

UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE CIENCIAS

ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN

*Video análisis aplicado al seguimiento de jugadores
de fútbol*

SEMINARIO DE TESIS I

Autor: Alexander Leonardo Lique Lamas

Asesor: PhD. Marco Antonio Alanía Vicente

Enero, 2023

Resumen

El fútbol es uno de los deportes más populares del mundo, con millones de seguidores en todos los continentes. En los últimos años, con el avance de las tecnologías de grabación de video, ha surgido un gran interés en el análisis de video del fútbol.

El análisis de video del fútbol se ha convertido en un gran negocio, ya que permite a los clubes y entrenadores obtener información valiosa sobre el rendimiento de los jugadores, el estado físico de los mismos, detectar patrones de juego y otros aspectos importantes del juego. Además, con el uso de la inteligencia artificial y el computer vision, se ha podido desarrollar tareas como el seguimiento de jugadores, donde se busca analizar la trayectoria de los mismos en el campo, con el objetivo de extraer información valiosa para el análisis.

En el presente trabajo, se enfocó en la tarea de detección de jugadores de fútbol utilizando algoritmos de aprendizaje profundo, específicamente DETR y Faster R-CNN, sobre el dataset de la competición SoccerNet. Se evaluaron los resultados utilizando las métricas establecidas por COCO (Common Objects in Context) utilizadas en el paper original de DETR, con el objetivo de comparar y analizar el rendimiento de ambos algoritmos en esta tarea específica. Los resultados obtenidos mostraron que, en general, Faster R-CNN tuvo un desempeño superior a DETR en la mayoría de las métricas evaluadas.

Índice general

Resumen	III
1. Introducción	2
1.1. Motivación	3
1.2. Objetivos	3
1.3. Estructura del Seminario	3
2. Estado del Arte	5
2.1. Video Análisis en el Fútbol	5
2.2. Machine Learning en el Fútbol	6
2.3. Detección de jugadores	6
3. Marco Teórico	8
3.1. Redes Convolucionales	8
3.1.1. Convolución	8
3.1.2. Pooling	9
3.1.3. ResNet	10
3.1.4. Faster R-CNN	11
3.1.5. Feature Pyramid Networks	11
3.2. Redes Transformers	12
3.2.1. Mecanismos de Atención	13
3.2.2. Codificación posicional	13
3.2.3. Atención de producto punto escalado	14
3.2.4. Multi-Head Attention	15
3.2.5. Codificador-Decodificador	15
3.2.6. Detection Transformers	16

3.3. Transfer Learning	19
4. Metodología	20
4.1. Conjunto de datos	20
4.2. Preparación del conjunto de datos	21
4.3. Métricas	22
4.3.1. Intersección sobre la Unión (IoU)	22
4.3.2. Mean Average Prescision	23
4.3.3. Métrica de COCO	24
4.4. Entrenamiento	25
5. Resultados y Discusiones	26
5.1. Experimentación	26
5.2. Resultados de los modelos	27
5.3. Discusiones	27
6. Conclusiones y Trabajo Futuro	30
6.1. Conclusiones	30
6.2. Trabajo Futuro	30
A. Resultados de la inferencia de los modelos	38
A.0.1. Resultados sobre las primeras épocas de entrenamiento	38
A.0.2. Resultados con los mejores pesos del entrenamiento	40
A.0.3. Secuencia de frames de la inferencia de Faster R-CNN	42
A.0.4. Secuencia de frames de la inferencia de DETR	43
A.0.5. Detecciones erróneas de Faster R-CNN	44
A.0.6. Detecciones erróneas de DETR	45

Índice de figuras

3.1.	Operación de convolución [22]	9
3.2.	Filtro outline	9
3.3.	Operación de <i>Max Pooling</i> [23]	10
3.4.	ResNet18 [26]	10
3.5.	Faster R-CNN [27]	11
3.6.	Arquitectura FPN [28]	12
3.7.	Representación de la arquitectura Faster R-CNN con una FPN [29]	12
3.8.	(izquierda) Atención de producto punto escalado. (derecha)Multi-head attention consta de varios capas de atención que se ejecutan en paralelo.[33]	15
3.9.	Arquitectura del modelo Transformers [33]	16
3.10.	Detection Transformer [36]	17
3.11.	Arquitectura del <i>transformer</i> DETR [36]	18
18figure.caption.47		
3.13.	Representación general del método de aprendizaje de transferencia, La red A está pre entrenada en un conjunto de datos D_A y para una tarea T_A . Se toma una parte de ella para especializarla sobre una tarea T_B con un dataset D_B añadiendole algunas capas que seran entrenadas [37].	19
4.1.	Pipeline del trabajo	20
4.2.	Conjunto de datos de SoccerNet	21
4.3.	Nuevo formato creado para leer el conjunto de datos como se puede ver, se tiene tanto la dirección donde se ubica la imagen y su Ground thrue de coordenadas a detectar.	22
4.4.	Intersección sobre la unión	22

4.5. Ejemplo de IoU	23
4.6. Análisis del IoU bajo un umbral $u = 0.5$	24
4.7. Métricas de COCO	25
4.8. Google colaboratory	25
A.1. Inferencia de Faster R-CNN y DETR sobre un frame de SoccerNet test.	38
39figure.caption.86	
A.3. Inferencia de Faster R-CNN	40
41figure.caption.96	
A.5. frames	42
A.6. frames	43
A.7. frames	44
A.8. frames	45

Índice de Acrónimos

BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network
COCO	Common Objects in Context
DETR	Detection Transformer
Faster R-CNN	Faster Region based Convolutional
GPT	Generative Pretrained Transformers
IoU	Intersection over Union
mAP	Mean Average Precision
ResNet	Residual Neuronal Network.
RNN	Recurrent Neural Network

Agradecimientos

Agradezco a mi familia por inculcarme los valores y apoyarme siempre en mis estudios.

A mis compañeros de ACECOM, gracias por animarme a seguir aprendiendo sobre este extraordinario campo que es la inteligencia artificial.

A mi asesor el Dr. Marcos Antonio Alanía Vicente por brindarme su tiempo y apoyo a lo largo de la realización de esta tesis.

Capítulo 1

Introducción

El fútbol es uno de los deportes más seguidos a nivel mundial, considerado el “deporte rey”, ya que alberga componentes políticos, sociales y económicos que en consecuencia ha generado que sea uno de los espectáculos más vistos.

Con el rápido desarrollo de las tecnologías para la retransmisión y la grabación de los partidos de fútbol. El analizar dichos videos es de crucial importancia tanto para el sector periodístico que desea informar acerca de algunas estadísticas así como también para los entrenadores; evaluar el rendimiento, generar estrategias o preparar las futuras sesiones de entrenamiento [1]. Para generar dicha información se realiza un seguimiento de los jugadores, por ello se requiere que estos últimos usen sensores en su vestimenta que capturen su posición, además de usar múltiples de cámaras de alta resolución instaladas en el campo de fútbol. A menudo estas soluciones requieren de millones de dólares que, en consecuencia, genera que solo los equipos profesionales en las altas competición puedan adquirirlos [2]. Por otra parte, los avances en los algoritmos de inteligencia artificial sobre el *Computer vision* ha abierto nuevas opciones para poder analizar la posición del jugador a través de los videos. Analizando tareas generales como lo es la detección de objetos y los algoritmos de *tracking* [3].

En el presente trabajo se llevará a cabo un estudio comparativo entre dos modelos pre-entrenados en la tarea de detección de objetos: Detection Transformer (DETR) y Faster R-CNN. Estos modelos serán aplicados en el dominio específico de la detección de jugadores de fútbol en frames de videos de fútbol.

1.1. Motivación

La Inteligencia Artificial ha tenido una gran revolución en estos últimos años y gracias a la masiva cantidad de datos y la mejora en la capacidad de hardware, que poseemos hoy en día. Este campo ha tenido una gran acogida debido a la diversidad de aportaciones en diferentes campos y el fútbol no es exento de ello. Como estudiante de la carrera de ciencia en computación tengo gran interés en relacionar el campo del *Deep learning* mediante los algoritmos basados en las redes neuronales para detectar a los jugadores en el campo de fútbol y posteriormente el seguimiento de estos.

1.2. Objetivos

El objetivo de este seminario es presentar la arquitectura de *vision transformer* DETR y aplicarlo en la detección de jugadores de fútbol sobre videos de fútbol. Específicamente, los objetivos de este trabajo son:

- Realizar un estudio sobre la arquitectura Detection Transformer y compararlo con Faster R-CNN.
- Entrenar dichos modelos sobre el dataset de la competición de SoccerNet.
- Comparar dichas arquitecturas según las métricas establecidas en la literatura.

1.3. Estructura del Seminario

Para brindar al lector una idea global del contenido de este trabajo, a continuación se hace una breve descripción del propósito de cada capítulo presente en este seminario de tesis.

- **Introducción:**

En este capítulo introductorio se comenta sobre las motivaciones que me llevaron a escoger el tema del seguimiento de jugadores de fútbol para el video análisis.

■ **Estado del Arte:**

En este capítulo consiste en hacer un estudio, de los trabajos que se han venido realizando en este campo y ver que técnicas se han ido aplicando o desarrollando.

■ **Marco Teórico:**

En este capítulo se aborda brevemente los fundamentos para necesarios para entender este proyecto.

■ **Experimentos y Resultados:**

En este capítulo describo los experimentos realizados hasta esta entrega y los resultados que obtuve de ellos.

■ **Conclusiones y Trabajo a Futuro:**

En este capítulo se exponen las conclusiones obtenidas de este trabajo. Adicionalmente, se proponen trabajos a futuros que se le pueden realizar al trabajo entregado.

Capítulo 2

Estado del Arte

En el presente capítulo se hará una revisión de los trabajos desarrollados sobre la detección de los jugadores de fútbol en videos. Para ello empezaremos describiendo sobre el video análisis. Luego presentamos el uso del *Machine learning* en el fútbol. Posteriormente, hablaremos sobre la problemática a resolver, siendo esta la detección de jugadores y finalmente se comentará sobre algunos trabajos para resolver este problema.

2.1. Video Análisis en el Fútbol

El análisis de videos de fútbol se ha vuelto cada vez más importante debido a los avances en tecnologías digitales de grabación. Esto ha permitido la extracción de datos valiosos a través de dichos videos, los cuales son utilizados para tareas como la generación de *highlights*, el seguimiento de jugadores, inserción de contenido o el análisis táctico [4]. Estas tareas son esenciales tanto para periodistas al momento de informar sobre el juego, como para entrenadores para obtener información sobre el rendimiento de sus jugadores y tomar decisiones estratégicas.

El video en el fútbol ha experimentado un gran crecimiento en los últimos años, convirtiéndose en un negocio rentable en el mercado. Con la aparición de diversas herramientas y software como Stats perfom¹, TRACAB², InStat³, entre otros, se ha podido ofrecer servicios para el seguimiento, generación de datos y diseño de esquemas tácticos de manera más precisa y eficiente.

¹Stats perfom <https://www.statsperform.com/>

²TRACAB: <https://tracab.com/>

³InStat: <https://www.instatsport.com/en/>

2.2. Machine Learning en el Fútbol

Campos como el *Machine learning* han cobrado un gran impulso en los últimos años en diversas áreas [5]. En el mundo de fútbol podemos encontrar aplicaciones como [6] donde se intenta usar algoritmos de clusterización para agrupar y buscar similitudes entre los jugadores con el objetivo de permitir a los clubes fichar en relación con sus necesidades. En lo que concierne a tareas de *Computer vision*, podemos ver en [7, 8, 9, 10] acerca de la detección de eventos en los partidos.

Siendo esta tarea de interés para el análisis de ciertas situaciones en el juego y/o emplearlo para automatizar la generación de highlights. Por último podemos encontrar la tarea del seguimiento de jugadores de fútbol [11] el cual consiste en dado una serie de frames a los jugadores detectados, se les asignará un ID. Para lograr dicha tarea primero debemos hablar sobre la detección de jugadores, el cual se comentará a continuación en la siguiente sección.

2.3. Detección de jugadores

La detección de objetos es una de las tareas del *Computer vision* al cual la comunidad científica ha prestado mucha atención en las últimas décadas. Siendo los modelos basados en redes neuronales los que mejor resultado dieron sobre esta tarea [12]. A partir de esta podemos encontrar aplicaciones como la detección de rostros, mascarillas, matrículas entre otros. En lo que concierne al mundo del deporte podemos encontrar aplicaciones de esta en el hockey, rugby, fútbol, etc. [13, 14, 15, 16]. En el contexto del fútbol, la tarea en concreto consistiría en partir de una serie de frames el modelo debe detectar la posición de los jugadores en el campo. De [3, 17] presentan el uso del algoritmo de YOLOv3 para resolver este problema; sin embargo, tienen el problema de que el algoritmo sufre el problema de encontrar jugadores que están en pequeñas proporciones además de la oclusión de los jugadores. De [18] podemos ver que usan la arquitectura de Feature Pyramid Network para poder detectar tanto los jugadores y el balón, además de intentar resolver el problema de detectar objetos pequeños. Así mismo, [19]

propone utilizar las técnicas de destilación usando la *Faster R-CNN* con *RestNet 50* y 18 capas como *backbone* (maestro-estudiante respectivamente) para intentar detectar a los jugadores que se encuentran con dimensiones relativas pequeñas en el frame del video. [20] también utiliza la destilación sobre *YOLOv3* y *YOLOv3-tiny* (como maestro y estudiante) aplicándolo sobre una *wide-angle fisheye camera* para dar otro enfoque sobre el tipo de cámara ha emplear. Por último, surgen iniciativas como [21] que tratan de incorporar los modelos basados en *Transformers* para la detección de los jugadores en otro deporte como lo es el fútbol americano.

Capítulo 3

Marco Teórico

En este capítulo se abordará el marco teórico necesario para comprender este proyecto.

3.1. Redes Convolucionales

Las redes convolucionales son un tipo de algoritmo perteneciente al campo del Deep Learning. El cual toman como base las operaciones de la convolución sobre las imágenes y así poder encontrar patrones. Para poder entender el funcionamiento de este tipo de red se va a detallar brevemente algunas características que la componen y posteriormente se pasará a definir las arquitecturas que se han realizado en este trabajo y que están basadas en este tipo de red.

3.1.1. Convolución

La operación de convolución (para imágenes) consiste en aplicar filtros de tal forma que estos puedan extraer información más relevante sobre esta. Dicha operación matemática, como se puede apreciar en la Fig. 3.1, consiste en que el filtro (kernel) recorra toda la imagen, en proporción del tamaño que definamos, y que en dicho proceso vaya multiplicándose con los píxeles de esta y que nos genere una nueva imagen filtrada.

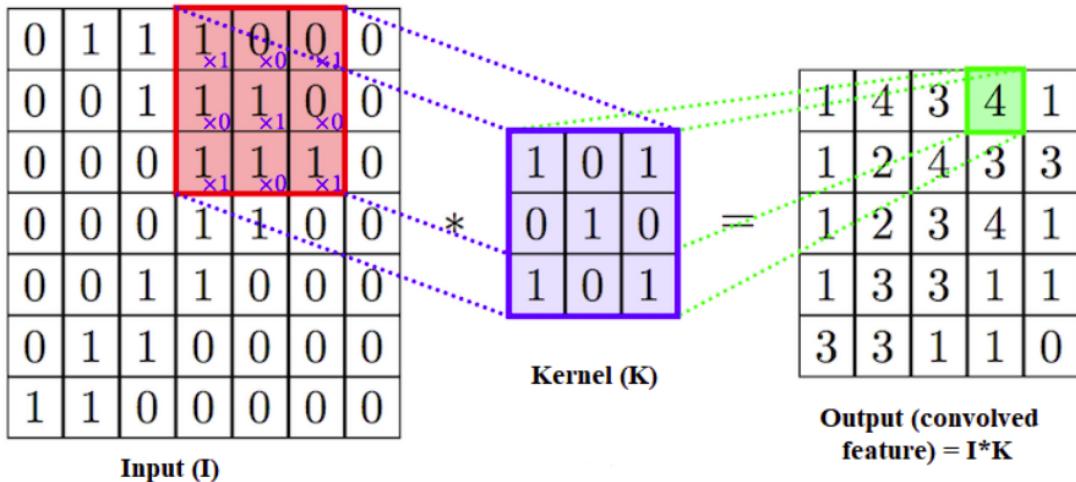


FIGURA 3.1: Operación de convolución [22]

Dependiendo del tipo de filtro que nosotros escogamos. Obtendremos diferentes tipos de efectos tales como detectar bordes¹ (ver Figura 3.2), suavizado, eliminación de ruido, etc.

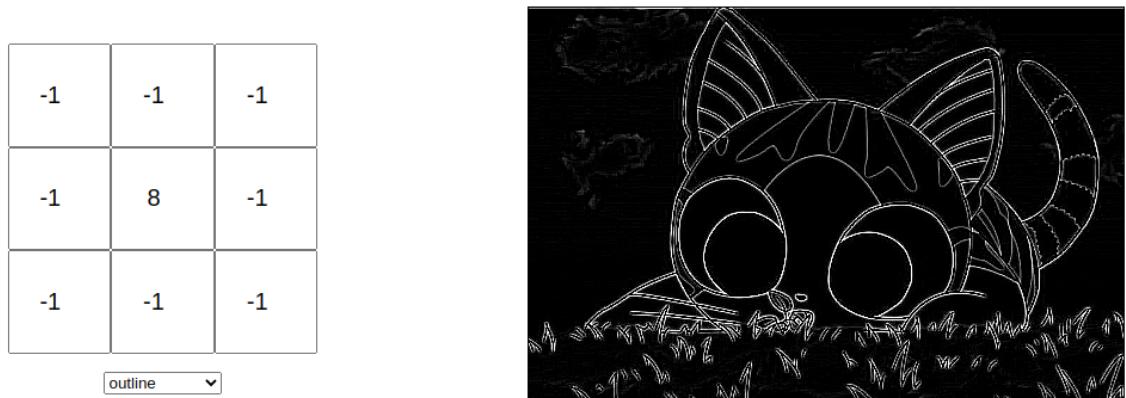


FIGURA 3.2: Filtro outline

3.1.2. Pooling

La operación de pooling consiste en tomar la salida de la convolución y reducir su tamaño pero preservando la información más relevante de esta. Una de las operaciones de Pooling más conocidas es el Max Pooling (ver Fig. 3.3) el cual consiste en tomar el máximo valor de pixel en una región de la imagen.

¹Esta imagen fue creada gracias a <https://setosa.io/ev/image-kernels/>

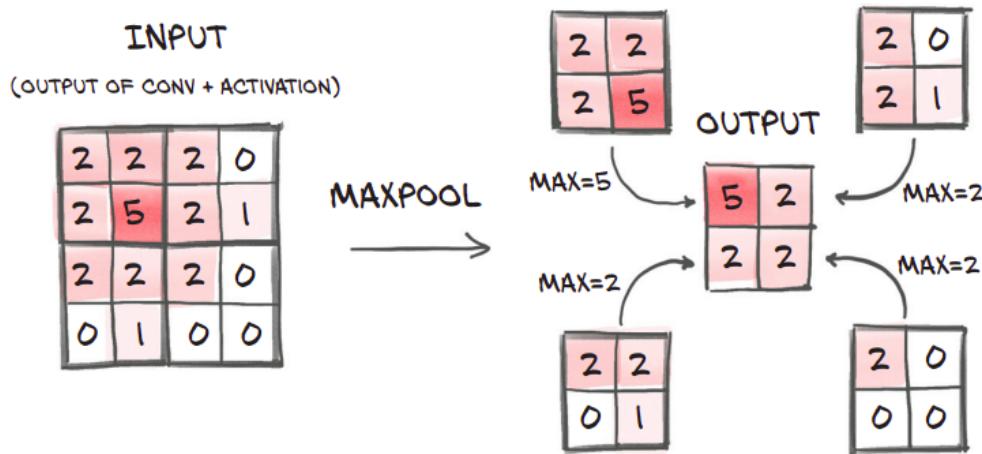


FIGURA 3.3: Operación de Max Pooling [23]

3.1.3. ResNet

Las redes neuronales residuales son un tipo de arquitectura basada en las redes convolucionales propuesta en 2015 [24]. Está diseñada para poder saltarse algunas conexiones o una serie de bloques de capas convolucionales y pooling (ver Figura 3.1) con el objetivo de tratar de resolver el problema del desvanecimiento del gradiente. Siendo, dicha arquitectura, aprovechada en muchas tareas del *Computer vision* como un extractor de características además de su escalabilidad, es decir, la facilidad con la que se le puede ir añadiendo más capas, es por ello, que aún es aprovechada para técnicas como la destilación como podemos ver en [25].

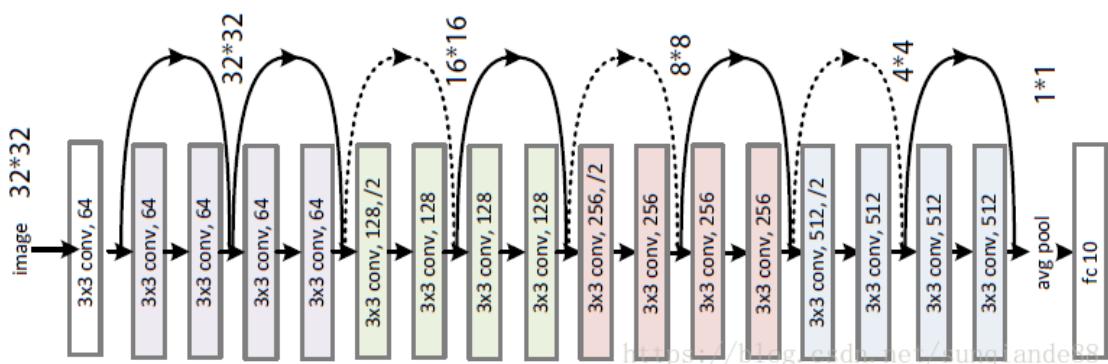


FIGURA 3.4: ResNet18 [26]

3.1.4. Faster R-CNN

Esta arquitectura pertenece a la familia de las R-CNN el cual fue publicado en 2016. Está conformada, como podemos apreciar en la Figura 3.2, por una arquitectura de red convolucional como *BackBone* (por lo general una VGG o ResNet) la cual generará mapas de características que serán aprovechados por la red neuronal de región propuesta. El cual se encargará de verificar si hay un objeto, además de predecir los anchor boxes. Finalmente, se pasa por una serie de capas de red neuronal, la cual mediante una *regression layer* y *softmax* predecirán los *boxes* del objeto y su clasificación [27].

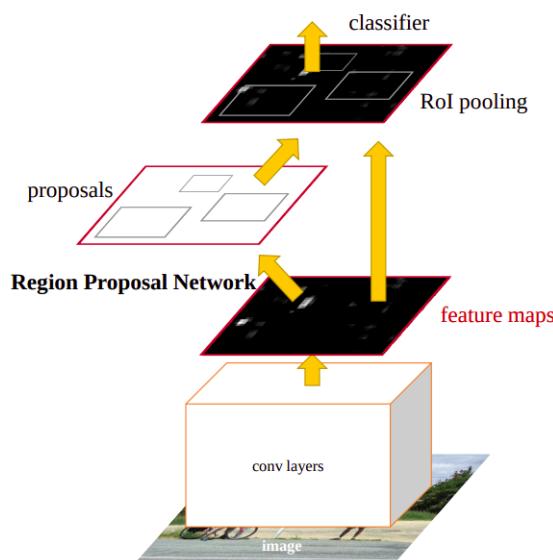


FIGURA 3.5: Faster R-CNN [27]

3.1.5. Feature Pyramid Networks

Identificar objetos sobre diferentes tamaños es uno de los más grandes retos que encontramos en esta tarea. Por ello, [28] propuso un *feature extractor* que construye una pirámide de mapas de características a partir de la entrada de la red, donde cada nivel de la pirámide contiene información de diferente escala y resolución. Esta arquitectura se conforma dos redes: La *Bottom-up*, la cual es un conjunto CNN que se encargaran de extraer las características en diferentes escalas y resoluciones de la imagen de entrada y así construir la pirámide de características a partir de

ella. Por otro lado, tenemos el *top-down* el cual se encarga de concatenar los mapas de características a diferentes niveles usando la concatenación y el upsampling.

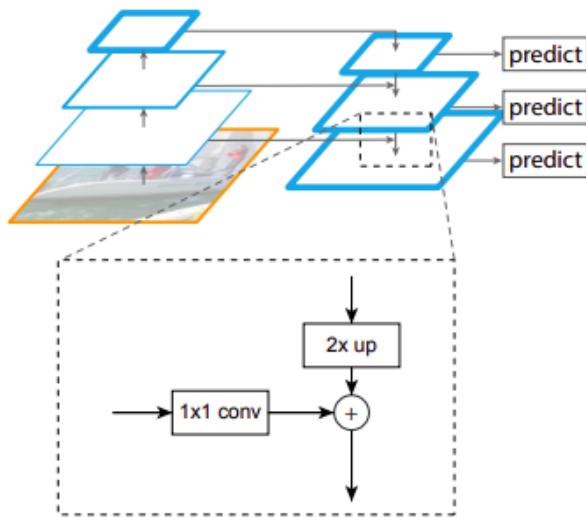


FIGURA 3.6: Arquitectura FPN [28]

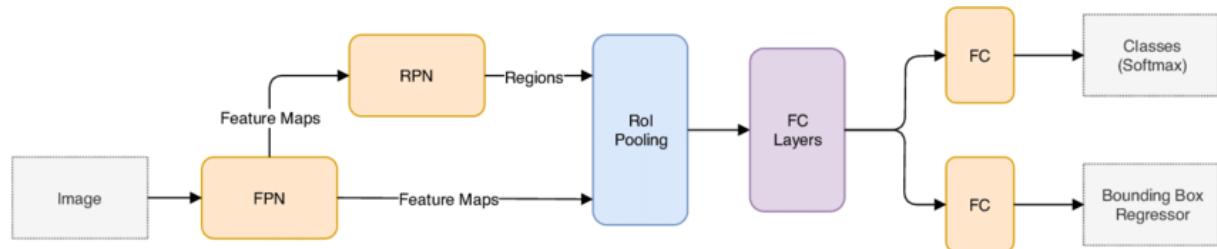


FIGURA 3.7: Representación de la arquitectura Faster R-CNN con una FPN [29]

3.2. Redes Transformers

Los modelos Transformers son una de las arquitecturas más utilizadas en la actualidad, siendo el campo del procesamiento del lenguaje natural (NLP) donde se están realizando grandes aportaciones (e.g GPT-3 , BERT [30, 31]). Además de su reciente impulso sobre el campo del *Computer vision* como se puede ver en los *Vision transformers* [32].

Una de las ventajas de estos modelos es que se pueden parallelizar, a diferencia de otras arquitecturas predecesoras, como las RNN, que no podían debido a su

naturaleza secuencial. Los autores de la arquitectura de *transformers* [33] propusieron un modelo basado en modelos de codificador - decodificador, apoyándose en los mecanismos de atención. Para poder entender dicha arquitectura a continuación describiremos las partes más importantes de esta.

3.2.1. Mecanismos de Atención

Esta estrategia fue propuesta para resolver el problema de memoria que sufren las RNN ante largas secuencias de texto.

En [34] nos explica que se toma una serie de vectores que representaremos como $h_i, \forall i \in 1 \leq i \leq n, \forall n \in \mathbb{N}$ y a los cuales en un instante de tiempo t dado un vector s_t se multiplicará con dichos vectores h_i obteniendo una puntuación (A mayor puntaje se obtendrá una mayor relación).

$$\text{score}(s_t, h_i) = s_t^T h_i \quad (3.1)$$

Seguidamente, se calcula los pesos de atención mediante la función *Softmax*.

$$\alpha_{ti} = \frac{e^{\text{score}(s_t, h_i)}}{\sum_{j=1}^n e^{\text{score}(s_t, h_j)}} \quad (3.2)$$

Finalmente, mediante una combinación lineal se calculará los vectores de contexto c_{ti} .

$$c_{ti} = \sum_{i=1}^n \alpha_{ti} h_i \quad (3.3)$$

3.2.2. Codificación posicional

A diferencia de su arquitectura predecesora, la RNN donde la posición estaba enmarcada por su procesamiento secuencial. En el caso de los *transformers*, gracias a los mecanismos de atención, se procesará sobre toda la entrada dada. Sin embargo, debemos tomar en cuenta que el tipo de datos que recibirá son de naturaleza secuencial (i.e si importa el orden). Se propuso el uso de funciones

senos y cosenos para representar la posición para nuestras secuencias de datos de entrada.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \quad (3.4)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \quad (3.5)$$

- **pos** Posición de la palabra en la oración
- **i**: representa las diferentes dimensiones del codificador posicional y los embeddings.
- d_{model} : Tamaño del word embedding; como la del codificador posicional

3.2.3. Atención de producto punto escalado

Este fue el mecanismo de atención propuesto en el paper de los Transformers. Dando como ejemplo el caso de un tipo de entrada en forma de secuencia de texto. Pasamos el vector de entrada (i.e los word embedding's) a través de 3 redes neuronales para generar tres matrices Q, K y V.

Se multiplicaran tanto Q como K de tal manera que nos dé un puntaje que representará el grado de asociación entre las palabras, las cuales reescalaremos dividiendo sobre un el tamaño del vector K. Después, usamos la función softmax para llevarlo a probabilidades donde valores más cercanos a 1 serán a los que se les estará prestando más atención. Por último, a esta matriz de puntaje se le multiplica por el vector V para generar nuevos vectores (a través de una combinación lineal entre V y la salida de la Softmax) que contendrán la codificación de información de contexto más importante para cada palabra con otras.

Es representado matemáticamente como:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (3.6)$$

- **Q**: Matriz de Queries.

- \mathbf{K} : Matriz de Keys
- \mathbf{V} Matriz de values
- d_k : Dimensión de keys

3.2.4. Multi-Head Attention

El paper [33] propuso trabajar no solo con una función de atención, sino con múltiples *head attention* de manera que el modelo pueda atender información de diferentes subespacios de representación y pueda aprovechar la capacidad de ser paralelizable para un mejor tiempo de cómputo. Esto permitiría aprender asociaciones de palabras y/o grupos en diferentes niveles. Esta es representada como la concatenación de múltiples *head attention*.

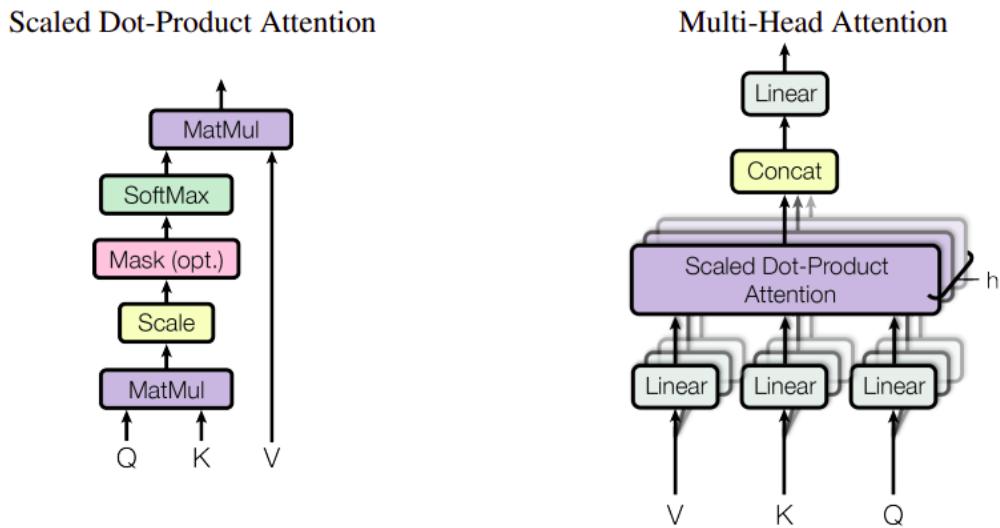


FIGURA 3.8: (izquierda) Atención de producto punto escalado. (derecha) Multi-head attention consta de varios capas de atención que se ejecutan en paralelo.[33]

3.2.5. Codificador-Decodificador

Ya definida las partes más importantes las cuales conforman esta arquitectura. Ahora podemos pasar explicar como está conformada. El paper propone el uso de

un codificador y decodificador que están compuestos por varios módulos que se pueden apilar, los cuales se ven en la Figura 3.9 como Nx, lo que indica que se está apilando N veces, y en el paper original se apilan 6 de ellos. Dentro del codificador tenemos una multi-head attention layer, seguida de una norm layer y un residual block para luego pasar a una forward layer que también tiene una residual y norm layer. El decodificador, por su parte, también tiene 6 capas, sin embargo, se agrega una capa adicional de *multi-head attention* entre la *forward layer* y la *first multi-head attention*. Adicionalmente, se le agrega una *masking layer* a la primera *multi-head attention*, esto debido a que para el entrenamiento oculta los *tokens* a predecir. En la última capa del *decoder* se le agrega una capa lineal y una *softmax* la cual representará la probabilidad de salida sobre el vocabulario del *Transformers*.

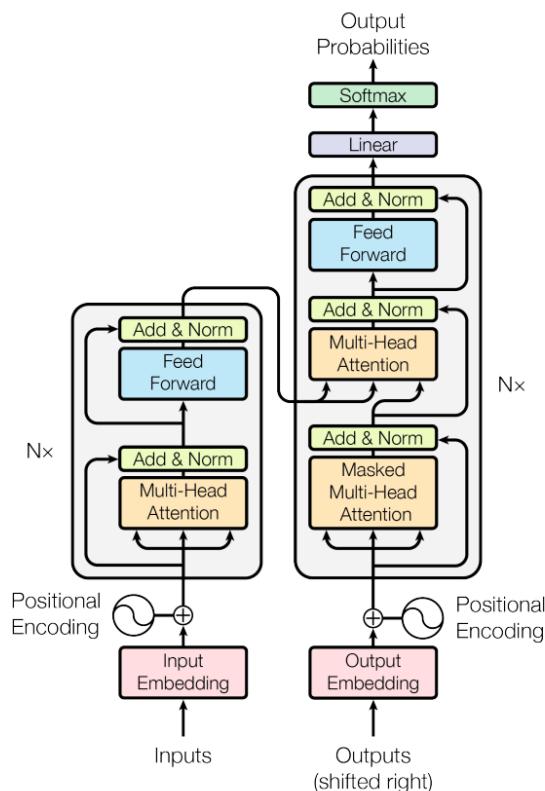


FIGURA 3.9: Arquitectura del modelo Transformers [33]

3.2.6. Detection Transformers

Como se comentó en el inicio de esta sección en los últimos años se ha popularizado el uso *transformers* para el campo de la visión [35]. Siendo DETR uno

de los primeros algoritmos en implementar las redes *transformers* para resolver la tarea de detección.

A diferencia de los modelos tradicionales basados en redes convolucionales. DETR propone el uso de una arquitectura Transformers que reemplazará el uso de los métodos convencionales como el non-maximum suppression o la generación de anchor boxes (Estrategias usadas por los algoritmos basados en redes convolucionales). Por otro lado, en contraste con el transformer convencional propuesto por [33] donde recibe una entrada de texto. DETR hace uso de un extractor de características (ResNet50), como se ve en la Fig. 3.10, al cual se le pasará como entrada una $x_{imagen} \in R^{3XH_0XW_0}$ de la cual se generará mapas de características $f \in R^{CXHXW}$ donde $C = 2048$, $H = \frac{H_0}{32}$ y $W = \frac{W_0}{32}$ después se reduce la dimensionalidad de C a d_{model} y se aplana dichos mapas, sobre el HXW, obteniendo un vector $z \in R^{d_{model}XHW}$ para sumarse con los *positional encoding*. Luego es pasado sobre el *encoder* del *transformer* donde se aplicará el mismo proceso del encoder original, como se puede ver en la Fig. 3.11.

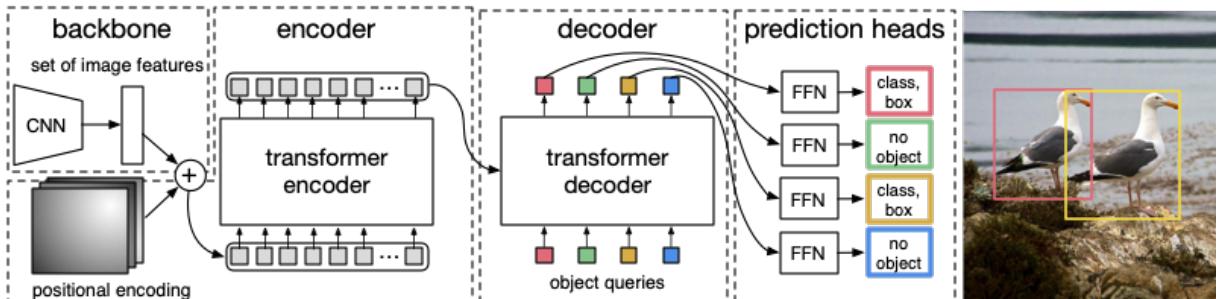


FIGURA 3.10: Detection Transformer [36]

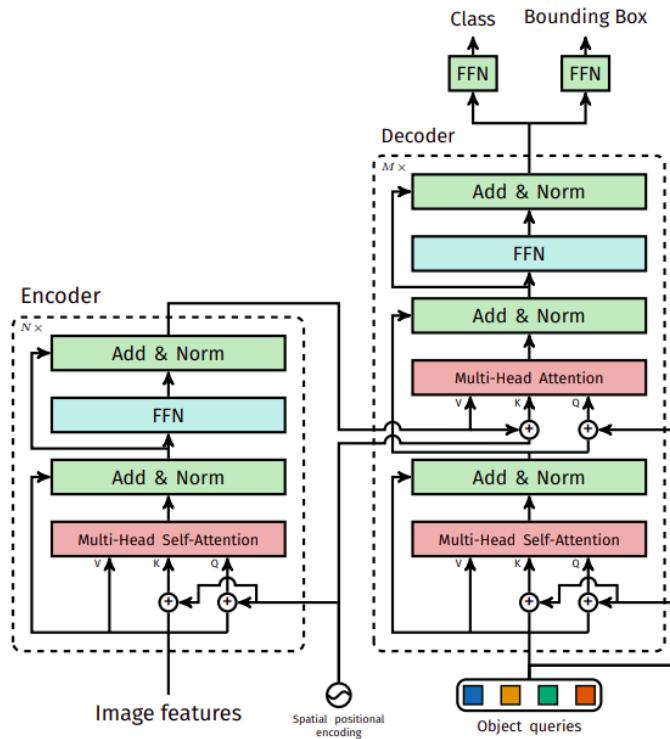


FIGURA 3.11: Arquitectura del *transformer* DETR [36]

En la Fig. 3.12 podemos apreciar una de forma intuitiva como se estaría representando el mecanismo self-atención sobre los píxeles que conforman al objeto que está detectando el encoder, en este caso de ambos Pugs.



FIGURA 3.12: Visualización de self-attention del codificador ²

²Esta imagen fue creada gracias al Google Colaboratory de Facebook research, *DETR's hands on Colab Notebook*. https://colab.research.google.com/github/facebookresearch/detr/blob/colab/notebooks/detr_attention.ipynb#V1OPEeVOYTEV

3.3. Transfer Learning

Uno de los retos que podemos encontrar en el campo del *deep learning* es el poder hacer que estos sean capaces de poder generalizar en dominios muy específicos donde el corpus de datos no sea tan accesible. Sin embargo, para que estos modelos puedan tener buenos resultados se necesita de grandes datasets lo que conlleva a la gran dificultad para entrenar modelos desde cero debido al cómputo que requiere. Por tanto, surge la idea de poder transferir el conocimiento que tiene una red en una tarea X hacia otra tarea Y. Dentro de las técnicas de transfer learning tenemos el finetuning el cual consiste en tomar una red entranda en una tarea y re entrenar toda la red neuronal con o sin capas añadidas. Otra técnica sería congelar los pesos del modelo y agregar algunas capas a la red y entrenar solo dichas capas.

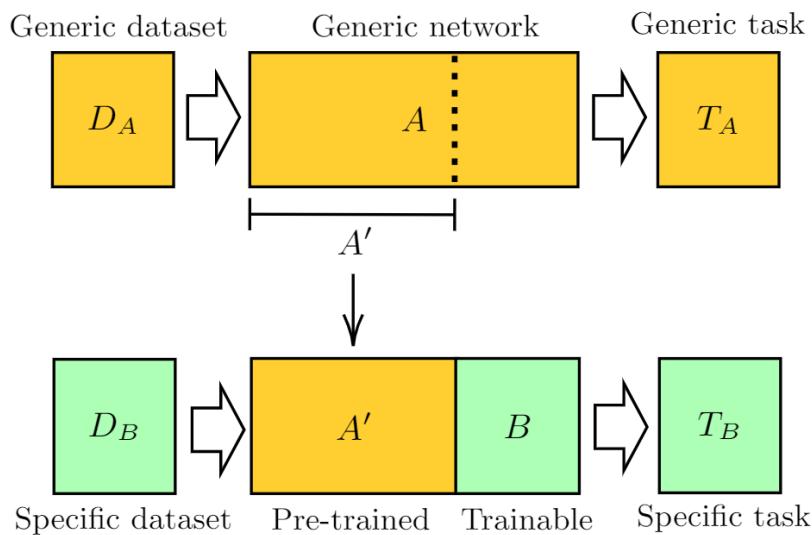


FIGURA 3.13: Representación general del método de aprendizaje de transferencia, La red A está pre entrenada en un conjunto de datos D_A y para una tarea T_A . Se toma una parte de ella para especializarla sobre una tarea T_B con un dataset D_B añadiéndole algunas capas que serán entrenadas [37].

Capítulo 4

Metodología

En este capítulo detallaremos el proceso que se realizó para la detección a los jugadores (En la Fig. 4.1 podemos ver el flujo de trabajo). Se describirá el conjunto de datos usado, así como también las especificaciones del entrenamiento y las métricas.

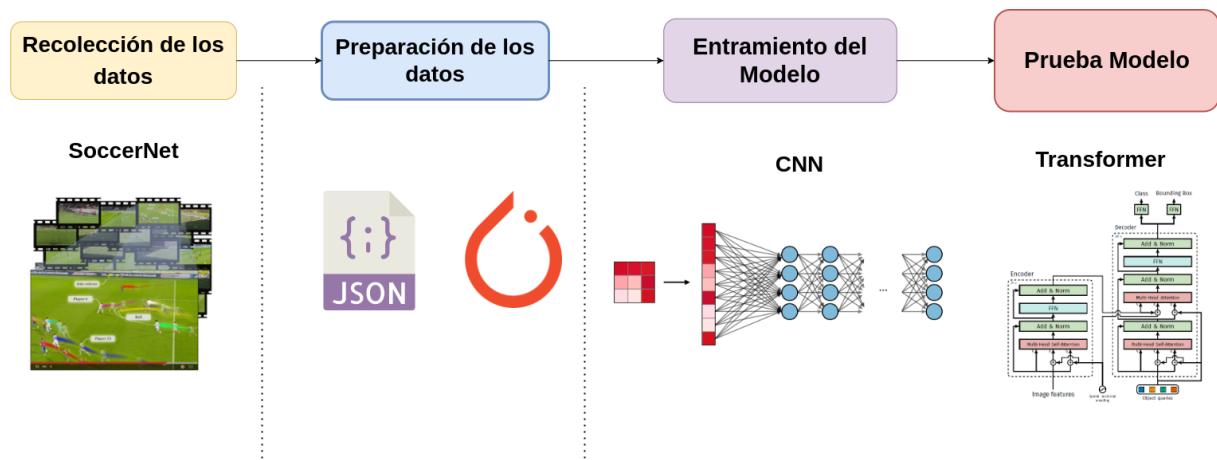


FIGURA 4.1: Pipeline del trabajo

4.1. Conjunto de datos

Para la recolección del conjunto datos se usó el de la competición de SoccerNet¹ [11] este consta de diferentes tareas como *Field Localization*, *Camera Calibration*, *Player Re-Identification*, *player tracking* (se tomó este último). Este consta 12 juegos separados en entrenamiento y test. En particular, esto representa 57 clips de 30

¹Link para descargar el conjunto de datos de SoccerNet: <https://www.soccer-net.org>

segundos para el entrenamiento, 49 clips para el test. Cada secuencia de las imágenes que fueron extraídas de los videos están enúmeradas desde el 000001.jpg hasta 000750.jpg y los *bounding boxes* a predecir siguen el formato de x_min, y_min, ancho, alto los cuales están guardados un archivo txt. La clases no fueron tomadas en cuenta ya que los autores no realizaron un etiquetado sobre este dataset.

4.2. Preparación del conjunto de datos

Para la preparación del conjunto de datos se asoció la ubicación de las imágenes con su respectivo bounding boxes, los cuales fueron guardados en archivos JSON (train/test). Finalmente, al archivo JSON del conjunto de test se dividió en (val / test).

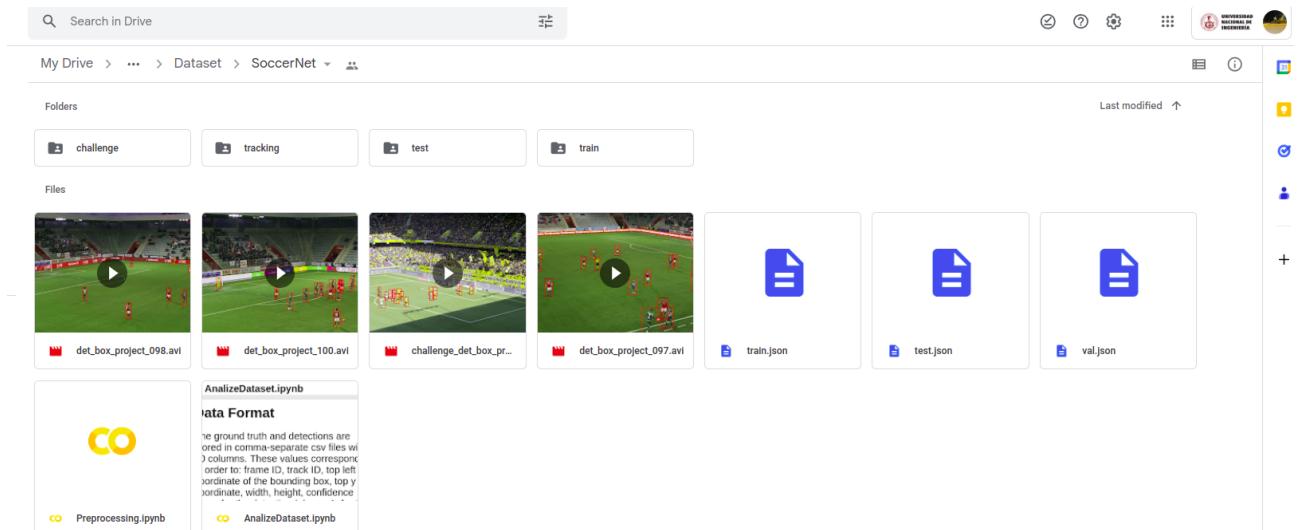
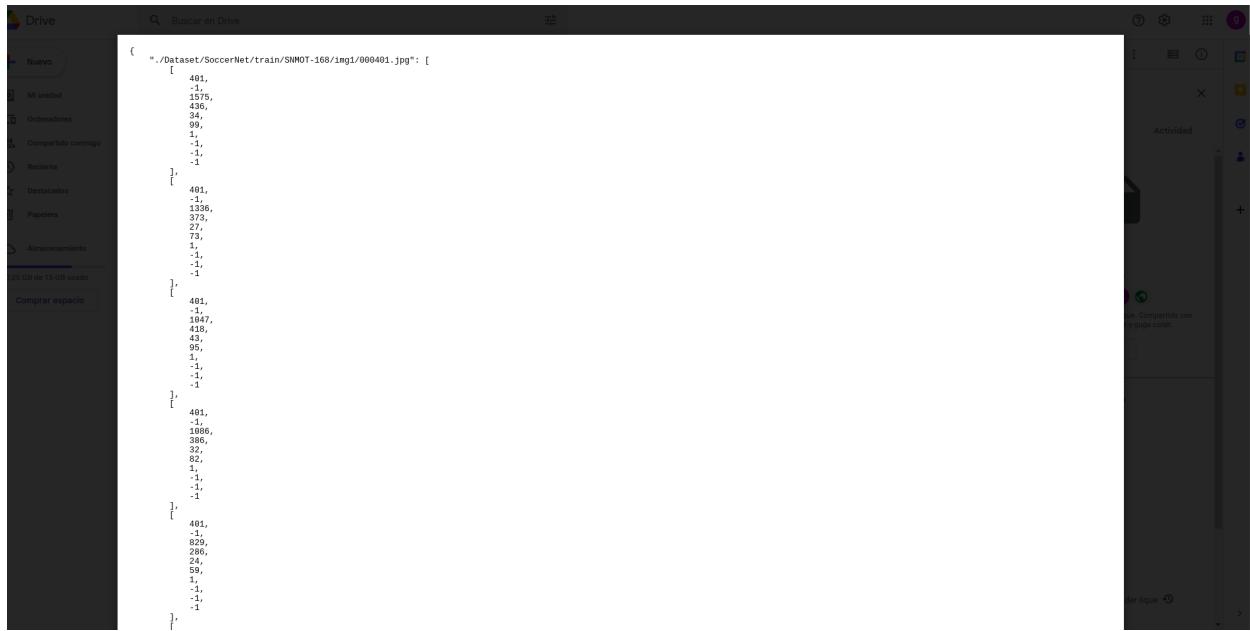


FIGURA 4.2: Conjunto de datos de SoccerNet



```

{
  "./Dataset/SoccerNet/train/SN MOT-168/img1/000401.jpg": [
    {
      "401, 1, 1375, 436, 34, 39, 1, 1, 1, -1
    },
    {
      "401, 1, 1336, 373, 27, 73, 1, 1, 1, -1
    },
    {
      "401, 1, 1047, 418, 42, 92, 1, 1, 1, -1
    },
    {
      "401, 1, 1896, 386, 32, 82, 1, 1, 1, -1
    },
    {
      "401, 1, 829, 286, 24, 59, 1, 1, 1, -1
    }
  ]
}
  
```

FIGURA 4.3: Nuevo formato creado para leer el conjunto de datos como se puede ver, se tiene tanto la dirección donde se ubica la imagen y su Ground thrue de coordenadas a detectar.

4.3. Métricas

4.3.1. Intersección sobre la Unión (IoU)

Esta métrica consiste en poder calcular el porcentaje del area concuerdan tanto el bounding box predicho y el esperado. Se puede expresar como :

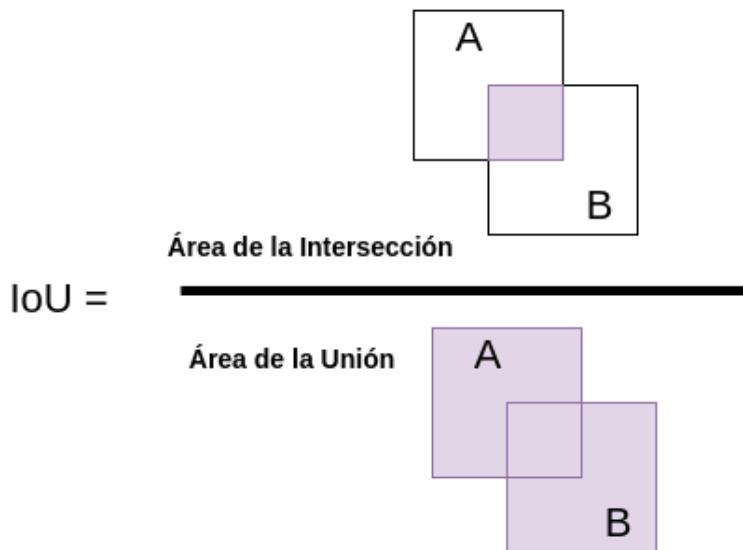


FIGURA 4.4: Intersección sobre la unión

En base a esta fórmula se pueden presentar los siguientes casos:

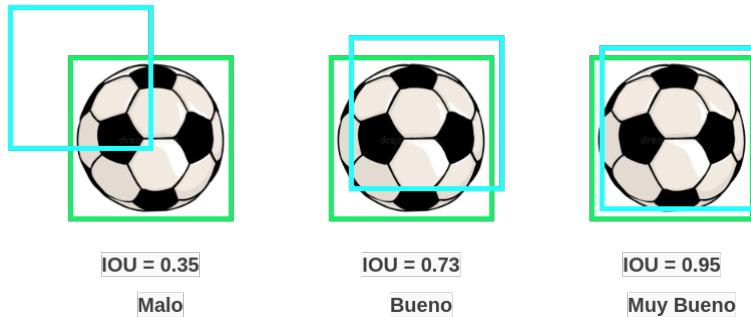


FIGURA 4.5: Ejemplo de IoU

Se puede ver de la Fig 4.3 que dependiendo del área de intersección entre los valores predichos y esperado tendremos un bajo o alto valor en el IoU.

4.3.2. Mean Average Prescision

Para poder entender la métrica de mAP debemos empezar calculando algunos valores de la matriz de confusión. Para ello a cada bounding box predicho se le calcula su IoU score y bajo un umbral de IoU denotado u definiremos los verdaderos positivos (TP), falsos positivos (FP) y falsos negativos (FN).

- **Verdadero Positivo (TP):** Dado un umbral si el IoU, del bounding box predicho, es mayor a dicho umbral.
- **Falso Positivo (FP):** Dado un umbral si el IoU, del bounding box predicho, es menor a dicho umbral.
- **Falso Negativo (FN):** Si el modelo es incapaz de predecir un bounding box cuando en verdad habia un objeto a detectar.

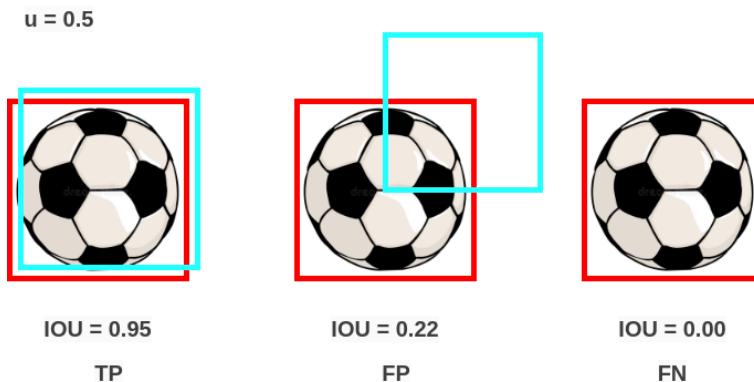


FIGURA 4.6: Análisis del IoU bajo un umbral $u = 0.5$

Ahora ya teniendo los valores de la matriz de confusión se calculó la precision y recall sobre todas las imágenes. Y calculamos el área bajo la curva *precision-recall* (la cual denominaremos como AP). Finalmente realizamos el procedimiento anterior respecto a todas las clases detectadas en la imagen y calculamos el promedio como se ve en la formula de abajo.

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i \quad (4.1)$$

4.3.3. Métrica de COCO

Las métricas de COCO se utilizan a menudo en la investigación y el desarrollo de sistemas de detección de objetos para medir y comparar su rendimiento bajo diferentes enfoques y algoritmos.

En esta métrica de evaluación la podemos catalogar bajo el tamaño del objeto detectado, así como también bajo un umbral, como se puede ver en la Fig. 4.7. La principal métrica de COCO es mAP donde se computa el AP sobre diferentes umbrales con un paso de 0.05 representados como $AP@[.5:.05:.95]$ lo cual se puede representar como:

$$mAP_{COCO} = \frac{mAP_{0.50} + mAP_{0.55} + mAP_{0.60} + \dots + mAP_{0.85} + mAP_{0.90} + mAP_{0.95}}{10} \quad (4.2)$$

²Métricas de la competición de COCO <https://cocodataset.org/#detection-eval>

```

Average Precision (AP):
    AP                                % AP at IoU=.50:.05:.95 (primary challenge metric)
    APIoU=.50                      % AP at IoU=.50 (PASCAL VOC metric)
    APIoU=.75                      % AP at IoU=.75 (strict metric)

AP Across Scales:
    APsmall                         % AP for small objects: area < 322
    APmedium                        % AP for medium objects: 322 < area < 962
    APlarge                          % AP for large objects: area > 962

Average Recall (AR):
    ARmax=1                         % AR given 1 detection per image
    ARmax=10                        % AR given 10 detections per image
    ARmax=100                       % AR given 100 detections per image

AR Across Scales:
    ARsmall                         % AR for small objects: area < 322
    ARmedium                        % AR for medium objects: 322 < area < 962
    ARlarge                          % AR for large objects: area > 962

```

FIGURA 4.7: Métricas de COCO ²

4.4. Entrenamiento

Para el entrenamiento se usó Google Colaboratory. El cual me proveyó de una GPU tesla T4 y cuda v11.2. En el experimento se usó 9073 imágenes para el entrenamiento, 1072 imágenes para la validación y 1288 para el test debido al costo en tiempo de etrenamiento sobre los modelos. Por último se aplicó la técnica del *finetuning* sobre Faster R-CNN y DETR.



FIGURA 4.8: Google colaboratory

Capítulo 5

Resultados y Discusiones

En el presente capítulo se comentará acerca de los experimentos, resultados y discusiones del presente trabajo.

5.1. Experimentación

Se probaron diferentes configuraciones. Para DETR, se usó los mismo mostrados en el paper orginal respecto al #de lotes, el tamaño de imagen (800x800) así mismo variando la taza de aprendizaje y optimizador donde, finalmente, se terminó decantando por los mostrados en la tabla (Ya que no hubo una gran variación en el resultado del entrenamiento) además de entrenarse por 20 épocas (1h por época).

Por otro lado, para la Faster R-CNN, al igual que DETR, se varió la taza de aprendizaje y el optimizador. Al igual que DETR mantuvo el tamaño de lotes (8) e imágenes (800x800) y se entrenó por 5 épocas (35 min por época).

Modelo	#épocas	#lotes	Taza de aprendizaje	Optimizador
Faster RCNN+ResNet50+FPN	5	8	2e-5	Adam
DETR	20	8	1e-5	AdamW

CUADRO 5.1: Hiperparámetros y Optimizadores usados en el experimento

5.2. Resultados de los modelos

En esta sección se mostrará los resultados de los modelos (Faster R-CNN y DETR) sobre el conjunto de test de SoccertNet aplicando las métricas de COCO.

Model	#parámetros	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN+ResNet50+FPN	41076761	55.74	91.74	61.22	26.64	59.25	57.52
DETR + ResNet50	41286406	49.53	88.97	51.15	17.48	54.07	65.19

CUADRO 5.2: Resultados del entrenamiento de los modelos bajo la métrica de COCO

5.3. Discusiones

- Del Apéndice A.0.2, se puede ver que ambos modelos pueden detectar a los jugadores en el campo, con la diferencia de que en la imagen 1, donde hay muchos jugadores juntos, Faster R-CNN detectó un jugador más que DETR. Además, en la imagen 2, DETR sí logró detectar el balón. Del Apéndice A.0.3 y A.0.4, se puede observar que en la secuencia de frames, DETR ha sido capaz de detectar el balón de forma continua en los 4 frames, a diferencia de Faster R-CNN, que solo lo ha detectado en el frame 1 y 4.

En el Apéndice A.0.5, los errores más comunes que encontramos en Faster R-CNN son la pérdida del balón y de jugadores, que están prácticamente cubiertos (como se puede ver en los frames 2 y 3 respectivamente). Sin embargo, a pesar de esto, este modelo solo predice una vez los jugadores y el balón, lo que le permite obtener un buen resultado. A diferencia de DETR, que ha detectado, en ocasiones, al balón y a jugadores que Faster R-CNN no, tiene muchos retos. No obstante, muestra el potencial de los mecanismos de atención al permitir tener un análisis global de las imágenes y comprender mejor la relación entre los objetos y su escena en comparación con los modelos basados en CNN, que realizan un análisis en pequeñas regiones de la imagen, DETR en ocasiones detecta demás sobre un mismo jugador y es propenso a

detectar el balón de manera errónea, lo cual penalizará su puntaje bajo las métricas de COCO.

- De los resultados obtenidos en la tabla 5.2, se puede ver que Faster R-CNN obtuvo mejores puntajes bajo diferentes métricas de COCO. Para AP_{50} y AP_{75} , podemos ver que Faster R-CNN es un 2.77 % y 10.0 % más preciso que DETR bajo un umbral de 0.5 y 0.75, respectivamente. Además, se puede ver que hay una gran diferencia entre el AP_{50} y AP_{75} en ambos modelos, lo cual es esperable, ya que al aumentar el umbral estamos considerando que más entes detectados en el campo son falsos positivos, lo cual reduce la métrica. Como resultado, el AP global que obtuvimos fue 55.74 y 49.53 para Faster R-CNN y DETR, respectivamente (ver fórmula 4.2). Con respecto a las métricas relacionadas al tamaño, como AP_S , AP_M y AP_L , podemos ver que para el AP_S Faster R-CNN obtuvo un 9.16 % más, lo que significa que es más preciso para detectar solo objetos pequeños que DETR. Esto es interesante, ya que aunque DETR a veces predice el balón donde Faster R-CNN no, DETR es más propenso a equivocarse cuando pierde el balón al momento en el que es pateado y se deforme (ver sección A.0.6, Frame 1). Para el AP_M , Faster R-CNN obtuvo un 5.18 % más, lo que muestra que es más preciso que DETR para detectar jugadores que estén más alejados de la cámara. Sin embargo, en la métrica AP_L , DETR tuvo un mejor resultado que Faster R-CNN, lo que indica que DETR detecta con un 7.67 % más precisión a los objetos más grandes, en este caso, a los jugadores que estén más cerca de la cámara.
- Sobre lo comentado en el item anterior Faster R-CNN ha obtenido mejores resultados que DETR. Esto es debido a que si bien DETR es un enfoque innovador que trata de implementar los modelos transformers para la detección. Faster R-CNN es un modelo más antiguo, lo que ha permitido ser utilizado en diferentes trabajos previos y del cual es bien conocido su presición y fiabilidad. En cambio, DETR al ser un modelo más reciente y aún estar en desarrollo, podemos encontrar problemas como su estabilidad, lo que conlleva que dicho modelo tenga dificultades en converger a una solución

óptima, y en su lugar, quedarse en atrapado en una solución local de bajos resultados como sucedió en este caso. Además de que los modelos basados en *transformers* es sabido que requieren de mucho tiempo de entrenamiento y una gran cantidad de datos. Por otro lado, debido a la complejidad mismo dataset de la competición de SoccerNet el cual no nos provia de una clase que separe a los jugadores de los árbitros y el balón. Esto influyó sobre el puntaje de los modelos a los cuales se les hizo difíciles distinguir entre las diferentes formas que puede encontrar como el balón y los jugadores. Lo cual evidencia lo que se comentó anteriormente sobre la fiabilidad de la Faster R-CNN que se pudo adaptar mejor a este dataset en comparación de DETR.

Capítulo 6

Conclusiones y Trabajo Futuro

En este trabajo, se muestra el potencial que tienen los modelos *transformers* para la detección de jugadores, que si bien no se obtuvo resultados mejores que la Faster R-CNN, en la mayoría de las métricas de COCO, se ha mostrado a través de los experimentos y resultados. La importancia que tiene los mecanismos de atención para un análisis global de la imagen y que permitió encontrar algunos entes que la Faster R-CNN no pudo. No obstante, aún se tienen varios retos para estas arquitecturas como mejorar su lenta convergencia, su precisión al momento de detectar objetos en diferentes proporciones.

6.1. Conclusiones

- CONCLUSION 1: Se muestra la robustez de la Faster R-CNN la cual obtuvo mejores resultados que DETR en el dataset de SoccerNet debido a su mayor estabilidad en la convergencia.
- CONCLUSION 2: Se muestra el potencial que poseen los mecanismos de atención de los vision transformers para la detección.

6.2. Trabajo Futuro

Basándose en la investigación realizada en este seminario de tesis y de cara a futuras investigaciones, propongo las siguientes opciones:

- Entrenar los modelos para realizar una detección más especializada para que distinga entre dos equipos.
- Seguir ampliando el estudio sobre modelos más avanzados de *Vision transformer* para la detección o netamente del balón.
- También se planea reducir el tiempo de inferencia de estos modelos para el *Real-time* investigando técnicas de destilación.
- A futuro se plantea investigar algoritmos para el *trackeo* de jugadores que hagan uso de estos algoritmo de detección.

Bibliografía

- [1] David Diego Jiménez Fernández. «Seguimiento de jugadores en partidos de fútbol mediante procesado de vídeo». En: (2018), <https://biblus.us.es/bibing/proyectos/abreproj/12433/fichero/PFC-2433\bibrangedashJIMENEZ.pdf>.
- [2] Wenbin Huang et al. «Open Dataset Recorded by Single Cameras for Multi-Player Tracking in Soccer Scenarios». En: *Applied Sciences* 12.15 (2022). ISSN: 2076-3417. DOI: 10.3390/app12157473. URL: <https://www.mdpi.com/2076-3417/12/15/7473>.
- [3] Francisco Javier Pascual Vidal. «Detección y seguimiento de jugadores en escenas deportivas usando deep learning». En: (sep. de 2020), <http://hdl.handle.net/2183/26829>.
- [4] Huang-Chia Shih. «A Survey of Content-Aware Video Analysis for Sports». En: *IEEE Transactions on Circuits and Systems for Video Technology* 28.5 (2018), págs. 1212-1231. DOI: 10.1109/TCSVT.2017.2655624.
- [5] Vincent Boucher. «MONTRÉAL.AI ACADEMY: ARTIFICIAL INTELLIGENCE 101 FIRST WORLD-CLASS OVERVIEW OF AI FOR ALL VIP AI 101 CHEATSHEET». En: (ene. de 2021), <https://www.montreal.ai/ai4all.pdf>.
- [6] C Soria Polo. «Diseño y aplicación de técnicas de machine learning para optimizar el Scouting en clubes de fútbol». En: (ene. de 2021).
- [7] Xin Zhou et al. «Feature Combination Meets Attention: Baidu Soccer Embeddings and Transformer based Temporal Detection». En: *CoRR* abs/2106.14447 (2021). arXiv: 2106.14447. URL: <https://arxiv.org/abs/2106.14447>.

- [8] Andreas Husa et al. «Automatic Thumbnail Selection for Soccer Videos Using Machine Learning». En: *Proceedings of the 13th ACM Multimedia Systems Conference*. MMSys '22. Athlone, Ireland: Association for Computing Machinery, 2022, págs. 73-85. ISBN: 9781450392839. DOI: 10 . 1145 / 3524273 . 3528182. URL: <https://doi.org/10.1145/3524273.3528182>.
- [9] Lia Morra et al. «Slicing and Dicing Soccer: Automatic Detection of Complex Events from Spatio-Temporal Data». En: jun. de 2020, págs. 107-121. ISBN: 978-3-030-50346-8. DOI: 10.1007/978-3-030-50347-5_11.
- [10] Ali Karimi, Ramin Toosi y Mohammad Ali Akhaee. «Soccer Event Detection Using Deep Learning». En: *CoRR* abs/2102.04331 (2021). arXiv: 2102.04331. URL: <https://arxiv.org/abs/2102.04331>.
- [11] A. Cioppa et al. «SoccerNet-Tracking: Multiple Object Tracking Dataset and Benchmark in Soccer Videos». En: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Los Alamitos, CA, USA: IEEE Computer Society, jun. de 2022, págs. 3490-3501. DOI: 10 . 1109 / CVPRW56347 . 2022 . 00393. URL: <https://doi.ieeecomputersociety.org/10.1109/CVPRW56347.2022.00393>.
- [12] Zhengxia Zou et al. «Object Detection in 20 Years: A Survey». En: *CoRR* abs/1905.05055 (2019). arXiv: 1905 . 05055. URL: <http://arxiv.org/abs/1905.05055>.
- [13] Fei Wu et al. *A Survey on Video Action Recognition in Sports: Datasets, Methods and Applications*. 2022. DOI: 10 . 48550 / ARXIV . 2206 . 01038. URL: <https://arxiv.org/abs/2206.01038>.
- [14] Maria Koshkina, Hemanth Pidaparthi y James H Elder. «Contrastive Learning for Sports Video: Unsupervised Player Classification». En: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, págs. 4528-4536.

- [15] Adrien Maglo, Astrid Orcesi y Quoc-Cuong Pham. *Efficient tracking of team sport players with few game-specific annotations*. 2022. DOI: 10.48550/ARXIV.2204.04049. URL: <https://arxiv.org/abs/2204.04049>.
- [16] Jacek Komorowski, Grzegorz Kurzejamski y Grzegorz Sarwas. «FootAndBall: Integrated Player and Ball Detector». En: *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP*, INSTICC. SciTePress, 2020, págs. 47-56. ISBN: 978-989-758-402-2. DOI: 10.5220/0008916000470056.
- [17] Thulasya Banoth y Mohammad Farukh Hashmi. *Ball and Player Detection Tracking in Soccer Videos Using Improved YOLOV3 Model*. Abr. de 2021. DOI: 10.21203/rs.3.rs-438886/v1.
- [18] Jacek Komorowski, Grzegorz Kurzejamski y Grzegorz Sarwas. «FootAndBall: Integrated player and ball detector». En: *CoRR* abs/1912.05445 (2019). arXiv: 1912.05445. URL: <http://arxiv.org/abs/1912.05445>.
- [19] Samuel Hurault, Coloma Ballester y Gloria Haro. «Self-Supervised Small Soccer Player Detection and Tracking». En: *CoRR* abs/2011.10336 (2020). arXiv: 2011.10336. URL: <https://arxiv.org/abs/2011.10336>.
- [20] Anthony Cioppa et al. «Multimodal and multiview distillation for real-time player detection on a football field». En: *CoRR* abs/2004.07544 (2020). arXiv: 2004.07544. URL: <https://arxiv.org/abs/2004.07544>.
- [21] Hongshan Liu et al. *Deep Learning-based Automatic Player Identification and Logging in American Football Videos*. 2022. DOI: 10.48550/ARXIV.2204.13809. URL: <https://arxiv.org/abs/2204.13809>.
- [22] Farhana Sultana, Abu Sufian y Paramartha Dutta. «Advancements in Image Classification using Convolutional Neural Network». En: *CoRR* abs/1905.03288 (2019). arXiv: 1905.03288. URL: <http://arxiv.org/abs/1905.03288>.
- [23] Eli Stevens, Luca Antiga y Thomas Viehmann. *Deep learning with PyTorch*. Manning Publications, 2020.

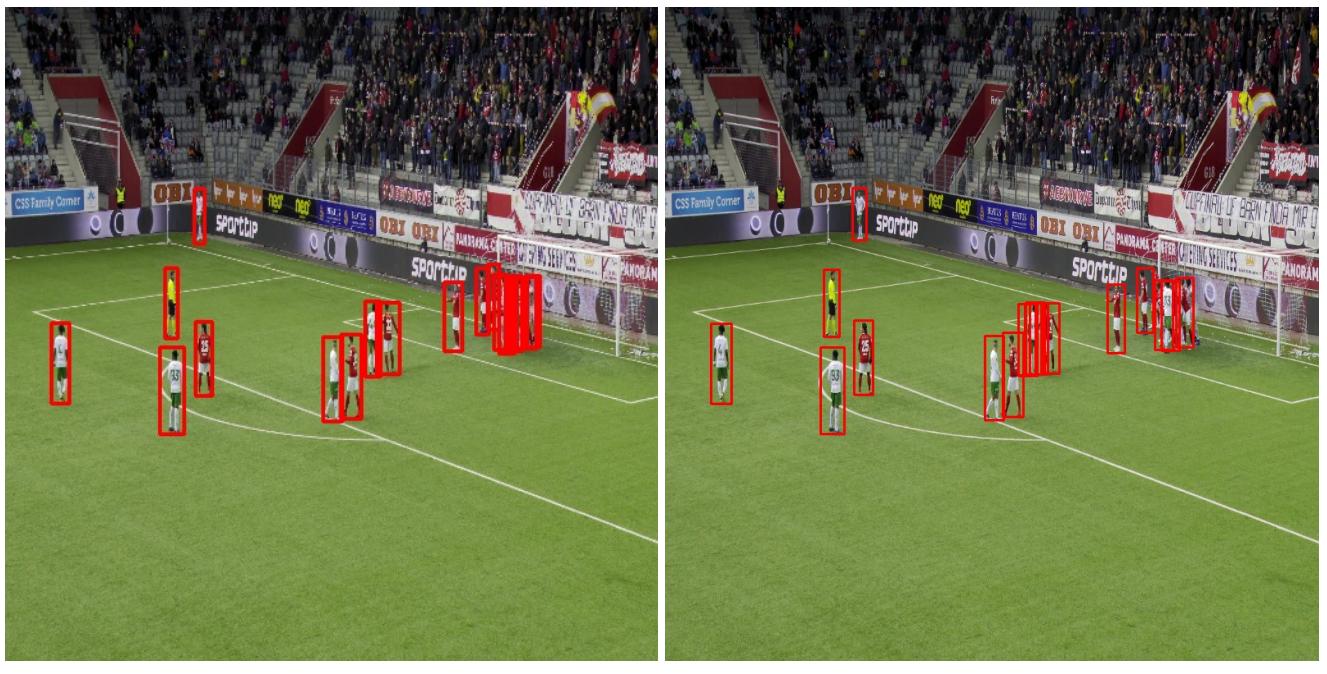
- [24] Kaiming He et al. «Deep Residual Learning for Image Recognition». En: *CoRR* abs/1512.03385 (2015). arXiv: 1512 . 03385. URL: <http://arxiv.org/abs/1512.03385>.
- [25] Tao Huang et al. *Knowledge Distillation from A Stronger Teacher*. 2022. DOI: 10 . 48550 / ARXIV . 2205 . 10536. URL: <https://arxiv.org/abs/2205.10536>.
- [26] Edouard Duchesnay, Tommy Löfstedt y Feki Younes. *Convolutional neural network¶*. URL: https://duchesnay.github.io/pystatsml/deep_learning/dl_cnn_cifar10_pytorch.html.
- [27] Shaoqing Ren et al. «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks». En: *CoRR* abs/1506.01497 (2015). arXiv: 1506 . 01497. URL: <http://arxiv.org/abs/1506.01497>.
- [28] Tsung-Yi Lin et al. «Feature Pyramid Networks for Object Detection». En: *CoRR* abs/1612.03144 (2016). arXiv: 1612 . 03144. URL: <http://arxiv.org/abs/1612.03144>.
- [29] Diego Rafael Lucio et al. «Simultaneous Iris and Periocular Region Detection Using Coarse Annotations». En: oct. de 2019. DOI: 10 . 1109 / SIBGRAPI . 2019 . 00032.
- [30] Katikapalli Subramanyam Kalyan, Ajit Rajasekharan y Sivanesan Sangeetha. «AMMUS : A Survey of Transformer-based Pretrained Models in Natural Language Processing». En: *CoRR* abs/2108.05542 (2021). arXiv: 2108 . 05542. URL: <https://arxiv.org/abs/2108.05542>.
- [31] Jacob Devlin et al. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». En: *CoRR* abs/1810.04805 (2018). arXiv: 1810 . 04805. URL: <http://arxiv.org/abs/1810.04805>.
- [32] Alexey Dosovitskiy et al. «An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale». En: *CoRR* abs/2010.11929 (2020). arXiv: 2010 . 11929. URL: <https://arxiv.org/abs/2010.11929>.

-
- [33] Ashish Vaswani et al. «Attention Is All You Need». En: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: <http://arxiv.org/abs/1706.03762>.
 - [34] Dzmitry Bahdanau, Kyung Hyun Cho y Yoshua Bengio. «Neural machine translation by jointly learning to align and translate». English (US). En: 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015. Ene. de 2015.
 - [35] Salman H. Khan et al. «Transformers in Vision: A Survey». En: *CoRR* abs/2101.01169 (2021). arXiv: 2101.01169. URL: <https://arxiv.org/abs/2101.01169>.
 - [36] Nicolas Carion et al. «End-to-end object detection with transformers». En: *European conference on computer vision*. Springer. 2020, págs. 213-229.
 - [37] Andrea Mari et al. «Transfer learning in hybrid classical-quantum neural networks». En: *Quantum* 4 (oct. de 2020), pág. 340. DOI: 10.22331/q-2020-10-09-340. URL: <https://doi.org/10.22331%2Fq-2020-10-09-340>.

Apéndice A

Resultados de la inferencia de los modelos

A.0.1. Resultados sobre las primeras épocas de entrenamiento



(A) Faster R-CNN

(B) DETR

FIGURA A.1: Inferencia de Faster R-CNN y DETR sobre un frame de SoccerNet test.

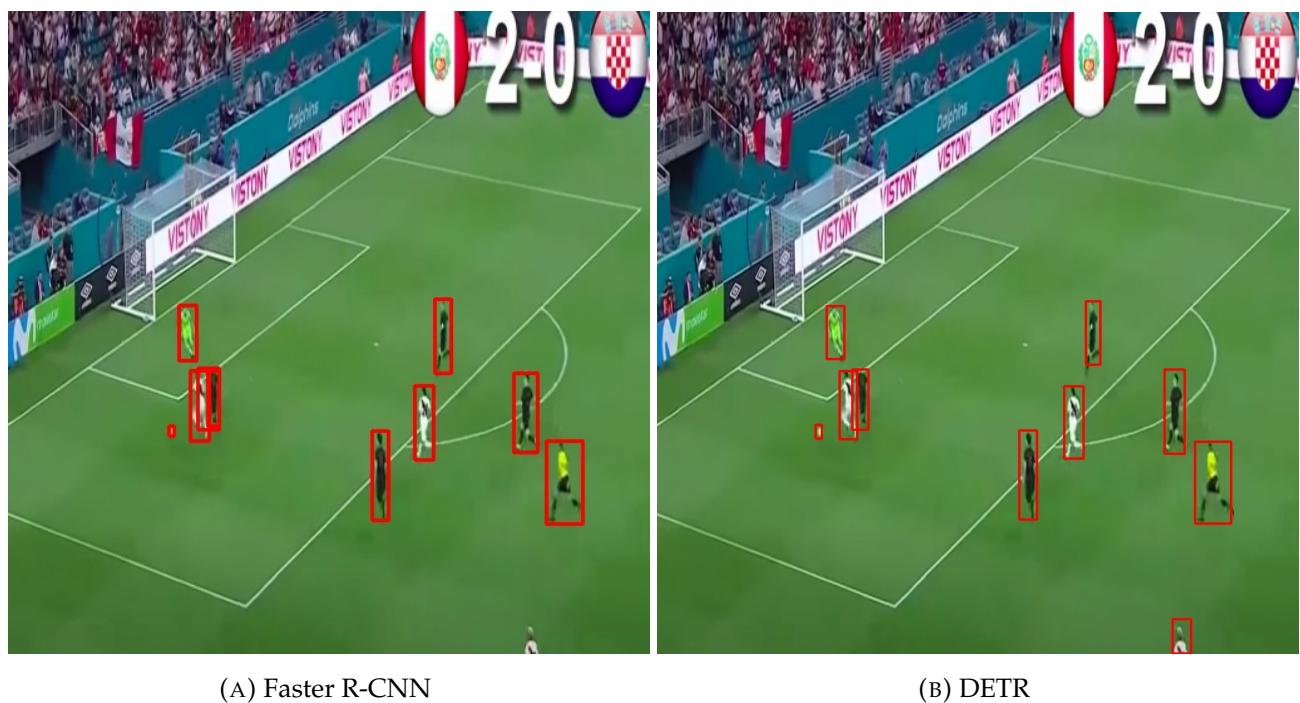
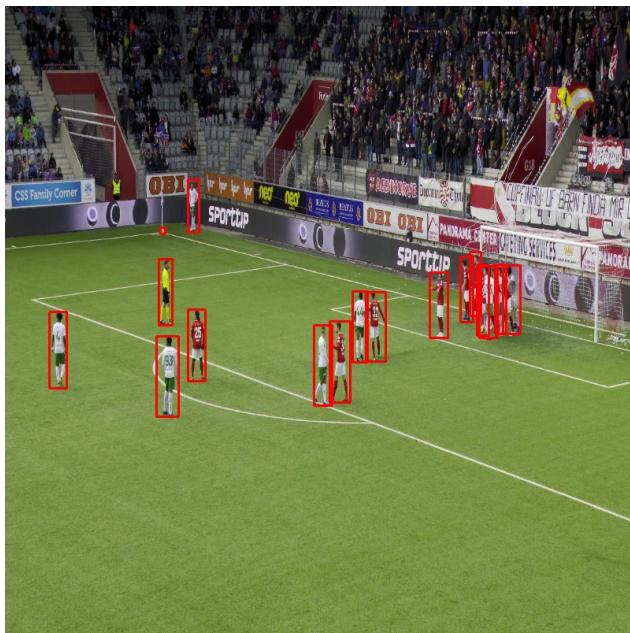


FIGURA A.2: Inferencia de Faster RCNN y DETR sobre Perú vs Croacia.¹

¹Perú vs Croacia, frames extraídos de: <https://www.youtube.com/watch?v=06qhXzs5j08>

A.0.2. Resultados con los mejores pesos del entrenamiento



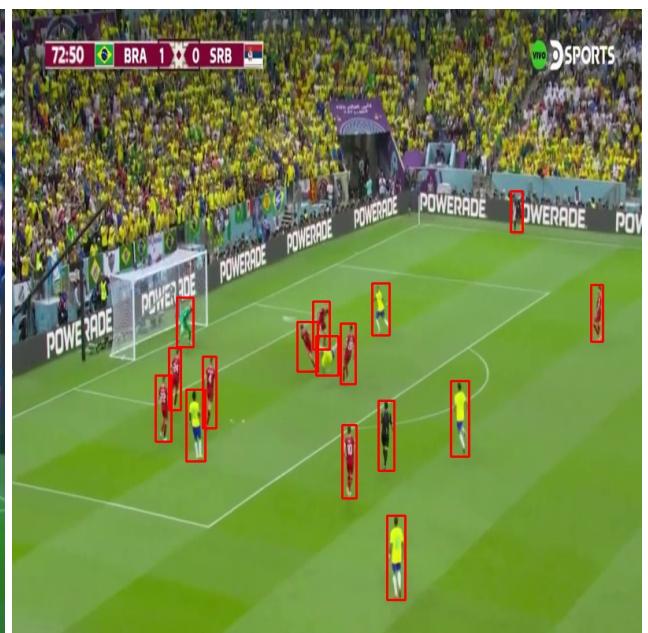
(A) Imagen 1



(B) Imagen 2



(C) Imagen 3



(D) Imagen 4

FIGURA A.3: Inferencia de Faster R-CNN

FIGURA A.4: Inferencia de DETR²

²Argentina vs Francia, Mundial Qatar 2022. Frames extraídos de <https://www.youtube.com/watch?v=wQ6sEkorkfs&t=34s>

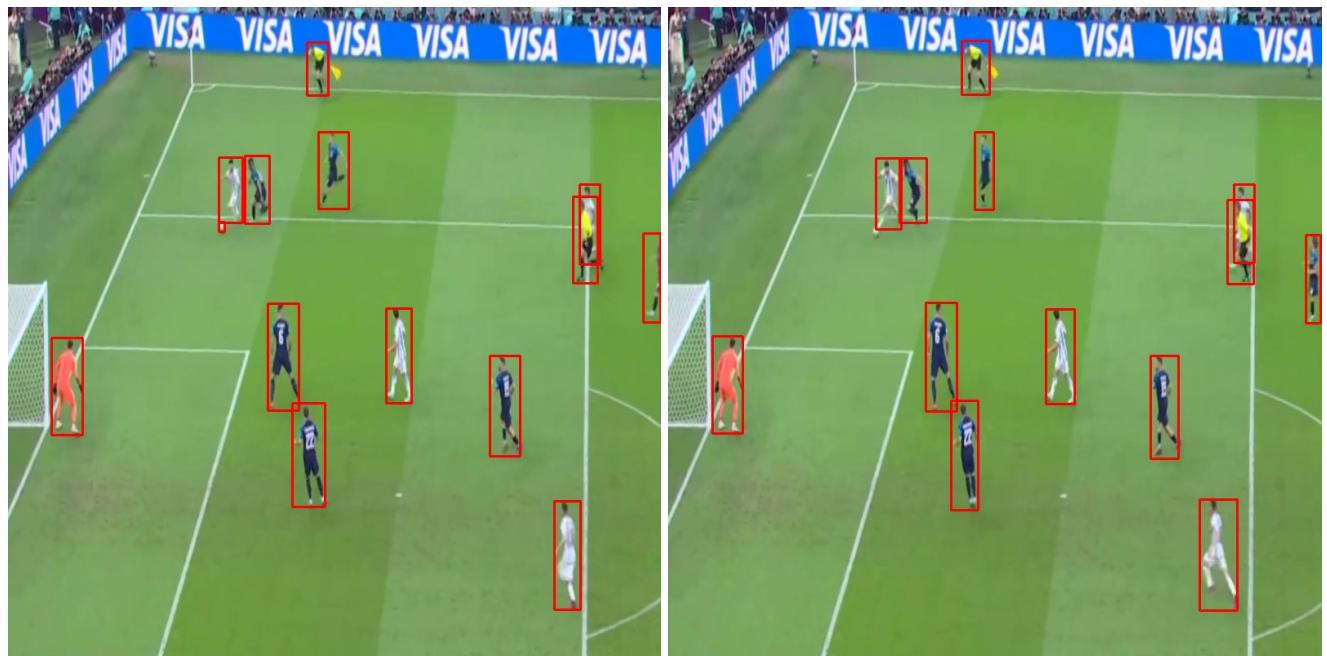
²Brazil vs Serbia, Mundial Qatar 2022. Frames extraídos de <https://www.youtube.com/watch?v=gzsAKz5GBnQ>

A.0.3. Secuencia de frames de la inferencia de Faster R-CNN



(A) Frame 1

(B) Frame 2



(C) Frame 3

(D) Frame 4

FIGURA A.5: Inferencia del Argentina vs Croacia, Mundial Qatar 2022.³

³Argentina vs Croacia , Mundial Qatar 2022. Frames extraídos de: <https://www.youtube.com/watch?v=Y1FBKUNAxwY>

A.0.4. Secuencia de frames de la inferencia de DETR



FIGURA A.6: Inferencia del Argentina vs Croacia, Mundial Qatar 2022.

A.0.5. Detecciones erróneas de Faster R-CNN

FIGURA A.7: Detecciones de Faster R-CNN

A.0.6. Detecciones erróneas de DETR

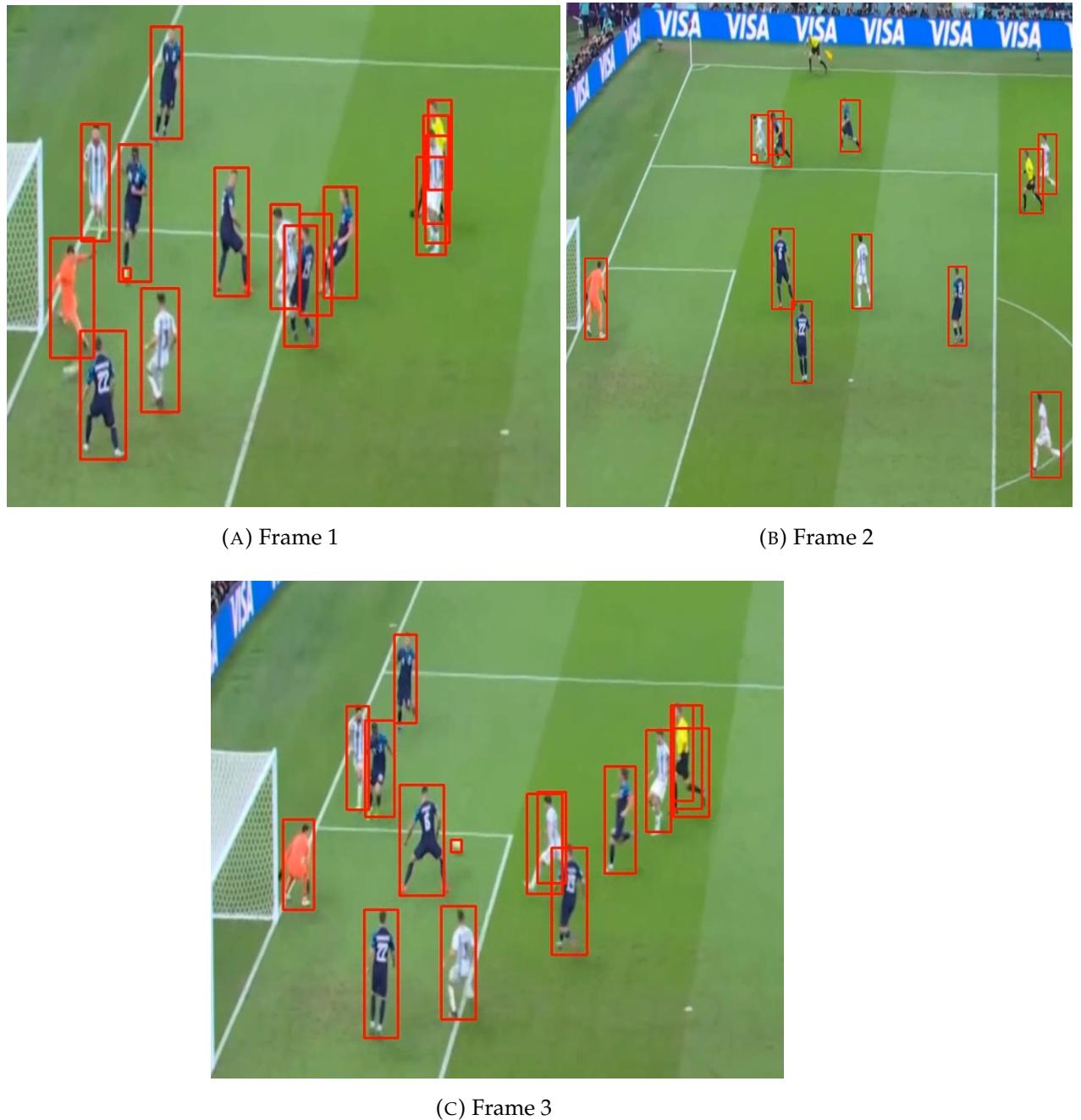


FIGURA A.8: Detecciones de DETR