Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Ciencia de Datos e Inteligencia Artificial

Trabajo Fin de Grado

Estudio de plataformas para la creación de competiciones aplicado a la tarea de clasificación de imágenes

Autor: Yimin Zhou
Tutor(a): Roberto Valle Fernández

Madrid, Junio 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ciencia de Datos e Inteligencia Artificial

Título: Estudio de plataformas para la creación de competiciones aplicado a la tarea de clasificación de imágenes

Junio 2024

Autor: Yimin Zhou
Tutor: Roberto Valle Fernández
Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Resumen

En el contexto actual, las competiciones de inteligencia artificial son fundamentales para impulsar el desarrollo y la evaluación de modelos. Este trabajo se enfoca en estudiar las distintas plataformas para la creación de competiciones, centrándose en una competición de clasificación de imágenes de satélite.

Para desarrollar la competición, se ha hecho uso de dos herramientas distintas: EvalAI y Codabench. Ambas plataformas son reconocidas por su capacidad para organizar competiciones de aprendizaje automático. Una vez desarrollada la competición en ambas plataformas, se llevó a cabo una comparación entre ellas tanto a nivel del administrador como de los participantes.

Además, se llevó a cabo una simulación que implicó la incorporación de resultados obtenidos por diversos participantes una vez creada la competición para analizar el rendimiento de las plataformas en un entorno de competición realista.

Palabras clave: competición, EvalAI, Codabench, clasificación de imágenes, evaluación.

Abstract

In the current context, artificial intelligence competitions are essential for driving the development and evaluation of models. This work focuses on studying various platforms for creating competitions, with a focus on an image classification competition using satellite images.

To develop the competition, two different tools were used: EvalAI and Codabench. Both platforms are recognized for their ability to organize machine learning competitions. Once the competition was developed on both platforms, a comparison was made between them at both the administrator and participant levels.

Additionally, a simulation was conducted that involved incorporating results obtained by various participants after the competition was created to analyze the performance of the platforms in a realistic competition environment.

Keywords: competition, EvalAI, Codabench, image classification, evaluación.

Tabla de contenidos

Capítulo 1

Introducción

En este primer capítulo se expondrán la motivación para la elaboración de este trabajo (Sec. 1.1), los objetivos planteados (Sec.1.2) y la estructura del trabajo (Sec. 1.3).

1.1. Motivación

Las competiciones desempeñan un papel fundamental en la comunidad de inteligencia artificial (IA). Estos eventos ofrecen una serie de beneficios:

- **Motivación de los participantes**. Las competiciones motivan a los participantes al proporcionar un entorno competitivo donde pueden ver los resultados de sus competidores y aspirar a superarlos.
- **Automatización de las comparativas de resultados**. Las competiciones automatizan la comparación de los resultados obtenidos por los participantes, lo que agiliza el proceso de evaluación.
- **Privacidad del conjunto de pruebas**. En muchos casos, los conjuntos de datos utilizados para entrenar y validar los modelos pueden contener información sensible o confidencial. En una competición, se puede garantizar la privacidad del conjunto de pruebas al no revelar las anotaciones del conjunto a los participantes. Es decir, los participantes pueden competir sin necesidad de conocer los detalles específicos del conjunto de datos de prueba.

En este contexto, se propone la creación de una competición aplicada a un problema de clasificación de imágenes. Se empleará un subconjunto de imágenes seleccionadas de xView [1], donde cada imagen contiene la representación de un objeto. El objetivo de la competición es desarrollar algoritmos y modelos de visión por computador que puedan identificar con precisión el objeto presente en cada imagen.

Una vez creada la competición, se llevará a cabo una simulación que involucrará la integración de los resultados obtenidos por diversos participantes.

1.2. Objetivos

El presente trabajo se enfoca en:

- Estudiar distintas plataformas de creación de competiciones en tareas de IA.
- Seleccionar la plataforma más adecuada y crear una competición para la tarea de clasificación de imágenes.
- Participación en la competición con diferentes modelos de Deep Learning.

Para ello, se propone los siguientes objetivos:

- **Estudio de plataformas**:
  - Elección de las herramientas para la elaboración de la competición.
  - Diseño de la competición.
- **Creación de la competición**:
  - Desarrollo de la competición.
  - Despliegue de la competición.
- **Participación en la competición**:
  - Entrenamiento de modelos.
  - Análisis de los resultados obtenidos.

1.3. Estructura del trabajo

El presente trabajo se estructura de la siguiente manera:

- **Capítulo 1: Introducción**. Se exponen las motivaciones del trabajo, los objetivos principales y se presenta la estructura general del documento.
- **Capítulo 2: Estado del arte**. Se presenta la revisión bibliográfica realizada, exponiendo las principales herramientas que existen para la creación de competiciones de aprendizaje automático.
- **Capítulo 3: Base de Datos**. Se describe el conjunto de datos utilizado en la competición, incluyendo su origen y características.
- **Capítulo 4: Desarrollo de la competición**. Este capítulo aborda la creación de la competición utilizando dos herramientas distintas, destacando el proceso y las consideraciones técnicas involucradas en cada plataforma.
- **Capítulo 5: Participación en la competición**. Se exponen los distintos modelos usados en la competición creada en dos plataformas, así como sus análisis y comparaciones.
- **Capítulo 6: Trabajo futuro**. Se discuten posibles direcciones para futuras investigaciones y desarrollos.

- **Capítulo 7: Análisis de impacto**. Se discuten los impactos del presente trabajo en relación con los Objetivos de Desarrollo Sostenible (ODS).

Capítulo 2

Estado del arte

En este segundo capítulo se presentarán las distintas herramientas que existen hoy en día para la creación de competiciones.

2.1. Kaggle

Kaggle es una plataforma comunitaria en línea para científicos de datos y entusiastas del aprendizaje automático. Fue fundada en 2010 por Anthony Goldbloom y Jeremy Howard y adquirida por Google en 2017 [2].

La plataforma ofrece conjuntos de datos públicos, cuadernos de aprendizaje automático y tutoriales para ayudar a los usuarios a aprender y practicar sus habilidades en ciencia de datos y aprendizaje automático. Pero una de las características que hizo a Kaggle un recurso tan popular fueron sus competiciones. Kaggle organiza diversos concursos patrocinados por organizaciones, que van desde la predicción de resultados médicos a la clasificación de imágenes [3]. Los participantes pueden enviar sus modelos y ver sus resultados en una clasificación pública, así como recibir comentarios de otros competidores y de la comunidad. Además de poder participar en las competiciones existentes, Kaggle permite a los usuarios crear sus propias competiciones.

Hasta la fecha, Kaggle cuenta con más de 17 millones de usuarios registrados. Desde su inicio, se han organizado más de 5.600 competiciones en la plataforma. Durante estas competiciones, se han realizado más de 13 millones de envíos por parte de los participantes. Estos datos demuestran el impacto significativo que Kaggle ha tenido en la comunidad de ciencia de datos y aprendizaje automático.

A pesar de su popularidad, Kaggle tiene varias limitaciones. Una de ellas es que, al ser una plataforma de código cerrado, Kaggle tiene limitaciones para admitir tareas de IA complejas que requieren métricas personalizadas distintas de las habituales disponibles a través de la plataforma. Otra limitación es la imposibilidad de alojar una competición con divisiones de prueba o trabajadores privados. Es decir, no se puede configurar una competición en Kaggle de tal manera que la parte de prueba del conjunto de datos o los participantes sean privados y no estén disponibles públicamente. Esto puede ser problemático para

proyectos que trabajan con datos sensibles o que requieren confidencialidad en ciertas etapas de la competición.

2.2. EvalAI

EvalAI es una plataforma de código abierto desarrollada por CloudCV, una plataforma en la nube creada en 2013 por estudiantes y profesores del "Machine Learning and Perception Lab" de Virginia Tech (ahora en Georgia Tech). Su objetivo principal es hacer que la investigación en Inteligencia Artificial (IA) sea más reproducible, proporcionando herramientas para que los investigadores puedan construir, comprar y compartir algoritmos de última generación de manera más accesible [4]. CloudCV está enfocado en proporcionar herramientas para la investigación en IA que sean accesibles para una amplia gama de usuarios. La plataforma EvalAI, como parte de este esfuerzo, se ha convertido en una solución escalable para evaluar modelos de aprendizaje automático, permitiendo a investigadores, estudiantes y científicos de datos crear, colaborar y participar en desafíos de IA en todo el mundo.

La plataforma, desde su lanzamiento en 2017, ha sido adoptada por más de 25 organizaciones, tanto de la industria como académicas, incluyendo empresas como Facebook y eBay, así como instituciones como Stanford y MIT. Además, organizaciones de investigación como Mapillary Research e IBM Research han optado por utilizar versiones derivadas de EvalAI para alojar sus propios desafíos internos, en lugar de desarrollar sus propias soluciones desde cero [5].

EvalAI cuenta con una comunidad activa que supera los 18.000 usuarios registrados y más de 200 competiciones alojadas hasta la fecha. Además, ha habido más de 180.000 envíos de participantes realizados en estas competiciones, lo que demuestra la participación activa y el interés continuo en los desafíos alojados en la plataforma.

Las plataformas tradicionales como Kaggle se centran en tareas de IA tradicionales como clasificación de imágenes, reconocimiento de texto y detección de objetos, que siguen un protocolo estándar donde los modelos pueden ser evaluados de manera aislada utilizando métricas automáticas simples como *accuracy*, *precision* o *recall*. Sin embargo, con el éxito de las técnicas de aprendizaje profundo en una amplia variedad de tareas (como generación de imágenes), surge la necesidad de herramientas que permitan la evaluación de sistemas IA en contextos más complejos. EvalAI, en contraste, posee ciertas características que permite la evaluación de sistemas IA más complejos [6]:

1. **Evaluación humana** en el ciclo de trabajo de modelos de aprendizaje automático. EvalAI permite la evaluación de modelos de IA mediante la interacción con humanos en tiempo real. Esto es crucial para tareas complejas donde la evaluación automatizada no es suficiente, como la generación de lenguaje natural o el diálogo.

2. Capacidad de **ejecutar el código del usuario en un entorno dinámico** para respaldar la evaluación de agentes interactivos. Esto es fundamental

para tareas donde los agentes interactúan con un entorno cambiante, como el aprendizaje por refuerzo.

3. Admite ***pipelines* de evaluación personalizados** compatibles con cualquier lenguaje de programación. Esto permite a los usuarios diseñar y ejecutar sus propias evaluaciones según los requisitos específicos de su tarea o problema.

4. Un **número arbitrario de fases de desafíos y divisiones de conjuntos de datos**. Esto es esencial para competiciones que requieren diferentes configuraciones de evaluación, como pruebas de validación, pruebas de rendimiento y pruebas de desafío.

5. **Evaluación remota** en una piscina de trabajadores privados. Esto es útil para competiciones que requieren recursos computacionales especiales o una evaluación confidencial.

En cuanto a la arquitectura de EvalAI, esta fue diseñada teniendo en cuenta la escalabilidad y portabilidad. La mayoría de los componentes dependen en gran medida de tecnologías de código abierto: Docker, Django, Node.js y PostgreSQL. Las piezas clave de la plataforma son las siguientes [6]:

- **Orquestación**. Se hace uso de los contenedores Docker [7] para ejecutar la infraestructura. Además, se despliegan todos los contenedores en Amazon Elastic Container Service (ECS) que automáticamente escala el clúster para satisfacer las demandas computacionales.

- **Servidores Web**. EvalAI emplea Django [8], un *framework* de desarrollo web de alto nivel que fomenta el desarrollo rápido y limpio de aplicaciones web. Este componente se encarga de acceder y modificar la base de datos mediante API, y de enviar solicitudes de evaluación a una cola.

- **Cola de mensajes**. Son los responsables de dirigir los envíos de usuarios a un grupo de trabajadores para procesar dichos envíos. EvalAI utiliza Amazon Simple Queue Service (SQS) [9] para el bróker de mensajes, que asegura la consistencia y confiabilidad de la cola de mensajes.

- **Nodos de trabajo**. Para cada competición, se crea un grupo de nodos de trabajo dedicados a evaluar envíos del mismo. Estos nodos se generan como contenedores Docker que se ejecutan dentro de Elastic Container Service (ECS) [10], lo que supone múltiples ventajas. En primer lugar, los nodos de trabajo están aislados para evitar conflictos de dependencias entre competiciones. En segundo lugar, el grupo de nodos de trabajo puede escalar independientemente según las demandas de la competición.

- **Almacenamiento**. En EvalAI se utiliza PostgreSQL, un sistema de gestión de bases de datos relacional de código abierto, como almacenamiento de datos principal. Todas las tablas residen en una única base de datos llamada “evalai”.

2.3. CodaLab/ Codabench

CodaLab [11] es una plataforma web de código abierto que permite a investigadores, desarrolladores y científico de datos colaborar, con el objetivo de avanzar en los campos de investigación donde se utiliza el aprendizaje automático y la computación avanzada [12]. Originalmente, CodaLab fue creado en 2013 como una empresa conjunta entre Microsoft y la Universidad de Stanford. La visión era crear un ecosistema para llevar a cabo investigaciones computacionales de manera más eficiente, reproducible y colaborativa, combinando hojas de trabajo (*worksheets*) y competiciones [13].

Las *worksheets* permiten capturar *pipelines* de investigación complejos de manera reproducible y crear lo que se conoce como "executable papers". Esto permite a los investigadores documentar y compartir sus métodos, datos y resultados de una manera que no solo es estática (por ejemplo, a través de un artículo publicado), sino que también permite a otros reproducir y ejecutar el código y los análisis asociados.

Por otro lado, las competiciones reúnen a toda la comunidad científica para abordar problemas de datos y computación más desafiantes en la actualidad. La plataforma permite a los usuarios participar en competiciones y crear sus propios desafíos [14].

A día de hoy, se han organizado más de 2.600 competiciones en Codalab. El sitio web de Codalab cuenta con más de 56.000 usuarios registrados, mientras que el número total de participantes en las competiciones asciende a más de 90.000 personas. Además, se han realizado más de 630.000 envíos a lo largo de todas las competiciones.

La plataforma Codalab Competitions se basa en una arquitectura moderna y flexible que utiliza una variedad de tecnologías para proporcionar funcionalidades robustas y escalables. El sistema se compone de las siguientes piezas claves:

- **Servicio web**. Al igual que EvalAI, Codalab Competitions utiliza Django, que proporciona todas las vistas con las que interactúan los usuarios. Este *framework* se encarga de gestionar todas las entidades de datos del sistema, como usuarios, competiciones y envíos.
- **Cola de mensajes**. En Codalab Competitions, los envíos se evalúan de manera asíncrona, utilizando trabajadores de cómputo. Este proceso sigue un enfoque clásico de productor-consumidor, utilizando RabbitMQ como un bróker de mensajes de y un cliente Celery como una cola de tareas, donde se ejecutan procesos de larga duración de forma asíncrona.
- **Trabajadores**. En Codalab Competitions, los trabajadores se encargan de ejecutar tareas. Algunas de las tareas que se ejecutan incluyen la creación de una competición, la evaluación de una solución presentada por un usuario, el envío de correos electrónicos masivos, volver a ejecutar todas las presentaciones de los usuarios en una fase de la competición y programar tareas.

- **Docker**. Docker se utiliza para gestionar el desarrollo local y desplegar entornos, lo que proporciona un mayor nivel de reproducibilidad.
- **Almacenamiento**. Para almacenar datos de competiciones, presentaciones, datos de entrada, resultados de presentaciones, logotipos, etc., Codalab utiliza servicios de almacenamiento en la nube como Amazon S3 o Microsoft Azure Storage. Estos servicios ofrecen una alta disponibilidad, durabilidad y escalabilidad, lo que garantiza que los datos estén siempre accesibles y seguros. Los datos se pasan típicamente a través de URL con firmas en lugar de transmitir los datos reales. Una URL firmada es una URL que proporciona permisos y tiempo limitados para realizar una solicitud [15]. Esto simplifica la gestión de acceso y garantiza la seguridad de los datos. Además, para la gestión de datos estructurados, como usuarios, competiciones y envíos, CodaLab Competitions utiliza PostgreSQL como su base de datos principal.

Codabench [16] es una plataforma de código abierto que permite organizar pruebas de IA. La plataforma se empezó en 2019 y se puso en producción en 2023. Se trata de una versión mejorada de la plataforma anterior, Codalab Competitions. La novedad de Codabench está en la posibilidad de crear *benchmarks*, que es una prueba de rendimiento o comparativa utilizada para medir el rendimiento de algoritmos de IA. Estos *benchmarks* pueden ser diseñados de manera flexible y reproducible, lo que facilita la evaluación justa y precisa de diferentes algoritmos en diversas tareas de inteligencia artificial. Con Codabench, los usuarios pueden crear y compartir *benchmarks* personalizados, estableciendo protocolos de evaluación específicos y formatos de datos adaptados a sus necesidades.

Actualmente, Codabench tiene más de 3.600 usuarios y más de 500 competiciones creadas. Además, ha habido más de 27.000 presentaciones realizadas en estas competiciones, lo que destaca la significativa actividad y uso de la plataforma.

2.4. Comparativa entre plataformas de creación de competiciones

Las tres plataformas poseen una amplia comunidad activa. Si se realiza una comparación, uno de los puntos a destacar es que Kaggle es de código cerrado, mientras que EvalAI y CodaLab/Codabench son de código abierto. Esta característica hace que las dos últimas plataformas sean más personalizables y adaptables a las necesidades específicas de los usuarios.

Como resultado, ambas plataformas ofrecen la posibilidad de incluir métricas personalizadas para la evaluación de los resultados. Como se ha comentado, algunas tareas de IA requieren el uso de métricas personalizadas para evaluar los resultados. Kaggle, a pesar de su popularidad y amplia base de usuarios, no admite métricas personalizadas, lo que limita su uso para tareas que requieren evaluaciones específicas. En cambio, las otras dos plataformas sí admiten este tipo de métricas.

La diferencia entre EvalAI y Codabench o Kaggle es la capacidad de EvalAI para supervisar manualmente la evaluación de modelos. Se trata de una característica muy útil en casos donde se requiere validación humana de resultados, como pueden ser los algoritmos generativos tipo ChatGPT.

A continuación se presenta una tabla comparativa de las cuatro plataformas.

Características	kaggle	EvalAI	CodaLab Codabench
Comunidad Activa	⋆⋆⋆⋆	⋆⋆	⋆⋆⋆
Código abierto	×	✓	✓
Métricas personalizadas	×	✓	✓
Evaluación humana	×	✓	×

Cuadro 2.1: Comparativa de las plataformas.

Tras analizar las herramientas disponibles, la idea es utilizar las plataformas EvalAI y Codabench para desarrollar la competición. Esta elección se basa en la característica de que ambas plataformas son *open-source* y permiten la incorporación de métricas personalizables. Además, se ha optado por Codabench en lugar de CodaLab Competitions, dado que Codabench representa una versión mejorada de esta última.

Aunque EvalAI destaca por su capacidad para la evaluación humana, esta característica no resulta necesaria para la competición planificada. Esto se debe a que la base de datos que se usará en la competición está completamente anotada, como se detallará en el siguiente capítulo (3). Por lo tanto, no se requiere una supervisión manual de los resultados de los participantes. De esta manera, Codabench, también se presenta como una elección adecuada.

Capítulo 3

Base de datos

En este tercer capítulo se explicará el conjunto de datos utilizado en la competición. Se presentarán las características que posee el conjunto de datos (Sec.3.1) y se explicará el archivo de las anotaciones (Sec.3.2) utilizado para comparar los resultados de los participantes.

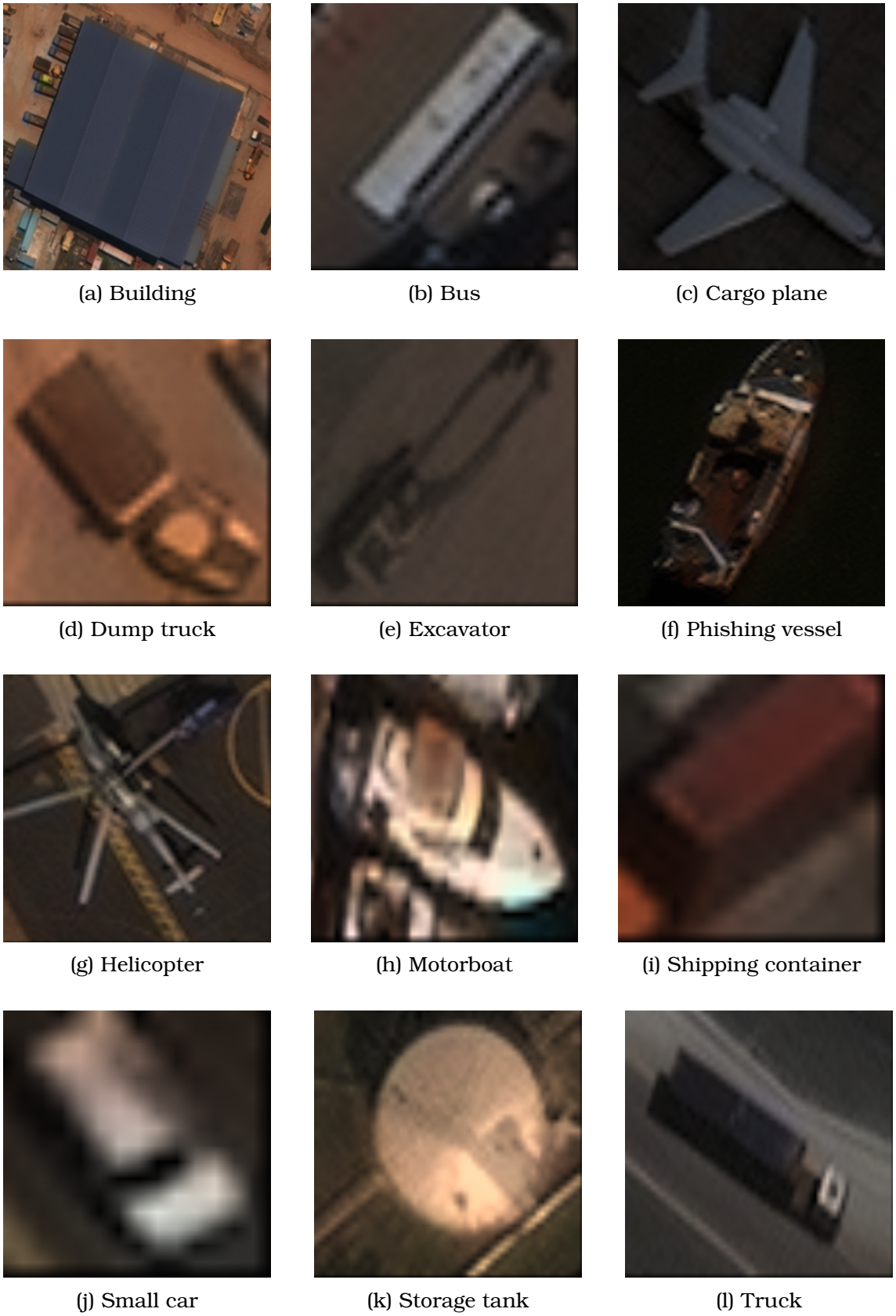
3.1. Características

El conjunto de datos utilizado en la competición se proporciona a los participantes para que puedan desarrollar y entrenar sus modelos de reconocimiento de objetos en imágenes de satélite. Las imágenes provienen del conjunto de datos xView. xView es un gran conjunto de datos de detección de objetos con aproximadamente 1 millón de objetos de 60 clases. Contiene imágenes anotadas manualmente de diferentes escenas de todo el mundo, adquiridas con el satélite WorldView-3 a 0.3 metros de distancia de muestreo del suelo. En total hay 846 imágenes anotadas. Para la competición, se han dividido estas anotaciones en 761 y 85 imágenes para entrenamiento y pruebas respectivamente.

De todas las anotaciones solo se seleccionarán las 12 clases más representadas. Como resultado, se utilizarán 9809 y 1089 imágenes para entrenamiento y prueba respectivamente para la tarea de detección de imágenes. Las imágenes resultantes se redimensionan a 600×600 píxeles.

Luego, para el reconocimiento de imágenes, se recortan las imágenes anteriores utilizando sus recuadros delimitadores anotados para extraer un subconjunto de objetos de interés. En este proceso, cada objeto anotado en las imágenes de detección se convierte en una imagen separada de objeto individual. De este modo, se recopilan 21.377 y 2.635 objetos para el entrenamiento y prueba, respectivamente. Las imágenes resultantes se redimensionan a 224×224 píxeles.

A continuación se muestra una imagen de cada categoría:



(a) Building (b) Bus (c) Cargo plane

(d) Dump truck (e) Excavator (f) Phishing vessel

(g) Helicopter (h) Motorboat (i) Shipping container

(j) Small car (k) Storage tank (l) Truck

Figura 3.1: Clases de la Base de Datos

Como se ha explicado anteriormente, hay 21.377 y 2.635 imágenes para el entrenamiento y prueba, respectivamente. La distribución de estas imágenes en las 12 categorías es la siguiente:

Clase	Cantidad en *train*	Cantidad en *test*
Cargo plane	635	83
Helicopter	70	1
Small car	4290	487
Motorboat	1069	394
Phishing vessel	706	93
Dump truck	1236	122
Excavator	789	57
Building	4689	542
Storage tank	1469	243
Shipping container	1523	66

Cuadro 3.1: Distribución de las imágenes.

3.2. Anotaciones de la base de datos

El objetivo de la competición es generar un archivo JSON que siga la misma estructura que las anotaciones originales. Este archivo JSON permitirá la evaluación automática de las predicciones realizadas por los participantes, comparándolas con las anotaciones reales. El formato del fichero de anotaciones JSON es el siguiente:

- **info**: esta sección proporciona información general sobre el conjunto de datos.
- **images**: aquí se enumeran las imágenes del conjunto de datos con sus detalles.
  - **image_id**: identificador único de la imagen.
  - **filename**: ruta de acceso a la imagen.
  - **width** y **height**: dimensiones de ancho y alto de la imagen.
- **annotations**: esta sección contiene las anotaciones de las imágenes, donde se especifica los siguientes elementos:
  - **image_id**: identificador de la imagen anotada a este objeto.
  - **category_id**: categoría del objeto presente en la imagen.
  - **bbox**: coordenadas del cuadro delimitador que rodea a dicho objeto, en este caso, el tamaño de la imagen.

- **categories**: aquí se enumeran las categorías de objetos presentes en las imágenes con sus detalles:
  - **id**: identificador único de la categoría.
  - **name**: nombre de la categoría.
  - **supercategory**: grupo jerárquico al superior al que pertenece la categoría. Por ejemplo, las categorías "Small car" y "Bus" tienen como supercategoría "Passenger vehicle".

A continuación se muestra un breve fragmento del documento JSON.


```
{
    "info": {
        "description": "DIUx xView 2018",
        "url": "http://xviewdataset.org/",
        "version": "2023"
    },
    "images": {
        "0": {
            "image_id": "3b7de4a3-3b8b-4988-b13a-ba8330cb3880.
               tif",
            "filename": "xview_train/Building/3b7de4a3-3b8b
               -4988-b13a-ba8330cb3880.tif",
            "width": 224,
            "height": 224
        },
        ...
    },
    "annotations": {
        "0": {
            "image_id": "3b7de4a3-3b8b-4988-b13a-ba8330cb3880.
               tif",
            "category_id": "Building",
            "bbox": [
                0,
                0,
                224,
                224
            ]
        },
    ...
    },
    "categories": {
        "0": {
            "id": 13,
            "name": "Cargo plane",
            "supercategory": "Fixed wing aircraft"
        },
```

```
            ...
        }
}
```

Listing 3.1: Documento JSON con las anotaciones del conjunto de entrenamiento

Para cada conjunto de datos (entrenamiento y prueba) hay un archivo de anotaciones. Sin embargo, solo el archivo de anotaciones del conjunto de entrenamiento se hará público.

Capítulo 4

Desarrollo de la competición

En este capítulo se presentará la creación de la competición en las plataformas de EvalAI (Sec.4.1) y Codabench (Sec.4.2), además de una comparación entre ambas herramientas (Sec.4.3).

4.1. EvalAI

Para crear la competición en EvalAI es necesario disponer de una cuenta en la plataforma. EvalAI cuenta con un servidor de producción (`https://eval.ai/`) y un entorno de prueba (`https://staging.eval.ai/`. El servidor de prueba es funcionalmente idéntico al servidor de producción, con la diferencia de que está destinado a realizar pruebas y ajustes. Esto permite a los usuarios experimentar y configurar sus competiciones sin afectar a las competiciones reales en curso. En el presente trabajo se ha optado por el entorno de prueba para explorar y familiarizarse con la plataforma en la organización de competiciones. Es importante tener en cuenta que se necesita una cuenta en el servidor correspondiente para crear la competición; es decir, si se desea crearla en el entorno de prueba, se requiere una cuenta en dicho servidor.

EvalAI proporciona una plantilla inicial en GitHub (`https://github.com/Cloud-CV/EvalAI-Starters`) para iniciar el proceso de creación de competiciones.

4.1.1. Configuración en GitHub

En GitHub, se utiliza el repositorio proporcionado como plantilla para configurar la competición. Para ello, se accede a la página principal del repositorio y, sobre la lista de archivos, se selecciona la opción “Use this template”. Luego, se elige “Create a new repository”.

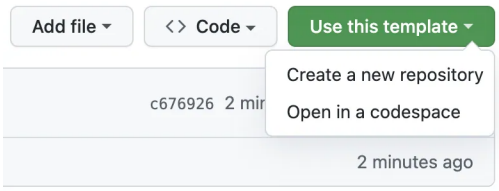


Figura 4.1: Creación del repositorio en GitHub.

Una vez que se crea el repositorio, se debe generar un token de acceso personal de GitHub (Ver A.1).

Una vez que se ha configurado el repositorio de GitHub, se realizará cambios necesarios en los archivos de la plantilla original. La estructura de directorios original contiene varios archivos, de los cuales solo se conservarán los elementos necesarios para la competición, eliminando aquellos que no sean relevantes. A continuación se muestra la estructura de directorios en GitHub:

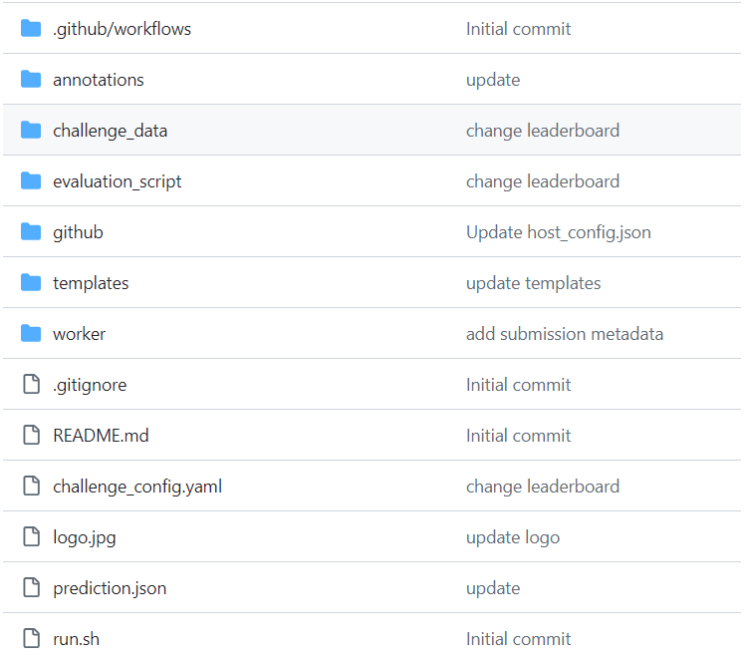


Figura 4.2: Estructura de archivos de la competición en EvalAI.

Tal y como se observa en la figura 4.2, se distinguen varios elementos en la estructura de directorios. Dichos elementos son los siguientes:

- **logo.jpg**: logotipo de la competición.
- **competition.yaml**: archivo de configuración de la competición (Ver 4.1.3).
- **prediction.json**: archivo de prueba utilizado para validar el programa de puntuación, simulando el envío de un participante durante la competición.

- **.github**: carpeta que contiene las configuraciones de flujo de trabajo de GitHub.
- **github**: carpeta que contiene las configuraciones de github, las dependencias necesarias y el archivo de validación y creación de la competición (`challenge_processing_script.py`) necesario para la subida de la competición (Ver 4.1.6). Es importante destacar que solo se necesita realizar cambios en el archivo `host_config.json` (Ver 4.1.2).
- **templates**: carpeta que contiene los archivos HTML utilizados para proporcionar descripciones detalladas de la competición y que se visualizan en el servidor de EvalAI.
- **annotations**: carpeta que almacena los archivos JSON con las anotaciones utilizados para comparar las predicciones de los participantes.
- **evaluation_script**: carpeta que contiene el programa de puntuación (`main.py`) utilizado para evaluar las predicciones de los participantes (Ver 4.1.5).
- **challenge_data**: carpeta que contiene los *scripts* para probar el programa de puntuación localmente. Concretamente, contiene el programa de puntuación, igual que el de la carpeta `evaluation_script`.
- **worker**: carpeta que contiene archivos para probar el programa de evaluación localmente. Concretamente, contiene el archivo Python para correr el archivo de evaluación localmente.

En los siguientes apartados (Sec.4.1.2 - 4.1.5), se explicarán los archivos que se modificarán.

4.1.2. Archivo de configuración del host

El archivo de configuración del host (`host_config.json`) se encuentra ubicado en la carpeta `github`. Este archivo contiene los parámetros de configuración específicos del host. El archivo JSON se muestra de la siguiente manera:

```
{
        "token": "<token>",
        "team_pk": "1032",
        "evalai_host_url": "https://staging.eval.ai"
}
```

Listing 4.1: Documento host_config.json

A continuación, se explicarán los parámetros del documento JSON:

- **token**: se trata del token de autenticación del usuario de EvalAI. Para obtener el token, se debe acceder a la cuenta de EvalAI (servidor de producción o de prueba, dependiendo de donde se esté creando la competición). Una vez iniciada la sesión, en la página de perfil, se debe hacer clic en "Get your Auth Token" y luego copiar el token generado.

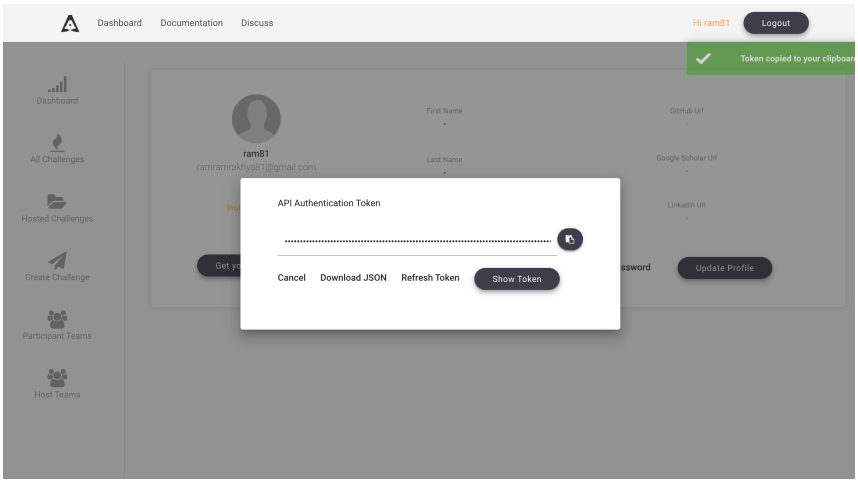


Figura 4.3: Token de Autenticación.

- **team_pk**: se trata del ID del equipo host de EvalAI. Si no se dispone de ningún equipo, se procede a su creación desde EvalAI. Para ello, se accede a la pestaña de "Host Teams" y en el apartado de "Create a New Team" crear un nuevo equipo.

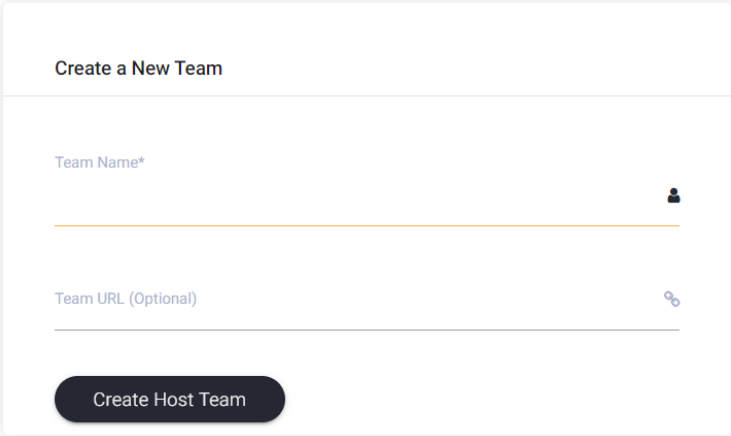


Figura 4.4: Creación de un nuevo equipo host.

Una vez que se dispone de un equipo, se procede a copiar el ID del equipo creado.

- **evalai_host_url**: se usa `https://eval.ai` para el servidor de producción y `https://staging.eval.ai` para el servidor de prueba. En el caso del proyecto, como se va a crear la competición en el servidor de prueba, se selecciona la segunda opción.

4.1.3. Archivo de configuración YAML

El archivo de configuración (`challenge_config.yaml`) es el archivo que contiene los parámetros y ajustes específicos que definen las características y comportamiento de la competición. Este archivo es fundamental para configurar aspectos como las reglas de la competición, las métricas de evaluación, los conjuntos de datos utilizados, las fechas de inicio y fin, entre otros detalles relevantes para la competición. En el presente trabajo se han definido los siguientes parámetros:

- **title**: título de la competición.
- **short_description**: una pequeña descripción de la competición, preferiblemente, menos de 140 caracteres.
- **description**: ruta del archivo HTML donde se especifica una descripción más completa de la competición.
- **evaluation_details**: ruta del archivo HTML donde se describen los detalles de la competición.
- **terms_and_conditions**: ruta del archivo HTML donde se describen los términos y condiciones de la competición.
- **image**: ruta del logotipo de la competición. La imagen debe tener formato `jpg`, `jpeg` o `png`.
- **submission_guidelines**: ruta del archivo donde se describen las pautas de envío de la competición.
- **evaluation_script**: ruta del archivo Python que define cómo se evaluarán los resultados de los participantes.
- **leaderboard_description**: una descripción de la tabla de clasificación.
- **remote_evaluation**: se especifica si la evaluación se hará en una máquina remota. En el presente trabajo, se establece en `False`, indicando que la evaluación no se realizará en una máquina remota.
- **is_docker_based**: se especifica si la competición está basado en Docker. En el presente trabajo, se establece en `False`, indicando que no se basa en Docker.
- **start_date**: fecha y hora de inicio de la competición. El formato se da en formato YYYY-MM-DD HH:MM:SS, siguiendo el estándar de la zona horaria UTC.
- **end_date**: fecha y hora de finalización de la competición. El formato se da en formato YYYY-MM-DD HH:MM:SS, siguiendo el estándar de la zona horaria UTC.
- **published**: se especifica si la competición será pública o no. En el presente trabajo, se establece `True`, indicando que la competición será pública.
- **leaderboard**: se trata de la tabla de clasificación de la competición. Presenta los siguientes campos:

- **id**: un entero positivo que indica el ID de la tabla de clasificación.
- **schema**: contiene información sobre las filas de la tabla de clasificación. Presenta los siguientes campos:
  - **labels**: una lista de las columnas que conforman la tabla. En este caso particular, se han definido tres columnas: accuracy, precision y recall.
  - **default_order_by**: establece el orden por defecto de la tabla de clasificación. En el presente trabajo se ha definido el accuracy como criterio de orden.

- **challenge_phases**: se tratan de las fases de la competición. En este caso particular, solo habrá una única fase, que será la de clasificar las imágenes del conjunto de prueba. Presenta los siguientes subcampos.
  - **id**: ID de la fase.
  - **name**: nombre de la fase.
  - **description**: ruta al archivo HTML con una descripción de la fase.
  - **leaderboard_public**: un valor booleano (`True` o `False`) que especifica si la tabla de clasificación es pública o privada. En el presente trabajo se establece como `True`, que indica que la tabla es pública.
  - **is_public**: un valor booleano (`True` o `False`) que especifica si la fase se muestra o no a los participantes. En este caso se ha establecido el valor a `True`, indicando que la fase se muestra a los participantes.
  - **is_submission_public**: un valor booleano (`True` o `False`) que especifica si los envíos de los participantes son públicos o no. En este caso, se ha establecido el valor a `True`, indicando que los envíos de los participantes serán públicos.
  - **start_date**: fecha y hora de inicio de la fase. El formato se da en formato YYYY-MM-DD HH:MM:SS, siguiendo el estándar de la zona horaria UTC.
  - **end_date**: fecha y hora de finalización de la fase. El formato se da en formato YYYY-MM-DD HH:MM:SS, siguiendo el estándar de la zona horaria UTC.
  - **test_annotation_file**: ruta al archivo donde se encuentran las soluciones con las anotaciones correspondientes del conjunto de prueba.
  - **codename**: identificador único para cada fase de la competición. Este valor se utiliza para asignar los resultados devueltos por el programa de evaluación a una fase particular de la competición. Este nombre debe coincidir con el nombre en clave especificado en el programa de evaluación.
  - **max_submissions_per_day**: número máximo de envíos al día.

- **max_submissions_per_month**: número máximo de envíos al mes.
- **max_submissions**: número máximo de envíos.
- **allowed_submission_file_types**: lista de tipos de archivos permitidos en los envíos. En el presente trabajo, solo se admite archivos con extensión `.json`.
- **submission_meta_attributes**: metadatos personalizados que los participantes pueden agregar a sus envíos. En este caso, se creará el metadato de "Model", donde el usuario deberá especificar el modelo que ha usado para resolver la tarea de la competición. Presenta los siguientes subcampos:
  - **name**: nombre del metadato.
  - **description**: pequeña descripción del metadato.
  - **required**: un valor booleano (`True` o `False`) que especifica si el metadato es obligatorio. En este caso, se establece a `True` para que aparezca el modelo utilizado en la tabla de clasificación.

- **dataset_splits**: son divisiones de datos que definen el subconjunto de pruebas en el que se evaluarán los envíos. En este caso, solo se definirá un subconjunto que será el subconjunto de *test*. Presenta los siguientes subcampos:
  - **id**: número entero que representa el ID del subconjunto de datos.
  - **name**: nombre del subconjunto de datos.
  - **codename**: identificador único para cada división del conjunto de datos. Hay que tener en cuenta que dicho identificador se utiliza para asignar los resultados devueltos por el programa de evaluación a una división del conjunto de datos particular. Por lo tanto, hay que asegurarse que no haya una división de conjunto de datos que tengan el mismo nombre. Además, es importante que el *codename* de la división del conjunto de datos coincida con lo especificado en el programa de evaluación.
- **challenge_phase_splits**: una división de fase de competición es una relación entre una fase específica de la competición (`challege_phase`) y las divisiones del conjunto de datos asociadas (`dataset_splits`). Esta relación se utiliza para establecer privacidad de los envíos (públicos o privados) para distintas divisiones del conjunto de datos en diferentes fases del desafío. Presenta los siguientes subcampos:
  - **challenge_phase_id**: ID del `challenge_phase`.
  - **leaderboard_id**: ID del `leaderboard`.
  - **dataset_split_id**: ID del `dataset_split`
  - **visibility**: establece la visibilidad de las métricas para esta división de fase de competición en la tabla de clasificación. Presenta tres posibles

valores:

- **1**: visible para el creador de la competición.
- **2**: visible para el creador y participante que realizó el envío de resultados.
- **3**: visible para todos los participantes de la tabla de clasificación. En el presente trabajo, se ha optado por esta opción.

- **leaderboard_decimal_precision**: entero positivo utilizado para variar la posición decimal de la tabla de clasificación. En este caso, se establecerá en 2.
- **is_leaderboard_order_descending**: un valor booleano que especifica si el orden de la tabla de clasificación es descendente. Para este caso, se asignará el valor `True`.

4.1.4. Archivos HTML

Los archivos HTML se utilizan para proporcionar información detallada sobre la competición. Dichos archivos se encuentran en el directorio `templates`. En la competición realizada, se definen los siguientes archivos:

- **description.html**: contiene la descripción general de la competición.
- **challenge_phase_1_description.html**: contiene la descripción de la fase de la competición. En este caso, de la única fase que hay, la del test.
- **evaluation_details.html**: contiene información sobre cómo van a ser evaluados los envíos de los participantes.
- **submission_guidelines.html**: contiene información sobre cómo realizar los envíos a la competición.
- **terms_and_conditions.html**: contiene información sobre los términos y condiciones de la competición.

4.1.5. Programa de puntuación evaluación

El programa de puntuación o evaluación es el archivo Python que se ejecuta para determinar las puntuaciones de los envíos de los participantes. La lógica para evaluar el envío de los participantes es personalizable.

A continuación se muestra el programa de puntuación implementado para evaluar los resultados de los participantes.

```
import json
from sklearn.metrics import accuracy_score, precision_score,
 ↪  recall_score


def load_json(filename):
```

```
    with open(filename, "r") as file:
        data = json.load(file)
    return data


def evaluate(test_annotation_file, user_submission_file,
↪   phase_codename, **kwargs):
    print("Starting Evaluation.....")
    submission_metadata =
    ↪   kwargs['submission_metadata']['submission_metadata']
    print(submission_metadata)
    output = {}
    if phase_codename == "test":
        print("Evaluating for Test Phase")
        # TRUE DATA
        json_true = load_json(test_annotation_file)
        sorted_true = sorted(json_true['annotations'].values(),
        ↪   key=lambda x: x['image_id'])
        ytrue = [image['category_id'] for image in sorted_true]

        # PREDICTED DATA
        json_pred = load_json(user_submission_file)
        sorted_pred = sorted(json_pred['annotations'].values(),
        ↪   key=lambda x: x['image_id'])
        ypred = [image['category_id'] for image in sorted_pred]

        # METRICS
        accuracy = accuracy_score(ytrue, ypred)
        precision = precision_score(ytrue, ypred,
        ↪   average='macro', zero_division='warn')
        recall = recall_score(ytrue, ypred, average='macro',
        ↪   zero_division='warn')

        print("accuracy:", accuracy)

        # LEADERBOARD
        output['result'] = [
            {
                "test_split": {
                    "Model": [item['value'] for item in
                    ↪   submission_metadata if item['name'] ==
                    ↪   'model'][0],
                    "Accuracy": accuracy,
                    "Precision": precision,
                    "Recall": recall
                }
            }
```

```
        ]

        # To display the results in the result file
        output["submission_result"] = output["result"][0]
        print("Completed evaluation for Test Phase")
    return output
```

Este programa de evaluación presenta dos funciones: `load_json` y `evaluate`. El primero es una función que recibe como argumento el nombre de un archivo que contiene datos en formato JSON. Esta función carga los datos desde el archivo JSON especificado y los devuelve como una estructura de datos en Python.

La función `evaluate` es la función que se utiliza para evaluar los envíos de los participantes. Dicha función recibe tres argumentos:

- **test_annotation_file**: representa la ruta local al archivo de las soluciones.
- **user_annotation_file**: representa la ruta local del archivo enviado por el usuario para una fase particular de la competición.
- **phase_codename**: es el *codename* de la fase de la competición. Esto se pasa como argumento para que el programa pueda tomar acciones según la fase de la competición.

La función lee los dos archivos JSON, el de referencia (con las soluciones reales) y el proporcionado por el usuario (con las predicciones) y los compara. Para la comparación se centra en la clave "annotations", en el campo "category_id", que contiene la clase asociada al objeto de la imagen. A partir de esto, se calculan las métricas (*accuracy*, *precision* y *recall*). Una vez realizado todo el cálculo, la función devuelve un resultado, que se utiliza para completar la tabla de clasificación. Tal y como aparece en el código, el resultado se almacena en una variable denominada `output`. Presenta las siguientes características:

- Presenta una clave denominada `result`, que es una lista que contiene entradas por división del conjunto de datos que está disponible para la fase de la competición en consideración. En este caso particular, solo hay una fase (test) y un conjunto de datos (test_split).
- Cada entrada de la lista es un diccionario que tiene como clave el *codename* de la correspondiente división de datos. En este caso particular, `test_split`.
- Cada uno de los diccionarios contiene varias claves que se muestran en la tabla de clasificación. En este caso, *Accuracy*, *Precision* y *Recall*.

Prueba del programa de puntuación en local

Antes de probar el programa de evaluación en el servidor, se puede probar el programa localmente. Para ello, se usarán los directorios `annotations`, `challenge_data` y `worker`. Las instrucciones para la ejecución son las siguientes:

1. Copiar todos los archivos de la carpeta `evaluation_script` al directorio

challenge_data/challenge_1. En el caso de la competición realizada, se deben copiar los archivos __init__.py y main.py, siendo este el programa de puntuación.

2. En el archivo worker/run.py, editar el *codename* de la fase de la competición (challenge_phase), el nombre del archivo de datos de referencia (annotation_file_path), el nombre del archivo de presentación de un usuario (user_submission_file_path) y los metadatos en los envíos (submission_metadata). En el caso de la competición, solo se probará la única fase que hay ("test") y los nombres y metadatos a utilizar serán los siguientes:
   - **challenge_phase**: "test".
   - **annotation_file_path**: "/annotations/xview_ann_test.json"
   - **user_submission_file_path**: "/prediction.json".
   - **submission_metadata**: { 'submission_metadata': {['name': 'model', 'type': 'text', 'value': 'VGG19', 'required': True, '$$hashKey': 'object:46', 'description': 'Model name']}
3. Ejecutar el comando python -m worker.run desde el directorio raíz, es decir, donde se encuentran los directorios annotations, challenge_data y worker. Al ejecutar este comando con éxito, se verificará que el programa de evaluación funcione correctamente localmente y, por ende, también funcione de manera adecuada en el servidor.

4.1.6. Subida de la competición al servidor

Una vez que se hayan realizado las modificaciones de los archivos de la competición, se usará de nuevo la plataforma GitHub para subir la competición al servidor de EvalAI.

Para este propósito, se debe crear otra rama con el nombre challenge bifurcado de la rama master en el repositorio. Solo los cambios en la rama challenge se sincronizarán con la competición en la plataforma de EvalAI. Por tanto, es fundamental integrar las modificaciones en la rama challenge. Para automatizar este proceso, se utilizará el flujo de trabajo de GitHub, el cual se activa mediante un *push* o *pull-request* hacia la rama challenge. Posteriormente, se esperará a que la compilación se complete con éxito. La compilación consiste en la validación y actualización de la competición. El proceso implica varias etapas:

- **Dependencias**: se instalan todas las dependencias necesarias en el archivo github/requirements.txt.
- **Validación**: se ejecuta el programa (github/challenge_processing_script.py) para validar la configuración de la competición.
- **Creación o actualización de la competición**: si la validación es exitosa, entonces se crea o se actualiza la competición en EvalAI.

Los registros de la compilación pueden visualizarse de la siguiente manera:

1. En la página principal del repositorio, seleccionar “Actions”.

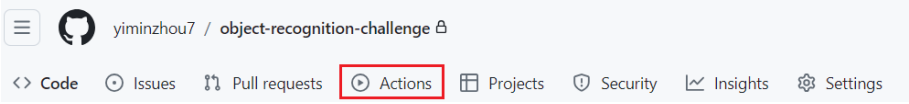


Figura 4.5: *GitHub Actions*.

2. En la barra derecha, hacer clic en el flujo de trabajo que se desee visualizar.

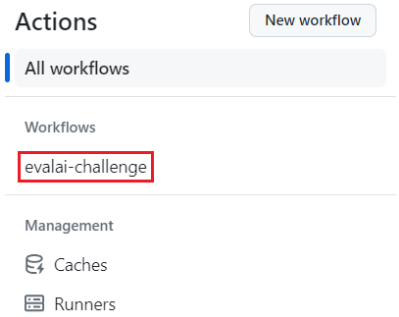


Figura 4.6: *GitHub workflows*.

3. Desde la lista de ejecuciones del flujo de trabajo, hacer clic en el nombre de la ejecución para ver el resumen de la ejecución del flujo de trabajo.
4. En la sección de “Jobs”, hacer clic en el trabajo que se quiera visualizar. En el caso de la competición, solo hay uno: “build”.

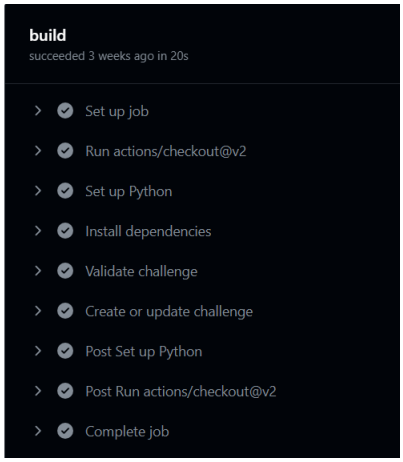


Figura 4.7: *GitHub job*.

Una vez terminada la compilación, si la configuración de la competición contiene errores, entonces se abrirá automáticamente un `issue` en el repositorio con los detalles de los errores encontrados. En caso contrario, la competición se creará exitosamente en el servidor de EvalAI.

Para visualizar la competición, es necesario acceder al servidor de EvalAI (ya sea en producción o en entorno de prueba, dependiendo de dónde esté alojada). Una vez dentro de la cuenta, hay que dirigirse a la sección de desafíos alojados (*Hosted challenges*). Para que el desafío esté disponible públicamente, el administrador de EvalAI lo tiene que aprobar. Esta aprobación se solicita mediante el envío de una solicitud al administrador.

Antes de enviar la solicitud de aprobación, es necesario realizar pruebas con la competición creada. Esto implica que el creador de la competición debe realizar una prueba de carga de archivo, similar a la que haría un participante. Para ello, se puede subir el archivo (`prediction.json`) que se ha utilizado para probar el programa de puntuación localmente en el apartado 4.1.5. Para subir el archivo, hay que dirigirse a la sección "Submit" de la competición. Una vez cargado el archivo, se podrán revisar los archivos subidos en la sección "My Submissions". Es importante que el archivo subido tenga el estado (*Status*) como "Finished", lo que indica que la operación se ha completado correctamente.

Una vez completada la verificación, se procede a enviar la solicitud de aprobación. Esto se realiza haciendo clic en "Send Approval Request" y esperando la aprobación del administrador de EvalAI. El tiempo que tarda en aprobarse la competición puede ser entre 3 a 21 días.

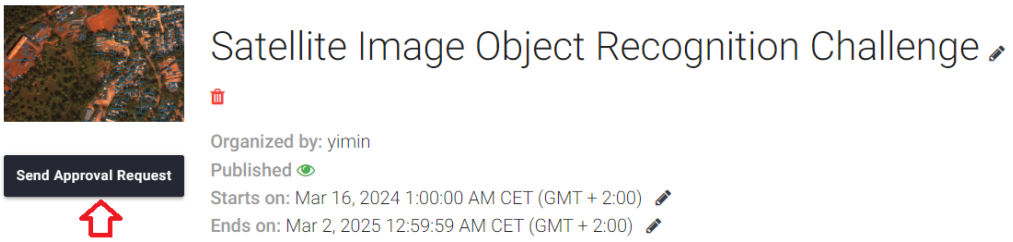


Figura 4.8: Solicitud de aprobación de la competición.

La competición creada se puede ver siguiendo el siguiente enlace: `https://staging.eval.ai/web/challenges/challenge-page/818/overview`.

4.1.7. Modificaciones tras la subida al servidor

EvalAI da la posibilidad de realizar modificaciones tras la subida de la competición al servidor. Para ello, solo es necesario subir los archivos modificados al repositorio de GitHub y hacer un *pull-request* a la rama *challenge*. Sin embargo, es importante tener en cuenta que hay aspectos del archivo de configuración (`challenge_config.yaml`) que no se pueden cambiar después de subir la competición a la plataforma, es decir, tras realizar el *pull-request* a la rama `challenge`. Esto incluye la eliminación de divisiones de fase de la competición (`challenge_phase_splits`).

4.1.8. Participación

Para poder participar en la competición, se debe tener una cuenta en EvalAI, en este caso, en el entorno de prueba. Es importante destacar que la cuenta del servidor de producción no es válida para participar en la competición, ya que esta está creada en el servidor de prueba.

Una vez que se tenga una cuenta en el servidor de prueba, se debe dirigir a la sección "Participate" y seleccionar el equipo correspondiente en "My Participant Teams" para inscribirse en la competición. Si no se tiene un equipo, se puede crear en la sección "Create New Team".

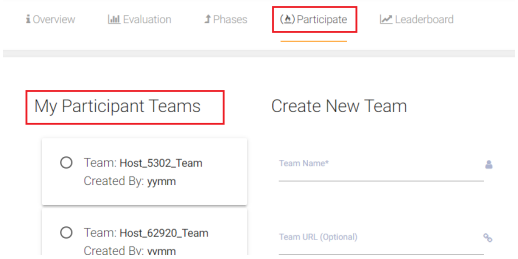


Figura 4.9: Registro en la competición.

Para realizar el envío de los resultados, los participantes deben acceder a la sección de "Submit" y posteriormente dirigirse a "Make Submission". A continuación, se debe seleccionar la fase correspondiente para realizar el envío, la cual en este caso solo hay una posible (*Test Phase*) y elegir el tipo de envío, que en este caso "Upload file". El archivo que debe subirse es el archivo JSON con las predicciones. Además, se debe especificar el nombre del modelo y si se quiere que la puntuación se haga pública (en la tabla de clasificación).

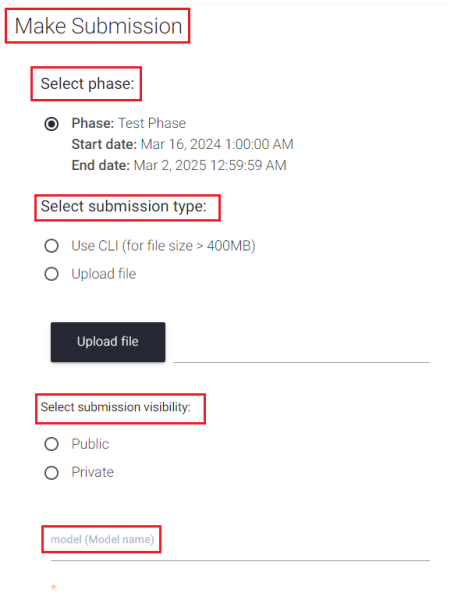


Figura 4.10: Subida del archivo de predicciones.

Una vez que el archivo ha sido subido, es necesario esperar a que la plataforma evalúe los resultados. En el apartado “My Submissions” se puede ver el archivo subido. El estado (*Status*) debe indicar “Finished”. El tiempo de evaluación de los resultados es aproximadamente 3 a 5 minutos.

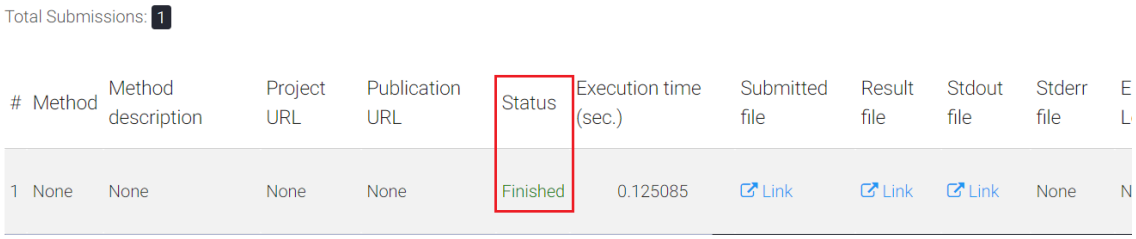


Figura 4.11: Estado del envío en EvalAI.

Una vez que la evaluación haya concluido, se puede ver los resultados obtenidos en la tabla de clasificación en el apartado “Leaderboard”.

Figura 4.12: Tabla de clasificación de EvalAI.

Para visualizar los metadatos introducidos en el envío, en este caso, el nombre del modelo, se debe hacer clic en “View” en la columna de “Meta Attributes”.

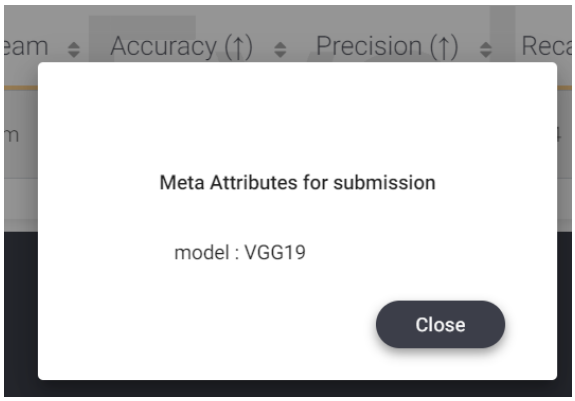


Figura 4.13: Metadatos del envío.

4.2. Codabench

Para crear una competición en Codabench, es necesario disponer de una cuenta en la plataforma (`https://www.codabench.org/`). La competición puede ser

creada a través de la interfaz en línea de Codabench o mediante la carga de un archivo comprimido en formato `.zip` en la plataforma. Para este propósito, se ha elegido la segunda alternativa. A continuación se presenta la estructura del archivo `.zip`

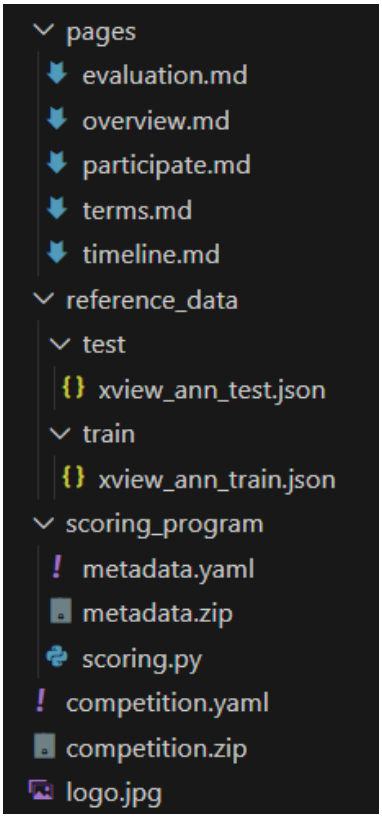


Figura 4.14: Estructura del paquete ZIP de la competición en Codabench

Tal y como se observa en la figura 4.14, se distinguen distintos componentes dentro de la estructura del archivo comprimido. Dichos elementos son los siguientes:

- **logo.jpg**: logotipo de la competición.
- **competition.yaml**: archivo de configuración de la competición (Ver 4.2.1).
- **pages**: carpeta que contiene los archivos Markdown que se usan para definir las páginas de la competición.
- **reference_data**: carpeta que almacena los archivos JSON con las anotaciones utilizados para comparar las predicciones de los participantes.
- **scoring_program**: carpeta que contiene el programa de evaluación (`scoring.py`) y un archivo (`metadata.yaml`) que contiene la clave para ejecutar el programa de puntuación (Ver 4.2.3).

En los siguientes apartados (4.2.1 - 4.2.3), se explicarán en detalle los archivos mencionados para crear la competición.

4.2.1. Archivo de configuración YAML

El archivo de configuración (`competition.yaml`) es el archivo más importante del paquete. En este él se determina la estructura y el diseño de la competición, junto con detalles adicionales. En el archivo de configuración de este presente trabajo se han definido las siguientes opciones:

- **title**: título de la competición.
- **description**: pequeña descripción de la competición.
- **image**: una imagen de la competición.
- **registration_auto_approve**: define si las solicitudes de participación se aprueban automáticamente, es decir, no requiere aprobación del administrador de la competición. En el presente trabajo se le ha puesto el valor de `True`.
- **docker_image**: se trata de la imagen Docker que se utilizará en la competición para correr los programas necesarios, en este caso, el programa de puntuación. La imagen que se ha utilizado es `codalab/codalab-legacy:py39`, que es una imagen básica que usa python 3.9 incluye paquetes como numpy sklearn, etc.
- **terms**: información de los términos de participación.
- **pages**: los apartados de la competición. En el trabajo se han puesto los apartados de "Overview", "Evaluation", "Participation" y "Timeline".
- **tasks**: las tareas a realizar en la competición. En este caso particular, solo hay una tarea: la clasificación de imágenes del conjunto de prueba según su respectiva clase. Aquí se debe especificar:
  - **index**: índice de la tarea.
  - **name**: nombre de la tarea.
  - **scoring_program**: ruta donde se encuentra el programa de puntuaciones utilizado para evaluar los resultados de los participantes.
  - **reference_data**: ruta donde se encuentran las soluciones con las anotaciones correspondientes del conjunto de prueba.
- **phases**: las fases de la competición. En este caso particular, solo habrá una fase, que será la de clasificar las imágenes del conjunto de prueba. Aquí se debe especificar:
  - **name**: nombre de la fase de la fase.
  - **start**: fecha de comienzo de la competición.
  - **end**: fecha final de la fase. Si no se especifica, entonces la competición no tiene fecha final.
  - **max_submissions_per_day**: número máximo de envíos al día.
  - **max_submissions**: número máximo de envíos totales.

- **tasks**: se trata de un *array* de números que hacen referencia al índice de las tareas para esta fase.
- **leaderboard**: la tabla clasificatoria con los resultados de los participantes. Se debe especificar lo siguiente:
  - **title**: título de la tabla clasificatoria.
  - **key**: clave para que el programa de puntuación escriba.
  - **submission_rule**: regla que establece el comportamiento de la tabla de clasificación con respecto a nuevos envíos. Puede tener los siguientes valores:
    - **Add**: solo permite agregar un envío.
    - **Add_And_Delete**: permite a los usuarios agregar un único envío a la tabla y eliminarlo.
    - **Add_And_Delete_Multiple**: permita a los usuarios agregar varios envíos a la tabla y eliminarlos. En este caso se elegirá esta opción, pues le da más flexibilidad al usuario.
    - **Force_Last**: fuerza solo el último envío.
    - **Force_Latest_Multiple**: fuerza el último envío para agregarlo a la tabla de clasificación (múltiple). La tabla también contiene los envíos anteriores.
    - **Force_Best**: fuerza solo el mejor envío a la tabla de clasificación.
  - **columns**: se refiere al conjunto de columnas que conforman la tabla. En este caso particular, se han definido tres columnas: *accuracy*, *precision* y *recall*.

- **fact_sheet**: un JSON con los metadatos de cada envío. En este caso, se creará el metadato de "Model", donde el usuario deberá especificar el modelo que ha usado para resolver la tarea de la competición. Presenta los siguientes subcampos:
  - **key**: clave del metadato.
  - **type**: la manera con la que el participante inserta el metadato. Puede ser:
    - **"checkbox"**: casilla de verificación `True` o `False`. En este caso se necesita añadir el subcampo `selection: [true, false]`.
    - **"text"**: cuadro de texto donde el participante escribe una respuesta. En este caso, se necesita añadir el subcampo `selection: ""`.
    - **"select"**: menú desplegable.
  - **title**: nombre del metadato.

- **is_required**: valor ("true" o "false") que especifica si es obligatorio introducir el metadato al hacer el envío de resultados. En este caso, se pondrá a "true" para que el participante introduzca el nombre del modelo.
- **is_on_leaderboard**: valor ("true" o "false") que especifica si el metadato se mostrará en la tabla de clasificación. En este caso, se pondrá "true".

4.2.2. Páginas de la competición

Las páginas de la competición se utilizan para proporcionar información detallada sobre la competición. Dichas páginas se encuentran en la carpeta `pages`. En la competición hay tres páginas:

- **overview.md**: proporciona una descripción más completa de la competición.
- **evaluation.md**: contiene información sobre los criterios de evaluación.
- **participate.md**: contiene información sobre cómo participar en la competición.
- **terms.md**: establece las reglas y condiciones específicas que los participantes deben aceptar al unirse a la competición.
- **timeline.md**: presenta información sobre las fechas de inicio y finalización de la competición.

4.2.3. Programa de puntuación o evaluación

El programa de puntuación es el archivo que se ejecuta para determinar las puntuaciones de los envíos de los participantes [17]. Se realiza en función de los resultados de predicción de los participantes con los datos de referencia. Generalmente, es un script como un archivo Python, como es en el caso del presente trabajo. El programa se denomina `scoring.py` y se encuentra en la carpeta "scoring_program".

Este programa se combina con un archivo `metadata.yaml` que tiene una clave `command` que se refiere al comando utilizado para ejecutar el programa de puntuación. En el caso de la competición realizada, se ejecuta el siguiente comando:

```
python3 scoring.py
```

Esto especifica que python3 ejecutará el programa de puntuación.

A continuación se muestra el programa de puntuación usado para determinar las puntuaciones de los envíos de los participantes.

```
import json
import os
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
```

```
def load_json(filename):
    with open(filename, "r") as file:
        data = json.load(file)
    return data


reference_dir = os.path.join('/app/input', 'ref')
prediction_dir = os.path.join('/app/input', 'res')
score_dir = '/app/output'
print('Reading prediction')

# TRUE DATA
json_true = load_json(os.path.join(reference_dir,
                                   'xview_ann_test.json'))
sorted_true = sorted(json_true['annotations'].values(),
                     key=lambda x: x['image_id'])
ytrue = [image['category_id'] for image in sorted_true]

# PREDICTED DATA
files_in_dir = os.listdir(prediction_dir)
json_files = [file for file in files_in_dir
              if file.endswith('.json')]
if len(json_files) == 1:
    json_pred = load_json(os.path.join(prediction_dir,
                                       'prediction.json'))
    sorted_pred = sorted(json_pred['annotations'].values(),
                         key=lambda x: x['image_id'])
    ypred = [image['category_id'] for image in sorted_pred]

    # METRICS
    print('Checking Metrics')
    accuracy = accuracy_score(ytrue, ypred)
    precision = precision_score(ytrue, ypred,
                                average='macro',
                                zero_division='warn')
    recall = recall_score(ytrue, ypred,
                          average='macro',
                          zero_division='warn')
    print('Scores:')
    scores = {
        'accuracy': accuracy,
        'precision': precision,
        'recall': recall
    }
    print(scores)
```

```
    with open(os.path.join(score_dir,
                           'scores.json'), 'w') as score_file:
        score_file.write(json.dumps(scores))

else:
    raise ValueError("Error: There is no json file.")
```

El código proporcionado realiza una comparación entre dos archivos JSON: uno que contiene los datos reales y otro con las predicciones del participante. Se centra específicamente en la clave "annotations", en el campo "category_id", que contiene la clase asociada al objeto de la imagen. Según esta información, se calcula los valores de *accuracy*, *precision* y *recall* de los envíos.

El programa de puntuación genera un archivo `scores.json` que contiene los resultados de cada columna de la tabla de clasificación. Las claves utilizadas en el archivo deben coincidir con las claves de las columnas de la tabla de clasificación definidas en el archivo `competition.yaml`. Después de calcular el envío de un participante, el archivo `scores.json` se ve de la siguiente manera:

```
{
    'accuracy': 0.3199240986717268,
    'precision': 0.1946606400481965,
    'recall': 0.21475122549048142
}
```

Listing 4.2: Documento scores.json generado por el programa de puntuación

4.2.4. Subida de la competición al servidor

Una vez que la configuración de la competición esté finalizada, se procederá a cargarla en el servidor de Codabench. Para este propósito, se requiere la creación de un archivo comprimido (ZIP) que contenga todos los elementos necesarios para la competición, según se detalla en la figura 4.14.

Para cargar la competición en el servidor, se debe iniciar sesión en la cuenta de Codabench, dirigirse a la sección de BenchMark y acceder a "Management".

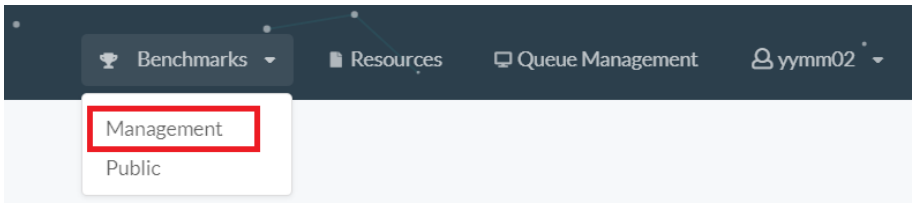


Figura 4.15: Acceso a *Management*.

A continuación, se debe seleccionar "Upload" y cargar el archivo ZIP preparado previamente con todos los archivos necesarios.

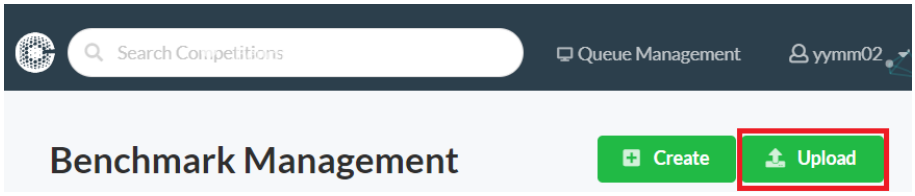


Figura 4.16: Subida de la competición al servidor.

Para visualizar la competición creada, es necesario dirigirse la sección de BenchMark y acceder a "Management" tal y como se muestra en la figura 4.15. Una vez allí, para que la competición esté disponible públicamente, se debe seleccionar la opción "Publish".

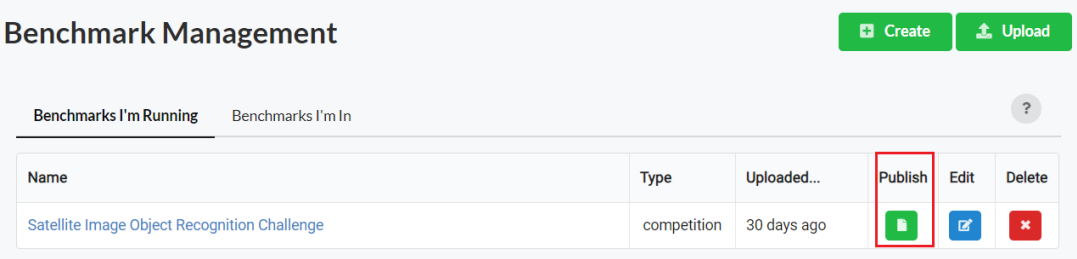


Figura 4.17: Publicar la competición.

Se puede visualizar la competición creada en el siguiente enlace: `https://www.codabench.org/competitions/2332/`

4.2.5. Modificaciones tras la subida al servidor

Al igual que EvalAI, Codabench también da la posibilidad de realizar modificaciones de la competición tras la subida a su servidor. Para ello, se debe acceder al apartado de "Benchmarks" y seleccionar "Management". Allí se mostrarán las competiciones creadas, y se puede seleccionar "Edit" para realizar las modificaciones necesarias.

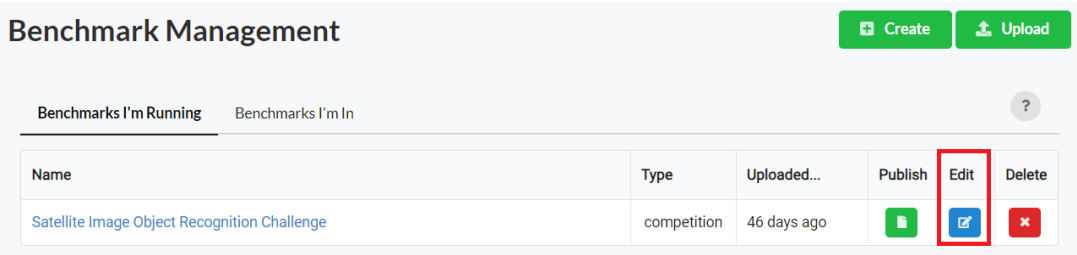


Figura 4.18: Edición de una competición creada.

Aquí solo se puede modificar la configuración de la competición, es decir, los elementos que aparecen en el archivo de configuración (`competition.yaml`).

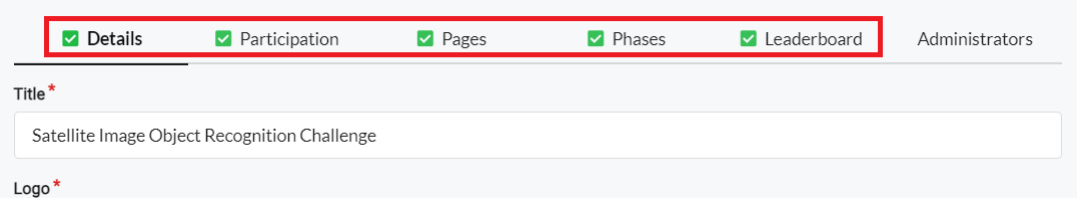


Figura 4.19: Edición de la configuración de la competición.

Si se quiere actualizar el programa de puntuación o los datos de referencia, es necesario seguir unos pasos específicos.

1. Dirigirse al apartado de “Resources” de la cuenta de Codabench.

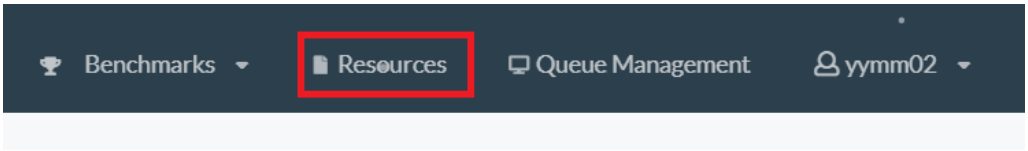


Figura 4.20: Recursos.

2. En la sección de “Datasets and programs”, se procede a cargar los archivos modificados seleccionando la opción “Add Dataset/Program”.

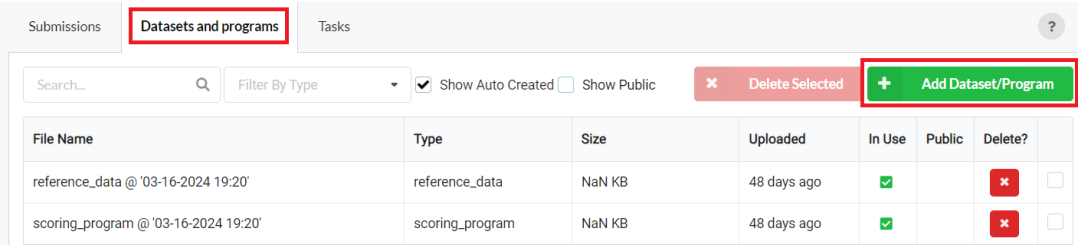


Figura 4.21: Subida de archivos modificados en Codabench.

3. Al añadir los archivos, se puede asignar un nombre y una descripción. En la opción “Type” se debe especificar el tipo de archivo subido (“Reference Data” o “Scoring Program” en este caso). Es importante tener en cuenta que los archivos deben estar comprimidos en formato ZIP antes de la carga.

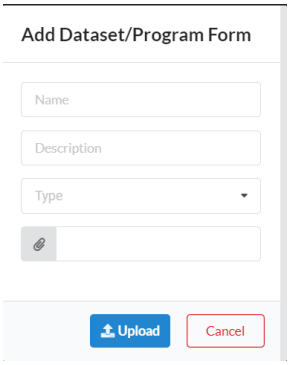


Figura 4.22: Carga de archivos en Codabench.

4. Después, se dirige al apartado de "Tasks", donde se encuentran todas las tareas creadas. Aquí se busca específicamente las tareas relacionadas con la competición en cuestión. Estas tareas se denomina de acuerdo a los nombres especificados en el archivo de configuración (`competition.yaml`). En este caso particular, solo hay una tarea denominada "Object Recognition Testing Task". En esta tarea, se selecciona"Edit".

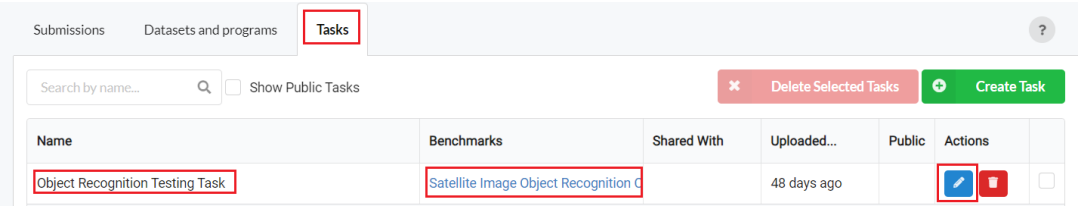


Figura 4.23: Modificación de tareas de la competición.

5. En la sección "Datasets and programs" se actualizan los archivos ("scoring program" y/o "reference data") por los subidos anteriormente.

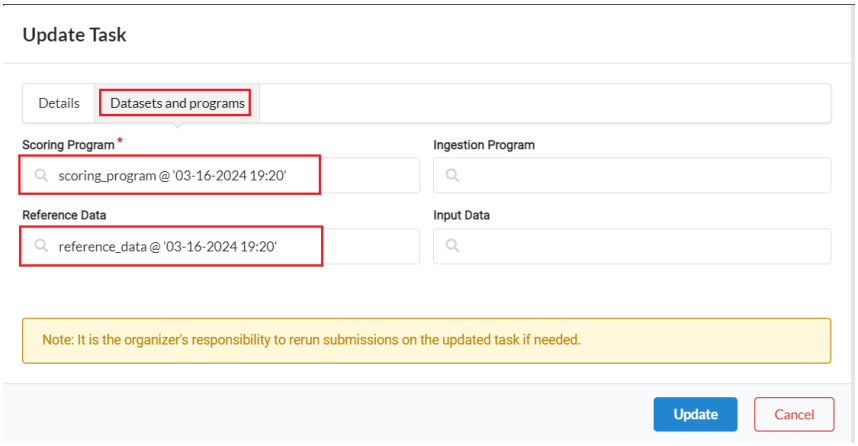


Figura 4.24: Actualización de los archivos en Codabench.

4.2.6. Participación

Para poder participar en la competición, se debe tener una cuenta en Codabench. En la competición, los participantes deben dirigirse a la sección de "My Submissions" y registrarse.

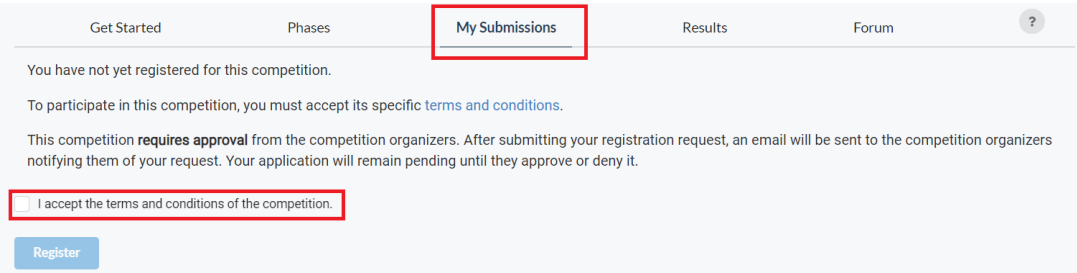


Figura 4.25: Registro en la competición.

Para realizar el envío de los resultados, los participantes deben acceder a la sección de "My Submissions" y luego dirigirse a la opción de "Submission Upload" para cargar el archivo de predicciones. Es fundamental tener en cuenta que el archivo de soluciones en formato JSON debe estar comprimido en un archivo ZIP, ya que Codabench solo acepta este formato para su lectura.

Una vez que el archivo ha sido cargado, es necesario esperar a que la plataforma evalúe los resultados. El estado (*Status*) debe indicar "Finished". Una vez que la evaluación haya concluido, el participante puede añadir su puntuación a la tabla de clasificación haciendo clic en "Add to leaderboard".

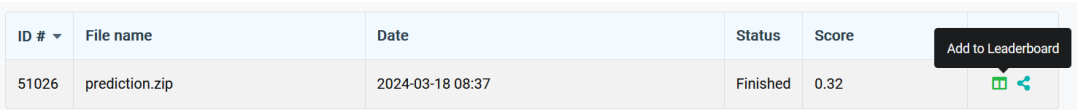


Figura 4.26: Estado del envío e incorporación de puntuaciones a la tabla de clasificación.

Para ver la tabla de clasificación se debe ir al apartado de "Results". La tabla se muestra de la siguiente manera.

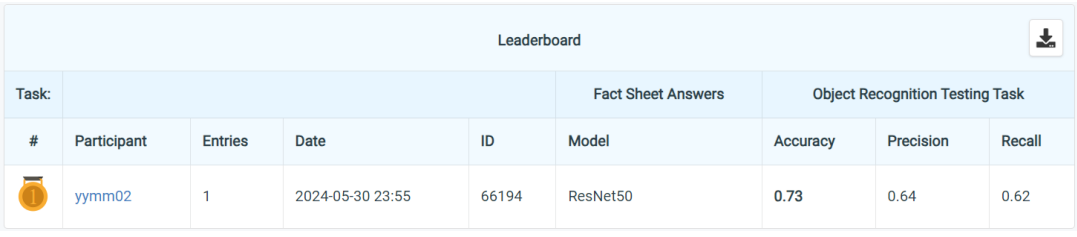


Figura 4.27: Tabla de clasificación de Codabench.

4.3. Comparación de EvalAI y Codabench

4.3.1. A nivel del administrador

A la hora de crear la competición, las dos herramientas (EvalAI y Codabench) necesitan modificar varios archivos, incluyendo los de configuración y los programas de puntuación. Sin embargo, como se ha visto en los apartados anteriores, EvalAI necesita modificar una cantidad mayor de archivos que Codabench. Por ejemplo, en el inicio de la competición, EvalAI requiere la modificación de archivos de configuración en GitHub, mientras que Codabench solo necesita ajustes en el archivo de configuración, los archivos Markdown y el programa de puntuación.

Otro punto a destacar es que EvalAI proporciona herramientas para verificar y validar la configuración de la competición antes de su carga a la plataforma. Utiliza *workflows* de GitHub para realizar pruebas automatizadas y comprobar que la competición se ejecuta correctamente. En cambio, Codabench no proporciona este tipo de comprobaciones. Además, EvalAI permite evaluar el programa de puntuación de manera local antes de subirlo al GitHub, como se ha visto en el apartado 4.1.5.

En caso de que sea necesario modificar algún archivo después de crear y cargar la competición en la plataforma, ambas herramientas permiten dicha modificación. Como se ha visto en las secciones 4.1.7 y 4.2.5, las modificaciones en EvalAI son más sencillas que en Codabench, pues solo es necesario subir los cambios a GitHub y a la rama correspondiente. En cambio, en el caso de Codabench, el procedimiento es distinto si se quiere modificar el archivo de configuración y el programa de puntuación o los datos de referencia. Por otro lado, EvalAI tiene algunas restricciones, pues hay elementos de la configuración que no se pueden cambiar tras su subida al servidor.

Resumen de las diferencias:

Características	EvalAI	Codabench
Cantidad de archivos a modificar	Más archivos	Menos archivos
Herramientas de validación	Workflows de GitHub	No tiene
Actualización tras subida al servidor	Más sencilla	Más complicada
Archivos modificables tras la subida	Hay restricciones	Más flexible

Cuadro 4.1: Resumen de las diferencias entre EvalAI y Codabench a nivel del administrador

4.3.2. A nivel del participante

A nivel del participante, crear una cuenta en EvalAI y Codabench es gratuito. Ambas plataformas ofrecen servicios de manera gratuita para los usuarios que desean organizar o participar en las competiciones.

Si se habla de las diferencias entre ambas plataformas, una de ellas es el proceso de envío de resultados a la competición. EvalAI permite el envío de resultados en múltiples formatos, en este caso, el formato JSON. En cambio, Codabench solo acepta envíos comprimidos en archivos ZIP, por lo que se necesitaba comprimir el archivo JSON en formato ZIP.

En términos de evaluación de los resultados, Codabench supera a EvalAI en velocidad. En Codabench, las puntuaciones se obtienen al instante, mientras que en EvalAI se requiere esperar aproximadamente 3 a 5 minutos.

Otra de las diferencias es la flexibilidad en la presentación de resultados en la tabla de clasificación. En EvalAI, si se realiza múltiples envíos y estos se hacen públicos, solo se mostrará el mejor envío en la tabla de clasificación. En cambio, Codabench permite el añadido y eliminación de múltiples envíos en la tabla de clasificación, dependiendo de cómo el administrador de la competición lo configure.

Características	**EvalAI**	**Codabench**
Formato de archivos de envío	Múltiples	ZIP
Velocidad de evaluación	3 - 5 minutos	$\sim$ 30 segundos
Número de envíos en la tabla	Flexible	Solo un envío

Cuadro 4.2: Resumen de las diferencias entre EvalAI y Codabench a nivel del participante.

Capítulo 5

Participación en la competición

En este capítulo se presentarán los distintos modelos empleados en la competición creada en las plataformas (EvalAI y Codabench). El objetivo es analizar el rendimiento y la capacidad de dichos modelos para realizar predicciones en el conjunto de datos de evaluación de la competición creada. Se llevará a cabo una revisión teórica de los distintos modelos (Sec.5.1 - 5.5). Posteriormente, se realizará una comparativa de dichos modelos (Sec.5.6).

5.1. VGG19

El modelo VGG19 es una arquitectura de red convolucional (CNN) ampliamente conocida y utilizada en tareas de visión por computadora. Fue propuesta por Karen Simonyan y Adrew Zisserman del equipo de investigación del Grupo Visual de Geometría (VGG) de la Universidad de Oxford en 2014 [18]. Dicho modelo alcanzó un *accuracy* de $92{,}7\,\%$ en el conjunto de datos de ImageNet, que es una de las puntuaciones más altas logradas.

La arquitectura de la red VGG19 se basa en una serie de capas convolucionales seguidas de capas totalmente conectadas para la clasificación de imágenes. Esta arquitectura se caracteriza por su simplicidad, utilizando bloques compuestos por capas convolucionales con filtros de tamaño 3×3. Estos bloques están seguidos por capas de *MaxPooling* para reducir el tamaño de los mapas de activación a la mitad. La parte de clasificación consiste en dos capas densas de 4096 neuronas cada una [19]. El número en VGG19 representa la cantidad de capas en la red.

5.2. ResNet50 y ResNet101

El modelo ResNet (*Residual Networks*) es una arquitectura de red convolucional revolucionó el campo del aprendizaje profundo al permitir el entrenamiento de redes mucho más profundas con un rendimiento mejorado. Fue propuesta por Kaiming He, Xiangyu Zhang, Shaoping Ren y Jian Sun en su artículo “Deep Residual Learning for Image Recognition”, publicado en 2016 [20].

La idea de ResNet es la introducción de bloques residuales. La idea central detrás de los bloques residuales es la utilización de conexiones de salto, también conocidas como conexiones de atajo, que omiten algunas capas intermedias de la red y conectan directamente las activaciones de una capa a capas posteriores. Esto crea lo que se conoce como un "bloque residual". Estos bloques facilitan el entrenamiento de redes más profundas al mitigar el problema del desvanecimiento del gradiente.

ResNet50 y ResNet101 son variantes de la arquitectura ResNet que se diferencian en el número de capas. ResNet50 tiene 50 capas y ResNet101 tiene 101 capas.

5.3. MobileNetV2

El modelo MobileNetV2 es una arquitectura de red convolucional diseñada para aplicaciones con recursos limitados, como dispositivos móviles y sistemas integrados. Fue desarrollada por Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov y Liang-Chieh Chen, y presentada en el artículo "MobileNetV2: Inverted Residuals and Linear Bottlenecks" en 2018 [21].

La arquitectura de MobileNetV2 se basa en dos conceptos principales: bloques residuales invertidos y cuellos de botella lineales. Los bloques residuales invertidos permiten la construcción de redes más profundas y más eficientes mediante la utilización de conexiones de salto, similar a ResNet, pero con una estructura invertida.

Los cuellos de botella lineales son capas convolucionales que reducen la cantidad de parámetros y la carga computacional de la red. Al utilizar activaciones lineales en lugar de las no lineales, el modelo conserva más información y mejora su capacidad para capturar detalles detallados [22].

5.4. DenseNet121

El modelo DenseNet (*Densely Connected Convolutional Networks*) es una arquitectura de red convolucional conocida por su eficiencia y precisión en tareas de clasificación de imágenes. Fue introducida por Gao Huang, Zhuang Liu, Laurens van der Maaten y Kilian Q. Weinberger en el artículo "Densely Connected Convolutional Networks" en 2016 [23].

En el modelo DenseNet, cada capa está conectada directamente a todas las capas subsiguientes dentro de un bloque de red. Por lo tanto, el término DenseNet proviene de "Redes Convolucionales Densamente Conectadas". DenseNet se distingue de otras arquitecturas de redes neuronales convolucionales por dos características clave. En primer lugar, presenta una estructura de bloques densamente conectados, lo que significa que cada capa está conectada a todas las demás capas de forma anticipada. En segundo lugar, incorpora capas de cuello de botella, que reducen la cantidad de parámetros sin reducir la cantidad de funciones aprendidas por la red [24].

DenseNet121 es una variante de la arquitectura DenseNet que consta de 121 capas en total.

5.5. EfficientNetB0

El modelo EfficientNet es una arquitectura de red convolucional conocida por su eficiencia en términos de rendimiento y recursos computacionales. Fue propuesta por Mingxing Tan y Quoc V, investigadores de Google, en el artículo "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" en 2019 [25].

EfficientNet es una red construida sobre un concepto llamado "escalado compuesto". La idea de este concepto es escalar tres dimensiones esenciales de una red neuronal: ancho, profundidad y resolución.

- **Ancho**: se refiere a la cantidad de canales en cada capa de la red. Al aumentar el ancho, mejora la precisión, ya que el modelo captura características más complejas.
- **Profundidad**: se refiere al número total de capas en la red. Los modelos más profundos pueden capturar representaciones de datos más complejas, pero también exigen más recursos computacionales. Por otro lado, los modelos menos profundos son computacionalmente eficientes, pero pueden tener menos precisión.
- **Resolución**: se refiere al tamaño de la imagen de entrada. Las imágenes con mayor resolución proporcionan información más detallada, pero requieren mayor coste computacional. En cambio, las imágenes con menos resolución consumen menos recursos, pero pueden provocar pérdida de detalles.

EfficientNetB0 es el modelo base de la familia EfficientNet. Es un modelo más ligero en términos de parámetros y operaciones, pero aun así ofreciendo un rendimiento competitivo.

5.6. Comparativa de modelos

En esta sección se llevará a cabo una comparación entre los seis modelos utilizados en la competición de reconocimiento de objetos en imágenes de satélite. La comparativa se ha realizado teniendo en cuenta las siguientes características:

- **Accuracy**
- **Precision**
- **Recall**
- **Tiempo (s)**: tiempo (en segundos) en procesar las 2.635 imágenes del conjunto de prueba haciendo uso de los *notebooks* de Kaggle utilizando su GPU (T4 x2).

- **Número de parámetros del modelo**

En la siguiente tabla se resumen las características de cada modelo:

Modelo	*Accuracy*	*Recall*	*Precision*	Tiempo (s)	Nº parámetros (M)
VGG19	53.662	44.237	45.842	7738	**139.619**
ResNet50	72.562	61.763	63.507	5469	23.612
ResNet101	68.046	58.637	62.652	7640	42.682
MobileNetV2	61.518	56.49	56.913	**2546**	2.273
DenseNet121	76.622	76.216	**75.536**	4502	7.05
EfficientNetB0	**76.926**	**77.447**	71.632	2816	4.065

Cuadro 5.1: Comparativa de los modelos.

En la tabla 5.1, se puede ver que los modelos EfficientNetB0 y DenseNet121 obtienen mejores resultados en términos de *accuracy*, *recall* y *precision*. Además, el modelo EfficientNetB0 es uno de los modelos que posee menor tiempo de ejecución, solo por detrás de MobileNetV2. Esto demuestra que dicho modelo es ligera en términos de parámetros, aun ofreciendo un rendimiento competitivo.

Otro punto a destacar es el tiempo de ejecución, donde se observa que el modelo MobileNetV2, diseñado para dispositivos con recursos limitados, tiene el menor tiempo de ejecución, con el menor número de parámetros. Sin embargo, dicho modelo no presenta los mejores resultados, pues presenta una arquitectura más ligera en comparación con los demás modelos. Por otro lado, el modelo VGG19 es el que presenta mayor número de parámetros y tiempo de ejecución, pues los demás modelos presentan estructuras que reducen el número de parámetros.

Los resultados obtenidos con los distintos modelos se han subido a la competición creada en las dos plataformas:

Rank	Participant team	Accuracy (↑)	Precision (↑)	Recall (↑)	Last submission at	Meta Attributes
1	Host_5302_Team B	0.77	0.72	0.77	8 minutes ago	View
2	Host_5302_Team B	0.77	0.76	0.76	21 minutes ago	View
3	lit	0.73	0.64	0.62	1 hour ago	View
4	Host_5302_Team B	0.68	0.63	0.59	49 minutes ago	View
5	Host_5302_Team B	0.62	0.57	0.56	36 minutes ago	View
6	Host_5302_Team B	0.54	0.46	0.44	1 hour ago	View

(a) Tabla de clasificación en EvalAI.

Leaderboard								
Task:					Fact Sheet Answers	Object Recognition Testing Task		
#	Participant	Entries	Date	ID	Model	Accuracy	Precision	Recall
1	litstar	5	2024-05-31 07:45	66351	EfficientNetB0	**0.77**	0.72	0.77
2	litstar	5	2024-05-31 07:45	66349	DenseNet121	**0.77**	0.76	0.76
3	yymm02	2	2024-05-30 23:55	66194	ResNet50	**0.73**	0.64	0.62
4	litstar	5	2024-05-31 07:40	66346	ResNet101	**0.68**	0.63	0.59
5	yymm02	2	2024-05-31 07:45	66348	MobileNetV2	**0.62**	0.57	0.56
6	litstar	5	2024-05-31 07:39	66342	VGG19	**0.54**	0.46	0.44

(b) Tabla de clasificación en Codabench.

Figura 5.1: Resultados en las plataformas EvalAI y Codabench.

Capítulo 6

Trabajo futuro

En los trabajos futuros, se planea investigar la posibilidad de instalar y ejecutar la competición en un entorno local, en lugar de depender de los servidores de EvalAI y Codabench. Durante el desarrollo de este proyecto, se intentó llevar a cabo esta implementación local, pero sin éxito.

Ejecutar la competición localmente proporcionaría varias ventajas:

- **Pruebas más rápidas y directas**. Permitiría realizar pruebas de forma más rápida y directa, reduciendo el tiempo de espera asociado a la subida y ejecución de tareas en servidores remoto. Por ejemplo, en EvalAI, era necesario esperar a que los administradores de EvalAI aprobasen la competición para que pudieran participar los demás usuarios.
- **Personalización de la infraestructura**. Se podría personalizar y modificar la infraestructura de la competición según las necesidades del proyecto. Por ejemplo, en el caso de EvalAI, no se le daba la opción al participante de publicar varios resultados en la tabla de clasificación.
- **Reducción de la dependencia de servidores en la nube**. Reducir esta dependencia asegura que el trabajo no se vea interrumpido por problemas de conectividad o mantenimiento del servidor.

Este enfoque no solo será beneficioso para los organizadores de la competición, sino que también mejorará la experiencia de los participantes, proporcionando un entorno de competición más adaptado a las necesidades específicas de cada competición.

Capítulo 7

Análisis de impacto

El desarrollo de competiciones de inteligencia artificial, como las abordadas en este proyecto, tiene un impacto significativo en diversos ámbitos y está estrechamente vinculado con los Objetivos de Desarrollo Sostenible (ODS) establecidos por la ONU. Los ODS son un conjunto de 17 objetivos globales diseñados para abordar los mayores desafíos de la humanidad, incluyendo la pobreza, la desigualdad, el cambio climático, la degradación ambiental, la paz y la justicia.

En el ámbito tecnológico y científico, el proyecto se enfoca en la creación y análisis de competiciones de clasificación de imágenes satelitales utilizando plataformas como EvalAI y Codabench. Este tipo de competiciones impulsa la innovación en el aprendizaje automático al permitir probar y evaluar diferentes modelos en un entorno competitivo, promoviendo el avance tecnológico y alineándose con el objetivo 9: Industria, Innovación e Infraestructura.

En el ámbito educativo y formativo, las competiciones de IA son una herramienta educativa valiosa que proporciona a estudiantes y profesionales la oportunidad de aplicar sus conocimientos en escenarios prácticos y realistas, alineándose con el objetivo 4: Educación de Calidad, al promover oportunidades de aprendizaje continuo y preparación para el mercado laboral.

Bibliografía

[1] DIUx. (2018) Diux xview 2018 detection challenge. [Online]. Available: http://xviewdataset.org/

[2] C. Uslu. (2022) What is Kaggle? [Online]. Available: https://www.datacamp.com/blog/what-is-kaggle

[3] Coursera. (2023) What Is Kaggle and What Is It Used For? [Online]. Available: https://www.coursera.org/articles/kaggle

[4] CloudCV. (2013) Cloudcv. [Online]. Available: https://cloudcv.org/

[5] Google. (2021) Cloudcv. building platforms for reproducible ai research. [Online]. Available: https://summerofcode.withgoogle.com/archive/2021/organizations/5332159690178560

[6] D. Yadav, R. Jain, H. Agrawal, P. Chattopadhyay, T. Singh, A. Jain, S. B. Singh, S. Lee, and D. Batra, "Evalai: Towards better evaluation systems for ai agents," vol. arXiv:1902.03570, 2019.

[7] (2024) Docker. [Online]. Available: https://www.docker.com/

[8] Django: The web framework for perfectionists with deadlines. [Online]. Available: https://www.djangoproject.com/

[9] A. W. Services. Amazon simple queue service. [Online]. Available: https://aws.amazon.com/sqs/

[10] ——. Amazon elastic container service (ecs). [Online]. Available: https://aws.amazon.com/ecs/

[11] A. Pavao, I. Guyon, A.-C. Letournel, D.-T. Tran, X. Baro, H. J. Escalante, S. Escalera, T. Thomas, and Z. Xu, "Codalab competitions: An open source platform to organize scientific challenges," *Journal of Machine Learning Research*, vol. 24, no. 198, pp. 1–6, 2023. [Online]. Available: http://jmlr.org/papers/v24/21-1436.html

[12] Microsoft. (2019) Codalab. [Online]. Available: https://www.microsoft.com/en-us/research/project/codalab/

[13] Codalab. News codalab. [Online]. Available: https://codalab.lisn.upsaclay.fr/highlights

[14] ——. About codalab. [Online]. Available: https://codalab-competitions.readthedocs.io/en/latest/About/

[15] Google. Signed URLs. [Online]. Available: https://cloud.google.com/storage/docs/access-control/signed-urls

[16] Z. Xu, S. Escalera, A. Pavão, M. Richard, W.-W. Tu, Q. Yao, H. Zhao, and I. Guyon, "Codabench: Flexible, easy-to-use, and reproducible meta-benchmark platform," *Patterns*, vol. 3, no. 7, p. 100543, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666389922001465

[17] Competition Bundle Structure. [Online]. Available: https://github.com/codalab/codabench/wiki/Competition-Bundle-Structure

[18] (2023) VGG: ¿Qué es este modelo? ¡Daniel te lo cuenta todo! [Online]. Available: https://datascientest.com/es/vgg-que-es-este-modelo-daniel-te-lo-cuenta-todo

[19] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *The International Conference on Learning Representations (ICLR)*, 2015.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[21] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[22] N. Sharma. (2023) What is MobileNetv2? Features, Architecture, Application and More. [Online]. Available: https://www.analyticsvidhya.com/blog/2023/12/what-is-mobilenetv2/

[23] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[24] S. Das. (2023) Implementing densenet-121 in pytorch: A step-by-step guide. [Online]. Available: https://medium.com/deepkapha-notes/implementing-densenet-121-in-pytorch-a-step-by-step-guide-c0c2625c2a60

[25] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, vol. 97, 2019, pp. 6105–6114.

Anexos

Apéndice A

Anexo

A.1. Creación de un token personal y asignación al repositorio en GitHub

Para crear un token personal en GitHub se realiza lo siguiente:

1. En la esquina superior izquierda, se debe hacer clic en la foto de perfil de GitHub. A continuación se selecciona la opción “Settings”.

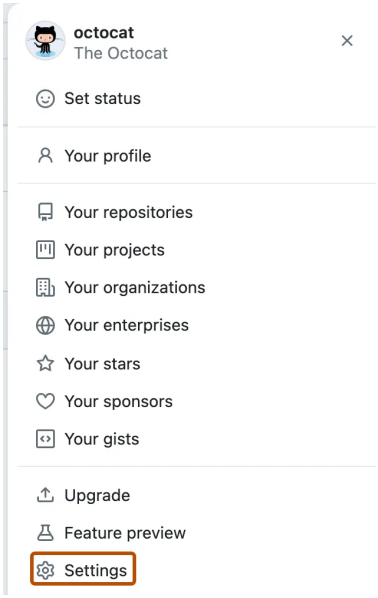


Figura A.1: Configuración GitHub.

2. En la barra lateral izquierda, se debe hacer clic en “<Developer settings>” y después en “Personal access tokens” seleccionando la opción de “Tokens (classic)”.

3. Seleccionar “Generate new token”.

4. En el campo “Note”, se debe proporcionar un nombre descriptivo al token.
5. Se añade una fecha de expiración al token.
6. Seleccionar los alcances del token: en el caso de la competición, “repo”.
7. Hacer clic en “Generate token”.
8. Se copia el token creado en el portapapeles.

Una vez que el token está copiado, se le agrega en los secretos del repositorio con el nombre `AUTH_TOKEN`.

Para ello, se sigue los siguientes pasos.

1. En la página principal del repositorio, seleccionar “Settings”.

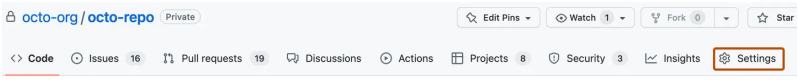


Figura A.2: Configuración del repositorio de GitHub.

2. En la sección de “Security”, seleccionar “Secrets and variables” y luego “Actions”.

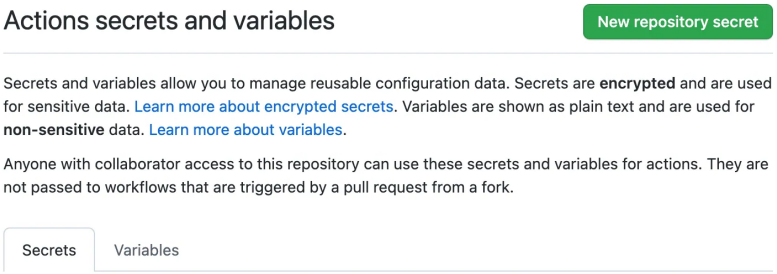


Figura A.3: Creación de un secreto en el repositorio.

3. En la pestaña de “Secrets”, crear un nuevo repositorio secreto: en el campo “Name” se rellena “AUTH_TOKEN” y en “Secrets” el token copiado previamente.

A.2. Informe de originalidad de Turnitin

En este apartado se muestra el informe de originalidad generado por Turnitin.

tfg_YIMIN_2324.pdf

ORIGINALITY REPORT

11%	9%	3%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universidad Politécnica de Madrid Student Paper	1%
2	repositorio.uam.es Internet Source	<1%
3	issuu.com Internet Source	<1%
4	Submitted to Universidad TecMilenio Student Paper	<1%
5	arxiv.org Internet Source	<1%
6	www.arxiv-vanity.com Internet Source	<1%
7	Submitted to Universidad Nacional de Educación a Distancia Student Paper	<1%
8	Submitted to University of Wales, Bangor Student Paper	<1%
9	academica-e.unavarra.es	

	Internet Source	<1 %
10	oa.upm.es Internet Source	<1 %
11	graphics.ewha.ac.kr Internet Source	<1 %
12	Submitted to Universidad de Málaga - Tii Student Paper	<1 %
13	Submitted to University of Sunderland Student Paper	<1 %
14	buildmedia.readthedocs.org Internet Source	<1 %
15	repository.kaust.edu.sa Internet Source	<1 %
16	Submitted to University of Nottingham Student Paper	<1 %
17	Submitted to London Metropolitan University Student Paper	<1 %
18	Submitted to Ana G. Méndez University Student Paper	<1 %
19	openaccess.thecvf.com Internet Source	<1 %
20	Submitted to UTEC Universidad de Ingenieria & Tecnologia	<1 %

Student Paper

21	Submitted to Corporación Universitaria Minuto de Dios, UNIMINUTO Student Paper	<1 %
22	Submitted to Gisma University of Applied Sciences GmbH Student Paper	<1 %
23	Submitted to University of Northumbria at Newcastle Student Paper	<1 %
24	openaccess.mef.edu.tr Internet Source	<1 %
25	www.slideshare.net Internet Source	<1 %
26	Submitted to Universidad de San Martín de Porres Student Paper	<1 %
27	eprints.hsr.ch Internet Source	<1 %
28	Submitted to ITESM: Instituto Tecnologico y de Estudios Superiores de Monterrey Student Paper	<1 %
29	Submitted to Obudai Egyetem Student Paper	<1 %
30	dl.dell.com Internet Source	<1 %

31	cloud.google.com Internet Source	<1 %
32	Submitted to Asia Pacific Instutute of Information Technology Student Paper	<1 %
33	sitb2021.win.tue.nl Internet Source	<1 %
34	vdocumento.com Internet Source	<1 %
35	Submitted to UNIBA Student Paper	<1 %
36	eddieblakeblog.wordpress.com Internet Source	<1 %
37	view.genial.ly Internet Source	<1 %
38	Submitted to Birkbeck College Student Paper	<1 %
39	Submitted to Universidad Carlos III de Madrid - EUR Student Paper	<1 %
40	agexporthoy.export.com.gt Internet Source	<1 %
41	learndigital.withgoogle.com Internet Source	<1 %

42	api.besoccer.com Internet Source	<1 %
43	cum.unex.es Internet Source	<1 %
44	de.slideshare.net Internet Source	<1 %
45	developers.google.com Internet Source	<1 %
46	doczz.es Internet Source	<1 %
47	hdl.handle.net Internet Source	<1 %
48	playground.citethisforme.com Internet Source	<1 %
49	sol.sbc.org.br Internet Source	<1 %
50	students.cs.ucl.ac.uk Internet Source	<1 %
51	www.consorci.org Internet Source	<1 %
52	"Newspapers collection management: printed and digital challenges", Walter de Gruyter GmbH, 2008 Publication	<1 %

53	experienceleague.adobe.com Internet Source	<1 %
54	gospelidea.com Internet Source	<1 %
55	prezi.com Internet Source	<1 %
56	repositorio.usm.cl Internet Source	<1 %
57	rinacional.tecnm.mx Internet Source	<1 %
58	www.cacic2016.unsl.edu.ar Internet Source	<1 %
59	www.ciclismoextremadura.org Internet Source	<1 %
60	www.coursehero.com Internet Source	<1 %
61	www.cs.cinvestav.mx Internet Source	<1 %
62	www.miembrosprodigy.com.mx Internet Source	<1 %
63	www.researchgate.net Internet Source	<1 %
64	Nuno Moutinho. "Sharing Information in a Virtual Community of Crowdfunding: The case	<1 %

of Kickstarter", Repositório Aberto da Universidade do Porto, 2014.
Publication

65	code.tutsplus.com Internet Source	<1 %
66	docs.citrix.com Internet Source	<1 %
67	help.blackboard.com Internet Source	<1 %
68	patents.google.com Internet Source	<1 %
69	pdffox.com Internet Source	<1 %
70	profesores.elo.utfsm.cl Internet Source	<1 %
71	publicaciones.eafit.edu.co Internet Source	<1 %
72	randolph.apsva.us Internet Source	<1 %
73	sched.mediaparty.info Internet Source	<1 %
74	translate.evernote.com Internet Source	<1 %
75	upcommons.upc.edu Internet Source	<1 %

76	www.easytobook.com Internet Source	<1 %
77	www.ehu.eus Internet Source	<1 %
78	www.scielo.cl Internet Source	<1 %
79	www.tdx.cat Internet Source	<1 %
80	www.voleyplaya.universalevents2004.com Internet Source	<1 %
81	Víctor Javier Garrido Peñalver. "Arquitectura de interoperabilidad para mejorar las capacidades de análisis de información y toma de decisiones", Universitat Politecnica de Valencia, 2024 Publication	<1 %
82	empiezoinformatica.wordpress.com Internet Source	<1 %
83	Abhinav Sharma, Dhiren Soneji, Aabha Ranade, Dhwani Serai, Priya RL, CS Lifna, Shashikant R Dugad. "Gujarati Script Recognition", Procedia Computer Science, 2023 Publication	<1 %
84	futur.upc.edu Internet Source	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Mon Jun 03 19:48:16 CEST 2024
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)