



# HARDVISION

IDENTIFICACIÓN INTELIGENTE DE  
COMPONENTES HARDWARE

Proyecto final CEIABDTA - 2024/2025

Iván Falcón Monzón





# ÍNDICE

## INTRODUCCIÓN

Presentación del problema, la solución HardVision y nuestros principales logros

01

## PROBLEMA ACTUAL VS. SOLUCIÓN HARDVISION

Contraste entre los desafíos de identificación manual y la eficiencia de la IA

02

## CONTEXTO Y JUSTIFICACIÓN

La necesidad del proyecto y los objetivos específicos que nos propusimos alcanzar

03

## METODOLOGÍA

El proceso de desarrollo estructurado, desde la concepción hasta la implementación

04

## HERRAMIENTAS UTILIZADAS

Tecnologías y frameworks esenciales que impulsaron el proyecto

05



# ÍNDICE

## FORMATO DE ANOTACIÓN YOLO

La estructura de los datos de entrenamiento que el modelo utiliza para aprender

06

## ENTRENAMIENTO DEL MODELO

Cómo se llevó a cabo el aprendizaje del modelo y sus parámetros clave

07

## MÉTRICAS

Indicadores clave de rendimiento para evaluar la precisión y fiabilidad del modelo

08

## EL MAP: LA MÉTRICA CLAVE

Explicación detallada del Mean Average Precision y su relevancia en detección de objetos

09

## VISUALIZACIÓN Y ANÁLISIS DE RESULTADOS

Cómo se interpretaron los datos y el comportamiento del modelo

10



# ÍNDICE



```
graph TD; A[ÍNDICE] --> B[PUESTA EN PRODUCCIÓN]; A --> C[RESULTADOS OBTENIDOS]; A --> D[CONCLUSIONES DEL PROYECTO]; A --> E[MEJORAS FUTURAS];
```

## PUESTA EN PRODUCCIÓN



La arquitectura y funcionalidades clave de la aplicación web HardVision

11

## RESULTADOS OBTENIDOS



Presentación de los logros del modelo y la aplicación en entornos reales

12

## CONCLUSIONES DEL PROYECTO



Síntesis de los principales logros y el impacto de HardVision

13

## MEJORAS FUTURAS



Próximos pasos y potencial de expansión del proyecto HardVision

14



01

## INTRODUCCIÓN Y SUMARIO

### SOLUCIÓN

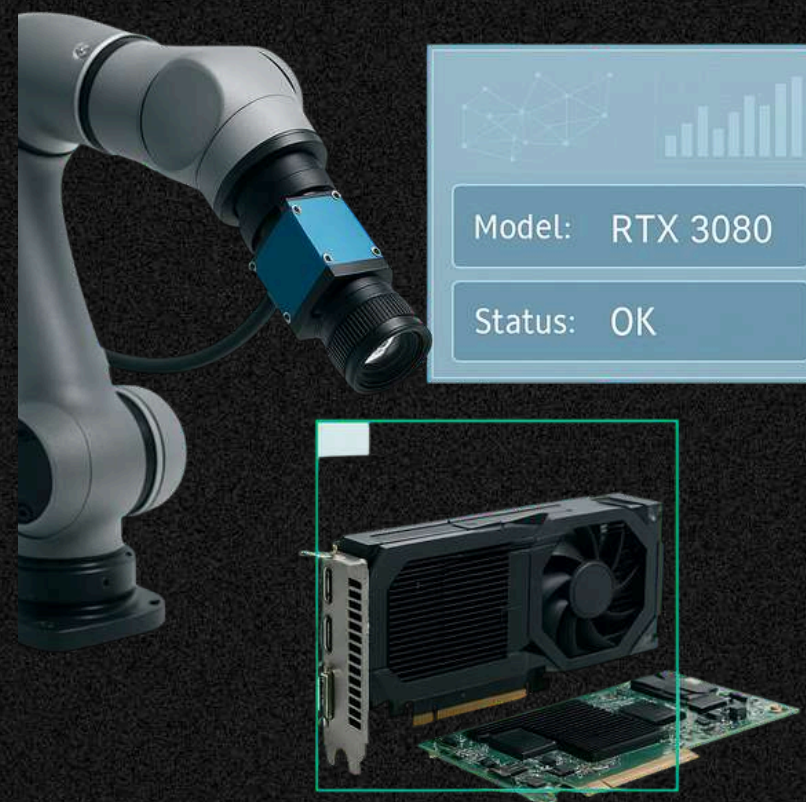
- IA para identificación visual (YOLOv8n)
- App web funcional

### PROBLEMA A RESOLVER

- Detección hardware: complejo y manual
- Necesidad de una herramienta visual

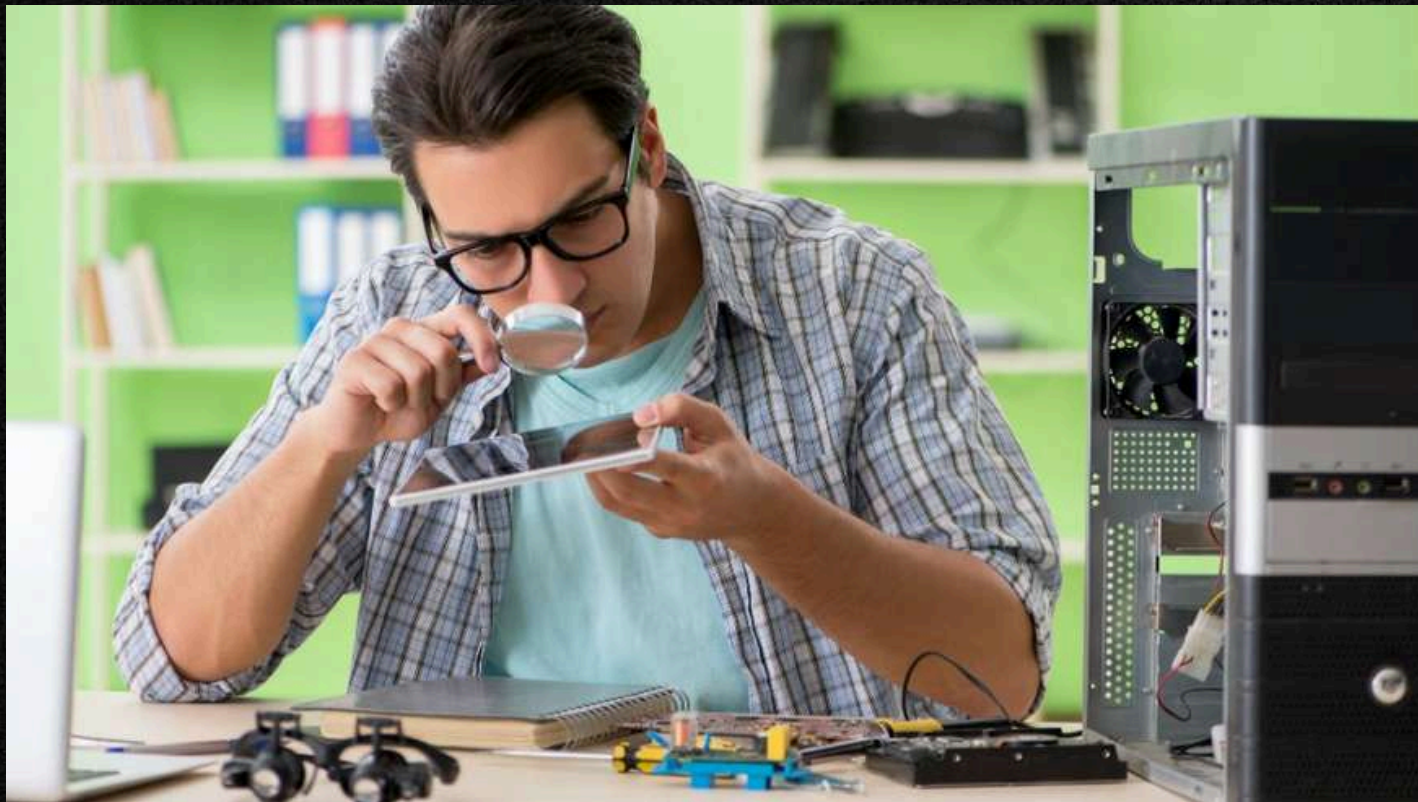
### LOGROS CLAVE

- Detección en tiempo real
- Varios datasets
- App web intuitiva (multilingüe)
- Entorno portátil y fácil

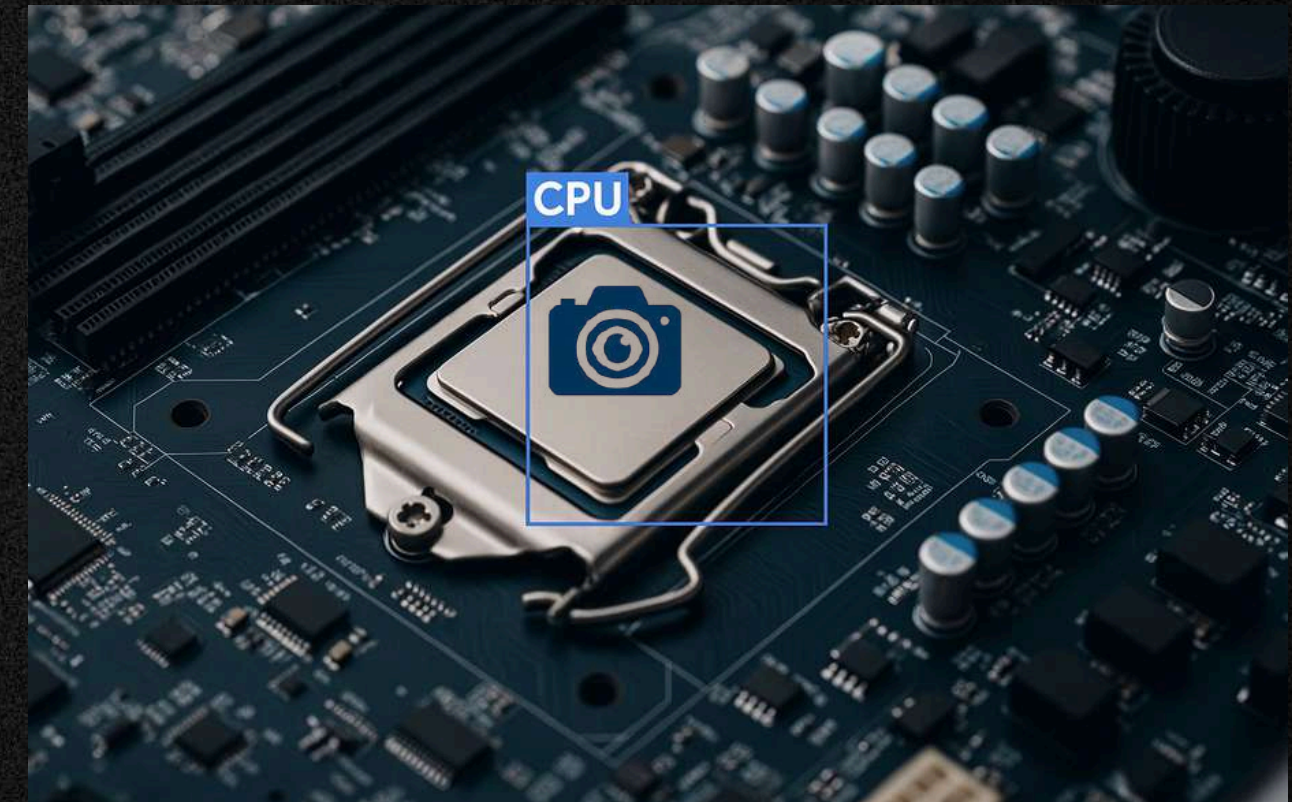




## PROBLEMA ACTUAL VS. SOLUCIÓN HARDVISION



- Manual
- Lento
- Errores humanos
- Falta de estandarización



- Automatizado
- Rápido
- Preciso (IA)
- Estandarizado



## CONTEXTO Y JUSTIFICACIÓN

- Digitalización actual (educación/empresas)
- Necesidad de gestionar activos hardware
- Solución eficiente y accesible

## OBJETIVO GENERAL

- Demostrar viabilidad IA para identificación visual de hardware



## OBJETIVOS ESPECÍFICOS

- Desarrollar modelo YOLOv8n
- Crear dataset personalizado
- Implementar API y web responsive
- Diseñar instalador local

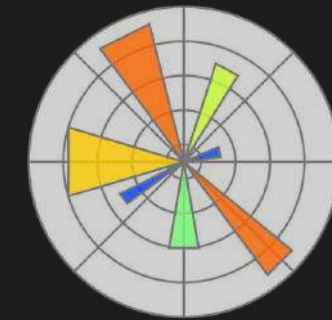




## Metodología: El Camino del Desarrollo



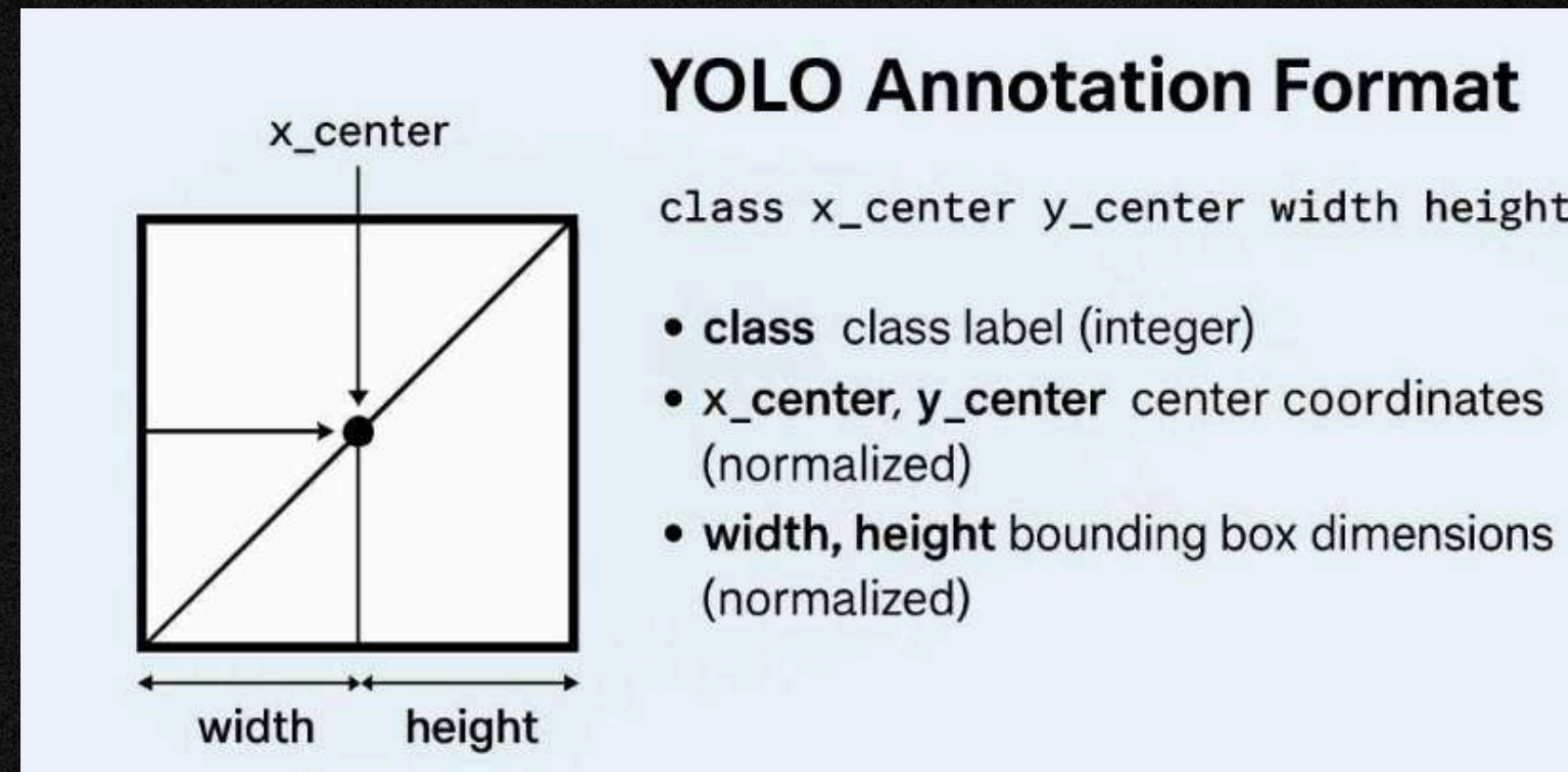




- **Dataset & Anotación:** Roboflow, Python
- **Visión:** Ultralytics YOLOv8, PyTorch, OpenCV
- **Desarrollo Web:** Flask, HTML/CSS/JS
- **Análisis & Visualización:** Matplotlib, Pandas, Scikit-learn, Tableau Public
- **Entornos:** Visual Studio Code y Google Colab



# FORMATO DE ANOTACIÓN YOLO



- **Archivo por Imagen:** Un .txt asociado a cada .jpg
- **Formato Línea:** ID\_Clase | X\_Centro | Y\_Centro | Ancho | Alto
- **Valores Normalizados:** Coordenadas y dimensiones de 0 a 1
- **Beneficio Clave:** Independencia de la resolución y escalabilidad

```
8 0.7521505125 0.6712586265625 0.742892275 0.6436991546875 0.43222375
0.758844315625 0.44418630312499996 0.787713878125 0.7521505125
0.6712586265625
```

```
8 0.7615349609375001 0.6966150515625 0.7544216484375 0.6702407078125
0.4457221546875 0.7847248765625 0.4567949875 0.8125901515624999
0.7615349609375001 0.6966150515625
```

```
3 0.5296875 0.55703125 0.290625 0.278125
```

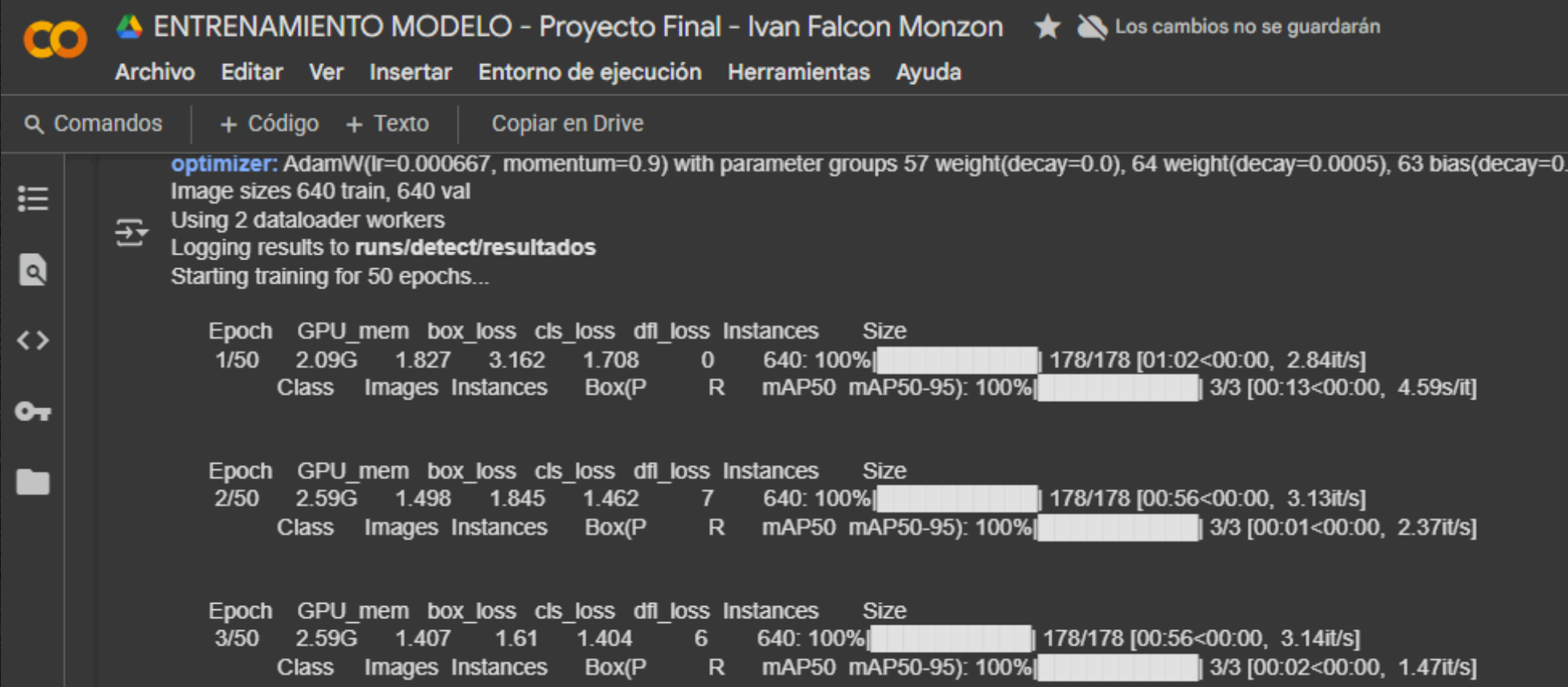
```
5 0.42611936562499997 0.692249253125 0.28671098281249996
0.33766483593749996 0.22642946562500002 0.3670831765625 0.36454841875
0.7146284390625001 0.42611936562499997 0.692249253125
```



# ENTRENAMIENTO DEL MODELO: EL PROCESO PRÁCTICO

El archivo `data.yaml` quedó configurado con 11 clases:

- `atx_12v`
- `atx_power`
- `cpu`
- `cpu-slot`
- `fan-bracket`
- `m.2-ssd-slot`
- `pcie-slot`
- `ram`
- `ram-slot`
- `ssd`
- `unknown`



The screenshot shows the Ultralytics YOLOv8 training interface. The top bar displays the project name "ENTRENAMIENTO MODELO - Proyecto Final - Ivan Falcon Monzon" and a warning "Los cambios no se guardarán". The main area shows the command line with the following text:

```
optimizer: AdamW(lr=0.000667, momentum=0.9) with parameter groups 57 weight(decay=0.0), 64 weight(decay=0.0005), 63 bias(decay=0.0)
Image sizes 640 train, 640 val
Using 2 dataloader workers
Logging results to runs/detect/resultados
Starting training for 50 epochs...
```

Below the command line, the training progress is shown for the first three epochs:

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
1/50	2.09G	1.827	3.162	1.708	0	640: 100% [01:02<00:00, 2.84it/s]
						Class Images Instances Box(P R mAP50 mAP50-95): 100% [00:13<00:00, 4.59s/it]
2/50	2.59G	1.498	1.845	1.462	7	640: 100% [00:56<00:00, 3.13it/s]
						Class Images Instances Box(P R mAP50 mAP50-95): 100% [00:01<00:00, 2.37it/s]
3/50	2.59G	1.407	1.61	1.404	6	640: 100% [00:56<00:00, 3.14it/s]
						Class Images Instances Box(P R mAP50 mAP50-95): 100% [00:02<00:00, 1.47it/s]

- **ultralytics:** implementación oficial de YOLOv8.
- **openimages:** para descarga de clases desde el dataset Open Images.
- **fiftyone:** para gestión visual y estructural de datasets.





# ENTRENAMIENTO DEL MODELO: MÉTRICAS

Métricas	Precisiones
Precisión final	0.9246
Recall final	0.9065
mAP@0.5 final 0.9206	0.9206
mAP@0.5:0.95 final	0.7909
El valor final de fitness compuesto	0.8038

- GPU Tesla T4
- Épocas: epochs=50 (50 épocas)
- Batch Size: batch=16 (Cada época procesa el dataset en lotes de 16 imágenes).
- Imágenes de Entrenamiento: 1733 images
- Imágenes de Validación: 83 imágenes

- Formato: .jpg
- Resolución: 640 x 640

**UNKNOWN\_CONF\_THRESHOLD** = 0.35 # Umbral para considerar una detección como 'desconocido'



# COMPRENDIENDO EL MAP: LA MÉTRICA CLAVE

## mAP (mean Average Precision):

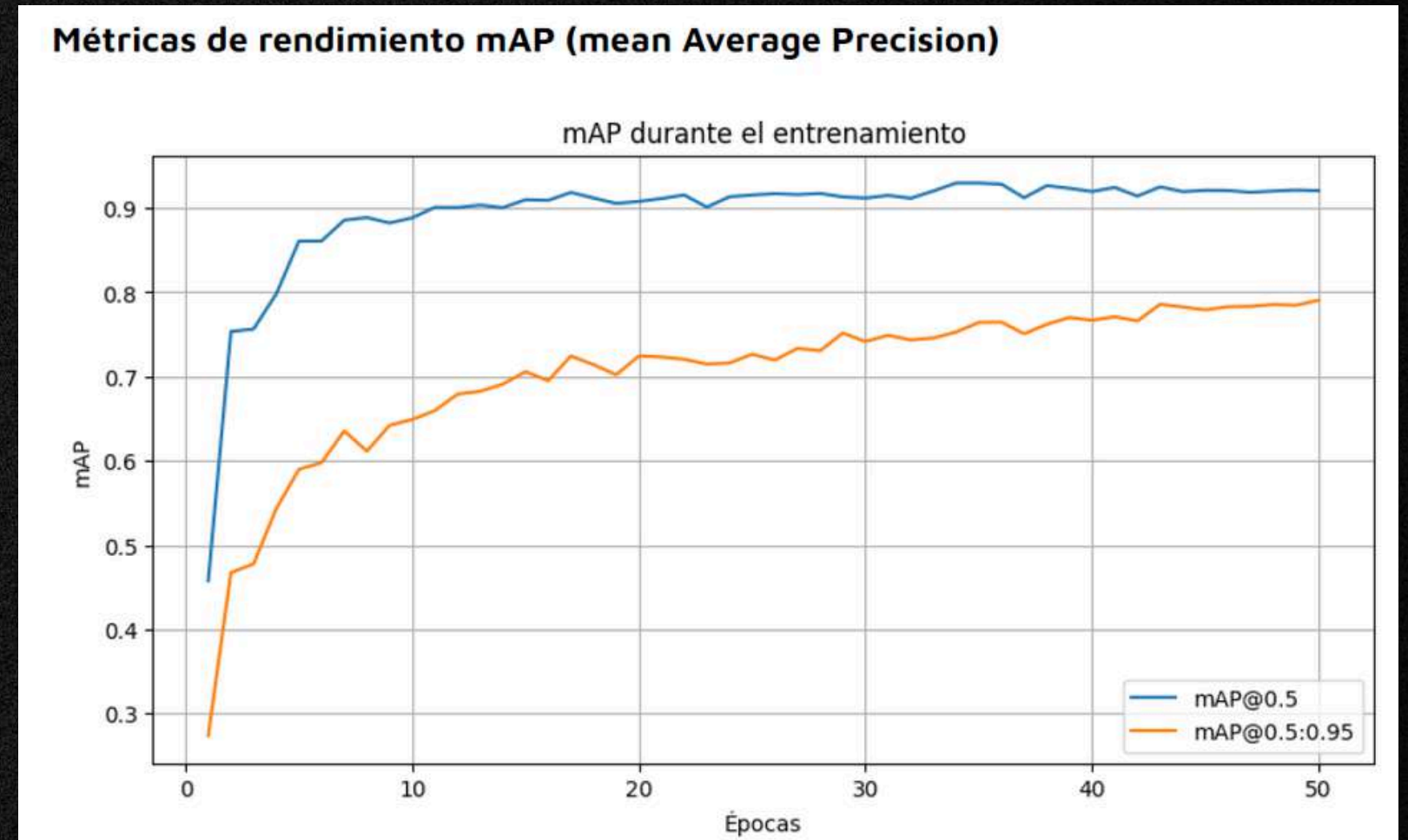
- Métrica clave para detección de objetos
- Representa la precisión media en todas las clases

## AP (Average Precision):

- Indica el equilibrio entre **Falsos Positivos** y **Falsos Negativos** por clase

## ¿Por qué importa?

- Evalúa el rendimiento general del modelo
- Valor alto en MAP = indica buen balance Precisión-Recall en todas las clases

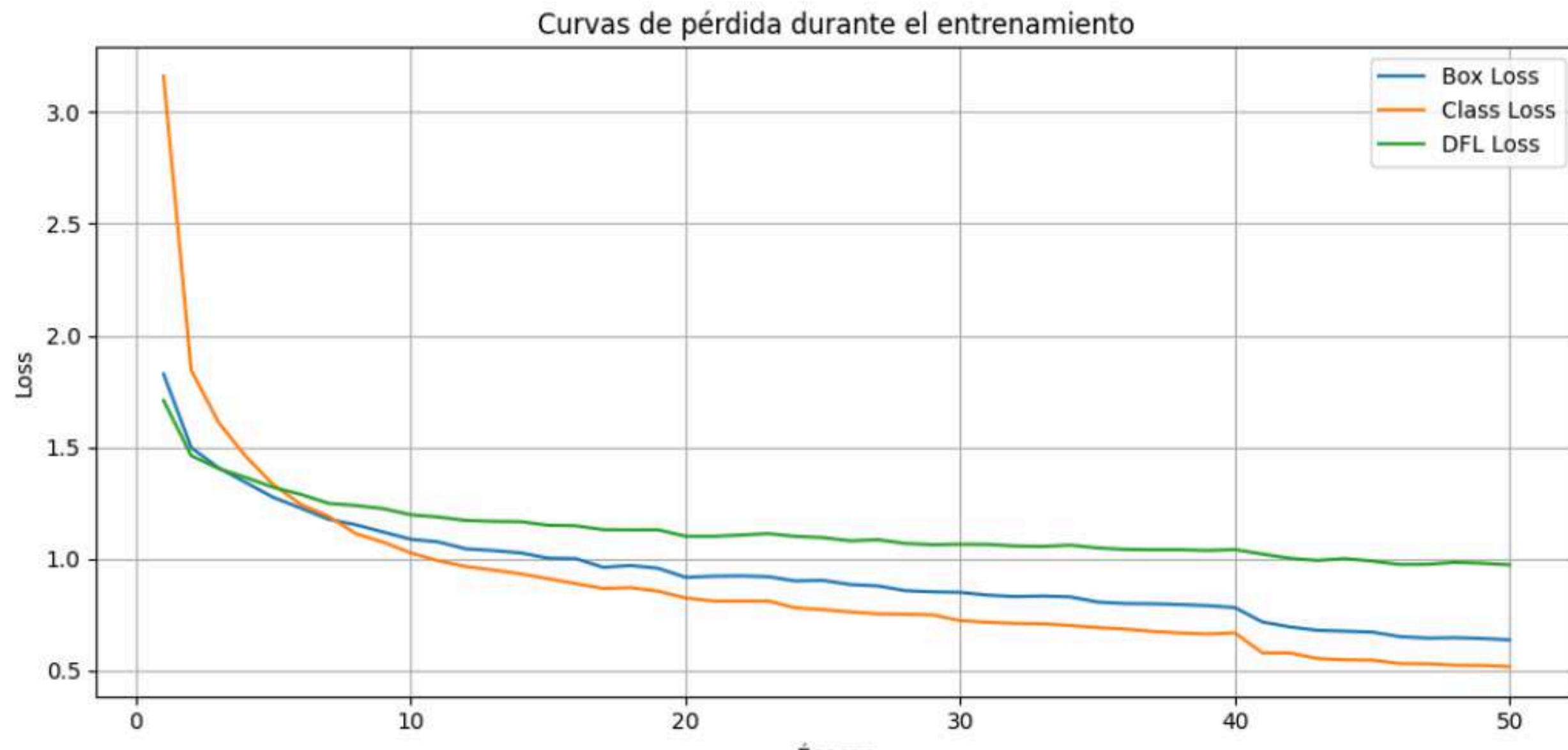




# VISUALIZACIÓN Y ANÁLISIS DE RESULTADOS

- **Herramientas:** Tableau Public, Matplotlib y Pandas

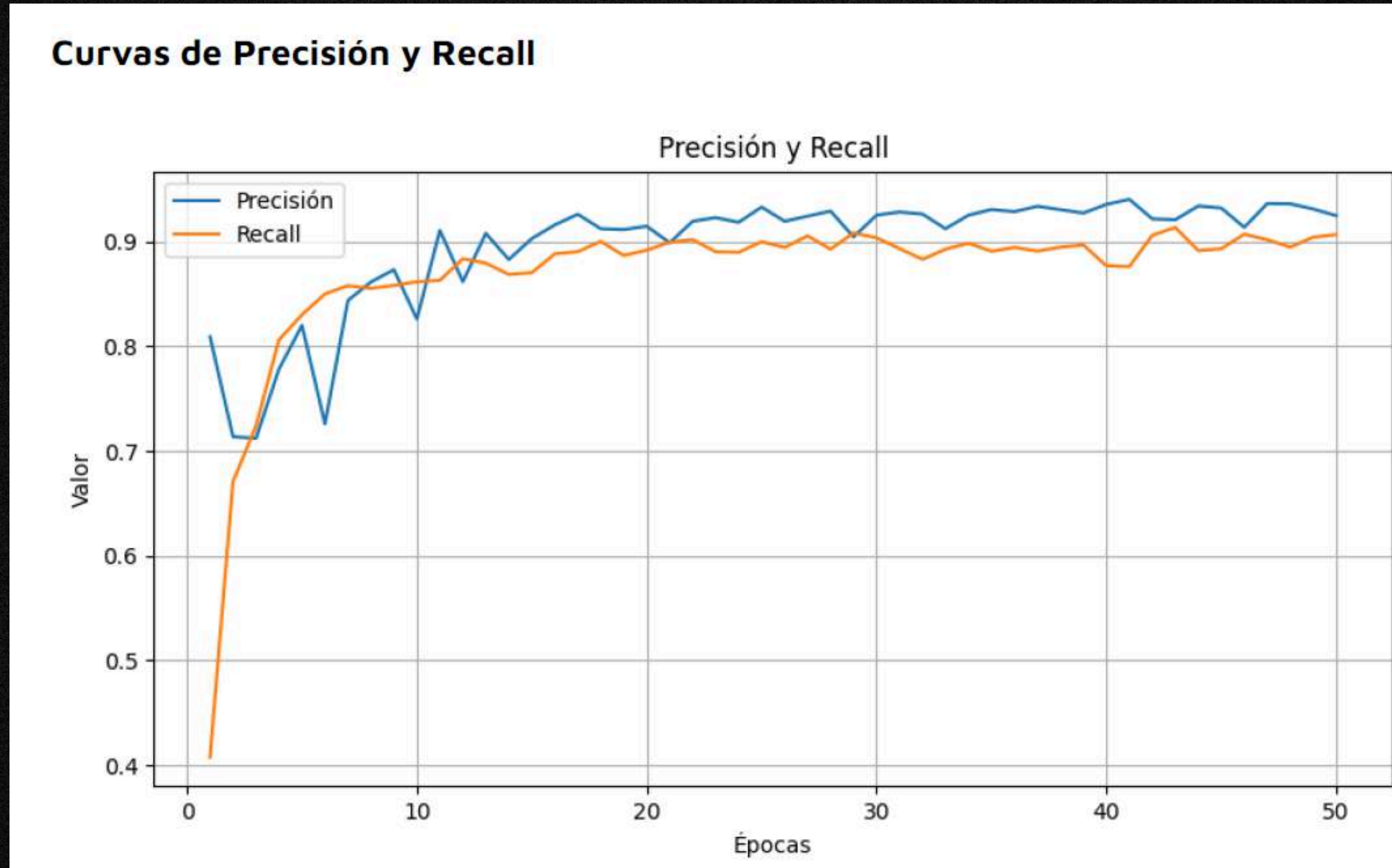
## Curvas de pérdida del entrenamiento





# VISUALIZACIÓN Y ANÁLISIS DE RESULTADOS

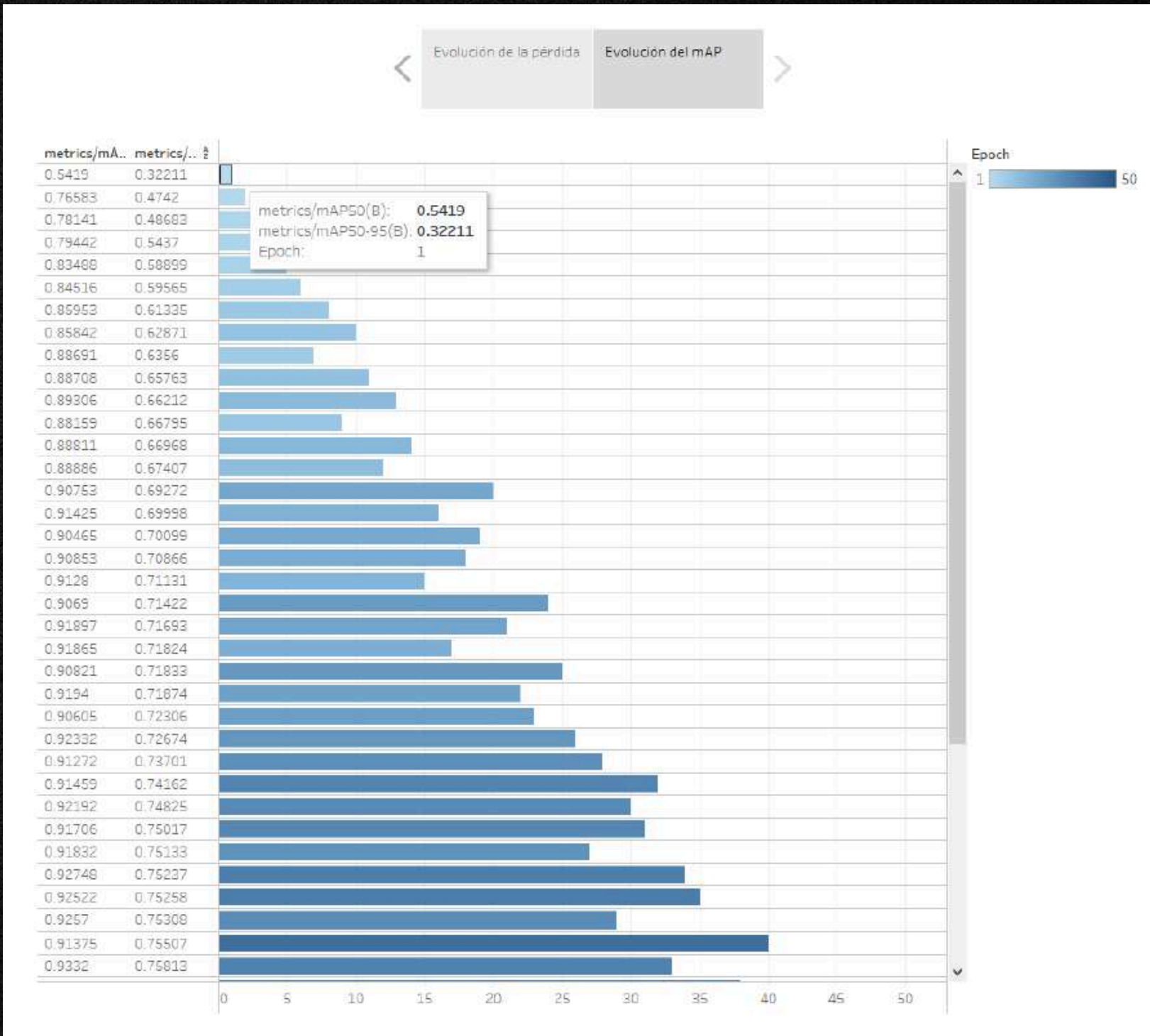
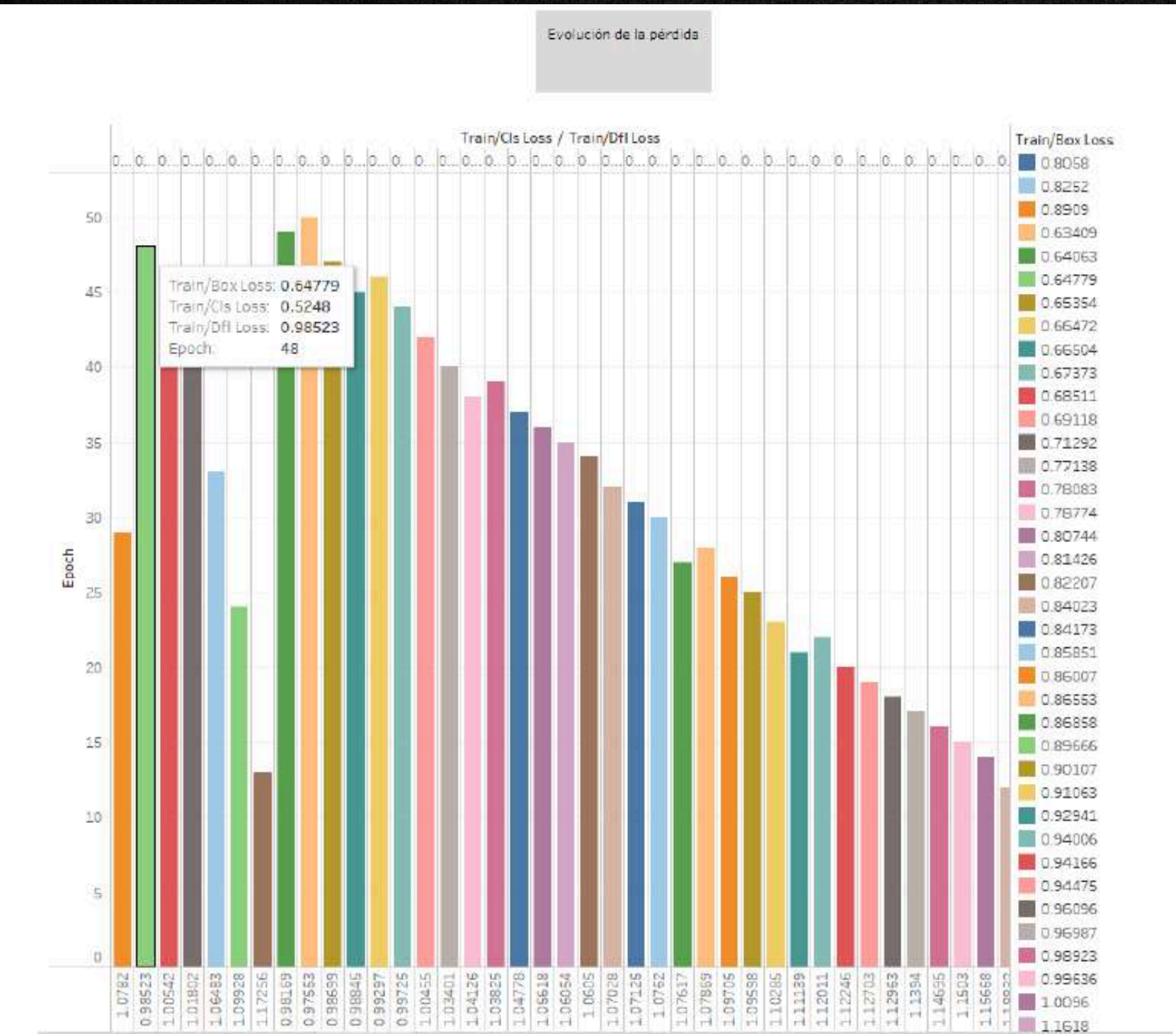
- **Herramientas:** Tableau Public, Matplotlib y Pandas





# VISUALIZACIÓN Y ANÁLISIS DE RESULTADOS

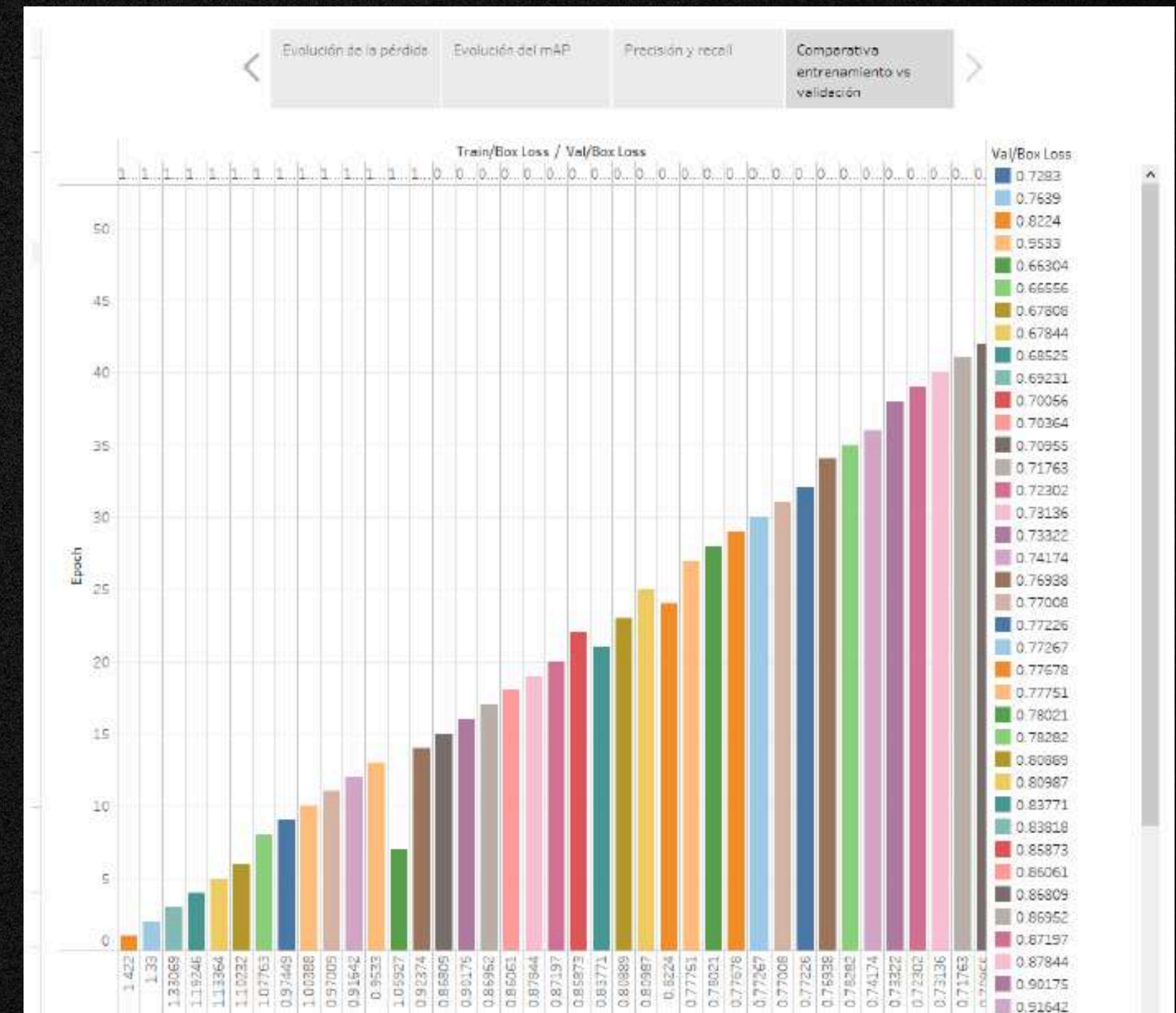
- **Herramientas:** Tableau Public, Matplotlib y Pandas





# VISUALIZACIÓN Y ANÁLISIS DE RESULTADOS

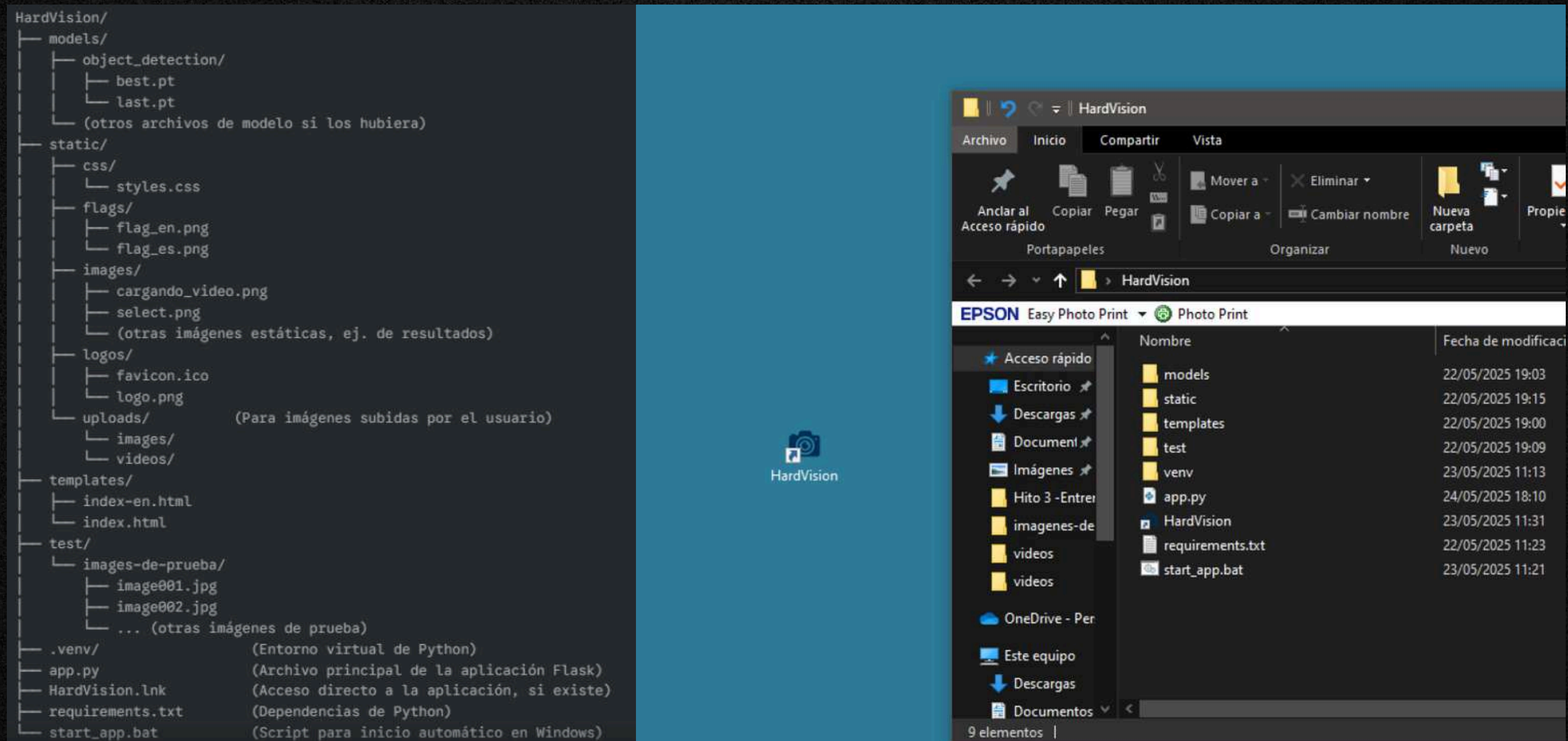
- **Herramientas:** Tableau Public, Matplotlib y Pandas





# PUESTA EN PRODUCCIÓN

- **Backend:** API REST con Flask, endpoints básicos para interacción con el modelo
- **Frontend:** Aplicación web de escritorio, interfaz amigable y multilingüe
- **Instalación:** Proceso automatizado para facilitar el despliegue en entornos locales

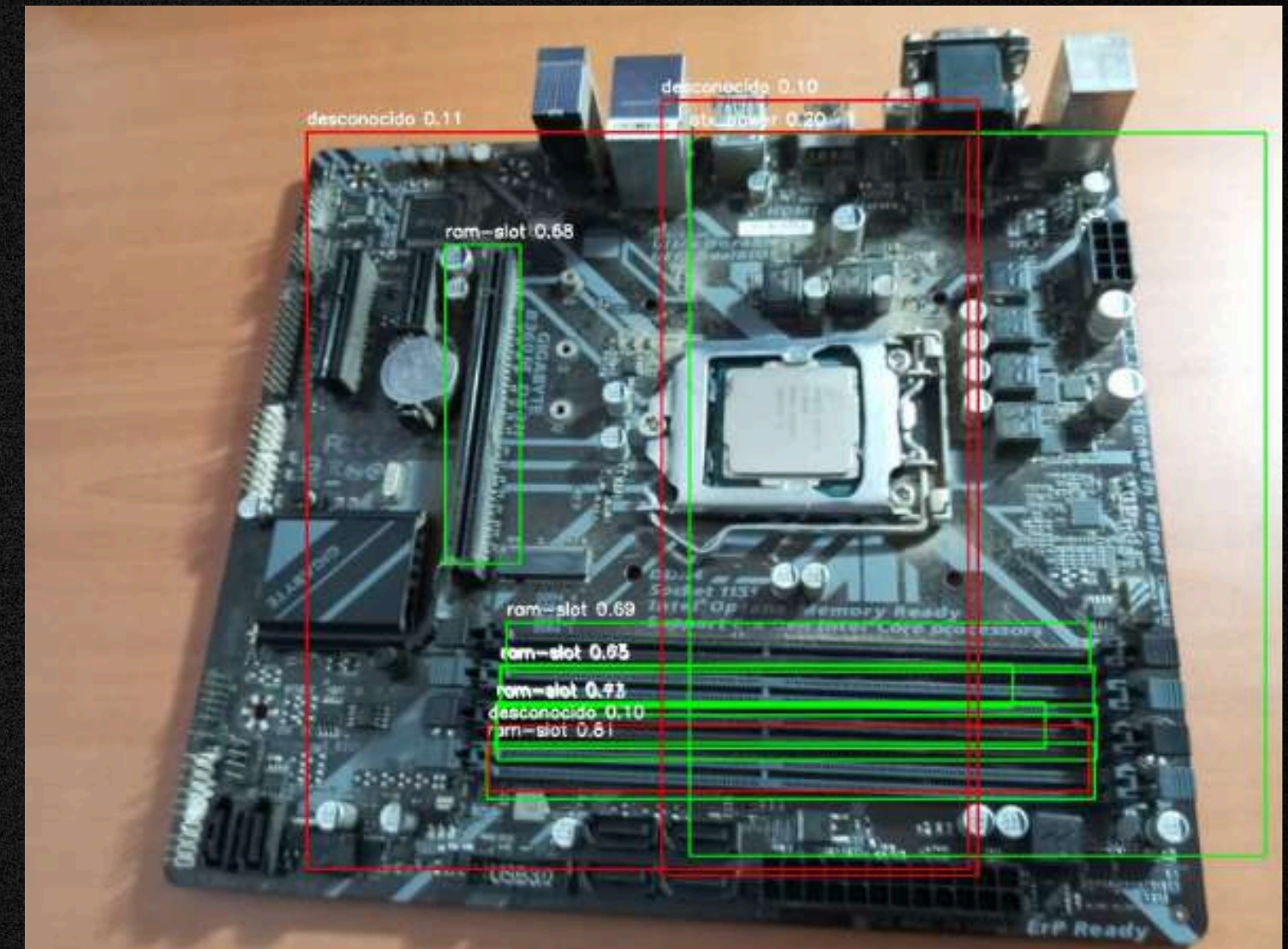




# RESULTADOS OBTENIDOS



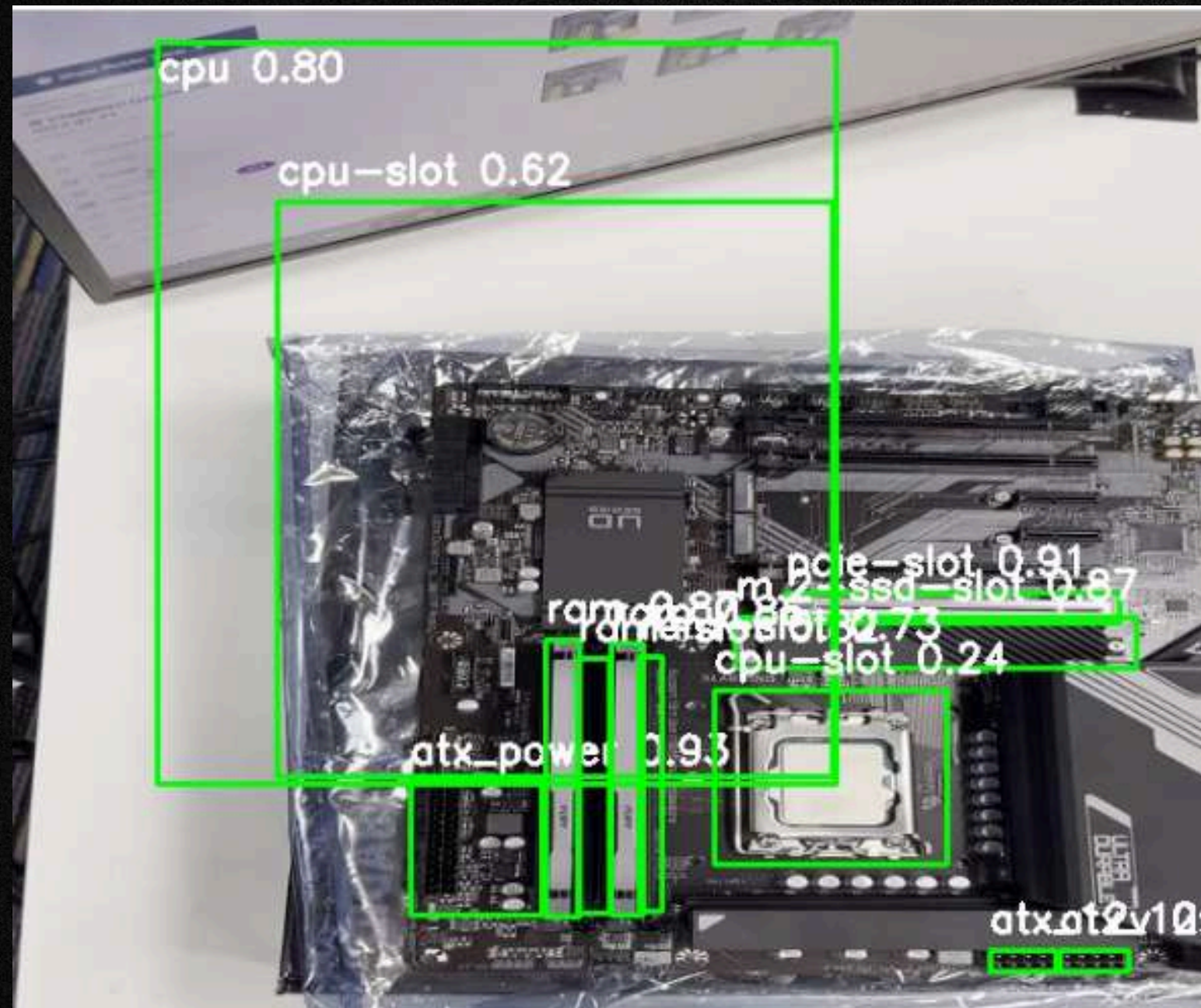
Resultado de inferencia sobre una imagen real de una placa base. El modelo detecta correctamente componentes clave como las ranuras RAM se observan etiquetas claras y cajas delimitadoras con alta confianza



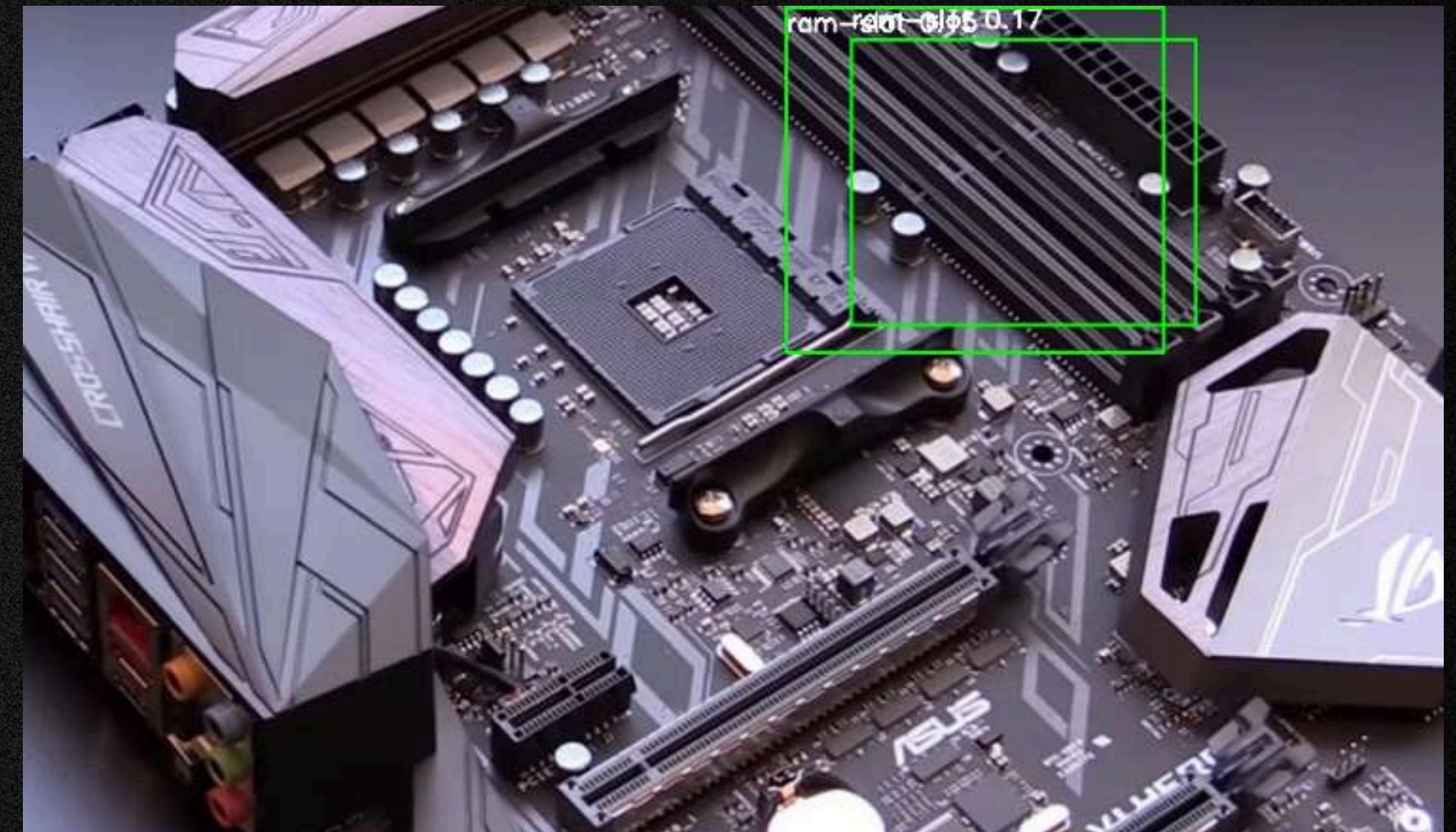
La misma placa base pero con otra colocación detecta más cosas



# RESULTADOS OBTENIDOS



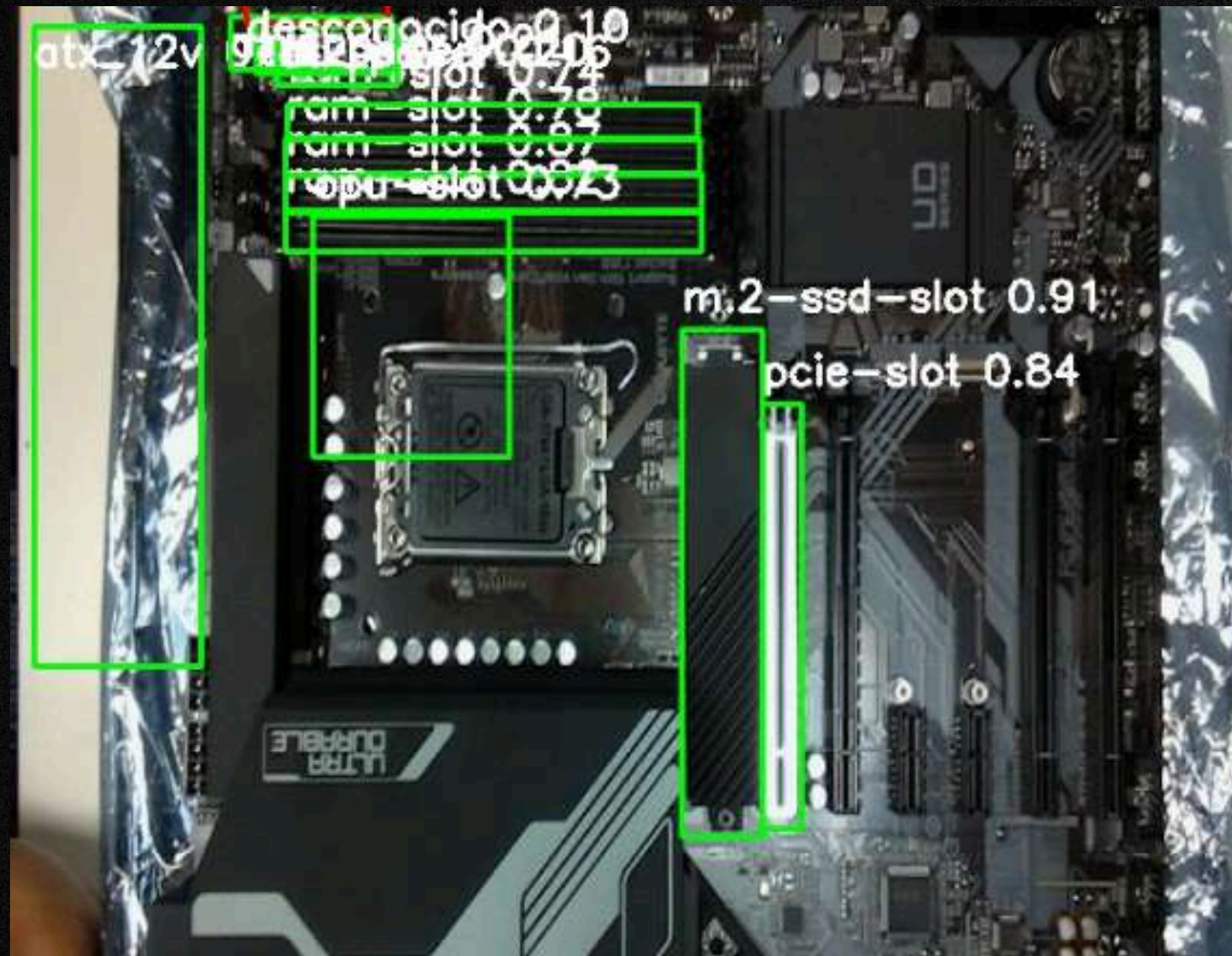
Otra placa base distinta, detecta cosas diferentes a la anterior



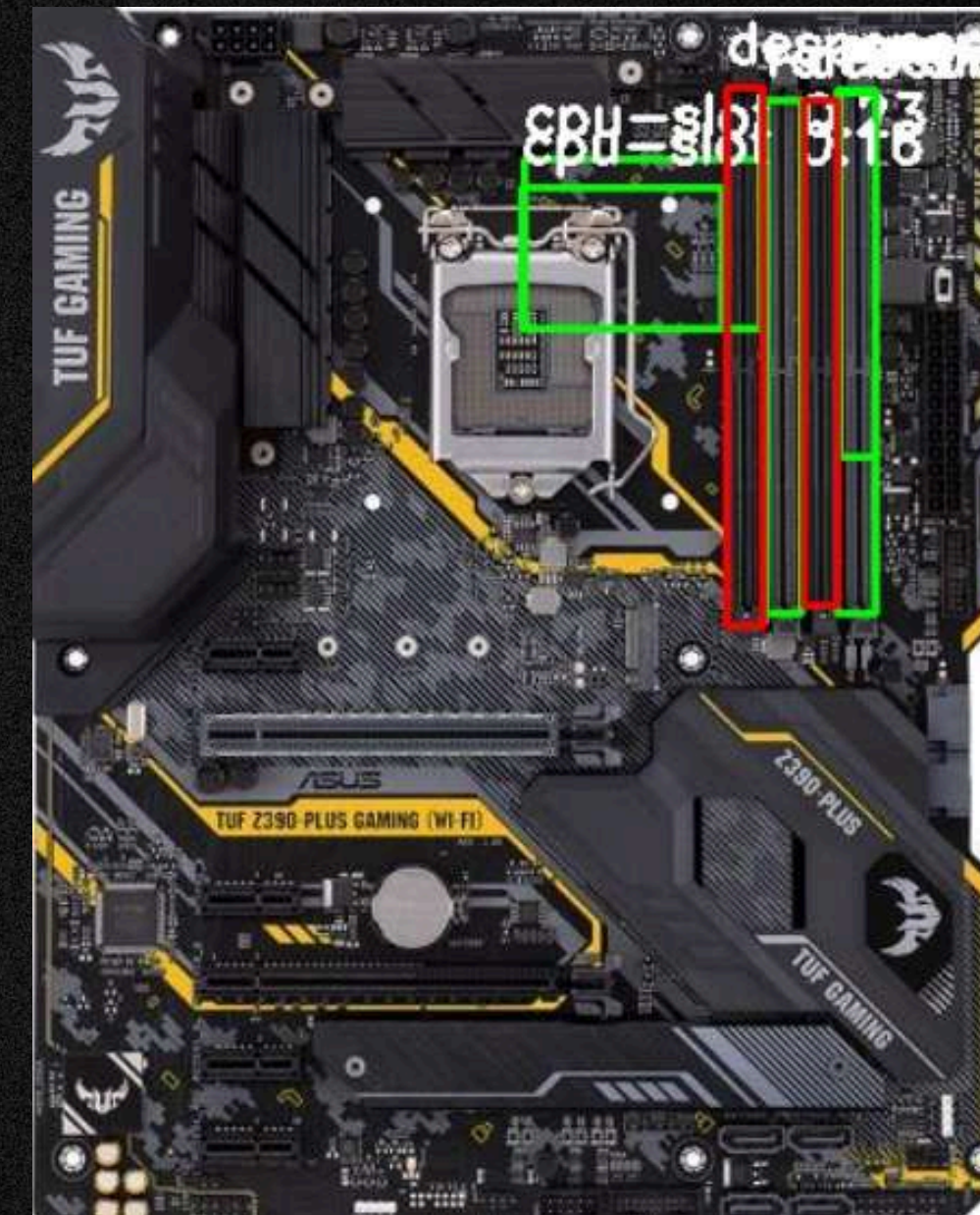
En esta otra imagen, sacada de internet, no detecta con tanta exactitud, está muy cerca



# RESULTADOS OBTENIDOS



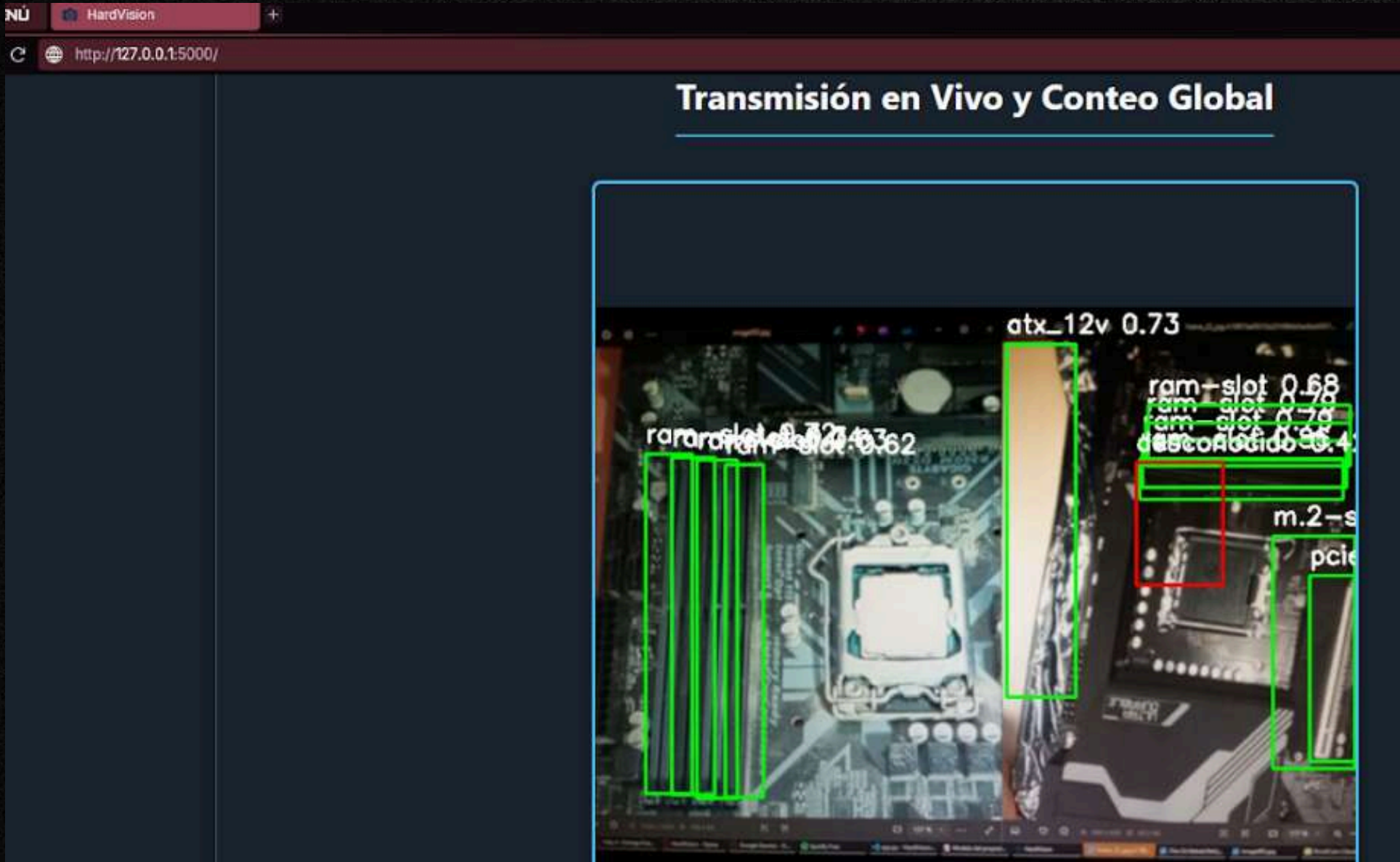
Otra placa diferente, detecta muchas clases con buena precisión



Placa sin fondo, sacada de internet, no logro el modelo un gran resultado



# RESULTADOS OBTENIDOS



Video funcionando en la página web con dos imágenes juntas pero diferente placa base

### Conteo de Objetos Detectados (acumulado del stream)

atx_12v:	atx_power:	cpu:
0	0	0
cpu-slot:	fan-bracket:	m.2-ssd-slot:
0	0	0
pcie-slot:	ram:	ram-slot:
0	0	4
ssd:	unknown:	desconocido:
0	0	0

Las cifras se actualizan cada 2 segundos.

Conteo de clases de la página web



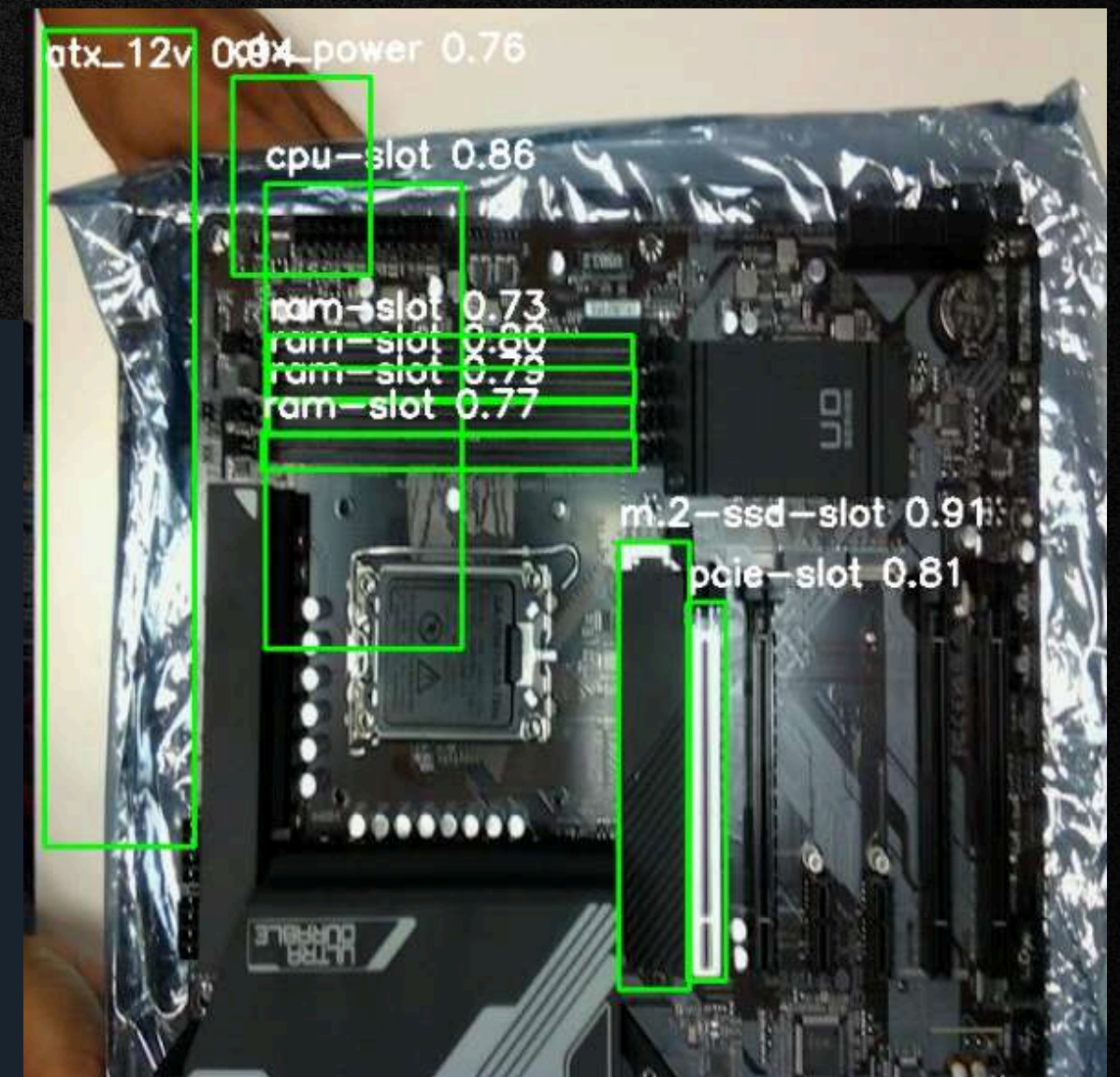
# CONCLUSIONES: LOS LOGROS DE HARDDVISION

- Viabilidad de IA para identificación hardware demostrada
- Detección precisa en tiempo real
- Dataset propio y relevante
- Aplicación web completa y usable
- Impacto: Herramienta funcional y accesible

Identificación de componentes hardware mediante YOLO y análisis predictivo



Desarrollado por Iván Falcón Monzón





# MEJORAS FUTURAS

**HardVision** demuestra la aplicabilidad de la IA para la identificación de hardware, mejorando la eficiencia en contextos reales

## Futuras mejoras:

- **Ampliación de clases** detectadas, optimización del modelo, integración con nuevos sistemas y ampliación de idiomas
- **Optimización Precisión:** Modelos más complejos (YOLOv8s/m), técnicas avanzadas
- **Segmentación Semántica:** Detección de contornos precisos
- **Historial de Detecciones:** Para auditoría e inventario
- **App Nativa:** Solución móvil independiente





# BIBLIOGRAFÍA / WEBGRAFÍA

## Herramientas y plataformas:

Roboflow – Gestión y anotación de datasets

- <https://roboflow.com/>

Tableau Public – Visualización interactiva de datos

- <https://public.tableau.com/>

Visual Studio Code – Entorno de desarrollo

- <https://code.visualstudio.com/>

Python – Lenguaje de programación

- <https://www.python.org/>

Flask – Framework web para Python

- <https://flask.palletsprojects.com/>

## Modelos y documentación técnica:

**Ultralytics YOLOv8 – Documentación oficial**

- <https://docs.ultralytics.com/>

**PyTorch – Framework de deep learning utilizado por YOLO**

- <https://pytorch.org/>

**OpenCV – Procesamiento de imágenes en Python**

- <https://opencv.org/>

**Scikit-learn – Herramientas de machine learning**

- <https://scikit-learn.org/>

**Matplotlib – Gráficas en Python**

- <https://matplotlib.org/>

**Pandas – Manipulación de datos en Python**

- <https://pandas.pydata.org/>



# REPOSITARIOS Y ENLACES

## **Carpeta compartida de Google Drive:**

[https://drive.google.com/drive/folders/1Xzf3wejBUWDtcBH4\\_ekeg6cCaAUSaqsC?usp=sharing](https://drive.google.com/drive/folders/1Xzf3wejBUWDtcBH4_ekeg6cCaAUSaqsC?usp=sharing)

## **Repositorio Github:**

[https://github.com/IvanFalconMonzon/CurEsplABD\\_ProyectoFinal\\_IvanFalconMonzon.git](https://github.com/IvanFalconMonzon/CurEsplABD_ProyectoFinal_IvanFalconMonzon.git)

## **Google colab (Entrenamiento del modelo y tablas predictivas):**

<https://colab.research.google.com/drive/11X6-vMe4TnmLKkDWmBzcQz4uRGPIjMRp?usp=sharing>

## **Dataset elegido:**

<https://universe.roboflow.com/gradresearch/gradresearch>

## **Otras opciones de datasets:**

<https://universe.roboflow.com/james-manalili/pc-parts-5uy7m/model/10>

<https://universe.roboflow.com/computer-vision-assembly-instructions/computer-hardware>



# DEMOSTRACIÓN



ENLACE