Desarrollo de un pipeline ETL para la formación y validación de modelos de machine learning

Marc Campo Cabrera

17 de junio de 2025

Resum – El proyecto "Desarrollo de un pipeline ETL para la formación y validación de modelos de machine learning" se centra en la primera fase del proceso de desarrollo de modelos de inteligencia artificial: la extracción, transformación y carga (ETL) de datos. El objetivo principal es diseñar un pipeline ETL automatizado y estructurado que permita gestionar la recolección, validación y almacenamiento de datos de forma continua. En concreto, en este trabajo los datos a procesar son imágenes. Así pues, se utilizarán lotes pequeños, como grupos de 20 imágenes, con el fin de optimizar el flujo de datos y evitar cuellos de botella. Además, se incorporarán reglas de validación que aseguren la integridad y calidad de los datos, detectando posibles inconsistencias antes de su almacenamiento definitivo.

Palabras clave – Machine learning, pipeline, ETL, Python, bases de datos, automatizacion, Prefect, CVAT, Recolección de datos, Preprocesamiento de imágenes, Validación de datos, Anotación.

Abstract– This project, titled "Development of an ETL Pipeline for the Training and Validation of Machine Learning Models," focuses on the initial phase of AI model development: the extraction, transformation, and loading (ETL) of data. The main objective is to design an automated and structured ETL pipeline capable of managing data collection, validation, and storage in a continuous process. Specifically, in this work, the data to be processed are images. Therefore, small batches, such as groups of 20 images, will be used to optimize data flow and avoid bottlenecks. Additionally, validation rules will be implemented to ensure the integrity and quality of the data, detecting potential inconsistencies before they are stored.

Keywords– Machine learning, pipeline, ETL, Python, databases, automation, Prefect, CVAT, Data collection, Image preprocessing, Data validation, Annotation.

1 Introducción - Contexto

El proyecto "Desarrollo de un pipeline ETL para la formación y validación de modelos de machine learning" se enmarca dentro del proceso de desarrollo de modelos de machine learning. Se ha desarrollado en una empresa que implementa soluciones basadas en IA para mejorar las capacidades humanas.

En este contexto, el entrenamiento del algoritmo se divide en tres fases principales:

1. **Fase de Extracción, Transformación y Carga (ETL) [1] [2] [3] [4]:** En esta fase, se recopilan los datos provenientes de diversas fuentes, se aplican las validaciones necesarias y se descartan aquellos datos que no son relevantes [5]. Después de ser filtrados, los datos se almacenan en una base de datos con sus respectivas etiquetas (labels) [6] [7], lo que facilita que el algoritmo de machine learning pueda utilizarlos de manera eficiente.

2. **Entrenamiento del modelo:** Una vez los datos están correctamente almacenados y etiquetados, el algoritmo comienza su entrenamiento. Durante este proceso,

• E-mail de contacto: 1633330@uab.cat
• Mención realizada: Tecnologías de la Información
• Trabajo tutorizo por: Ana Cortés (Departamento de Arquitectura de Computadores y Sistemas Operativos)
• Curs 2024/25

se realizan validaciones intermedias y se ajustan los parámetros del modelo

3. **Test y validación final:** En esta última fase, se evalúan los resultados generados por el algoritmo. Se revisan posibles errores y se etiquetan los datos de salida para futuras mejoras. Este paso es clave para asegurar que el algoritmo funciona correctamente y que su precisión es la que se desea antes de utilizarlo en un entorno real.

Dentro de todo este flujo de trabajo, el trabajo de final de grado (TFG) se centrará en la primera fase (ETL), cuyo objetivo es generalizar y centralizar todo el flujo de datos inicial. La idea principal es construir un pipeline ETL que permita gestionar la recolección, validación y almacenamiento de datos de forma automatizada y estructurada.

Para estructurar mejor el desarrollo del proyecto, la implementación inicial se hará sobre un subconjunto específico de datos, por ejemplo, simbología. Esto permitirá validar el funcionamiento del pipeline en un ámbito controlado antes de extenderlo a otras fuentes de datos.

Uno de los puntos clave de este proceso ETL es que debe ser **continuo y cíclico [8]**. No basta con una carga masiva de datos única, sino que la recolección, validación y almacenamiento deben realizarse de manera **ininterrumpida** para ir proporcionando datos constantemente al algoritmo. Para lograrlo, se trabajará con **lotes de imágenes/videos cerrados**. En lugar de procesar grandes volúmenes de datos de una sola vez, se manejarán conjuntos más pequeños, por ejemplo, grupos de 20 imágenes/videos a la vez. Mientras un lote de imágenes/videos está validando y almacenando en la base de datos [9], otro lote de imágenes/videos será recortado y procesado en paralelo. Esto permitirá un flujo de datos optimizado y evitará cuellos de botella en el sistema.

Además de la automatización del pipeline, otro aspecto importante será garantizar la integridad y calidad de los datos procesados. Para ello, se definirán una serie de reglas y validaciones que permitan detectar datos inconsistentes o erróneos antes de almacenarlos. Estas validaciones pueden incluir detección de duplicados, control de formatos, verificación de etiquetas incorrectas, entre otros. [10]

2 Estado del arte

En los últimos años, los pipelines ETL (Extracción, Transformación y Carga) se han vuelto una pieza clave dentro del procesamiento de datos, especialmente en entornos relacionados con la inteligencia artificial y el aprendizaje automático. Gracias a estos sistemas, es posible automatizar tareas repetitivas como la recogida y validación de datos, asegurando que lleguen en buenas condiciones a las fases de entrenamiento y evaluación del modelo [1, 2, 4].

Dentro del campo del machine learning, se ha demostrado que la calidad de los datos utilizados tiene un impacto directo en el rendimiento final del modelo. Por eso, cada vez es más común aplicar reglas de validación y limpieza de datos antes de entrenar cualquier algoritmo. Estas validaciones pueden incluir desde comprobar que no haya duplicados hasta verificar que los formatos de los archivos o las etiquetas asignadas sean correctas [5, 6, 7].

También es importante destacar la necesidad de automatizar este proceso. Cuanto más grande sea el volumen de datos, más difícil es gestionarlo manualmente. Automatizar el pipeline no solo ahorra tiempo, sino que además reduce los errores humanos y garantiza que el flujo de datos sea constante y sin interrupciones [8].

Existen distintas herramientas y tecnologías que permiten construir pipelines ETL escalables y adaptables. Estas soluciones están pensadas para trabajar con múltiples tipos de datos y orígenes, manteniendo siempre la trazabilidad del proceso y facilitando el mantenimiento y futuras ampliaciones del sistema [10].

El trabajo que se plantea en este TFG parte de estas ideas y busca adaptarlas a las necesidades concretas de la empresa, donde el flujo de información es constante y donde resulta fundamental tener una estructura que asegure la calidad y disponibilidad de los datos para entrenar modelos de forma eficiente.

3 Objetivo

El objetivo principal de este Trabajo de Fin de Grado es diseñar, desarrollar e implementar un pipeline ETL eficiente y automatizado para la fase de procesamiento de datos en el entrenamiento de modelos de machine learning dentro del entorno de la empresa. Este pipeline tendrá que permitir la extracción, transformación y carga de datos de manera estructurada y continua, garantizando la calidad e integridad de los datos para el modelo.

Deberá seguir los siguientes pasos:

- **Establecer un mecanismo de filtrado de datos:** Crear y aplicar un conjunto de normas y filtros que posibilitaran la eliminación de información errónea, incompleta o innecesaria antes de su almacenamiento en la base de datos.
- **Integrar con plataformas de intervención manual:** Es importante que el pipeline se integre con un sistema de intervención manual (para esas tareas que lo requieran) enfocado en la validación y corrección de los procesos automáticos.
    1. Procesamiento automático: El pipeline ejecuta cierta lógica o procesos que pre anotan los datos (ej. Reencuadrar automáticamente los símbolos) aunque puede haber errores
    2. Procesamiento manual: Los resultados del procesamiento automático se presentan a anotadores humanos para que puedan corregir errores cometidos antes de que los datos entren al conjunto de entrenamiento o test.
- **Establecer la estandarización de la ETL:** Crear un flujo de trabajo organizado que facilite la recopilación y validación de datos de forma eficaz para minimizar la intervención manual y garantizar la calidad de los datos almacenados.
- **Automatizar la recolección de datos en lotes cerrados:** Implementar un sistema de procesamiento constante que permita recopilar y almacenar información en conjuntos específicos (por ejemplo, 20 imágenes/videos o lo que se necesite simultáneamente), garantizando así la optimización del flujo de datos.

- **Crear un sistema modular y escalable:** Es crucial para establecer una estructura que facilite la incorporación de nuevas fuentes de datos en el pipeline en un futuro.
- **Establecer la continuidad del proceso de trabajo en el sistema:** Desarrollar sistemas que garanticen el funcionamiento constante del sistema sin interrupciones, lo que facilitará la recopilación y análisis de datos de manera automática y periódica

4 METODOLOGÍA

Para el desarrollo de este proyecto, se seguirá una metodología basada en un enfoque iterativo e incremental, dando prioridad a la automatización y validación continua del pipeline ETL.

Primeramente, se realizará un estudio detallado de los requisitos del pipeline dentro del flujo de trabajo de la empresa. Se definirán:

- Los tipos de datos que se procesarán
- Las reglas de validación y filtrado que se desean aplicar a cada tipo de dato.
- Las labels de identificación que se desean aplicar a cada tipo de dato.
- Las herramientas y tecnologías que utilizar en el desarrollo del pipeline.

Después se planteará más detalladamente la estructura del pipeline ETL, estableciendo los módulos principales.

- Extracción de datos: Obtención de las imágenes/videos desde distintas fuentes
- Transformación y validación: Aplicación de las reglas de filtrado y etiquetado definidas previamente.
    1. Procesamiento automático de los datos
    2. Integración con procesamientos manuales para las tareas que lo requieran. Para que humanos puedan validar si el procesamiento automático se está realizando correctamente.
- Carga a la base de datos: Almacenamiento estructurado y preparado para el modelo de machine learning.
- Automatización y control: Implementación de procesos para garantizar la ejecución ininterrumpida.

Una vez definidos los módulos, se desarrollará la primera iteración del pipeline ETL. En esta fase se desarrollará una primera versión del pipeline centrado exclusivamente en el procesamiento de un tipo de dato (ej. Simbología).

Se hará la extracción de imágenes/videos, el procesamiento y validación (más la integración con los procesos manuales), el almacenamiento en la base de datos y la automatización del proceso de la manera en la que se definió en la fase anterior.

Cada módulo será probado individualmente antes de ser integrado.

Una vez desarrollada esta primera versión del pipeline se realizará pruebas de distintos escenarios para validar la robustez y eficiencia del pipeline:

- Pruebas de rendimiento: Evaluación de tiempos de procesamiento.
- Pruebas de validación de datos: Comprobación de la precisión de los filtros y reglas que se están aplicando.
- Pruebas de integración: Verificación de la compatibilidad con la base de datos y el modelo de machine learning.

Se ajustarán los parámetros y se optimizará el pipeline en caso necesario.

Después de haber validado el pipeline para el primer tipo de datos, realizará las validaciones posteriores al entrenamiento del algoritmo con su correspondiente etiquetado, con el fin de calificar el trabajo realizado y corregirlo en futuras versiones

Por último, se generará la documentación técnica del pipeline, incluyendo:

- Descripción del sistema y su arquitectura.
- Manual de uso y despliegue.
- Guía de mantenimiento y futuras mejoras

5 RESULTADOS DEL TRABAJO

Los resultados obtenidos a lo largo del desarrollo del Trabajo de Fin de Grado demuestran que se ha alcanzado un alto grado de cumplimiento respecto a los objetivos establecidos en la planificación inicial. El pipeline ETL (Extract, Transform, Load) desarrollado es plenamente funcional y modular, permitiendo el procesamiento automatizado de datos con un enfoque escalable y mantenible.

Cabe destacar que ha sido muy importante durante todo el proceso de desarrollo que se ha mantenido un contacto constante con los responsables técnicos de la empresa colaboradora, lo cual ha permitido adaptar y ajustar el diseño a las necesidades reales del entorno productivo. Estas revisiones periódicas han sido fundamentales para asegurar que el sistema cumple con los criterios técnicos, de calidad y de escalabilidad requeridos por la organización.

5.1. Cumplimiento de los objetivos

Los principales objetivos planteados al inicio del proyecto han sido abordados satisfactoriamente:

- **Automatización de la búsqueda de imágenes**: Se ha conseguido implementar con éxito la integración con la API de búsqueda de Google, con un módulo robusto capaz de procesar múltiples símbolos de forma eficiente y controlada.

- **Control del flujo mediante Prefect [11]**: El uso del orquestador Prefect ha permitido estructurar el flujo completo del pipeline, dotándolo de mayor control, trazabilidad y flexibilidad. Ademas de proporcionar reintentos automáticos en caso de fallo en cualquier punto del pipeline.

- **Sistema de anotación manual integrado**: Aunque inicialmente se planteó usar Label Studio [13], tras un análisis más profundo se optó por CVAT debido a sus mayores capacidades. La integración mediante *webhooks* [14] y descarga automática de anotaciones permite cerrar el ciclo del pipeline de forma efectiva.

- **Persistencia y seguimiento**: Se ha desarrollado una base de datos local para almacenar los estados intermedios de los datos, facilitando la auditoría y reanudación de flujos si es necesario.

- **Eficiencia y paralelización**: Gracias al uso de multiprocessing y al modelo *producer-consumer*, se ha logrado reducir notablemente el tiempo de procesamiento, haciendo el pipeline apto para entornos reales.

5.2. Relación entre los resultados obtenidos y los previstos

Los resultados obtenidos están en línea con lo previsto en el análisis del contexto y los objetivos planteados. Durante la planificación se contempló una solución que fuese adaptable a un entorno industrial, y los resultados actuales reflejan esta intención: el sistema está preparado para escalar, adaptarse a diferentes orígenes de datos y responder de manera automática a eventos en la herramienta de anotación.

Cabe destacar que, aunque algunas decisiones tecnológicas se ajustaron durante el desarrollo (como el cambio de plataforma de anotación), estas adaptaciones han permitido obtener mejores resultados y una solución más profesional y robusta que la inicialmente prevista.

5.3. Alcance y viabilidad técnica

El trabajo desarrollado es perfectamente abordable por un ingeniero informático con conocimientos en Python, sistemas distribuidos, tecnologías web y bases de datos, ademas del conocimiento sobre los actuales algoritmos de *machine learning*. Todas las tecnologías utilizadas (Prefect, CVAT, API REST, *multiprocessing*) son de libre acceso y tienen una curva de aprendizaje razonable. Además, el sistema ha sido diseñado teniendo en cuenta la reutilización y la documentación, de forma que puede mantenerse y ampliarse fácilmente en un entorno real.

Los recursos disponibles (hardware local, acceso a APIs gratuitas y software open source) han sido suficientes para demostrar la viabilidad de la solución planteada. Por tanto, los resultados obtenidos son completamente alcanzables dentro del marco de un Trabajo de Fin de Grado de una Ingeniera Informática.

5.4. Descripción detallada de los módulos desarrollados

A continuación, se detallan los principales módulos que componen el pipeline ETL desarrollado, indicando su funcionalidad y cómo se integran entre sí dentro del sistema general.

5.4.1. Obtención y registro de símbolos

El primer módulo desarrollado es el encargado de leer los símbolos (palabras clave) desde un fichero Excel, que actúan como punto de partida para el resto del pipeline. Estos símbolos se registran automáticamente en la base de datos local, lo que permite hacer un seguimiento del estado de cada uno (nuevo, en cola, procesando o terminado). Además, incluye también una fecha de la última actualización para facilitar la recuperación en caso de que un *worker* se quede atascado, esto ultimo se detallará más adelante. Este registro garantiza trazabilidad y evita duplicidades en fases posteriores del flujo.
La figura 1 muestra un ejemplo de la base de datos una vez añadidos los símbolos con su correspondiente estado y última fecha de actualización.

id	symbol	status	updated_at
1	pepsi	new	2025-04-22 06:35:57.054356+00
2	coca_cola	new	2025-04-23 10:21:15.953971+00
3	nestea	done	2025-04-28 08:31:33.190046+00
4	fanta_naranja	new	2025-04-23 10:21:29.187895+00
5	fanta_limon	done	2025-04-28 08:31:22.476514+00
6	tesla	new	2025-04-23 10:21:29.575845+00
7	nike	done	2025-04-23 10:24:47.802522+00
8	adidas	processing	2025-04-28 08:32:47.030191+00
9	sprite	new	2025-04-14 11:17:21.877226+00
10	red_bull	new	2025-04-23 10:25:16.54223+00
11	apple	new	2025-04-23 10:25:46.516782+00
12	samsung	new	2025-04-15 11:04:45.950498+00
13	mcdonald's	new	2025-04-15 11:05:55.869462+00
14	burger_king	new	2025-04-22 06:36:02.322201+00

Fig. 1: Base de datos de símbolos

5.4.2. Búsqueda y descarga automatizada de imágenes

Utilizando la *Google Custom Search API*, se ha implementado un sistema que permite obtener imágenes relacionadas con cada símbolo registrado. Este módulo se ha diseñado para ser robusto frente a errores comunes (como enlaces rotos o URLs inválidas) y permite realizar búsquedas eficientes por lotes, controlando el número de resultados y respetando las cuotas de la API.

5.4.3. Simulación local de la búsqueda con la API (*mock*)

Para facilitar el desarrollo y las pruebas del pipeline sin depender constantemente de los límites de uso de la API de Google, se ha implementado un sistema de *mock* que simula las respuestas de la búsqueda.

Este módulo permite cargar resultados prealmacenados en ficheros JSON, emulando así la respuesta real que se obtendría al realizar una consulta en la API. De esta forma,

se pueden probar todos los módulos del pipeline de forma local y sin consumir cuotas de la API. (ver figura 2)

La integración con el pipeline es transparente, permitiendo activar o desactivar el uso del *mock* mediante el parámetro `use_mock=True` dentro del flujo `etl_flow`. En caso de activarse, los resultados se leen desde disco en vez de realizar la consulta real.

```python
class MockSearchSymbol:
    def __init__(self, base_folder="mock_data"):
        self.base_folder = base_folder

    def load_results(self, symbol: str) -> list[dict]:
        file_path = os.path.join(self.base_folder, f"{symbol}_search_results.json")
        if not os.path.exists(file_path):
            print(f"⚠️ Archivo mock no encontrado: {file_path}")
            return []

        with open(file_path, "r", encoding="utf-8") as f:
            data = json.load(f)

        print(f"✅ Mock cargado para símbolo: {symbol}")
        return data
```

Fig. 2: Carga local de resultados simulados para un símbolo

Esta funcionalidad ha permitido avanzar significativamente en el desarrollo del pipeline y realizar pruebas en entornos desconectados o con conectividad limitada.

5.4.4. Filtrado automático de imágenes

Una vez descargadas, las imágenes pasan por un sistema automático de validación basado en detección de duplicados mediante hashes (hash perceptual), verificación de formato y comprobación de integridad básica. Este proceso asegura que las imágenes almacenadas cumplan un mínimo de calidad para ser utilizadas en las etapas posteriores de anotación y entrenamiento.

5.4.5. Almacenamiento estructurado

Las imágenes válidas se organizan en un sistema de almacenamiento estructurado, junto con sus metadatos asociados (identificador, símbolo, hash asociado y estado). Este diseño permite una integración sencilla con otros módulos del sistema y asegura la consistencia del dataset en todo momento. El estado se va modificando según las imágenes van avanzando en el proceso del pipeline.

- **Descarga de la imagen**: Cuando la imagen se descarga de la API (o del `mock`, en el caso de estar testeando), se guarda en la base de datos con el estado `downloaded`, que es el estado por defecto en la inserción en la base de datos.
- **CVAT**: Después, cuando la imagen ya se ha procesado, filtrado y se ha añadido a la plataforma de labeling de CVAT. La imagen pasa al estado `cvat`, para que se pueda saber de manera clara si la imagen esta pendiente de hacer el etiquetado manual.
- **Anotación validada**: Finalmente, cuando la imagen ya se ha etiquetado manualmente y se ha hecho una validación (también manual). La imagen pasa ahora al estado final: `annotated`.

La figura 3 visualiza un ejemplo de la base de datos. Cada imagen contiene el uuid, el símbolo al que pertenece, el hash perceptual y, a la derecha de todo, el estado en el que se encuentran.

Se puede ver que cada imagen tiene un estado diferente según el punto en el que esté del pipeline. Esto permite un control rápido y hace que se pueda revisar en cualquier momento cómo va avanzando la pipeline.

e6ef9e2e-20e2-4890-b392-510c72a6b7d7.jpg	coca_cola	ffffb1010101ffff	cvat
ff39d373-300c-490a-9113-bfb3247b5902.jpg	coca_cola	7f01607e5e70787f	annotated
cf00473e-e0e1-4122-8b04-f6f7434678cd.jpg	coca_cola	18001818787c7c78	cvat
ceb10880-52a3-463b-a2f0-bfb21ba1112d.jpg	coca_cola	c090b0b8f8fe0c1c	cvat
63ebce7e-0d47-4693-9aa0-4bf0891b833b.jpg	coca_cola	7effbf1b84000000	cvat
cd187ba8-5ea4-4860-a9ce-65adfe85201f.jpg	coca_cola	e7e7e7e37b737040	downloaded
67a419c4-38ac-4258-8a03-b12a0a7f062b.jpg	coca_cola	ffffc38181c3ffff	downloaded
f3a5b174-b3f6-4586-9d80-f4f5c2cddc5d.jpg	coca_cola	ffff70040080ffff	downloaded
ff96823c-1e3e-48e7-8f1f-12614e99f898.jpg	coca_cola	ffffc0b500033f00	downloaded
809f36e3-dd93-4a00-830e-59677146c0e4.png	pepsi	e763e3e3e16363e3	downloaded
7ae02108-6b3b-4832-a7c5-66e7d526e59e.jpg	pepsi	ffe3e0f3ebc1c140	downloaded
6be247cb-023c-4eb3-acca-68c82e9a83c1.jpg	pepsi	ffffffff000000f0	cvat
b827a52a-f897-40d2-9521-f3b2e745ed1d.jpg	pepsi	fff8f80000f8f8ff	cvat

Fig. 3: Base de datos de ficheros

5.4.6. Automatización y orquestación del pipeline

Para coordinar todos los módulos anteriores, se ha utilizado la herramienta Prefect [11], que permite definir y monitorizar flujos de trabajo de forma modular. Gracias a Prefect se ha logrado un control detallado del estado de ejecución de cada símbolo, así como un sistema tolerante a errores y adaptable a diferentes configuraciones.

La figura 4 muestra un ejemplo de una ejecución del pipeline.

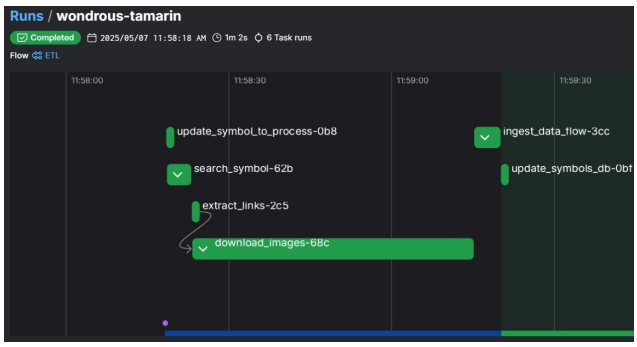

Fig. 4: Ejemplo de ejecución con Prefect

5.4.7. Integración con plataforma de etiquetado (CVAT)

Se ha preparado la integración con CVAT, la herramienta de anotación utilizada en la empresa actualmente. Esta integración permite exportar imágenes por lotes y recuperar posteriormente las etiquetas generadas de forma manual.

La recuperación de estas anotaciones de las imágenes de CVAT se hace mediante *webhooks* [14]) y un servidor en escucha. Cuando las anotaciones están en el estado de validación y se completa esa validación, el servidor recoge esta anotación automáticamente y la guarda en local para que se utilice posteriormente en el algoritmo.

5.4.8. Creación de servidor con Python para recoger la anotaciones

Como se ha comentado previamente, es fundamental contar con un mecanismo que permita al pipeline detectar de forma automática cuándo se ha completado la anotación manual de las imágenes en la plataforma CVAT. Para ello, se ha implementado un servidor en Python que actúa como receptor de eventos mediante un *webhook* configurado desde la propia instancia de CVAT [12].

El servidor estará ejecutando en una máquina en la misma red para que sea accesible desde el contenedor con el CVAT.

Este *webhook* permite que CVAT notifique en tiempo real cualquier cambio relevante que ocurra dentro de un proyecto de anotación (creación de tareas, finalización de anotaciones, actualizaciones de estados, etc.). De este modo, no es necesario realizar comprobaciones periódicas o manuales, lo cual optimiza el flujo y reduce la carga del sistema.

El servidor ha sido diseñado para mantenerse en escucha en todo momento. Cada vez que se recibe una petición desde CVAT, el sistema analiza los datos del evento entrante y los filtra para detectar únicamente aquellos que corresponden a la finalización de una tarea de tipo `validation`. Una vez identificado uno de estos eventos, el servidor realiza las siguientes acciones:

1. Verifica si el evento corresponde a la tarea esperada y si su estado es `completed`.

2. Actualiza el estado correspondiente en la base de datos del pipeline, marcando que la anotación ha sido completada.

3. Descarga de forma automática el fichero de anotaciones directamente desde CVAT y lo almacena localmente en formato JSON para que se mas fácil manipularlo.

Este mecanismo asegura que la recogida de datos anotados se realice de forma automática, fiable y sin intervención manual, permitiendo su posterior uso directo por parte del módulo de entrenamiento del modelo. Además, se garantiza la trazabilidad del proceso, ya que todos los eventos procesados quedan registrados y se mantiene una copia local de cada anotación recuperada.

La figura 5 muestra un ejemplo de una anotación recibida automáticamente en el servidor. En esta se puede ver que se ha recogido un rectángulo (donde el anotador ha marcado que hay un símbolo) con los puntos (esquinas) delimitadas por los ”`points`”


```
[
  {
    "id": 920223504,
    "label_id": 4716358,
    "type": "rectangle",
    "frame": 56,
    "group": 0,
    "source": "manual",
    "occluded": false,
    "outside": false,
    "z_order": 0,
    "rotation": 0.0,
    "points": [
      162.97513107541636,
      141.90075105546384,
      781.5701681316477,
      761.7354575246336
    ],
    "attributes": [],
    "elements": []
  }
]
```


Fig. 5: Ejemplo de anotacion

5.4.9. Ejecución concurrente mediante multiprocesamiento

Con el objetivo de acelerar el procesamiento de símbolos y aprovechar al máximo los recursos del sistema, se ha implementado un sistema de multiprocesamiento basado en la librería estándar `multiprocessing` de Python.

Este enfoque permite lanzar múltiples instancias del pipeline en paralelo, de forma que varios símbolos puedan ser procesados simultáneamente, cada uno en un proceso independiente. Para evitar colisiones entre procesos y garantizar la consistencia de los datos, se ha diseñado una lógica de control basada en el estado de cada símbolo en la base de datos.

A través de una consulta con `SELECT ... FOR UPDATE SKIP LOCKED`, cada proceso obtiene un símbolo libre, lo marca como `processing` y lanza el pipeline correspondiente. De esta manera, se garantiza que no haya solapamientos entre procesos, incluso cuando se ejecutan en paralelo.

Esta estrategia ha permitido una mejora significativa en el rendimiento del sistema y ha sido clave para escalar el procesamiento de símbolos en entornos de producción o testeo masivo.

5.4.10. Modelo de ejecución distribuida: productor-consumidor

Para coordinar el trabajo de múltiples instancias del pipeline de forma ordenada y escalable, se ha adoptado un patrón de arquitectura *productor-consumidor*.

En este esquema, un proceso productor se encarga de consultar la base de datos y obtener los símbolos con estado `"new"` o pendientes de procesar. Estos símbolos se colocan en una cola de trabajo que es consumida por múltiples workers concurrentes.

Cada worker actúa como un consumidor que:

1. Extrae un símbolo de la cola.

2. Cambia su estado a `"processing"` en la base de datos (usando `SELECT ... FOR UPDATE SKIP LOCKED` para evitar colisiones).

3. Ejecuta el pipeline completo para ese símbolo.

4. Actualiza el estado del símbolo a `"done"` si ha sido procesado correctamente.

Este enfoque permite lanzar múltiples procesos en paralelo sin riesgo de solapamiento, mejora la eficiencia del sistema y garantiza la trazabilidad del estado de cada símbolo a lo largo del flujo de trabajo. Además, facilita la recuperación de símbolos atascados en estado `"processing"` mediante mecanismos de control periódicos.

De manera que cada worker revisa si hay un worker atascado. Se considera `atascado` cuando lleva mas de 30 minutos en `"processing"` (esto se comprueba con el campo `updated_at` de la tabla de símbolos), y si esta atascado, reinicia el estado de ese worker.

5.4.11. Documentación técnica y control de versiones

Paralelamente al desarrollo del pipeline, se ha ido documentando cada módulo, herramienta y decisión técnica adoptada. Esto incluye esquemas del sistema, manuales de uso y estructuras de configuración. También se ha llevado un control de versiones con Git [15] (levantado en Bitbucket [16]) para mantener la trazabilidad de los cambios.

6 CONCLUSIONES

El desarrollo de este Trabajo de Fin de Grado ha permitido diseñar e implementar un sistema ETL funcional y automatizado, orientado a la recolección, validación y preparación de datos para el entrenamiento de modelos de *machine learning* en un entorno empresarial real. A continuación, se resumen las principales conclusiones extraídas a partir del trabajo realizado, así como los aspectos pendientes y posibles líneas de mejora futura:

- **Arquitectura modular y escalable:** Se ha validado con éxito el diseño de una arquitectura dividida en módulos independientes, lo que ha facilitado tanto el desarrollo iterativo como la identificación y corrección de errores. Este enfoque modular permitirá escalar el sistema fácilmente hacia nuevos tipos de datos o fuentes.
- **Automatización eficiente del flujo de datos:** Gracias al uso de herramientas como Prefect y la implementación de flujos concurrentes, el pipeline es capaz de gestionar automáticamente tareas como la descarga, filtrado, validación, almacenamiento y exportación de imágenes. Esto reduce drásticamente la intervención manual y mejora la fiabilidad del proceso. Solo se usará la intervención manual para el proceso de etiquetado (y validación del etiquetado) que es esencial que se haga manualmente para poder entrenar al algoritmo de manera correcta y eficiente.
- **Uso de herramientas alineadas con el entorno técnico:** Aunque inicialmente se valoraron herramientas modernas como Label Studio, se ha priorizado la integración con CVAT, que ya estaba en uso en el entorno real. Esta decisión ha demostrado ser más eficaz para garantizar la compatibilidad y reducir la curva de adopción por parte del equipo técnico y los etiquetadores.
- **Estrategias para el desarrollo eficiente:** La implementación de un sistema de *mock* para simular resultados de la API de búsqueda ha sido clave para el desarrollo y testeo sin restricciones externas. Además, el patrón productor-consumidor con múltiples workers ha permitido escalar el procesamiento en paralelo de forma segura y eficiente.
- **Colaboración con el entorno profesional:** A lo largo del proyecto, se ha mantenido un contacto continuo con el equipo técnico de la empresa, lo que ha permitido alinear las decisiones técnicas con sus necesidades reales. Esto ha favorecido una visión práctica y profesional del problema, mejorando la aplicabilidad del sistema desarrollado.
- **Viabilidad del enfoque propuesto:** El resultado del trabajo demuestra que es viable construir un sistema automatizado, robusto y adaptado a flujos de datos reales mediante el uso de tecnologías actuales y buenas prácticas de desarrollo. La base sentada con este TFG permite seguir evolucionando el pipeline hacia nuevas funcionalidades como validación semántica, priorización por calidad o integración con sistemas de entrenamiento automático.

Posibles extensiones futuras

- **Integración con otros tipos de datos**, como texto o vídeo, de datos que puedan ser diferentes a los de simbología (que son los que se han usado para este trabajo), adaptando el pipeline actual a nuevos formatos.
- **Interfaz de usuario para visualización y gestión**, que permita a usuarios no técnicos revisar estados, consultar estadísticas y lanzar tareas desde una interfaz gráfica. Ya que ahora mismo no existe ningún tipo de interfaz gráfica para la detección y control de errores.
- **Recolección y análisis de resultados del modelo:** Aunque el trabajo se ha centrado en la preparación de datos y la automatización del proceso de entrenamiento, una extensión lógica sería implementar mecanismos para recopilar y analizar los resultados obtenidos por los modelos entrenados. Esto permitiría evaluar de forma objetiva la calidad del dataset generado y cerrar el ciclo de validación del sistema propuesto.

En conjunto, este proyecto ha sido una experiencia completa que no solo ha permitido aplicar los conocimientos adquiridos durante la carrera, sino también enfrentarse a los desafíos reales que conlleva el desarrollo de sistemas escalables en entornos de producción.

AGRADECIMIENTOS

Quiero agradecer en primer lugar a mi tutora, Ana Cortés, por su orientación, implicación y apoyo constante a lo largo del desarrollo del Trabajo de Fin de Grado.

También quiero expresar mi agradecimiento al equipo técnico con el que he tenido la oportunidad de colaborar, por compartir su experiencia y por facilitar el acceso a herramientas y entornos reales que han enriquecido notablemente esta experiencia académica.

Finalmente, agradezco a mi familia y amistades por su apoyo incondicional durante todo el proceso, tanto en lo personal como en lo académico.

Gracias a todos los que, de una forma u otra, han contribuido a que este proyecto se pueda llevar a cabo.

REFERENCIAS

[1] ≪What is ETL (Extract, Transform, Load)? — IBM≫. [En línea]. Disponible en: https://www.ibm.com/think/topics/etl

[2] ≪¿Qué es ETL? - Explicación de extracción, transformación y carga (ETL) - AWS≫, Amazon Web Services, Inc. [En línea]. Disponible en: https://aws.amazon.com/es/what-is/etl/

[3] ≪Extract, transform, load≫, Wikipedia. 2 de diciembre de 2024. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?=Extract,_transform,_load&oldid=1260714724

[4] ≪What is ETL Pipeline? Process, Considerations, and Examples≫, ProjectPro. [En línea]. Disponible en: https://www.projectpro.io/article/how-to-build-etl-pipelineexample/526

[5] ≪Top 5 Validation Methods In Machine Learning — by Mehmet Ali TOR — Medium≫. [En línea]. Disponible en: https://medium.com/@mehmetalitor/top-5-modelvalidation-methods-in-machine-learning-77eed8d08937

[6] ≪¿Qué es el etiquetado de datos? - Explicación del etiquetado de datos - AWS≫, Amazon Web Services, Inc. [En línea]. Disponible en: https://aws.amazon.com/es/what-is/data-labeling/

[7] ≪Data Labeling Essentials for Machine Learning Success≫. [En línea]. Disponible en: https://keylabs.ai/blog/data-labeling-essentials-for-machine-learning-success/

[8] I. Gowani, ≪ETL Automation Best Practices≫, DATAVERSITY. [En línea]. Disponible en: https://www.dataversity.net/etl-automation-best-practices/

[9] ≪10 Best Databases for Machine Learning and AI [2025]≫, GeeksforGeeks. [En línea]. Disponible en: https://www.geeksforgeeks.org/databases-for-machine-learning-ai/

[10] A. Reeve, Managing Data in Motion: Data Integration Best Practice Techniques and Technologies (The Morgan Kaufmann Series on Business Intelligence), Primera edición. Morgan Kaufmann.

[11] ≪Pythonic, Modern Workflow Orchestration For Resilient Data Platforms — Prefect≫. [En línea]. Disponible en: https://www.prefect.io/

[12] ≪Leading Image & Video Data Annotation Platform — CVAT≫. [En línea]. Disponible en: https://www.cvat.ai/

[13] ≪Open Source Data Labeling≫, Label Studio. [En línea]. Disponible en: https://labelstud.io/

[14] [D. González, ≪Qué es un *Webhook*), para qué sirve y cómo funciona≫, David González. [En línea]. Disponible en: https://davizgonzalez.com/blog/que-es-un-webhook/

[15] ≪Git≫. [En línea]. Disponible en: https://git-scm.com/

[16] Atlassian, ≪Bitbucket — Solución de Git para equipos que usan Jira≫, Bitbucket. [En línea]. Disponible en: https://bitbucket.org/product/es