


```
%pip install ultralytics
```

 [Mostrar salida oculta](#)

```
from google.colab import drive
```

```
# Montar Google Drive
print("Montando Google Drive...")
drive.mount('/content/drive')
print("Google Drive montado.")
```

 Montando Google Drive...  
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/driv  
Google Drive montado.

```
# Instalar/Actualizar Ultralytics
print("\nInstalando/Actualizando Ultralytics a la última versión para soporte YOLOv11...")
%pip install -U ultralytics
print("Ultralytics instalado/actualizado.")
```

 [Mostrar salida oculta](#)

```
from google.colab import drive
import os
import torch
import time
from ultralytics import YOLO
```

```
# Verificar disponibilidad de GPU
print("\nVerificando disponibilidad de GPU...")
device_to_use = "0" if torch.cuda.is_available() else "cpu"
print(f"Usando dispositivo: {device_to_use}")
```

```
# Definir rutas del dataset y modelo
data_yaml_path = "/content/drive/MyDrive/Colab Notebooks/LSECNuYoloII/dataset/data - colab.yaml"
project_directory = "/content/drive/MyDrive/Colab Notebooks/LSECNuYoloII/runs"
run_name = "yolo10_adam_colab"
```

```
# Lista de modelos a intentar en orden de preferencia
initial_model_preferences = ['yolov11s.pt', 'yolov10s.pt']
checkpoint_path = os.path.join(project_directory, "detect", run_name, "weights", "last.pt")
```

```
# Cargar modelo con lógica de prioridades
print(f"\nBuscando modelo entre: {initial_model_preferences}")
model = None
```

```
if os.path.exists(checkpoint_path):
    print(f"Cargando desde checkpoint: {checkpoint_path}")
    try:
        model = YOLO(checkpoint_path)
    except Exception as e:
        print(f"Error al cargar checkpoint: {e}")
```

```
if model is None:
    for preferred_model in initial_model_preferences:
        try:
            model = YOLO(preferred_model)
            print(f"Modelo {preferred_model} cargado exitosamente.")
```

```
        break
    except Exception as e:
        print(f"Error al cargar {preferred_model}: {e}")

if model is None:
    print("Error crítico: No se pudo cargar ningún modelo.")
    exit()

# Definir hiperparámetros
optimizer = "sgd" #"adam" #"sgd"
lr0 = 0.001 #0.001
lrf = 0.01 #0.2 #0.1
momentum = 0.8 #0.8 #0.937

# Imprimir el modelo que se va a usar
print(f"\nEl modelo que se usará para el entrenamiento es: {model}")

# Entrenar el modelo
print("\nIniciando entrenamiento del modelo...")
results = model.train(data=data_yaml_path, epochs=100, optimizer=optimizer, lr0=lr0, lrf=lrf, momentum=momentum)

print("\nEntrenamiento completado! Resultados guardados en:")
print(os.path.join(project_directory, "detect", run_name))
```



clase3	15	15	0.972	0.8	0.875	0.865
clase4	11	11	0.94	1	0.995	0.96
clase5	7	7	0.711	0.857	0.925	0.925
clase6	9	9	0.773	1	0.928	0.882
clase7	3	3	0.592	0.972	0.863	0.844
clase8	9	9	0.824	0.778	0.886	0.847
clase9	7	7	0.873	0.982	0.96	0.921
clase10	9	9	0.889	1	0.995	0.709

Speed: 0.3ms preprocess, 5.0ms inference, 0.0ms loss, 1.4ms postprocess per image

Results saved to **runs/detect/yolo10\_adam\_colab**

Entrenamiento completado! Resultados guardados en:

/content/drive/MyDrive/Colab Notebooks/LSECNuYoloII/runs/detect/volo10 adam colab