CREACIÓN MODELO ENTRENADO CON IA

En esta documentación aprenderás a **crear y entrenar un modelo de inteligencia artificial (IA)** desde cero para la detección de objetos. Aprenderás todo el flujo de trabajo, desde la creación del dataset, pasando por el entrenamiento del modelo, hasta su uso en código para detectar objetos en imágenes o video en tiempo real.

El objetivo de este tutorial es que, al finalizar, puedas:

- Tener un dataset correctamente anotado y preprocesado.
- Entrenar un modelo de detección de objetos utilizando Roboflow
- Descargar el modelo entrenado y usarlo en tus proyectos, por ejemplo en robots o sistemas de visión.

¿Qué es Roboflow?

Roboflow es una plataforma en la nube que permite a a crear, organizar y entrenar datasets de manera fácil. Con Roboflow se pueden:

- Subir imágenes y anotarlas para tareas de detección de objetos, clasificación o segmentación.
- Exportar el dataset en formatos compatibles con frameworks populares como YOLO, TensorFlow, PyTorch y ONNX.
- Entrenar modelos directamente en la nube o descargar el dataset para entrenarlo localmente (es lo que vamos a hacer)

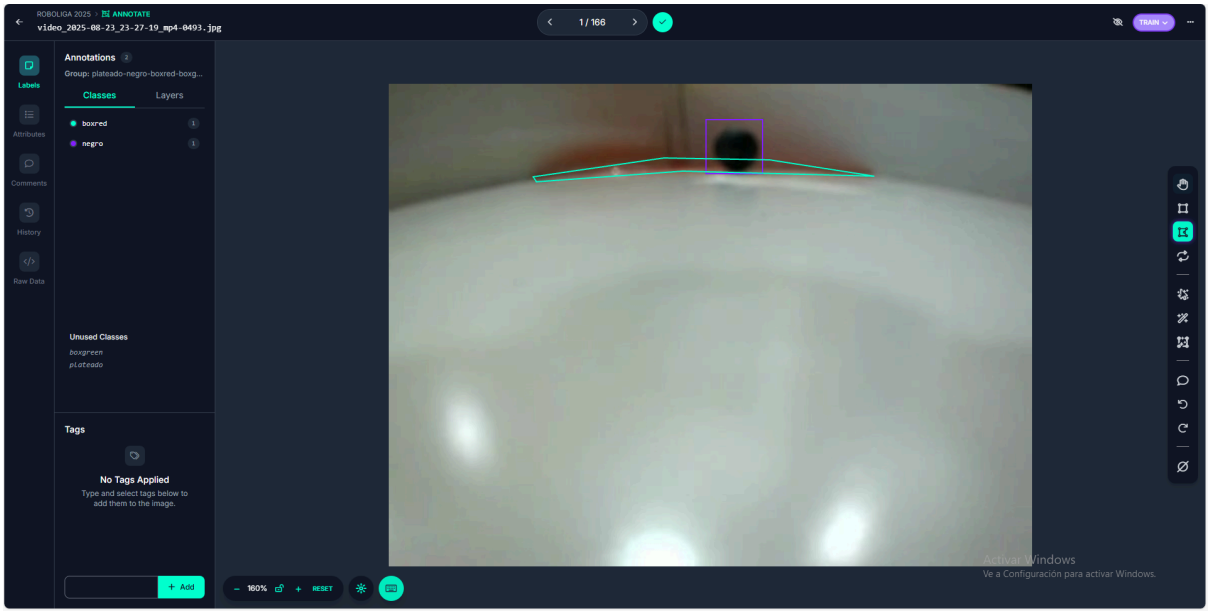
Creación y entrenamiento de un modelo con Roboflow

Paso 1: Crear un proyecto en Roboflow

1. Ingresar a Roboflow y registrarse.
2. Hacer clic en **“Create New Project”**.
3. Seleccionar **tipo de proyecto**: `Object Detection`.
4. Poner un nombre al proyecto (por ejemplo: `Roboliga 2025`).
5. Elegir el **formato de anotación** (YOLO, COCO, etc.) y crear el proyecto.

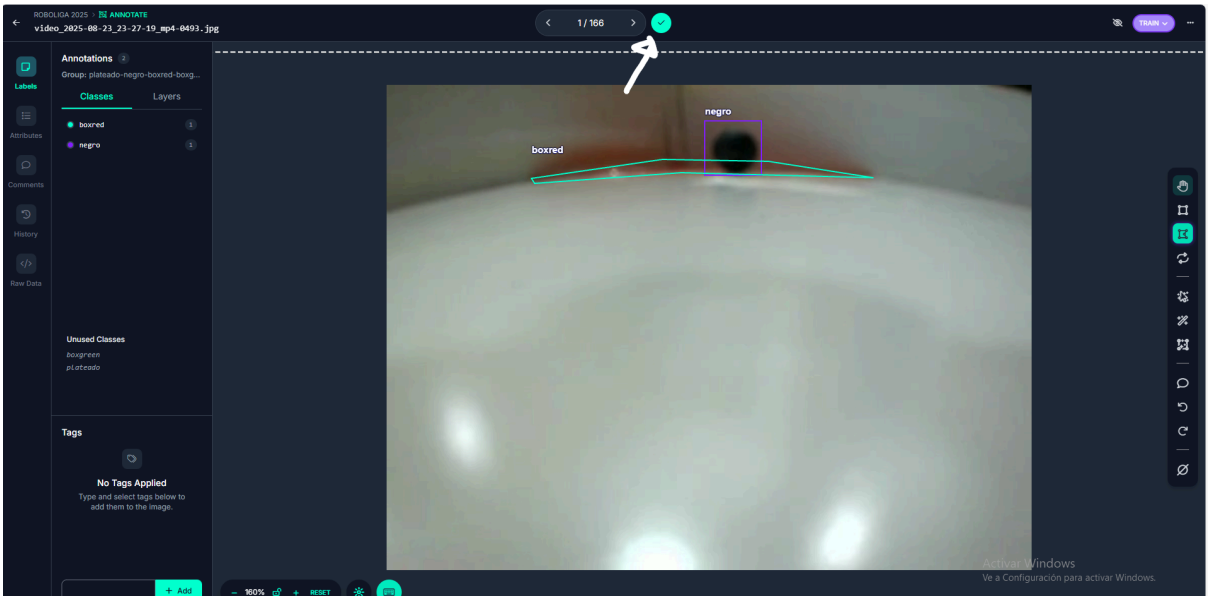
Paso 2: Subir imágenes y etiquetar

1. Hacer clic en “Upload Images” dentro del proyecto.
2. Seleccionar las imágenes del dataset.
3. Asignar las clases o labels (por ejemplo: `boxgreen`, `boxred`, `negro`, `plateado`).
4. Revisar que los bounding boxes estén correctamente colocados.

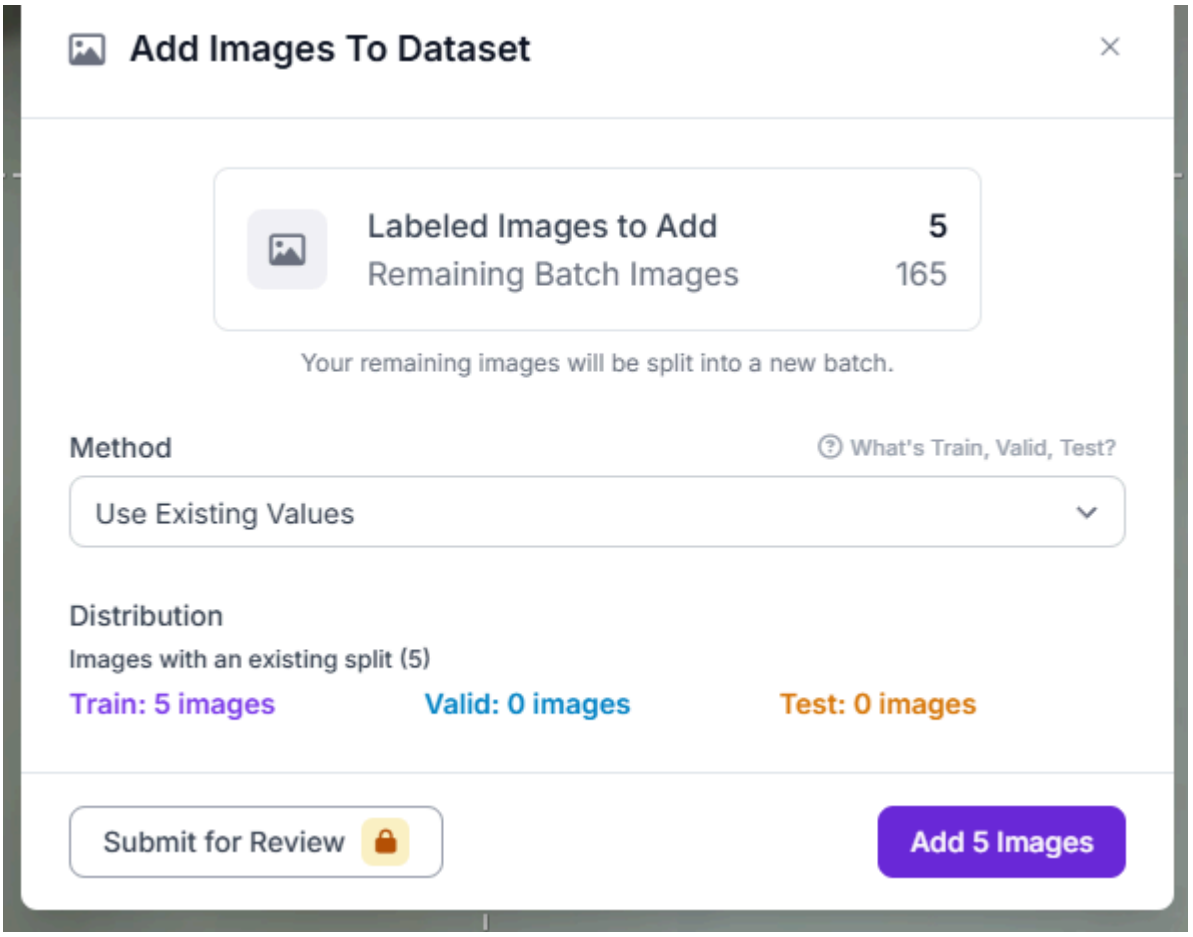


Paso 3: Generar dataset listo para entrenamiento

1. Dentro del proyecto, hacer clic en la opción indicada (la flecha en la imagen).

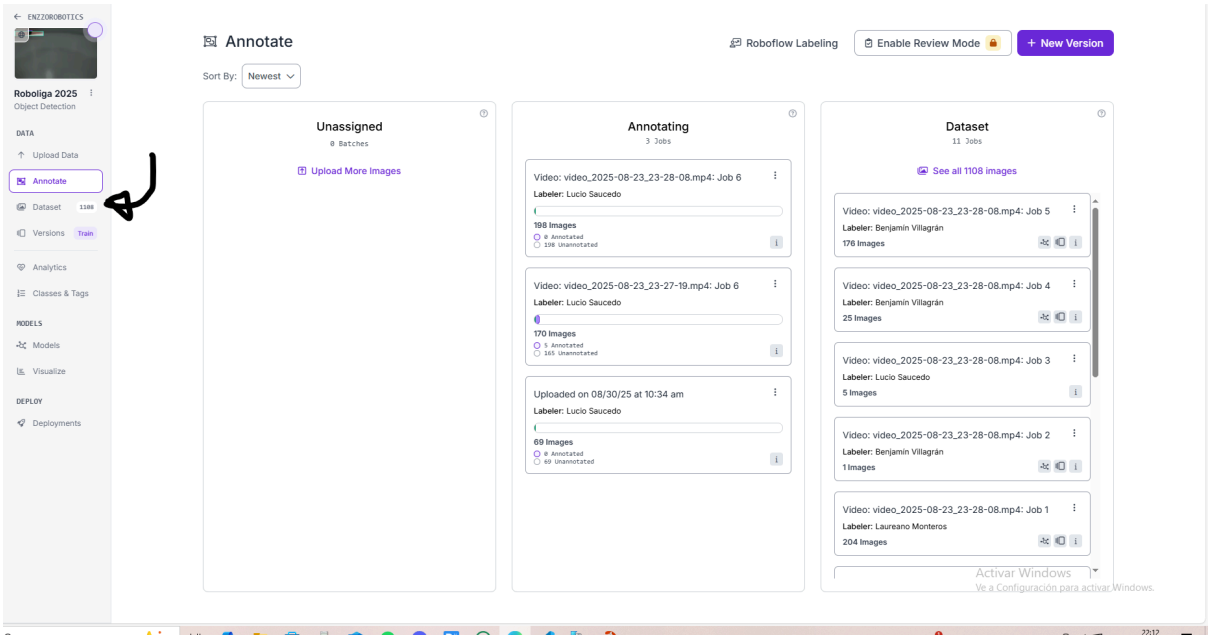


2. Esto añadirá al **dataset final** todas las imágenes que previamente revisaste y etiquetaste, garantizando que estén listas para el entrenamiento.

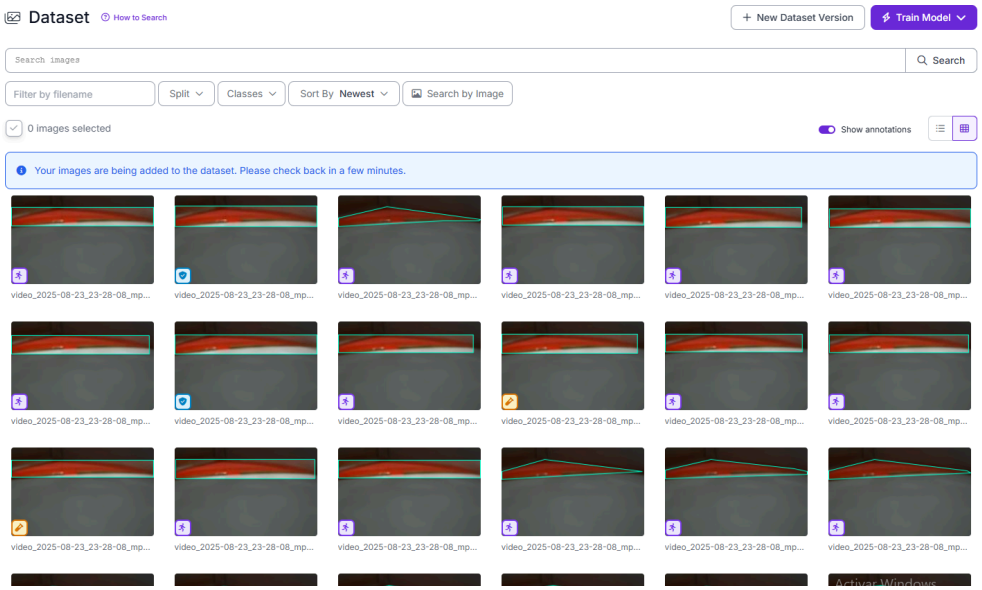


Paso 4: Entrenamiento en Roboflow

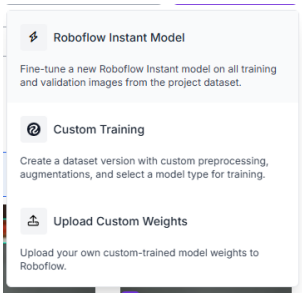
1. Una vez creado el dataset, dirigirse a la pestaña correspondiente al entrenamiento (indicado con la flecha en la imagen).



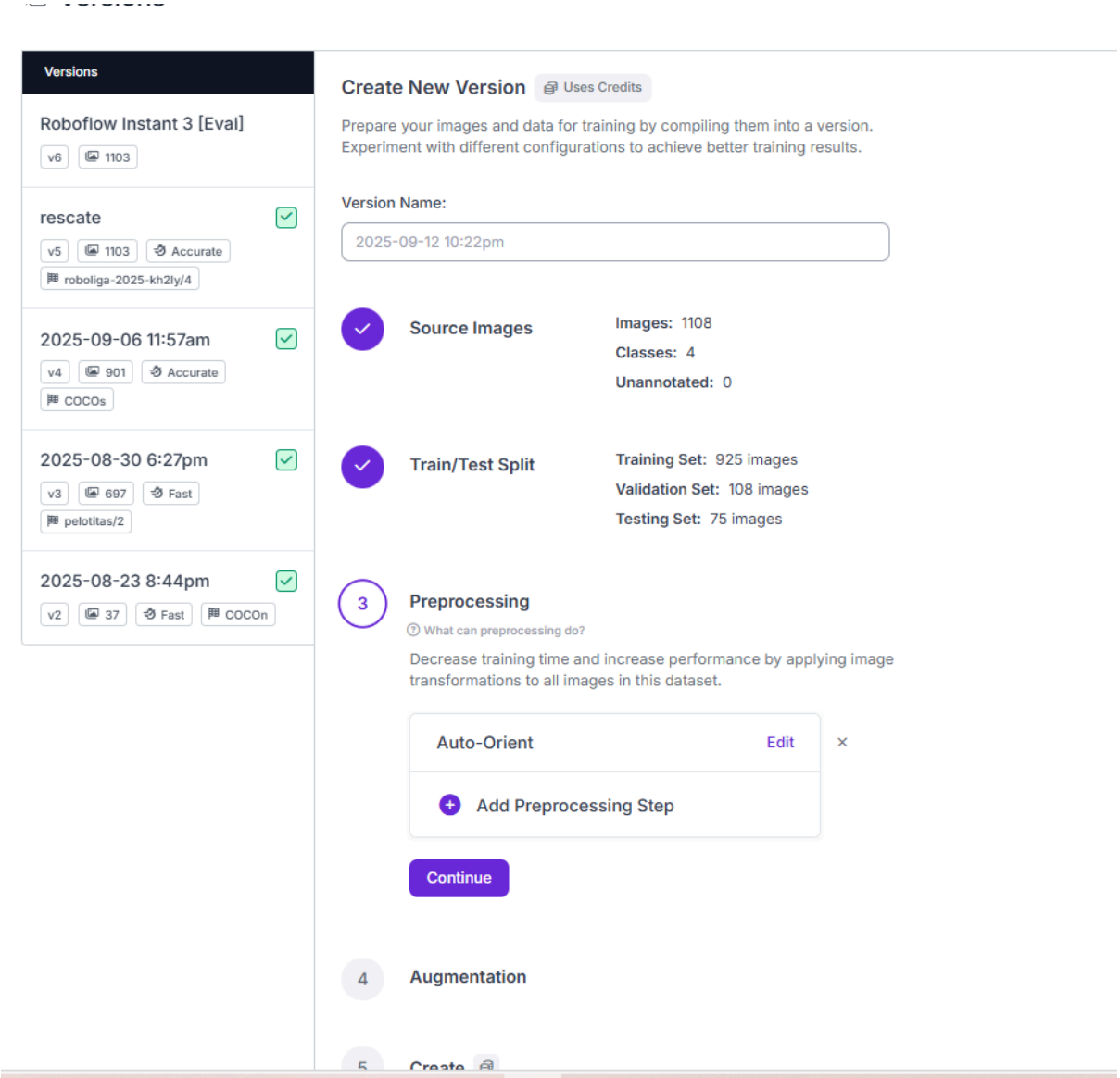
2. Hacer clic en **“Train Model”**, ubicada en la esquina superior derecha.



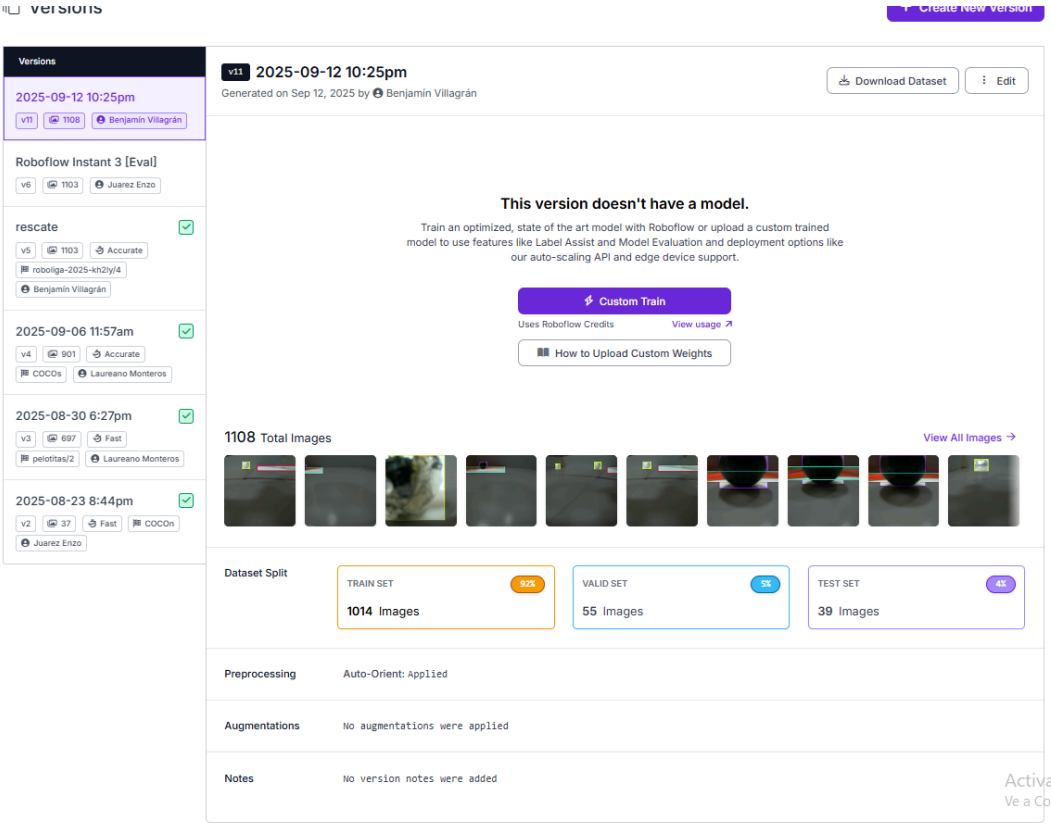
3. Se abrirán tres opciones; seleccionar **“Custom Training”**.



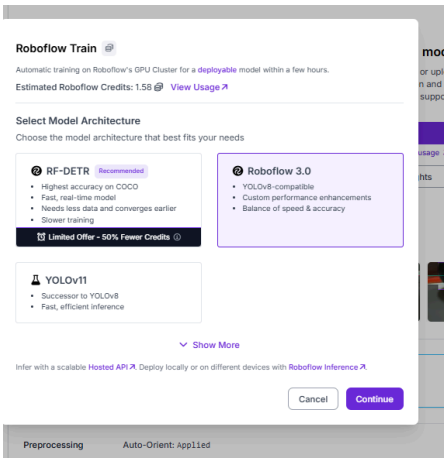
4.Asignar un **nombre a la versión del modelo** que se va a crear.



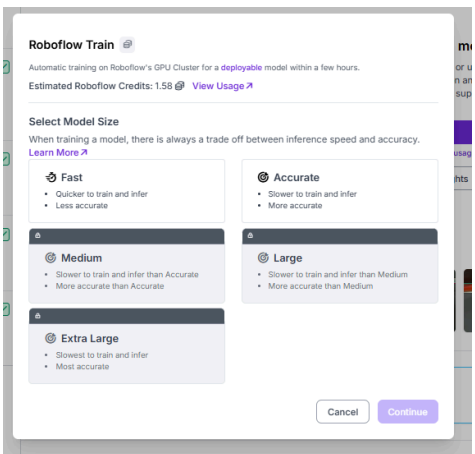
5. Seguir el asistente y hacer clic en **“Continuar”** hasta crear la versión del modelo.



6. Roboflow indicará que la versión aún no tiene un modelo entrenado. Hacer clic en **“Custom Train”** y seleccionar **Roboflow 3.0** como framework.

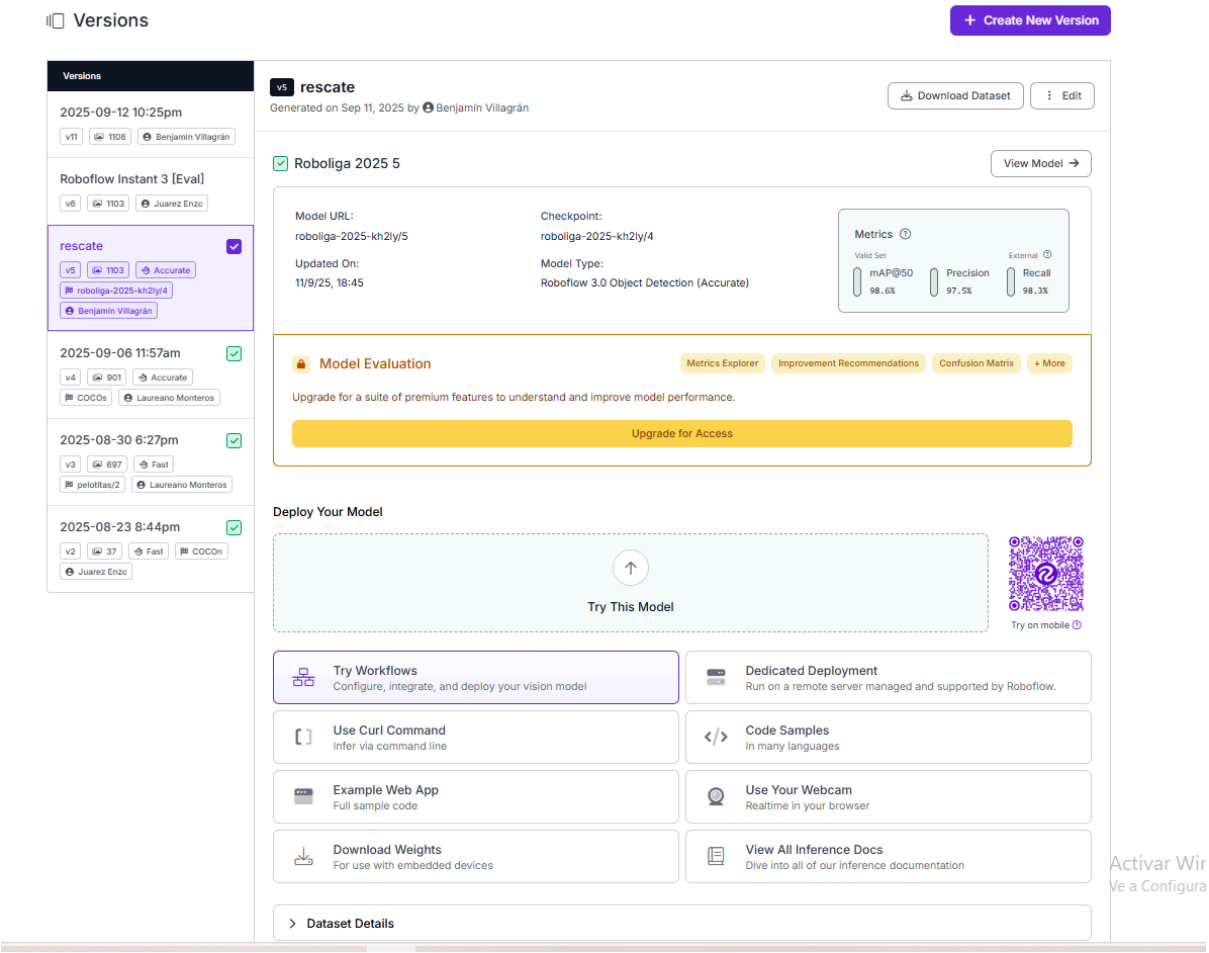


7. Elegir la opción **“Accurate”** para priorizar la precisión del modelo durante el entrenamiento.



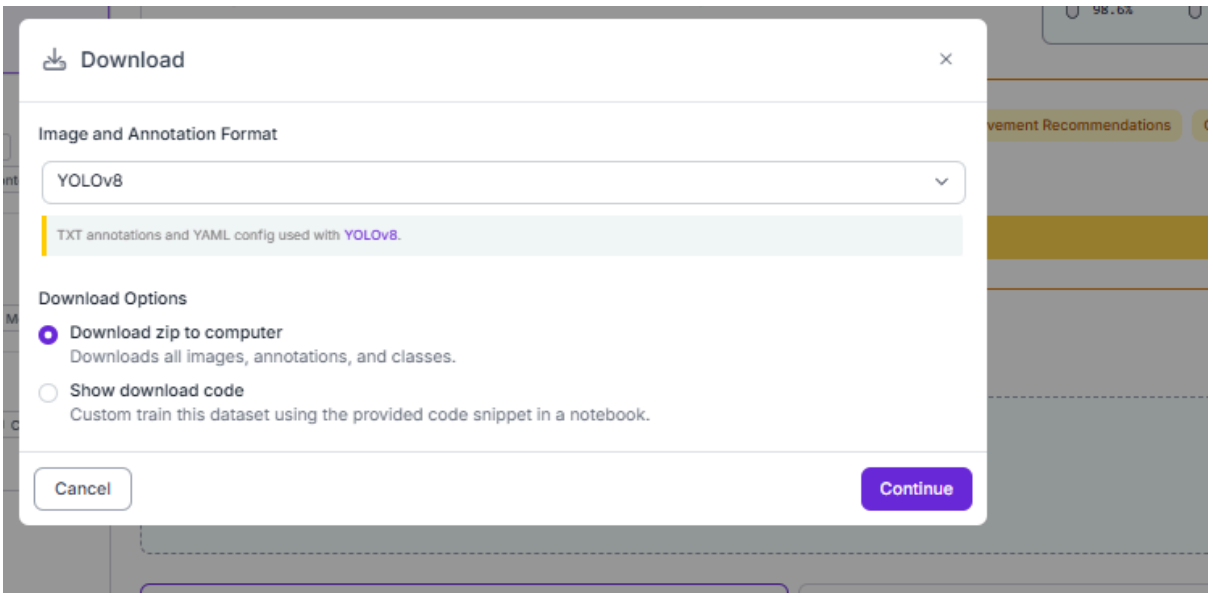
8.Tras configurar los parámetros, Roboflow comenzará a entrenar el dataset. El tiempo de entrenamiento dependerá del tamaño del dataset y de los recursos de la nube disponibles.

9.Una vez finalizado, dirigirse a la sección **“Versions”** para visualizar el modelo entrenado.



Nota: Roboflow no permite descargar el modelo entrenado de forma gratuita.

Para continuar trabajando localmente, se puede descargar el **dataset con todas las anotaciones ya realizadas.**
10. Hacer clic en **“Download Dataset”** y seguir el asistente haciendo clic en **“Continuar”**.



11. Esto permitirá usar el dataset para entrenar el modelo localmente con cualquier framework compatible, como YOLOv8 o PyTorch.

Entrenamiento en Google Colab

Después de haber creado y preparado nuestro **dataset en Roboflow**, ahora vamos a dar el siguiente paso: **entrenar nuestro modelo de detección de objetos utilizando Google Colab**.

En esta etapa aprenderemos a:

1. **Cargar el dataset descargado de Roboflow** en Colab, que ya contiene todas las imágenes y etiquetas revisadas.

2. **Configurar el entorno de entrenamiento**, incluyendo la instalación de librerías necesarias como ultralytics.

3. **Entrenar el modelo** de manera sencilla y guiada.

4. **Evaluar y guardar el modelo entrenado**, listo para usarlo en proyectos locales, robots o aplicaciones de visión por

computadora.

Esta etapa aprovecha la **potencia de procesamiento de Colab** para entrenar modelos de manera eficiente, incluso si tu computadora local no tiene GPU.

¿Qué es Google Colab?

Google Colab (Colaboratory) es una plataforma gratuita de Google que permite **escribir y ejecutar código en Python directamente desde el navegador**, sin necesidad de instalar nada en tu computadora.

Sus principales ventajas son:

- **Uso de GPU y TPU gratuitas:** Ideal para entrenar modelos de inteligencia artificial y redes neuronales sin depender de la potencia de tu computadora local.

- **Entorno en la nube:** Todos los archivos y proyectos se pueden guardar en Google Drive, lo que facilita acceder y compartir los proyectos desde cualquier lugar.

- **Integración con librerías de IA y visión por computadora:** Se pueden instalar y usar frameworks como **YOLOv8, TensorFlow, PyTorch, OpenCV**, entre otros.

- **Ideal para principiantes y profesionales:** Permite experimentar con proyectos de IA sin necesidad de configuraciones complejas.

En este tutorial utilizaremos Colab para **entrenar nuestro modelo de detección de objetos usando el dataset previamente creado en Roboflow**, aprovechando su GPU para un entrenamiento rápido y eficiente.

Tutorial: Entrenamiento del modelo en Google Colab

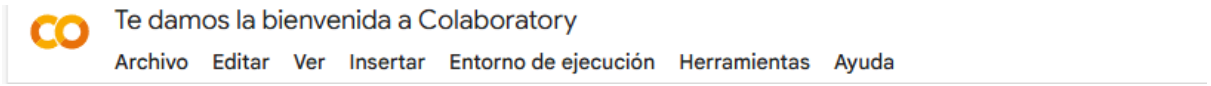
Paso 1: Abrir Google Colab

1. Ingresar a Google Colab.

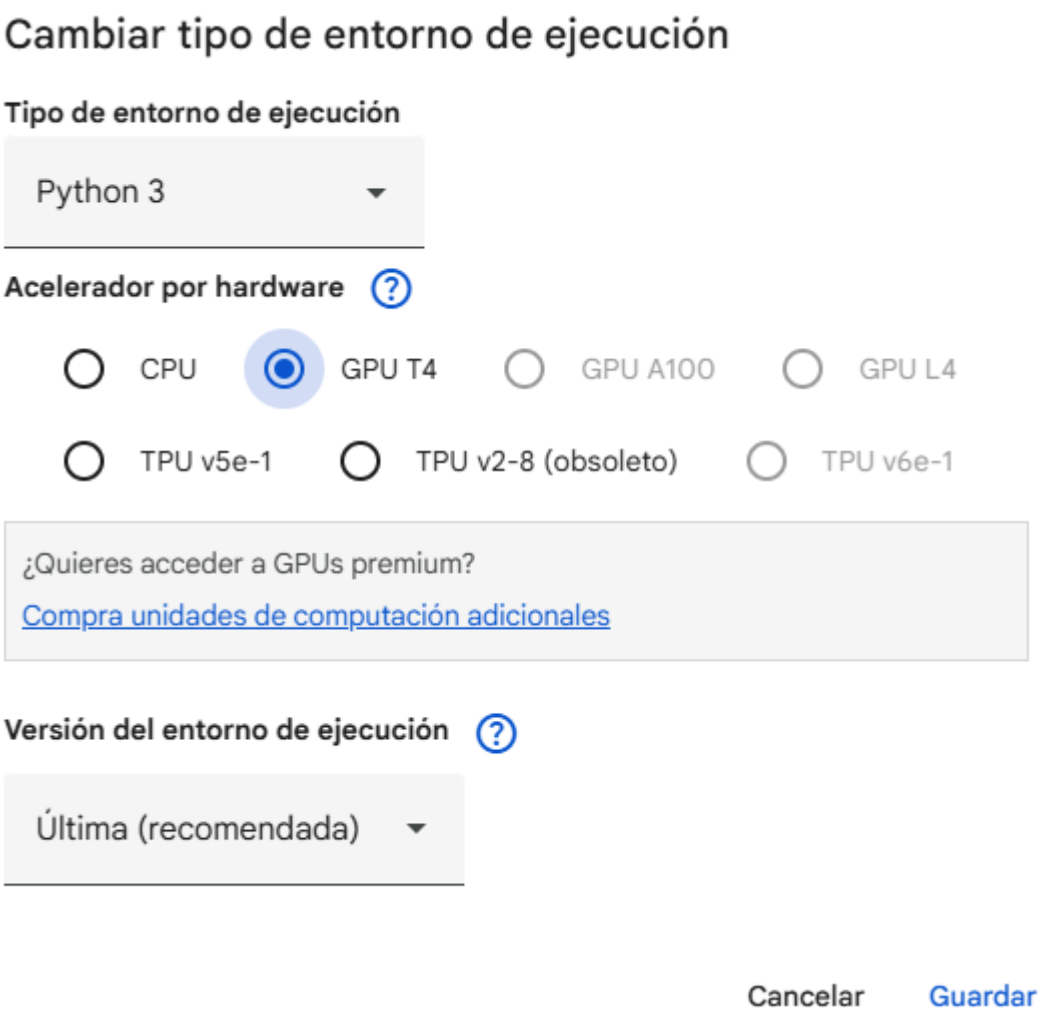
2. Hacer clic en **"Nuevo cuaderno"** (New Notebook) para crear un proyecto vacío donde trabajaremos nuestro entrenamiento.

Paso 2: Configurar la GPU

1. Ir al menú **"Entorno de ejecución"** → **"Cambiar tipo de entorno de ejecución"**.



2. En la opción **Acelerador de hardware**, seleccionar **GPU**.

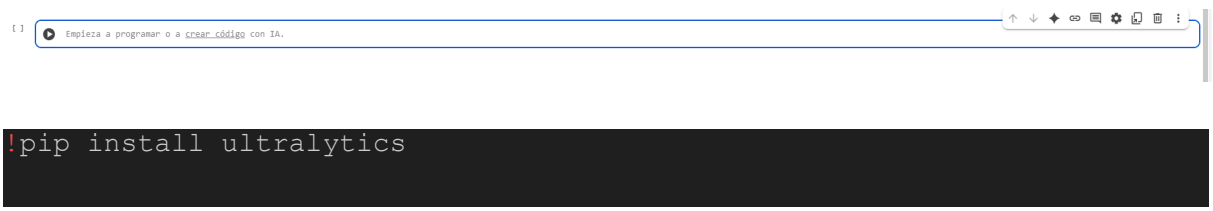


3. Hacer clic en **Guardar**.

   Esto permitirá que nuestro modelo se entrene más rápido, aprovechando la potencia de la GPU de Colab.

Paso 3: Instalar librerías necesarias

En la primera celda del cuaderno, copiar y ejecutar el siguiente código para instalar YOLOv8 y otras dependencias:



Esto instalará todo lo necesario para trabajar con nuestro dataset y entrenar el modelo.

Paso 4: Cargar y extraer el dataset en Google Colab

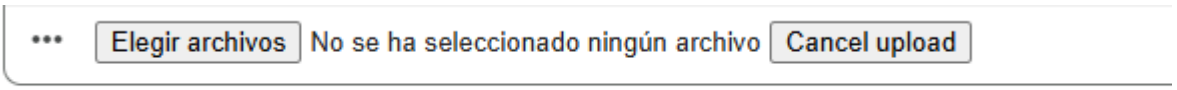
1. Subir el archivo ZIP de tu dataset descargado de Roboflow:

```
from google.colab import files



# Aparecerá un botón para seleccionar tu archivo ZIP

files.upload()
```



2. Una vez subido, extraer el contenido del ZIP a la carpeta `dataset`:

```
!unzip dataset.zip -d dataset
```

Esto creará una carpeta llamada `dataset` con todas las imágenes y etiquetas listas para entrenar el modelo.

Paso 5: Entrenar el modelo

```
from ultralytics import YOLO

# Cargar un modelo YOLOv8 base (ej. yolov8n.pt)
```

```
model = YOLO("yolov8n.pt")

# Entrenar el modelo usando el dataset extraído

model.train(data="dataset/data.yaml", epochs=50)
```

Asegúrate de que dentro del ZIP exista el archivo `data.yaml` que indica las rutas de imágenes y etiquetas. Ten en cuenta que si no seleccionaste la opcion de GPU tardara horas en cambio si la seleccionaste tardara mucho menos tiempo

Paso 6: Descargar y probar el modelo entrenado

1. Una vez finalizado el entrenamiento, el modelo se guarda automáticamente en:

`runs/detect/train/weights/best.pt`

2. Para descargarlo a tu computadora, ejecutar en Colab:

```
from google.colab import files

files.download("runs/detect/train/weights/best.pt")
```

Esto abrirá un cuadro de diálogo para guardar el archivo.

3. Este archivo `.pt` contiene el **modelo entrenado** y puede ser usado para:
    - Inferencia en tu computadora local.
    - Integración en un robot o proyecto de visión por computadora.
    - Exportarlo a otros formatos (ONNX, CoreML, etc.) si lo necesitás.

Consideraciones para Raspberry Pi

El archivo `.pt` que descargamos contiene el **modelo completo entrenado** y funciona muy bien en una computadora con GPU potente.

Sin embargo, en una **Raspberry Pi**:

- La potencia de procesamiento es limitada.
- El modelo `.pt` puede ser **muy pesado** y generar **baja velocidad de inferencia**.
- Esto puede causar que el robot o la aplicación se vuelva lenta o no pueda procesar video en tiempo real.

Solución:
Para usar el modelo en dispositivos con recursos limitados, como la Raspberry Pi, es recomendable **convertirlo a un formato más ligero**, por ejemplo:

- **ONNX (`.onnx`)**: Permite usar el modelo con librerías optimizadas para ARM.

En el próximo paso veremos cómo **exportar nuestro modelo `.pt` a ONNX**, lo que permitirá ejecutar la detección en la Raspberry Pi de manera más eficiente.

Paso 7: Exportar el modelo a ONNX para Raspberry Pi

Para que nuestro modelo funcione de manera eficiente en dispositivos con recursos limitados como la Raspberry Pi, lo convertiremos a **ONNX**, un formato más ligero y compatible con frameworks optimizados para ARM.

1. Exportar el modelo entrenado

Después de entrenar tu modelo y tener `best.pt`, ejecuta la siguiente celda en Colab:

```
from ultralytics import YOLO

# Cargar el modelo entrenado

model = YOLO("runs/detect/train/weights/best.pt")

# Exportar a ONNX

model.export(format="onnx")
```

Esto creará un archivo `best.onnx` en la misma carpeta donde se encuentra tu `.pt`.

2. Descargar el modelo ONNX a tu computadora

```
from google.colab import files



# Descargar el archivo ONNX

files.download("runs/detect/train/weights/best.onnx")
```

Ahora tienes tu modelo en formato ONNX, listo para ser transferido a la Raspberry Pi.

Probando el modelo en la Raspberry Pi

Una vez que hayas descargado tu modelo entrenado (`.onnx`) desde Google Colab, ya estás listo para probarlo en tu Raspberry Pi.

1. Instalar librerías necesarias

Antes de ejecutar el código, asegúrate de tener estas librerías instaladas en tu Raspberry Pi:

```
sudo apt update

sudo apt install -y python3-pip libatlas-base-dev

pip3 install --upgrade pip

pip3 install ultralytics numpy

pip3 install opencv-python
```

2. Código de ejemplo para probar el modelo

```
import cv2
import numpy as np
```

```python
from ultralytics import YOLO

# ---- CONFIG ----
MODEL_PATH = "/home/pi/Desktop/roboliga.onnx"  # ruta de tu modelo
VIDEO_PATH = 0  # 0 = cámara por defecto, o ruta de un video
SCORE_THRESHOLD = 0.4
CLASS_NAMES = ['boxgreen', 'boxred', 'negro', 'plateado']

# ---- cargar modelo ----
model = YOLO(MODEL_PATH)
print("Modelo cargado correctamente.")

# ---- abrir cámara o video ----
cap = cv2.VideoCapture(VIDEO_PATH)
if not cap.isOpened():
    raise SystemExit("No se pudo abrir la cámara o video")

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # ---- detección ----
    results = model.predict(frame, conf=SCORE_THRESHOLD, verbose=False)

    # ---- dibujar bounding boxes ----
    for res in results:
        for box in res.boxes:
            cls_id = int(box.cls[0])
            score = float(box.conf[0])
            x1, y1, x2, y2 = map(int, box.xyxy[0].cpu().numpy()) if
hasattr(box.xyxy[0], "cpu") else map(int, box.xyxy[0])
            label = f"{CLASS_NAMES[cls_id]} {score:.2f}"
            color = (0, 255, 0)
            cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)
            cv2.putText(frame, label, (x1, y1-5),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

    # ---- mostrar ----
    cv2.imshow("Detección ONNX", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

```
cap.release()
cv2.destroyAllWindows()
```