Proyecto Prácticas Profesionalizantes II

BGH Grupo Cables

Profesores:

* Caballero, Nicolás.
* Magaldi, Federico.
* Mirabete, Martín.

Alumnos:

* Constenla, Nahuel.
* Cussi, Marcelo.
* Godoy Del Castillo, Maria Florencia.
* Retamar, Cristian.

Documento Desarrollo Practicas Profesionalizantes II Proyecto Cables - BGH

**Resumen**

En este documento se detallarán los pasos realizados para el producto final , desde la definición de los objetivos de investigación, los pasos para el etiquetado, el entrenamiento del modelo y se analizaron además los resultados de las métricas.

# Objetivo del proyecto

Desarrollar un sistema de inteligencia artificial (IA) que detecte la correcta conexión de los cables con sus respectivos colores en la placa Main durante los procesos de ensamblaje, para asegurar la calidad del producto, reducir errores, mejorar la precisión, eficiencia, y reducir costos de retrabajo.

La detección deberá identificar que tanto el color del cable como la ubicación del mismo sea la correcta para cada modelo de equipo que se esté fabricando. Se tomara como una detección NO conforme cualquier otra distribución que no corresponda a especificada en la instrucción de trabajo o la falta de conexión del mismo.

**Objetivos específicos**

* Definir el modelo de entrenamiento y sus librerías.
* Recopilar imágenes de las placas main para el armado del data set.
* Etiquetar de imágenes con LabelIMG
* Entrenar el modelo y analizar las métricas.
* Realizar la interfaz para el usuario (GUI)

# Definición del modelo y sus librerías

El modelo YOLOv8m (You Only Look Once) es una arquitectura avanzada para la detección de objetos en imágenes y videos, diseñada para combinar precisión y velocidad en tareas de visión por computadora.

YOLO realiza tareas de visión por computadora, como:

* Detección de objetos: Identifica y clasifica múltiples objetos en una imagen, marcándolos con cuadros delimitadores (bounding boxes).
* Segmentación de instancias (opcional): Identifica la forma exacta de los objetos.
* Estimación de pose (opcional): Determina las posiciones clave del cuerpo en imágenes o videos.

El modelo es capaz de reconocer objetos en tiempo real, lo que lo hace útil para aplicaciones como seguridad, conducción autónoma, monitoreo industrial y análisis deportivo.

**¿Cómo es el entrenamiento?**

Se entrena con conjuntos de datos etiquetados que incluyen imágenes y las correspondientes anotaciones de objetos (clases, ubicaciones, etc.).

* Utiliza técnicas de aprendizaje profundo con herramientas como PyTorch o TensorFlow.
* Soporta formatos de datos comunes, como COCO, YOLO, y VOC.

**Implementación**

Puede integrarse en aplicaciones mediante bibliotecas como Ultralytics (para PyTorch) o a través de frameworks de despliegue en tiempo real como ONNX Runtime o TensorRT.

Es compatible con dispositivos desde servidores de alta potencia hasta sistemas embebidos como NVIDIA Jetson.

**Uso práctico**

Se carga un modelo preentrenado.

Se procesa una imagen o un video, obteniendo las predicciones en tiempo real.

**Ventajas principales de la utilización de este modelo para el presente proyecto.**

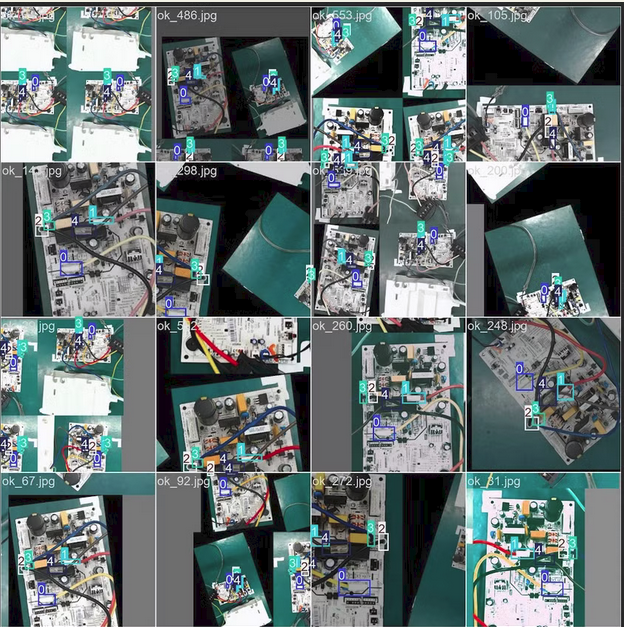
* Precisión: Detecta objetos pequeños y grandes con alta exactitud.
* Rapidez: Optimizado para tareas en tiempo real.
* Flexibilidad: Admite transfer learning para ajustar el modelo a tareas específicas. Para futuras líneas de investigación.

# Recopilación y Procesamiento de Datos

Se realizaron capturas de imágenes en formato jpg en diversas posiciones en la mesa de trabajo, con las posiciones correctas de los cables en sus respectivos lugares, y también en resultados no conformes.

Las imágenes fueron procesadas con la herramienta LabelIMG, que ahora es parte de la comunidad de Label Studio. Es una herramienta etiquetado de datos de código abierto para imágenes, texto, hipertexto, audio, vídeo y datos de series de tiempo.

Esta herramienta permite realizar un etiquetado que es fácil de delimitar e identificar para el entrenamiento del modelo. Las referencias son “0” - Amarillo, “1” Rojo, “2” - Negro, “3” - Azul, “4” - Marrón. El resultado de las etiquetas se guarda en un archivo de formato .txt, el cual se utilizó para el entrenamiento del modelo.

**Proceso de Etiquetado**

# Entrenamiento del modelo y el análisis de las métricas

Se dividieron los datos etiquetados en conjuntos de entrenamiento y de validación. Se entrena un modelo YOLOv8 utilizando un conjunto de datos definido en dataset.yaml de 501 Imágenes en total 400 en train y 101 en val. El entrenamiento se realiza durante 100 épocas con un tamaño de lote de 8 y un redimensionamiento de imágenes a 640x640 píxeles. El modelo utiliza el optimizador AdamW, conocido por su capacidad de regularización, y un programador de tasa de aprendizaje cosenoidal para mejorar la convergencia del entrenamiento.

Modelo de entrenamiento

|  | model.train(  data="/content/drive/MyDrive/YOLOv8/dataset.yaml",  epochs=100,  batch=8,  imgsz=640,  optimizer="AdamW",  cos\_lr=True  ) |
| --- | --- |

Desglosando este fragmento de código se puede decir que

Se entrenó en 100 epocas: Una época corresponde a un pase completo por todo el conjunto de datos de entrenamiento.Más épocas permiten un ajuste más fino del modelo, pero aumentan el tiempo de entrenamiento.

**batch=8:**

Tamaño del batch (lote): Número de imágenes procesadas juntas en cada paso del entrenamiento. Valores más altos pueden acelerar el entrenamiento, pero requieren más memoria de GPU.

**imgsz=640:**

Tamaño de las imágenes (en píxeles).

Todas las imágenes se redimensionan a 640x640 píxeles antes de ser procesadas por el modelo. Un tamaño mayor puede mejorar la precisión, pero incrementa el tiempo de entrenamiento.

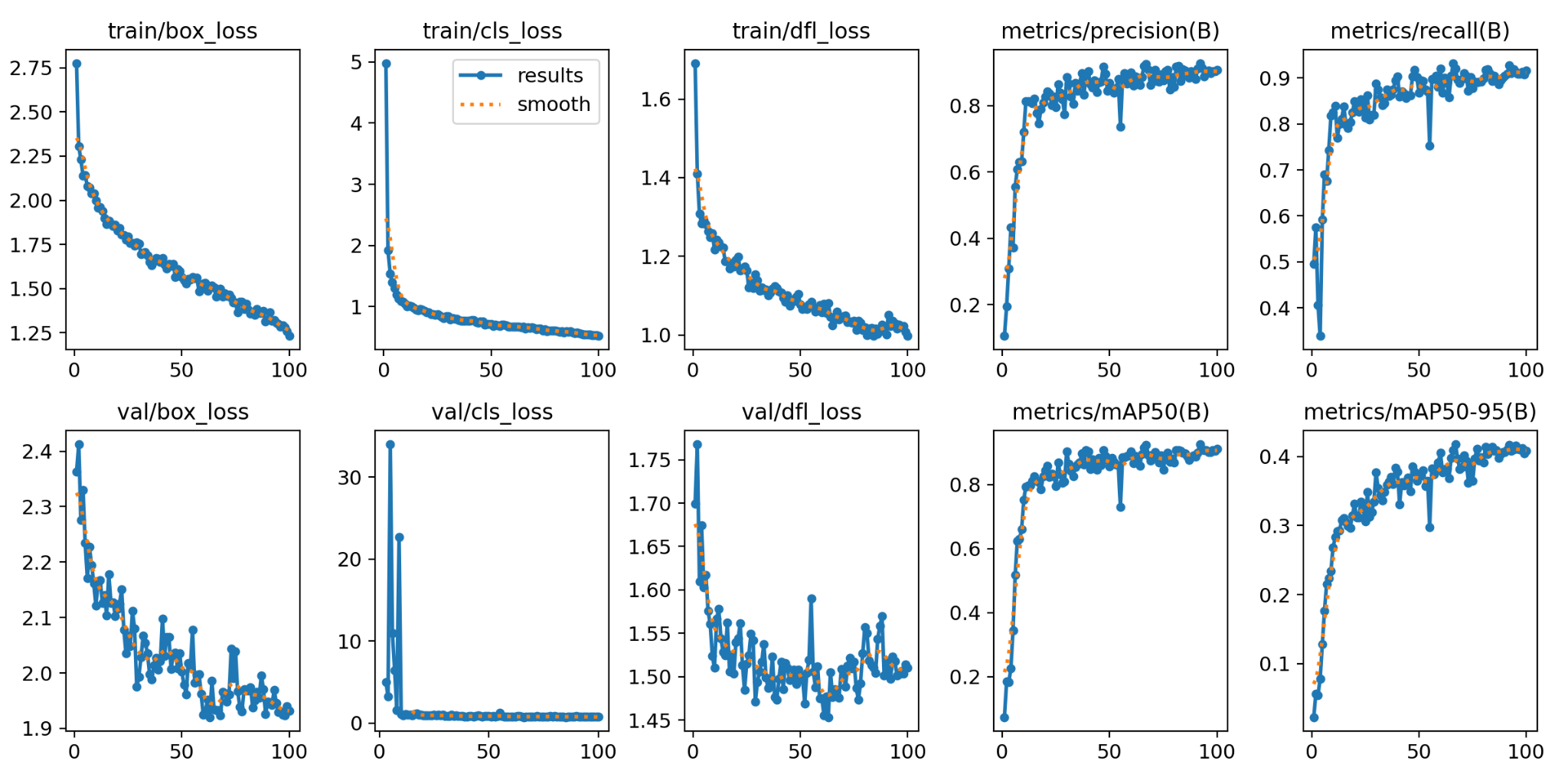
**optimizer="AdamW":**

Especifica el optimizador utilizado durante el entrenamiento. En este caso, se usa AdamW (Adam con decaimiento del peso), que mejora la regularización y evita el sobreajuste en redes profundas.

**cos\_lr=True:**

Activa el uso de un programador de tasa de aprendizaje cíclica (cosenoidal). Ajusta dinámicamente la tasa de aprendizaje durante el entrenamiento, comenzando alta, disminuyendo hacia el medio, y luego aumentando nuevamente hacia el final.

Arrojando como resultado los siguientes gráficos



Estas gráficas representan las métricas y pérdidas del entrenamiento y validación del modelo.

**Pérdidas de entrenamiento y validación (superior e inferior izquierda)**

Las pérdidas (train/box\_loss, train/cls\_loss, train/dfl\_loss y sus equivalentes de validación) muestran una disminución progresiva, lo que indica que el modelo está aprendiendo a realizar predicciones más precisas con el tiempo.

La convergencia estable de estas curvas sugiere que el modelo no está sobreentrenando.

**Precisión y exhaustividad (metrics/precision y metrics/recall):**

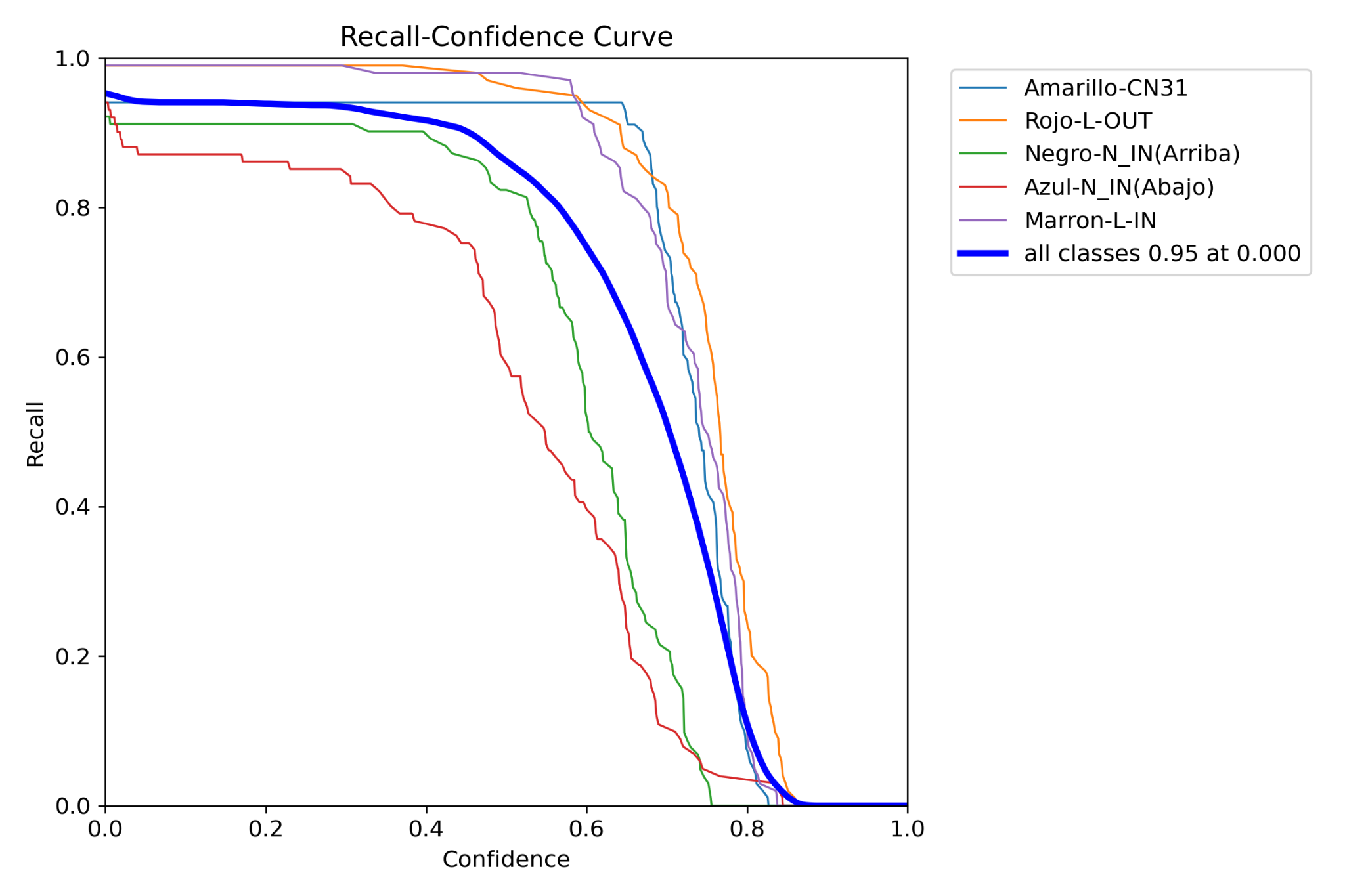
Ambas métricas aumentan consistentemente, indicando que el modelo mejora tanto en identificar correctamente los objetos como en no dejar fuera instancias relevantes.

**mAP (Mean Average Precision):**

El mAP a diferentes umbrales (mAP50 y mAP50-95) también mejora, mostrando que el modelo está obteniendo un mejor desempeño global en la tarea de detección.

Estos resultados sugieren que el modelo en general tiene un buen desempeño y es adecuado para la tarea que indicamos.

**Precision Recall - Curve**

****

Esta es una **curva Precision-Recall (PR)**, que evalúa el desempeño del modelo de clasificación considerando la relación entre **precisión** y **recobrado (recall)** a diferentes umbrales de decisión.

En los ejes del gráfico podemos observar:

Eje x (Recall): Mide cuántos casos positivos reales el modelo es capaz de identificar correctamente. Va de 0 a 1.

Eje y (Precision): Mide cuántas de las predicciones positivas del modelo son realmente correctas. Va de 0 a 1.

Curvas individuales por clase:

Cada línea de color representa el desempeño del modelo para una clase específica.

"Rojo-L-OUT" (naranja) tiene un área bajo la curva (AUC) de 0.971, lo que indica un desempeño excelente.

"Azul-N\_IN(Abajo)" (rojo) tiene un AUC más bajo de 0.826, lo que sugiere que el modelo tiene más dificultad para identificar correctamente esta clase.

Línea azul gruesa (all classes):

Representa el mAP@0.5 (Mean Average Precision con umbral de IoU = 0.5), que es un promedio del desempeño del modelo en todas las clases.

El valor de 0.923 indica que, en general, el modelo tiene un buen desempeño.

Comparación entre clases:

Las clases con curvas que se acercan más al vértice superior derecho (donde Recall = 1 y Precision = 1) son las mejores clasificadas.

Clases como "Marron-L-IN" tienen un desempeño sobresaliente con 0.991.

Por otro lado, "Azul-N\_IN (Abajo)" tiene un desempeño más bajo, posiblemente debido a confusiones con otras clases.

Los demás resultados junto con estas métricas analizadas se pueden observar dentro del proyecto en la carpeta nombrada como Imágenes-metricas-colab-BGH\_final, dentro de esta carpeta además hay un archivo results.csv con los valores de los entrenamientos

# Ejecución del modelo

Se hará una entrega del programa en un ejecutable .exe, deberán contar con una conexión a una cámara y en este caso el programa será capaz de reconocer las posiciones de los cables con sus respectivos colores siendo identificado con el nombre como nos fue dada en la documentación.

CN31 - AMARILLO

L-OUT - ROJO

N\_IN (arriba) - NEGRO

N\_IN(abajo) - AZUL

L-IN - MARRON