Proyecto final PDI

Detección de señales de tránsito con YOLO y despliegue del modelo

Entrenamiento, exportación (TorchScript/LiteRT) e inferencia en tiempo real

Luis Eduardo Miño Castillo
Juan Camilo Miño Castillo

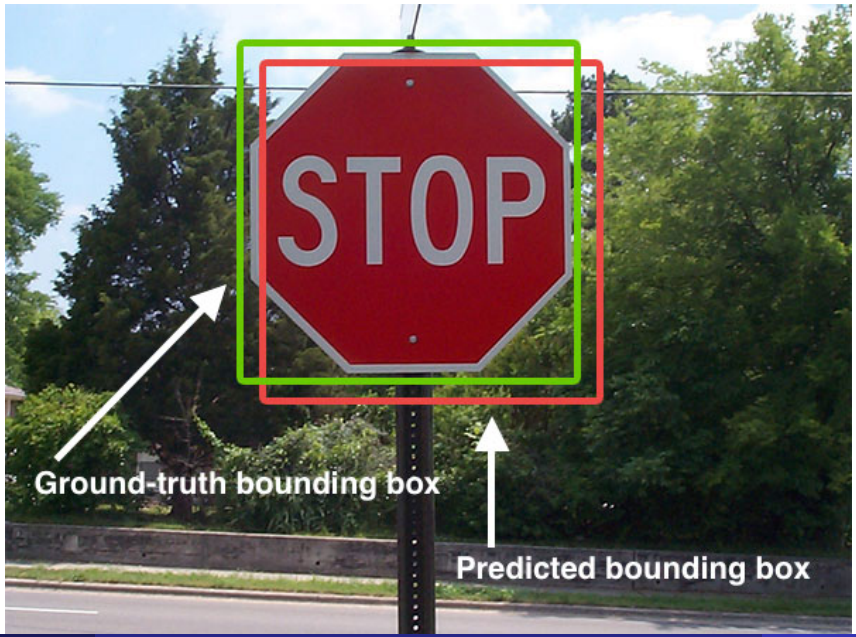
Universidad Nacional de Colombia

12 de diciembre de 2025

Contenido

¿Qué problema resolvemos?

- En la vía, **cada segundo cuenta**: una señal puede marcar la diferencia entre una maniobra segura y un riesgo.
- Buscamos que un sistema de visión **entienda la escena** y **reconozca señales y semáforos en tiempo real**.
- **Objetivo:** localizar cada señal en el frame y clasificarla en una de nuestras **15 clases** (límites de velocidad, *Stop*, semáforo rojo/verde).
- **Impacto:** base para **ADAS**, analítica vial, simuladores y despliegue en **hardware embebido** (portabilidad y eficiencia).



Solución propuesta

Tipo de tarea: Detección de objetos.

Modelo: YOLO (p.ej., YOLOv8).

Pipeline:

1. Dataset etiquetado en formato YOLO (`.txt` por imagen).
2. Entrenamiento del detector.
3. Inferencia: imagen $\rightarrow$ cajas + clases.
4. Exportación para despliegue (p.ej., TorchScript).

Dataset (origen, cantidad, clases)

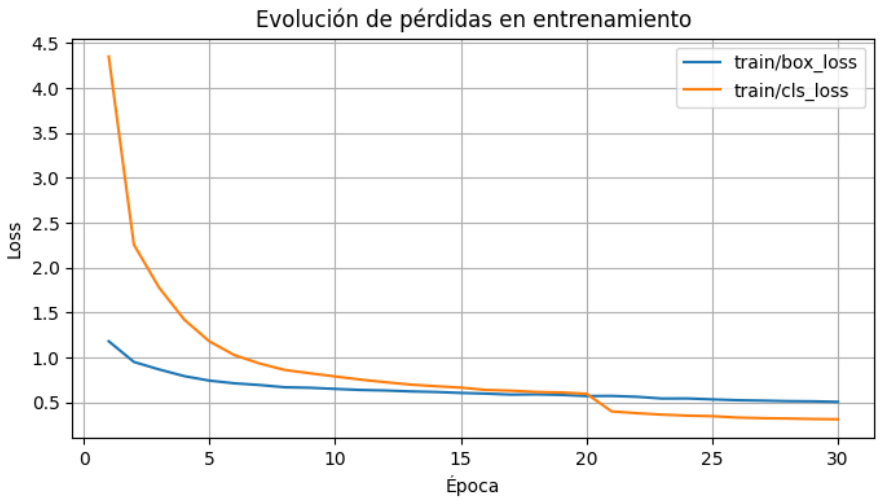
- **Origen:** Versión balanceada del dataset en Roboflow (clases con tamaño similar), basada en el dataset original de Kaggle.
- **Formato:** imágenes + anotaciones en formato YOLO (`class x_center y_center width height` normalizados).
- **Tamaño: 8099** imágenes.
- **Clases: 15** clases: *Green Light, Red Light, Speed Limit 10, Speed Limit 20, Speed Limit 30, Speed Limit 40, Speed Limit 50, Speed Limit 60, Speed Limit 70, Speed Limit 80, Speed Limit 90, Speed Limit 100, Speed Limit 110, Speed Limit 120, Stop*.
- **Casos de uso:**: (ej.: asistencia a la conducción/ADAS, simuladores, analítica vial, prototipos embebidos).

Split	**# Imágenes**	**%**
Train	6758	83 %
Valid	802	10 %
Test	539	7 %

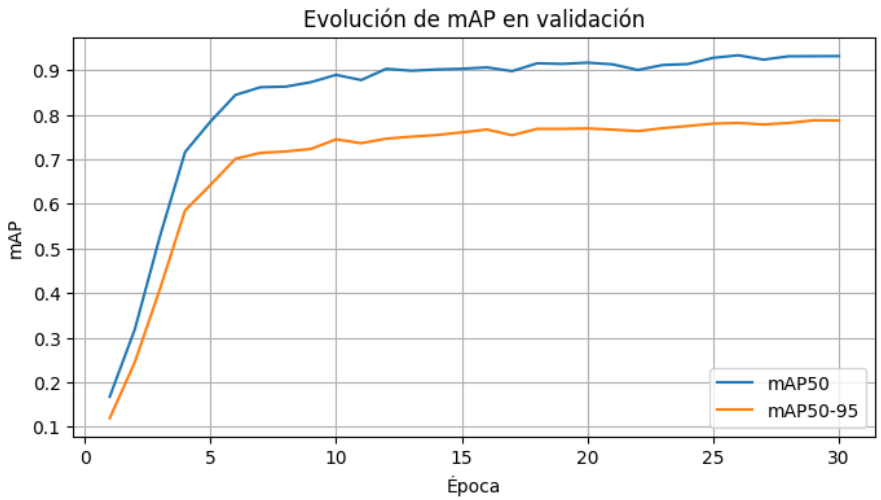
Arquitectura y entrenamiento

- **Modelo:** YOLOv8 (Ultralytics) **Variante:** s (n/s/m/...)
- **Input:** 512×512 **Clases:** 15 **Imágenes:** 8099
- **Tarea:** detección de objetos (caja delimitadora + clase + confianza)
- **Datos:** Train 6758 (83 %), Val 802 (10 %), Test 539 (7 %)
- **Hiperparámetros:**
  - Épocas: 30 Batch: 16 LR: 0.001 `imgsz`: 640
  - Optimizador: SGD Early stopping: 15
  - Aumentos (ej.): *flip(H), scale, rotation, blur, brightness*
  - Hardware: GPU (GPU/CPU, RAM)
- **Métricas reportadas: mAP@0.5**, **mAP@0.5:0.95**, precisión, recall

Métricas y gráficas de entrenamiento



mAP por época



Pérdida (loss) por época

- Mejor checkpoint: `best.pt` (época: 30)
- Resultados finales: mAP@0.5 = 0.932 mAP@0.5:0.95 = 0.787
- Precisión = 93.6
- Recall = 87.1

Inferencia en Python (demo)

- Entrada: imagen / webcam / video del simulador
- Salida: cajas + clase + confianza dibujadas sobre el frame

Snippet (ejemplo):

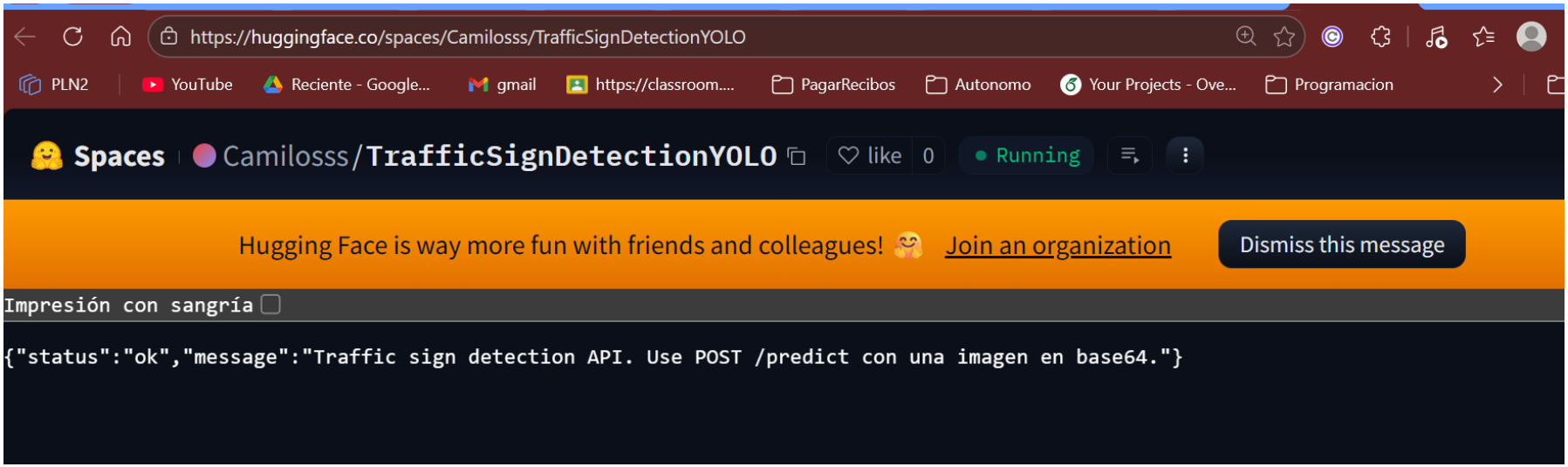
```
model = YOLO("best.pt")  o "best.torchscriptïmg = cv2.imread("demo.j

t0 = time.time() res = model.predict(img, conf=0.25, imgsz=640) dt
= (time.time() - t0) * 1000
annotated = res[0].plot() cv2.imwrite("demo_out.jpg", annotated)print(f" L
dt : ,1f ms")
```

Demo visual en un video

Entrada

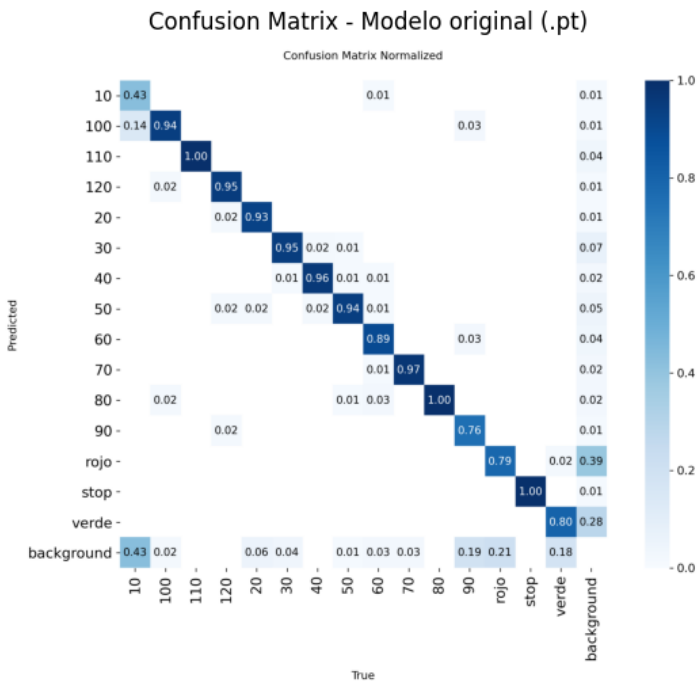
Salida con detecciones

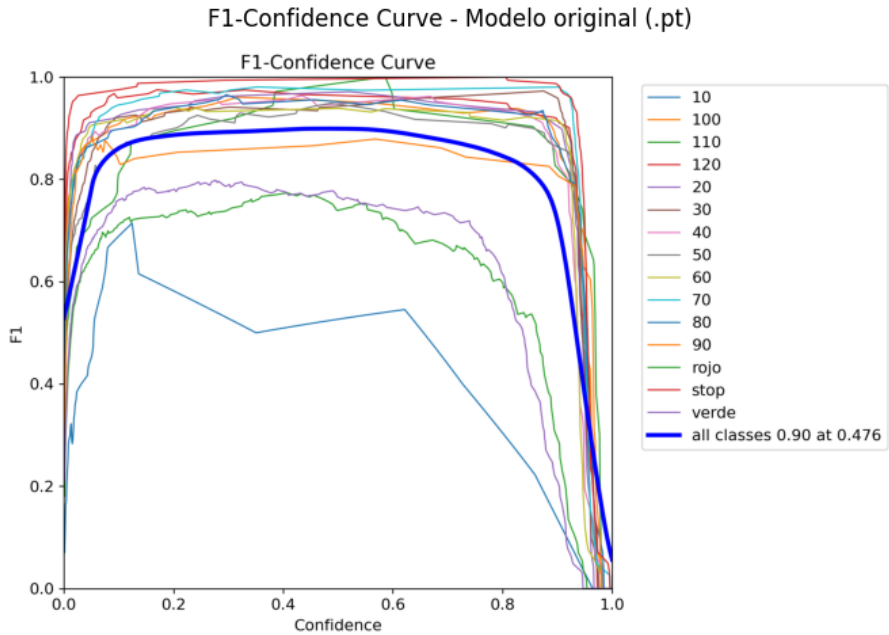
Despliegue en HuggingFace



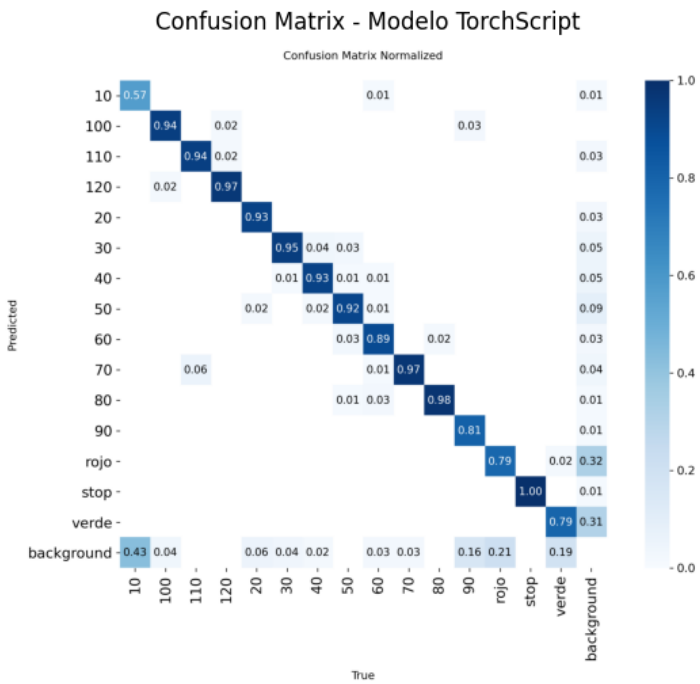
- **Objetivo:** publicar el modelo y/o demo para inferencia.
- **Repositorio:** `https://huggingface.co/spaces/Camilosss/TrafficSignDetectionYOLO`
- **Contenido:**
  - Pesos (`best.pt` / `.torchscript`)
  - `README.md` con instrucciones

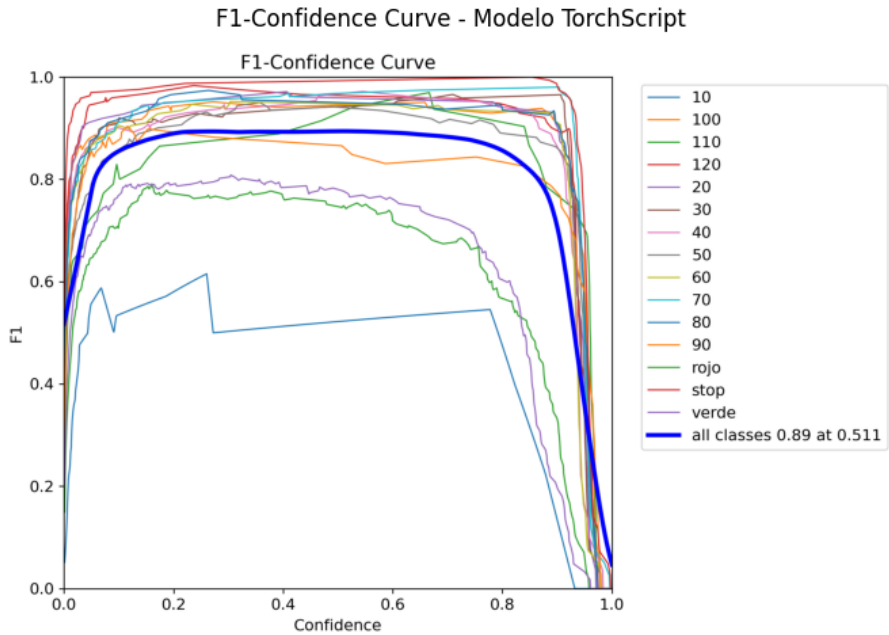
Resultados


Confusion Matrix - Modelo original (.pt)
Confusion Matrix Normalized
Predicted
True
10
100
110
120
20
30
40
50
60
70
80
90
rojo
stop
verde
background


F1-Confidence Curve - Modelo original (.pt)
F1-Confidence Curve
F1
Confidence
all classes 0.90 at 0.476

Resultados-TorchScript



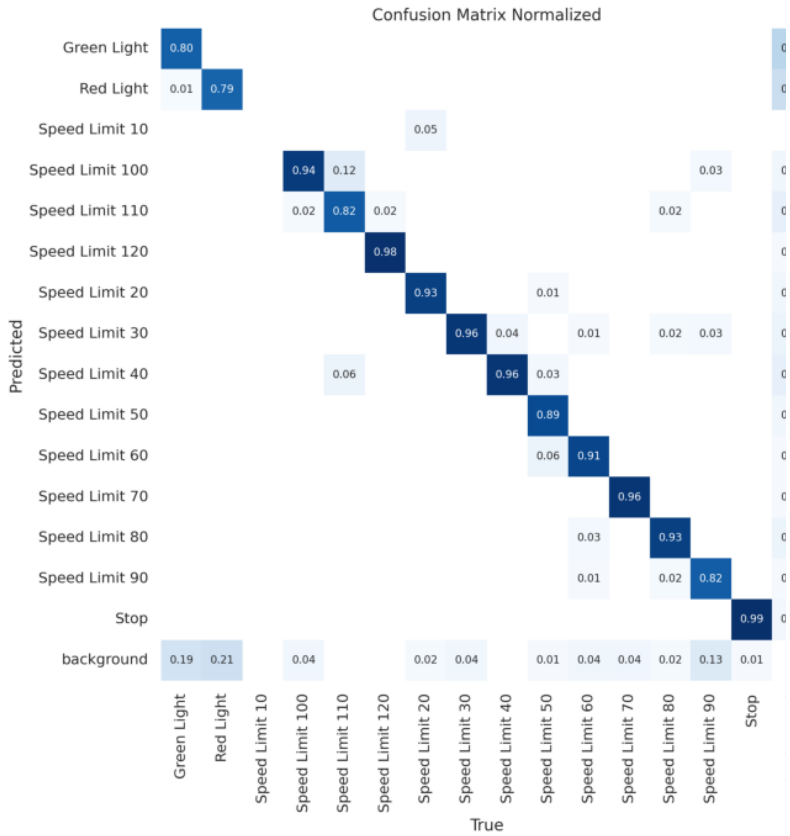


Limitaciones

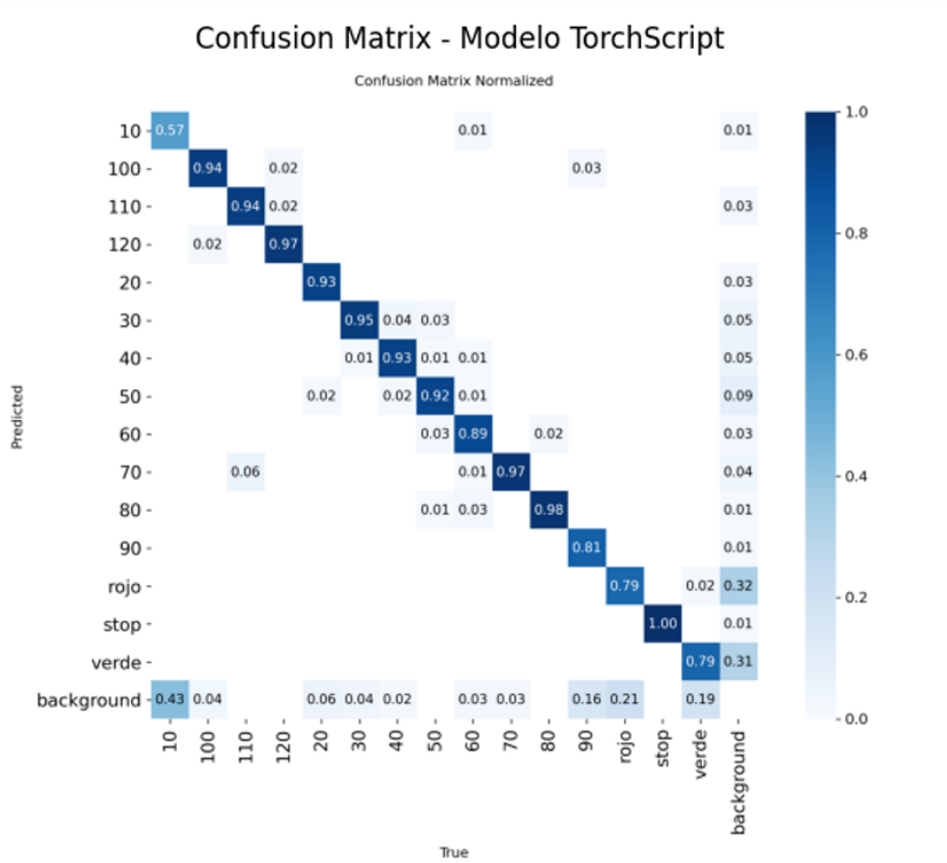
- **Variedad del dataset:** más tipos de señales.
- **Condiciones difíciles:** baja iluminación, motion blur, oclusiones, distancia grande.
- **Falsos positivos:** objetos con formas/colores similares a señales.
- **Dominio:** si el dataset es de simulador, puede no generalizar perfecto al mundo real (y viceversa).
- **Tiempo real:** en CPU puede bajar FPS si se usa alta resolución.

Estado del arte: enfoques y modelos

Evolución del enfoque



Comparación

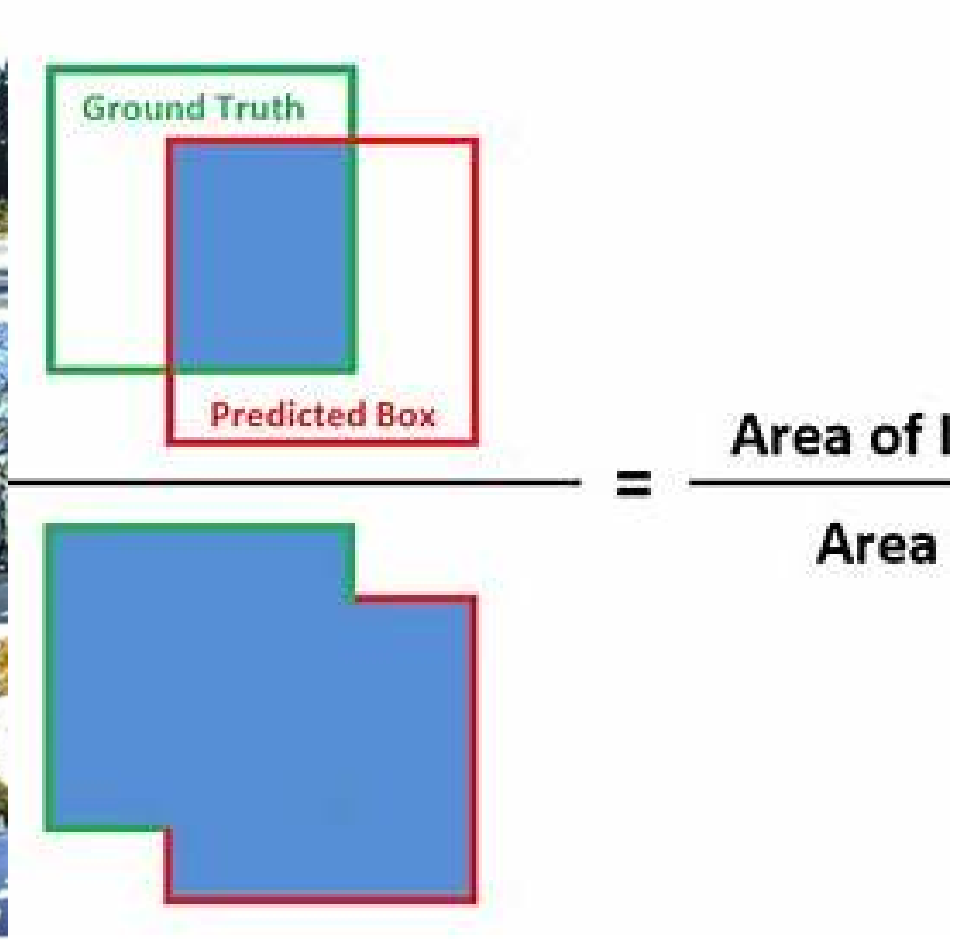


Estado del arte: datasets y evaluación

Datasets de referencia

GTSDB y Mapillary

Evaluación en detección



mAP + IoU + FPS

Conclusiones

- Se entrenó un detector de señales basado en YOLO para **localizar y clasificar** señales.
- La exportación (TorchScript/LiteRT) permite **mejor portabilidad** y potencial mejora en rendimiento.
- Se implementó inferencia en Python con medición de latencia/FPS.
- El despliegue en HuggingFace facilita compartir el modelo y reproducir la demo.

Trabajo futuro:

- Aumentar dataset y balance por clase; más escenarios (noche/lluvia/desenfoque).
- Evaluación por clase (mAP por clase) + matriz de confusión.
- Optimización para embebidos: cuantización, pruning, TensorRT/TFLite.

Referencias I

- UN-GCPDS. *ProcesamientoDigitalImagenes*. GitHub repository. Disponible en: https://github.com/UN-GCPDS/ProcesamientoDigitalImagenes
- Ultralytics. *YOLO Documentation*. Disponible en: https://docs.ultralytics.com/.
- Roboflow. *Roboflow Documentation*. Disponible en: https://roboflow.com/.
- Ultralytics. *YOLOv8 Documentation*. https://docs.ultralytics.com/models/yolov8/
- German Traffic Sign Detection Benchmark (GTSDB). https://benchmark.ini.rub.de/gtsdb_dataset.html
- Ertler, C. et al. (2020). *The Mapillary Traffic Sign Dataset for Detection and Classification on a Global Scale*. https://arxiv.org/abs/1909.04422

Referencias II

- Ji, B. et al. (2024). *Improved YOLOv8 for small traffic sign detection*. (ScienceDirect)
- Zhang, L. et al. (2025). *TSD-DETR: A lightweight real-time detection transformer for traffic sign detection*. (ScienceDirect)
- Darabi, P. K. *Traffic Signs Detection Using YOLOv8* (Kaggle Notebook). `https://www.kaggle.com/code/pkdarabi/traffic-signs-detection-using-yolov8`