uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Informática
2022-2023

Trabajo Fin de Grado

"Redes neuronales para la detección y análisis de la trayectoria de la pelota en partidos de pádel"

Samuel Reyes Sanz

Tutora

Inés María Galván León

Leganés, 2023

RESUMEN

Este trabajo se centra en la adaptación y entrenamiento de una Red Neuronal Convolucional para la detección de pelotas en vídeos de pádel, una tarea para la cual no existía una solución previa basada en Inteligencia Artificial. Para ello, se generó un conjunto de datos a partir de vídeos del torneo World Padel Tour, que se desglosaron en fotogramas y se etiquetaron manualmente.

Partiendo del código de la red TrackNetV2, se aplicó la transferencia de aprendizaje utilizando un modelo previamente entrenado con vídeos de tenis. Este enfoque mostró una mejora significativa en la detección de pelotas, obteniendo hasta un 50% de mejora en algunos experimentos. Se exploraron y evaluaron también otras modificaciones en la red, como la incorporación de distintas funciones de pérdida, técnicas de dropout y skip-connections, y la introducción de un proceso de validación.

El resultado final es un modelo que demuestra un rendimiento eficiente en la detección de pelotas, incluso en situaciones difíciles donde las pelotas están parcialmente ocultas. Este modelo puede operar en tiempo real con una tarjeta gráfica de gama media, lo que representa un avance importante para la aplicación en situaciones en vivo y abre la puerta a futuras aplicaciones en la generación automática de estadísticas a partir de vídeos de pádel.

Palabras clave: redes neuronales convolucionales; transferencia de aprendizaje; segmentación de imágenes; pádel; función de pérdida.

ABSTRACT

This work focuses on the adaptation and training of a Convolutional Neural Network for ball detection in padel videos, a task for which there was no previous solution based on Artificial Intelligence. To this end, a dataset was generated from videos of the World Padel Tour tournament, which were broken down into frames and manually tagged.

Building on the code of the TrackNetV2 network, transfer learning was applied using a model previously trained with tennis videos. This approach showed a significant improvement in ball detection, achieving up to a 50% improvement in some experiments. Several modifications to the network were also explored and evaluated, such as the incorporation of different loss functions, dropout techniques and skip-connections, and the introduction of a validation process.

The outcome is a model that demonstrates efficient performance in ball detection, even in challenging situations where the balls are partially hidden. This model can operate in real-time with a mid-range graphics card, representing a significant advancement for application in live situations and paving the way for future applications in the automatic generation of statistics from paddle tennis videos.

Keywords: convolutional neural networks; transfer learning; image segmentation; padel; loss function.

DEDICATORIA

Este trabajo está dedicado a todas las personas que han estado a mi lado en esta etapa tan importante de mi educación. Gracias a mi familia y amigos por acompañarme y apoyarme durante estos años. Agradezco a todos los profesores que me han transmitido su motivación por la informática, especialmente a Inés, por su orientación y asistencia en este trabajo.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

LISTA DE ABREVIATURAS

WPT	*World Padel Tour*
RAM	*Random Access Memory*
HHD	*Hard Drive Disk*
SSD	*Solid State Disk*
RGB	*Red Green Blue*
HSV	*Hue Saturation Value*
CNN	*Convolutional Neural Networks*
RNN	*Recurrent Neural Networks*
LSTM	*Long Short-Term Memory*
GAN	*Generative Adversarial Networks*
IA	Inteligencia Artificial
GPU	*Graphical Processing Unit*
GNU	*General Public License*
ReLU	*Rectified Lineal Unit*
CUDA	*Compute Unified Device Architecture*
cuDNN	*NVIDIA CUDA Deep Neural Network library*
ICPAI	*International Conference on Pervasive Artificial Intelligence*
MISO	*Multiple Input, Single Output*
MIMO	*Multiple Input, Multiple Output*
BCE	*Binary Cross Entropy*
WBCE	*Weighted Binary Cross Entropy*

1. INTRODUCCIÓN

Este trabajo tiene como objetivo principal abordar la compleja tarea de detección de objetos pequeños y en rápido movimiento en vídeos, a través de la aplicación de técnicas avanzadas de Inteligencia Artificial. En el ámbito específico de este estudio, el objeto de interés es la detección de pelotas en videos de partidos de pádel. La tarea implica el uso Redes Neuronales Convolucionales (CNN), una de las herramientas más prometedoras en el campo de la visión por computadora, para analizar secuencias de vídeo y rastrear con precisión el movimiento de la pelota. Este enfoque, si bien es técnicamente desafiante debido al pequeño tamaño de la pelota, su alta velocidad y la variabilidad de las condiciones de iluminación y de fondo, tiene un gran potencial para contribuir significativamente al análisis del juego en el deporte del pádel. A través de este análisis detallado, se puede ofrecer una mejor comprensión de la dinámica del juego, proporcionar datos valiosos para la mejora del rendimiento de los jugadores y eventualmente ayudar a democratizar el acceso a las estadísticas de juego, que hasta ahora han sido posibles sólo con el uso de costosos sistemas de seguimiento profesional.

1.1 Motivación

El pádel es un deporte en auge que cada vez acoge más adeptos. De hecho, se trata de uno de los deportes que más han crecido los últimos años. En Europa durante los años 2020 y 2021 se han abierto 98 pistas y 28 clubes cada semana, experimentando un crecimiento de pistas en España del 13% y en Italia y Suecia del 374% y 388%, respectivamente [1].

Su público mayoritario se localiza en la franja de los 18 y 55 años, por lo que es un deporte que puede practicar casi todo el mundo. Sin embargo, se trata de un deporte relativamente moderno y muy focalizado en países como España o Argentina [2]. Su origen se remonta a finales de los años 60. Debido a esto, no existen tantas herramientas enfocadas a su estudio como para deportes más establecidos como el tenis, futbol, baloncesto, etc.

El análisis de vídeos deportivos es fundamental en el mundo del deporte, ya que proporciona una fuente de información detallada para entrenadores, atletas y analistas. Permite evaluar y mejorar el rendimiento de los atletas, desarrollar tácticas y estrategias más eficaces, e identificar patrones y tendencias que pueden no ser perceptibles a simple vista. En esencia, la capacidad de analizar vídeos deportivos puede ofrecer una ventaja competitiva y contribuir significativamente a la toma de decisiones estratégicas y al desarrollo de habilidades deportivas.

En muchos deportes, se hace uso de la Inteligencia Artificial para facilitar la extracción de estas estadísticas o identificación de patrones. Sin embargo, en el deporte pádel, debido a su reciente popularidad, no existen herramientas de esta índole. Para la obtención de estadísticas complejas, se necesita de una base de herramientas como la detección de la posición de los jugadores o la pista, clasificación de golpes, entre otras. Este trabajo se centrará en el análisis de la trayectoria de la pelota, concretamente, en la detección de la posición de la pelota en videos.

1.2 Objetivos

El objetivo principal de este proyecto consiste en la implementación y validación de un sistema basado en Inteligencia Artificial (IA) que, dada una entrada de video de un partido de pádel, analice la trayectoria de la pelota detectando su posición en cada fotograma.

El ámbito del pádel actualmente posee sistemas de seguimiento y análisis muy sofisticados y precisos, sin embargo, estos suelen ser complejos y costosos, lo que limita su accesibilidad y uso en escenarios de menor escala o con presupuestos más ajustados. Por lo tanto, uno de los objetivos clave de este proyecto es ofrecer una alternativa más asequible y escalable a estos sistemas existentes.

La consecución de este objetivo abrirá la puerta a poder realizar una recolección eficiente de estadísticas de partidos de pádel, permitiendo un análisis más detallado de los mismos con un bajo coste.

Teniendo en cuenta esta base, a continuación, se especifican los objetivos:

1. Generar un conjunto de datos para el deporte pádel con etiquetas que al menos indiquen la posición de la pelota en cada imagen. Este conjunto será utilizado para el entrenamiento y validación de los modelos de IA.
2. Crear y/o adaptar un método basado en IA capaz de detectar la posición de una pelota en un fotograma perteneciente al deporte pádel. Para cubrir este objetivo se han planteado dos subobjetivos: estudiar otras redes neuronales que realicen esta tarea para deportes como el tenis o el bádminton y adaptar su arquitectura y entrenamiento al conjunto de datos de pádel.
3. Conseguir utilizar el sistema en tiempo real utilizando una tarjeta gráfica (GPU) de gama media. Es decir, que el sistema sea capaz de predecir el resultado de al menos 25 fotogramas por segundo.

1.3 Entorno socioeconómico

En el mundo del deporte, la recopilación y el análisis de estadísticas son fundamentales para mejorar las tácticas de los jugadores, personalizar sus entrenamientos u ofrecer una experiencia de visualización más inmersiva. Este trabajo presenta una herramienta potencialmente valiosa en este contexto, utilizando técnicas de IA para seguir y la trayectoria de una pelota de pádel a partir de videos de partidos.

El sistema propuesto en este trabajo puede tener varias aplicaciones prácticas, todas ellas centradas en la optimización del análisis de los partidos de pádel. Una de las aplicaciones más directas sería servir como base para la creación de otros sistemas de generación de estadísticas con funcionalidades como rastrear la cantidad de golpes, la velocidad de la pelota o incluso la eficiencia de los jugadores. También, podría servir como parte de un sistema para anotar los puntos de partidos de pádel, lo que podría ser especialmente útil en torneos de aficionados donde no se dispone de un árbitro.

El uso de tecnologías de análisis de vídeo, como la propuesta en este trabajo, tiene el potencial de minimizar el hardware necesario en comparación con los sistemas actuales de generación de estadísticas. Esto es relevante desde un punto de vista medioambiental, ya que reducir la cantidad de hardware reduce la huella de carbono asociada con su fabricación. Además, si se consigue reducir el tamaño del modelo para que sea más rápido, se reduciría también el consumo de energía.

A largo plazo, el objetivo es utilizar este trabajo como punto de partida para desarrollar un sistema automático de generación de estadísticas de bajo coste. Este sistema podría ser de interés para empresas como World Padel Tour (WPT), que podrían aprovechar esta tecnología para recopilar sus estadísticas de manera más económica y rápida. Un paralelismo se puede trazar con el fútbol, donde ciertas estadísticas de los partidos son calculadas manualmente por personas. La automatización de este proceso, usando nuestro sistema, podría reducir los costos y el tiempo de procesamiento, al mismo tiempo que proporcionaría un análisis más preciso y detallado de los partidos.

En resumen, este proyecto tiene el potencial de tener un impacto significativo en el deporte del pádel. Puede ofrecer una nueva herramienta para el análisis de partidos y ayudar a democratizar el acceso a la tecnología de análisis de deportes para clubes y organizaciones con presupuestos más bajos. A través de su implementación y uso, puede contribuir a la eficiencia energética y la sostenibilidad medioambiental en el sector del deporte.

1.4 Marco regulador

Parte de este trabajo se ha basado en el uso de videos de pádel obtenidos de la plataforma de YouTube. Cada uno de estos videos, subidos por el canal llamado World Padel Tour, se encuentra protegidos bajo la Licencia Estándar de YouTube que, por lo general, no permite la reutilización del contenido sin permiso del titular de los derechos [3]. Sin embargo, para este estudio se ha considerado la doctrina del "uso justo", una excepción legal que permite el uso limitado de material protegido por derechos de autor para fines específicos, incluyendo la investigación [5].

El "uso justo" es una doctrina legal que está especialmente desarrollada en países como Estados Unidos codificada en la sección 107 de la “Ley de Derechos de Autor” de 1976 [5] y promovida también en Europa a partir de “Guías de copyright y uso justo” [4]. Bajo este marco, se consideran cuatro factores para determinar si un uso específico de material con derechos de autor se considera "uso justo":

> 1) Se considera el objetivo y la naturaleza del uso del material. Es decir, si tiene fines comerciales o si sus propósitos son educativos sin ánimo de lucro.
>
> 2) La esencia de la obra es también un factor relevante. Esto puede incluir el tipo de obra, su publicación y otros aspectos.

3) Se evalúa la cantidad y la importancia de la parte de la obra que se utiliza en comparación con la obra completa.

4) Finalmente, se tiene en cuenta si el uso de la obra puede probar un impacto en su valor o en sus ventas.

En el caso de presente trabajo, el uso de los videos de pádel tiene un fin investigativo y no lucrativo. Además, la porción utilizada de la obra es muy pequeña en comparación a la obra completa, ya que se han utilizado un total de 11 minutos extraídos de numerosas horas de video. Estos argumentos refuerzan la justificación basada en el "uso justo". Sin embargo, se debe reconocer que existe una incertidumbre legal inherente al aplicar la doctrina de "uso justo" a casos concretos, ya que los tribunales evalúan cada caso en función de sus circunstancias específicas.

Respecto a las herramientas y bibliotecas de software utilizadas en el proyecto, están sujetas a varias licencias de código abierto. El lenguaje de programación Python, se distribuye bajo una licencia de código abierto compatible con la Licencia General Pública (GNU) pero menos restrictiva en ciertos aspectos [6]. Tensorflow, una plataforma de aprendizaje automático de código abierto, y Keras, una interfaz de alto nivel para Tensorflow, se distribuyen bajo la licencia Apache 2.0 [7]. Esta licencia permite el uso, la distribución y la modificación del software, así como la distribución del software modificado bajo otras licencias. Las bibliotecas adicionales de Python utilizadas, incluyendo pandas, numpy o matplotlib, también se distribuyen bajo licencias de código abierto que permiten un uso amplio con pocas restricciones [8,9,10]. Los controladores CUDA y cuDNN, necesarios para la computación en GPU, son propiedad de NVIDIA y se distribuyen bajo la licencia de software de NVIDIA, que permite el uso gratuito en un contexto de desarrollo [11].

Para asegurar la conformidad con las normativas en el campo de la IA, es imperativo adherirse al nuevo marco de regulación establecido por la Comisión Europea el 21 de abril de 2021 [12] que es aplicable a las entidades tanto públicas como privadas que utilicen dicha tecnología. Este marco propone una estrategia basada en el nivel de riesgo, consistente en cuatro categorías: riesgo inadmisible, alto, limitado y mínimo, dependiendo del peligro que suponga el uso de dicha IA.

En el caso de este trabajo, se presenta un "riesgo limitado" según el marco de la Comisión Europea. Aunque no compromete la seguridad de las personas, podría suscitar preocupaciones menores sobre privacidad y derechos de autor debido al uso de vídeos públicos de YouTube. Sin embargo, estos riesgos son manejables con el debido cumplimiento de las regulaciones y la atención adecuada a las licencias de uso.

En conclusión, este trabajo ha seguido cuidadosamente los marcos reguladores pertinentes, tanto en relación con el uso de vídeos protegidos por derechos de autor como con las licencias de código abierto de las herramientas de software utilizadas. Aunque el uso de estos vídeos puede llevar cierto grado de incertidumbre legal bajo la doctrina de "uso justo", se ha tomado todas las precauciones posibles para justificar este uso dentro

del contexto de la investigación académica. Además, se ha clasificado el proyecto como de "riesgo limitado" en términos del nuevo marco regulatorio de la IA de la Comisión Europea, y se han gestionado de forma adecuada las posibles preocupaciones sobre la privacidad y los derechos de autor. De este modo, este proyecto sirve como un ejemplo de cómo se pueden navegar los desafíos regulatorios y éticos al desarrollar nuevas aplicaciones de la IA en el ámbito de los videos deportivos.

1.5 Estructura

En este apartado se presenta un breve resumen sobre cada uno de los capítulos que forman este trabajo:

- **Estado del arte:** En este capítulo, se ofrece una revisión exhaustiva de las técnicas de visión por computadora y aprendizaje profundo empleadas para la detección y seguimiento de objetos en videos. En particular, se explora la evolución de las CNN, poniendo un énfasis especial en las redes TrackNet, diseñadas específicamente para la detección de pelotas en videos deportivos.

- **Generación del conjunto de datos del deporte pádel:** Este capítulo detalla el proceso de generación de datos. Se describe cómo se recopilaron los videos de partidos de pádel, cómo se etiquetaron las posiciones de las pelotas en los videos y cómo se preparó el conjunto de datos para el entrenamiento y la validación de la red.

- **Modelo de aprendizaje profundo y entrenamiento de la red:** En este capítulo, se presenta el diseño de la CNN utilizada, explicando las modificaciones que serán evaluadas sobre la estructura base de la red TrackNet. Además, se proporciona una guía detallada para utilizar el código desarrollado, tomando en cuenta los requisitos particulares de hardware y software.

- **Análisis de resultados:** Este capítulo se dedica a la presentación y al análisis crítico de los resultados obtenidos, así como los detalles de los experimentos realizados. Se discute la precisión de la detección de la posición de la pelota, la efectividad de las modificaciones implementadas y las limitaciones identificadas.

- **Planificación y presupuesto:** Este capítulo ofrece un resumen detallado de la planificación del proyecto, incluyendo el tiempo dedicado a cada tarea y los recursos necesarios. Se presenta un presupuesto detallado, en el que se incluye el tiempo de trabajo y el coste del hardware.

- **Conclusiones y futuros trabajos:** Finalmente, se resume el trabajo realizado, se destacan los logros y se discuten las posibles mejoras y extensiones para trabajos futuros. Se reflexiona sobre las implicaciones del trabajo en el campo del análisis de vídeo en el deporte de pádel y se sugieren posibles aplicaciones y mejoras.

2. ESTADO DEL ARTE

El propósito de este capítulo es ofrecer una visión general del campo de la visión artificial, enfocándose en los avances y desarrollos clave relacionados con la detección de pelotas en vídeos deportivos. De esta manera, se establecerán las bases teóricas y prácticas necesarias para el diseño y la implementación de la solución del problema abordado.

2.1 Aprendizaje profundo

El aprendizaje profundo, también conocido como *deep learning* es un subconjunto del aprendizaje automático. Dentro de este tipo de aprendizaje, podemos encontrar a las redes neuronales profundas, que se caracterizan por estar compuestas de muchas capas y parámetros. También se diferencia del aprendizaje automático en el tipo de datos con el que trabaja. Los algoritmos de aprendizaje automático aprovechan mejor los datos estructurados y etiquetados para hacer predicciones o clasificaciones. No obstante, el *deep learning* reduce la necesidad de preprocesamiento de datos que normalmente requiere el aprendizaje automático. Estos algoritmos son capaces de ingerir y procesar datos no estructurados, tales como texto e imágenes.

Supongamos que se quiere diseñar una red que identifique si en una imagen aparece un coche. El enfoque que debe tomar un proceso de aprendizaje automático consiste en extraer características de las imágenes y a través de ellas entrenar un modelo que obtenga clasificaciones en función de estas características. En cambio, en el aprendizaje profundo el enfoque cambia, éste se encarga del proceso de extracción de características, eliminando parte de la dependencia de los expertos humanos y pudiendo encontrar relaciones o características que los humanos no serían capaces de encontrar. En la figura 2.1 se puede observar la diferencia de ambos procesos [13].

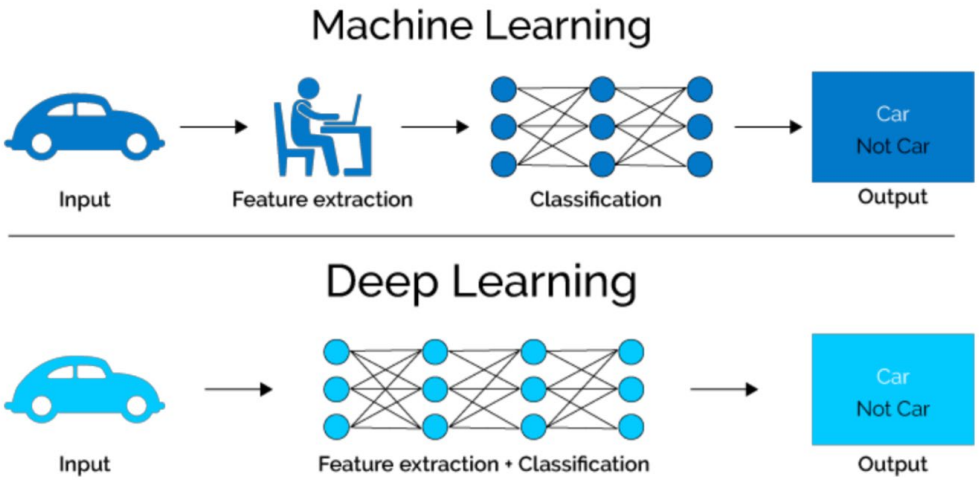


Figura 2.1 Diferencias entre aprendizaje automático y aprendizaje profundo [13]

En el ámbito de las redes neuronales y el aprendizaje profundo, existen diversos tipos de arquitecturas que han sido desarrolladas y aplicadas a distintos problemas y dominios. Entre las arquitecturas más destacadas se encuentran las CNN, los Transformers [14], las redes neuronales recurrentes (RNN), los *autoencoders* [15], las redes de memoria largo-corto plazo (LSTM) [16] o las redes neuronales generativas adversarias (GAN) [17]. Cada una de estas arquitecturas posee características y aplicaciones específicas que las hacen adecuadas para abordar distintos tipos de problemas en el campo del aprendizaje profundo.

2.2 Redes convolucionales

Las Redes Neuronales Convolucionales (CNN) son un tipo de arquitecturas de aprendizaje profundo especialmente diseñadas para el procesamiento de imágenes y reconocimiento de patrones [18]. Estas redes se han utilizado ampliamente en tareas de visión artificial, como clasificación de imágenes [19], segmentación semántica [20], y detección de objetos [21][22].

Una CNN típica incluye múltiples capas interconectadas, como capas convolucionales, capas de agrupamiento (*pooling*), capas de normalización, capas *softmax* y capas completamente conectadas [18]. A continuación, se explica el propósito y funcionamiento de cada una de ellas.

Las capas convolucionales constituyen el componente principal de las CNN. Dichas capas emplean filtros o núcleos convolucionales en las entradas, lo que posibilita la identificación de características locales en las imágenes. Cada filtro se especializa en la detección de un patrón concreto, como bordes, texturas o colores. La convolución se lleva a cabo al desplazar el filtro a lo largo de la imagen de entrada y calcular el producto escalar entre los valores del filtro y la región correspondiente de la imagen. Este proceso produce un mapa que resalta la presencia de las características identificadas por el filtro [18].

El objetivo de las capas de agrupamiento (pooling) es disminuir la dimensionalidad de los mapas de características creados por las capas convolucionales. El agrupamiento se realiza al seleccionar un valor representativo de una región del mapa de características, como el valor máximo (*max-pooling*) o el promedio (*average-pooling*). Esta operación permite a las CNN ser más robustas a pequeñas variaciones en la posición y escala de los objetos en las imágenes [18].

La capa de normalización por lotes, propuesta por Ioffe y Szegedy [23], busca mejorar la estabilidad y el desempeño de las redes neuronales durante el entrenamiento. Esta capa normaliza las entradas en cada lote de datos, ajustando la media y la varianza de las características. La normalización por lotes facilita el uso de tasas de aprendizaje más elevadas y disminuye la sensibilidad a la inicialización de los pesos de la red. Además, en muchos casos, esta técnica también actúa como un regularizador, lo que disminuye la necesidad de aplicar técnicas de regularización adicionales, como el *dropout* (abandono).

Las capas completamente conectadas (*fully connected*) se emplean generalmente al final de las CNN para fusionar las características extraídas en las capas previas y generar una salida final, como las probabilidades de pertenecer a diferentes clases en tareas de clasificación. En el caso de la detección de objetos, estas capas pueden estar acompañadas de capas adicionales para predecir la ubicación y tamaño de los objetos detectados [24].

La capa *softmax* suele utilizarse como última capa en una red neuronal para tareas de clasificación. Su función principal es transformar las puntuaciones de salida de la red en probabilidades normalizadas para cada clase de objeto [25]. La función *softmax* toma como entrada un vector de números reales y lo convierte en un vector de probabilidades, de modo que la suma de todas las probabilidades sea igual a 1. Esta capa es especialmente útil cuando se desea obtener una distribución de probabilidad sobre las posibles clases en tareas de clasificación multiclase. También puede resultar útil para resolver una tarea de clasificación, escogiendo la salida con mayor probabilidad como la clase de la entrada.

2.3 Detección de pequeños objetos en movimiento

La detección de pequeños objetos en movimiento en imágenes o videos es un área de investigación de gran interés en el campo de la IA y la visión por computadora. Esta tarea es particularmente compleja debido a varios factores, como la baja resolución de los objetos, la rápida variación de la apariencia y la forma, la presencia de oclusiones, el ruido en la imagen y los cambios en las condiciones de iluminación. Además, en el contexto de aplicaciones deportivas o industriales, estos objetos a menudo se mueven a velocidades extremadamente altas, lo que dificulta aún más su detección y seguimiento. A pesar de estos desafíos, el avance en algoritmos y modelos de inteligencia artificial que pueden abordar eficazmente la detección de objetos pequeños en movimiento ofrece un enorme potencial en varios campos, como la automatización industrial, la robótica, el análisis de videos deportivos y la seguridad. La investigación en esta área continúa avanzando, y las técnicas de aprendizaje profundo, como las CNN, han mostrado resultados prometedores en términos de precisión y solidez en la detección y seguimiento de objetos en movimiento. A continuación, se mostrarán varios ejemplos de soluciones que pretenden abordar este problema en el contexto de la detección de pelotas en videos deportivos.

Una aproximación sencilla para abordar este problema es utilizar la librería OpenCV (*Open Source Computer Vision Library*), una biblioteca de código abierto ampliamente utilizada en el campo de la visión por computadora. Esta librería proporciona una amplia variedad de herramientas y algoritmos para tareas de procesamiento de imágenes, incluyendo la detección y reconocimiento de objetos, el seguimiento de objetos en movimiento, la estimación de la pose, análisis de imágenes y videos en tiempo real, etc. En el ámbito de la detección de pelotas en videos deportivos podemos encontrar trabajos como [26], donde se pretende detectar pelotas de voleibol mediante las herramientas que ofrece OpenCV, primero detectando objetos en movimiento y, posteriormente, seleccionando los que sigan trayectorias parabólicas. En [27] se implementa un sistema que es capaz de detectar la posición de pelotas de tenis de mesa dado un video grabado

con un plano cenital. En este trabajo el enfoque es distinto al anterior, se detecta el área de la mesa y se buscan en él, objetos de color blanco. Si se encuentra, en el siguiente fotograma se utiliza un radio de búsqueda alrededor de la posición anterior. En caso de no encontrarse encima de la mesa, se utiliza un detector de objetos en toda la imagen y se discrimina según su forma (objeto redondo). Ambos enfoques proporcionan buenos resultados en comparación a su bajo consumo de recursos y facilidad de implementación, poseyendo la ventaja de no necesitar un conjunto de datos etiquetado.

En [28] se pretende detectar la pelota de tenis de mesa para uso arbitral. El enfoque propuesto combina técnicas de procesamiento de imágenes y algoritmos de seguimiento de objetos para detectar la posición y trayectoria de la pelota en tiempo real. Para tomar las imágenes se utilizan cámaras de alta velocidad desde 2 perspectivas. Después, se realiza un preprocesamiento para eliminar el ruido y mejorar la calidad de la imagen. Esto puede incluir técnicas como el filtrado, la ecualización del histograma y la corrección del contraste. Para mejorar la identificación de la pelota se combinan métodos que son capaces de detectar bordes y otros basados en el color y la forma. Una vez que se ha identificado la posición de la pelota en cada imagen, se emplea un algoritmo de seguimiento de objetos para estimar su posición y trayectoria en 3D a lo largo del tiempo. Esto permite estimar la posición de la pelota incluso en presencia de oclusiones o ruido en las imágenes. Como desventaja, este sistema requiere de al menos 2 cámaras y está enfocado en solucionar un problema muy concreto.

En [29] se pretende detectar la pelota en videos de partidos de tenis usando técnicas de segmentación que analizan color y forma. El color de la pelota (generalmente amarillo o blanco) se identifica utilizando un umbral adaptativo en el espacio de color HSV (*Hue Saturation Value*). Después, se filtran los contornos de la imagen resultante para localizar aquellos que tienen una forma y tamaño similares a una pelota de tenis. Esta detección se realiza en cada fotograma del video. Para estimar la trayectoria de la pelota, se utiliza un enfoque basado en el flujo óptico, que calcula el movimiento aparente de los objetos entre fotogramas consecutivos. Combinando esto con la posición de la cámara y la corrección de perspectiva, se logra una estimación precisa de la trayectoria en 3D de la pelota en el video. El seguimiento de la pelota se lleva a cabo con un enfoque de seguimiento basado en la trayectoria, que considera tanto la posición como la velocidad de la pelota en el tiempo. Este enfoque permite rastrear de manera fiable la pelota a lo largo del video, incluso si se producen oclusiones o cambios en su apariencia. Este algoritmo es más robusto que los anteriores, sin embargo, la detección basada en forma y color puede resultar poco efectiva en algunos casos.

En [20], se presenta una metodología para detectar y seguir el movimiento de la pelota de tenis de mesa utilizando una red de multiplexación de características temporales (TFMN). Esta red consta de dos partes principales: una red de características espaciales y una red de características temporales. La red de características espaciales es responsable de aprender las características visuales de la pelota en cada fotograma, mientras que la red de características temporales aprende las relaciones temporales entre los fotogramas consecutivos. Ambas redes se combinan para generar una representación de

características temporales y espaciales de la secuencia de imágenes. Además, se utiliza un módulo de atención para ponderar las características en función de su importancia, permitiendo que la red detecte las más relevantes. Finalmente, se entrena la red utilizando una función de pérdida que tiene en cuenta tanto la ubicación de la pelota, como su trayectoria a lo largo del tiempo. La metodología propuesta en este artículo ofrece una solución efectiva y precisa. Sin embargo, su implementación requiere de una gran pericia en el ámbito de las redes *Transformers* [14].

2.4 Detección de pelotas utilizando redes neuronales

El uso de las Redes Neuronales Convolucionales es especialmente útil para la detección de pequeños objetos en movimiento, ya que estas redes son capaces de extraer automáticamente características de bajo nivel, como bordes y texturas, y combinarlas para formar representaciones de alto nivel, como formas y patrones. Esta capacidad permite a las CNN reconocer y localizar objetos pequeños y en movimiento, a pesar de cambios en la iluminación, la perspectiva y la apariencia. Además, las CNN son robustas frente a oclusiones y pueden adaptarse a variaciones en el tamaño y la forma de los objetos de interés. En el caso específico de la detección de pelotas en videos deportivos, las CNN pueden aprender las características más relevantes de las pelotas, como su color, forma y trayectoria, lo que permite una detección precisa y un seguimiento confiable en secuencias de imágenes o videos.

En esta sección, se describen diferentes implementaciones de la red TrackNet [31], las cuales son empleadas para detectar pequeños objetos en movimiento, como pelotas en videos deportivos. Estas redes han sido utilizadas como base para el trabajo realizado.

2.4.1 TrackNet

TrackNet [31] es una red capaz de predecir la posición de pelotas pequeñas en videos de deportes. Originalmente, esta red se diseñó para detectar pelotas en videos de tenis, pero después se ha modificado para usarse en otros deportes como el bádminton, tenis de mesa, etc. Además, ha servido de base para que otras redes mejoren su eficiencia o precisión como TrackNetV2 [32] o MonoTrack [33].

La composición de TrackNet incluye una CNN y una red neuronal deconvolucional (DeconvNet) [34]. El objetivo del modelo es generar mapas de calor que identifican la posición del objeto analizado. Este enfoque ha demostrado ser útil en diversas aplicaciones como [35] y [36], en las que se utilizan los mapas de calor para detectar la posición de seres humanos. La red TrackNet utiliza fotogramas consecutivos con el fin de aprender patrones de trayectoria, siendo el número de fotogramas de entrada ajustable según los parámetros de la red. En el caso del modelo base de TrackNet, se implementan dos versiones de la red para evaluar su efectividad: una con un una entrada de un solo fotograma y otra con tres fotogramas consecutivos como entrada.

El mapa de calor consiste en una distribución gaussiana de dos dimensiones, centrada en la pelota. La varianza de esta distribución gaussiana es calculada considerando el diámetro medio de la pelota es de 5 píxeles. Esta distribución puede observarse en la Figura 2.2.

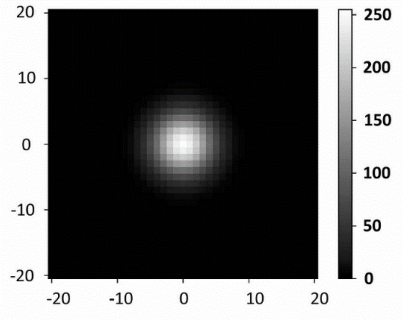


Figura 2.2 Ejemplo de mapa de calor producido por la red TrackNet [31]

Los detalles de implementación de TrackNet se presentan en la Figura 2.3, en la que se muestra la arquitectura de la red, incluyendo las diferentes capas, número de filtros, entradas y salidas. La entrada de la red puede ser un número determinado de fotogramas consecutivos. El tamaño de cada fotograma es de 640 x 360 píxeles, compuestos por 3 capas (RGB) dado que se trata de imágenes en color. Las primeras 13 capas siguen el diseño de la red VGG-16, una arquitectura propuesta por K. Simonyan y A. Zisserman en 2014 [37], conocida por su simplicidad y efectividad en tareas de reconocimiento de imágenes y clasificación. Esta red, hace uso de la función de activación ReLU (*Rectified Lineal Unit*) y de la capa *max-pooling* para mejorar el rendimiento y reducir la complejidad computacional. En cambio, las capas 14-24 hacen referencia a DeconvNet [34], una arquitectura de red neuronal diseñada para realizar tareas de segmentación semántica en imágenes, propuesta por H. Noh, S. Hong y B. Han en 2015 [38]. La red final, muestra una arquitectura simétrica entre las capas agrupamiento (*max-pooling*) y sobremuestreo (*upsampling*). Esto permite analizar las imágenes y después poder recuperar la información en la misma resolución de entrada.

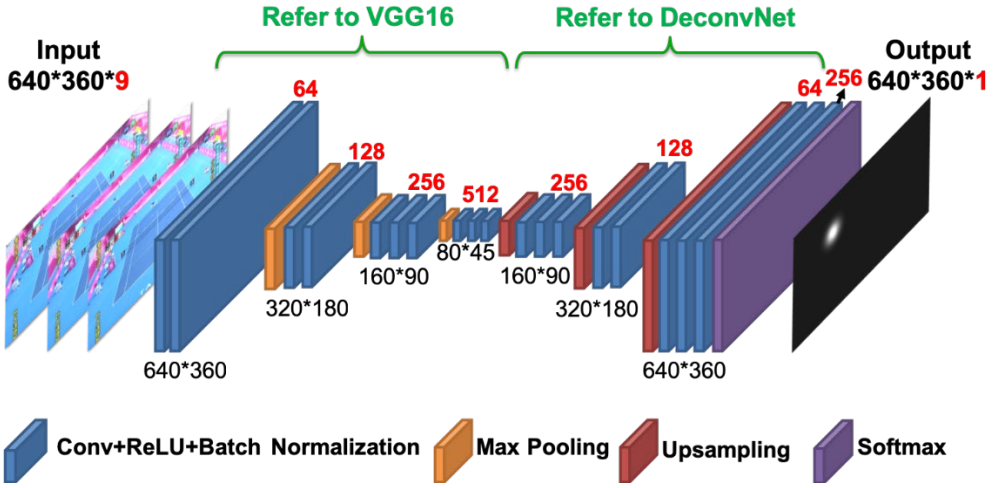


Figura 2.3 Estructura de la red convolucional utilizada en TrackNet [31]

El mapa de calor de que aparece en la Figura 2.3 no está disponible directamente en la salida de la red. La red genera un mapa de calor de detección con valores continuos en el rango de [0, 255] para cada píxel. A partir de estos mapas de calor, la función softmax (Ecuación 2.1) calcula la probabilidad de cada valor de profundidad (k) para cada píxel (i, j). Una vez realizado este cálculo, se selecciona la profundidad con la mayor probabilidad como el valor del mapa de calor del píxel (Ecuación 2.2) [31].

$$P(i,j,k) = \frac{e^{L(i,j,k)}}{\sum_{l=0}^{255} e^{L(i,j,l)}} \tag{2.1}$$

$$h(i,j) = \arg max_k \, P(i,j,k) \tag{2.2}$$

Una vez generado el mapa de calor de detección, la coordenada de la pelota se determina mediante los siguientes dos pasos. Primeramente, se convierte de manera individual cada píxel en un mapa de calor binario blanco y negro mediante un umbral t. Si un píxel tiene un valor igual o superior a t, se establece el píxel en 255. Por el contrario, si un píxel tiene un valor menor que t, se establece el píxel en 0. El umbral se fija en un valor de 128, tomando en consideración el radio promedio de una pelota de tenis. El segundo paso implica la utilización del Método del Gradiente de Hough, una técnica de extracción de características que se emplea para detectar círculos en imágenes [43]. Si se identifica exactamente un círculo, se devuelve su centroide. En otros casos, se considera que la red no detecta ninguna pelota.

Para entrenar la red TrackNet, se utilizó el conjunto de datos utilizado para evaluar el algoritmo propuesto por Archana [29], que incluye 20.844 fotogramas capturados en varios partidos de tenis. Se aplicó validación cruzada con un 70% de los datos y 30% restante se empleó como conjunto de prueba. Además, se utilizó el *loss* (función de pérdida) de validación para seleccionar el mejor modelo y evitar así el *underfitting* (infrajuste) y *overfitting* (sobreajuste). En la Figura 2.4 se muestra la evolución de la función de *loss* en entrenamiento y validación a lo largo del aprendizaje. La función de pérdida utilizada se denomina *pixel-wise cross entropy* (entropía cruzada en píxeles) [40] que evalúa la discrepancia entre las predicciones del modelo y las etiquetas a nivel de píxel, penalizando las predicciones incorrectas.

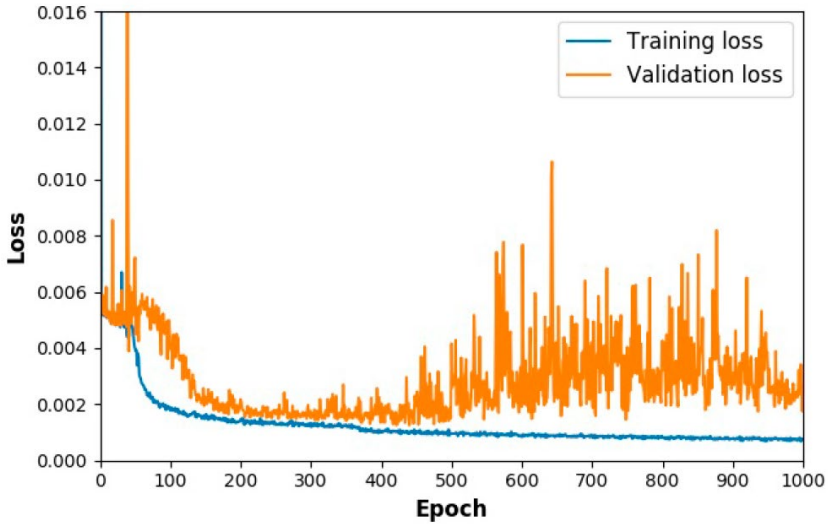


Figura 2.4 Curva del loss durante el entrenamiento de TrackNet [31]

Los experimentos se llevaron a cabo comparando el rendimiento de varios modelos de TrackNet y otros algoritmos existentes en términos de precisión, *recall* (exhaustividad) y medida F1, como se muestra en la tabla 2.1. El modelo de Archana se refiere a la técnica de procesamiento de imágenes propuesta por Archana y Geetha [29], el modelo I de TrackNet corresponde a la implementación, que recibe un solo fotograma como entrada, mientras que el modelo II recibe 3 fotogramas consecutivos. Además, se incluye un tercer modelo (Modelo II'), que cuenta con la misma implementación que el modelo II, pero se añade a su entrenamiento un segundo conjunto de datos de 16.118 fotogramas de partidos con diferentes colores de pistas. Esto se introduce con el objetivo de reducir el sobreaprendizaje de la red [31].

Como se puede observar en la tabla 2.1, las distintas implementaciones de TrackNet consiguieron mejorar los resultados obtenidos con el algoritmo de Archana. Entre ellos destaca el Modelo II, que recibe 3 fotogramas como entrada, lo cual ayuda a que la red identifique patrones en las trayectorias que sigue la pelota y no solo en la imagen. Además, con el Modelo II' se obtiene una pequeña mejora, fruto de reducir el sobreajuste de la red [31].

TABLA 2.1
MÉTRICAS PARA DIFERENTES MODELOS [31]

Modelo	Precisión	*Recall*	Medida F1
Modelo de Archana	92.5%	74.5%	82.5%
Modelo I TrackNet	95.7%	89.6%	92.5%
Modelo II TrackNet	99.8%	96.6%	98.2%
Modelo II' TrackNet	99.7%	97.3%	98.5%

2.4.2 TrackNetV2

En la ICPAI (*International Conference on Pervasive Artificial Intelligence*) de 2020, se presenta una segunda versión de la red TrackNet, llamada TrackNetV2 [32]. Esta red pretende reducir el elevado consumo de recursos de GPU y acelerar el proceso de predicción de la red TrackNet. Además, busca obtener un modelo más adecuado para el deporte de bádminton, ya que con TrackNet se obtenían resultados menos precisos debido a la dificultad para detectar las pelotas en los fotogramas de este deporte.

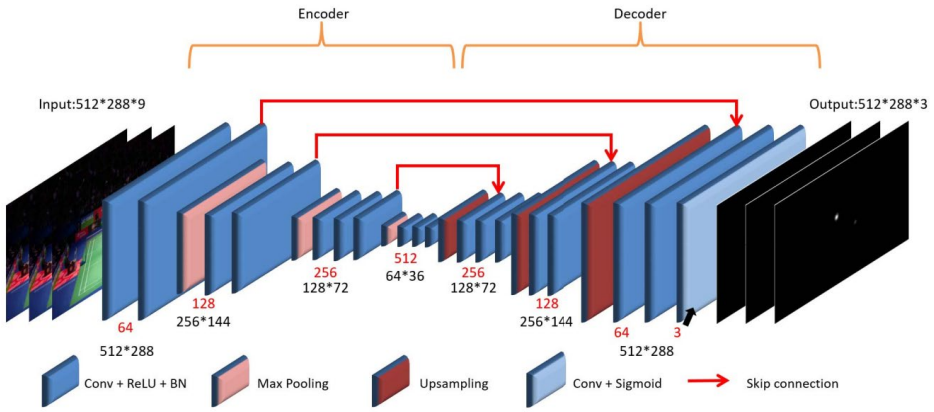


Figura 2.5 Arquitectura de la red TrackNetV2 [32]

Para lograr estos objetivos, los autores realizaron varios cambios en la arquitectura de TrackNet. En primer lugar, se redujo el tamaño de entrada de la red de 640x360 a 512x288, lo que permite un menor consumo de recursos y acelerar el proceso de entrenamiento y predicción. En segundo lugar, se modificó la arquitectura de la red, pasando de una estructura MISO *(Multiple Input, Single Output*) a una MIMO (*Multiple Input, Multiple Output*), para que la red tenga como salida 3 fotogramas, en lugar de 1, lo cual acelera el proceso de predicción. Esto se debe a que en una misma propagación hacia delante de las entradas se generan tres salidas en lugar de una.

Otra mejora implementada en TrackNetV2 es la introducción de *skip-connections* (conexiones de salto), también conocidas como *residual connections* (conexiones residuales). Estas conexiones unen capas de la red no son consecutivas, como se observa en la Figura 2.5, permitiendo a la información saltar capas en redes neuronales profundas. Las conexiones típicas entre una capa y la siguiente no desaparecen, sino que coexisten con las conexiones residuales. Esto permite abordar el problema del desvanecimiento del gradiente (*vanishing gradient*) [41] provocado por la gran profundidad de la red y mejorar el rendimiento en la clasificación y detección de objetos en imágenes [43]. De este modo, estas conexiones facilitan la comunicación directa entre capas distantes, mejorando la eficiencia del entrenamiento y la propagación de características de bajo nivel a lo largo de la red [42]. Las *skip-connections* se han adoptado en diversas arquitecturas de CNN, como ResNet [46] y U-net [44]. En ResNet, mejoran la eficiencia del entrenamiento y el rendimiento en clasificación de imágenes. En U-net, se emplean para mejorar la reconstrucción de imágenes en aplicaciones de segmentación, permitiendo la recuperación de detalles finos en las imágenes reconstruidas.

Para TrackNetV2, las conexiones residuales se aplican de la misma forma que en U-net, con el objetivo de recuperar la extracción de características que se realiza en las primeras capas de la red y que, posteriormente se va perdiendo a medida que se reduce el tamaño de las capas. Como las pelotas son objetos muy pequeños y poco visibles, el uso de *skip-connections* podría favorecer a recuperar las características extraídas en las primeras capas. En la Figura 2.5, se aprecia el uso de las *skip connections* que enlazan capas del mismo tamaño entre la parte convolucional y la deconvolucional, permitiendo recuperar características extraídas en la convolución. Estas conexiones residuales son concatenativas, lo que significa que combinan la salida de la capa de origen con la salida de la capa a la que se conectan, uniendo ambas salidas en un vector más largo.

Otra de las modificaciones propuestas en la red TrackNetV2, es el tipo de salida de la red: en lugar de 256 canales que deben ser transformados a un mapa de calor, ahora se utiliza un único canal compuesto por una matriz de dos dimensiones en la que cada posición representa un píxel de la imagen. Cada elemento de la matriz es un número real que indica la probabilidad de que en ese píxel se encuentre una pelota en la imagen. Esta modificación permite un ahorro de memoria y tiempo.

Por último, los autores rediseñaron la función de pérdida (*loss*) para adaptarla a la salida de la red TrackNetV2. Esta nueva función de pérdida está basada en la entropía cruzada binaria ponderada. Próximamente, se analizará esta función en profundidad.

Para el entrenamiento de la red, se utilizó el optimizador Adadelta con una tasa de aprendizaje inicial de 1. Este optimizador ajusta la tasa de aprendizaje para cada parámetro utilizando una ventana deslizante, lo cual mejora la convergencia del entrenamiento y reduce el problema de la rápida disminución de la tasa de aprendizaje que ocurre en otros optimizadores tradicionales [45]. Los autores emplearon un nuevo conjunto de datos del deporte badminton con distintos fondos y perspectivas con el objetivo de reducir el sobreaprendizaje. En total se recopilaron 55563 fotogramas más 13200 extras como conjunto de prueba para poder comparar los distintos modelos [44].

Los resultados del experimento (Tabla 2.2) evidenciaron las mejoras implementadas en la red. Respecto a la precisión de la red, las distintas métricas muestran una notable mejora en la detección de los volantes de bádminton por parte del modelo. Esto es debido tanto a las mejoras de la arquitectura como la ampliación del conjunto de entrenamiento, que como veremos en el capítulo 3, juega un papel fundamental en la creación de modelos de aprendizaje profundo [46].

TABLA 2.2
COMPARACIÓN DE LOS MODELOS DE TRACKNET Y TRACKNETV2 PARA EL DEPORTE BÁDMINTON, AMBOS MODELOS TIENEN UNA ENTRADA DE 3 FOTOGRAMAS [8] [43]

Modelo	Precisión	*Recall*	F1
TrackNet (3-1)	85.0%	57.7%	68.7%
TrackNetV2 (3-3)	97.2%	85.2%	90.8%

En términos de velocidad de procesamiento de la red, también se obtuvieron notables mejoras. En la tabla 2.3, se compara la velocidad y características de varios modelos. Se puede detectar que la implementación de los mapas de calor, reducir el tamaño de entrada y devolver 3 fotogramas en lugar de 1 reducen considerablemente el tiempo de procesamiento, obteniendo una mejora del 1106%. La modificación que aumenta más la eficiencia es la modificación de la salida de la red utilizando 3 fotogramas, mientras que las conexiones residuales reducen levemente la eficiencia del procesamiento.

TABLA 2.3
COMPARACIÓN DE VELOCIDAD ENTRE MODELOS DE TRACKNET Y TRACKNETV2 [32]

Modelo	Tamaño de entrada	Mapas de calor	Conexiones residuales	Velocidad
TrackNet 3-1	640 x 360			2.64
TrackNetV2 3-1	640 x 360	✓		10.42
TrackNetV2 3-1	512 x 288	✓		14.15
TrackNetV2 3-1	512 x 288	✓	✓	12.88
TrackNetV2 3-3	512 x 288	✓	✓	31.84

En resumen, TrackNetV2 introduce varias mejoras en comparación con TrackNet, como la reducción del tamaño de entrada, el cambio en la arquitectura de la red, la implementación de *skip-connections* y la modificación de la función de pérdida. Estas mejoras permiten reducir el consumo de recursos y aumentar la velocidad y precisión en la detección de objetos en deportes como el bádminton.

2.4.3 MonoTrack

En el artículo "*MonoTrack: Shuttle trajectory reconstruction from monocular badminton video*" [33], los autores presentan una metodología para reconstruir la trayectoria del volante en videos de partidos de bádminton. Proponen una versión mejorada de la red TrackNetV2 llamada MonoTrack. MonoTrack incluye cambios en la arquitectura de la red TrackNetV2 para adaptarla a las condiciones específicas del bádminton y mejorar su rendimiento en la detección y seguimiento del volante. En la Figura 2.6 se muestra la arquitectura de esta red.

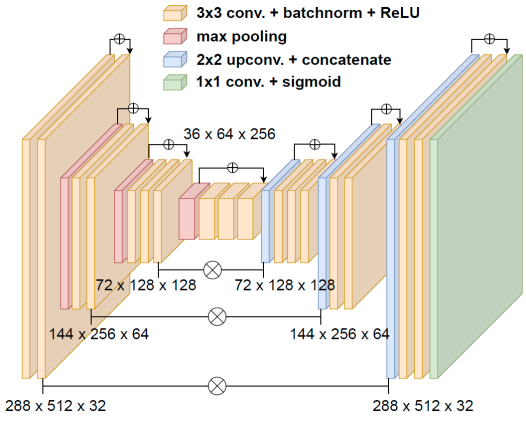


Figura 2.6 Arquitectura de la red Monotrack [33]

La primera mejora consistía en una modificación de la arquitectura de la red (Figura 2.6) disminuyendo el número de parámetros de los 11.3 millones de TrackNetV2 [32] a 2.9 millones de MonoTrack. Esto se consiguió reduciendo a la mitad el número de filtros de cada capa. Esta modificación reduce considerablemente la duración del entrenamiento y aumenta la velocidad de predicción de la red.

Otra modificación que se aplicó a la red es la implementación de *skip connections* en cada capa convolucional, además de las ya implementadas en TrackNetV2 que unían las capas convolucionales y deconvolucionales. En este caso se utilizan conexiones residuales aditivas, que a diferencia de las concatenativas, suman la salida de una capa a la salida de la capa a la que están conectadas. Esto permite perder menos información entre estas capas.

Otro cambio significativo que propusieron los autores fue la función de pérdida utilizada. En lugar de la entropía cruzada binaria ponderada, aquí se optó por utilizar una combinación de *Dice Loss* y entropía cruzada binaria. Esta combinación permite reducir el desbalanceo de clases que este problema de segmentación contiene. Posteriormente, se analizará esta función de pérdida en profundidad.

Para aprovechar el entrenamiento de la red TrackNetV2 se hizo uso del *Distillation learning* o *knowledge distillation* (aprendizaje por destilación). Este método consiste en transferir el conocimiento de un modelo grande y complejo, llamado "modelo maestro", a un modelo más pequeño, conocido como "modelo alumno". La idea es mantener un rendimiento similar con menor consumo de recursos y mayor eficiencia en la ejecución. Durante el proceso de destilación, el modelo alumno es entrenado para imitar las salidas del modelo maestro, utilizando como guía no solo las etiquetas verdaderas sino también las probabilidades de clase proporcionadas por el modelo maestro. Esta técnica permite al modelo alumno aprender representaciones más ricas y generalizar mejor que si se entrenara solo con las etiquetas verdaderas [47].

2.5 Transferencia de aprendizaje

La transferencia de aprendizaje o *transfer learning* consiste partir de un modelo ya entrenado para una tarea y volver a entrenarlo empleando nuevos datos con el objetivo de utilizarlo en una tarea similar. Esta técnica, suele aplicarse cuando se necesita ajustar un modelo o modificar la tarea que resuelve y no se poseen muchos datos de entrenamiento [48]. Esto evita que el modelo deba ser entrenado desde 0, ahorrando cantidad de datos necesarios y tiempo de entrenamiento y diseño de la red. Además, se aprovecha el conocimiento ya adquirido por otros modelos pudiendo obtener modelos que sean capaces de generalizar mejor.

Normalmente, cuando se aplica transferencia de aprendizaje [49] a una CNN se suelen eliminar las capas totalmente conectadas o *fully-connected*, ya que estas son las que calculan el resultado final de la red, mientras que las capas convolucionales están enfocadas en extraer características de las entradas. Sin embargo, existe el peligro de

provocar sobreajuste, debido a que la red ya está entrenada y puede ajustarse demasiado a los nuevos datos. Una posible solución a este problema es utiliza una tasa de aprendizaje baja, de modo que los pesos no sufran cambios bruscos.

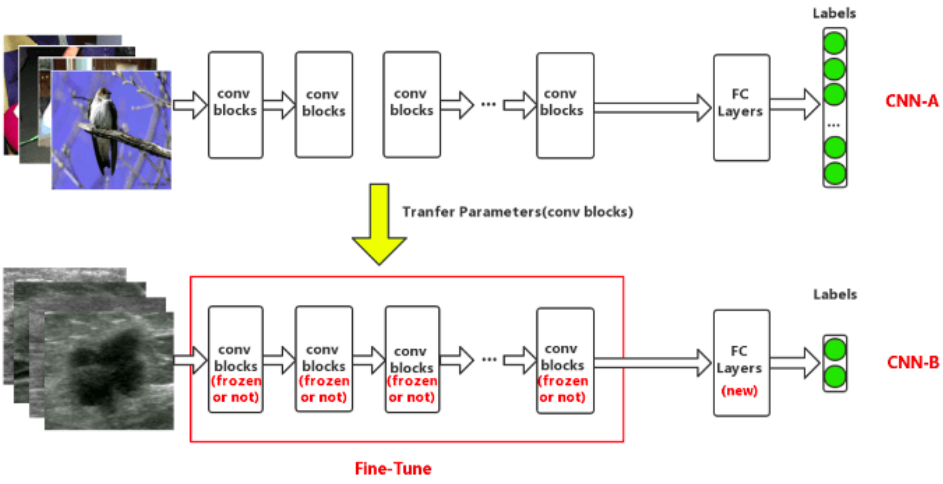


Figura 2.7 Transferencia de aprendizaje y fine-tunning [49]

Opcionalmente, cuando se emplea transferencia de aprendizaje se puede aplicar *fine-tunning*, que consiste en congelar ciertos pesos de la red, de forma que no se actualicen al ser reentrenados. El *fine-tunning* puede dividirse en 3 tipos. El primero es útil cuando el nuevo conjunto de datos es extremadamente pequeño, y consiste en congelar todos los pesos de las capas convolucionales y eliminar los de las capas totalmente conectadas. El segundo tipo es similar al primero, pero en este, algunos de los pesos de las capas convolucionales se descongelan, permitiendo su reajuste, siendo su uso recomendable con conjuntos de datos moderados. Finalmente, si se posee un conjunto de datos suficientemente grande, se puede optar por descongelar todos los pesos de la red [50]. En la figura 2.7, se puede observar la aplicación de *transfer learning* y *fine-tunning*, donde las capas convolucionales de la nueva red coinciden con las de las de la red anterior y sus pesos pueden estar congelados o no. Mientras que, las capas totalmente conectadas (*Fully Connected Layers*) son creadas desde 0 para la nueva red.

3. GENERACIÓN DEL CONJUNTO DE DATOS DEL DEPORTE PÁDEL

Para realizar el entrenamiento de cualquier modelo de aprendizaje automático es necesario poseer un conjunto de datos. Dada la reciente popularidad del pádel, no existen bancos de imágenes clasificadas como en otros deportes como el tenis o bádminton. Por ello, el primer paso para la realización de este trabajo ha sido la generación de un conjunto de datos para el entrenamiento de los modelos de IA. En este capítulo, se describirá el proceso empleado para la recogida, etiquetado y procesamiento de los datos.

La calidad de este conjunto es un aspecto fundamental que marcará el límite de precisión superior del modelo [46]. Los factores que se han tenido en cuenta para obtener un adecuado conjunto de datos fueron: eliminar ruido en las etiquetas, datos heterogéneos, relevancia e interpretabilidad [51] [24]. Con estas medidas, se pretende conseguir un conjunto de datos consistente, que englobe la mayor parte de posibles situaciones que se puedan dar, incluyendo las más destacables y que este sea un fiel reflejo de la realidad. Por tanto, antes de comenzar su generación, se han tenido cuenta ciertas cuestiones que se discutirán a continuación.

Una de las principales características del aprendizaje profundo es el gran tamaño de los conjuntos de datos que se requieren. Esto se debe a que las redes de aprendizaje profundo pueden contener muchos parámetros y se necesitan muchas instancias para poder realizar un aprendizaje adecuado. La figura 3.1 [52] muestra como el rendimiento de los algoritmos de aprendizaje automático y aprendizaje profundo varían respecto a la cantidad de datos que se posee. Como se observa, en el aprendizaje profundo, la cantidad de datos es un aspecto fundamental para conseguir un buen modelo. Por ejemplo, en [53], los autores usaron técnicas de aprendizaje profundo para clasificar 300 millones de imágenes, y observaron que el rendimiento había aumentado logarítmicamente a medida que se incrementaba el conjunto de datos de entrenamiento. Con ejemplos como este, y varios estudios realizados, se ha llegado al consenso de que el rendimiento de los modelos de aprendizaje profundo sigue creciendo a medida que aumentan los datos siguiendo una ley potencial [52]. Aunque es necesario señalar, que también puede ocurrir que no porque el conjunto de datos sea mayor, el rendimiento del modelo deba que ser mejor. En un ejemplo que aparece en [54], sus autores utilizan CNN con un conjunto de datos de 100 millones de imágenes, y detectan que el aprendizaje se estanca a partir de 50 millones de imágenes.

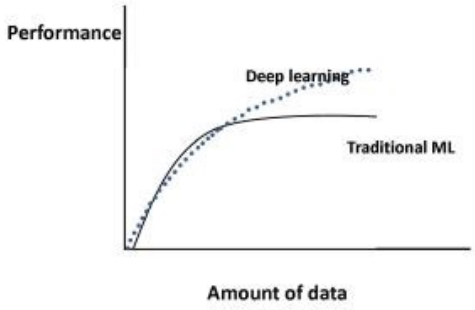


Figura 3.1 Rendimiento del aprendizaje automático y profundo según el tamaño del conjunto de datos [9]

Crear un conjunto de datos excesivamente grande puede conducir a otros problemas como la falta de memoria RAM, debido a que, durante el proceso de entrenamiento, las imágenes convertidas a vectores numéricos se almacenan en esta memoria, saturándola. La memoria RAM suele ser de menor tamaño que la de los discos de estado sólido (SSD) o los discos duros (HDD), por ello, no puede almacenar tantos datos a la vez. Este problema puede solucionarse reduciendo el tamaño de los lotes de aprendizaje, lo que implica que para realizar el ciclo de aprendizaje se tengan que almacenar menos instancias en la memoria RAM. Esto puede conducir otro problema, el aumento del tiempo de aprendizaje. Este tiempo aumenta, ya que la actualización de los pesos se efectúa cuando se termina de propagar cada lote de entrenamiento, por lo que al reducir el tamaño de estos lotes, el conjunto de datos posee más lotes y se producen muchas más actualizaciones de los pesos en cada ciclo de aprendizaje [55].

Otro posible factor que reduciría este elevado uso de la memoria es el tamaño de las imágenes, reduciéndolo, se podría elevar el número de fotogramas que puede almacenarse en la memoria RAM.

También se debe tener en cuenta que, si la red se encuentra ya entrenada con otros ejemplos, no sería necesario utilizar muchas instancias dado que los pesos de la red deben realizar pequeños reajustes (la red ya tiene información en sus pesos), no tan grandes como realizar el entrenamiento partiendo de pesos aleatorios. Como se ha mencionado en la sección 2.5, este proceso es conocido como *transfer-learning.* Estas consideraciones junto al hardware y tiempo disponible se han tenido en cuenta para la generación del conjunto de datos.

En la primera versión de *TrackNet* [31], los autores emplearon un conjunto de datos de 20,844 fotogramas de un mismo torneo más 16,118 fotogramas de diversas pistas, con el objetivo de aumentar la robustez del modelo, reduciendo la posibilidad de generar sobre ajuste y permitiendo reconocer la pelota de tenis en distintos entornos con mayor precisión. Además, se probó a aplicar *transfer-learning* para ajustar el modelo ya entrenado para su uso con videos de bádminton. Para ello, se reentrenó el modelo con 18,242 fotogramas procedentes de partidos de bádminton. En ambos conjuntos de datos, se dividieron en conjunto de entrenamiento (70%) y conjunto de prueba (30%).

Una vez estudiadas las anteriores consideraciones y teniendo en cuenta el marco de trabajo en la primera versión de *Tracknet*, se llegó a la conclusión de que, para obtener un buen desempeño del modelo, se debía generar un conjunto de datos similar en tamaño al utilizado para reentrenar *el modelo* con videos de bádminton (18,242 fotogramas). Respecto al tamaño de las imágenes, se empleó el mismo tamaño que el utilizado en TrackNetV2 (512x288 píxeles) [32] con el objetivo de no tener que modificar la red y tener que entrenarla desde cero. Además, se determinó empíricamente que, con los recursos de hardware disponibles, aumentar su tamaño no era factible en tiempo y almacenamiento.

La competición seleccionada para extraer los videos que formarán el conjunto de datos será WPT [56]. Este es el campeonato profesional de pádel más importante del mundo. En él, compiten los mejores jugadores de pádel, que recorren diferentes ciudades, realizando entre 15 y 20 torneos de esta competición cada año. Al ser WPT la principal competición de este deporte, se tomaron las distintas decisiones de la creación del conjunto de datos basándose en las características de sus torneos.

Según el reglamento de la Federación Internacional de Pádel: "El color del suelo debe ser solo uno, uniforme y claramente diferente del de las paredes, y preferencialmente verde, azul o pardo-terroso en sus variantes de tonalidades." [57]. Sin embargo, se analizaron los videos de todos los torneos de WPT del año 2022 y se determinó que en esta competición siempre se emplea un suelo de color azul, y en el 70% de las ocasiones el suelo externo a la pista era de color rojo, como se puede observar en la figura 4.2. Por esta razón, los videos seleccionados para componer el conjunto de datos se seleccionaron basándose en estas características, dado que el objetivo es obtener un sistema con un buen rendimiento en videos de este campeonato.

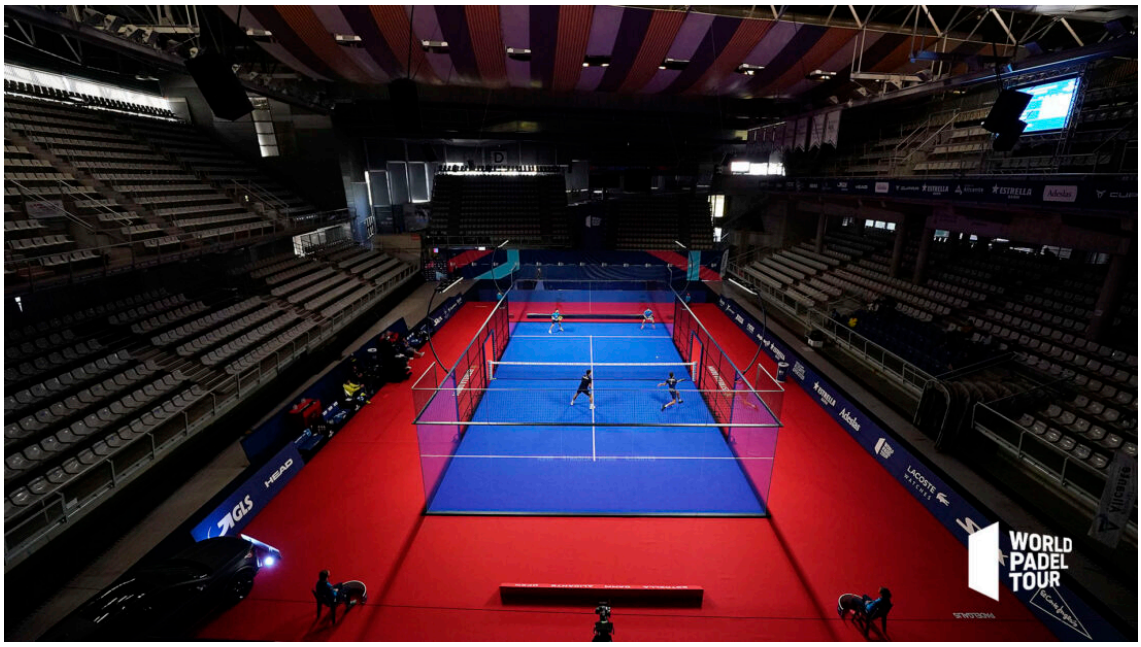
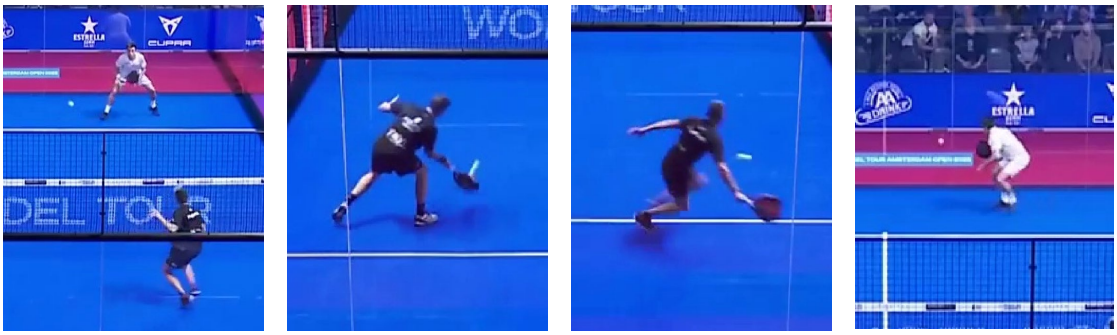


Figura 3.2 Pista de pádel del Estrella Damn Alicante Open 2020 [58]

Se han extraído 25 jugadas del canal oficial de WPT en youtube [59] procedentes de dos partidos: los cuartos de final del Open de Amsterdam 2022 y la final del Open de Santander 2022. En ambos torneos, el suelo de la pista era azul y el externo rojo. Las jugadas se han escogido con el objetivo de representar la mayor diversidad de situaciones. Se tomaron los videos de los partidos mencionados y se extrajeron varias jugadas, descartando las similares hasta llegar a la cifra de fotogramas esperada. En total, se recabaron 10 minutos y 50 segundos de video a 25 fotogramas por segundo (16,245 fotogramas). Se ha establecido la cifra de 25 fotogramas por segundo debido a que los videos de youtube estaban grabados a esta tasa de fotogramas. A pesar de que la red *TrackNet* se entrena con imágenes de 512x288 píxeles, para facilitar el etiquetado de las imágenes, inicialmente se emplearon fotogramas con resolución 1280x720 píxeles, para posteriormente, aplicar una transformación al conjunto de imágenes y etiquetas.

3.1 Descripción de las entradas y salidas de la red

Para la generación de los datos se han mantenido los atributos requeridos por *Tracknet*, generando un fichero csv que los contenga. Estos son: “Frame”, “Visibility”, “X” and “Y”. “Frame” es el nombre del archivo que contiene el fotograma. “Visibility” es un valor binario que indica si la pelota se encuentra en el fotograma (1) o no (0). Los atributos “X” e “Y” indican las coordenadas en la imagen donde se encuentra la pelota. Además de estas características, se extrajeron otras para facilitar futuras extensiones del modelo, o el entrenamiento de otras redes, que puedan predecir otros estados como: los botes de las pelotas en el suelo y cristal, los golpes de los jugadores, etc. Para ello, se diseñaron dos atributos más: “Status” y “Occluded”. “Status” indica el estado en el que se encuentra la pelota: volando (valor 0), golpeada (valor 1), botando en el suelo (valor 2) y rebotando en el cristal o verja (valor 3), como se puede observar en la figura 4.3. “Occluded” denota si la pelota se encuentra en el plano, pero está oculta detrás de algún elemento como un jugador (valor 1) o si es visible (valor 0).



Status = 0 *Status = 1* *Status =2* *Status = 3*

Figura 3.3 Imágenes extraídas de un partido de pádel [60]

Dado que el atributo "Occluded" puede generar cierta confusión en cuanto a la visibilidad de la pelota para la red, se optará por crear dos conjuntos de datos distintos. En uno de ellos, las pelotas ocultas se clasificarán como no visibles (visibility=0), y en el otro, se considerarán como visibles (visibility=1). Estos conjuntos de datos se denominarán “default” y “occluded” respectivamente. Este enfoque se lleva a cabo para evaluar cómo la red maneja la segmentación de las pelotas que no son visibles a simple vista.

En futuros capítulos, se discutirá cómo la red diseñada se entrenará dividiendo el conjunto de datos en tres partes: entrenamiento, validación y prueba. Durante esta división, es vital garantizar que los datos en cada conjunto sean representativos del conjunto de datos completo. Este paso es especialmente importante dado el desequilibrio de las clases en el problema que se aborda. Teniendo en cuenta esto, se divide el conjunto de datos según diferentes jugadas, con el objetivo de conseguir la mayor variabilidad en cada conjunto.

La proporción de la división es aproximadamente de 65% para entrenamiento, 15% para validación y otro 15% para pruebas. Para verificar la calidad de esta división, empleamos un modelo de las redes TrackNet que ha sido previamente entrenado para otros deportes. Esto nos permite comparar los resultados de la evaluación en cada conjunto y buscar consistencia. En la tabla 3.1 se proporciona un resumen de las métricas y los tamaños de estos conjuntos. En futuros capítulos, se analizarán y describirán con detalle estas métricas, así como el modelo utilizado para el entrenamiento.

TABLA 3.1
DESCRIPCIÓN Y MÉTRICAS DE LOS SUBCONJUNTOS DE DATOS

Conjunto	Entrenamiento	Validación	Prueba
Tamaño	11.023	2.568	2.986
	66.89%	14.97%	18.12%
Recall	74.41%	70.0%	75.87%
Specificity	18.21%	11.66%	26.23%
Precission	50.25%	43.01%	52.60%
Accuracy	54.67%	40.12%	51.99%
F1	31.51%	26.64%	31.06%

El conjunto de datos resultante cumple los factores que se determinaron inicialmente para obtener un conjunto de datos adecuado: ausencia de ruido en las etiquetas, datos heterogéneos y relevancia representando jugadas diversas, e interpretabilidad siendo los atributos extraídos fieles a la realidad.

3.2 Herramientas utilizadas para la generación de los datos

En la generación y etiquetado del conjunto de datos, se utilizaron y diseñaron una serie de herramientas que facilitaron las siguientes tareas: producir los fotogramas a partir de un video, etiquetar las imágenes, separar los atributos necesarios para entrenar la red *TrackNetV2* del resto y transformar los datos etiquetados a matrices numéricas para poder realizar el entrenamiento.

Para transformar los videos en fotogramas se ha diseñado un *script* (secuencia de comandos) utilizando el lenguaje de programación Python. Este programa, recibe como entrada la ruta de un video y la ruta donde se guardarán los fotogramas que componen el video. Mediante el uso de la librería OpenCV [61] se extraen los fotogramas guardándolos en la ruta especificada.

Respecto al etiquetado de las imágenes, se ha hecho uso del programa *LabelingTool.exe* disponible en el repositorio de GitHub de una de las implementaciones de la red TrackNet [62]. Esta herramienta está programada en el lenguaje MATLAB y tiene una interfaz gráfica que permite etiquetar los fotogramas de forma eficiente (figura 4.4). El programa posibilita desplazarse por las distintas imágenes contenidas en la ruta especificada utilizando las teclas de desplazamiento. También, permite determinar la posición de la pelota haciendo click sobre ella y estableciendo los parámetros *Visibility* y *Status* seleccionándolos en el menú de la derecha. La salida que produce la herramienta es un archivo csv que contiene las siguientes columnas: *Frame*, *Visibility*, *X*, *Y*, *Status* y *Ocluded*.

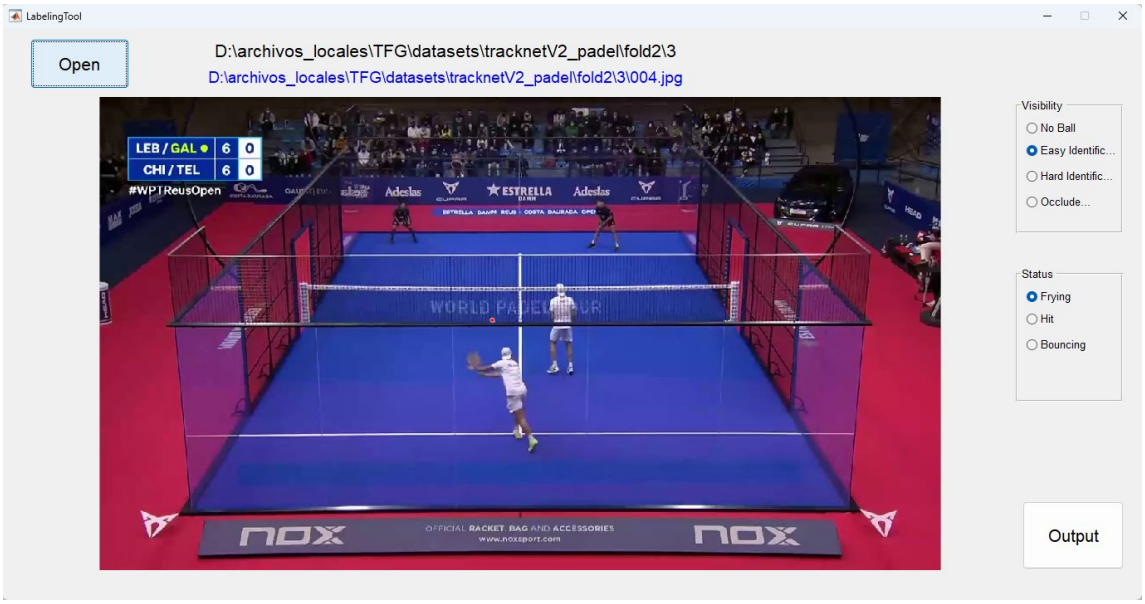


Figura 3.4 Interfaz gráfica del programa LabelingTool.exe [62]

Sin embargo, los atributos extraídos por el programa *LabelingTool.exe* no coinciden con los utilizados por las redes TrackNet. En consecuencia, se ha decido asociar el estado *Bouncing* a los botes en el suelo y la visibilidad *Hard Identification* al bote en los cristales. Para conseguir un conjunto de datos con el formato esperado, se ha programado un *script* en Python que procese las salidas de la herramienta de etiquetado (*relabeling.py*). Este *script*, recibe como entrada un archivo en formato csv con las salidas etiquetadas y produce 2 ficheros: *FullLabel.csv*, en el que se encuentran todos los atributos etiquetados en su correcto formato, y *TNLabel.csv*, en el que sólo aparecen los atributos necesarios para entrenar la red TrackNetV2 [63]. El resultado de estas transformaciones puede observarse en la figura 3.5, en la que se muestra el fichero salida de *LabelingTool.exe* y los dos ficheros resultados de ejecutar el *script relabeling.py*.

file name	visibility	x-coordinate	y-coordinate	status
000.jpg	1	709	189	0
001.jpg	1	711	215	0
002.jpg	1	711	212	0
003.jpg	1	711	237	0
004.jpg	1	712	266	0
005.jpg	1	712	295	0
006.jpg	1	711	325	1
007.jpg	1	724	323	0
008.jpg	1	740	308	0
009.jpg	1	754	294	0
010.jpg	1	765	282	0
011.jpg	1	774	273	0
012.jpg	1	783	265	0
013.jpg	1	791	258	0
014.jpg	1	795	253	0

Label.csv

Frame	Visibility	X	Y	Status	Occluded
0	1	709	189	1	0
1	1	711	215	1	0
2	1	711	212	1	0
3	1	711	237	1	0
4	1	712	266	1	0
5	1	712	295	1	0
6	1	711	325	2	0
7	1	724	323	1	0
8	1	740	308	1	0
9	1	754	294	1	0
10	1	765	282	1	0
11	1	774	273	1	0
12	1	783	265	1	0
13	1	791	258	1	0
14	1	795	253	1	0

FullLabel.csv

Frame	Visibility	X	Y
0	1	709	189
1	1	711	215
2	1	711	212
3	1	711	237
4	1	712	266
5	1	712	295
6	1	711	325
7	1	724	323
8	1	740	308
9	1	754	294
10	1	765	282
11	1	774	273
12	1	783	265
13	1	791	258
14	1	795	253

TNLabel.csv

Figura 3.5 Formatos de las distintas salidas

Finalmente, el repositorio de TrackNetV2 [63] incluye un *script* de Python llamado *gen_data3.py*. Este fichero se encarga de transformar los fotogramas etiquetados en las matrices numéricas con las que se entrenará la CNN. Para ejecutar este programa, se deben especificar los siguientes argumentos: directorio donde se encuentran las imágenes, ruta del fichero que contiene las etiquetas, tamaño de los lotes para entrenar la red y directorio donde se guardan las matrices numéricas.

Debido a que la red TrackNetV2 [63] tiene una entrada de 3 fotogramas, esta se considera la unidad mínima de los lotes de entrenamiento (instancia). A su vez, cada fotograma estará compuesto de 3 matrices numéricas debido a que se utilizan imágenes en color, una por cada color primario (rojo, verde y azul). Por ejemplo, si se utiliza un tamaño de lote de 4, cada lote contendrá 4 instancias, 12 fotogramas o 36 matrices numéricas. El *script* que convierte los fotogramas en matrices y los agrupa en lotes, *gen_data3.py*, selecciona los grupos de 3 imágenes utilizando un paso de 1. Es decir, la primera instancia está compuesta por los fotogramas 1, 2 y 3, la segunda instancia por los fotogramas 2, 3 y 4, y así, sucesivamente. Esto implica que un mismo fotograma se encontrará en 3 instancias distintas, aumentando el tamaño del conjunto de entrenamiento.

4. MODELO DE APRENDIZAJE PROFUNDO Y ENTRENAMIENTO DE LA RED

Para este proyecto se ha utilizado como base el código de TrackNetV2 [63]. El repositorio publicado por la Universidad Nacional Chiao Tung de Taiwan incluye los ficheros necesarios para poder utilizar la red, entre ellos: generación del conjunto de datos, arquitectura de la red, entrenamiento, evaluación del modelo y predicción de videos utilizando el modelo.

En este capítulo se detallará en profundidad los diversos aspectos de esta implementación de la red, así como las modificaciones realizadas para su correcto funcionamiento en el ámbito del deporte pádel.

4.1 Requisitos de hardware y software

Esta versión de Tracknet está implementada utilizando la librería de software de código abierto TensorFlow, que se utiliza ampliamente para la implementación de algoritmos de aprendizaje profundo, como las CNN [64]. Esta librería ofrece una plataforma eficaz y adaptable para diseñar, entrenar e implementar modelos de aprendizaje automático, facilitando la experimentación y el desarrollo de soluciones en inteligencia artificial [65]. TensorFlow resulta particularmente útil en el diseño de CNN debido a su habilidad para gestionar de manera eficiente operaciones matriciales y tensoriales de gran envergadura, que son esenciales en el cómputo de las capas convolucionales y de agrupación. Asimismo, TensorFlow brinda soporte para aceleración mediante hardware a través de GPU y TPU, permitiendo un entrenamiento y evaluación más veloz de los modelos de redes neuronales [66].

A fin de aprovechar la aceleración por hardware en TensorFlow, es necesario utilizar una GPU NVIDIA con arquitecturas Pascal, Volta, Turing o Ampere [67]. Para el entrenamiento del modelo que se describirá más adelante, se empleó la GPU NVIDIA GeForce RTX 3060 V2 de 12 GB GDDR6, un dispositivo de gama media lanzado en febrero de 2021 y basado en la arquitectura Ampere [68]. La RTX 3060 V2 está equipada con 12 GB de memoria GDDR6 y 3584 núcleos CUDA. Además, cuenta con 112 núcleos tensor, unidades especializadas diseñadas para acelerar las operaciones de inteligencia artificial y aprendizaje profundo, en particular las multiplicaciones de matrices y las convoluciones [69].

Dado que el conjunto de datos consiste en matrices numéricas que representan imágenes y requieren una cantidad considerable de espacio (33 MB por lote, si el tamaño del lote es de 4), la memoria integrada en la GPU resulta crucial para el rendimiento del modelo.

TensorFlow es compatible con diversos sistemas operativos, como Ubuntu, Windows y macOS [70]. No obstante, se recomienda emplear Ubuntu 16.04 o versiones posteriores para lograr una configuración óptima y un mejor rendimiento. Para este proyecto, se ha optado por utilizar el sistema operativo Windows 10.

A fin de que TensorFlow pueda utilizar el hardware de NVIDIA, es imprescindible instalar los controladores de la GPU, CUDA (*Compute Unified Device Architecture*) y cuDNN (*NVIDIA CUDA Deep Neural Network library*). Para garantizar un correcto funcionamiento de esta integración, es necesario emplear versiones específicas de cada software, las cuales se detallan en la tabla 4.1 [71].

TABLA 4.1
VERSIONES DEL SOFTWARE UTILIZADO [71]

Software	Versión
Python	3.10
TensorFlow	2.11
CUDA	11.2
cuDNN	8.1

Asimismo, se han empleado las últimas versiones de las siguientes bibliotecas de python, las cuales no presentan incompatibilidades: numpy, piexif, keras, pandas, opencv-python y matplotlib.

4.2 Descripción del código de TrackNetV2

El repositorio se divide inicialmente en los modelos de una salida y tres salidas. Este trabajo se centrará en el modelo de tres salidas ya que con él se obtienen mejores resultados y la red es más rápida como se mostró en las tablas 2.2 y 2.3 [31] [32].

Respecto a los *scripts* relacionados con la generación del conjunto de datos, destaca el programa *gen_data3.py*, que a partir de un conjunto de datos etiquetado genera matrices de valores decimales de 32 bits que representan las imágenes y las salidas de la red. Hay que tener en cuenta que una instancia de entrenamiento está formada por 3 fotogramas consecutivos en color (RGB), es decir, que cada instancia contiene 9 matrices numéricas. Además, este script se encarga de redimensionar las imágenes y agruparlas en lotes. La entrada del programa está formada por los fotogramas generados y un archivo csv que contenga la posición y visibilidad de la pelota en cada fotograma acorde al formato mostrado en la figura 3.5 (*TNLabel.csv*). La salida está compuesta por dos tipos de ficheros, la matriz que representa las imágenes de la instancia de entrenamiento y la matriz que indica la posición donde se encuentra la pelota en cada fotograma.

En el fichero *TrackNet3.py* se define la arquitectura de la CNN. En él se incluyen elementos como: tipo de capa, tamaño del *kernel*, número de filtros, *padding*, etc. Para diseñar la arquitectura de la red, se utiliza Keras, una biblioteca de código abierto que simplifica el uso de Tensorflow.

La biblioteca TensorFlow permite el uso de la transferencia de aprendizaje. Posee funciones que congelan o desbloquean los pesos de cada capa, elimina o crea capas, etc [72]. En el código utilizado para realizar el entrenamiento de la red se dispone la opción de cargar los pesos de un modelo ya entrenado. En este caso, no se congelan los pesos de

ninguna capa, ni se eliminan capas totalmente conectadas, ya que la red TrackNet no posee este tipo de capas.

El *script* para realizar el entrenamiento de la red, *train_TrackNet3.py*, se encarga de entrenar una CNN dado un conjunto de entrenamiento. Al comienzo del entrenamiento, se decide si entrenar el modelo desde cero o cargar los pesos de un modelo previamente entrenado. La biblioteca TensorFlow facilita la implementación de transferencia de aprendizaje, proporcionando funciones para congelar o desbloquear pesos de cada capa y para eliminar o crear capas [72]. En este caso, no se congelan los pesos de ninguna capa ni se eliminan capas totalmente conectadas, ya que la red TrackNet no posee este tipo de capas. Luego, se entrena la red para cada lote del conjunto de datos utilizando una función de coste personalizada que se describirá en la sección 4.4. Este proceso se repite para el número de épocas definido. Al final de cada época, se evalúa el modelo utilizando las métricas precisión, exhaustividad (*recall*) y exactitud, así como el número de verdaderos y falsos positivos y negativos. Esto se realiza generando las predicciones del conjunto de entrenamiento y decidiendo si la salida es correcta utilizando un valor de tolerancia, que establerá el número de pixeles de diferencia que se permitirán entre la salida de la red y las etiquetas de los datos para establecer el acierto de la red. Cabe destacar que esta implementación de TrackNet no emplea validación cruzada ni conjunto de validación en el entrenamiento. Otro factor importante es la utilización del optimizador Adadelta, que adapta la tasa de aprendizaje, basándose en la magnitud de las actualizaciones recientes.

Para generar la salida del modelo sobre un conjunto de datos se dispone del fichero *predict3.py*, que recibe como entrada un video y un modelo ya entrenado. La salida de la ejecución de este *script* está compuesta por las predicciones en el mismo formato que se utiliza para la generación del conjunto de entrenamiento y un video en el que se muestra la predicción. En este video, se marca en cada fotograma un punto rojo dónde el modelo ha predicho la posición de la pelota, como se muestra en la figura 4.1.

Figura 4.1 Fotograma extraído del partido de Lebrón y Galán contra Chingotto y Tello de Reus de 2022 con la predicción del modelo final obtenido [73].

4.3 Modificaciones implementadas sobre el código base

Con el objetivo de mejorar la implementación de la red se han implementado una serie de modificaciones sobre el código base. Primeramente, se han diseñado una serie de *scripts* en Python para realizar tareas repetitivas, como calcular las distintas métricas dado un conjunto de predicciones y sus etiquetas con *detect_FN_script.py* y *get_metrics_script.py*, así como *scripts* para generar el conjunto de datos (ficheros .npy) de varias jugadas de pádel a la vez con *group_data_batch_script,py* y *gen_data_script.py*.

Para seleccionar el mejor modelo de entre un conjunto de modelos entrenados, se ha implementado un proceso de validación durante el entrenamiento de la red. La incorporación de la evaluación de la red en un conjunto de validación (diferente al de entrenamiento) durante el aprendizaje es la metodología más adecuada para parar decidir el número de ciclos óptimo y poder prevenir el sobreaprendizaje, por lo que se consideró de especial interés incluir esta modificación en el código base. Esto implicó una modificación del código del fichero *train_TrackNet3.py* para incorporar esta mejora. El código revisado ahora prevé, al final de cada época, generar las predicciones del conjunto de validación e invocar la función de pérdida. Esto permite la generación de un gráfico comparativo de estos valores de pérdida (entrenamiento y validación) a través de las épocas de aprendizaje (Figura 4.2).

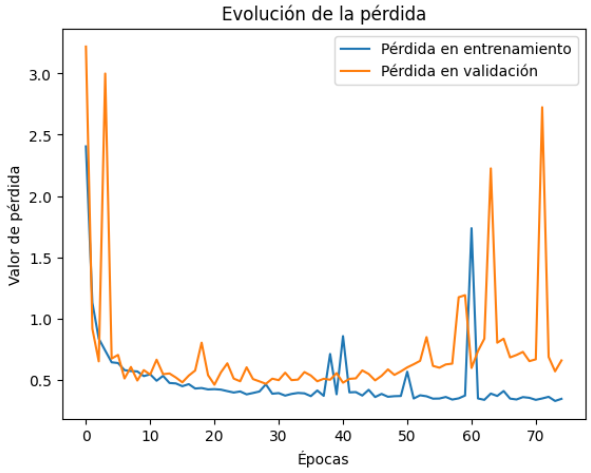


Figura 4.2 Evolución del valor de pérdida a través de las épocas sobre el conjunto de entrenamiento y validación.

Además, y debido a que uno de los principales objetivos del trabajo es realizar un estudio comparativo de diferentes funciones de pérdida, se propondrán varias funciones para evaluar su rendimiento y eficacia. Estas incluyen la función de pérdida entropía cruzada binaria ponderada proporcionada en el código base de TrackNetV2 [32], así como una combinación de entropía cruzada binaria y pérdida de dados propuesta en el artículo MonoTrack [33]. También se tomarán en cuenta funciones de pérdida ampliamente utilizadas para problemas similares, como la pérdida focal [74] y una variante de la entropía cruzada ponderada basada en mapas de calor [75]. En la sección 4.4 se detallará en profundidad cada una de ellas, así como sus ventajas y desventajas.

Para realizar una comparativa de estas diferentes funciones de pérdida, es necesario calcular otras métricas para facilitar esta comparación. Así, durante la predicción de los conjuntos de validación y entrenamiento, se evalúan las siguientes métricas: sensibilidad, especificidad, precisión, precisión, valor f1, tasa de verdaderos positivos y tasa de verdaderos negativos. Estas medidas, se calculan al final de cada época, de modo que también se pueden obtener gráficos que muestran su evolución a lo largo de las épocas para los conjuntos de entrenamiento y validación (Figura 4.3).

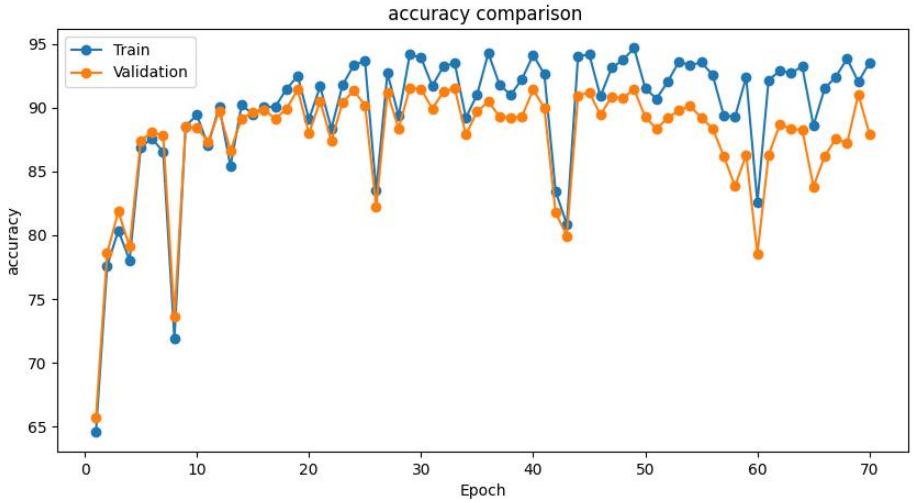


Figura 4.3 Comparación de la exactitud de los conjuntos de validación y entrenamiento para un experimento de la red.

4.4 Función de pérdida

La elección de la función de pérdida representa una de las cuestiones más debatidas en el contexto de la red TrackNet [31]. En cada versión estudiada de la red (TrackNet, TrackNetV2, MonoTrack), se emplea una función de pérdida diferente. Los autores de los artículos sobre TrackNetV2 [32] y MonoTrack [33] sugieren que una posible mejora de la red podría ser la identificación de una función de pérdida más efectiva. Entre los problemas asociados con las funciones de pérdida actuales se incluyen la inestabilidad de la convergencia y el manejo de clases altamente desbalanceadas. Esto se debe a la predominante cantidad de píxeles que se clasifican como fondo de imagen en comparación con aquellos que contienen parte del objeto clasificado.

Estas funciones se integran en el proceso de entrenamiento como funciones de pérdida personalizadas. Cada función se aplica a cada píxel de las imágenes, contribuyendo a la pérdida total. Tras entrenar cada lote, se calcula el promedio de las pérdidas, dividido por la suma del tamaño de los lotes, las dimensiones de la imagen y el número de capas que contiene una imagen (tres en el caso de las imágenes a color). Este valor se agrega a los valores de pérdida del resto de los lotes, y al final de cada época se obtiene la media dividiéndolo por el número de lotes. De este modo, al finalizar el entrenamiento completo, se genera una lista con el valor promedio de pérdida de cada época.

A continuación, se describirán tanto las funciones de pérdida implementadas en las redes TrackNet como otras propuestas basadas en el estado del arte actual. A excepción de la función de pérdida preexistente en el código base (4.1), las demás funciones han sido diseñadas e implementadas utilizando las funciones matemáticas proporcionadas por las bibliotecas *Keras* y *Tensorflow*, sin recurrir a implementaciones previas. Posteriormente, estas funciones serán evaluadas, con el propósito de identificar la más adecuada para el problema que se está abordando en el marco del deporte pádel. Para simplificar su comprensión, utilizaremos la misma notación en todas ellas, donde $\hat{y}$ representa el valor real de la etiqueta de la instancia (*ground truth*) e y la predicción del modelo. Las etiquetas y predicciones poseen un valor entre 0 y 1 que representa la probabilidad de que exista una pelota en cada posición de la imagen.

La función de pérdida que los autores de TrackNetV2 [32] describen y que está implementada en el código base, se basa en la entropía cruzada binaria ponderada (WBCE) definida por la ecuación 4.1, donde se utiliza un coeficiente de ponderación. Este coeficiente aplica una ponderación adicional basada en las predicciones del modelo, en lugar de aplicar simplemente un peso basado en las etiquetas de clase. Estas ponderaciones adicionales varían entre $(1 - y)^2$ y y^2, lo que significa que la función de pérdida aumentará cuando el modelo se equivoque en su predicción, reduciendo en parte el problema de desbalanceo de clases.

$$WBCE = -[(1-y)^2 \hat{y} \log(y) + y^2 (1-\hat{y}) \log(1-y)] \tag{4.1}$$

En la red MonoTrack [33], se utiliza una combinación de varias funciones de pérdida: entropía cruzada binaria (4.2) y *Dice Loss* (4.3). La función *Dice Loss* mide la similitud entre dos muestras. En el caso de la segmentación de imágenes, mediría la superposición entre la predicción del modelo y etiqueta verdadera, siendo el resultado 1 para una superposición perfecta y 0 para ninguna superposición. Esta combinación (4.4) permite reducir el desbalanceo de clases utilizando una aproximación distinta a la de la entropía cruzada binaria.

$$\text{BCE} = -(\hat{y} \log(y) + (1-\hat{y}) \log(1-y)) \tag{4.2}$$

$$Dice(y, \hat{y}) = \frac{2y\hat{y}+\epsilon}{||y||+||\hat{y}||+\varepsilon} \tag{4.3}$$

$$BCE\ Dice = (1-\alpha)\ BCE(y, \hat{y}) + \ \alpha\ DICE(y, \hat{y}) \tag{4.4}$$

La pérdida focal (*Focal Loss*) [74], presentada en la fórmula 4.6, se diseñó para abordar el desequilibrio entre las clases en la detección de objetos. Puede mejorar la detección de objetos pequeños al reducir el impacto de las clasificaciones erróneas de los fondos (clase mayoritaria). Sin embargo, su rendimiento puede ser sensible a la elección de los hiperparámetros (α y γ). El término p_t representa la probabilidad del modelo de que la

etiqueta sea correcta y α_t es un coeficiente utilizado para manejar el desequilibrio de clases, dando más peso a las clasificaciones de la clase minoritaria. Según el valor que se establezca para el parámetro α, se estará dando más peso a los ejemplos negativos (α < 0.5) o a los positivos (α > 0.5). En la figura 4.4 se muestra como el parámetro γ afecta al valor de pérdida. Este parámetro controla la tasa de reducción de la contribución de los ejemplos fáciles a la pérdida total. Con un valor mayor de γ los ejemplos fáciles tendrán una reducción aún mayor en su contribución a la pérdida total, mientras que, con un valor de 0, la función de pérdida funcionará como la entropía cruzada.

$$\begin{aligned} Focal\ Loss &= -\alpha_t * (1 - p_t)^{\gamma} * \log(p_t) \\ p_t &= \hat{y} * y + (1 - \hat{y}) * (1 - y) \\ \alpha_t &= \hat{y} * \alpha + (1 - \hat{y}) * (1 - \alpha) \end{aligned} \tag{4.6}$$

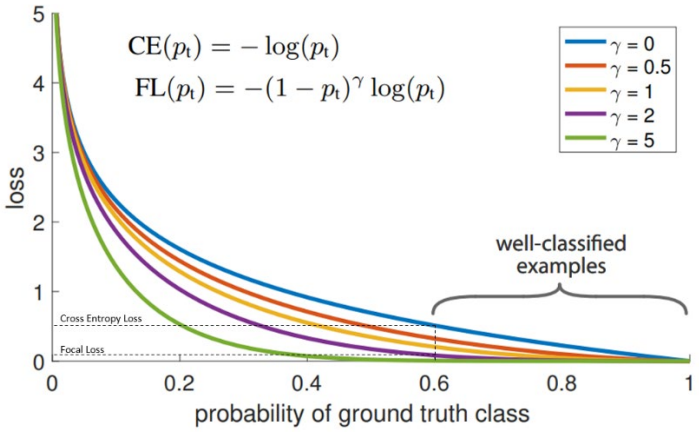


Figura 4.4 Contribución de la pérdida según el valor del parámetro gamma. [65]

La función 4.7, que denominaremos *Heat Maps*, es una variante de la entropía cruzada binaria [75], diseñada para manejar desequilibrios de clase en problemas de clasificación binaria. En este contexto, cuando $\hat{y}$ es igual a 1, significa que existe una pelota de pádel en esa posición específica. Esta función utiliza el concepto de "máscaras" o "mapas de calor" para las predicciones de la detección de la pelota (positivas) y las de no detección (negativas). La "máscara positiva" (*pos_mask*) es un mapa binario donde todas sus posiciones son 0, excepto donde $\hat{y}$ es 1, es decir, donde se ha detectado la pelota. De forma similar, la "máscara negativa" (*neg_mask*) es un mapa binario donde sus valores son 1 si $\hat{y}$ es menor que 1, indicando que la pelota no se detectó en esa posición, y 0 en caso contrario. A continuación, se calculan de manera separada las pérdidas para las predicciones de detección (*pos_loss*) y no detección (*neg_loss*) de la pelota, asignando un peso mayor a las posiciones donde se esperaba detectar la pelota pero no se hizo (falsos negativos) mediante el término de peso $(1 - \hat{y})^4$. La pérdida total se obtiene sumando las pérdidas positivas y negativas, pero si hay al menos un ejemplo positivo, la pérdida total se normaliza por el número de detecciones correctas (*N_pos*). En caso de que no se haya

detectado ninguna pelota, la pérdida total será simplemente la pérdida negativa. En resumen, esta función de pérdida está diseñada para minimizar los errores de detección de la pelota y dar más importancia a los aciertos en situaciones de desequilibrio de clases.

$$
\begin{aligned}
pos\_loss &= -(\log(y)\,(1-y)^2 \text{pos\_mask}) \\
neg\_loss &= -(\log(1-y)\,y^2(1-\hat{y})^4 \times \text{neg\_mask}) \\
Heat\ Maps &= \begin{cases} \dfrac{pos\_loss + neg\_loss}{N_{pos}} & si\ N_{pos} > 0 \\ neg\_loss & si\ N_{pos} = 0 \end{cases}
\end{aligned}
\tag{4.7}
$$

Además, con el objetivo de poder comparar y evaluar la mejora de las distintas funciones de pérdida basadas en la entropía cruzada binaria ponderada, WBCE (4.1) y *Heat Maps* (4.7), se ha decidido implementar la función de entropía cruzada binaria sin ponderaciones definida en la ecuación 4.2.

4.5 *Dropout y skip-connections*

En esta sección, se explican dos modificaciones que se implementarán sobre la arquitectura de la CNN para posteriormente evaluar su eficacia. El objetivo principal de dichas modificaciones es optimizar la precisión de la red en la detección de pelotas de pádel, así como incrementar su eficiencia, entendida como la capacidad para realizar propagaciones de la red en un tiempo más reducido.

La incorporación de técnicas como el *dropout* puede mejorar el desempeño de la red y mitigar el sobreajuste, un problema habitual en el entrenamiento de redes neuronales profundas. El *dropout* es un método que, aleatoriamente, desactiva ciertas neuronas durante el proceso de entrenamiento, previniendo así la dependencia excesiva de la red de cualquier neurona en particular y fomentando una mejor generalización [76]. Además, al reducir el número de pesos en la red, se puede incrementar la velocidad de la misma.

En lo que respecta a la CNN que se desarrolla en este trabajo, no contiene capas completamente conectadas, donde generalmente se aplica el *dropout*. Sin embargo, en estudios como [84], se ha demostrado que aplicar *dropout* antes de la capa de *MaxPooling* puede arrojar resultados positivos.

En cuanto a la inclusión de las *skip-connections*, éstas se han incorporado tomando como referencia el diseño de la red MonoTrack, que se describió en la sección 2.4.3 de este documento. Este tipo de conexiones contribuyen a mitigar el problema de degradación del gradiente durante el entrenamiento de redes neuronales profundas, tal como se explica en la sección 2.4.2. En el marco de nuestro modelo, las *skip-connections* se sitúan entre la primera y última capa de cada nivel de la red, tal como se puede observar en la Figura 2.6.

5. EXPERIMENTOS Y ANÁLISIS DE RESULTADOS

En este capítulo se describe el proceso llevado a cabo para efectuar los diversos experimentos requeridos para obtener el modelo final, capaz de detectar pelotas de pádel de manera efectiva y eficiente. Para cada experimento se detallarán los parámetros empleados y se discutirán los resultados obtenidos, así como su impacto en el rendimiento de la red.

5.1 Contexto de los experimentos

En el capítulo dedicado a la generación del conjunto de datos, se establecieron tres categorías para la visibilidad de la pelota: no visible, oculta y visible. Sin embargo, la clasificación de la red es binaria, atribuyendo el valor 1 a las pelotas visibles y el 0 a las no visibles. Se generan así dos conjuntos de datos distintos. En el primero, se considera a las pelotas ocultas como no visibles, y se ha nombrado a este conjunto como **"default"**. El segundo conjunto de datos clasifica el estado de oculto como visible, proporcionando un medio para evaluar el rendimiento de la red en la detección de pelotas ocultas, un aspecto comentado en los artículos de las redes TrackNet [31] [33] [32] debido a la dependencia temporal que presenta la red. Este conjunto de datos se identifica con el nombre de **"occluded"**.

Como se mencionó en la sección 3.1, la elección del tamaño del lote en el entrenamiento de CNN conlleva varias implicaciones, dado que determina cuántas muestras se procesan antes de actualicen los pesos. En cuanto a la elección del tamaño del lote para los conjuntos de datos, se buscó maximizar este parámetro sin sobrepasar la capacidad de memoria de la GPU encargada del entrenamiento. En el caso del conjunto de datos "default", se eligió un tamaño de lote de 5 instancias, lo que equivale a un total de 15 fotogramas. Para minimizar el uso de memoria en los experimentos con el conjunto "occluded", en los que se emplearon un mayor número de épocas, se redujo el tamaño del lote a 4 instancias, resultando en 12 fotogramas por lote.

El proceso para evaluar las diversas modificaciones propuestas para la red se desarrolla de la siguiente manera. En primer lugar, se seleccionará la función de pérdida que arroje los resultados más prometedores. Una vez definida, dicha función permanecerá fija para los siguientes experimentos, que contemplarán la implementación de *dropout* y *skip-connections*. Si alguna de estas modificaciones mejora el rendimiento de la red, se integrarán en el modelo final. Todos estos experimentos se llevan a cabo utilizando el conjunto de datos denominado "default". Finalmente, se evaluará el modelo definitivo con el conjunto de datos "occluded", con el objetivo de evaluar el rendimiento de la red cuando la pelota se encuentra oculta. Debido al tiempo que requiere cada experimento, que oscila entre 40 y 70 minutos por época, no se lleva a cabo una búsqueda exhaustiva de hiperparámetros o distintas combinaciones de los mismos. Sin embargo, se opta por los valores más frecuentemente utilizados o se justifica la elección de determinados valores.

Como se mencionó previamente en el capítulo dedicado a la generación del conjunto de datos, este se divide en tres subconjuntos: entrenamiento, validación y prueba. Para determinar el mejor modelo de cada experimento, se seleccionará el modelo de la época que presente la menor pérdida en el conjunto de validación, y los distintos experimentos se compararán utilizando métricas obtenidas en el conjunto de prueba. En la figura 5.1, se muestra la evolución de las pérdidas de entrenamiento y validación a lo largo de las épocas.

En la figura 5.1 se observa que, a partir de la época 60, se presentan grandes oscilaciones en el valor de la pérdida de validación. Este fenómeno no es sorprendente ya que en el trabajo TrackNet [31] se presenta una gráfica con oscilaciones similares (Figura 2.4). Estas fluctuaciones pueden deberse a diversos factores. Un motivo clave podría ser el uso de un conjunto de datos desbalanceado, como el utilizado en este caso, que puede causar oscilaciones en la función de pérdida. Esto se debe a que el modelo tiende a sobreajustarse a la clase más numerosa, dificultando la minimización de la pérdida en la clase menos representada [78]. Por otro lado, un tamaño de lote demasiado pequeño puede llevar a estimaciones del gradiente más ruidosas, lo que puede hacer que el valor de pérdida oscile [79]. En el caso de esta red, no se puede aumentar mucho más por las restricciones de memoria del equipo donde se entrena el modelo. Finalmente, otra posibilidad es que, al tratarse de una red con una alta complejidad (11.334.531 parámetros), esta pueda adaptarse en exceso al conjunto de entrenamiento, causando sobreajuste. Como se evidencia, este problema puede estar relacionado con varias causas, las cuales necesitarían de un análisis exhaustivo. Sin embargo, al seleccionar el modelo con la menor pérdida en validación, se está evitando la elección de estos modelos que presentan "defectos" o anomalías en su rendimiento.

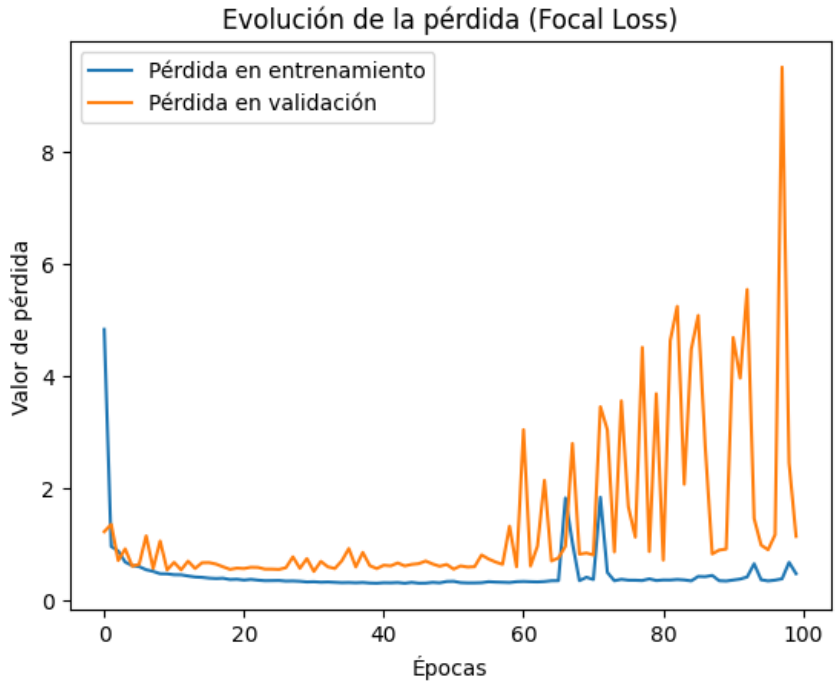


Figura 5.1 Evolución del valor de pérdida de entrenamiento y validación utilizando la función de pérdida Focal Loss

La evaluación de la CNN se lleva a cabo mediante varias métricas para medir su rendimiento en la correcta detección de pelotas en fotogramas de pádel. A continuación, se describirá cada una de ellas explicando qué información aportan junto a la forma de calcularas. En las ecuaciones que describen las distintas métricas se utiliza la siguiente nomenclatura: verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN). Además, en las tablas de resultados, se encuentra una distinción adicional entre los falsos negativos: FP1 se refiere a las predicciones que detectan una pelota en una ubicación incorrecta, mientras que FP2 representa las predicciones que detectan una pelota cuando en realidad no hay ninguna.

La métrica de *recall* (Sensibilidad) (5.1) mide la cantidad de instancias positivas que el modelo ha conseguido detectar sobre todas las positivas existentes. *Specificity* (Especificidad) (5.2) es similar a *recall*, pero en este caso se centra en las instancias negativas que ha conseguido detectar correctamente. *Precision* (Precisión) (5.3) proporciona información sobre la proporción de identificaciones positivas que fueron efectivamente correctas. *Accuracy* (Exactitud) (5.4) mide la proporción total de predicciones correctas, tanto positivas como negativas. *F1 Score* (5.5) es una métrica que combina *precision* y *recall* en una sola cifra. En contextos en los que se quiere equilibrar tanto la *precision* como el *recall*, el *F1 Score* es una excelente métrica para evaluar el rendimiento del modelo, por lo que será altamente considerada en este análisis. En conjunto, estas métricas proporcionan una visión completa del rendimiento, para así poder realizar una comparativa entre diferentes modelos sobre el mismo conjunto de datos [80].

$$Recall = \frac{TP}{TP + FN} \tag{5.1}$$

$$Specificity = \frac{TN}{TN + FP} \tag{5.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.3}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5.4}$$

$$F1 = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \tag{5.5}$$

Con el propósito de establecer un punto de referencia para comparar las mejoras logradas con la red TrackNetV2 al entrenarse con un conjunto de datos específico de pádel, se cuenta con un modelo preexistente en el código base [63]. Este modelo fue entrenado previamente con imágenes de partidos de bádminton y tenis. Este servirá como patrón comparativo para evaluar el rendimiento de nuestra red al procesar el conjunto de prueba de pádel. Además, este modelo también servirá para inicializar los pesos de la red y poder aplicar transferencia de aprendizaje. En la tabla 5.1 se presentan las métricas resultantes de la evaluación del modelo con el conjunto de prueba de ambos conjuntos de datos,

denominados "default" y "occluded". Entre las observaciones que se pueden hacer sobre estos conjuntos, destaca el gran número de detecciones de pelotas en posiciones incorrectas (FP1).

TABLA 5.1
EVALUCACIÓN DEL MODELO BASE SOBRE LOS CONJUNTOS DE DATOS "DEFAULT" Y "OCLUDED"

Conjunto	[TP, TN, FP1, FP2, FN]	Recall	Specifity	Precision	Accuracy	F1
Default	[1173 376 944 113 373]	75.87%	26.23%	52.60%	51.99%	31.06%
Occluded	[1174 344 965 91 405]	74.35%	24.57%	52.64%	50.95%	30.82%

La arquitectura inicial de la red que se empleará en los experimentos se presenta en la Figura 2.5. Con respecto a los hiperparámetros más relevantes, establecemos la tasa de aprendizaje en 1, siguiendo la recomendación de los autores del optimizador Adadelta basada en varios experimentos [81].

5.2 Estudio de las diferentes funciones de pérdida

Para evaluar las diferentes funciones de pérdida, se lleva a cabo una serie de experimentos en las mismas condiciones, utilizando una función de pérdida distinta en cada uno. Posteriormente, se selecciona el mejor modelo obtenido, el correspondiente a la época con el menor valor de pérdida sobre el conjunto de validación. Finalmente, se calculan las métricas de cada modelo para el conjunto de prueba. Estos experimentos se realizan sobre el conjunto de datos "default" a lo largo de 75 épocas. Este número se ha determinado considerando el tiempo que se necesita para entrenar el modelo durante una época y la cantidad de experimentos que se planean realizar. Además, basándose en observaciones de otros experimentos realizados con 100 épocas, los mejores modelos se suelen obtener entre las épocas 20 y 40.

Algunas funciones de pérdida requieren la configuración de ciertos hiperparámetros. Para la función *Focal Loss*, se establecen los valores $\gamma=2$ y $\alpha=0.75$, según se especifica en la ecuación 4.6. El valor de α asigna más peso a los ejemplos positivos (la clase minoritaria), mientras que el valor de γ hace que la red se concentre en los ejemplos más difíciles de clasificar, como se ilustra en la figura 4.2. Además, estos valores se seleccionaron teniendo en cuenta los parámetros establecidos por los autores de esta función de pérdida en varios problemas tratados en el artículo en el que se presenta [74]. En el caso de la función de pérdida *BCE Dice*, se aplican los pesos sugeridos en el artículo de Monotrack [33], ya que tratan el mismo problema. Según el artículo, se asigna un peso de 0.1 a la componente de *Dice Loss* y un peso de 0.9 a la componente de BCE.

En la tabla 5.2 se presentan los resultados obtenidos, en términos de las métricas descritas en la sección 5.1, tras los experimentos realizados. Lo que primero destaca es la mejora en comparación con los modelos base que se entrenaron con datos de otros deportes, cuyos resultados se sintetizan en la tabla 5.1. Esta mejora se refleja principalmente en la

precisión de la detección de la posición de la pelota, ya que se reduce considerablemente el número de falsos positivos de tipo 1 (FP1). En relación con la comparación entre las diferentes funciones de pérdida, se observa una mejora entre la BCE y sus variantes: WBCE, *Heat Maps* y *BCE Dice*. Respecto a la función de pérdida utilizada en la red TrackNetV2 (WBCE), también se nota una mejora con las nuevas funciones propuestas: *Heat Maps*, *BCE Dice* y *Focal Loss*. Las funciones que más destacan son *BCE Dice* y *Focal Loss*, que obtienen las mejores métricas.

Prestando atención en los ciclos en los que se obtiene el menor error de validación, se puede detectar que, gracias a la transferencia de aprendizaje, se requieren menos ciclos para ajustar los pesos de la red, tal como se discutió en el capítulo 2.5. Esto demuestra la eficacia de la transferencia de aprendizaje para acelerar el entrenamiento.

TABLA 5.2
EVALUCACIÓN DE LAS FUNCIONES DE PÉRDIDA

Función de pérdida	Ciclo de parada	(TP, TN, FP1, FP2, FN)	Recall	Specificity	Precision	Accuracy	F1
BCE	23	[1203 686 645 124 343]	85.23%	50.62%	67.71%	69.61%	37.73%
WBCE	21	[2056 477 207 12 227]	90.05%	68.53%	90.37%	85.02%	45.10%
Heat Maps	18	[2114 474 215 15 161]	92.92%	67.32%	90.18%	86.87%	45.76%
BCE Dice	20	[2129 476 177 13 184]	92.04%	71.47%	91.80%	87.44%	45.96%
Focal Loss	20	[2144 474 173 15 173]	92.53%	71.60%	91.93%	87.88%	46.11%

Dada la similitud en los resultados producidos por las funciones *Heat Maps*, *BCE Dice* y *Focal Loss*, se seguirán realizando experimentos con éstas, ya que es posible que, variando ciertas condiciones, la arquitectura de la red o los parámetros, alguna pueda mostrar un rendimiento superior a las demás. Sin embargo, se limitará el alcance de las pruebas a *BCE Dice* y *Focal Loss* con el objetivo de reducir el volumen de experimentos, optimizando así el uso de los recursos y el tiempo.

5.3 Estudio de *dropout* y *skip-connections*

Para explorar posibles mejoras en la red a través de la implementación del *dropout* y las *skip-connections*, se examinaron por separado. En el caso del *dropout*, se utilizó un valor de 0.25, lo que significa que se desactivaría el 25% de las conexiones en las capas a las que se aplicó el *dropout*. Este valor se utiliza con frecuencia y puede ser una buena opción para un primer intento de evaluar la efectividad del *dropout*. Se considera un valor no demasiado agresivo pero lo suficientemente sustancial para que se perciba su impacto. En cuanto a las *skip-connections*, no requieren de ningún hiperparámetro que ajustar.

En la tabla 5.3 se presentan los resultados de aplicar *dropout* sobre la red TrackNetV2. Como se observa, las métricas resultantes empeoran ligeramente respecto al modelo sin dropout (-1.12% de media). Al comparar las dos funciones de pérdida, *Focal Loss*

mantiene una ligera ventaja respecto a *BCE Dice*. A pesar de no provocar un empeoramiento sustancial en la eficacia de la red, no se puede considerar el *dropout* como una mejora para el problema abordado.

TABLA 5.3
EVALUCACIÓN DE LA INCLUSIÓN DE *DROPOUT* EN LA RED

Función de pérdida	(TP, TN, FP1, FP2, FN)	Recall	Specificity	Precision	Accuracy	F1
Focal Loss	[2094 475 222 14 174]	92.32%	66.80%	89.87%	86.23%	45.54%
BCE Dice	[2022 484 253 5 215]	90.38%	65.22%	88.68%	84.12%	44.76%

En la tabla 5.4 se presentan las métricas obtenidas sobre el conjunto de prueba tras implementar las *skip-connections* en la arquitectura de la red TrackNetV2. Como es evidente, se observa un empeoramiento de los resultados, principalmente al utilizar la función *Focal Loss*, lo que sugiere que esta modificación puede ser descartada. Es posible que en la red MonoTrack, las *skip-connections* funcionaran mejor debido al resto de cambios que implementaron respecto a la red TrackNetV2, como la reducción de filtros y conexiones.

TABLA 5.4
EVALUCACIÓN DE LA INCLUSIÓN DE *SKIP-CONNECTIONS* EN LA RED

Función de pérdida	(TP, TN, FP1, FP2, FN)	Recall	Specificity	Precision	Accuracy	F1
Focal Loss	[1454 484 110 5 926]	61.09%	80.80%	92.67%	65.05%	36.81%
BCE Dice	[1914 484 162 5 414]	82.21%	74.34%	91.97%	80.49%	43.41%

5.4 Comportamiento del modelo final con pelotas ocultas

Debido a que las modificaciones propuestas sobre la arquitectura de la red no aportan mejoras, se analizará el desempeño del modelo en la predicción de la posición de pelotas ocultas. Para este experimento, se empleará la arquitectura original y las funciones de pérdida *Focal Loss* y *BCE Dice*. Se utilizará el conjunto de datos "occluded", lo que facilitará el entrenamiento del modelo con datos que reflejen estas condiciones específicas.

La tabla 5.5 presenta los resultados obtenidos al entrenar y evaluar los dos modelos utilizando el conjunto de datos "Occluded". Por otro lado, la tabla 5.6 muestra los resultados de la evaluación de los modelos entrenados con el conjunto "Default" y evaluados sobre los datos del conjunto "Occluded". Este enfoque se ha llevado a cabo para valorar si existe una mejora o deterioro en el rendimiento del modelo dependiendo de los datos con los que se ha entrenado.

Los resultados evidencian que el entrenamiento del modelo con el conjunto de datos "Occluded" podría deteriorar el desempeño de la red en ciertos casos que se predijeron correctamente con el modelo entrenado con el conjunto "Default". Tras analizar los

resultados de las predicciones, es relevante señalar que algunos de los casos del conjunto de prueba en los que la pelota está oculta son detectados por los modelos entrenados con el conjunto "Default". Concretamente, de las 35 pelotas ocultas presentes en el conjunto de prueba, tres son detectadas por estos modelos.

Este comportamiento puede atribuirse al hecho de que en los vídeos de partidos de pádel existen oclusiones que se extienden a lo largo del tiempo, las cuales, al durar más de tres fotogramas, dificultan la detección de la red debido a la falta de una referencia clara de la ubicación de la pelota. Estas oclusiones prolongadas pueden ser las responsables del deterioro del entrenamiento de la red, ya que se presentan varios fotogramas en los que no se distingue ninguna pelota, mientras que la etiqueta indica lo contrario. En resumen, se evidencia que es más beneficioso entrenar el modelo con el conjunto de datos "Default", y como resultado de este entrenamiento, la red será capaz de identificar algunas situaciones en las que la pelota esté oculta o su detección sea complicada.

TABLA 5.5
EVALUACIÓN DEL MODELO FINAL ENTRENADO CON EL CONJUNTO DE DATOS "OCCLUDED" SOBRE EL CONJUNTO "OCCLUDED"

Función de pérdida	(TP, TN, FP1, FP2, FN)	Recall	Specificity	Precision	Accuracy	F1
Focal Loss	[2003 433 170 2 371]	84.37%	71.57%	92.09%	81.77%	44.03%
BCE Dice	[2003 433 170 2 371]	84.37%	71.57%	92.09%	81.77%	44.03%

TABLA 5.6
EVALUACIÓN DEL MODELO FINAL ENTRENADO CON EL CONJUNTO DE DATOS "DEFAULT" SOBRE EL CONJUNTO "OCCLUDED"

Función de pérdida	(TP, TN, FP1, FP2, FN)	Recall	Specificity	Precision	Accuracy	F1
Focal Loss	[2147 429 179 6 218]	90.78%	69.86%	92.06%	86.47%	45.71%
BCE Dice	[2132 428 180 7 232]	90.18%	69.59%	91.93%	85.93%	45.52%

5.5 Eficiencia del modelo

Uno de los principales objetivos del presente trabajo era conseguir utilizar el sistema en tiempo real utilizando una GPU de gama media. Para evaluar el cumplimiento de este objetivo, se medirá el tiempo que la red toma para generar predicciones sobre un video. Para ello, se utilizará la métrica fotogramas por segundo (FPS), que indica la cantidad de imágenes que el sistema puede procesar en un segundo. Para que se considere que el sistema es capaz de procesar videos en tiempo real, debería poder manejar alrededor de 25-30 FPS, ya que este es el estándar de FPS que un vídeo muestra por segundo.

Para llevar a cabo estos experimentos, se emplearon dos GPUs: una de gama baja (NVIDIA GTX 1660 Ti) y otra de gama media-baja (NVIDIA GeForce RTX 3060 V2). Los modelos que se emplearán serán los entrenados con *Focal Loss*, con y sin *dropout*.

No es necesario probar otras funciones de pérdida, ya que estas solo afectan en el entrenamiento de la red y no en su uso. Sobre esta base, se midieron los tiempos requeridos para hacer predicciones y para procesar un video completo, donde el procesamiento incluye la tarea de marcar la ubicación de la pelota en cada fotograma del video con un punto rojo. Para realizar estas operaciones, se utilizó el *script predict3.py*, descrito en la sección 4.2.

En la tabla 5.7 se muestran las métricas obtenidas en estas pruebas. Analizando estos resultados, se puede llegar a la conclusión de que la inclusión de *dropout* en la red no aporta mejoras de velocidad. También se puede afirmar que la red es capaz de procesar videos en tiempo real ya que utilizando la GPU NVIDIA GTX 1660 Ti, que está un poco por debajo de la gama media, casi se logra llegar a los 25 FPS.

TABLA 5.7
VELOCIDAD DE PREDICCIÓN DE LA RED

Tarjeta gráfica	Modelo	Predicción	Predicción y procesado
NVIDIA GTX 1660 Ti (gama baja)	*Focal Loss*	15.19 FPS	13.11 FPS
	Focal Loss y *dropout*	15.31 FPS	13.87 FPS
NVIDIA GeForce RTX 3060 V2 (gama media-baja)	*Focal Loss*	24.7 FPS	22.42 FPS
	Focal Loss y *dropout*	24.52 FPS	22.58 FPS

6. PLANIFICACIÓN Y PRESUPUESTO

El presente trabajo se ha desarrollado en varias fases. En esta sección se describirá la planificación ejecutada para alcanzar los objetivos propuestos. Seguidamente, se detallan las etapas y el tiempo estimado dedicado a cada una de ellas, proporcionando además un presupuesto aproximado.

6.1 Planificación

Para la planificación del trabajo, se ha comenzado definiendo las tareas necesarias para cumplir los objetivos propuestos. A continuación, se especificará cada una:

1. Planteamiento y objetivos: esta etapa inicial implica la definición clara del problema a tratar y la formulación de los objetivos específicos del proyecto.

2. Investigación del estado del arte: en esta etapa, se realiza un estudio exhaustivo de los trabajos existentes en el campo de la segmentación de video y la detección de pequeños objetos en movimiento, particularmente enfocados en deportes como el tenis, bádminton, tenis de mesa, entre otros. Esta fase permite entender las técnicas y metodologías actuales, identificar sus fortalezas y debilidades y elegir la metodología a utilizar.

3. Estudio de redes convolucionales: este paso se centra en la comprensión profunda de las CNN, centrándose en los modelos que se utilizarán como base para la solución final. Esta etapa resulta crucial para el posterior desarrollo del modelo y su capacidad para detectar la pelota en los vídeos de los partidos de pádel de forma eficiente.

4. Generación del conjunto de datos: en esta etapa, se recopilan vídeos de partidos de pádel y se transforman en fotogramas, que se etiquetan manualmente. Este conjunto de datos se utilizará para entrenar y validar el futuro modelo.

5. Modificaciones implementadas sobre el modelo: esta fase consiste en la implementación de diversas modificaciones y ajustes sobre la red que se toma como base. Estas modificaciones incluyen: ajuste de parámetros, cambios en la arquitectura, inclusión de un conjunto de validación para decidir el número de épocas óptima y la implementación de diferentes funciones de pérdida para realizar el entrenamiento de las redes.

6. Experimentos: una vez implementada la red, se llevan a cabo una serie de experimentos para valorar las modificaciones implementadas, así como evaluar el rendimiento y eficacia en la detección de la pelota en los vídeos de partidos de pádel. Los resultados de estos experimentos se analizan y discuten para obtener conclusiones.

7. Elaboración de la Memoria: en esta última etapa, se redacta la memoria del proyecto, documentando todos los pasos del proyecto, desde el planteamiento inicial hasta los resultados finales. Esta memoria incluye una descripción detallada de la metodología

utilizada, los resultados obtenidos, la discusión de estos resultados y las conclusiones. Además, comprende un análisis de los aspectos legales y socioeconómicos, así como de la planificación y el presupuesto del proyecto.

Para la planificación de las tareas, se ha utilizado el diagrama de Gantt [82], una herramienta de gestión de proyectos que visualiza las tareas, su duración estimada y las dependencias entre ellas en forma de gráfico de barras. Esto ha facilitado la planificación y el seguimiento del progreso del proyecto. En este proyecto, debido a la falta de un horario fijo de trabajo, se ha utilizado el diagrama de la Figura 6.1 como una herramienta para establecer periodos de tiempo aproximados y determinar el orden de ejecución de cada tarea considerando las dependencias entre ellas, representadas con flechas. Estas dependencias indican que una tarea sucesora no puede comenzar hasta que la tarea predecesora esté completada. La tarea de elaboración de la memoria se ha separado del resto debido a que depende de todas las demás tareas, pero puede desarrollarse en paralelo a ellas.

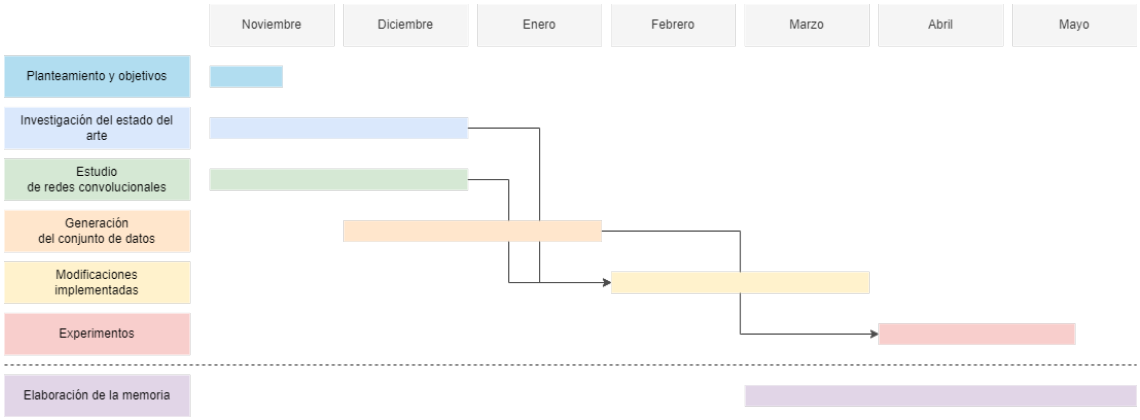


Figura 6.1 Diagrama de Gantt para la planificación del trabajo.

En la tabla 6.1 se define el tiempo dedicado a cada tarea en horas. Para calcular el tiempo dedicado a cada una, se ha utilizado la versión gratuita de la herramienta toggl [83], un software que permite a los usuarios registrar y contabilizar el tiempo dedicado a diferentes tareas y proyectos. En el caso de la tarea que abarca la ejecución de los experimentos, se añade una estimación del tiempo total requerido para ejecutar el conjunto de experimentos. Esta estimación se basa en un promedio de 72 horas por experimento y un total de 14 experimentos llevados a cabo.

TABLA 6.1
TAREAS Y EL TIEMPO DEDICADO A CADA UNA

Tarea	**Duración (horas)**
Planteamiento y objetivos	15 horas
Investigación del estado del arte	40 horas
Estudio de redes convolucionales	30 horas
Generación del conjunto de datos	85 horas
Modificaciones implementadas sobre el modelo	55 horas
Experimentos	25 horas (+ 1008 horas)
Elaboración de la memoria	70 horas
Total	**320 horas**

6.2 Presupuesto

El presente capítulo busca proporcionar una evaluación detallada y minuciosa del costo asociado con la realización del presente trabajo. Los costos se desglosan principalmente en tres categorías: coste humano, coste de hardware y coste de software.

Primero, respecto al coste humano, este proyecto fue realizado por un único individuo que asumió las funciones de un Analista de Datos Junior, el cual suele tener un salario medio de 12€ por hora en España [84]. Por lo tanto, teniendo en cuenta el tiempo total invertido en todas las tareas, que asciende a 320 horas, se estima un coste humano de 3.840€.

En cuanto al coste de hardware, se utilizaron dos equipos para diferentes tareas. Para el entrenamiento de la CNN, se utilizó un ordenador equipado con un procesador Intel i7-12700k y una GPU NVIDIA GEFORCE RTX 3060 V2 12GB GDDR6. El coste de este equipo se estima en aproximadamente 1.500€ (www.pccomponentes.com). Considerando su uso durante 1008 horas y una vida útil estimada de 5 años (43.800 horas), el coste de utilización del hardware para el entrenamiento de la CNN asciende a aproximadamente 34,3€.

Por otro lado, el equipo empleado para el resto de tareas contó con un procesador Intel i7-10750H, una GPU NVIDIA GTX 1660ti. El costo de este equipo es de aproximadamente 1.200€ (www.pccomponentes.com). Considerando su uso durante 320 horas y una vida útil estimada de 5 años (43.800 horas), el coste de utilización de este hardware para el resto de tareas asciende a aproximadamente 8,75€.

Además de los costos ya mencionados, es crucial tener en cuenta el costo de energía que consumen ambos equipos. El consumo energético de los ordenadores puede variar dependiendo del modelo y las tareas que se están realizando, pero un estimado promedio para estos dos equipos podría ser de alrededor de 350W y 200W, respectivamente. Teniendo en cuenta un costo promedio de la electricidad de 0,12€ por kWh en España [85], y la cantidad de horas que cada equipo ha estado funcionando, el coste energético

para el ordenador de entrenamiento sería de alrededor de 42,34€, y para el equipo utilizado en el resto de tareas sería de 7,68€.

Por último, el coste del software empleado en este proyecto es nulo, ya que todo el software utilizado es de código abierto y, por lo tanto, gratuito. A excepción de la herramienta toggl, que dispone de un plan de uso gratuito.

TABLA 6.2
DESGLOSE DE COSTES DEL PROYECTO

Categoría	**Descripción**	**Horas de uso**	**Coste**
Recursos humanos	Analista de Datos Junior	320 horas	6.400€
Hardware	Ordenador para entrenamiento	1008 horas	34,3€
	Ordenador para el resto de tareas	320 horas	8,75€
Energía	Ordenador de entrenamiento	1008 horas	42,34€
	Ordenador para el resto de tareas	320 horas	7,68€
Software	Software de código abierto	1328 horas	0€
Total			**6493.07€**

En conclusión, considerando todos los aspectos mencionados anteriormente, el costo total estimado para la realización del presente trabajo asciende a 6.475,92€. En la tabla 6.2 se ofrece un resumen detallado del coste del proyecto.

7. CONCLUSIONES Y FUTUROS TRABAJOS

6.3 Conclusiones

En este trabajo se ha abordado el reto de adaptar y entrenar una Red Neuronal Convolucional para detectar pelotas en vídeos de pádel, una tarea para la que no existía previamente una solución que empleara técnicas de Inteligencia Artificial. Las ventajas de este enfoque incluyen su compatibilidad con vídeos de partidos grabados con una sola cámara y la capacidad de utilizarlo en sistemas de bajas prestaciones. Además, este trabajo puede servir de base para abordar otros retos, como la creación de sistemas más complejos que puedan generar estadísticas automáticamente a partir de vídeos de este deporte.

Para abordar este desafío, el primer paso fue generar un conjunto de datos a partir de vídeos del torneo WPT, los cuales fueron descompuestos en fotogramas y posteriormente etiquetados manualmente. Una vez generado este conjunto de datos, y a partir del código base de la red TrackNetV2 [32], se entrenó la red con los nuevos datos, aprovechando un modelo ya entrenado con vídeos de tenis, aplicando el concepto de transferencia de aprendizaje. Asimismo, se implementaron otras modificaciones sobre la red, como la evaluación de varias funciones de pérdida, la introducción de un procedimiento de validación, cambios en la arquitectura de la red como la incorporación de *dropout* y *skip-connections*, y otras modificaciones sobre el código original.

La aplicación de transferencia de aprendizaje, utilizando el nuevo conjunto de datos, ha demostrado una gran mejora de los resultados sobre el conjunto de pruebas, logrando hasta un 50% de mejora en algunos experimentos en comparación con el modelo entrenado con tenis. Adicionalmente, los mejores modelos de estos experimentos se han obtenido durante los primeros ciclos de aprendizaje, gracias a la inicialización de los pesos a partir de otro modelo.

Respecto a las funciones de pérdida utilizadas, los resultados mostraron mejoras para todas ellas con respecto a la red inicial. Esta observación refuerza la idea de que la transferencia de aprendizaje es esencial para el proyecto. Entre todas las funciones de pérdida probadas, las dos que mejores resultados arrojaron fueron *BCE Dice*, propuesta por los autores de la red MonoTrack [33], y *Focal Loss*, ideada a partir de la investigación del estado del arte.

Con relación a las modificaciones realizadas en la arquitectura de la red, como la implementación de técnicas de *dropout* y *skip-connections*, estas no proporcionaron los resultados esperados en el contexto del pádel. A pesar de ello, su exploración fue útil para entender mejor el comportamiento de la red y determinar qué técnicas funcionan mejor en este tipo de tareas.

El modelo final mostró un buen rendimiento en la tarea de detección de pelotas, incluso en casos difíciles donde las pelotas estaban parcialmente ocultas. Además, es relevante

resaltar que el sistema desarrollado es capaz de operar en tiempo real utilizando una GPU de gama media. Esto evidencia la eficiencia del modelo y su utilidad práctica en situaciones en vivo.

6.4 Futuros Trabajos

Los resultados obtenidos en este trabajo abren un camino hacia el desarrollo de proyectos futuros que podrían ampliar la eficiencia y la aplicabilidad del sistema presentado.

Una de las posibles direcciones de mejora está relacionada con la exploración de otras arquitecturas de red. La arquitectura TrackNetV2 empleada en este proyecto podría mejorarse mediante la implementación de técnicas más avanzadas, como los *Transformers*. Estas arquitecturas, que han demostrado ser altamente eficaces en tareas de procesamiento del lenguaje natural, podrían proporcionar mejoras sustanciales en la detección de objetos y, en particular, en la detección de la pelota en vídeos de pádel.

Otra línea de trabajo futuro se relaciona con la ampliación del sistema de estadísticas. Actualmente, el sistema está enfocado en la detección de la pelota en vídeos de pádel. Este podría ser el inicio para la elaboración de un sistema más sofisticado capaz de rastrear no solo la pelota, sino también la trayectoria en tres dimensiones, sus botes o las acciones de los jugadores. Estas ampliaciones permitirían la generación de estadísticas más completas y detalladas, lo cual sería de gran valor para espectadores, entrenadores y jugadores. Esto podría aplicarse creando sistemas auxiliares que empleen la red para la detección de la pelota, o ampliando las salidas de la red con otros atributos.

Finalmente, otro aspecto para el desarrollo futuro sería el enriquecimiento del conjunto de datos. Este trabajo ha mostrado la eficacia de la transferencia de aprendizaje en la detección de la pelota en vídeos de pádel. No obstante, se podría obtener una mejora en los resultados ampliando y diversificando el conjunto de datos. Esto implicaría la inclusión de vídeos capturados desde diferentes ángulos, en diferentes pistas y con distintas cámaras. Un conjunto de datos más amplio y diverso contribuiría a reducir el *overfitting* y a generar un modelo más robusto, capaz de funcionar eficazmente con vídeos de clubs o partidos amateurs.

8. BIBLIOGRAFÍA

[1] Ariv Sport. “El padel crece por el mundo: ¿Cuántos jugadores y pistas hay?”, Blog post, [En Línea]. Disponible en: https://www.arivsport.com/el-padel-crece-por-el-mundo-cuantos-jugadores-y-pistas-hay/.

[2] Europa Press. “Crecimiento mundial del récord: 181 clubes de pádel en los últimos años”. 19-05-2019. [En Línea] Disponible en: https://www.europapress.es/deportes/noticia-crecimiento-mundial-record-181-clubes-padel-anos-20220519150643.html.

[3] YouTube, "Términos del Servicio" YouTube, Enero 2022. [En línea]. Disponible en: https://www.youtube.com/static?template=terms

[4] Central European University, "Copyright and Fair Use Guidelines for Students" Central European University, Enero 2022. [En línea]. Disponible en: https://documents.ceu.edu/file/531/download?token=EbS6mJgG#:~:text=Fair%20use%20is%20allowed%20only,used%2C%20and%20not%20complete%20works

[5] U.S. Copyright Office, "Copyright Law of the United States and Related Laws Contained in Title 17 of the United States Code", U.S. Copyright Office, Octubre 2021. [En línea]. Disponible en: https://www.copyright.gov/title17/92chap1.html#107

[6] Python Software Foundation, "Python License", Python Software Foundation. [En línea]. Disponible en: https://docs.python.org/3/license.html.

[7] TensorFlow Authors, "TensorFlow License," TensorFlow. [En línea]. Disponible en: https://github.com/tensorflow/tensorflow/blob/master/LICENSE.

[8] NumFOCUS, "Numpy License," NumPy. [En línea]. Disponible en: https://numpy.org/doc/stable/license.html.

[9] NumFOCUS, "Pandas License," pandas. [En línea]. Disponible en: https://pandas.pydata.org/about/license.html.

[10] Matplotlib Developers, "Matplotlib License", Matplotlib. [En línea]. Disponible en: https://matplotlib.org/stable/users/license.html.

[11] NVIDIA Corporation, "NVIDIA Software License Agreement", NVIDIA. [En línea]. Disponible en: https://docs.nvidia.com/cuda/eula/index.html.

[12] European Commission, “New rules for Artificial Intelligence”. 2021. [En línea]. Disponible en: https://ec.europa.eu/commission/presscorner/detail/en/QANDA_21_1683#1

[13] PrivateWall magazine, “Inteligencia artificial: Machine & Deep Learning”, 2018. [En línea]. Disponible en: https://www.privatewallmag.com/inteligencia-artificial-machine-deep-learning/

[14] G A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser y I. Polosukhin, "Attention is all you need," en Advances in Neural Information Processing Systems, 2017, pp. 5998-6008. Disponible en: https://arxiv.org/abs/1706.03762

[15] Baldi, P. "Autoencoders, Unsupervised Learning, and Deep Architectures" Proceedings of ICML Workshop on Unsupervised and Transfer Learning, en Proceedings of Machine Learning Research. (2012). Disponible en: https://proceedings.mlr.press/v27/baldi12a.html

[16] S. Hochreiter y J. Schmidhuber, "Long short-term memory,"Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997. Disponible en: https://www.researchgate.net/publication/13853244_Long_Short-term_Memory

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville y Y. Bengio, "Generative adversarial nets", en Advances in Neural Information Processing Systems, 2014, pp. 2672-2680. Disponible en: http://papers.neurips.cc/paper/5423-generative-adversarial-nets.pdf

[18] Y. LeCun, L. Bottou, Y. Bengio, y P. Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 11-11-1998. [En línea]. Disponible en: https://ieeexplore.ieee.org/document/726791.

[19] A. Krizhevsky, I. Sutskever y G. Hinton, "ImageNet classification with deep convolutional neural networks", Communications of the ACM. 24-05-2017. [En línea]. Disponible en: https://proceedings.neurips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[20] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation", arXiv, 14-11-2014. [En línea]. Disponible en: https://arxiv.org/abs/1411.4038.

[21] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 2014, pp. 580-587. Disponible en: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6909475

[22] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN", 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 2980-2988, doi: 10.1109/ICCV.2017.322. Dispobible en: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8237584

[23] S. Ioffe y C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", arXiv preprint arXiv:1502.03167, Feb. 2015. [En línea]. Disponible en: https://arxiv.org/abs/1502.03167.

[24] KIEFER, “Cornelia. Assessing the Quality of Unstructured Data: An Initial Overview”, en LWDA. 2016. p. 62-73. Disponible en: https://ceur-ws.org/Vol-1670/paper-25.pdf

[25] C. M. Bishop, “Pattern Recognition and Machine Learning”. New York: Springer, 2006.

[26] M. Albon, "Ball Tracking in Volleyball with OpenCV and TensorFlow," Towards Data Science, 15 de septiembre de 2018. [En línea]. Disponible en: https://towardsdatascience.com/ball-tracking-in-volleyball-with-opencv-and-tensorflow-3d6e857bd2e7

[27] A. Vila Abad, "Ball detection in sports videos", Trabajo final de máster, Facultat de Matemàtiques i Informàtica, Universitat de Barcelona, Barcelona, 2017. [En línea]. Disponible en: https://diposit.ub.edu/dspace/handle/2445/119897

[28] H. Myint, P. Wong, L. Dooley and A. Hopgood, "Tracking a table tennis ball for umpiring purposes", 2015 14th IAPR International Conference on Machine Vision Applications (MVA), Tokyo, Japan, 2015, pp. 170-173, doi: 10.1109/MVA.2015.7153160. [En línea] Disponible en: https://ieeexplore.ieee.org/document/7153160

[29] M, Archana y Geetha, M. (2015). “Object Detection and Tracking Based on Trajectory in Broadcast Tennis Video”, Procedia Computer Science. 58. 225-232. 10.1016/j.procs.2015.08.060. [En línea] Disponible en: https://www.researchgate.net/publication/283184656_Object_Detection_and_Tracking_Based_on_Trajectory_in_Broadcast_Tennis_Video

[30] W. Li, X. Liu, K. An, C. Qin, and Y. Cheng, “Table Tennis Track Detection Based on Temporal Feature Multiplexing Network” Sensors, vol. 23, no. 3, p. 1726, Feb. 2023, doi: 10.3390/s23031726. [En línea]. Disponible en: http://dx.doi.org/10.3390/s23031726

[31] Y.-C. Huang, I.-N. Liao, C.-H. Chen, T.-U. İk, and W.-C. Peng, “TrackNet: A Deep Learning Network for Tracking High-speed and Tiny Objects in Sports Applications”, arXiv, 2019. Disponible en: https://arxiv.org/pdf/1907.03698

[32] N.-E. Sun, Y.-C. Lin, S.-P. Chuang, T.-H. Hsu, D.-R. Yu, H.-Y. Chung, and T.-U. İk, "TrackNetV2: Efficient Shuttlecock Tracking Network", en 2020 International Conference on Pervasive Artificial Intelligence (ICPAI), 2020, pp. 86-91, doi: 10.1109/ICPAI51961.2020.00023. [En línea]. Disponible en: https://ieeexplore.ieee.org/document/9302757

[33] P. Liu and J.-H. Wang, "MonoTrack: Shuttle trajectory reconstruction from monocular badminton video", arXiv preprint arXiv:2204.01899, 2022. [En línea]. Disponible en: https://doi.org/10.48550/arXiv.2204.01899

[34] M. Mostajabi, "Review: DeconvNet & Unpooling Layer Semantic Segmentation", Towards Data Science, 17-02-2020. [En línea]. Disponible en: https://towardsdatascience.com/review-deconvnet-unpooling-layer-semantic-segmentation-55cf8a6e380e

[35] V. Belagiannis and A. Zisserman, "Recurrent Human Pose Estimation", 2017. [En línea]. Disponible en: https://arxiv.org/pdf/1605.02914.pdf

[36] Pfister, J. Charles, and A. Zisserman, "Flowing ConvNets for Human Pose Estimation in Videos", CoRR, vol. abs/1506.02897, 2015. [En línea]. Disponible en: https://arxiv.org/abs/1506.02897

[37] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", arXiv preprint arXiv:1409.1556, 2014. [En línea]. Disponible en: https://arxiv.org/abs/1409.1556

[38] H. Noh, S. Hong, and B. Han, "Learning Deconvolution Network for Semantic Segmentation", in Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1520-1528, doi: 10.1109/ICCV.2015.178. [En línea]. Disponible en: https://arxiv.org/abs/1505.04366

[39] Wikipedia contributors, "Circle Hough Transform", Wikipedia, The Free Encyclopedia, [En línea]. Disponible en: https://en.wikipedia.org/wiki/Circle_Hough_Transform.

[40] V. Rathod, "Understanding Focal Loss for Pixel-Level Classification in Convolutional Neural Networks", Medium, 21 de noviembre de 2019. [En línea]. Disponible en: https://medium.com/swlh/understanding-focal-loss-for-pixel-level-classification-in-convolutional-neural-networks-720f19f431b1.

[41] Chi-Feng Wang, “The Vanishing Gradient Problem”, *Medium*, 23-04-2019. [En línea]. Disponible en: https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484

[42] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., & Murphy, K. “Speed/accuracy trade-offs for modern convolutional object detectors”. ArXiv. /abs/1611.10012. 2016. [En línea]. Disponible en: https://arxiv.org/abs/1611.10012

[43] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90. 2016. [En línea]. Disponible en: https://ieeexplore.ieee.org/document/7780459

[44] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),

2015, pp. 234-241. [En línea]. Disponible en: https://link.springer.com/chapter/10.1007/978-3-319-24574-4_28

[45] Keras Team, "Adadelta," Keras, 2021. [En línea]. Disponible en: https://keras.io/api/optimizers/adadelta/

[46] C. Cortes, L. D. Jackel, W. P. Chiang, 'Limits on Learning Machine Accuracy Imposed by Data Quality', en *Advances in Neural Information Processing Systems*, 1994, vol. 7. Disponible en: https://proceedings.neurips.cc/paper/1994/file/1e056d2b0ebd5c878c550da6ac5d3724-Paper.pdf

[47] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the Knowledge in a Neural Network. arXiv preprint arXiv:1503.02531. [En línea]. Disponible en: https://arxiv.org/abs/1503.02531

[48] Hussain, M., Bird, J.J., Faria, D.R. (2019). A Study on CNN Transfer Learning for Image Classification. En: Lotfi, A., Bouchachia, H., Gegov, A., Langensiepen, C., McGinnity, M. (eds) Advances in Computational Intelligence Systems. UKCI 2018. Advances in Intelligent Systems and Computing, vol 840. Springer, Cham. Disponible en: https://doi.org/10.1007/978-3-319-97982-3_16

[49] Mwiti, D. Transfer learning guide: A practical tutorial with examples for images and text in Keras, neptune.ai. 2023. Disponible en: https://neptune.ai/blog/transfer-learning-guide-examples-for-images-and-text-in-keras

[50] Alvin, T.P. Transfer learning and Convolutional Neural Networks (CNN), Medium. MLearning.ai. 2022. Disponible en: https://medium.com/mlearning-ai/transfer-learning-and-convolutional-neural-networks-cnn-e68db4c48cca

[51] Taleb, M. A. Serhani, and R. Dssouli, "Big Data Quality Assessment Model for Unstructured Data," presentada en 2018 International Conference on Innovations in Information Technology (IIT), Al Ain, United Arab Emirates, pp. 69–74, doi: 10.1109/INNOVATIONS.2018.8605945. Disponible en: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8605945

[52] T. Mitsa, "How do you know you have enough training data?", *Medium*, 23-04-2019. [En línea]. Disponible en: https://towardsdatascience.com/how-do-you-know-you-have-enough-training-data-ad9b1fd679ee.

[53] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era". arXiv, 2017. Disponible en: https://arxiv.org/pdf/1707.02968

[54] Joulin, A., "Learning Visual Features from Large Weakly Supervised Data", arXiv, 2015. Disponible en: https://arxiv.org/abs/1511.02251

[55] Y. Xia, “How dataset size or Ram Chokes Your Deep Learning for Computer Vision”, *Medium*. Disponible en: https://becominghuman.ai/how-dataset-size-or-ram-chokes-your-deep-learning-for-computer-vision-eb788d670ec1.

[56] “World Padel Tour”. Wikipedia, La enciclopedia libre. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=World_Padel_Tour&oldid=149112672

[57] Federación Internacional de Pádel. Reglamento de juego F.I.P. 2017. Disponible en: https://www.worldpadeltour.com/media-content/2017/03/REGLAMENTO-DE-JUEGO-FIP-2017.pdf

[58] “La suerte Está Echada para el Estrella damm alicante open 2021: World padel tour,” *worldpadeltour.com*, 15-04-2021. Disponible en: https://www.worldpadeltour.com/noticias/competicion/la-suerte-esta-echada-para-el-estrella-damm-alicante-open-2021/.

[59] “World padel tour,” *YouTube*. [En línea]. Disponible en: https://www.youtube.com/channel/UCK59dYVs3Wgwoe73nDTH6jw.

[60] World Padel Tour. *Quarter-Finals Highlights Bela/Coello Vs Paquito/Di Nenno Amsterdam Padel Open 2022*. 01-10-2022. [Video en línea]. Disponible en: https://www.youtube.com/watch?v=BqnUYkYvkrA

[61] Bradski G. The OpenCV Library. Dr. Dobb's Journal of Software Tools. 2008. Disponible en: https://pypi.org/project/opencv-python/

[62] National Chiao Tung University. TrackNet. 2020. [Código fuente]. Disponible en: https://nol.cs.nctu.edu.tw:234/open-source/TrackNet/tree/master

[63] Yu-ching. TrackNetV2. 2021. [Código fuente]. Disponible en: https://nol.cs.nctu.edu.tw:234/open-source/TrackNetv2

[64] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, "TensorFlow: Large-scale machine learning on heterogeneous systems", 2015. [En línea]. Disponible en: https://www.tensorflow.org

[65] TensorFlow, "Why TensorFlow", 2021. [En línea]. Disponible en: https://www.tensorflow.org/about

[66] TensorFlow, "Use TPUs", 2021. [En línea]. Disponible en: https://www.tensorflow.org/guide/tpu

[67] TensorFlow, "Hardware requirements", [En línea]. Disponible en: https://www.tensorflow.org/install/gpu#hardware_requirements.

[68] NVIDIA, "Compare Graphics Cards", [En línea]. Disponible en: https://www.nvidia.com/es-es/geforce/graphics-cards/compare/?section=compare-specs.

[69] NVIDIA, "Tensor Cores", [En línea]. Disponible en: https://www.nvidia.com/en-us/data-center/tensorcore/.

[70] TensorFlow, "Install TensorFlow", [En línea]. Disponible en: https://www.tensorflow.org/install.

[71] TensorFlow, "Software requirements", [En línea]. Disponible en: https://www.tensorflow.org/install/source?hl=es-419#gpu.

[72] karan842, "CNN vs transfer learning", Kaggle. 2022. Disponible en: https://www.kaggle.com/code/karan842/cnn-vs-transfer-learning

[73] J. Doe, "Lebrón/Galán VS Chingotto/Tello: El partido de los puntazos en Reus", YouTube, [En línea]. Disponible en: https://www.youtube.com/watch?v=QiLuapREUnQ

[74] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal Loss for Dense Object Detection", arXiv preprint arXiv:1708.02002, 2018. Disponible en: https://arxiv.org/pdf/1708.02002

[75] A. Bulat and G. Tzimiropoulos, "Human Pose Estimation via Convolutional Part Heatmap Regression", en Computer Vision – ECCV 2016, Springer International Publishing, 2016, pp. 717-732. doi: 10.1007/978-3-319-46478-7_44. Disponible en: https://doi.org/10.1007%2F978-3-319-46478-7_44

[76] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever y R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," Journal of Machine Learning Research, vol. 15, no. 56, pp. 1929-1958, 2014. [En línea]. Disponible en: http://jmlr.org/papers/v15/srivastava14a.html.

[77] H. Wu y X. Gu, "Towards dropout training for convolutional neural networks," Neural Networks, vol. 71, pp. 1-10, Nov. 2015. [En línea]. Disponible en: https://doi.org/10.1016/j.neunet.2015.07.007

[78] Johnson, J.M., Khoshgoftaar, T.M. "Survey on deep learning with class imbalance", Journal of Big Data, 2019. https://doi.org/10.1186/s40537-019-0192-5

[79] Sharma, A. "Exploit your Hyperparameters: Batch Size and Learning Rate as Regularization", Towards Data Science. Noviembre 2019. [En línea]. Disponible en: https://towardsdatascience.com/exploit-your-hyperparameters-batch-size-and-learning-rate-as-regularization-9094c1c99b55

[80] "Precisión, Recall, F1, Accuracy en Clasificación", iArtificial, [En línea]. Disponible en: https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/.

[81] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method", arXiv preprint arXiv:1212.5701, 2012. [En línea]. Disponible en: https://arxiv.org/abs/1212.5701

[82] "Diagrama de Gantt", Wikipedia. Abril 2023. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Diagrama_de_Gantt

[83] Toggl, “Toggl Track”. [En línea]. Disponible en: https://toggl.com/track/

[84] Glassdoor, "Salario de Analista de Datos Junior en España". [En línea]. Disponible en: https://www.glassdoor.es/Sueldos/junior-data-analyst-sueldo-SRCH_KO0,19.htm#:~:text=En%20Espa%C3%B1a%2C%20el%20sueldo%20medio,trabajan%20de%20Junior%20Data%20Analyst.

[85] Papernest, "¿Cuál es el precio de la luz en 2023?", [En línea]. Disponible en: https://www.papernest.es/energia/precio-de-la-luz/