UNIVERSIDAD NEBRIJA

Modelo de Computer Vision para la Detección de Eventos en Fútbol

UNIVERSIDAD NEBRIJA GRADO EN

INGENIERÍA INFORMÁTICA

MEMORIA TRABAJO EVALUACIÓN DE CAPACIDADES EN LA EMPRESA

Luis Ruiz Moreno

16 de octubre de 2024

UNIVERSIDAD NEBRIJA

Modelo de Computer Vision para la Detección de Eventos en Fútbol

UNIVERSIDAD NEBRIJA GRADO EN

INGENIERÍA INFORMÁTICA

MEMORIA TRABAJO EVALUACIÓN DE CAPACIDADES EN LA EMPRESA

Luis Ruiz Morneo

16 de octubre de 2024

Adolfo Abalo Cascallar)

Dn Luis Ruiz Moreno autoriza a que el presente trabajo se guarde y custodie en los repositorios de la Universidad Nebrija y además SI autoriza a su disposición en abierto.

Índice

Glosario de términos

[En esta sección se deben definir aquellos términos y acrónimos más relevantes que hayan sido incluidos en la presente memoria.]

→ **Término:** Definición

Índice de tablas

No table of figures entries found.

1. Resumen

Esta memoria recoge el desarrollo de mis prácticas curriculares en NTT Data y la creación de un modelo de computer vision para la detección y análisis de eventos en partidos de fútbol. En ella se detalla el recorrido que me llevó a acceder a esta experiencia profesional, los objetivos planteados y el crecimiento personal y técnico que he experimentado desde el inicio de mi contrato. Asimismo, se documenta el proceso de diseño, implementación y validación del modelo de visión por computadora, que busca identificar elementos clave en el terreno de juego para generar información relevante.

2. Prácticas

2.1. Antecedentes

Este apartado presenta de forma resumida los antecedentes que han guiado la elección de mis prácticas. Se abordan, en primer lugar, los procesos de selección que he vivido, destacando lecciones aprendidas en las entrevistas más significativas. Además, doy una visión general de NTT Data, la empresa donde haré las prácticas, luego detallo la formación académica y complementaria que me han ayudado a conseguir estas prácticas. Finalmente, se exponen los motivos que me llevaron a seleccionar esta empresa.

2.1.1. ¿Qué entrevistas he hecho?

Entre todas las vacantes que solicité —algunas aceptadas, muchas rechazadas—, destacaron cuatro procesos de selección que han sido significativos, ya sea para mi desarrollo profesional o para ir mejorando durante mi carrera en la búsqueda de las mejores prácticas posibles.

Por ser mi primer proceso y, por lo tanto, mi primera entrevista que hacía, no solo a la hora de encontrar prácticas curriculares, sino la primera que hacía en general, destaco mi proceso con Accenture. Tuve una llamada con ellos por teléfono para que me contaran más sobre el proceso y para citarme a lo que sería mi primera entrevista. Este proceso se parece, por cómo fueron sucediendo las cosas, al proceso que hice con BBVA. Son procesos en los que me encontré muy, muy cómodo en las entrevistas, pero que me rechazaron porque me requerían un nivel de inglés que no tenía. Estos dos procesos los destaco porque creo que el haber tenido un muy buen inglés me hubiera abierto muchísimas más puertas, y aprenderlo se ha convertido en una de mis prioridades durante este año. Para los procesos siguientes, me dediqué a preparar las preguntas más comunes en inglés. Esto se debió a que muchas vacantes utilizaban el inglés como filtro inicial, aunque el puesto en sí no necesariamente requiriera su uso diario.

Un proceso que destacó por la motivación, ya que era sobre una vacante la cual me suponía un reto al ser de ciberseguridad, fue con KPMG. Durante la carrera, no hemos profundizado en

ciberseguridad, pero a la vez veo que es una de las ramas más seguras a futuro que hay en el mundo de la informática y por ello no he dejado de aplicar a este tipo de procesos.

La vacante era en un proyecto llamado CiberLab, enfocado en la formación de ciberseguridad para roles de "blue team" o "red team", es decir, orientado a formarte como atacante o como defensor para poner a prueba los sistemas o protegerlos. A pesar de mi interés por la vacante, tuve que rechazar una segunda entrevista presencial porque acepté las condiciones de la empresa en la que estoy actualmente, y por el número de horas no podría no aceptar las condiciones y que posteriormente me rechazaran en este proceso.

Por último, destaco el proceso con NTT Data, por ser la compañía en la que finalmente estoy haciendo mis prácticas. No fue la entrevista más exigente, pero como la mayoría de los procesos que he hecho, seguía un patrón similar: primero hacían una llamada y luego te citaban para una entrevista. Dependiendo del proceso, al final te notificaban por correo o había una entrevista más con alguien que probablemente sería una persona de tu futuro equipo o cercana a él. Como en las demás entrevistas, esta siguió el mismo guion, repitiéndose las mismas preguntas: "¿Dónde te ves de aquí a 5 años?", "Háblame de ti", "¿Cuáles son tus principales fortalezas y debilidades?"

2.1.2. Datos de la empresa

NTT Data España es una filial de la empresa matriz japonesa NTT Data Corporation. En España se encarga de consultoría tecnológica y desarrollo de soluciones digitales, siendo una de las empresas más grandes en ello.

En España, hay más de 20 mil empleados en diferentes áreas, pero específicamente en el departamento de Data & Analytics (DNA) en el cual estoy somos hoy en día alrededor de 550 personas.

Dentro de este departamento, hay diferentes grupos que trabajan en diferentes sectores como puede ser por ejemplo la banca o el sector telco. En mi grupo en concreto estamos trabajando para un proyecto llamado UGG. Unsere Grüne Glasfaser (UGG) es una empresa alemana que se encarga de llevar fibra óptica a las zonas más rurales. Es una empresa conjunta 50/50 donde Telefónica posee el 50% y el 50% restante de Alllianz.

Mi grupo este compuesto por 9 personas, incluyéndome a mí, y nuestra tarea principal dentro del proyecto de UGG es la gestión integral del almacenamiento de datos. Nos encargamos de todas las etapas, desde los procesos de ETL (Extracción, Transformación y Carga), pasando por el almacenamiento de datos analíticos, la creación de modelos de datos, hasta la visualización de estos.

2.1.3. Información Previa (asignaturas, cursos previos, conocimiento previo, etc.)

Aparte de lo aprendido en la universidad, he realizado algunos unos cursos para complementar mi formación. Primero, gracias a la universidad y al Santander, tuve acceso a un curso llamado

Santander Nebrija Búsqueda de Empleo. Este curso ofrece videos para mejorar tu CV, a encontrar las mejores vacantes, preparar una entrevista, escribir una buena carta de presentación, …

Otra cosa que detecte mientras solicitaba vacantes de análisis de datos es que todas pedían conocer una herramienta de visualización como Tableau o Power Bi. Como en la carrera no dimos ninguna de estas herramientas, hice un curso de Power Bi que me enseñaba los fundamentos. Justo en las prácticas que he comenzado utilizamos esta herramienta así que me ha venido fantástico.

También llevaba mucho tiempo queriéndome sacar una certificación de algún servicio de Cloud (Azure, AWS, Google Cloud, …). Empecé con la certificación de Google Cloud Associate, la cual me ha servido mucho para entender todos los productos que existen en la nube. Aunque esta preparación la he parado, ya que he preferido enfocarme en las certificaciones de Microsoft Azure, dado que el proyecto en el que estoy haciendo las prácticas trabaja con Azure. Es cierto que no terminé la certificación de Google, pero me ha sido útil para empezar a prepararme las de Azure.

2.1.4. Motivos para elegir estas prácticas

Primero antes de iniciarme a buscar vacantes definí varios requisitos, para hacer más cómoda mi búsqueda y encontrar el mejor sitio posible:

- **Empresas grandes:** Consideraba que trabajar con una organización grande me iba a ofrecer grandes ventajas. La primera es que tengo más opciones de trabajar en proyectos grandes que presentan retos complejos, la posibilidad de trabajar con mucha gente diferente y en distintos sectores. Además, la mayoría de las empresas a partir de un tamaño, la remuneración ascendía y al ser unas prácticas a tiempo completo y de larga duración, era una motivación mayor.
- **Opiniones amigos y familiares:** Muchos compañeros de la universidad los cuales conozco habían hecho las prácticas el año pasado por lo que recurrí a ellos, aparte cada vez que avanzaba alguna fase del proceso con una empresa que conocían amigos o familiares, les preguntaba sobre ella. Aunque la opinión de empresas grandes no es tan determinante porque la experiencia de uno no representa la empresa, pero si me ayudaba a conocer la filosofía de esta.
- **Tipo de vacante:** Me enfoqué en tres áreas específicas que me resultan particularmente interesantes:
    - Desarrollo Backend
    - Análisis de datos
    - Ciberseguridad

En este caso NTT Data cumplía todos los requisitos: tenía unas buenas condiciones de remuneración, es una empresa grande que me permite el trabajar en diferentes proyectos en

diferentes ámbitos. Las opiniones de amigos y antiguos compañeros que están en o que han estado eran positivas, y el tipo de vacante se ajustaba a mis ambiciones.

2.2. Objetivos de las prácticas

Por lo que está definido en el contrato de prácticas y por lo que he visto en mis primeras semanas, el objetivo es que adquiera una experiencia completa en el manejo de datos a través de todas las etapas, desde la ingesta hasta la visualización de indicadores clave (KPIs). Estas etapas incluyen el procesamiento, la transformación y el modelado de datos para extraer información valiosa que luego pueda ser utilizada por el equipo de UGG para la toma de decisiones.

Durante el proyecto, he trabajado con los productos de Microsoft relacionados con la arquitectura Cloud de Azure y la capa de visualización con Power BI. Como objetivo personal, me he propuesto trabajar en paralelo para certificarme a través de Microsoft Learn en los productos de Azure que vaya utilizando en el proyecto.

Como parte del proyecto, también tiene peso el documentar todas las nuevas funcionalidades y componentes de la base de datos.

2.3. Onboarding

El primer día me citaron personalmente con todos los "Student" (es como llaman al cargo de becario en NTT) que entraban esa semana. Nos explicaron detalladamente cuáles eran las condiciones y los beneficios de los que podíamos aprovecharnos mientras hacíamos nuestras prácticas (formación, vacaciones, flexibilidad de horarios, etcétera). También, como día de bienvenida, conocí a mi grupo y me explicaron sobre el proyecto en el que iba a trabajar. En cuanto a la forma de trabajar, hay que destacar que la manera de organizar las tareas es a través de una reunión diaria y utilizamos una metodología de trabajo ágil.

En mi grupo tenemos diferentes funciones, desde el proceso de ETL, visualización, etc. Esto hace que, aunque sea un grupo multidisciplinar, todos saben de todas las tareas: desde consultas SQL, creación de dashboards, generación de tablas y vistas en la base de datos, canalizaciones para modelado de datos, etcétera. Esto ha hecho que, desde el primer día, lo que he hecho es asistir a muchas reuniones con mis compañeros para ver cómo resuelven las tareas, y en tan solo dos semanas he visto el progreso: de cómo en los diez primeros días solo veía y, poco a poco, dejo de observar y ya voy haciendo.

Durante estas primeras semanas, paralelamente también he estado formándome en la plataforma de Microsoft Learn para conocer bien las herramientas con las que trabajamos, como todo el módulo de Azure Synapse Analytics, que reúne todas las herramientas para trabajar sobre un almacenamiento de datos empresarial.

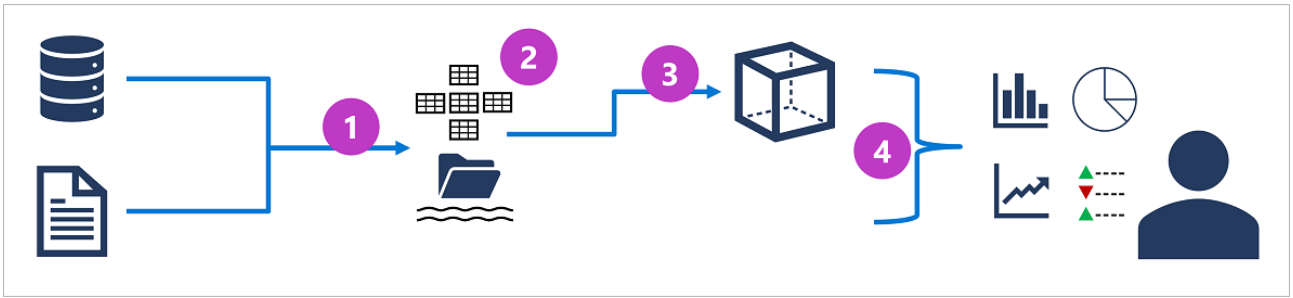
Aparte, justo ha coincidido que ha habido una reunión trimestral de todo el departamento de Data & Analytics (DNA), lo cual me ha ayudado a conocer qué otros proyectos había, conocer a gente

que trabaja en mí mismo departamento, ver los objetivos que tienen para los "Student" y discutir temas como la entrada de la IA, cómo va a cambiar el perfil de los empleados o la forma de trabajar en los proyectos.

2.4. Funciones y tareas en las prácticas. (evidencias)

En el contrato de prácticas se resumía mi función como:

Participar en todo el ciclo del tratamiento de datos en la implementación de casos de uso. Esto incluye la ingesta en el datalake (1), el procesamiento, transformación y modelado para el cálculo de KPI's (2 y 3) y desarrollo de la capa de visualización (4). Este proceso se indicaba ya que se realizaría en el prisma de Azure y que la capa de visualización se trataría en Power Bi.



Ciertamente como señalaba el convenio, he participado en todo el ciclo del dato, como puede ser solucionar algún error en el pipeline, añadir nuevos datos y crear diferentes tablas. También he formado parte, aunque en menor medida de la fase de visualización en Power Bi. Como regla general siempre que se ha creado algo nuevo o se modificaba había que documentarlo, la documentación de Business Rules y Datamarts han sido parte de la experiencia adquirida en las prácticas.

Otra función que he tenido es apoyar a un compañero a gestionar aquellas incidencias que iban surgiendo, como fallos en pipelines, fallos en algunos datos, duplicados, …

2.5. Relaciones de problemas planteados y procedimientos para su resolución.

Voy a destacar dos problemas o tareas que se me han planteado durante esta primera fase de prácticas y cuál ha sido su resolución.

2.5.1. Business Rule para Action Dashboard

El Action Dashboard es uno de los desarrollos de los que mi equipo a trabajado durante finales de diciembre y principios de enero. El action dashboard tiene como propósito el mostrar en un panel los números cuantitativos de cuantas direcciones necesitan de atención según diferentes problemas, aquellas que han fallado la sincronización de la ONT, las que falta el cableado, si han tenido algún problema con algún elemento de la red, si necesitan algún tipo de seguimiento, … Una manera de tener en un panel centralizado el número de cuantas work orders necesitan que se accione sobre ellos.

Uno de los indicadores a representar en el Action Dashboard necesita de una lógica de negocio por detrás. La Business Rule lo que hace es recoger las órdenes de trabajo en las que se ha producido un fallo de sincronización ONT, detallando el momento del fallo, a la empresa que gestionó inicialmente el fallo, también determina que si ha superado el work order ha superado el LSA, la marcará como candidata a cambiar de empresa.

La Business Rule ya estaba prácticamente ejecutada, pero había que añadirle unos filtros para que solo estuvieran aquellas que provenían de una lista de municipios específicos y excluir otras relacionadas con unos DP's en concreto.

La lista de DP's y Municipios, los cargué en dos tablas que como referencia para hacer los cruces con la business rule principal. Estas tablas las cree a través de un pipeline en Azure Data Factory que me permite copiar los datos de un CSV a una tabla permitiéndome un control total sobre el tipado de los campos de una manera visual.

La solución consiste en primero seleccionar las ordenes de trabajo de la tabla maestra de todas las ordenes de trabajo que la llamo a y la cruzo con un link que cruza las orders con los DP. Luego hacemos el cruce con el enlace entre DP's y Municipios para determinar el municipio asociado. Finalmente, se filtran los datos para conservar únicamente aquellos que coinciden con la tabla referencia de municipios y, al mismo tiempo, se descartan los registros que coinciden con la anterior tabla de referencia creada con la lista de DP's a excluir.

```
,dp_municipio_filtro AS (
    SELECT DISTINCT a.numero
    FROM a
    INNER JOIN [WO_L_DP] dp
        ON a.numero = dp.numero
    INNER JOIN [DP_L_MUNICIPIOS] ldp
        ON ldp.[NOMBRE] = dp.nombre
    INNER JOIN [MUNICIPIOS] mcc
        ON mcc.[MUNICIPIO_ID] = ldp.[MUNICIPIO_ID]
    LEFT JOIN [DP_EXCEPCION] dpExc
        ON dpExc.[NOMBRE] = dp.[NOMBRE]
    WHERE dpExc.[NOMBRE] IS NULL
)
```

2.5.2. Monitoring Concept

Muchas de las funcionalidades que llevamos a cabo tienen como fase final la visualización de los datos en un dashboard en PowerBi. La gran mayoría de los datos tienen como objetivo el monitorear diferentes métricas, por lo que una de las claves es que los datos del dashboard estén actualizados.

El cada vez que entras a un dashboard actualizar los datos puede ser un proceso lento e innecesario si por ejemplo alguien ha accedido a ese dashboard hace poco y ya ha actualizado los datos. Por ello se planteó agregar elemento visual que indique si los datos que están mostrándose están actualizados o no.

A través de PowerQuery calcularemos métricas para que vaya modificándose un icono en el dashboard. Dependiendo del color indicará una cosa:

- **ROJO**: Indica que los datos del dashboard no están actualizados.
- **AMBAR**: Indica que los datos del dashboard están actualizados, pero que lo que no está actualizado es la propia tabla de la base de datos.
- **VERDE**: Indica que los datos del dashboard están actualizados.

También hay un formato en el que solo hay rojo y verde que indicará únicamente si están o no están actualizados.

2.6. Aprendizajes y desarrollo profesional (habilidades adquiridas)

He aprendido mucho sobre el entorno de Azure, he visto realmente las posibilidades reales actuales que te brinda los proveedores de servicios Cloud. Mas específicamente aquellos servicios para la ingesta y transformación de los datos. Aparte e afianzado mi habilidad con SQL la cual llevaba tiempo sin practicar.

A nivel de soft skills, apartando el aprendizaje más técnico, he aprendido a nivel de trabajar en equipo, como interactuar con diferentes perfiles de personas a las que has conocido desde cero y que me van a acompañar durante todos los días, como comunicarme, como organizarme con ellos y adaptarme a su forma de trabajo, han sido las aptitudes que he ido desarrollando durante este tiempo.

El tener en paralelo las prácticas laborales, el desarrollo del TFG, el desarrollo de la memoria con su proyecto relacionado y mi ámbito personal ha sido un desafío a la hora de planificación y gestión del tiempo aparte de aprender a comprometerme y saber decir que no. Creo que es de las

cualidades que más me van a ayudar a nivel general durante mi vida y que seguro que también me queda mucho por aprender.

Para el segundo periodo de prácticas me propongo en certificar oficialmente esas aptitudes técnicas de Microsoft Azure a través de Microsoft Learn y más en el ámbito personal pues mejorar mi inglés porque me va a ser imprescindible en un futuro tener un buen nivel si voy a querer acceder a mejores puestos.

2.7. Metodologías utilizadas

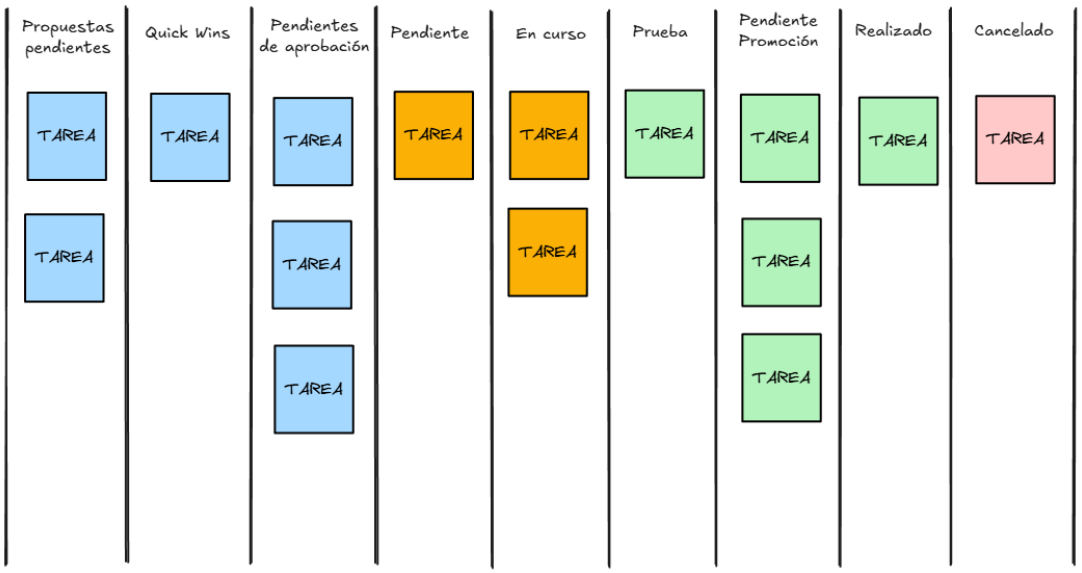
En la metodología de trabajo utilizada veo similitudes con diferentes metodologías ágiles, siendo un enfoque bastante flexible y de flujo continuo.

Entro sobre las 8:00 – 8:30, y es a las 9:00 cuando tenemos nuestra reunión diaria, a partir de esta reunión que tenemos todo el equipo planeamos todo el día. Hacemos seguimiento de las tareas en curso, la situación de cada uno con sus tareas, si hace falta dar por finalizada alguna y el repartir nuevas tareas. El seguimiento de las tareas lo hacemos a través de Microsoft Planner.

Después de esta reunión y de estar repartidas las tareas, suelo agendar con los compañeros las reuniones, en caso de ser necesarias para alguna tarea en concreto. Tenemos una metodología flexible y, por ejemplo, a veces salen incidencias a lo largo del día que tenemos que abordar.

Veo similitudes con una metodología Scrum con las reuniones diarias, podría ser similar al evento de “Daily Standups”, en esta reunión organizamos las tareas, pero lo solemos hacer a nivel diario no hay definidos sprints fijos como en una metodología scrum.

Una cosa esencial de las metodologías ágiles es la utilización de herramientas de seguimiento, y es lo que hacemos con Microsoft Planner a nivel de equipo y Azure DevOps a nivel de proyecto. El uso de un tablero es propio de un modelo de trabajo Kanban, donde hay una pizarra/panel de tareas donde van pasando de estado, es un flujo de trabajo continuo y flexible priorizando tareas según urgencias y necesidades.



2.8. Herramientas utilizadas en las prácticas (evidencias).

He utilizado diversas herramientas, todas del entorno o marco de trabajo que proporciona Microsoft:

- Microsoft Teams como herramienta de comunicación, que permite integrar en ella aplicaciones como Microsoft Planner, lo utilizamos como tablón de tareas para planificar, o de almacenamiento como OneDrive.
- Para realizar las consultas SQL al servidor y editar alguna regla de negocio o vista utilizo Microsoft SQL Server Managment Studio. A veces utilizo VSCode ya que estoy más acostumbrado al entorno.
- Los servicios en la nube contratados, como las bases de datos, están alojadas en Azure y es gracias a su servicio de Azure Synapse donde puedo acceder a todos los procesos internos de la BDD.
- Otra herramienta que utiliza mi equipo, para la gestión del proyecto, es Azure Devops. En mi caso solo la utilizo cuando necesito información más detallada de la tarea
- Para la parte de visualización utilizo Microsoft Power Bi, en general para crear o modificar dashboards para el cliente.
- Como herramientas ofimáticas, a diario, utilizo Microsoft Word y Excel. Estas herramientas me sirven sobre todo para documentar.

Aparte a través de navegador siempre accedo al Service Now de UGG, que me permite estar al tanto de si surge alguna incidencia.

2.9. Logros, resultados y discusión.

Pues como he ido redactando en los apartados anteriores ha habido un desarrollo técnico, pero también un desarrollo profesional y personal.

El mayor logro en la parte técnica ha sido familiarizarme con los servicios de Azure, más en concreto con los servicios que hacen el proceso de ETL como es Azure Data Factory. A nivel de habilidades, el trabajar en un entorno de trabajo ágil creo que me va a aportar mucho en un futuro porque es algo muy común actualmente en que los equipos se organicen dentro de este marco de trabajo.

Como comentaba en el apartado de *Aprendizajes y desarrollo profesional* en el ámbito personal, tener en paralelo las prácticas laborales, el desarrollo del TFG, el desarrollo de esta memoria ha sido un desafío a la hora de planificación y gestión del tiempo. Creo que es el mayor logro y sobre todo el que más me va a ayudar a todos los ámbitos de mi vida.

Al final para lo que me ha servido este primer periodo de prácticas y creo que es el objetivo principal de ellas es el poder validar que puedes pertenecer a un equipo y que eres válido para poder aplicar a un trabajo de estas características.

3. Desarrollo

3.1. Introducción

Consiste en desarrollar un modelo de computer vision (vision por computadora) para la detección de eventos en partidos de fútbol. Los elementos detectados incluyen jugadores, el balón, árbitro y sus movimientos dentro del campo. Esta detección se plantea como base para llevar a cabo análisis automatizados que generen información relevante y útil.

El tema resulta relevante debido al creciente papel de la tecnología en el deporte. La combinación de inteligencia artificial y computer vision tiene el potencial de transformar completamente el análisis deportivo. Además, aprender sobre esta tecnología crea oportunidades en otros sectores como la seguridad, la industria, el entretenimiento o la agricultura, donde estas tecnologías tienen aplicaciones significativas. En el caso específico del fútbol, el uso de datos para optimizar estrategias, evaluar el rendimiento y tomar mejores decisiones se ha consolidado como una práctica clave tanto en el presente como en el futuro.

Actualmente, ya existen productos en el mercado que abordan necesidades similares. Por ejemplo, en la detección de objetos en imágenes, destacan modelos de propósito general como YOLO (You Only Look Once) y bibliotecas como OpenCV (Open Computer Vision). Además, las principales tecnológicas han desarrollado servicios en la nube especializados en visión por computadora, como Amazon Rekognition, Azure Computer Vision o Google Cloud Vision AI, que facilitan soluciones avanzadas de manera escalable y accesible.

En el ámbito específico del fútbol, también se han creado herramientas y modelos diseñados para este propósito en concreto. Empresas como Driblab y Hawk-Eye han desarrollado sistemas que colaboran no solo con equipos de élite, sino también con organizaciones como la FIFA (Federación Internacional de Fútbol Asociación). Estas tecnologías están diseñadas para optimizar el análisis del rendimiento de los jugadores y mejorar las decisiones arbitrales.

Desarrollar un modelo propio para la detección y el análisis de eventos en el fútbol no solo es relevante desde una perspectiva tecnológica, sino que también es una manera para mí, de ganar conocimiento sobre una tecnología en auge.

3.1.1. Motivación

Durante este año vi como una oportunidad en el TFG y en esta memoria para poder aprender nuevas cosas y abrirme a nuevos campos. Ya que son dos proyectos que requieren de tiempo quisiera que me aportaron algo nuevo o que me abrieran nuevos horizontes. En el caso del TFG es algo sobre bioinformática y en este proyecto me propuse aprender alguna nueva tecnología o técnica para

Entonces combiné, una tecnología que está en auge y extendida a todos los sectores y el futbol que es un tema de interés. Elegí el futbol porque necesita ser de un tema que me gustará desarrollar y aprender, una manera de aprender muchísimo contenido de una forma que me interesa.

3.1.2. Análisis de mercado y necesidades

Actualmente se utiliza en todas las facetas del fútbol: desde ayuda para el arbitraje y asistencia para el cuerpo técnico de los equipos, hasta el enriquecimiento de la experiencia del espectador con datos en tiempo real, como ocurre en la Liga EA Sports, donde se calculan las probabilidades de gol en directo según las circunstancias del momento.

Sin embargo, todavía hay mucho camino por recorrer, ya que esta tecnología no está extendida a todas las ligas ni a todos los equipos, limitándose únicamente a aquellos que son profesionales.

3.1.3. Objetivos

La realización de este trabajo tiene como principal propósito desarrollar un modelo de visión por computadora capaz de detectar y analizar eventos en partidos de fútbol, para ofrecer futuras soluciones innovadoras. El objetivo específico se detalla a continuación:

- Detección de Jugadores, Árbitros y Balón

Diseñar y entrenar un modelo que identifique con precisión los elementos clave de un partido: los jugadores, el árbitro y el balón. Este paso es esencial para comprender y analizar los eventos que tienen lugar durante el juego.

3.1.4. Requisitos técnicos

3.1.5. Análisis de mercado

3.2. Marco técnico

3.3. Equipo de trabajo y metodología

3.4. Proyecto

Aquí desarrollaré como ha sido el proceso de todo el proyecto.

3.4.1. Modelo desde 0

Cuando hablo de crear un modelo desde cero, es el no utilizar ya un modelo pre entrenado. El objetivo es el entrenar una red neuronal para ver qué resultados podemos llevar teniendo una anotación manual y creando mi propio dataset. La creación de este modelo tiene como objetivo no tener un resultado idóneo si no crear un código que genere un modelo básico CNN y a ver como con nuestro pequeño dataset que resultado de detección podemos conseguir.

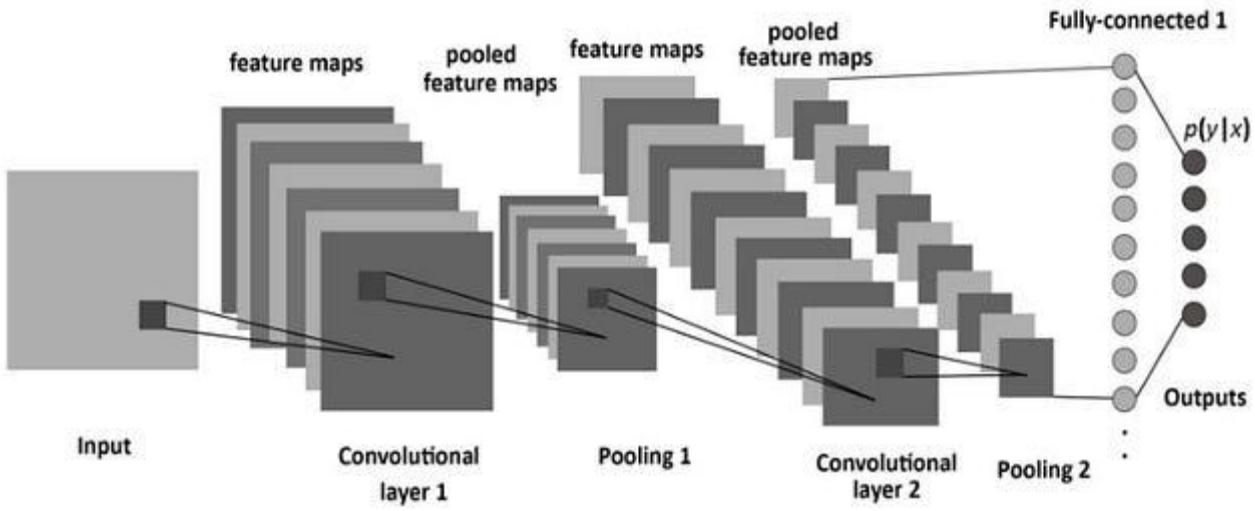
Toda la parte del conjunto de datos, profundizaré en la parte del modelo fine tuneado, en este apartado comentaré la arquitectura de un CNN simple y una prueba para ver qué resultados da.

La red neuronal convolucional encapsulada en el objeto SoccerObjectDetector del archivo model_create tiene como objetivo que, a partir de una imagen de entrada, predecir:

- **Las coordenadas de las cajas**, donde se encuentra l objeto detectado en la imagen
- **La clase**, es decir si la detección es un árbitro, un balón o un jugador. A estas clases le he puesto un máximo ya que debe haber máximo 1 árbitro, 1 balón y 22 jugadores.

La arquitectura del modelo la vamos a dividir en cuatro módulos.

1. **Módulo de entrada,** se encarga en recibir la imagen y la redimensiona a las medidas del modelo, en este casi 1024x1024.
2. **Etapa de capas convolucionales y de pooling,** una vez que la imagen está ajustada a 1024×10241024, se introduce en una serie de **capas convolucionales** (Conv2d) y **capas de pooling** (MaxPool2d). Esta etapa es esencial para **extraer patrones y rasgos característicos** de la imagen. Este módulo este compuesto por:
    a. Capas convolucionales, que cada una produce un conjunto de mapas de características, que reflejan la presencia de patrones en distintas regiones.
    b. ReLU, es una función de activación que transforma los valores negativos en cero y deja pasar sin cambios los valores positivos, permite que la red maneje mejor la información y detecte patrones de forma más clara.
    c. MaxPooling, tras la convolución y la activación, se aplica un pooling que reduce la dimensión espacial de los mapas de características. Esta etapa condensa la información más relevante.
3. **Etapa de capas completamente conectadas**, al terminar la etapa convolucional, la salida conserva mucha de la información de la imagen, pero en un formato ya reducido y codificado. Esta etapa tiene una fase de **aplanamiento** que junta en una única línea la información extraída de la imagen, luego una etapa de **FC1**, que aprende a combinar características, una **función de activación** como en la etapa convolucional y una fase ultima de **dropout** que hace que, durante el entrenamiento, de forma aleatoria se apagan la mitad de las neuronas para evitar que la máquina se "aprenda de memoria" las fotos de entrenamiento. Esta etapa consigue **combinar** toda la información y **tomar decisiones** para decidir las coordenadas de las cajas que mejor se ajusten a cada objeto.
4. **Módulo de salida,** esta etapa entrega dos tipos de salida, las cuatro coordenadas del objeto $(Xmin, Ymin, Xmax, Ymax)$ y la clasificación del objeto, es decir si es la bounding box es de un árbitro, un balón o un jugador.



Explicación del código

Para la creación de la CNN he utilizado la librería PyTorch, en su momento desarrollada por Meta y actualmente administrada por su propia fundación PyTorch Foundation. La CNN estará encapsulada en una clase llamada SoccerObjectDetector. Al crear el modelo se van inicializando todas esas capas (convoluciones, funciones de activación, pooling, ...) que examina la imagen para encontrar patrones y características.

```
class SoccerObjectDetector(nn.Module):
    def __init__(self, num_classes):
        super(SoccerObjectDetector, self).__init__()

        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, stride=1, padding=1)
        self.relu1 = nn.ReLU()
        self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv2 = nn.Conv2d(16, 32, kernel_size=3, stride=1, padding=1)
        self.relu2 = nn.ReLU()
        self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2)

        self.conv3 = nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1)
        self.relu3 = nn.ReLU()
        self.pool3 = nn.MaxPool2d(kernel_size=2, stride=2)
```

Después de que las capas convolucionales extraen las características, las capas completamente conectadas toman estas características y las usan para tomar decisiones. Las capas completamente conectadas toman estas características y las usan para tomar decisiones.

```
        self.fc1 = nn.Linear(64 * (IMAGE_SIZE[0] // 8) * (IMAGE_SIZE[1] // 8), 512)
        self.relu4 = nn.ReLU()
        self.dropout = nn.Dropout(0.5)

        # Capa de salida para las coordenadas
        # 4 valores por caja y máximo de 24 objetos por imagen
        self.fc_bbox = nn.Linear(512, 24 * 4)

        # Capa de salida para las clases (probabilidades)
        self.fc_class = nn.Linear(512, 24 * num_classes)

    def forward(self, x):
        x = self.pool1(self.relu1(self.conv1(x)))
        x = self.pool2(self.relu2(self.conv2(x)))
        x = self.pool3(self.relu3(self.conv3(x)))
```

```
        x = x.view(x.size(0), -1)  # Aplanar la salida de las capas convolucionales

        x = self.dropout(self.relu4(self.fc1(x)))

        bbox_output = self.fc_bbox(x)
        class_output = self.fc_class(x)

        return bbox_output, class_output
```

Este código muestra la arquitectura del modelo, pero para que este modelo aprenda debe pasar por un proceso de entrenamiento de esa máquina. Lo que hacemos con ese búcle es repetir el proceso de aprendizaje muchas veces, tantas como EPOCHS hayamos definido, en el modelo de fine tuning explicaré el significado de todos estos parámetros. El método .train() le indica al modelo que está en modo aprendizaje, este entrenamiento lo haremos por lotes, que habremos especificado su tamaño en BATCH_SIZE.

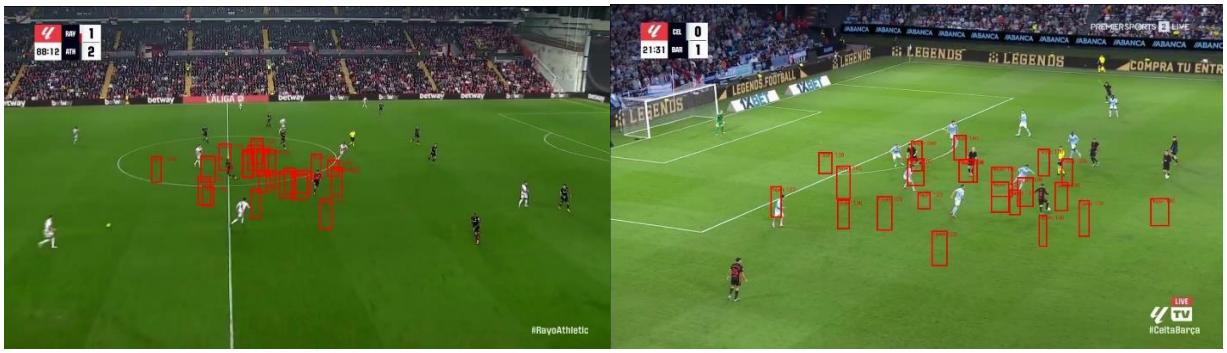
El método .to(DEVICE) nos permite en caso de tener GPU, trasladar las fotos y etiquetas ahí para que la gráfica pueda procesarlas rápidamente. Las tarjetas gráficas operan de manera matricial, y por eso son tan eficientes ya que una imagen es una matriz de pixeles.

```
for epoch in range(EPOCHS):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target_boxes, target_labels = data.to(DEVICE), target["boxes"].to(DEVICE),
target["labels"].to(DEVICE)
        target = {"boxes": target_boxes, "labels": target_labels}

        optimizer.zero_grad()
        bbox_output, class_output = model(data)
        bbox_loss, class_loss = calculate_loss(bbox_output, class_output, target)
        total_loss = bbox_loss + class_loss

        total_loss.backward()
        optimizer.step()

        if batch_idx % 10 == 0:
            print(f"Epoch: {epoch+1}/{EPOCHS}, Batch: {batch_idx}/{len(train_loader)}, "
                  f"Total Loss: {total_loss.item():.4f}, BBox Loss:
{bbox_loss.item():.4f}, "
                  f"Class Loss: {class_loss.item():.4f}")
```

En la línea bbox_output, class_output = model(data) es realmente donde el modelo está entrenando. Le damos un lote de datos y usando su capa anteriormente definida intenta predecir donde están las cajas (bbox_ouput) y que hay en cada caja (class_output).

Resultado del modelo

Por último, comentar la salida que como me esperaba, con el tamaño y calidad del dataset, no da buenos resultados. De ahí el potencial de hacer fine tuning de los modelos ya pre entrenados, ahora mismo es donde está el foco de la IA en utilizar esos modelos grandes que no puede entrenar una persona individual con su ordenador, como es mi caso, para poder tunearlo y que consiga un gran desempeño en una tarea específica. Por ejemplo, el modelo que he utilizado como base del modelo final esta entrenado con un dataset de entre 200 mil y 300 mil imágenes anotadas, además de una arquitectura muchísima más compleja.



Lo que parece estar ocurriendo es que el modelo ha detectado patrones en la distribución de las cajas, más que las formas concretas de los jugadores.

En ambas imágenes el modelo siempre marca cuatro cajas en línea en un lado del campo, lo que corresponde a la línea defensiva de un equipo. Luego, en la mayoría de los casos, hay una caja posicionada justo delante o detrás de esos cuatro defensas, haciendo referencia a la ubicación del delantero rival.

Por otro lado, el modelo muestra una distribución más irregular en la zona del centro del campo. Esto tiene sentido, ya que los mediocampistas suelen tener más libertad de movimiento y no siguen una estructura tan fija como la defensa o el ataque.

El modelo parece no haber identificado objetos en sí, sino más bien relaciones espaciales entre las cajas y sus posiciones relativas.

3.4.2. Modelo Fine Tuning

Como hemos visto en nuestro modelo desde cero, crear un modelo es algo costoso, ya que requiere primeramente de muchos datos, de mucho tiempo y aparte el entrenamiento es muy caro en cuanto a procesamiento.

Por eso es tan importante el concepto de fine-tuning, el concepto de consiste en ajustar un modelo ya pre entrenado que ha aprendido a reconocer diversidad de cosas y ajustarlo (de ahí la palabra tunear) a una tarea en concreto. Es útil a la hora de que cogemos un modelo que ya tiene mucho conocimiento y con menos datos y tiempo lo enfocamos a la tarea que queramos que cumpla.

En nuestro caso vamos a hacer fine-tuning sobre el modelo de YOLO (You Only Look Once), este modelo es revolucionario porque permite el detectar múltiples objetos en una sola pasada de la imagen. YOLO tiene una arquitectura modular que entonces permite que se adapte fácilmente a las necesidades requeridas, este es el porqué de su elección.

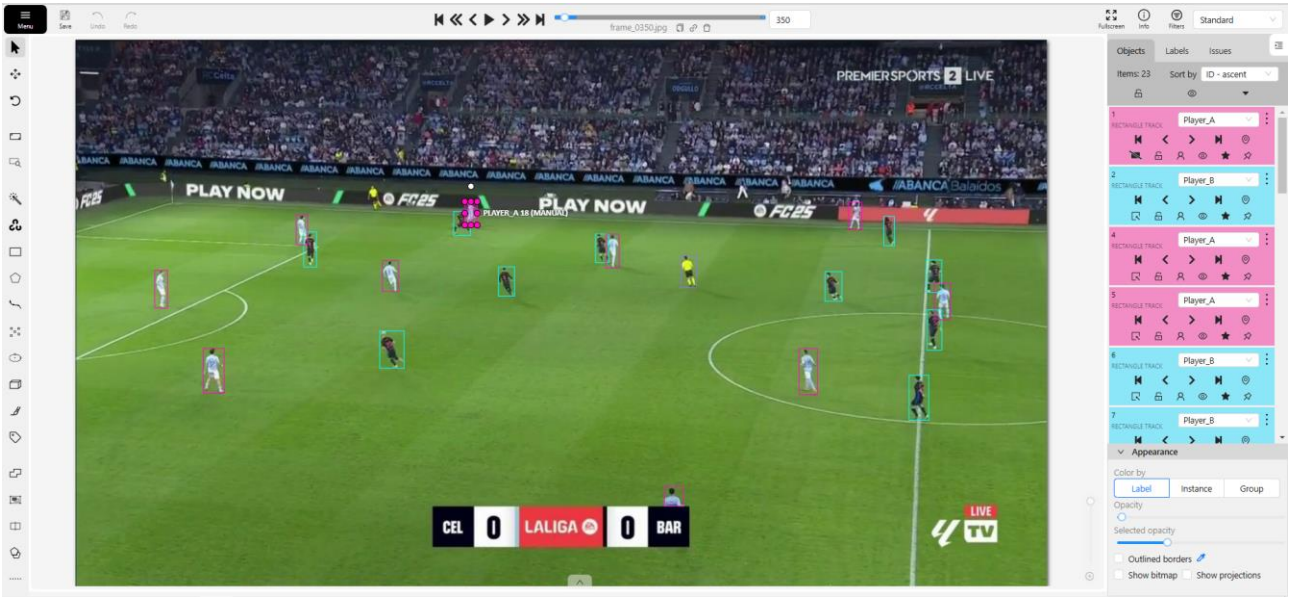
Conjunto de datos

En este punto sigo manteniendo el hacer todo lo más desde cero y lo más artesanal posible, para conocer cuál es el proceso completo en la creación de un modelo. El conjunto de datos original fueron videos de partidos en este caso de LaLiga EaSports, los videos me permiten descargar

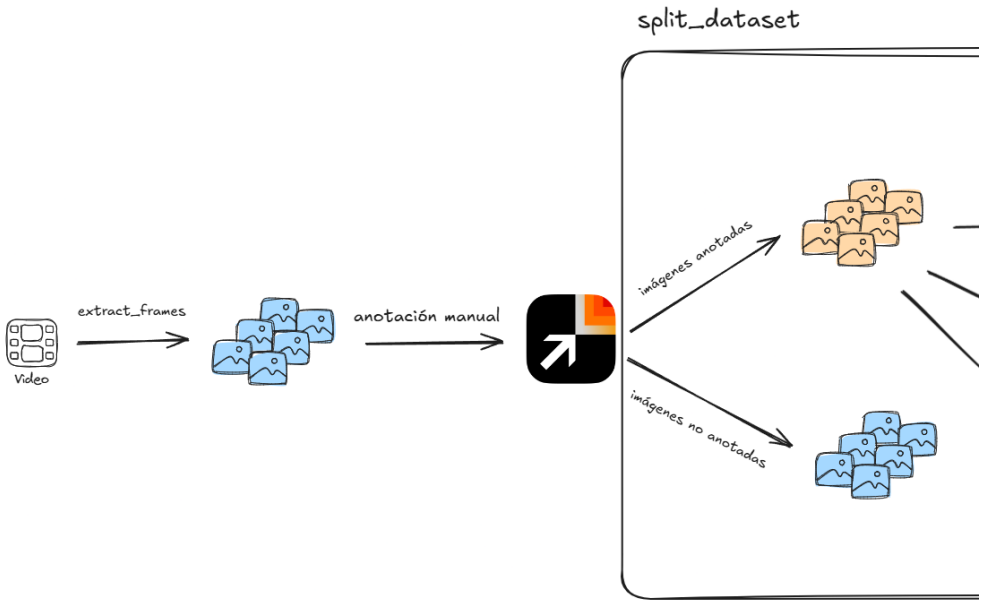
mucha información en bruto y luego extraer los frames. En mi caso para extraer los frames de los videos genere un método en el archivo utilities.py

```
def extract_frames(video_path, output_dir, frame_rate):
    """
    Extrae frames de un video y los guarda en una carpeta.

    Args:
        video_path (str): Ruta al archivo de video.
        output_dir (str): Carpeta donde se guardarán los frames.
        frame_rate (int): Cada cuántos frames extraer (pj:1 = todos los frames)
"""
    # Crear la carpeta de salida si no existe
    os.makedirs(output_dir, exist_ok=True)

    # Cargar el video
    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        print(f"Error al abrir el video: {video_path}")
        return

    frame_count = 0
    extracted_count = 0

    # Iterar sobre los frames del video
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret: # Si no hay frame para leer, salir del bucle
            break

        # Extraer los frames deseados según el frame_rate
        if frame_count % frame_rate == 0:
            frame_path = os.path.join(output_dir,
f"frame_{extracted_count:04d}.jpg")
            cv2.imwrite(frame_path, frame)
            extracted_count += 1

        frame_count += 1

    cap.release()
    print(f"Se han extraído {extracted_count} frames y guardado en '{out-
put_dir}'.")
```

Una vez que ya teníamos los frames era el momento de hacer las anotaciones de los frames, en esta etapa he utilizado software externo para hacer las anotaciones manualmente, en este caso he utilizado CVAT. Aunque hay software como roboflow que permite más opciones, por su fácil curva de aprendizaje y su extensa versión gratuita, me quede con la aplicación CVAT. Para la fase de anotación es importante primero dejar claras las clases que va a detectar el modelo. Como hemos comentado anteriormente, serán tres: árbitro, balón y jugador.



Una vez hechas las anotaciones, el software de CVAT nos permite exportarlas en el formato que nosotros prefiramos, en mi caso utilice el formato YOLO. Es el formato propio que utiliza el modelo, por lo que pensé que trabajar con esas anotaciones sería el formato más optimizado. El formato YOLO genera un archivo .txt para cada frame, es decir si tenemos el frame_0405.jpg, este formato generará un archivo frame_0405.txt con las anotaciones. Las anotaciones encapsulan la información de esta manera **<class_id> <x_center> <y_center> <width> <height>.** Las class_id es el identificado de la clase de nuestro objeto, en este caso como he dicho antes, nuestro modelo tendrá tres clases (Referee, Ball y Player) por lo que un class_id = 0 indicará que es una anotación de un árbitro, un class_id = 1 indicará que es una anotación del balón y, por último, un class_id = 2 refleja que es una anotación de un jugador. En cuanto a los demás parámetros (<x_center> <y_center> <width> <height>) indican la posición de la caja dentro de la imagen.

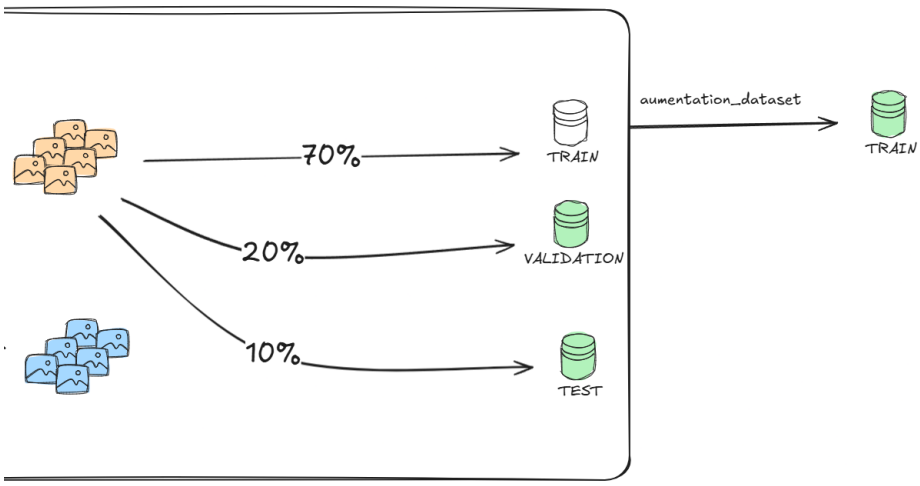


En mi caso anoté 609 imágenes, de las cuales 337 eran de un partido en concreto, las demás decidí anotar de varios partidos diferentes para tener diferentes ángulos de visión, diferentes

colores, iluminaciones, etc. Hice el mix de los frames de los frames con el método mix_frames en el archivo utilities.py.

Una vez que ya tengo las anotaciones y mi dataset bruto, tengo que hacer ciertas tareas de preprocesamiento para optimizar el posterior entrenamiento. Por ejemplo, decido eliminar las imágenes que no tienen anotaciones, dependiendo de cuál sea nuestro objetivo esto no siempre se debe hacer, pero no era mi prioridad el que sepa gestionar aquellos frames en los que no haya ningún objeto ya que en esta aplicación sería fácil detectar esos frames con tareas de post procesamiento.

Después de quedarme con las imágenes que contenían anotaciones (450 imágenes de las 609 totales), debíamos hacer dividir estas en tres apartados: train, validation y test. El dividir las imágenes en tres dataset es imprescindible porque cada uno cumple una tarea, uno para aprender, otro para evaluar el entrenamiento periódicamente y, por último, el test que evalúa la capacidad final del modelo. Los porcentajes de cada dataset siguen la regla del 70/20/10 que es un estándar en la tarea de “split dataset”.



Por último, he hecho dos tareas de procesamiento que manipulan directamente las imágenes. La primera consiste en redimensionar todas las imágenes a 640x640, con el objetivo de normalizar las imágenes. Es como una manera de adaptarlas a la arquitectura del modelo y encontrar un balance entre precisión y computo. La última tarea, antes de entrenar el modelo, es la de aumentar el dataset.

Por cada imagen del dataset de entrenamiento original, he generado 10 versiones modificadas de la misma. He aplicado las siguientes modificaciones al conjunto de entrenamiento:

- **Flip Horizontal:** Se voltea la imagen horizontalmente. Una modificación muy valiosa en el futbol ya que el campo es simétrico y las jugadas pueden ocurrir en los dos campos. No altera la esencia del partido de futbol esta modificación, por eso el que sea tan valiosa.

- **Grayscale:** Convierte las imágenes a escala de grises, esto obliga al modelo a no centrarse únicamente en la información que le da el color, centrándose más en por ejemplo la forma de los jugadores.
- **Shear** (±4° Horizontal, ±15° Vertical): Aplica deformaciones en la imagen, una modificación que viene bien para que el modelo este preparado para imágenes con una perspectiva diferente o para imágenes que están un poco distorsionadas.
- **Saturation:** La saturación permite el simular con las imágenes originales cambios de iluminación o efectos climáticos que se pueden presentar en un partido.

El aumento del dataste lo hacemos con el método augment_dataset, el augmentations_per_image permite cuantas imágenes modificadas quieres por imagen original.

```
def augment_dataset(input_dir, output_dir, augmentations_per_image):
    # Crear el directorio de salida si no existe
    os.makedirs(output_dir, exist_ok=True)

    # Definir la pipeline de transformaciones
    transform = A.Compose([
        A.HorizontalFlip(p=0.5),
        A.Affine(shear={'x': (-4, 4), 'y': (-15, 15)}, p=0.5),
        A.ToGray(p=0.15),
        A.ColorJitter(saturation=0.19, p=1),  # Saturation between -19% and +19%
    ], bbox_params=A.BboxParams(format='yolo', label_fields=['category_ids']))
```

Lo más importante de este método es el pipeline de transformaciones, este pipeline dice cuáles van a ser las transformaciones que va a hacer sobre las imágenes originales, modificando este pipeline podríamos añadir las transformaciones que queramos. La librería que utilizo para hacer las modificaciones es albumentations.

Este método va recorriendo un modelo un directorio de entrada va haciendo las modificaciones detalladas en el pipeline y posteriormente las guardas en un output de salida con las anotaciones. Voy a mostrar varios las modificaciones que hace este método para una misma imagen:

Para esta imagen genera las siguientes 10 modificaciones, viendo las modificaciones se ve la importancia de este paso, al entrenar varias situaciones diferentes con una misma imagen.

Esta parte proceso, en un modelo de nuestras características con un conjunto de datos tan pequeño es una de las partes más importantes ya que nos permite sacar el máximo jugo a nuestro dataset.

	IMÁGENES	IMÁGENES CON ANOTACIONES	IMÁGENES SIN ANOTACIONES		DATASET SIN AUMENTAR	DATASET AUMENTADO
TOTAL	609	450	159		349	3490
TRAIN					67	67
VALIDATION					34	34
TEST						

El dataset lo puedes encontrar en https://universe.roboflow.com/futbolcv/acuity-ctzp4/dataset/5 para poder descargarlo.

Procesamiento

En mi caso el procesamiento lo voy a llevar a cabo en Roboflow pero porque tiene una prueba gratuita que ofrece un gran procesamiento y me permite con el mismo número de datos generar un modelo muchísimo más preciso del que estaba consiguiendo con el procesamiento de mi ordenador.

Arquitectura de YOLO

La arquitectura de YOLOv11, en la que está basado mi modelo se basa en un diseño modular que ha evolucionado a partir de sus predecesores y que siempre que hay una nueva versión de YOLO su objetivo es lograr una mayore detección de objetos en tiempo real que sus versiones anteriores, para ser más eficiente y preciso. En lugar de construir un modelo desde cero, como hacíamos nosotros en la primera fase del proyecto, YOLOv11 se fundamenta en componentes predefinidos que han sido optimizados en versiones anteriores.

Cuando hablamos de que es una arquitectura modular, es que el modelo se organiza en tres bloques fundamentales:

1. **Backbone:**

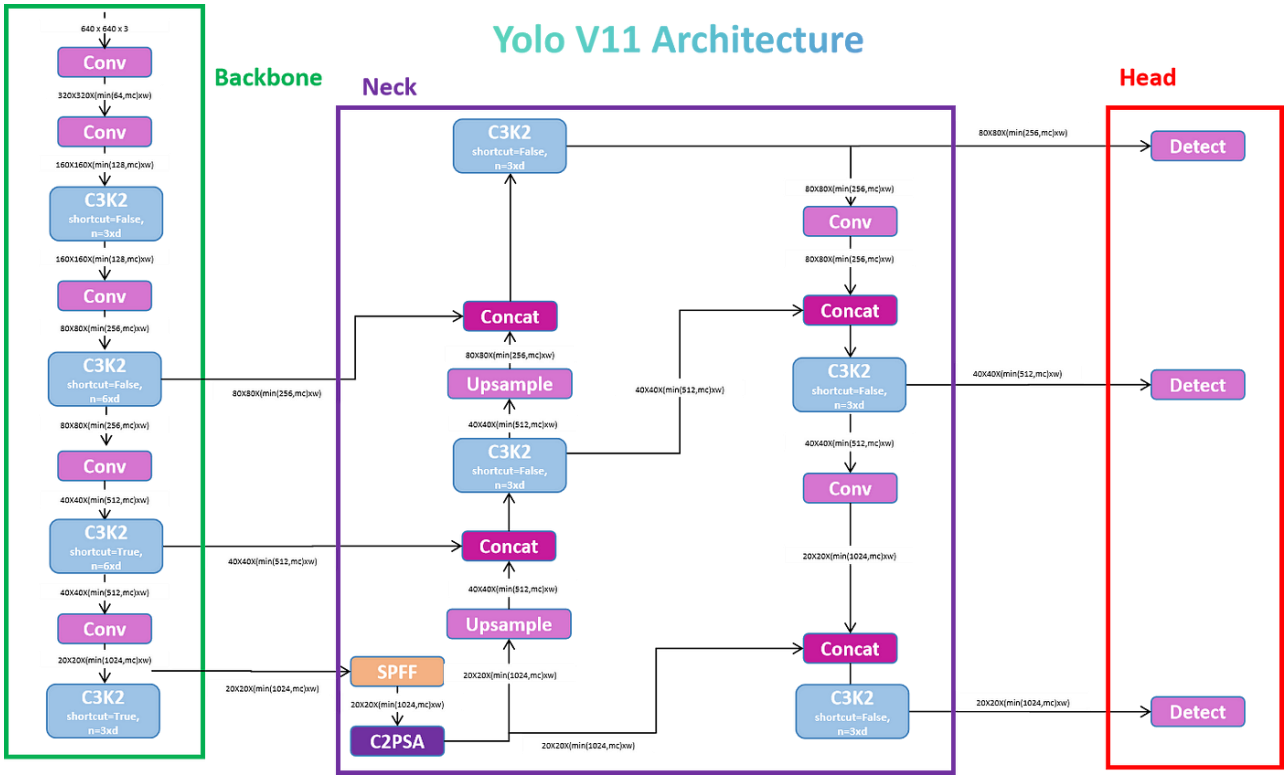
   Este bloque se encarga de extraer las características más relevantes de la imagen. En YOLOv11 se utiliza una serie de capas convolucionales que, además de reducir progresivamente las dimensiones espaciales, incrementan el número de canales para capturar distintos patrones. Un componente clave aquí es el bloque **C3k2**, mejora los anteriores ya que permite reducir la carga sin perder capacidad de extraer detalles.

2. **Neck:**

   Después de extraer las características, el neck se encarga de fusionarlas y combinarlas a diferentes escalas. Para ello, se integran módulos como el **SPPF**, que utiliza diversas operaciones de pooling para agrupar información espacial a distintas resoluciones, y el **C2PSA**, que añade mecanismos de atención para resaltar regiones de la imagen que son especialmente relevantes. Esta etapa es fundamental para que permita detectar objetos tanto grandes como pequeños.

3. **Head:**

   En la fase final, la head transforma la información integrada en predicciones concretas. Se utilizan varias instancias del bloque C3k2 junto con capas de **CBS**, que ayudan a estabilizar y refinar los datos antes de generar las salidas finales (en mi caso, serían las coordenadas de las cajas).



Una vez explicada la arquitectura del modelo base, tengo que explicar en qué modulo se realiza ese proceso de fine-tuning que voy a llevar a cabo. En nuestro caso, al aplicar fine-tuning sobre YOLOv11, aprovechamos la robusta arquitectura preentrenada para redirigir su capacidad hacia las clases que deseamos detectar.

El proceso se centra especialmente en las capas finales (la head) y, en ocasiones, en algunas partes del neck, dejando congeladas (sin actualizar) las capas del backbone que ya han aprendido representaciones generales. Así, el modelo mantiene el conocimiento adquirido en etapas previas y se "tunéa" para especializarse en, por ejemplo, detectar objetos particulares en un entorno concreto. Este proceso de llevarse a cabo el cambio en los módulos finales de la arquitectura hace que disminuyan los recursos computacionales para llegar a una alta precisión en una tarea específica.

Elección de Hiperparámetros

La selección y ajuste de hiperparámetros es una fase crítica en el entrenamiento del modelo. Los hiperparámetros son configuraciones externas al modelo que controlan directamente el proceso de aprendizaje. La correcta selección de los valores de los parámetros es esencial para lograr un modelo óptimo que generalice bien a datos no vistos y cumpla con los objetivos de detección planteados.

Voy a comentar el significado de alguno de estos parámetros que te deja

- **epochs:** Define el número de iteraciones completas sobre el conjunto de datos de entrenamiento. Un valor adecuado es crucial para permitir que el modelo aprenda patrones relevantes sin caer en sobreajuste o subajuste.

batch: Determina el número de imágenes procesadas simultáneamente en cada iteración. Influye en la estabilidad del entrenamiento y en la utilización de la memoria GPU.

- **patience :** Implementa una estrategia de "early stopping". Si la métrica de validación (por defecto, la pérdida) no mejora durante 3 épochs consecutivas, el entrenamiento se detiene automáticamente, previniendo el sobreajuste y ahorrando tiempo de cómputo.
- **imgsz :** Establece el tamaño de las imágenes de entrada al modelo. Un tamaño adecuado permite capturar detalles importantes sin incrementar excesivamente la carga computacional.
- **lr0 (Tasa de aprendizaje inicial)**: Es como el "tamaño del paso" que da el modelo al aprender. Un valor adecuado permite un aprendizaje rápido y preciso, evitando pasos demasiado grandes o pequeños.
- **momentum:** Añade "inercia" al aprendizaje, como un impulso. Ayuda a acelerar el progreso en la dirección correcta y a superar pequeños obstáculos en el camino.
- **weight_decay:** Es una "penalización" para pesos muy grandes en el modelo. Fomenta modelos más simples y evita que se "memorice" demasiado los datos de entrenamiento, previniendo el sobreajuste.
- **optimizar:** Permite al modelo YOLO seleccionar automáticamente el optimizador más adecuado, simplificando la configuración y potencialmente mejorando el rendimiento. En nuestro caso no lo pondríamos, porque estamos configurándolos manualmente, pero es una buena manera de que el modelo utilice una configuración adecuada.
- **augment:** En nuestro caso, será False ya que en nuestros modelos hemos hecho el aumento de dataset a través de nuestro método augment_dataset.
- **workers:** Define el número de procesos paralelos utilizados para cargar los datos, optimizando la velocidad de lectura y procesamiento.

La selección de los hiperparámetros es fundamental porque influyen en la capacidad del modelo para aprender patrones significativos de los datos. Un ajuste inadecuado puede resultar en un modelo que sobre ajuste los datos de entrenamiento (memorizándolos en lugar de generalizar), o que subajusta (no aprendiendo patrones relevantes).

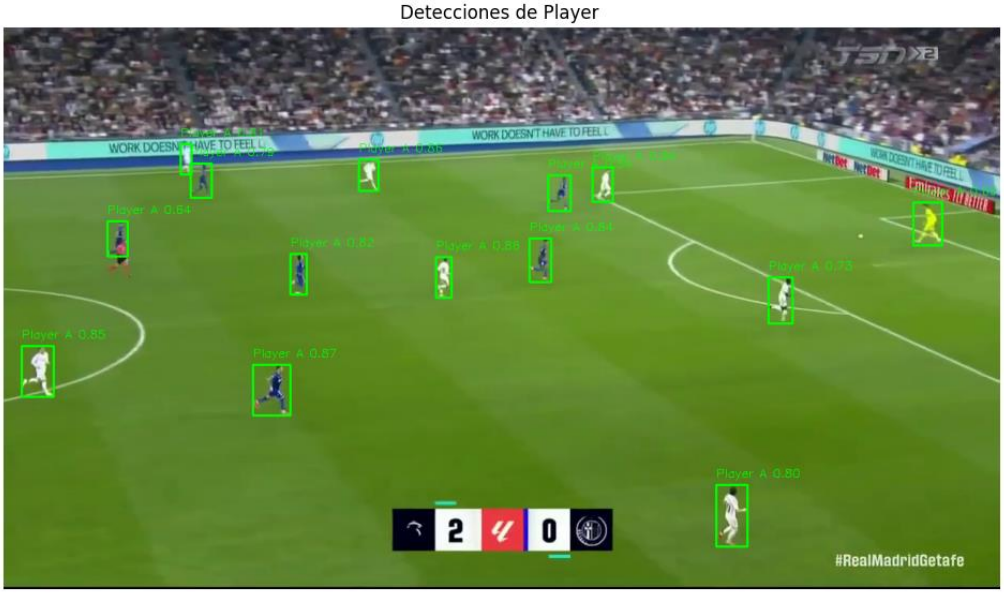
Por ejemplo, una configuración posible podría ser la del archivo fineTuning_YOLO.py:

```
from ultralytics import YOLO

model = YOLO('yolo11m.pt')
results = model.train(
    data='data.yaml',    # Indica la ruta del archivo data.yaml
    epochs=50,
    batch=16,
    patience=3,
```

```
    imgsz=648,
    device=0,
    lr0=0.01,
    momentum=0.937,
    weight_decay=0.0005,
    augment=False,
    save=True,  # Guarda el modelo en cada epoch
    project='runs/fine_tuning', # Directorio donde se guardarán los resultados
    name='acuity11m_v3',     # Nombre del modelo
    workers=16,
    verbose=True     # Muestra información detallada del entrenamiento
)
```

Para el entrenamiento del modelo, se ha optado por utilizar la plataforma Roboflow. Esta decisión se basa en que Roboflow tenía una prueba gratuita y me permitía usar más recursos computacionales que a los que podía llegar con los recursos propios. Esto me ha permitido agilizar el proceso de desarrollo y experimentación, superando con creces los resultados que esperaba. De todas maneras, he podido entrenar dos modelos de manera local que son acuity11m_v1 y acuity11m, y son modelos con un buen desempeño

Detecciones de Player



No obstante, he creado el fineTuning_YOLO. Este script está diseñado para permitir el fine-tuning del modelo YOLOv11m utilizando hardware propio. De esta manera, se ofrece una alternativa en el repositorio para replicar el entrenamiento o ajustarlo con sus propios datasets y recursos locales.

Proceso de Fine-Tuning

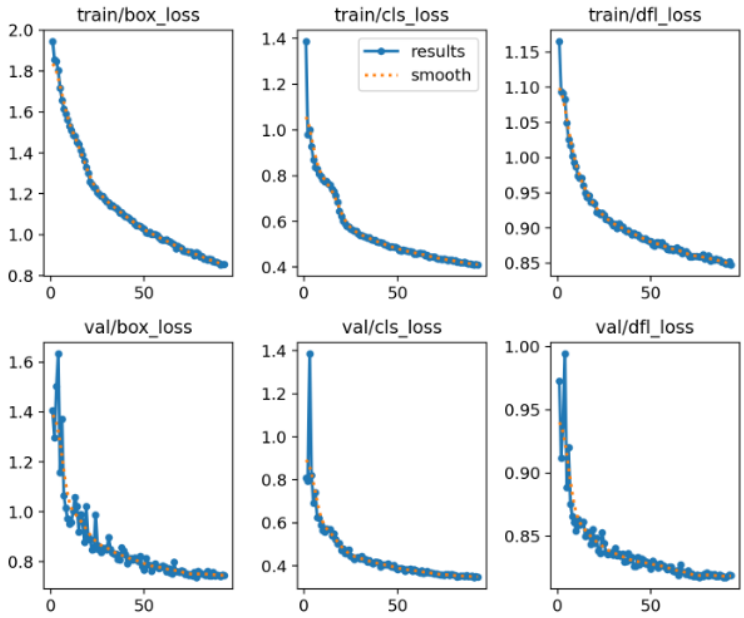
Durante el proceso de fine-tuning del modelo, se van calculando métricas que permiten observar la calidad del modelo de detección. Voy a explicar varias de estas métricas para luego poder ver cómo se ha ido desarrollando en mi modelo en concreto:

- **Box Loss:** Hace referencia al error en la predicción de las coordenadas de los cuadros delimitadores (bounding boxes). Un valor alto de box loss indica que las predicciones de

localización (posición y tamaño) de los objetos son imprecisas. Un valor bajo significa que el modelo está ajustando bien el contorno de los objetos.

- **Class Loss (cls_loss)**: Está relacionado con la clasificación de las clases (p. ej., si el modelo detecta correctamente a qué clase pertenece cada objeto). Un valor alto indica que el modelo está teniendo dificultades para asignar la categoría correcta; un valor bajo indica que está acertando en las clases.
- **Object Loss (dfl_loss)**: Suele medir el grado en que el modelo detecta la presencia de un objeto en un punto determinado del espacio de la imagen. Cuando es muy alto, significa que el modelo no está reconociendo la existencia del objeto o está confundido; a medida que disminuye, indica que el modelo está aprendiendo a identificar con más seguridad dónde hay objetos.

Estas pérdidas se monitorean tanto en el conjunto de entrenamiento como en el de validación. El objetivo es que los valores de *loss* en ambos conjuntos disminuyan de forma progresiva, lo cual sugiere que el modelo está aprendiendo. De ahí que en la gráfica de abajo haya una métrica para cada uno de los dataset.

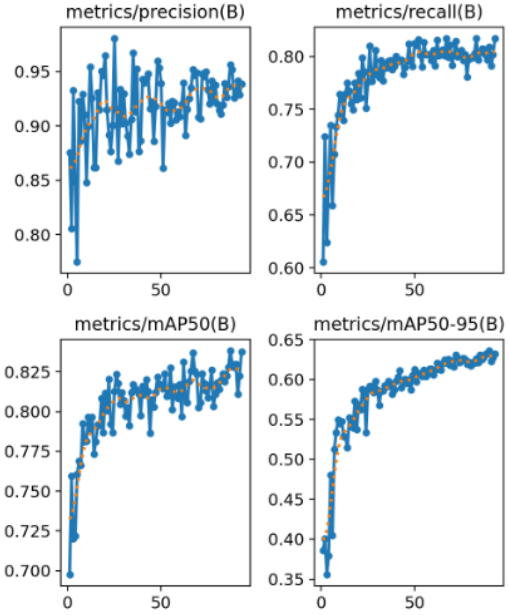


Pues como vemos, las métricas siguen un patrón común:

1. **Picos altos al inicio:** Durante las primeras épocas, el modelo apenas comienza a ajustarse, por lo que el error de predicción es alto.
2. **Descenso progresivo.** Conforme avanza el entrenamiento las curvas tienden a bajar, eso significa que el modelo va aprendiendo, cada vez ajusta mejor tanto la posición de las cajas, como la asignación de la clase.
3. **Estabilización:** En torno a las últimas épocas, el descenso se ralentiza y la curva se va aplanando. Esto indica que el modelo está llegando a un punto donde cada vez es más difícil mejorar su aprendizaje.

Otras métricas comunes son la de **precisión** y **recall**, además de los tipos de **mean Average Precision (mAP)**. Voy a definir los conceptos para poder analizar las gráficas y nuestro desempeño en la fase de validación:

- **Precisión**: Indica que porcentaje de las detecciones por el modelo son correctas. Si el modelo detecta 10 objetos y 9 de ellos son verdaderamente lo que se buscaba, la precisión será de 0.9.
- **Recall:** Mide el porcentaje de los objetos de la imagen que fueron detectados. Para ver mejor la diferencia con la precisión, precisión indica de los detectados cuales fueron correctos y recall, de todos los objetos reales cuantos se detectaron.
- **mAP@50:** El 50 significa que es el mean Average Precision pero a IoU=0.5. Es una métrica habitual que es la que nos indica cuán de bien detecta y clasifica el modelo, promediando entre TODAS las clases. Un mAP cercano a 1, indica que el modelo localiza muy bien los objetos.
- **mAP@50:95:** En este casi, se promedia a distinto umbral de IoU (mide cuanto se solapan la caja predicha y la real). Es una métrica que tiene el mismo objetivo que la anterior, pero es más estricta.



En las gráficas se observa como a lo largo de la creación del modelo, mAP va incrementando progresivamente. Esto indica que el modelo está prediciendo correctamente. El problema viene cuando vemos las métricas por clase y hay una que esta muchísimo más baja que las demás, en este caso mi modelo es bastante peor con la detección de la pelota. Es normal, para el dataset tan pequeño, al ser un objeto más difícil que jugador y arbitro ya que es más pequeño, a veces es indistinguible.

Evaluación final del modelo

Después se aplica el modelo final al conjunto de testa que anteriormente en la parte de preparación de datos nombrábamos que era importante para la evaluación final del modelo. Estas imágenes no la se ha visto el modelo ni durante el entrenamiento ni durante la validación y es su manera de medir el desempeño real.

Vemos un mAP@50 global del 91%, superior al de la validación que era un 83% y además que la clase 1 (balón) también ha mejorado de un 50% a un 74%. Con un dataset de validación y test tan pequeños puede ser que sus imágenes tengan características que lo hagan más fácil. Aunque hayan subido las métricas del test en comparación con el de validación, se puede observar que van por el mismo camino y no difieren demasiado.

He probado con varias imágenes, y las métricas ya nos avanzan que desempeño va a tener el modelo. Vemos como los jugadores de futbol, los detecta bien, con un grado alto de confiabilidad, aparte en esta imagen, el balón también lo detecta correctamente.

En la segunda imagen por ejemplo vemos un caso específico (un jugador caído) con solo un 21% de confianza, posiblemente porque su postura es muy distinta de la habitual (está en el suelo) y el modelo duda más al clasificarlo como “jugador”.

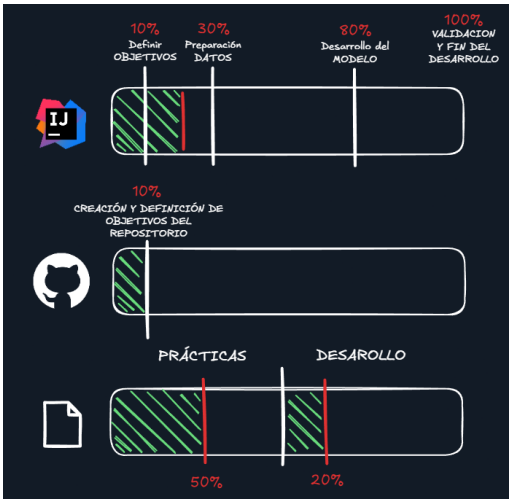
3.4.3. Resumen de contribuciones y productos desarrollados

- Modelo desde cero: Se creó una CNN básica para detección de jugadores, árbitros y balones, con resultados limitados debido a la falta de datos y capacidad computacional.
- Dataset propio: Se extrajeron frames de partidos, se anotaron manualmente y se aplicaron técnicas de aumento de datos.
- Fine-tuning sobre YOLOv11: Se ajustó un modelo preentrenado, logrando un mAP@50 del 91%, con mejor detección de jugadores y árbitros, pero con dificultades en el balón.
- Evaluación: La prueba confirmó un buen desempeño.

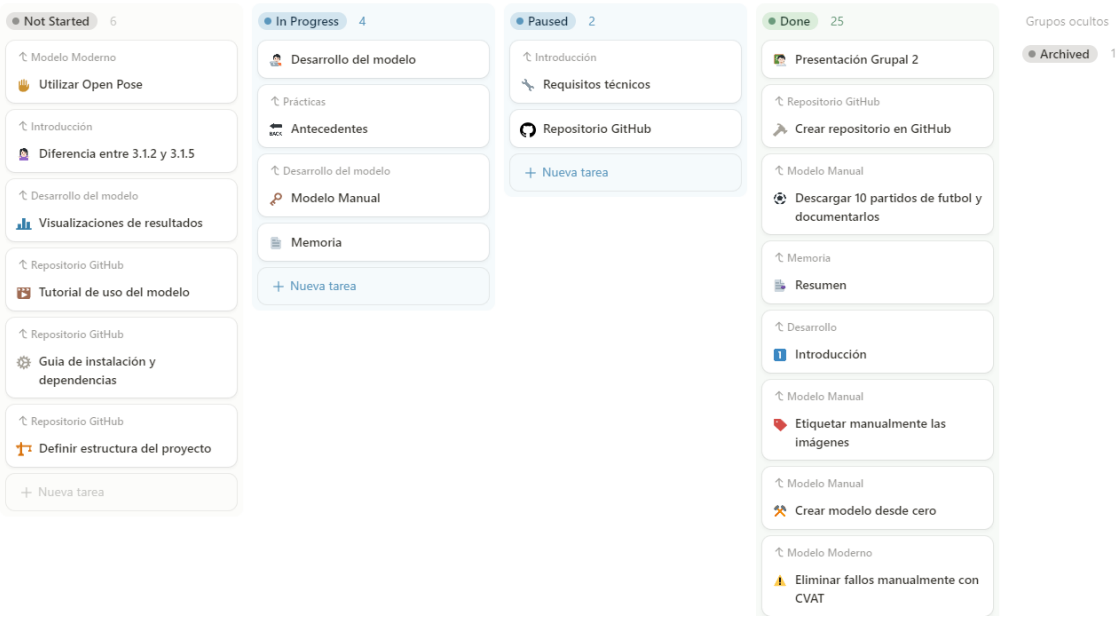
3.4.4. Planificación temporal

La planificación del proyecto la he llevado a cabo utilizando la herramienta de Notion, es un software enfocado en gestión de proyectos, aunque tiene infinitos usos. Esta aplicación me permite de manera fácil, y en un entorno amigable el poder gestionar las tareas y a la vez el poder tener todo el contenido que voy desarrollando de una manera organizando.

He dividido la planificación en cuatro ramas, el desarrollo del modelo, el repositorio de github, la memoria y otro último bloque para las tareas que puedan ir surgiendo como las presentaciones grupales.



Utilizo esta herramienta porque creo una base de datos de proyectos y tareas, y luego puedo crear vistas de una manera sencilla de los datos, como un cronograma o una vision de tablero para ir cambiando el estado de las tareas (Not Started, In Progress, Paused, Done o Archived).



3.4.5. Recursos empleados

El IDE utilizado que ha sido la herramienta principal de proyecto ha sido IntelliJ IDEA de JetBrains, en mi caso he utilizado la versión Ultimate que es de pago, pero tengo de manera gratuita gracias a todas las herramientas que proporciona GitHub Education.

Aparte he utilizado el programa CVAT para la anotación de las imágenes, ha sido la primera vez que lo he utilizado y tiene todas las herramientas necesarias para la anotación. Para agilizar un poco el proceso adquirí durante un mes la versión de pago “Solo” para las anotaciones de track, que utiliza un método que no es perfecto, pero genera anotaciones a partir de las últimas imágenes anotadas. También tenía anotaciones automáticas ilimitadas, pero no las he utilizado para seguir el proceso manual.

El cómputo del entrenamiento ha sido a través de la prueba gratuita de Roboflow, aparte aquel computo que haya hecho mi ordenador como la creación del modelo desde 0, lo ha hecho mi ordenador personal que tiene el siguiente hardware: CPU AMD Ryzen 5 5600X 6-Core Processor y una velocidad base de 3,70 GHz, una GPU NVIDIA GeForce RTX 3060 Ti de 16GB y una memoria RAM de 16 GB.

3.4.6. Trabajo desarrollado

3.5. Resultados y discusión

3.6. Conclusiones

En conclusión, considero que el verdadero potencial como usuario radica en aprender a realizar fine-tuning sobre modelos grandes ya preentrenados, ya que esto permite adaptar estas poderosas herramientas a necesidades específicas sin necesidad de entrenarlas desde cero. Esta capacidad no solo optimiza el rendimiento y la eficiencia, sino que también abre la puerta a aplicaciones más personalizadas y precisas, maximizando el valor que se puede obtener de la inteligencia artificial en distintos ámbitos.

3.7. Líneas futuras

Donde mas veo futuro en estos modelos, es que no solo lleguen al deporte profesional si no a los deportes más amateur. Y que cualquier equipo de futbol pueda poner su móvil a grabar y poder sacarle provecho a la tecnología como hacen los equipos profesionales.

4. Bibliografía

YOLOv11 explained: Next-level object detection with enhanced speed and accuracy
Medium (Autor: Nikhil Rao)
https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f71

Documentación oficial de YOLO11 (Overview)
Ultralytics
https://docs.ultralytics.com/es/models/yolo11/#overview

Sitio oficial de PyTorch
PyTorch (PyTorch Foundation)
https://pytorch.org/

Issue #10375 (Repositorio Ultralytics)
GitHub - Ultralytics
https://github.com/ultralytics/ultralytics/issues/10375

Mean Average Precision
Blog de Roboflow
https://blog.roboflow.com/mean-average-precision/

Precision and Recall
Blog de Roboflow
https://blog.roboflow.com/precision-and-recall/

Red Neuronal Convolucional desde 0
Medium (Autor: Daniel Dhats)
https://medium.com/@danieldhats7/red-neuronal-convolucional-desde-0-f3150ba0b57e

Convolutional Neural Networks (Explicación)
IBM
https://www.ibm.com/es-es/topics/convolutional-neural-networks

Redes Neuronales Convolucionales con CIFAR-10 (Sección informativa)
Universidad Politécnica de Madrid (UPM) - dcain.etsin.upm.es
https://dcain.etsin.upm.es/~carlos/bookAA/05.7_RRNN_Convoluciones_CIFAR_10_INFORMATIVO.html