensibilidad.** *Grid/AMG* implica mayor coste computacional que las variantes con *bboxes* y es sensible a parámetros como *points_per_side*, umbrales de IoU/estabilidad, *downscale* y tope de máscaras. Los ajustes usados deben leerse como **compromisos de ingeniería** por limitaciones de hardware/tiempo, no como mejoras intrínsecas del algoritmo.

Conexión con el objetivo: se documentan los **compromisos** que condicionan cualquier despliegue real y la reproducibilidad de la evaluación.

5. Guía práctica.

- Si hay **localización fiable por objeto**, usar ***bboxes* específicos** con **selección multimáscara**: mayor precisión y separación de instancias.
- Si sólo hay **una pista por clase**, *Bbox global* es aceptable en escenas compactas; **pierde** con dispersión.
- **Sin pistas (no asistido)**, preferir **Grid/AMG** frente a imagen completa; considerar **rejillas adaptativas** y **filtrado de propuestas**.
- **Evitar imagen completa** como único *prompt* en fútbol: no aporta segmentación útil en la práctica.

Conexión con el objetivo: estas pautas operativizan la **línea base** y orientan comparaciones futuras en condiciones realistas.

6. **Implicaciones para el resto del trabajo.** Los resultados apuntan a dos rutas de mejora: (i) mecanismos de **propuesta/localización previos a SAM** que no filtren *ground truth* (p. ej., rejillas adaptativas, detección genérica, agrupación de máscaras) y (ii) ***fine-tuning* específico de dominio** para reducir la dependencia de pistas espaciales y robustecer clases difíciles como el balón pequeño u ocluido.

Conexión con el objetivo: el capítulo de *Fine-tuning* aborda explícitamente esta segunda vía y **cuantifica su ganancia** frente a la línea base establecida aquí.

Delimitado el techo *zero-shot* y no asistido, evaluamos si un ajuste específico del *mask decoder* mejora la utilidad práctica.

CAPÍTULO 5

Fine-tuning de SAM

5.1 Introducción

Este capítulo estudia el *fine-tuning* de *Segment Anything* (SAM) en el dominio del fútbol con dos clases de interés: **jugadores** y **balón**. La motivación es superar las líneas base **no asistidas** establecidas previamente (**Grid/AMG** y **BBox de imagen completa**) sin depender de pistas de localización, y cuantificar la distancia respecto a variantes **asistidas** basadas en *bboxes* (global por clase y específica por instancia), que en el capítulo de evaluación sin entrenamiento se utilizaron como techos de referencia. Para evitar filtración de información, el foco comparativo principal se mantiene en métodos **sin ayuda**, mientras que las configuraciones asistidas se usan únicamente como orientación del rendimiento alcanzable bajo guía espacial. El objetivo práctico es seleccionar el **modelo recomendado por escenario** (sólo jugadores; jugadores+balón; condiciones variadas) documentando el compromiso entre **precisión** y **coste computacional**.

5.2 Fine-tuning

Configuración del modelo. Se utiliza **SAM ViT-B** con **encoder congelado** y entrenamiento restringido al *mask decoder*. Optimizador Adam ($\text{lr} = 10^{-5}$) y pérdida BCEWithLogitsLoss. Cuando el tamaño del conjunto lo permite, se ejecutan **50 épocas**; en entrenamientos por lote (*per-batch*) se reduce el número de épocas para contener el tiempo total.

Datasets y particiones.

- **Vinalesa v1/v2.** Conjuntos de 13 y 27 imágenes (COCO) con las dos clases de interés. Partición 70/15/15 (train/val/test), con *batch size* 1 (v1) y 2 (v2).
- **Split 80-20 (27 imágenes).** División 80/20 para evaluar generalización (test con 20 %).
- **Soccer Dataset (720 frames).** Tres lotes: *Batch 1* (sólo jugadores), *Batch 2* (jugadores+balón), *Batch 3* (variado). Entrenamiento **por lote** con **5 épocas** y evaluación por lote.

Métricas. La métrica principal es **mIoU**. Además, en el split 80-20 se reportan **IoU**, **Dice**, **precisión** y **recall** para **Global**, **Balón** y **Jugador**.

Restricciones de cómputo. Se mantiene un protocolo realista: *batch size* reducido, número de épocas acotado en entrenamientos por lote y, cuando procede, límites de máscaras por imagen. Estas decisiones buscan reproducibilidad en hardware modesto sin presentar “mejoras” debidas únicamente a más cómputo.

Con la configuración fijada, presentamos los experimentos y resultados.

5.3 Experimentos

Vinalesa v1 vs v2 (50 épocas, decoder-only)

Tabla 5.1: Evolución de pérdidas (entrenamiento/validación) en Vinalesa v1 (13) y v2 (27).

Conjunto	Pérdida inicial (train)	Pérdida final (train)	Pérdida inicial (val)	Pérdida final (val)
Vinalesa v1 (13)	0,0216	0,0031	0,0088	0,0062
Vinalesa v2 (27)	0,0292	0,0039	0,0101	0,0067

Lectura. Ambos entrenamientos convergen de forma estable; v2 ofrece validación más robusta y curvas más suaves por mayor diversidad y *batch size* ligeramente superior, reduciendo riesgo de sobreajuste.

Split 80-20 (27 imágenes)

Tabla 5.2: Métricas en test (split 80/20).

Categoría	IoU	Dice	Precisión	Recall
Global	0.7252	0.8297	0.4375	0.9065
Balón	0.8515	0.9197	0.4620	0.9898
Jugador	0.7023	0.8134	0.4330	0.8914

En la Figura 5.1 se comparan IoU/Dice por categoría y en la Figura 5.2 se muestran precisión y *recall*. El **balón** alcanza IoU $\approx 0,85$ con *recall* cercano a 1, mientras que **jugadores** se sitúa en IoU $\approx 0,70$ con buena sensibilidad y margen de mejora en precisión (falsos positivos).

A continuación, resultados por lote en Soccer Dataset.

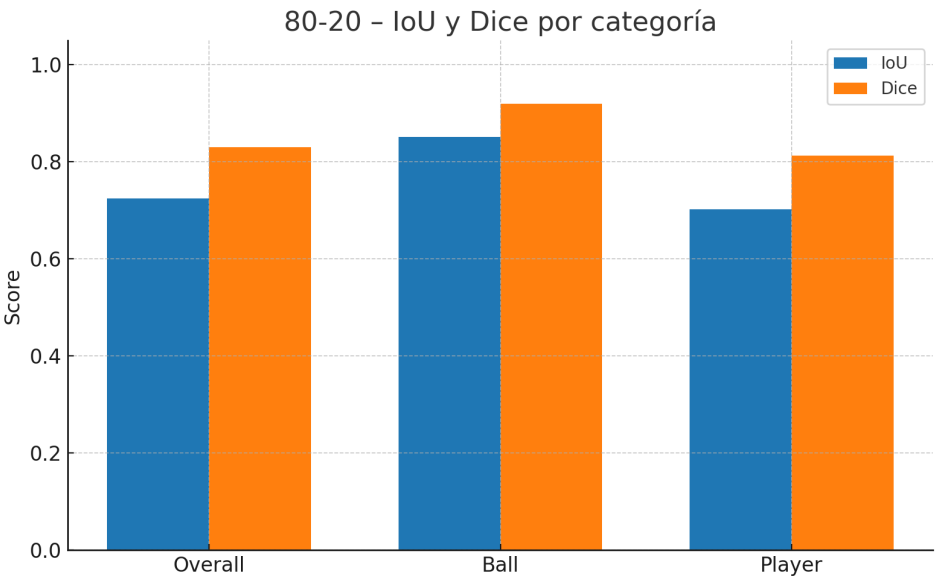


Figura 5.1: Split 80-20 — IoU y Dice por categoría (Global, Balón, Jugador).

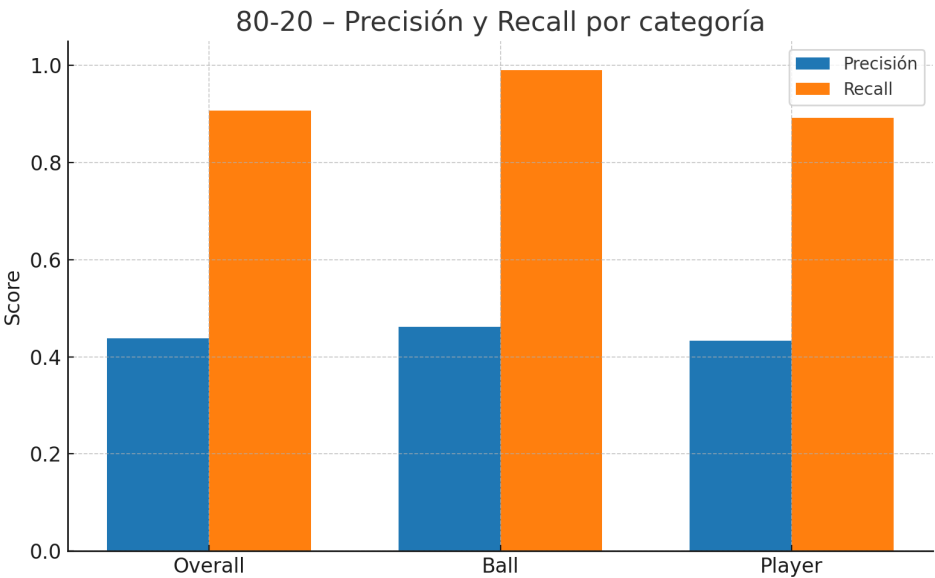


Figura 5.2: Split 80-20 — Precisión y Recall por categoría.

Soccer Dataset por lote (5 épocas por lote)

Tabla 5.3: Rendimiento por lote tras fine-tuning (mIoU en test).

oprule extbfMétodo	Jugadores (mIoU)	Balón (mIoU)
Bbox específicos (asistido)	0,800	0,805
Bbox global (asistido)	0,651	0,562
Bbox imagen completa (no asistido)	0,000	0,000

La Figura 5.3 resume los resultados por lote. El **Batch 1** destaca (sólo jugadores). El **Batch 3** es el más retador (variabilidad y balón pequeño/ocluído), sugiriendo que más épocas o datos específicos mejorarían el cierre de la brecha.

Para dimensionar la mejora, comparamos frente a líneas base.

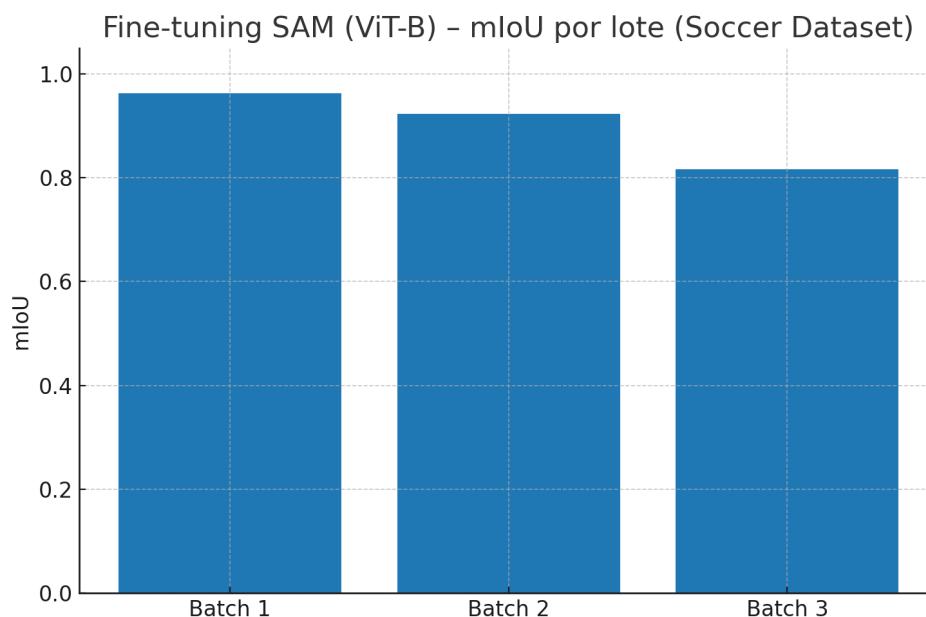


Figura 5.3: Fine-tuning SAM (ViT-B) — mIoU por lote en Soccer Dataset.

Comparación con líneas base

Para comparar de manera **justa** con métodos **no asistidos**, tomamos como referencia **Grid/AMG** y **BBox de imagen completa**; el *fine-tuning* se sitúa claramente por encima de ambos en todos los lotes. Como referencia adicional, frente al SAM base asistido con **BBox específicos** (cifras en torno a 0,80 de mIoU), el modelo **fine-tuned** presenta mejoras por lote, tal y como muestra la Figura 5.4.

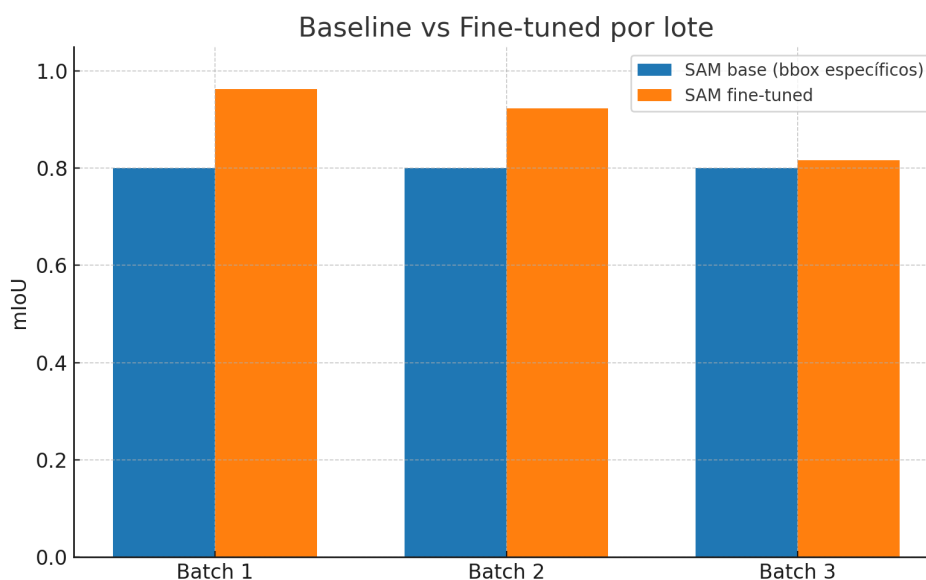


Figura 5.4: Comparación por lote: SAM base (bbox específicos) vs SAM *fine-tuned*.

Código ilustrativo (implementación)

Orquestación por lotes (Batch 1/2/3). El fragmento de código usado para lanzar el entrenamiento por lote está en el Anexo A.6.

Descubrimiento y *split* por batch (70/15/15). El código para crear el dataset por lote y realizar la partición reproducible se muestra en el Anexo [A.7](#).

Decoder-only: congelar *encoder* y optimizar sólo *mask decoder*. En este esquema de *fine-tuning*, todas las capas del **encoder** de imágenes permanecen congeladas (sin actualizar sus pesos), y únicamente se entrena el **mask decoder**. De esta forma, se reduce el riesgo de sobreajuste con pocos datos y se concentra la adaptación en la parte final del modelo, responsable de generar las máscaras. El código que implementa esta configuración se incluye en el Anexo [A.8](#).

“Sólo cambia el BBox”: comparativa justa de métodos. El código completo de esta función se incluye en el Anexo [A.9](#).

Embeddings de imagen, *prompt* de caja y *mask decoder*.

```

1 # 1) Embedding de imagen
2 image_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
3 image_trans = transform.apply_image(image_rgb)
4 image_t = torch.as_tensor(image_trans, device=device).permute(2,0,1).unsqueeze(0)
5 image_t = sam.preprocess(image_t)
6 with torch.no_grad():
7     image_embedding = sam.image_encoder(image_t)
8
9 # 2) Preparar la caja y codificar prompt
10 box_np = np.array([bbox], dtype=np.float32)
11 box_t = torch.as_tensor(transform.apply_boxes(box_np, image_rgb.shape[:2]),
12                          device=device)
13 with torch.no_grad():
14     sparse_emb, dense_emb = sam.prompt_encoder(points=None, boxes=box_t, masks=
15                                                None)
16
17 # 3) Decodificar y llevar a tama ~{n}o original
18 low_res_masks, _ = sam.mask_decoder(
19     image_embeddings=image_embedding,
20     image_pe=sam.prompt_encoder.get_dense_pe(),
21     sparse_prompt_embeddings=sparse_emb,
22     dense_prompt_embeddings=dense_emb,
23     multimask_output=False,
24 )
25 up = sam.postprocess_masks(
26     low_res_masks,
27     input_size=transform.get_preprocess_shape(image_rgb.shape[0], image_rgb.shape
28     [1], 1024),
29     original_size=image_rgb.shape[:2],
30 )
31 pred_logits = up[0,0,...]
```

Listing 5.1: Encodear imagen, aplicar bbox y decodificar máscara

pérdida y métrica (BCE+IoU) con bucle de entrenamiento/validación.

```

1 gt_t = torch.as_tensor(gt_mask_np, device=device, dtype=torch.float32)
2 loss = F.binary_cross_entropy_with_logits(pred_logits, gt_t)
3 if is_train:
4     optimizer.zero_grad()
5     loss.backward()
6     optimizer.step()
7
```

```

8 with torch.no_grad():
9     pred_bin = (torch.sigmoid(pred_logits) > 0.5).detach().cpu().numpy().astype(
        np.uint8)
10     iou = compute_iou_bin(pred_bin, gt_mask_np)
11 total_loss += float(loss.detach().cpu().item()); total_iou += iou; count += 1

```

Listing 5.2: BCEWithLogits + IoU y actualización del decoder

Collate para lotes con número variable de instancias. El código de esta función se incluye en el Anexo A.10.

Overlay cualitativo (GT en rojo, pred en verde). El código completo para generar las visualizaciones se muestra en el Anexo A.11.

5.4 Conclusiones — Fine-tuning de SAM

En coherencia con el objetivo del capítulo —*valorar si el fine-tuning decoder-only (ViT-B, encoder congelado) mejora de forma justa a las líneas base no asistidas y seleccionar el modelo recomendado por escenario bajo restricciones realistas*— los resultados confirman una ganancia clara y acotada.

1. **Mejora frente a líneas base no asistidas.** El *fine-tuning* aporta mejoras consistentes sobre las líneas base *no asistidas* (imagen completa y Grid / AMG), alcanzando **mIoU** de hasta **0,96** en escenarios con solo jugadores y **0,92** cuando el balón es estable. Esta lectura cumple el objetivo de comparar *fair* contra referencias sin pistas de localización y evidencia beneficio real del ajuste del *decoder*.
2. **Generalización con datos limitados.** Con un *split* 80–20 sobre 27 imágenes se mantiene un **IoU global** $\sim 0,73$ y un **Dice** $\sim 0,83$, alineado con el objetivo de estimar *generalización* efectiva bajo tamaño de datos reducido y cómputo contenido.
3. **Sensibilidad a la variabilidad (Batch 3).** En escenas más variadas y exigentes, la mejora es **moderada** con 5 épocas, lo que indica que para cumplir el objetivo de robustez por escenario conviene **incrementar datos y/o épocas** y diseñar **augmentation específico del balón** (objeto pequeño/ocluido).
4. **Recomendación por escenario.** Para satisfacer el objetivo práctico de seleccionar modelo: (i) usar el modelo del **Batch 1** en aplicaciones centradas en *jugadores*; (ii) el del **Batch 2** cuando interese *jugadores + balón* en condiciones estables; (iii) el del **Batch 3** para *variabilidad alta*. Si prima la cobertura global, considerar un **ensemble** de los tres.
5. **Alcance y restricciones.** Las ganancias se obtienen con **decoder-only** (ViT-B, encoder congelado) y bajo **limitaciones realistas de tiempo/hardware**; el objetivo de un ajuste *responsable* queda así satisfecho, documentando el *trade-off* precisión–coste y evitando depender de pistas externas.

CAPÍTULO 6

Conclusiones

6.1 Resumen del trabajo realizado

A lo largo de esta memoria se ha estudiado el alcance real de *Segment Anything Model* (SAM) en fútbol, partiendo de evaluaciones sin entrenamiento y avanzando hacia el *fine-tuning* específico del dominio. El trabajo se ha estructurado en tres pilares: (i) establecer líneas base justas con distintos *prompts* de caja, (ii) incorporar un método no asistido por rejilla (Grid/AMG) y (iii) analizar el beneficio del *fine-tuning* (decoder-only) con distintos tamaños de datos y particiones.

Experimento 1 — BBoxes como línea base. Se compararon tres niveles de información espacial: *bbox* de imagen completa (sin guía), *bbox* global por clase (*Bbox global style*) y *bbox* específica por instancia (una por jugador/ balón). Con **ViT-B** y flujo de evaluación constante, el **bbox específico** fijó la mejor referencia (mIoU *jugadores*: $\approx 0,80$; *balón*: $\approx 0,81$). La **caja global por clase** funcionó bien en escenas compactas, pero degradó con dispersión; la **imagen completa** resultó ineficaz (mIoU ≈ 0). El análisis por lotes mostró cómo la geometría (densidad y solapes) afecta de forma diferencial a cada estrategia, y los ejemplos cualitativos (Batch 1–3) ilustraron aciertos y fallos típicos.

Experimento 2 — Grid/AMG frente a BBox. Para una comparación *justa* entre métodos *no asistidos*, se añadió Grid/AMG y se contrastó específicamente con *bbox* de imagen completa. **Grid/AMG** superó con claridad a la imagen completa (p. ej., mIoU *jugadores* $\approx 0,48$ vs $0,00$; *balón* $\approx 0,07$ vs $0,00$), aunque se mantiene lejos del techo asistido marcado por *bbox* específica por instancia. Se documentaron además los **compromisos de cómputo** necesarios (p. ej., *points_per_side* moderado, *downscale*, tope de máscaras por frame) para hacer viable la evaluación en hardware limitado; se recalca que no son “mejoras” del método, sino ajustes operativos.

Fine-tuning — Decoder-only (ViT-B). Se estudió el ajuste del *mask decoder* con el *encoder* congelado en tres escenarios: (i) Vinales v1 (13 img) vs v2 (27 img), mostrando convergencia estable y validación más robusta con mayor diversidad; (ii) un **split 80–20** (27 imágenes) con métricas IoU/Dice/precisión/*recall* (IoU global $\sim 0,73$; balón $\sim 0,85$; jugadores $\sim 0,70$), y (iii) **entrenamiento por lote** en Soccer Dataset (5 épocas/lote), con mIoU de $\sim 0,96$ (Batch 1: sólo jugadores), $\sim 0,92$ (Batch 2: jugadores+balón) y $\sim 0,82$ (Batch 3: variabilidad y oclusiones). Estas cifras sitúan al modelo ajustado *por encima* de las líneas base no asistidas y próximo al techo asistido en escenarios favorables (jugadores grandes/compactos), manteniendo margen de mejora en escenas amplias y con balón pequeño/ocluido.

Aportaciones prácticas y reproducibilidad. Se desarrollaron *pipelines* de evaluación comparables (*misma métrica, mismo flujo*), *scripts* ilustrativos (selección multimáscara, comparación “sólo cambia el *bbox*”, *grid* densas con *batcheo* de puntos), y visualizaciones cualitativas homogéneas (paleta: GT en verde/rojo; SAM en azul/amarillo). Se documentaron supuestos, límites y decisiones de ingeniería (tiempo/memoria), y se ofrecieron recomendaciones de uso por escenario.

6.2 Objetivos conseguidos

A la luz de los resultados y en relación directa con los objetivos planteados (datos, evaluación de SAM en *zero-shot* y *fine-tuning*), se consideran alcanzados los siguientes hitos:

- **Datasets preparados para una evaluación justa.** Se seleccionó y acondicionó *Soccer-Panoptic* en un formato compatible con SAM (máscaras y *bboxes* por clase/instancia) y se creó el conjunto propio *Vinalesa FC* con segmentación manual. Con ambos se habilitó la derivación sistemática de los *prompts* comparados (imagen completa, caja global por clase y caja específica por instancia) y la partición por lotes de dificultad, cumpliendo el objetivo de disponer de datos adecuados, cercanos al dominio y reproducibles.
- **SAM en *zero-shot*: líneas base y “fairness” establecidos.** Se fijó una referencia cuantitativa y cualitativa con tres *prompts* de caja bajo idéntico protocolo de evaluación. Se demostró la ineficacia de la imagen completa ($mIoU \approx 0$) y la superioridad de la *bbox* específica por instancia como techo asistido (**mIoU jugadores: 0,800; balón: 0,805**), mientras que la caja global por clase (*Bbox_global*) resultó competitiva en escenas compactas y sensible a la dispersión. En métodos **no asistidos**, se incorporó *Grid/AMG* y se comparó con **bbox de imagen completa** (ambos sin guía), evidenciando mejoras claras de *Grid/AMG* (**jugadores: 0,476 vs 0,000; balón: 0,071 vs 0,000**) y su distancia respecto a *bboxes* asistidos. Con ello, el objetivo de una **línea base justa** queda cumplido.
- **Análisis cualitativo representativo.** Se añadieron ejemplos por lote (Batch 1–3), incluyendo un **caso difícil** donde *ningún método* capta bien el frame, para contextualizar las caídas de $mIoU$ y explicar con imágenes las diferencias entre enfoques (adherencia a contornos, fugas al fondo, fusión de instancias).
- **Fine-tuning con restricciones realistas.** Se ejecutó **decoder-only** (ViT-B, encoder congelado) en los escenarios planificados (versiones de *Vinalesa*, 80–20 y por lote), mostrando **ganancias consistentes** sobre las líneas base no asistidas y manteniendo resultados estables con datos limitados (**mIoU hasta 0,96** con sólo jugadores, **0,92** con balón estable; en 80–20 con 27 imágenes: **IoU $\sim 0,73$ y Dice $\sim 0,83$**). Se proponen **modelos recomendados por escenario** (sólo jugadores; jugadores+balón estable; variabilidad alta) y, cuando prima la cobertura, la opción de **ensemble**. El objetivo de evaluar un *fine-tuning* responsable y útil queda satisfecho.
- **Recomendaciones operativas y reproducibilidad.** Se ofreció una guía de uso por escenario y se documentaron **compromisos cómputo–tiempo** (rejilla, *downscale*, tope de máscaras) y su lectura correcta (no confundir con mejoras algorítmicas). Se incluyeron fragmentos de código *autoexplicativos* que muestran las decisiones clave (multimáscara, selección por IoU, comparación “sólo cambia el *bbox*”, *grid* batcheada) y *overlays* homogéneos GT/predicción, cumpliendo el objetivo de dejar una base replicable.

Asimismo, se identifican objetivos *parcialmente* alcanzados o con margen de mejora:

- **Robustez al balón pequeño/ocluido.** Aunque el *fine-tuning* mejora el balón en 80–20 ($\sim 0,85$ IoU), en escenas muy amplias y con oclusiones la precisión sigue siendo sensible; conviene aumentar datos/épocas y añadir *augmentation* específico del balón.
- **Escenas amplias con alta variabilidad (Batch 3).** El modelo ajustado mejora, pero persisten fallos en presencia de múltiples jugadores, solapes y líneas de campo; la separación de instancias sigue siendo el reto principal.
- **Coste de Grid/AMG.** Se mostró su utilidad frente a imagen completa, pero con coste mayor y sensibilidad a hiperparámetros; queda margen para **rejillas adaptativas** y filtrado eficiente de propuestas.

Resumen práctico de elección por escenario.

- **Con guía fiable por objeto:** usar *BBoxes específicas* con selección multimáscara (mejor precisión y separación).
- **Con una guía por clase y escena compacta:** Bbox global (simple y razonable; pierde con dispersión).
- **Sin guía (no asistido):** preferir **Grid/AMG** a imagen completa; ajustar *points_per_side*, *downscale* y tope de máscaras según recursos.
- **Con entrenamiento disponible:** **fine-tuning decoder-only** (ViT-B) por lote/dominio; considerar *ensemble* si se cubren escenarios muy distintos.

6.3 Trabajo futuro

A partir de los hallazgos, se proponen líneas de mejora realistas y ordenadas por impacto/viabilidad:

A. Métodos no asistidos y propuestas

- **Rejillas adaptativas y multiescala:** densidad de puntos mayor en regiones de alta entropía (bordes, movimientos), menor en fondo; *tiling* por pirámide de escalas para el balón.
- **Filtrado temprano de propuestas:** métricas de estabilidad/área y *top-k* por región; supresión no máxima entre máscaras solapadas para evitar *blobs* redundantes.
- **Pistas no derivadas de GT:** *priors* genéricos (detección ligera entrenada en dominios abiertos) para recortar la búsqueda sin incurrir en *leakage*.
- **Temporalidad:** fusión entre frames mediante *tracking* ligero (p.ej., *IoU tracking/optical flow*) para estabilizar contornos y mejorar el balón.

B. Fine-tuning y datos

- **Más datos y variedad:** ampliar Vinalesa y Soccer con escenas amplias, oclusiones y cámaras distintas; *split* estratificado por complejidad.
- **Pérdidas y *augmentation* orientadas a objetos pequeños:** ponderación por clase/área, *focal loss* y *copy-paste* del balón; *color jitter*, recortes, *motion blur*.
- **Descongelado progresivo:** explorar *unfreezing* parcial del *image encoder* (capas altas) tras estabilizar el *decoder*.
- **Ensembles y especialización:** modelos por lote/condición y combinación por votación/IoU; seleccionar modelo según metadatos de escena (zoom, número de jugadores).

C. Evaluación, ingeniería y despliegue

- **Métricas ampliadas:** además de mIoU y Dice, reportar AP por instancia, PQ y métricas temporales (consistencia y fragmentación).
- **Optimización:** inferencia en *mixed precision*, *batching* de *prompts*, exportación a ONNX/TensorRT; control del presupuesto (ms/frame).
- **Interfaz para campo:** prototipo simple para entrenadores (carga de vídeo, selección de método, resúmenes por jugada) con *overlays* y métricas básicas.
- **Privacidad y ética:** anonimización selectiva (rostros, dorsales) y políticas de retención, especialmente en fútbol base.

D. Validación externa

- **Cross-dataset:** evaluar en partidos/equipos distintos a los de entrenamiento; medir transferencia.
- **Estudios de usuario:** sesiones con cuerpo técnico para valorar utilidad percibida, tiempos ahorrados y *insights* tácticos.

En conjunto, el trabajo establece una base sólida y honesta para el uso de SAM en fútbol: clarifica el papel de las *pistas* espaciales, cuantifica el salto del *fine-tuning* con datos modestos y ofrece una ruta de mejora pragmática hacia sistemas más robustos, útiles y accesibles para clubes y entrenadores.

Orientación a cliente y validación por terceros

Este TFG se ha desarrollado como **prototipo** orientado a *prueba de concepto*. No hay, a fecha de entrega, un cliente con contrato ni despliegue operativo. Como siguiente paso, proponemos una **validación externa ligera** con terceros: (1) piloto con un cuerpo técnico local (p. ej., Vinalesa FC) para recoger feedback sobre utilidad percibida (métricas, visualizaciones, tiempos); (2) contraste con material público de *benchmarks* (p. ej., SoccerNet) para contextualizar resultados; (3) checklist de requisitos para un uso real (privacidad, rendimiento por frame, estabilidad y soporte). Este ciclo permitiría decidir si evolucionar el prototipo a herramienta interna o integrarlo en plataformas existentes.

Bibliografía

- [1] Alexander Kirillov, et al. Segment Anything. arXiv, 2023. <https://arxiv.org/abs/2304.02643>.
- [2] GeeksforGeeks. Creating Python Virtual Environment in Windows and Linux. Disponible en: <https://www.geeksforgeeks.org/creating-python-virtual-environment-windows-linux/>
- [3] Liquid Web. How To Set Up a Python Virtual Environment on Windows 10. Disponible en: <https://www.liquidweb.com/blog/how-to-setup-a-python-virtual-environment-on-windows-10/>
- [4] Roboflow Blog. What is Segment Anything 2 (SAM 2)?. Disponible en: <https://blog.roboflow.com/what-is-segment-anything-2/>
- [5] MathWorks. Getting Started with Segment Anything Model. Disponible en: <https://www.mathworks.com/help/images/getting-started-with-segment-anything-model.html>
- [6] Maucher, M. Segment Anything Model (SAM). Disponible en: <https://maucher.pages.mi.hdm-stuttgart.de/orbook/deeplearning/SAM.html>
- [7] Facebook Research. Segment Anything Model (SAM). Disponible en: <https://github.com/facebookresearch/segment-anything>
- [8] Roboflow Blog. How to Use the Segment Anything Model (SAM). Disponible en: <https://blog.roboflow.com/how-to-use-segment-anything-model-sam/>
- [9] ArXiv. Comprehensive Analysis of Segment Anything Model (SAM) in Medical Image Segmentation. Disponible en: <https://arxiv.org/html/2304.14660v6>
- [10] Kaggle. Soccer - Panoptic Segmentation Dataset. Disponible en: <https://www.kaggle.com/datasets/quantigoai/soccer-dataset>
- [11] Baeldung on Linux. Extracting Frames From a Video Using FFmpeg. Disponible en: <https://www.baeldung.com/linux/ffmpeg-extract-video-frames>
- [12] Gumlet Community. How to extract all the frames from a video using FFmpeg?. Disponible en: <https://community.gumlet.com/t/how-to-extract-all-the-frames-from-a-video-using-ffmpeg/729>
- [13] DeepSport Dataset, <https://paperswithcode.com/dataset/deepsport-dataset>.
- [14] Hyper.ai DeepSport Overview, <https://hyper.ai/deepsport>.
- [15] Autor et al., “Real-time CNN-based segmentation architecture for ball detection. . .”, año no especificado.

- [16] DeepSport GitHub Repository, <https://github.com/gabriel-vanzandycke/deepsport>.
- [17] Dataset Ninja: DeepSportRadar-v2 statistics, <https://datasetninja.com/deepsport>.
- [18] Ar5iv record: DeepSportRadar-v1 annotations, enlace no especificado.
- [19] Facebook Research. Segment Anything Model (SAM2). Consultat a <https://github.com/facebookresearch/sam2>.
- [20] YouTube. Segment Anything Model - Demo. Consultat a <https://www.youtube.com/watch?v=OhxJkqD1vuE>.
- [21] Meta AI Research. Segment Anything Model. Consultat a <https://ai.meta.com>.
- [22] Ultralytics Docs. *How to annotate and build datasets*. Disponible en: <https://docs.ultralytics.com>
- [23] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [24] Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [25] Dosovitskiy, A., Beyer, L., Kolesnikov, A., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*.
- [26] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [27] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [28] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [29] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*.
- [30] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation (DeepLabv3+). In *ECCV*.
- [31] Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *IEEE International Conference on Image Processing (ICIP)*.
- [32] Lin, T.-Y., Maire, M., Belongie, S., et al. (2014). Microsoft COCO: Common Objects in Context. In *ECCV*.
- [33] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

- [34] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2), 303–338.
- [35] Kirillov, A., He, K., Girshick, R., Rother, C., & Dollár, P. (2019). Panoptic Segmentation. In *CVPR*.
- [36] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*.
- [37] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. In *ECCV*.
- [38] Boccignone, G., Cioppa, A., Girdhar, R., et al. (2023). SoccerNet: A Scalable Dataset for Video Understanding in Soccer. *International Journal of Computer Vision (IJCV)*.
- [39] SuperAnnotate. *Why quality annotations matter in computer vision*. Disponible en: <https://www.superannotate.com>
- [40] Husein, A. et al. *Automatic Detection and Segmentation of Soccer Players using Mask R-CNN*. 2023. Disponible en: https://www.researchgate.net/publication/376699282_Detecting_and_Tracking_Player_in_Football_Videos_Using_Two-Stage_Mask_R-CNN_Approach
- [41] PadelTracker100 Dataset. *A Large-Scale Dataset for Padel Player and Ball Tracking*. 2025. Disponible en: <https://zenodo.org/records/14653706>

APÉNDICE A

Anexo de Listings

A.1 Estructura práctica del dataset

```
1 soccer_panoptic/  
2   images/seq{1..N}/frame_XXXXX.jpg  
3   panoptic/seq{1..N}/frame_XXXXX.png # máscara panóptica (ID por píxel)  
4   meta/seq{1..N}/frame_XXXXX.json # mapping ID -> {class, instance_id}  
5   masks_class/{players,ball}/seq*/frame_XXXXX.png  
6   masks_inst/players/seq*/frame_XXXXX_idYYYYY.png  
7   bboxes/{full|class_global|instances}/seq*/frame_XXXXX.json
```

Listing A.1: Estructura práctica del dataset

A.2 Derivación de máscaras y *prompts* desde panoptic

```
1 pn = cv2.imread(panoptic_png, cv2.IMREAD_UNCHANGED) # ID de instancia por píxel  
2 meta = json.load(open(panoptic_json)) # [{id, class_name, ...}, ...]  
3  
4 # Binarias por clase  
5 players_ids = [o["id"] for o in meta if o["class_name"]=="players"]  
6 ball_ids = [o["id"] for o in meta if o["class_name"]=="ball"]  
7 players_mask = np.isin(pn, players_ids)  
8 ball_mask = np.isin(pn, ball_ids)  
9  
10 # BBoxes específicas por instancia (jugadores)  
11 for inst_id in players_ids:  
12     inst_mask = (pn == inst_id)  
13     ys, xs = np.where(inst_mask)  
14     if ys.size == 0:  
15         continue  
16     x0, y0, x1, y1 = xs.min(), ys.min(), xs.max(), ys.max()  
17     save_bbox_json(inst_id, [int(x0), int(y0), int(x1), int(y1)])
```

Listing A.2: Derivación de máscaras por clase e instancias desde panoptic.

A.3 Funciones auxiliares de evaluación y visualización

```
1 def eval_with_bbox(predictor, img_rgb, gt, use_full_bbox=False):  
2     h, w = img_rgb.shape[:2]  
3     predictor.set_image(img_rgb)
```

```

4     bbox = np.array([0,0,w-1,h-1], np.float32) if use_full_bbox else
        get_mask_bbox(gt)
5     masks, scores, _ = predictor.predict(box=bbox, multimask_output=True)
6     return max(iou_bool(m, gt) for m in masks)

```

Listing A.3: Mismo método de selección; solo cambia el bbox

```

1 def create_overlay_visualization(image, gt_players, pred_players, gt_ball,
    pred_ball, alpha=0.6):
2     overlay = image.copy().astype(np.float32)
3     players_color, ball_color = (0,255,0), (0,0,255)
4     overlay[gt_players] = players_color
5     overlay[gt_ball] = ball_color
6     if pred_players.any():
7         only_pred = pred_players & ~gt_players
8         overlay[only_pred] = (1-alpha)*overlay[only_pred] + alpha*np.array(
            players_color)
9     if pred_ball.any():
10        only_pred = pred_ball & ~gt_ball
11        overlay[only_pred] = (1-alpha)*overlay[only_pred] + alpha*np.array(
            ball_color)
12    return np.clip(overlay, 0, 255).astype(np.uint8)

```

Listing A.4: Overlay cualitativo: GT sólido, predicción transparente

A.4 Comparación justa de métodos (BBox)

```

1 def evaluate_frame_fair_comparison(predictor, img_rgb, gt, use_full_bbox=False):
2     h, w = img_rgb.shape[:2]
3     predictor.set_image(img_rgb)
4     bbox = np.array([0,0,w-1,h-1], np.float32) if use_full_bbox else
        get_mask_bbox(gt.astype(np.uint8))
5     masks, scores, _ = predictor.predict(box=bbox, multimask_output=True)
6     return max(iou_bool(m.astype(bool), gt) for m in masks)

```

Listing A.5: Solo cambia el bbox: imagen completa vs bbox global por clase

A.5 Orquestación por lotes

```

1 def run_for_batch(batch_dir, model, epochs, device, seed, out_suffix,
    batch_script):
2     out_dir = batch_dir / out_suffix
3     ckpt = find_checkpoint(batch_dir, model)
4     cmd = [
5         sys.executable, str(batch_script),
6         "--batch_dir", str(batch_dir),
7         "--model", model, "--checkpoint", ckpt,
8         "--epochs", str(epochs), "--device", device, "--seed", str(seed),
9         "--out_dir", str(out_dir),
10    ]
11    print(f">>> Ejecutando: {' '.join(cmd)}")
12    result = subprocess.run(cmd)
13    if result.returncode != 0:
14        raise SystemExit(f"Entrenamiento falló para {batch_dir} (código {
            result.returncode})")

```

Listing A.6: Lanzar el entrenamiento por lote reutilizando el mismo script de batch

A.6 Descubrimiento y partición por batch

```

1 def discover_samples_by_batch(base_dir, batch_name):
2     samples = []
3     frames = sorted((Path(base_dir)/batch_name/"frames").glob("*.png"))
4     players_dir = Path(base_dir)/batch_name/"masks_players"
5     ball_dir = Path(base_dir)/batch_name/"masks_ball"
6     for f in frames:
7         pm = players_dir/f"mask_{f.name}"
8         bm = ball_dir/f"mask_{f.name}"
9         if pm.exists(): samples.append(Sample(str(f), str(pm), "players",
10                                             batch_name))
11         if bm.exists(): samples.append(Sample(str(f), str(bm), "ball", batch_name)
12                                         )
13     return samples
14
15 def split_samples(samples, seed, ratios=(0.7,0.15,0.15)):
16     n = len(samples); idx = list(range(n))
17     random.Random(seed).shuffle(idx)
18     n_tr = int(ratios[0]*n); n_va = int(ratios[1]*n)
19     to = lambda ids: [samples[i] for i in ids]
20     return to(idx[:n_tr]), to(idx[n_tr:n_tr+n_va]), to(idx[n_tr+n_va:])

```

Listing A.7: Creación del dataset por lote y partición reproducible

A.7 Fine-tuning decoder-only

```

1 sam = sam_model_registry[model_name] (checkpoint=f"checkpoints/sam_{model_name}.
2    .pth").to(device)
3 sam.train()
4 for p in sam.parameters(): # congelar todo
5     p.requires_grad = False
6 for p in sam.mask_decoder.parameters(): # entrenar solo decoder
7     p.requires_grad = True
8 optimizer = optim.Adam(sam.mask_decoder.parameters(), lr=1e-5, weight_decay=1e-4)

```

Listing A.8: Congelar encoder y preparar optimizador del decoder

A.8 Comparación de métodos con distintos BBoxes

```

1 def get_mask_bbox(mask):
2     ys, xs = np.where(mask > 0)
3     if ys.size == 0: return None
4     return np.array([xs.min(), ys.min(), xs.max(), ys.max()], np.float32)
5
6 if method == "sam_eval":
7     bbox = get_mask_bbox(gt_mask)
8 elif method in ("fullbbox", "overlap"):
9     h, w = img_rgb.shape[:2]
10    bbox = np.array([0,0,w-1,h-1], np.float32)

```

```
11 else:
12     raise ValueError("Metodo bbox desconocido")
```

Listing A.9: Definir el bbox según el método (bbox-global/fullbbox/overlap)

A.9 Collate para DataLoader

```
1 def collate_fn(batch):
2     all_data = []
3     for sample_list in batch:
4         all_data.extend(sample_list) # cada imagen puede aportar 0..N instancias
5     return all_data
```

Listing A.10: Unir listas de muestras heterogéneas en cada batch del DataLoader

A.10 Overlay cualitativo

```
1 def make_overlay(image_bgr, gt, pred, alpha=0.5):
2     ov = image_bgr.copy().astype(np.float32)
3     ov[gt.astype(bool)] = (0, 0, 255) # GT
4     ov[pred.astype(bool)] = (0, 255, 0) # Pred
5     return cv2.addWeighted(image_bgr.astype(np.float32), 1-alpha, ov, alpha, 0).
        astype(np.uint8)
```

Listing A.11: Guardar visualizaciones para inspección cualitativa

APÉNDICE B

Configuración del sistema

Este capítulo describe el entorno hardware y software utilizado, el proceso de inicialización del proyecto y la identificación de dispositivos. El objetivo es facilitar la reproducibilidad de los experimentos y dejar constancia de las limitaciones prácticas (tiempo/hardware) bajo las que se ejecutaron.

B.1 Entorno hardware y software

Hardware

- **Equipo:** Portátil.
- **CPU:** Intel 11th Gen Core i5-11400H @ 2.70 GHz (6 núcleos / 12 hilos).
- **RAM:** 16 GB.
- **GPU:** NVIDIA GeForce RTX 3050 Laptop GPU (VRAM 4 GB).
- **Almacenamiento:** (SSD/HDD) *espacio libre aproximado:* 2 GB.

Sistema operativo y controladores

- **Sistema operativo principal:** Windows 11.
- **Subsistema Linux:** WSL (**kernel reportado:** 5.15.167.4-microsoft-standard-WSL2, Ubuntu 24.04.2 LTS (noble)).
- **Driver NVIDIA (Windows):** 572.60.
- **CUDA en el sistema (nvcc):** 12.0 (V12.0.140).
- **CUDA runtime de PyTorch:** 12.8 (2.8.0+cu128).
- **cuDNN:** N/D.

Entorno Python y librerías

- **Gestor de entornos:** venv.
- **Python:** 3.12.3.
- **PyTorch:** 2.8.0+cu128; `torch.cuda.is_available = True`.

- **Paquetes clave:** opencv-python 4.9.0, numpy 1.26.4, matplotlib 3.10.5, segment-anything 1.0 (repo facebookresearch/segment-anything, main), tqdm 4.67.1, pycocotools 2.0.10, scikit-image (no instalado).
- **torchvision:** 0.23.0+cu128.
- **ffmpeg:** 6.1.1-3ubuntu5.

Modelos y *checkpoints* de SAM

- **Backbones usados:** ViT-B y ViT-H.
- **Origen de los pesos:** pesos oficiales/locales (colocados en la carpeta de pesos del proyecto).
- ***Fine-tuning: decoder-only*** (encoders congelados).
Implementación: los scripts `finetuning_by_batch.py`, `finetuning_comparative_robust.py` y `fine_tuning_vinales.py` desactivan `grad` en `image_encoder` y `prompt_encoder` y optimizan únicamente `mask_decoder`.
- **Semillas (*seeds*):** valor por defecto 42 (CLI `-seed`), aplicada en `random`, `numpy`, `torch` y `torch.cuda`; utilizada tanto en *splits* como en entrenamiento.

B.2 Fase de inicialización

Creación del entorno y dependencias

```

1 # Crear entorno (Python 3.12)
2 python -m venv .venv
3 # Activar entorno
4 source .venv/bin/activate # Linux/WSL
5 # .\.venv\Scripts\activate # Windows PowerShell
6
7 # Actualizar pip y wheel
8 python -m pip install --upgrade pip wheel
9
10 # Instalar dependencias principales (ejemplo minimo coherente con el proyecto)
11 pip install torch==2.8.0+cu128 torchvision==0.23.0+cu128 --index-url https://
    download.pytorch.org/whl/cu128
12 pip install opencv-python==4.9.0 numpy==1.26.4 matplotlib==3.10.5
13 pip install segment-anything==1.0 tqdm==4.67.1 pycocotools==2.0.10
14 # (scikit-image no instalado)

```

Listing B.1: Creación y activación del entorno `venv` e instalación con `pip`

Descarga de *checkpoints* y preparación de datos

```

1 # (Ubicar los .pth de ViT-B y ViT-H en la carpeta de pesos del proyecto)
2 # sam_vit_b.pth, sam_vit_h.pth
3
4 # Preparacion de datasets:
5 # - Descompresion de archivos
6 # - Extraccion de frames con ffmpeg
7 # - Obtencion de mascaras y asociacion frame-mascara

```

```

8
9 # Ejemplo de extraccion de frames (ajustar rutas/valores):
10 ffmpeg -i <video_entrada.mp4> -vf fps=<FPS> data/frames/%06d.png

```

Listing B.2: Descarga de pesos de SAM y preparación básica de datos

Semillas y reproducibilidad

```

1 import os, random, numpy as np, torch
2
3 SEED = 42
4 random.seed(SEED)
5 np.random.seed(SEED)
6 torch.manual_seed(SEED)
7 if torch.cuda.is_available():
8     torch.cuda.manual_seed_all(SEED)
9 # Opcional: forzar determinismo (no especificado por el usuario)
10 # torch.backends.cudnn.deterministic = True
11 # torch.backends.cudnn.benchmark = False

```

Listing B.3: Inicialización de semillas en Python/Numpy/PyTorch

Lanzamiento de experimentos

La ejecución se realiza lanzando los scripts del proyecto con los parámetros correspondientes (p.ej., rutas de datos, selección de *backbone*, activación de *multimask_output*, límites de Grid/AMG, etc.).

Ejemplos: `eval_with_visualization.py`, `eval_sam_eval_style.py`, `sam_grid.py`, `finetuning_by_batch.py`, `finetuning_comparative_robust.py`.

B.3 Identificación de dispositivos

GPU y *driver* (resumen de `nvidia-smi`)

```

1 NVIDIA-SMI 570.124.03 Driver Version: 572.60 CUDA Version: 12.8
2 GPU Name Temp Pwr Mem-Usage
3 0 NVIDIA GeForce RTX 3050 ... 57C 8W 694MiB / 4096MiB

```

SO y kernel

```

1 uname -a
2 Linux DESKTOP-1RH39TE 5.15.167.4-microsoft-standard-WSL2 #1 SMP Tue Nov 5
   00:21:55 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux
3
4 lsb_release -a
5 Distributor ID: Ubuntu
6 Description: Ubuntu 24.04.2 LTS
7 Release: 24.04
8 Codename: noble

```

Python, PyTorch y CUDA de Torch

```
1 python -V
2 # Python 3.12.3
3
4 # Resumen reportado:
5 # PyTorch: 2.8.0+cu128
6 # Torch CUDA (torch.version.cuda): 12.8
7 # CUDA disponible (torch.cuda.is_available): si
8 # torchvision: 0.23.0+cu128
```

ffmpeg y paquetes

```
1 ffmpeg -version
2 # ffmpeg version 6.1.1-3ubuntu5
3
4 # Paquetes clave (versiones reportadas):
5 # opencv-python: 4.9.0
6 # numpy: 1.26.4
7 # matplotlib: 3.10.5
8 # segment-anything: 1.0
9 # tqdm: 4.67.1
10 # pycocotools: 2.0.10
11 # scikit-image: no instalado
```

B.4 Notas y limitaciones

- **WSL 1/2:** aunque inicialmente se indicó WSL 1, la salida de `uname` identifica WSL 2. Se toma esta última como referencia del entorno de ejecución.
- **CUDA toolkit vs PyTorch CUDA:** el sistema dispone de `nvcc` 12.0; PyTorch se ejecuta con CUDA 12.8 (+cu128). El *driver* 572.60 proporciona compatibilidad con 12.8.
- **Espacio en disco:** el espacio libre reportado es 2 GB, lo que puede resultar ajustado para *checkpoints*, *caches* y resultados intermedios (se recomienda ampliar si es posible).
- **Determinismo:** se fija `seed=42` en `random`/`numpy`/`torch`; no se ha especificado el uso de flags de determinismo de cuDNN.



UNIVERSITAT
POLITÀCNICA
DE VALÈNCIA



ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. Fin de la pobreza.				x
ODS 2. Hambre cero.				x
ODS 3. Salud y bienestar.	x			
ODS 4. Educación de calidad.		x		
ODS 5. Igualdad de género.				x
ODS 6. Agua limpia y saneamiento.				x
ODS 7. Energía asequible y no contaminante.				x
ODS 8. Trabajo decente y crecimiento económico.				x
ODS 9. Industria, innovación e infraestructuras.	x			
ODS 10. Reducción de las desigualdades.	x			



Escola Tècnica
Superior d'Enginyeria
Informàtica

ETS Enginyeria Informàtica
Camí de Vera, s/n. 46022. València
T +34 963 877 210
F +34 963 877 219
etsinf@upvnet.upv.es - www.inf.upv.es





UNIVERSITAT
POLITÀCNICA
DE VALÈNCIA



ODS 11. Ciudades y comunidades sostenibles.				X
ODS 12. Producción y consumo responsables.				X
ODS 13. Acción por el clima.				X
ODS 14. Vida submarina.				X
ODS 15. Vida de ecosistemas terrestres.				X
ODS 16. Paz, justicia e instituciones sólidas.				X
ODS 17. Alianzas para lograr objetivos.				X

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Este Trabajo de Fin de Grado conecta de forma clara con varios de los Objetivos de Desarrollo Sostenible de la Agenda 2030. Aunque el proyecto nace en un ámbito muy concreto, el fútbol, sus implicaciones pueden ir mucho más allá y aportar a cuestiones que afectan directamente a las personas y a la sociedad.



Escola Tècnica
Superior d'Enginyeria
Informàtica

ETS Enginyeria Informàtica
Camí de Vera, s/n. 46022. València
T +34 963 877 210
F +34 963 877 219
etsinf@upvnet.upv.es - www.inf.upv.es



El ODS 3, que se centra en la Salut y el Bienestar, es sin duda el más evidente. La tecnología que se ha desarrollado, impulsada por la inteligencia artificial, no solo puede analizar jugadas o estadísticas, sino que también tiene el potencial de cuidar a los propios deportistas. Imaginemos poder identificar patrones de movimiento que nos avisen de una posible lesión antes de que suceda: esto significaría menos tiempo fuera de juego, entrenamientos más seguros y, en última instancia, una práctica deportiva más saludable. La salud abarca no solo lo físico, sino también lo mental, y sentirse protegido ante riesgos genera una sensación de tranquilidad y confianza.

Por otro lado, el ODS 4, que se refiere a la Educación de Calidad, se manifiesta en la importancia del acceso a estas tecnologías como herramientas de aprendizaje. Cualquiera puede experimentar con ellas, sin importar los recursos que tenga, ya que son modelos abiertos y gratuitos. Esto permite que estudiantes, docentes o incluso personas curiosas aprendan sobre inteligencia artificial, deporte o análisis de datos sin que el dinero sea un obstáculo. Es un gran paso hacia una educación más inclusiva y accesible.

El ODS 9, que se centra en la Industria, Innovación e Infraestructuras, está profundamente conectado con la esencia del proyecto. Trabajar con modelos tan recientes nos coloca en la vanguardia de la innovación, lo que a su vez impulsa nuevas ideas y aplicaciones. Lo que hoy se utiliza en el fútbol, mañana podría ser útil en otros campos, desde la medicina deportiva hasta el entretenimiento. Innovar no se trata solo de crear algo nuevo; también implica demostrar que la investigación académica puede ofrecer soluciones valiosas para la sociedad.

Por otro lado, el ODS 10, que aborda la Reducción de las Desigualdades, se vincula con la accesibilidad y gratuidad de la tecnología. En el ámbito deportivo, existen grandes disparidades entre los clubes grandes y los más pequeños. Mientras que algunos disfrutan de recursos casi ilimitados, otros apenas tienen lo esencial. Una herramienta abierta como esta puede ayudar a cerrar esa brecha, permitiendo que todos accedan a un análisis avanzado. Y esto no se limita al deporte; también se aplica en la educación y la investigación, donde cualquier persona, sin importar su origen, puede aprender y beneficiarse de estas tecnologías.

En resumen, este proyecto ilustra cómo la tecnología, cuando es abierta y accesible, puede contribuir a un mundo más saludable, justo, innovador y lleno de oportunidades de aprendizaje. El fútbol es solo el punto de partida, pero el verdadero valor radica en las puertas que se abren para las personas.