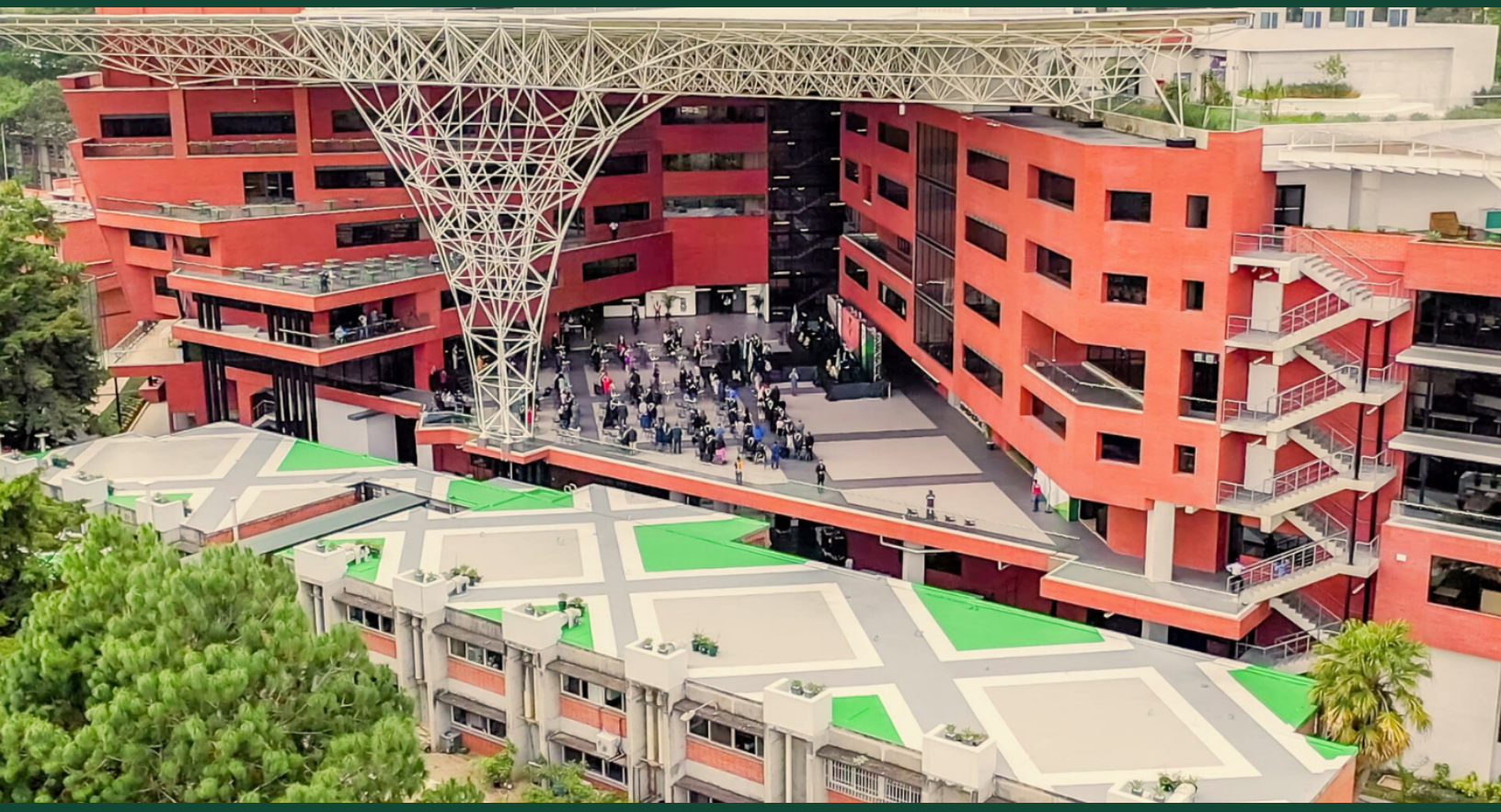
# Diseño e implementación de un sistema para la detección de equipo de protección personal basado en visión por computadora y aprendizaje automático

Edgar Omar Alejandro González Llamas



UVG | UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

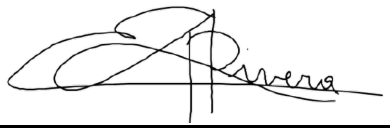

# Diseño e implementación de un sistema para la detección de equipo de protección personal basado en visión por computadora y aprendizaje automático

Trabajo de graduación presentado por Edgar Omar Alejandro González Llamas para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2025

Vo.Bo.:



(f) _____

Dr. Luis Alberto Rivera Estrada

(f) _____

M.Sc. Carlos Esquit Hernández

Fecha de aprobación: Guatemala, 20 de noviembre de 2025.

# Índice general

# Índice de figuras

# Índice de cuadros

# Resumen

La verificación constante del uso de equipo de protección personal (EPP) es esencial en entornos industriales, académicos y en laboratorios. Con el objetivo de automatizar esta tarea y reducir el riesgo de accidentes, se desarrolló un sistema de detección basado en visión por computadora y aprendizaje automático. El sistema utiliza redes neuronales convolucionales, específicamente la arquitectura YOLO, para identificar elementos como casco, chaleco y mascarilla en imágenes y video en tiempo real. Para su implementación fue necesario recopilar y etiquetar una colección de datos, realizar varios entrenamientos del modelo y diseñar una interfaz gráfica interactiva que permite visualizar detecciones, ajustar parámetros, así como generar alertas cuando no se cumpla con el uso adecuado del equipo. Los resultados obtenidos demuestran que el uso de modelos avanzados de YOLO, junto con una adecuada calidad de la colección de datos, permite alcanzar niveles altos de precisión. El sistema automatiza la supervisión del EPP, mejora la eficiencia operativa y contribuye a fortalecer la seguridad dentro de los espacios de trabajo.

# Abstract

Ensuring the proper use of personal protective equipment (PPE) is critical in industrial, academic, and laboratory environments. To automate this verification process and reduce the risk of accidents, a detection system based on computer vision and machine learning was developed. The system employs convolutional neural networks, specifically the YOLO architecture, to identify safety elements such as helmets, vests, and face masks in images and real-time video. The implementation required data collection and annotation, model training, and the development of an interactive graphical interface that displays detections, allows parameter adjustments, and generates automatic alerts. The results show that advanced YOLO models, combined with high-quality datasets, achieve strong detection performance. The proposed system automates PPE monitoring, improves operational efficiency, and contributes to enhancing safety in work environments.

# CAPÍTULO 1

## Introducción

La visión por computadora y el aprendizaje automático se han consolidado como herramientas fundamentales en múltiples áreas de la ingeniería, la industria y la seguridad. Entre sus aplicaciones más relevantes se encuentra la detección de objetos, una técnica que permite a los sistemas identificar, localizar y delimitar elementos de interés dentro de imágenes o video. Esta capacidad ha impulsado el desarrollo de sistemas capaces de automatizar procesos y apoyar la toma de decisiones en tiempo real.

En este proyecto se aborda la detección de forma visual de equipo de protección personal (EPP) aplicado a imágenes y videos. Es importante destacar que el alcance del trabajo se enfoca únicamente en la detección de elementos de interés, dejando de lado de integración de sistemas de control utilices para la implementación de alarmas.

Para cumplir con esta tarea se implementan redes neuronales convolucionales (CNN), las cuales cuentan con la capacidad de identificar patrones dentro de imágenes. Estas redes aprenden características relevantes, tales como bordes, formas y textura, a través de capas de neuronas artificiales interconectadas, lo que les permite extraer progresivamente la información más significativa de cada imagen y con ello comparar dicha información con bases de datos ya establecidas.

En el trabajo se emplea *Python* como lenguaje de programación principal debido a su versatilidad en el ámbito del aprendizaje automático y visión por computadora. Ofrece una gran cantidad de bibliotecas especializadas en el desarrollo de redes neuronales, que facilitan la implementación, entrenamiento y validación de modelos de detección de objetos. Gracias a estas herramientas, Python se ha consolidado como una de las plataformas más relevantes y eficientes para el desarrollo de proyectos basados en inteligencia artificial.

El propósito principal del trabajo es desarrollar un modelo adaptable, fácil de manipular y con capacidad de mejora continua, de manera que pueda emplearse como una herramienta capaz de verificar el uso adecuado de EPP. Con ello se busca apoyar los procesos de supervisión, facilitar el cumplimiento de normas de seguridad en áreas de riesgo y contribuir la mejora de una cultura laboral responsable y segura.

# CAPÍTULO 2

# Antecedentes

La seguridad industrial actualmente es base para la gran mayoría de trabajos en talleres, industrias o laboratorios, ya que las maquinarias y herramientas disponibles presentan dificultades para su manipulación. Para todo propietario de taller o industria es indispensable controlar que el protocolo establecido sea cumplido por el personal, para reducir al mínimo posible los accidentes. La solución más rápida es instalar un sistema de monitoreo con el fin de verificar el cumplimiento de las normas establecidas.

El verificar el uso del equipo de protección personal (EPP) debe ejecutarse de manera constante; en este caso, la visión por computadora presenta herramientas para esta tarea. Es posible de detectar objetos en imágenes o videos y clarificarlos según sea necesario y con dicha información ser capaces de notificar cuando un elemento del EPP no está presente en la área.

## 2.1. Investigaciones previas en Universidad del Valle Guatemala

En la UVG, se han presentados trabajos con modelos similares, por ejemplo el trabajo titulado *Desarrollo de un traductor de Lengua de Señas de Guatemala (LENSEGUA) mediante visión por computadora y aprendizaje automático* realizado por Ashley Stephanie Morales Aldana, en el cual se desarrolló e implemento un sistema por visión por computadora para el reconocimiento y segmentación del lenguaje de señas en tiempo real. Para el proyecto se utilizó *python* y sus librerías especializadas, como *opencv*, para el procesamiento de imágenes y clasificación de objetos [1].

En el trabajo titulado *Implementación de infraestructura a escala para la evaluación de algoritmos por visión de computador para vehículos autónomos* en el cual se utilizan herramientas de *tensorflow lite* y *openmv* para la detección de señales de tránsito y elementos de la carretera, con la diferencia de que en este proyecto se utilizó la plataforma web *edge*

*impulse*, capaz de proporcionar y entrenar una red convolucional específicamente para la detección de objetos establecidos por el usuario, en donde se logró un acierto promedio del 90 % en cada una de las validaciones realizadas [2].

## 2.2. Detección de objetos mediante red convolucional

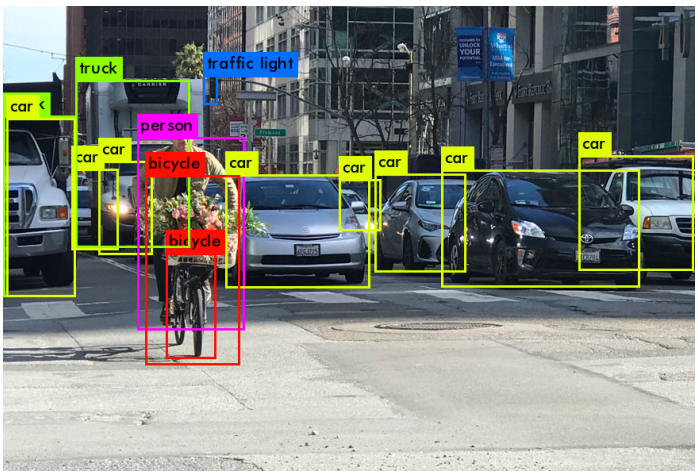
Actualmente, ya se han establecido métodos de detección y segmentación de imágenes para su respectivo análisis. Los autores de *EMOD: Efficient Moving Object Detection via Image Eccentricity Analysis and Sparse Neural Networks* nos presentan el desarrollo de estrategias enfocadas en el desarrollo de una red neuronal. En el trabajo se plantearon las estrategias para el procesamiento de datos, mediante imágenes y videos con el fin de segmentar y detectar los objetos [3].

El procesamiento de datos definido anteriormente, es indispensable para la detección de objetos, con el fin de definir aquellos elementos en movimiento para luego realizar una segmentación y clasificación de ellos. Mediante la segmentación y clasificación la red convolucional logra establecer qué objetos están presentes en la imagen.

Dentro de los algoritmos de redes neuronales convolucionales, la familia *you only look once* (YOLO) ha demostrado ser eficaz en tareas de detección de objetos en imágenes y video en tiempo real. Actualmente, se encuentran su versión 12, y gracias a las constantes actualizaciones ha sido adaptada a diversos contextos y aplicaciones específicas, como se puede observar en la figura 1.

Un ejemplo es el trabajo titulado *YOLO-Pose: Enhancing YOLO for Multi Person Pose Estimation Using Object Keypoint Similarity Loss*, en el cual se emplea la versión 5 de la arquitectura para estimar poses humanas. En dicho estudio, se utiliza el modelo denominado *YOLO-Pose*, enfocado en la segmentación de extremidades con el objetivo de determinar la postura de una o varias personas. Esta información permite determinar las interacciones que dichas personas cuentan con objetos a su alrededor [4].

**Figura 1.** Detección de objetos mediante YOLO versión 8




Nota. La imagen muestra la aplicación de visión por computadora y detección de objetos usando el modelo de YOLO versión 8. Imagen obtenida de [5].

Estas arquitecturas están desarrolladas mediante Python y disponen de modelos preentrenados y optimizados para la detección de objetos cotidianos. Además, permiten el entrenamiento desde cero a partir de bases de datos propias, lo que facilita su implementación en diversos entornos.

Python es un lenguaje de programación altamente usado para la implementación de visión por computadora, debido a las investigaciones previas realizadas y principalmente por las librerías ya existentes desarrolladas específicamente en el desarrollo de inteligencia aritifical (IA) y aprendizaje automático. Como presentan los autores de *object detection in rural roads using tensorflow API* tensorflow es un *framework* que esta optimizado para la clasificación de objetos y principalmente en la posibilidad de entrenar una red neuronal de manera supervisada con los objetos específicos que estamos interesados en evaluar. [6].

Empresas como *tesla* han desarrollado e implementado algoritmos de visión por computadora con la herramienta YOLO, demostrando su impacto en el ámbito tecnológico y principalmente afirma que no es necesario desarrollar una red neuronal convolucional desde cero, es posible utilizar herramientas establecidas para realizar aplicaciones en el campo de interés.

# CAPÍTULO 3

## Justificación

Desde la revolución industrial fue necesario definir normas que aseguraban al personal involucrado en trabajos con materiales pesados y trabajos en caliente. En primer lugar se estableció el uso de equipo de protección personal (EPP). En su presentación más básica, establecida por casco, chaleco reflectivo, botas y lentes, este conjunto se puede definir como equipó de sobreponer. Para el conjunto presentado, el personal no necesita un entrenamiento previo y es de fácil manipulación.

La facilidad de remover el EPP presenta un conflicto, ya que puede causar que el personal no lo utilice en momentos de peligro. Cabe resaltar que dicho equipo de protección varía según las necesidades de la empresa o a las adversidades que el personal puede enfrentar durante la jornada de trabajo. La revisión automática del EPP facilita los procesos industriales o en laboratorios, permitiendo que el personal trabaje en óptimas condiciones. La aplicación del sistema planteado en el trabajo en entornos educativos, tales como en los laboratorios de la UVG, fomenta la responsabilidad en los estudiantes y previene accidentes dentro de las instalaciones.

Con herramientas de detección de objetos, los colaboradores de la UVG pueden enfocar el tiempo disponible para abarcar temas de interés según la clase, evitando perder tiempo en la revisión individual de cada estudiante. Herramientas de visión por computadora presentan la ventaja de una revisión constante, lo que implica el notificar anomalías en el uso de EPP durante todo el tiempo que dure la sección.

# CAPÍTULO 4

# Objetivos

## 4.1. Objetivo general

Diseñar e implementar un sistema de detección de equipo de protección personal mediante visión por computadora y aprendizaje automático.

## 4.2. Objetivos específicos

- Investigar y definir algoritmos de visión por computadora, procesamiento de imágenes y aprendizaje automático adecuados para la segmentación de imágenes y detección de objetos.
- Obtener una colección de datos relevantes para el entrenamiento y validación de los algoritmos.
- Entrenar y validar los algoritmos con los datos obtenidos.
- Desarrollar una interfaz gráfica para el uso del sistema.

# CAPÍTULO 5

## Definición del problema

La verificación del uso adecuado de equipo de protección personal (EPP) se ha vuelto un desafío constante en entornos laborales donde los trabajadores están expuestos a riesgos físicos, químicos o mecánicos. Tradicionalmente, esta supervisión dependía de la observación manual realizada por personal encargado de seguridad, un proceso que resultaba susceptible a errores humanos, limitaciones de atención y falta de cobertura continua, especialmente en áreas con alta movilidad de personal.

El proyecto se enfocó en la necesidad de mejorar y automatizar el proceso de supervisión. A partir de esto, se desarrolló una solución basada en técnicas de visión por computadora y aprendizaje automático, la cual nos permite automatizar la detección del uso de EPP en tiempo real mediante cámaras de video. Al implementar el sistema, fue posible identificar los escenarios en los que las personas usan de forma adecuada su EPP y generar alertas cuando se detecte la ausencia de este, todo con el fin de fortalecer el impacto de los incidentes laborales.

Se centró en el diseño, implementación y validación de un modelo funcional capaz de reconocer los elementos de EPP en imágenes y video. El sistema alcanzó un nivel adecuado de desempeño para demostrar la viabilidad de la automatización de este proceso. El proyecto abordó de manera efectiva el problema de la supervisión manual del EPP al desarrollar una herramienta automatizada con el fin de mejorar la tarea.

# CAPÍTULO 6

# Marco teórico

## 6.1. Seguridad industrial

La intensificación de los procesos de manufactura y producción, impulsada por el consumo masivo, ha obligado a las empresas a expandir su infraestructura y fuerza laboral de manera constante. Sin embargo, este crecimiento conlleva un incremento en los riesgos laborales. La manipulación de maquinaria pesada, sustancias tóxicas y materiales voluminosos eleva la probabilidad de accidentes y enfermedades ocupacionales [7].

La nueva revolución industrial no solo implica mayor automatización, sino también la incorporación de nuevas tecnologías en las líneas de trabajo. En este contexto, resulta fundamental capacitar adecuadamente al personal y proporcionar las herramientas necesarias, especialmente aquellas relacionadas con las medidas de seguridad, con el fin de salvaguardar la integridad física durante la jornada laboral [8].

### 6.1.1. Marco legal

En Guatemala, la regulación de la seguridad industrial se sustenta en diversos instrumentos legales que buscan proteger la salud y el bienestar de los trabajadores. Entre ellos, el *Código de Trabajo de Guatemala* establece la obligación del empleador de garantizar condiciones seguras e higiénicas dentro del lugar de trabajo [9]. Asimismo, el *Instituto Guatemalteco de Seguridad Social (IGSS)* [10] supervisa aspectos relacionados con los riesgos profesionales, la atención médica y la prevención de enfermedades.

Adicionalmente, muchas empresas adoptan estándares internacionales como la norma ISO 45001 [11], la cual establece los requisitos mínimos para un sistema de gestión de seguridad y salud en el trabajo. El cumplimiento de estándares internacionales facilita el comercio de bienes y servicios y contribuye a mejorar la competitividad empresarial.

### 6.1.2. Equipo de protección personal (EPP)

En ocasiones, la seguridad industrial no depende de sistemas complejos, sino del uso adecuado de herramientas básicas diseñadas para proteger al trabajador. El equipo de protección personal (EPP) está conformado por elementos destinados a resguardar la piel, la vista, la respiración y otras áreas vulnerables del cuerpo. Según la *occupational safety and health administration* (OSHA), el EPP incluye dispositivos como cascos de seguridad, protección ocular y facial, guantes, protección respiratoria, calzado de seguridad y ropa especializada, los cuales deben seleccionarse según los riesgos presentes en el entorno de trabajo.

Algunos de estos elementos se ilustran en la Figura 2.

**Figura 2.** Elementos de un equipo de protección personal

Nota. La imagen muestra algunos de los elementos presentes en un equipo de protección personal de un trabajador de la industria. Imagen obtenida de [12].

### 6.1.3. Indicadores

Los indicadores de seguridad industrial permiten medir el comportamiento del personal dentro de un taller o planta, así como evaluar la efectividad de las medidas de prevención implementadas. Mantener un registro adecuado de incidentes y condiciones de trabajo facilita la toma de decisiones orientadas a reducir riesgos y mejorar el ambiente laboral.

Entre los principales indicadores utilizados en la industria se encuentran:

- Número de accidentes laborales por año.
- Tasa de frecuencia (accidentes por cada millón de horas trabajadas).
- Tasa de severidad (días perdidos debido a accidentes).
- Número de enfermedades ocupacionales detectadas.
- Número de accidentes por área de trabajo.

Estos indicadores permiten evaluar el impacto de las medidas de seguridad implementadas y ajustar las estrategias preventivas según los resultados obtenidos.

## 6.2. Aprendizaje automático

El aprendizaje automático, *machine learning*, es una rama de la inteligencia artificial que permite a los sistemas aprender a realizar tareas específicas sin ser programados explícitamente para cada una. Para ello, se emplean modelos matemáticos capaces de identificar patrones y realizar predicciones, clasificaciones o decisiones basadas en datos.

Los modelos de aprendizaje automático requieren un conjunto de datos representativo para comprender la tarea. Este puede estar compuesto por datos etiquetados (aprendizaje supervisado) o no etiquetados (aprendizaje no supervisado).

### 6.2.1. Aprendizaje supervisado

En esta modalidad, el modelo se entrena utilizando datos acompañados de etiquetas que representan la salida correcta. El objetivo es que el sistema reconozca los elementos presentes en la tarea y aprenda los rasgos más relevantes asociados a cada categoría.

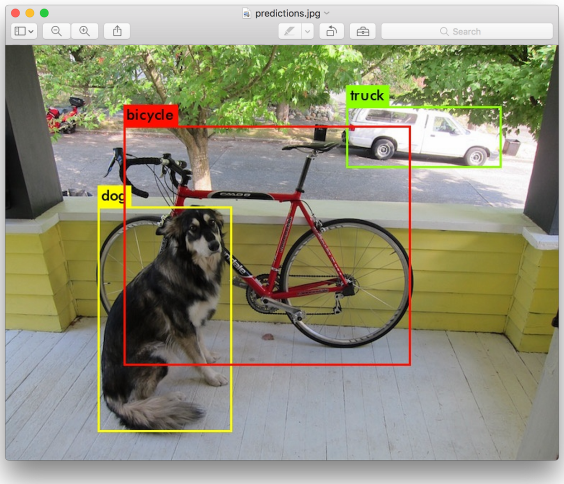
### 6.2.2. Aprendizaje no supervisado

En este caso, el modelo recibe datos sin etiquetas y debe identificar patrones, agrupamientos o estructuras internas por sí mismo. Esta metodología requiere poca intervención humana, dado que solo se establecen los parámetros generales del sistema.

### 6.2.3. Aprendizaje por refuerzo

En el aprendizaje por refuerzo, el modelo interactúa con un entorno y aprende mediante un sistema de recompensas. Cada acción tomada por el agente genera una recompensa positiva o negativa, lo que permite ajustar su comportamiento con el fin de maximizar la ganancia acumulada.

**Figura 3.** Ejemplo de etiquetado de imágenes




Nota. Ejemplo del proceso de etiquetado empleado en el aprendizaje supervisado, donde cada elemento de interés se delimita mediante un *bounding box*. Imagen obtenida de [13].

## 6.3. Visión artificial

La visión artificial permite que una máquina procese e interprete información mediante imágenes o video. Para ello, se emplean algoritmos capaces de extraer las características más relevantes como bordes, texturas, colores, formas y patrones espaciales, convirtiendo datos visuales en información útil para la toma de decisiones [14].

El procesamiento de imágenes constituye la base para esta tarea. Incluye operaciones como filtrado, corrección de iluminació o segmentación, las cuales permiten mejorar la calidad de la imagen o aislar regiones de interés. Una vez obtenidas estas representaciones, se aplican técnicas de mayor nivel, tales como clasificación, reconocimiento o seguimiento de objetos, utilizadas frecuentemente en modelos de aprendizaje automático y redes neuronales [15].

La visión artificial se ha vuelto fundamental en diversas aplicaciones modernas, principalmente en el área de inspección y control de calidad en manufactura, monitoreo industrial, conducción autónoma, seguridad y vigilancia. En la mayoría de estos entornos, los sistemas deben operar en tiempo real y bajo condiciones variables, lo que ha impulsado el desarrollo de arquitecturas de redes neuronales convolucionales capaces de aprender directamente a partir de los datos proporcionados con anterioridad. [16].

Para el trabajo de enfoca principalmente en detección de objetos, en donde la visión artificial permite localizar y clasificar múltiples elementos dentro de una escena. Estas tareas requieren el análisis de características visuales con algoritmos especializados, por ejemplo las arquitecturas *yolo* [17] o *mask r-cnn* [18], que actualmente constituyen estándares en el campo por su eficiencia y precisión.

## 6.4. Redes neuronales

Las redes neuronales artificiales son modelos de aprendizaje automático diseñados para aproximar relaciones complejas entre datos de entrada y salida [15], [19]. Estas redes se inspiran en el funcionamiento del cerebro humano, particularmente en la manera en que las neuronas procesan y transmiten información. Cada nodo o "neurona" artificial realiza operaciones matemáticas que permiten transformar, combinar y propagar la información a lo largo de varias capas de la red, con el objetivo de identificar patrones subyacentes en los datos.

El aprendizaje en redes neuronales se realiza generalmente mediante algoritmos de optimización y retropropagación, que ajustan los pesos de las conexiones para minimizar la diferencia entre la salida predicha y la real [20]. Gracias a esta capacidad de aprendizaje, las redes neuronales pueden abordar tareas complejas que serían difíciles de resolver mediante métodos tradicionales, como reconocimiento de imágenes, procesamiento del lenguaje natural, predicción de series temporales y sistemas de recomendación [21].

Una vez entrenadas correctamente, estas redes exhiben una notable capacidad de generalización, lo que les permite realizar predicciones precisas sobre datos que no han visto previamente. Esto las convierte en herramientas fundamentales en aplicaciones de inteligencia artificial en distintos ámbitos, desde la medicina hasta la robótica y la visión por computadora [19].

Además, las redes neuronales profundas o modelos de *deep learning* permiten construir arquitecturas con múltiples capas ocultas que capturan características jerárquicas de los datos, lo que ha impulsado avances significativos en reconocimiento de voz, análisis de imágenes médicas y vehículos autónomos [15], [22].

## 6.5. Tecnologías y herramientas

Para implementar redes neuronales en tareas de detección de objetos es necesario contar con un conjunto de herramientas tecnológicas que faciliten el desarrollo y la experimentación.

### 6.5.1. Lenguajes de programación y entornos

Python es uno de los lenguajes más utilizados en aprendizaje profundo debido a su sintaxis sencilla y a la amplia disponibilidad de librerías científicas como *numpy*, *pandas* y *matplotlib*. Además, entornos como jupyter notebook, google colab y kaggle notebooks permiten experimentar de manera interactiva y prototipar rápidamente.

### 6.5.2. frameworks de aprendizaje profundo

El uso de *framework*s facilitan la construcción y entrenamiento de redes neuronales. Entre los más utilizados se encuentran:

- **tensorFlow y keras**: ofrecen una interfaz flexible para la creación de modelos y una amplia colección de herramientas para redes convolucionales y arquitecturas avanzadas.
- ***Pytorch***: ampliamente utilizado en investigación por su enfoque dinámico y facilidad para depurar modelos complejos.

### 6.5.3. Herramientas de anotación y procesamiento de datos

El entrenamiento de modelos de detección de objetos requiere bases de datos correctamente etiquetadas. Herramientas como *labelme* presente en la figura 5, *cvat* y *roboflow* permiten delimitar regiones de interés mediante *bounding box*s y generar archivos de anotación en formatos estándar, como json. Asimismo, bibliotecas como *opencv* facilitan el preprocesamiento y la manipulación de imágenes.

**Figura 4.** Entorno de aplicación LabelMe


Nota. Ejemplo de la interfaz presente a la hora de realizar el etiquetado de las imagenes mediante la aplicación *labelme*. Imagen obtenida de [23].

### 6.5.4. Aplicaciones de las redes neuronales

Las redes neuronales han impactado múltiples áreas, entre ellas:

- **Seguridad industrial**: detección automática de equipo de protección personal.
- **Vehículos autónomos**: identificación de señales y otros vehículos.
- **Medicina**: análisis de imágenes radiológicas.
- **Robótica**: percepción del entorno para manipulación de objetos.
- **Seguridad y vigilancia**: reconocimiento facial y detección de comportamientos anómalos.

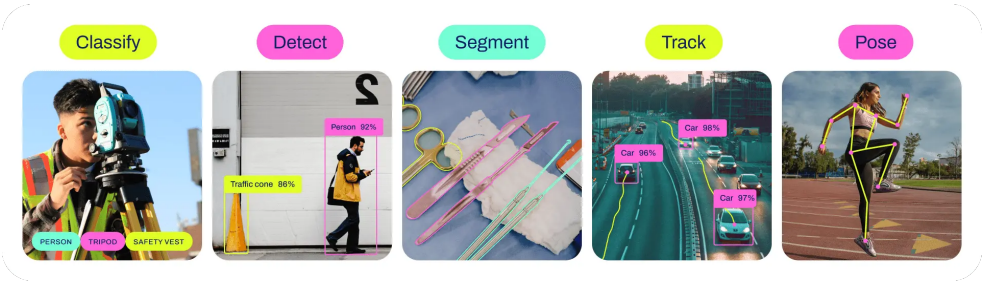
## 6.6. Fundamentos de la detección de objetos

La detección de objetos es un área central dentro de la visión por computadora y el aprendizaje automático, cuyo objetivo es identificar instancias de objetos dentro de imágenes o secuencias de video [24], [25]. Este proceso combina tareas de localización, clasificación y, en muchos casos, seguimiento, permitiendo aplicaciones que van desde la seguridad industrial hasta los sistemas autónomos [26].

- **Localización de objetos**: consiste en identificar la posición aproximada de un objeto en la imagen mediante un *bounding box*. Esto permite conocer no solo la presencia de un objeto, sino también su ubicación precisa, información esencial para tareas de seguimiento o análisis de interacción [27].
- **Detección de múltiples objetos**: permite localizar y reconocer simultáneamente varios objetos en una misma escena, asignándoles una categoría y una puntuación de confianza que indica la certeza del modelo. Este enfoque se utiliza ampliamente en monitoreo de tráfico, seguridad y líneas de producción [25], [26].
- **Clasificación de imágenes**: asigna una etiqueta global a la imagen completa, sin especificar la región donde se encuentra el objeto. Aunque más simple que la detección, constituye la base para arquitecturas más complejas y suele emplearse en el preentrenamiento de redes profundas [28].

La diferencia principal entre estos enfoques radica en la salida del sistema y en la estructura de los datos de entrada, las aplicaciones de cada modo de uso se representan de mejor manera en Mientras que en la clasificación basta con una etiqueta por imagen, en la localización y la detección de múltiples objetos es indispensable proporcionar coordenadas y categorías para cada región de interés. Además, la detección de objetos requiere redes convolucionales avanzadas y técnicas especializadas de entrenamiento.[29].

**Figura 5.** Posibles aplicaciones de redes neuronales y visión por computadora




Nota. Imagen obtenida de [30].

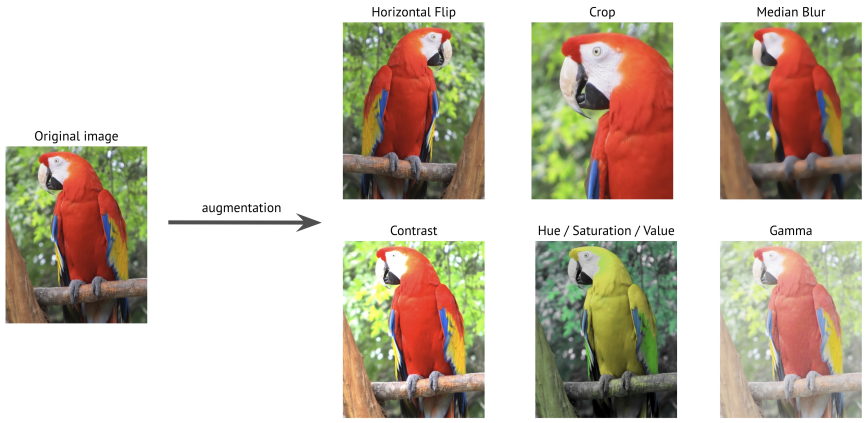
## 6.7. Aumento de datos

El aumento de datos consiste en generar nuevas imágenes a partir de las originales mediante transformaciones que simulan variaciones reales del entorno que podrían ocurrir

durante la implementación del modelo [31]. La Figura 6 muestra algunos ejemplos. Las modificaciones pueden consistir en:

- **Rotaciones**: giro de la imagen en diferentes ángulos.
- **Traslaciones**: desplazamiento horizontal o vertical.
- **Reflejos y volteos**: espejado horizontal o vertical.
- **Ruido**: inclusión de ruido para mejorar la robustez.
- **Escalado**: cambio de tamaño del objeto o la imagen.
- **Ajustes de brillo y contraste**: variaciones de iluminación para simular distintos escenarios.

**Figura 6.** Ejemplo de aumento de datos en imágenes




Nota. Ejemplos de transformaciones aplicadas durante el aumento de datos. Imagen obtenida de [32].

Este proceso permite incrementar de forma significativa el tamaño del conjunto de datos, mejorar la capacidad de generalización del modelo y reducir el riesgo de *overfitting* [31]. Mediante estas técnicas, el modelo será más robusto frente a cambios ambientales y variaciones visuales.

# CAPÍTULO 7

# Herramientas necesarias

Para desarrollar un sistema de aprendizaje automático existen diversos lenguajes especializados. En este proyecto se seleccionó *python* debido a su amplia disponibilidad de recursos, compatibilidad con librerías científicas y capacidad para integrar interfaces gráficas multiplataforma.

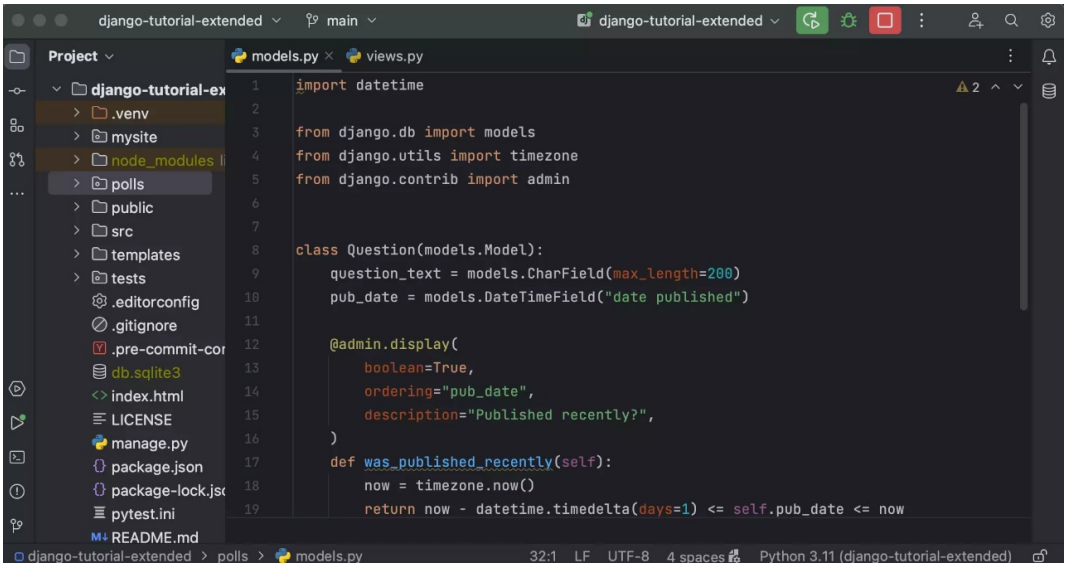
## 7.1. Lenguaje de programación

El primer paso consiste en instalar el entorno de programación. Para este trabajo se utilizó la versión *python 3.12.6*, descargada desde la página oficial [33]. Es indispensable contar con una versión actualizada debido a la compatibilidad requerida con librerías de aprendizaje automático. Para verificar la versión instalada puede emplearse el Código 15.1.

## 7.2. Entorno de trabajo

Dado que el procesamiento de imágenes y el entrenamiento de modelos requieren una cantidad considerable de recursos, se empleó un entorno de desarrollo capaz de gestionar proyectos complejos y entornos virtuales de manera eficiente. En este trabajo se utilizó *pycharm*, mostrado en la Figura 7. Este entorno facilita la depuración, administración de dependencias e integración de librerías de inteligencia artificial. Puede descargarse desde el sitio oficial de JetBrains [34].

**Figura 7.** Entorno de trabajo de *pycharm*




Nota. Captura representativa del entorno de desarrollo pycharm. Imagen obtenida de [35].

## 7.3. Recursos y bibliotecas de software

El uso de modelos de detección de objetos requiere herramientas que permitan construir y entrenar redes neuronales de forma flexible. Para este propósito, el *Framework Pytorch* ofrece una plataforma robusta y ampliamente utilizada.

Las bibliotecas necesarias para el procesamiento de imágenes, visualización y manipulación de datos se instalan mediante el Código 15.2. Estas herramientas permiten gestionar operaciones matriciales, lectura de imágenes y renderizado de visualizaciones.

Adicionalmente, es necesaria la instalación del entorno *Pytorch*, el cual puede aprovechar la capacidad de procesamiento de la Gpu mediante *Cuda* y *Cudnn*. Un ejemplo de instalación compatible con cuda se presenta en el Código 15.3. El comando específico puede variar según la configuración de hardware, por ello se recomienda consultar la guía oficial [36].

El equipo utilizado para este proyecto cuenta con las siguientes unidades de procesamiento:

- Intel UHD Graphics
- NVIDIA GeForce rtx 4060

Debido al uso intensivo de operaciones matriciales y procesamiento de imágenes, la Gpu *rtx* resulta la opción más adecuada, pues permite agilizar el proceso de entrenamiento y evitar conflictos de compatibilidad.

# CAPÍTULO 8

# Recopilación y creación de conjuntos de datos

En este proyecto se empleó la metodología de aprendizaje supervisado, ya que los elementos que se desean detectar se encuentran claramente definidos. Esta elección facilita el proceso de entrenamiento, pues cada objeto de interés se asocia a una etiqueta específica, lo que permite que el modelo aprenda de manera más eficiente y precisa las características relevantes.

Antes de iniciar el entrenamiento de los modelos de detección, fue indispensable conformar un conjunto de datos que representara los objetos de interés. Con el objetivo de emplear los modelos en actividades propias de entornos de taller y laboratorio, se estableció la detección de los siguientes elementos comunes:

- Mascarilla
- Casco
- Bata

## 8.1. Consideraciones necesarias para el conjunto de datos

El objetivo del proyecto es identificar escenarios donde el equipo de protección personal (EPP) se utiliza correctamente y donde está ausente. Por lo tanto, no basta con identificar los objetos individualmente; es indispensable incluir ejemplos donde el equipo esté presente y donde no lo esté. Un ejemplo de estas situaciones puede observarse en la Figura 8.

**Figura 8.** Imagen de ejemplo del conjunto de datos 1


Nota. La imagen presentada es un ejemplo de lo que se busca dentro del conjunto de datos. En este caso se observa claramente a una persona utilizando equipo de protección personal y a otra que no cuenta con ningún elemento. Imagen obtenida de [37].

Es esencial contar con una representación equilibrada de los casos con y sin EPP, lo cual permite mejorar la capacidad del modelo para generalizar en situaciones reales, donde existen variaciones de iluminación, postura y distancia.

## 8.2. Recopilación de bases de datos externas

Para llevar a cabo el entrenamiento de la red fue necesario reunir un conjunto de datos que representara adecuadamente el entorno de estudio. El tipo de imágenes ilustrado en la Figura 8 proviene de la base publicada en *mendeley data* [37], la cual contiene 2,320 imágenes clasificadas según el equipo de protección personal utilizado.

Con el fin de incorporar la detección de mascarillas, se añadió un segundo conjunto de datos disponible en la plataforma *kaggle* [38], el cual incluye 853 imágenes donde se muestra el uso y la ausencia de mascarilla.

## 8.3. Aumento de datos

En total se recopilaron más de 3,000 imágenes. Antes del aumento de datos, las imágenes originales tenían condiciones similares a las mostradas en la Figura 9.

**Figura 9.** Imagen original del conjunto de datos antes del aumento de datos

Nota. Imagen base del conjunto de datos recopilado, donde se observan escenarios con uso y ausencia de equipo de protección personal. Imagen obtenida de [37].

Para incrementar la variabilidad del conjunto de datos, se realizaron diferentes transformaciones sobre cada imagen. Ejemplos de estas modificaciones pueden observarse en las Figuras 10 y 11. Las transformaciones aplicadas fueron:

- Rotación de la imagen.
- Reflexión horizontal.
- Ajuste de brillo y contraste.

Estas variaciones permitieron aumentar significativamente el tamaño del conjunto de datos y mejorar la robustez del modelo.

**Figura 10.** Primer aumento de datos presente para cada imagen

Nota. Ejemplo de transformación mediante reflexión horizontal (rotación en el eje $y$) para que el modelo perciba las imágenes desde distintas perspectivas. Imagen propia.

**Figura 11.** Segundo aumento de datos presente para cada imagen

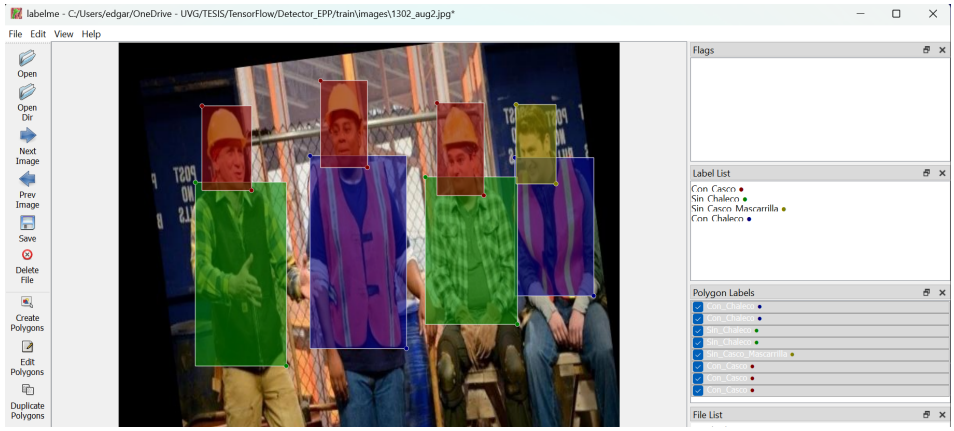
Nota. Ejemplo de ajuste de brillo y contraste aplicado a una imagen, con el fin de simular variaciones de iluminación presentes en entornos reales. Imagen propia.

Luego de aplicar el proceso de aumento de datos, el conjunto final alcanzó un total de 9,519 imágenes.

## 8.4. Etiquetado de imágenes

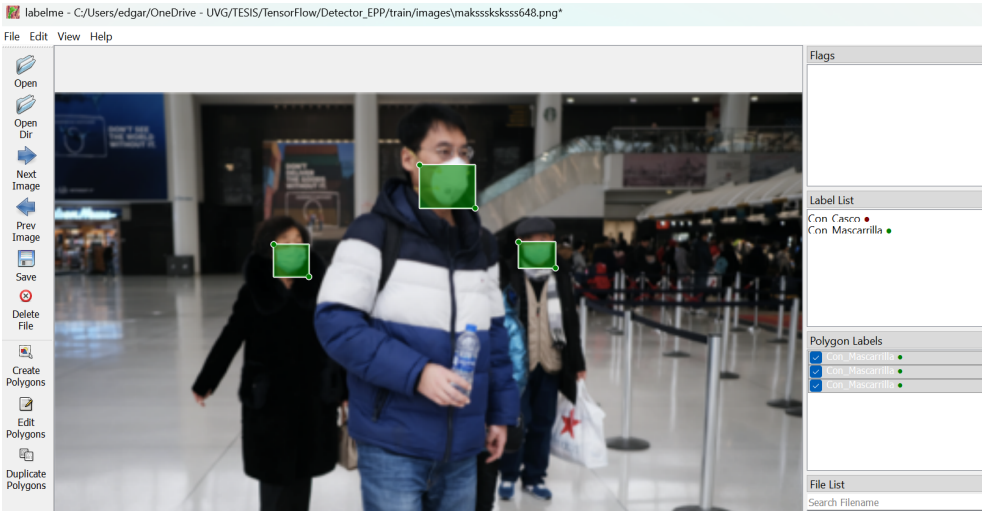
Posteriormente, se procedió al etiquetado de las imágenes mediante *bounding box*s. Este proceso puede observarse en las Figuras 12 y 13, donde se ilustran los elementos de interés.

**Figura 12.** Etiquetado de imágenes según los elementos presentes




Nota. Se observa el uso de *bounding box*s destinados a delimitar y clasificar los diferentes escenarios presentes en una imagen. Imagen propia.

**Figura 13.** Etiquetado de imágenes sobre el uso de mascarilla




Nota. En este caso se delimitan las zonas donde se espera el uso de mascarilla y se identifica si las personas presentes cumplen con dicha medida. Imagen propia.

La herramienta utilizada genera un archivo en formato json por cada imagen etiquetada. La estructura final de estos archivos se muestra en la Figura 14.

**Figura 14.** Archivos obtenidos luego de finalizar el etiquetado


Nota. Archivos resultantes del proceso de etiquetado. la imagen original y su correspondiente archivo en formato json con coordenadas y categorías asociadas. Imagen propia.

Finalmente, las etiquetas fueron convertidas al formato requerido por la arquitectura seleccionada y el conjunto se dividió en subconjuntos de entrenamiento, validación y prueba. Las categorías finales empleadas en el proceso de detección se resumen en la tabla 1.

**Tabla 1.** Categorías finales utilizadas en el proceso de detección de objetos.

| ID | Categoría |
|---|---|
| 0 | Con casco |
| 1 | Sin casco |
| 2 | Con chaleco |
| 3 | Sin chaleco |
| 4 | Con mascarilla |

Nota. En la tabla se presenta las categorías que se utilizan al momento del etiquetado así como su *ID* el cual es generando automáticamente por la aplicación

## 8.5. Recopilación de datos propios

Además de las bases de datos externas, fue necesario complementar el conjunto de imágenes mediante una recopilación propia realizada dentro de las instalaciones de la Universidad. Esta fase tuvo como propósito capturar escenarios reales, simulando las condiciones bajo las cuales se espera emplear el sistema final.

Las imágenes obtenidas presentan ambientes controlados y con buena iluminación, ya que este mismo entorno se proyecta para la implementación del sistema, como se aprecia en la Figura 15. La inclusión de datos representativos del entorno operativo incrementa la

validez del modelo y reduce el riesgo de *overfitting* en condiciones idealizadas o poco realistas [39].

**Figura 15.** Ejemplo de recopilación de datos propia

Nota. Fotografía capturada dentro de los laboratorios, simulando las condiciones reales de uso del sistema. Elaboración propia.

La recopilación de datos propia, presentada la tabla 2, permitió enriquecer el conjunto previamente recopilado, aportando casos específicos del entorno universitario. Esto es debido a la cantidad elevada de datos que se requieren para lograr un entrenamiento robusto. Cabe resaltar que todas las imágenes capturadas fueron etiquetadas siguiendo las mismas categorías definidas en la tabla 1.

**Tabla 2.** Cantidad de fotográficas tomadas por categoria

| **Cantidad** | **Categoría** |
|---|---|
| 100 | Con casco |
| 100 | Con mascarilla |
| 100 | Con chaleco |

Nota. Fotográficas tomadas dentro de la universidad implementando el uso de EPP

# CAPÍTULO 9

# Métodos de detección de objetos

Una vez conformado el conjunto de datos, se procedió a construir el modelo de detección. Tras evaluar distintas alternativas, se seleccionó la arquitectura *yolo* debido a su rapidez, precisión y facilidad de implementación, además de su capacidad para adaptarse a diferentes entornos.

## 9.1. Implementación de la arquitectura YOLO

La arquitectura *yolo* requiere que los datos estén organizados en una estructura específica de carpetas para entrenamiento y validación. La distribución general del Conjunto de datos se muestra en el Listado 9.1.

**Listing 9.1.** Estructura del dataset

```
dataset/
|-- train/
|   |-- images/
|   |-- labels/
|-- valid/
|   |-- images/
|   |-- labels/
|-- data.yaml
```

El archivo `data.yaml` es esencial, ya que especifica las rutas hacia las carpetas y define las categorías del conjunto de datos. Un ejemplo se presenta en el listado 9.2.

**Listing 9.2.** Archivo data.yaml

```
train: C:\\Users\\Direccion_carpeta\\
```

```
val: C:\\Users\\Direccion_carpeta\\

nc: 5  # Clases
names:
  0: Con_Casco
  1: Sin_Casco_Mascarilla
  2: Sin_Chaleco
  3: Con_Chaleco
  4: Con_Mascarilla
```

Es indispensable que los nombres de las clases coincidan exactamente con las etiquetas asignadas durante el proceso de etiquetado.

### 9.1.1. Preparación del conjunto de datos

Las imágenes fueron inicialmente anotadas en formato `.JSON`. Sin embargo, la arquitectura *yolo* requiere anotaciones en formato `.txt`. Ambos formatos contienen la misma información: clase y coordenadas normalizadas. Para la conversión se utilizó el script `jsonToTxt.py`, disponible en el repositorio [40].

Un ejemplo del formato final se muestra en el siguiente listado:

**Listing 9.3.** Archivo de etiquetas en formato .txt

```
0 0.658594 0.202344 0.198438 0.330469
3 0.692187 0.617188 0.273438 0.494531
```

Cada fila representa un objeto detectado, donde el primer valor corresponde a la clase y los siguientes especifican las coordenadas de la región de interés mediante *bounding box*.

# CAPÍTULO 10

## Entrenamiento y validación

El proceso de entrenamiento constituye una etapa fundamental en el desarrollo de sistemas de visión artificial, donde se determina la capacidad del modelo para extraer patrones relevantes a partir de los datos. En este capítulo se presenta el procedimiento seguido para entrenar el modelo de detección de objetos utilizando en la interfaz final.

## 10.1. Entrenamiento inicial

Con la estructura definida en los Listados 9.1 y 9.2, el siguiente paso fue instalar la librería *ultralytics*, desarrolladora del modelo presente en el Codigo 15.4

En primer lugar se selecciona el modelo `yolov8n` y se define un número reducido de épocas para evaluar el desempeño inicial. Los parámetros se presentan en la tabla 3.

**Tabla 3.** Parámetros generales para el primer entrenamiento

| **Modelo** | **Épocas** | **Tarea** |
|---|---|---|
| `yolov8n` | 20 | Segmentación |

Nota. Resumen de parámetros generales para el primer entrenamiento del modelo, se busca conocer como responde el modelo ante nuestro colección de datos antes de poder llegar a implementarlo.

El comando utilizado para iniciar el proceso se muestra en el Listado 10.1.

**Listing 10.1.** Comando de entrenamiento YOLO

```
yolo task=segment mode=train epochs=20 \
data=data.yaml model=yolov8n-seg.pt imgsz=640 batch=2
```

Cada parámetro del comando se detalla en la tabla 4.

**Tabla 4.** Descripción de parámetros del entrenamiento

| **Parámetro** | **Descripción** |
|---|---|
| `task=segment` | Define la tarea del modelo. |
| `mode=train` | Indica que el modo ejecutado es el de entrenamiento. |
| `epochs=15` | Número de iteraciones completas sobre el conjunto de entrenamiento. |
| `data=data.yaml` | Archivo con rutas y definición de clases. |
| `model=yolov8n-seg.pt` | Modelo base preentrenado. |
| `imgsz=640` | Resolución de entrada de las imágenes durante el entrenamiento. |
| `batch=2` | Cantidad de imágenes procesadas simultáneamente. |

Nota. Descripción especifica de cada parámetro para el entrenamiento del modelo, se debe corroborar que cada uno sea acorde a nuestro colección de datos.

Esta configuración permitió verificar la correcta lectura del conjunto de datos y obtener un desempeño preliminar del modelo. Posteriormente resultó posible ajustar hiperparámetros y emplear arquitecturas más complejas para mejorar el rendimiento.

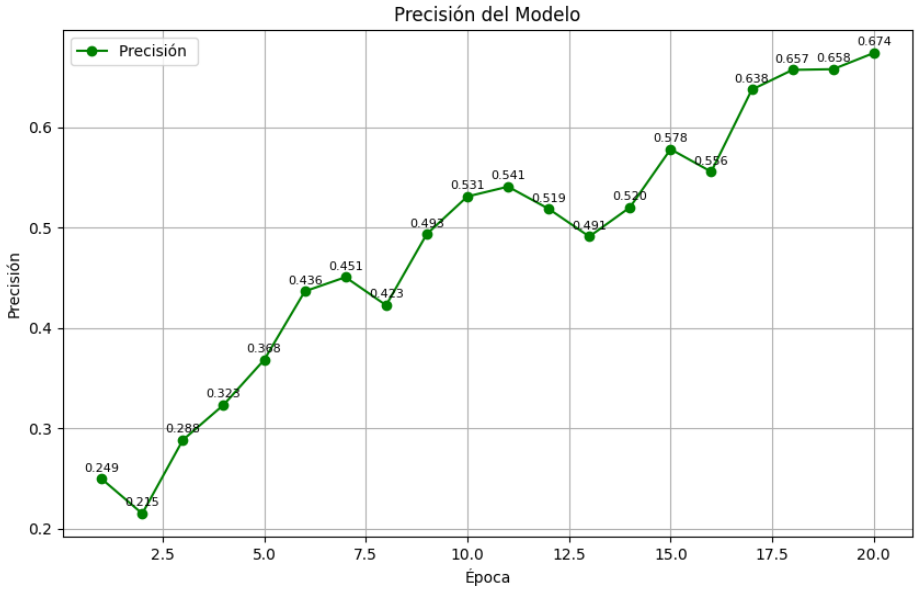
## 10.2. Validación preliminar

La etapa de validación tuvo como objetivo evaluar el desempeño inicial del modelo entrenado, determinar su capacidad para generalizar ante nuevas imágenes y comprobar su utilidad en escenarios reales. Para ello, se empleó el subconjunto de validación previamente separado del Conjunto de datos. Esta fase permitió identificar fortalezas, limitaciones y oportunidades de mejora del sistema antes de avanzar con el modelo.

### 10.2.1. Resultados de validación preliminar

La validación preliminar se realizó con el propósito de analizar el comportamiento del modelo durante su primera etapa de entrenamiento, utilizando la colección de datos disponible en esta fase. Esta evaluación temprana resulto útil para identificar deficiencias en las anotaciones, la distribución de clases o el preprocesamiento, lo que permite realizar ajustes sin invertir tiempo adicional.

**Figura 16.** Resultados obtenidos del primer entrenamiento del modelo




Nota. Gráfica representativa del comportamiento del modelo durante el primer entrenamiento. Elaboración propia.

En la Figura 17 se observa una tendencia creciente en la métrica de precisión conforme avanzan las épocas de entrenamiento. La precisión inicial, cercana al 25 %, evidencia que el modelo comenzó con un desempeño limitado, propio de una red sin patrones aprendidos. A medida que avanzó el proceso, la métrica mostró incrementos graduales hasta alcanzar aproximadamente un 67 % en la última época. Aunque este comportamiento indica que el modelo logró aprender algunas características relevantes del conjunto de datos, también revela fluctuaciones notorias entre ciertas épocas, lo que sugiere inestabilidad en el aprendizaje.

Dado que este resultado corresponde únicamente al primer entrenamiento, el nivel de precisión alcanzado aún no es satisfactorio para una implementación práctica. La gráfica evidencia la necesidad de mejorar el conjunto de datos.

Para identificar con mayor claridad las limitaciones del modelo, se examinaron los resultados obtenidos sobre el conjunto de validación. Esta revisión permite evaluar el desempeño del sistema en imágenes no vistas previamente y determinar si las clases están siendo reconocidas de forma correcta.

**Figura 17.** Resultados de validación del primer entrenamiento

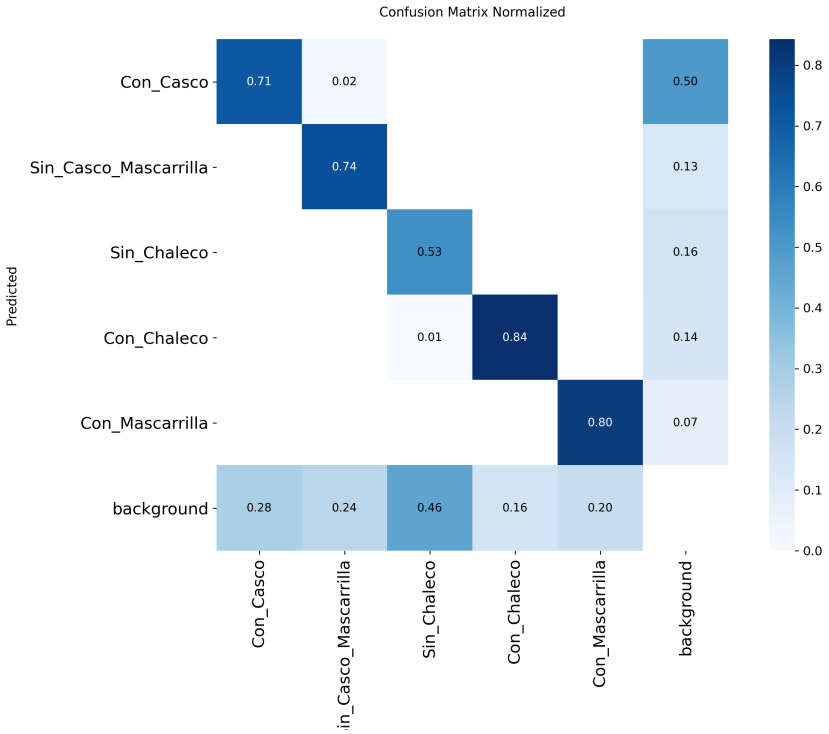

Nota. Resultados visuales generados por el modelo al aplicarse sobre el conjunto de validación. Elaboración propia.

En la Figura 17 se aprecia que el modelo presenta múltiples inconsistencias al clasificar las categorías de casco, chaleco y mascarilla. Aunque en algunos casos logra delimitar correctamente los rostros y cuerpos, las etiquetas asignadas muestran valores de confianza bajos y confusiones frecuentes entre clases. Por ejemplo, varias imágenes presentan predicciones simultáneas como "Sin Casco_Mascarilla" o "Sin Chaleco" cuando visualmente la persona sí porta parte del equipo, lo que evidencia dificultades para diferenciar patrones relevantes en situaciones variadas.

Asimismo, se observa una tendencia a sobreetiquetar objetos, generando detecciones redundantes sobre una misma persona o asignando clases incorrectas a individuos en segundo plano. Este comportamiento sugiere que el modelo aún no ha aprendido características distintivas suficientes para separar adecuadamente las clases.

Como último análisis, se examinó la matriz de confusión generada a partir del conjunto de validación. Permite evaluar el desempeño del modelo de forma detallada, ya que muestra la proporción de predicciones correctas e incorrectas para cada una de las clases consideradas.

**Figura 18.** Matriz de confusión del entrenamiento preliminar




Nota. Desempeño del modelo según predicciones, los valores más altos en la diagonal indican una correcta clasificación, mientras que los valores fuera de ella reflejan errores entre clases. Imagen generada junto los resultado de entrenamiento.

En la Figura 18 se observa que algunas clases, como "Con_Chaleco" y "Con_Mascarilla", presentan niveles de acierto relativamente altos (cercanos al 0.80), lo cual indica que el modelo logró identificar ciertos patrones relevantes de estas categorías. Sin embargo, otras clases muestran comportamientos más inconsistentes. Por ejemplo, la clase "Sin_Chaleco" presenta una dispersión notable de predicciones, siendo clasificada erróneamente como otras clases en una proporción considerable.

Un aspecto importante es la categoría *background*. Esta clase representa regiones o imágenes donde no existe ningún elemento perteneciente a las clases de establecidas. Su función es permitir que el modelo identifique cuándo no debe generar una detección. En YOLO, el *background* no se predice explícitamente; cuando no existe un objeto, el modelo simplemente no produce ninguna etiqueta. Por ello, no aparece una celda *background vs background* en la matriz, ya que dicha predicción no existe.

En esta matriz se observa que el fondo está siendo confundido con clases como "Sin_Casco", "Sin_Chaleco" y "Sin_Casco_Mascarilla", lo que evidencia la generación de falsos positivos incluso cuando no existe un objeto en la imagen. Esta situación sugiere que el modelo aún no logra separar adecuadamente las clases, posiblemente debido a anotaciones incompletas

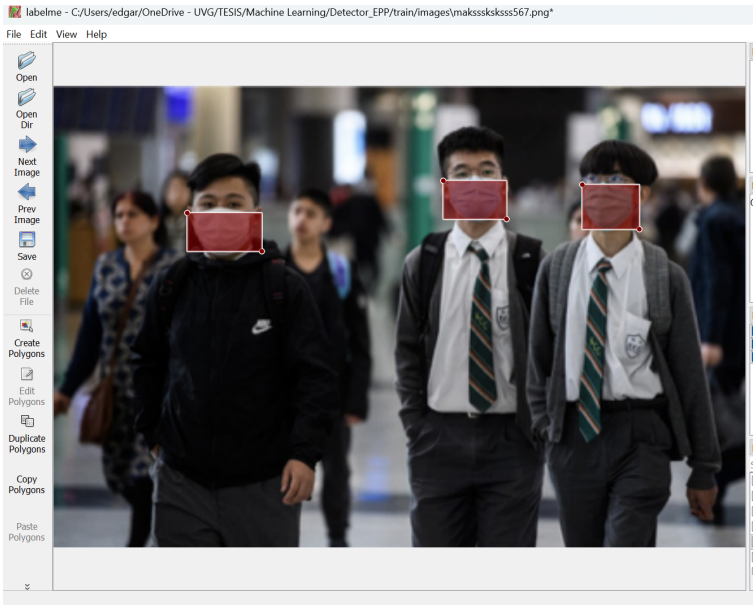
### 10.2.2. Modificaciones según resultados preliminares

Luego de verificar el comportamiento del modelo en tiempo real, se observó que presentaba dificultades notorias en la detección de los objetos localizados en la parte superior del cuerpo, específicamente las clases correspondientes a casco y mascarilla. Este problema se manifestó principalmente como un solapamiento entre ambas regiones, lo que generaba confusiones frecuentes en las predicciones y disminuía la precisión del modelo.

Para mejorar esta separación, fue necesario realizar ajustes en el proceso de etiquetado de las imágenes. El cambio principal consistió en modificar las anotaciones de la clase "Con_-Mascarilla", desplazando la *bounding box* ligeramente hacia la zona del mentón con el fin de delimitar únicamente la región correspondiente a la mascarilla.

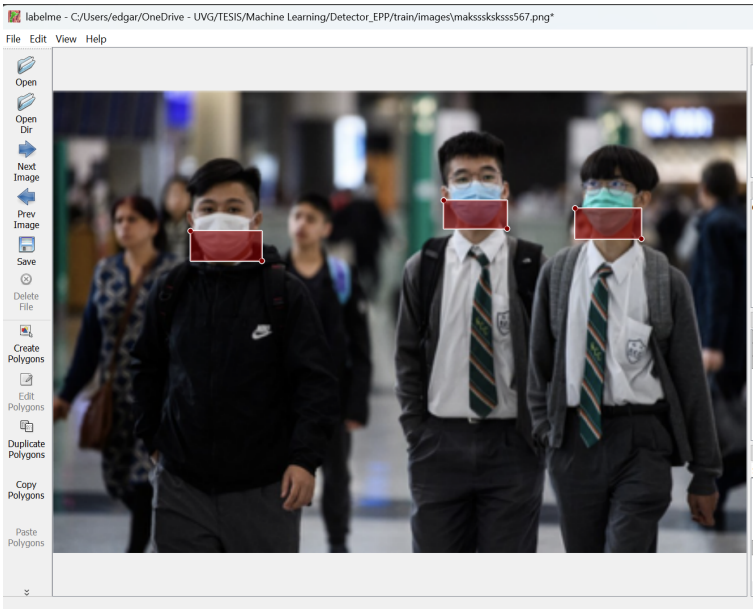
En la Figura 19 se muestra el etiquetado previo, donde es evidente que la *bounding box* de la mascarilla abarcaba una gran zona de la cabeza. Por otro lado, la Figura 20 presenta el etiquetado corregido, donde la región asociada a la mascarilla se encuentra delimitada de forma más precisa.

**Figura 19.** Etiquetado original de la clase "con_mascarilla" antes del ajuste




Nota. Primer etiquetado realizando específicamente de la clase *Con Mascarilla*, se observa que se ocupada una gran cantidad de la cara para especificar el objeto de interés. Elaboración propia.

**Figura 20.** Etiquetado corregido de la clase "con_mascarilla" después del ajuste




Nota. Segundo etiquetado realizando específicamente de la clase *Con Mascarilla*, se redujo parte de área utilizada, tomando específicamente la parte del mentón de cada rostro presente. Elaboración propia.

## 10.3. Entrenamiento con colección de datos modificada

Una vez realizadas las correcciones en el etiquetado de la clase "con_mascarilla", se procedió a ejecutar un nuevo entrenamiento con el objetivo de evaluar el efecto que estas presentan sobre el desempeño del modelo. Para asegurar una comparación directa con los resultados preliminares, se vuelve a realizar el entrenamiento con los mismos parámetros de entrenamiento.

Los parámetros se presentan nuevamente en la tabla 5, manteniendo el mismo modelo, cantidad de épocas y configuración general.

**Tabla 5.** Parámetros del entrenamiento posterior a la modificación del conjunto de datos

| Modelo | Épocas | Tarea |
|---|---|---|
| `yolov8n` | 20 | Segmentación |

Nota. Se vuelve a utilizar los mismos parámetros del primer entrenamiento para validar la evolución que el modelo tuvo luego de las modificaciones.

El comando empleado para este entrenamiento,es idéntico al utilizado en la fase preliminar, con el fin de validar exclusivamente el impacto de las mejoras en el conjunto de datos.

## 10.4. Validación con colección de datos modificada

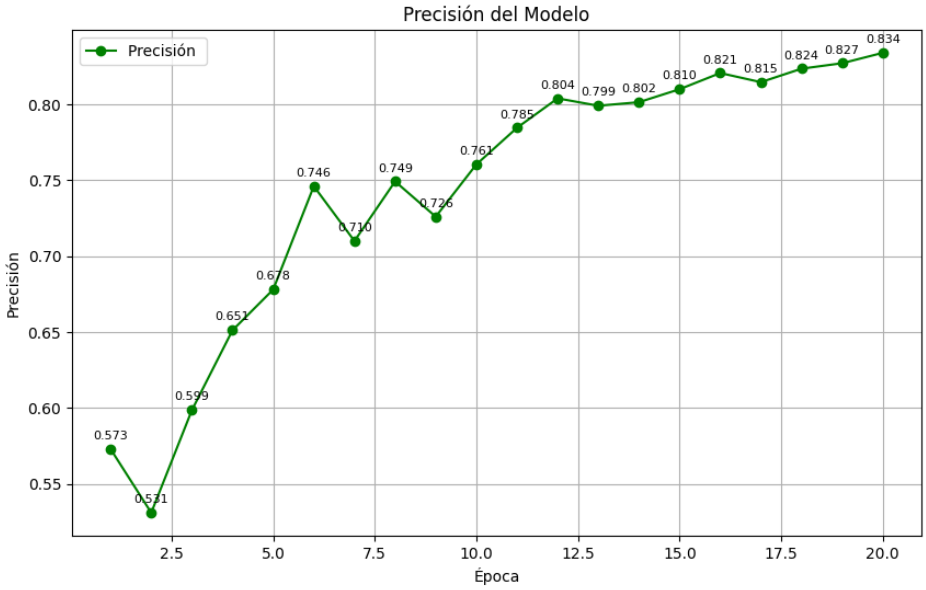
Una vez completado el nuevo entrenamiento utilizando el conjunto de datos modificado, se llevó a cabo una segunda etapa de validación con el propósito de determinar si las modificaciones realizadas fueron relevantes.

### 10.4.1. Resultados de validación con datos modificados

Nuevamente se analizaron las gráficas generadas durante el entrenamiento realizado con la colección de datos modificada. La Figura 21 muestra la evolución de la precisión del modelo a lo largo de las épocas, lo cual permite evaluar si las modificaciones aplicadas en las anotaciones contribuyeron a mejorar su comportamiento durante el aprendizaje.

En dicha gráfica se aprecia una tendencia ascendente más estable en comparación con los resultados obtenidos durante la validación preliminar. La precisión inicia en valores moderados y presenta incrementos progresivos hasta superar el 0.83 en las últimas épocas. El crecimiento sostenido de la curva sugiere que las correcciones realizadas en la clase "Con_Mascarilla" mejoraron la separación entre clases.

**Figura 21.** Resultados obtenidos del modelo con colección de datos modificada




Nota. Evolución del modelo luego de completar la totalidad de las epocas de entrenamiento, empleando la colección de datos modificada. Elaboración propia.

La Figura 22 muestra los resultados de la validación del modelo. Se observan algunos casos en los que las detecciones de "Con_Casco" y "Con_Chaleco" presentan una mayor

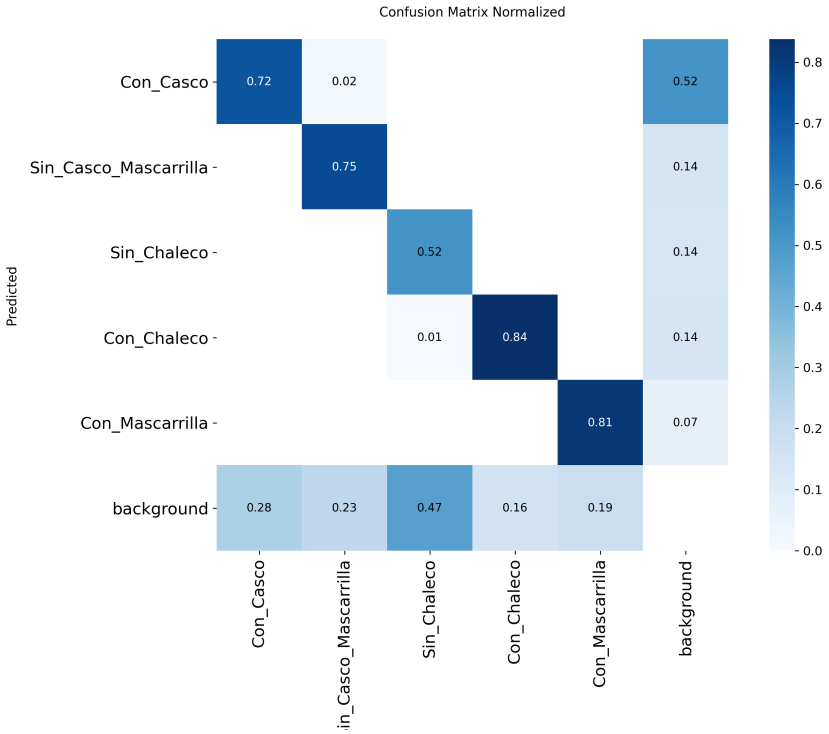
estabilidad, persisten errores importantes. Todavía se generan etiquetas incorrectas en imágenes donde las personas sí portan equipo, se generan detecciones duplicadas sobre un mismo individuo y se mantienen valores de confianza bajos en algunas predicciones.

**Figura 22.** Resultado de validación del modelo con el ajuste del conjunto de datos


Nota. Detecciones obtenidas por el nuevo modelo empleando la parte de sección de validación de la colección de datos, Imagen generada junto los resultado de entrenamiento.

Mientras que en la Figura 23 se obtuvo la matriz de confusión generada durante esta etapa. Se observa una ligera mejora en clases como “Con_Casco” y “Con_Mascarilla”, que alcanzan valores alrededor de 0.72 y 0.81 respectivamente, así como una mayor consistencia en “Con_Chaleco”. Aunque, clases como “Sin_Chaleco” continúan mostrando inestabilidad y confusiones frecuentes.

**Figura 23.** Matriz de confusión posterior al entrenamiento con datos modificados




Nota. Desempeño del modelo según predicciones, se busca comparar la respuesta entre clases. Imagen generada junto los resultado de entrenamiento.

En general, los resultados muestran mejoras puntuales derivadas de la corrección realizadas, sin embargo, el modelo aún presenta inconsistencias relevantes. Persisten confusiones entre clases y se mantiene la generación de falsos positivos.

## 10.5. Entrenamiento con modelo robusto

Luego de analizar los resultados obtenidos con los modelos diseñados por *ultralytics*, y considerando las limitaciones presentes en la detección de ciertas clases, especialmente en las categorías de la parte superior del cuerpo, se decidió emplear una arquitectura más robusta con el fin de mejorar la estabilidad y precisión del sistema. Se seleccionó el modelo `YOLOv8L`, una variante con mayor capacidad de generalización y con un comportamiento más consistente en escenarios complejos.

El objetivo principal de este entrenamiento fue comparar de este nuevo modelo frente a la versiones anteriores de YOLO utilizada, manteniendo la misma estructura de la colección de datos modificada. Se espera una separación entre clases, una reducción de falsos positivos y un desempeño más uniforme.

La tabla 6 presenta las diferencias más relevantes entre el modelo `YOLOv8n`, utilizado

**Tabla 6.** Comparación entre `YOLOv8n` y `YOLOv8l`.

| **Aspecto** | **YOLOv8n (Nano)** | **YOLOv8l (Large)** |
|---|---|---|
| Tamaño del modelo | Muy ligero; diseñado para correr en dispositivos con recursos limitados. | Modelo grande; requiere más memoria y capacidad de cómputo. |
| Número de parámetros | Aproximadamente 3.2 millones. | Aproximadamente 43 millones. |
| Velocidad de inferencia | Extremadamente rápida, ideal para tiempo real incluso en hardware modesto. | Más lenta; adecuada para entornos con GPU potentes. |
| Consumo de GPU/CPU | Muy bajo. | Alto, requiere GPU equipada. |

Nota. Características más relevantes de cada modelo YOLO utilizando para el sistema, se observan los lineamientos necesarios para la implementación de cada de modelo.

en los entrenamientos iniciales y el modelo `YOLOv8l`, el cual posee una mayor capacidad de aprendizaje. No obstante, esta mejora en el rendimiento implica un incremento considerable en el uso de recursos computacionales y en los tiempos de entrenamiento. Como referencia, el entrenamiento con `YOLOv8l` requirió aproximadamente 2.2 horas, mientras que los entrenamientos preliminares con `YOLOv8n` tomaron alrededor de 0.35 horas.

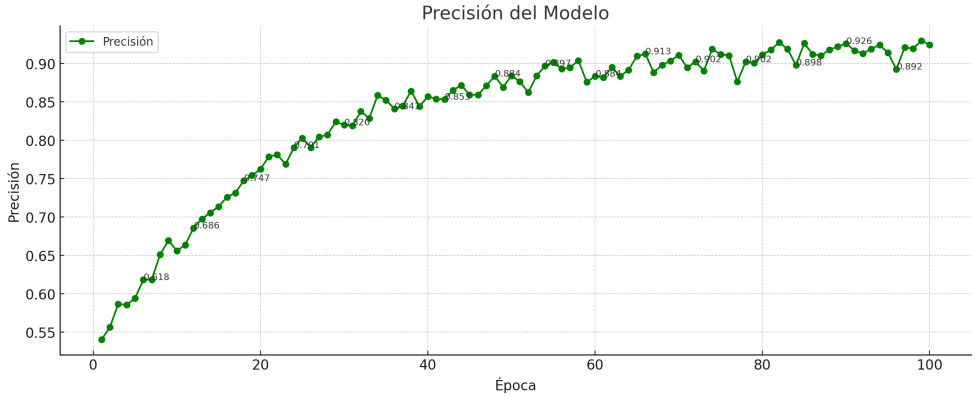
**Tabla 7.** Parámetros utilizados para el entrenamiento con `YOLOv8l`.

| **Modelo** | **Épocas** | **Tarea** |
|---|---|---|
| `yolo-l` | 100 | Segmentación |

Nota. Parámetros de entrenamiento finales con un modelo más robusto de la familia YOLO.

En la tabla 7 notamos los parámetros definidos para el entrenamiento con el modelo `YOLOv8l`, el cual requiere un mayor número de épocas debido a su capacidad superior de aprendizaje. En este caso se empleó el mismo comando utilizado previamente, modificando únicamente el modelo base y la cantidad de épocas según lo indicado en la tabla 7.

**Figura 24.** Resultados obtenidos con el modelo YOLO más robusto




Nota. Evoluciín final del modelo utilizando la base de datos modificada y un modelo preentrenado robusto. Elaboración propia.

La Figura 24 muestra la evolución del modelo durante el entrenamiento final. En esta gráfica se observa un comportamiento de aprendizaje estable, con incrementos progresivos que alcanzan valores superiores al 90 % en las últimas iteraciones. A diferencia de los entrenamientos previos, este modelo presenta una curva más consistente, con variaciones mínimas y una tendencia clara hacia la convergencia, lo que evidencia una mejor adaptación al conjunto de datos y una separación más precisa entre las clases.

# CAPÍTULO 11

## Interfaz gráfica e implementación del modelo de detección de objetos

Con el fin de integrar el modelo entrenado dentro de un entorno real, se desarrolló una interfaz gráfica en `Streamlit` que permite realizar la detección de equipo de protección personal (EPP) en tiempo real. Esta interfaz es el último componente del sistema, ya que combina la detección basada en Yolo con herramientas de visualización.

### 11.1. Manejo de cámara y captura de detecciones

El sistema identifica automáticamente las cámaras disponibles mediante `OpenCV` y presenta al usuario la opción de seleccionarlas desde la barra lateral. Una vez inicializada la captura, el video se procesa cuadro por cuadro para realizar la validación.

Además, la interfaz incorpora un mecanismo para capturar imágenes automáticamente cada vez que se detecta el uso completo de EPP. Estas capturas se almacenan en una carpeta creada en el escritorio del usuario, y la imagen más reciente se muestra en la barra lateral.

### 11.2. Distribución de la interfaz y detección en tiempo real

Para evaluar el desempeño final del modelo entrenado, se integró el sistema dentro de una interfaz gráfica desarrollada en `Streamlit`. Esta herramienta permitió visualizar en tiempo real los resultados del modelo, mostrar indicadores del uso de EPP y gestionar la comunicación con un dispositivo externo encargado de generar señal adicional. La Figura 25 muestra la apariencia general de la aplicación.

**Figura 25.** Interfaz gráfica de detección de EPP en tiempo real


Nota. Captura de pantalla del sistema en funcionamiento, mostrando detecciones, indicadores e interacción con la cámara y el puerto serial. Elaboración propia.

Se observa la estructura completa de la interfaz. En el panel lateral izquierdo se encuentra el módulo de conexión serial, que permite seleccionar el puerto `COM` y establecer comunicación con el microcontrolador receptor. Debajo se incluye la opción de seleccionar la cámara activa y se muestra la última detección completo del EPP almacenada por el sistema.

En el área central se presenta el video en tiempo real, procesado por el modelo de detección. Con cada cuadro capturado, el sistema identifica las clases objetivo (`Con_Casco`, `Con_Mascarilla` y `Con_Chaleco`) y superpone las etiquetas correspondientes con su nivel de confianza.A la derecha se encuentran los indicadores visuales correspondientes a las clases más relevantes. Cada indicador cambia de color según se detecte o no el elemento de protección personal respectivo.Finalmente, la interfaz incorpora un selector para ajustar el nivel de confianza para las predicciones, lo que permite estudiar e implementar el comportamiento del modelo en distintos escenarios y filtrar detecciones con valores bajos.

## 11.3. Conexión serial automática

La interfaz cuenta con un módulo de detección automática de puertos `COM`, permitiendo identificar y seleccionar el dispositivo conectado, todo esto con la finalidad de generar alertas externas cuando no se presente el uso del EPP.

CAPÍTULO 12

# Conclusiones

## 12.1. Conclusiones

- La cantidad de datos no garantiza un buen desempeño del modelo, la precisión depende principalmente de la calidad del etiquetado y de la representatividad del conjunto de datos en distintos escenarios reales.
- Las correcciones realizadas en las anotaciones, especialmente en las clases con mascarilla, mejoraron significativamente la separación entre categorías y redujeron la generación de falsos positivos.
- El uso de un modelo más robusto, como YOLOv8l, permitió obtener una precisión superior al 90 %, demostrando que arquitecturas de mayor capacidad ofrecen un desempeño más estable.
- La implementación de la interfaz gráfica en *Streamlit* permitió validar el modelo en tiempo real, facilitando su uso y demostrando la versatilidad del sistema en entornos reales.
- El sistema desarrollado establece una base para futuras mejoras, como la incorporación de nuevas clases de EPP, y principalmente la expansión de la colección de datos.

# CAPÍTULO 13

## Recomendaciones

- Utilizar un modelo YOLO de mayor capacidad (como YOLOv8l o YOLOv8x) desde las primeras etapas de entrenamiento con el fin de obtener resultados más estables y acelerar el análisis comparativo de precisión.
- Ampliar la colección de datos incorporando escenarios más realistas y variados, como diferentes condiciones de iluminación, ángulos de cámara, tipos de casco o mascarilla, y entornos industriales diversos.
- Evaluar la posibilidad de anidar nuevos elementos del EPP desde la interfaz gráfica para evitar realizar todo el proceso desde el comienzo.
- Incorporar un sistema de registro con el fin de contar con reportes estadísticos automáticos en base a los resultados del modelo.
- Integrar la interfaz con sistemas externos, como alarmas industriales a través de controladores tales como PLC.

# CAPÍTULO 14

## Referencias


[1] A. S. Morales, «Desarrollo de un traductor de Lengua de Señas de Guatemala (LENSEGUA) mediante visión por computadora y aprendizaje automático,» Trabajo de graduación de licenciatura, Universidad del Valle de Guatemala, 2024.

[2] G. A. Fong, «Implementación de infraestructura a escala para la evaluación de algoritmos por visión de computador para vehículos autónomos,» Trabajo de graduación de licenciatura, Universidad del Valle de Guatemala, 2023.

[3] X. Li, R. Nabati, K. Singh, E. Corona, V. Metsis y A. Parchami, «EMOD: Efficient Moving Object Detection via Image Eccentricity Analysis and Sparse Neural Networks,» en *2023 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, Los Alamitos, CA, USA: IEEE Computer Society, ene. de 2023, págs. 51-59. DOI: 10.1109/WACVW58289.2023.00010. dirección: https://doi.ieeecomputersociety.org/10.1109/WACVW58289.2023.00010.

[4] D. Maji, S. Nagori, M. Mathew y D. Poddar, « YOLO-Pose: Enhancing YOLO for Multi Person Pose Estimation Using Object Keypoint Similarity Loss,» en *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Los Alamitos, CA, USA: IEEE Computer Society, jun. de 2022, págs. 2636-2645. DOI: 10.1109/CVPRW56347.2022.00297. dirección: https://doi.ieeecomputersociety.org/10.1109/CVPRW56347.2022.00297.

[5] *Representación de visión con YOLOv8*, https://miro.medium.com/v2/resize:fit:1100/format:webp/0*A1bB6s2kyUuRB-W4.png, Imagen consultada en Medium, autoría y licencia no identificados, n.d.

[6] L. Barba-Guaman, J. E. Naranjo y A. Ortiz, «Object detection in rural roads using Tensorflow API,» en *2020 International Conference of Digital Transformation and Innovation Technology (Incodtrin)*, IEEE, oct. de 2020, págs. 84-88, ISBN: 978-1-6654-2319-9. DOI: 10.1109/Incodtrin51881.2020.00028.

[7] M. P. Groover, *Automation, Production Systems, and Computer-Integrated Manufacturing*, 4.ª ed. Pearson, 2015, ISBN: 978-0-13-349961-2.

[8] J. Barreiro, *Seguridad e Higiene Industrial*. McGraw-Hill, 2019, ISBN: 978-607-150-799-3.

[9] C. de la República de Guatemala, *Código de Trabajo de Guatemala*, https://mintrabajo.gob.gt, Accedido: 2025-11-25, 2022.

[10] Instituto Guatemalteco de Seguridad Social, *Reglamento de Riesgos Profesionales del IGSS*, https://www.igssgt.org, Accedido: 2025-11-25, 2021.

[11] International Organization for Standardization, *ISO 45001:2018 — Occupational Health and Safety Management Systems*, https://www.iso.org/standard/63787.html, Requisitos para sistemas de gestión de seguridad y salud en el trabajo, 2018.

[12] *Imagen*, https://encrypted-tbn3.gstatic.com/images?q=tbn:ANd9GcSnCB59Z_w5fOfonBM3zgRT1obCX6HMLtHOZslSrgOFdmTEcHd6, Accedido: 27 de septiembre de 2025, 2025.

[13] *Aplicación de clasificación de objeto mediante YOLOv8*, https://miro.medium.com/v2/resize:fit:1100/format:webp/0*oeGTIQ8KAxhCqdhl.png, Imagen consultada en Medium, autoría y licencia no identificados, n.d.

[14] R. C. Gonzalez y R. E. Woods, *Digital Image Processing*, 4.ª ed. Pearson, 2018, ISBN: 978-0-13-335672-4.

[15] I. Goodfellow, Y. Bengio y A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[16] Y. LeCun, Y. Bengio y G. Hinton, «Deep Learning,» *Nature*, vol. 521, n.º 7553, págs. 436-444, 2015. DOI: 10.1038/nature14539.

[17] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection,» en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, págs. 779-788.

[18] K. He, G. Gkioxari, P. Dollár y R. Girshick, «Mask R-CNN,» *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, n.º 2, págs. 386-397, 2017. DOI: 10.1109/TPAMI.2018.2844175.

[19] Y. LeCun, Y. Bengio y G. Hinton, «Deep learning,» *Nature*, vol. 521, n.º 7553, págs. 436-444, 2015.

[20] D. E. Rumelhart, G. E. Hinton y R. J. Williams, «Learning representations by back-propagating errors,» *Nature*, vol. 323, n.º 6088, págs. 533-536, 1986.

[21] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

[22] J. Schmidhuber, «Deep learning in neural networks: An overview,» *Neural Networks*, vol. 61, págs. 85-117, 2015.

[23] K. Wada, *LabelMe: Image Polygonal Annotation Tool*, https://github.com/wkentaro/labelme, 2016.

[24] R. Girshick, «Fast R-CNN,» en *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, págs. 1440-1448.

[25] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, «You only look once: Unified, real-time object detection,» en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, págs. 779-788.

[26] W. Liu et al., «SSD: Single shot multibox detector,» en *European Conference on Computer Vision (ECCV)*, 2016, págs. 21-37.

[27] R. Girshick, J. Donahue, T. Darrell y J. Malik, «Rich feature hierarchies for accurate object detection and semantic segmentation,» en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, págs. 580-587.

[28] A. Krizhevsky, I. Sutskever y G. E. Hinton, «ImageNet classification with deep convolutional neural networks,» en *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, 2012, págs. 1097-1105.

[29] S. Ren, K. He, R. Girshick y J. Sun, «Faster R-CNN: Towards real-time object detection with region proposal networks,» en *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015.

[30] Ultralytics, *YOLOv8 Capabilities*, `https://tinyurl.com/yolov8capabilities`, Imagen promocional de la arquitectura YOLOv8, 2025.

[31] Ultralytics, *YOLO Data Augmentation Guide*, `https://docs.ultralytics.com/es/guides/yolo-data-augmentation/`, Accedido: 16 de noviembre de 2025, 2024.

[32] Ultralytics, *Albumentations Augmentation (imagen) — Ultralytics Docs v0*, `https://github.com/ultralytics/docs/releases/download/0/albumentations-augmentation.avif`, Imagen incluida como parte del lanzamiento inicial de la documentación de Ultralytics Docs., 2024.

[33] P. S. Foundation, *Python Programming Language – Official Website*, Accedido: 27 de septiembre de 2025, 2024. dirección: `https://www.python.org`.

[34] JetBrains, *PyCharm: the Python IDE for Professional Developers*, Accedido: 27 de septiembre de 2025, 2024. dirección: `https://www.jetbrains.com/pycharm/`.

[35] JetBrains, *PyCharm Download Image*, Accedido: 27 de septiembre de 2025, 2025. dirección: `https://www.jetbrains.com/pycharm/download/img/download.png`.

[36] PyTorch Foundation, *Get Started Locally with PyTorch*, `https://pytorch.org/get-started/locally/`, Accedido: 27 de septiembre de 2025, 2025.

[37] M.-L. Huang e Y. Cheng, *Dataset of Personal Protective Equipment (PPE)*, ver. V2, Mendeley Data, 2025. DOI: `10.17632/zkzghjvpn2.2`. dirección: `https://doi.org/10.17632/zkzghjvpn2.2`.

[38] A. MVD, *Face Mask Detection*, Accedido: 24 de septiembre de 2025, 2025. dirección: `https://www.kaggle.com/datasets/andrewmvd/face-mask-detection`.

[39] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, 2010. DOI: `10.1007/978-1-84882-935-0`.

[40] AleG13, *Sistema de detección de equipo de protección personal (Python)*, `https://github.com/AleG13/Sistema-de-detecci-n-de-equipo-de-protecci-n-personal-Python`, Repositorio GitHub, versión X.Y.Z, consultado el DD Mes YYYY, 2025.

# CAPÍTULO 15

# Anexos

## 15.1. Verificación de versión de Python

```
python --version
```

## 15.2. Instalación de bibliotecas de software

```
pip install torchvision
pip install opencv-python
pip install numpy
pip install matplotlib
```

## 15.3. Instalación de PyTorch con CUDA

```
pip3 install torch torchvision --index-url \
https://download.pytorch.org/whl/cu126
```

## 15.4. Instalación del framework Ultralytics

```
pip install ultralytics
```

## 15.5. Comando de entrenamiento utilizado con YOLOv8n

```
yolo task=segment mode=train \
data=data.yaml model=yolov8n-seg.pt \
epochs=20 imgsz=640 batch=2
```

## 15.6. Comando de entrenamiento utilizado con YOLOv8l

```
yolo task=segment mode=train \
data=data.yaml model=yolov8l-seg.pt \
epochs=100 imgsz=640 batch=2
```

## 15.7. Estructura de la colección de datos y archivos de configuración

Se establece la estructura necesaria del conjunto de datos y los archivos de configuración utilizados para el entrenamiento del modelo.

### 15.7.1. Estructura completa del dataset

```
dataset/
|-- train/
|   |-- images/
|   |-- labels/
|-- valid/
|   |-- images/
|   |-- labels/
|-- test/
|-- data.yaml
```

### 15.7.2. Archivo `data.yaml` completo

```
train: C:\\Users\\Direccion_carpeta\\train\\images
val: C:\\Users\\Direccion_carpeta\\valid\\images
nc: 5
names:
  0: Con_Casco
  1: Sin_Casco_Mascarilla
  2: Sin_Chaleco
  3: Con_Chaleco
  4: Con_Mascarilla
```

### 15.7.3. Ejemplo de archivo de etiquetas en formato `.txt`

```
0 0.658594 0.202344 0.198438 0.330469
3 0.692187 0.617188 0.273438 0.494531
```

## 15.8. Resultados intermedios del entrenamiento

De acuerdo con el reglamento, los resultados intermedios (validaciones preliminares, curvas iniciales y matrices de confusión parciales) deben colocarse en anexos para no sobrecargar el desarrollo principal.

### 15.8.1. Curva de precisión del entrenamiento preliminar

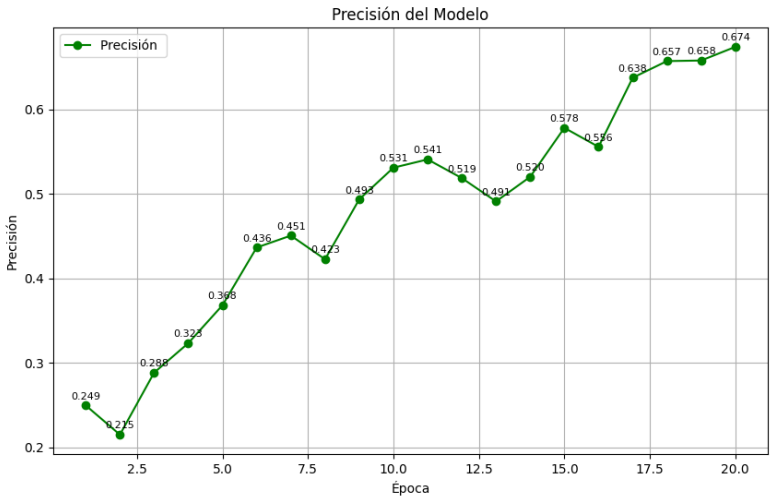



**Figura 26.** Curva de precisión del entrenamiento preliminar.

### 15.8.2. Validación preliminar del modelo


**Figura 27.** Resultados visuales de la validación preliminar.

### 15.8.3. Matrices de confusión intermedias

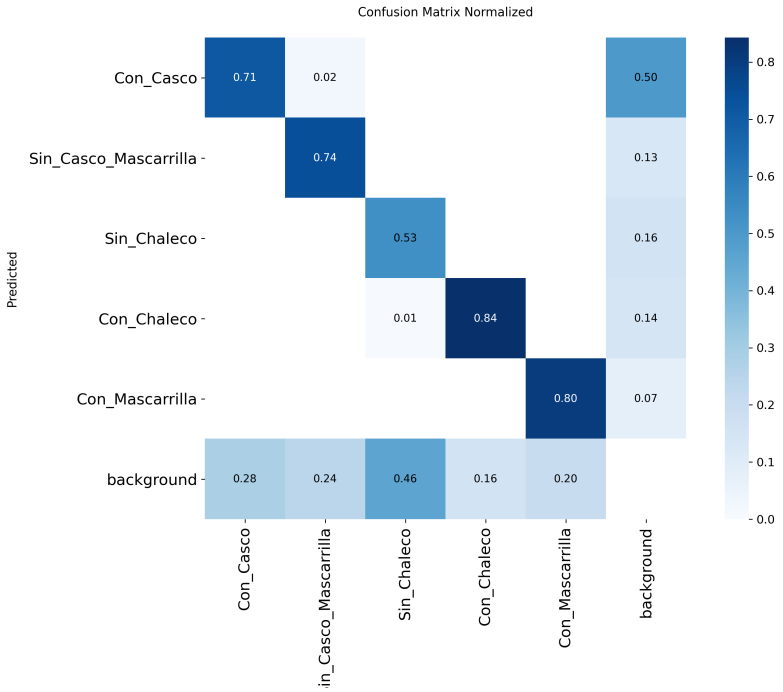



**Figura 28.** Matriz de confusión preliminar del modelo YOLOv8n.

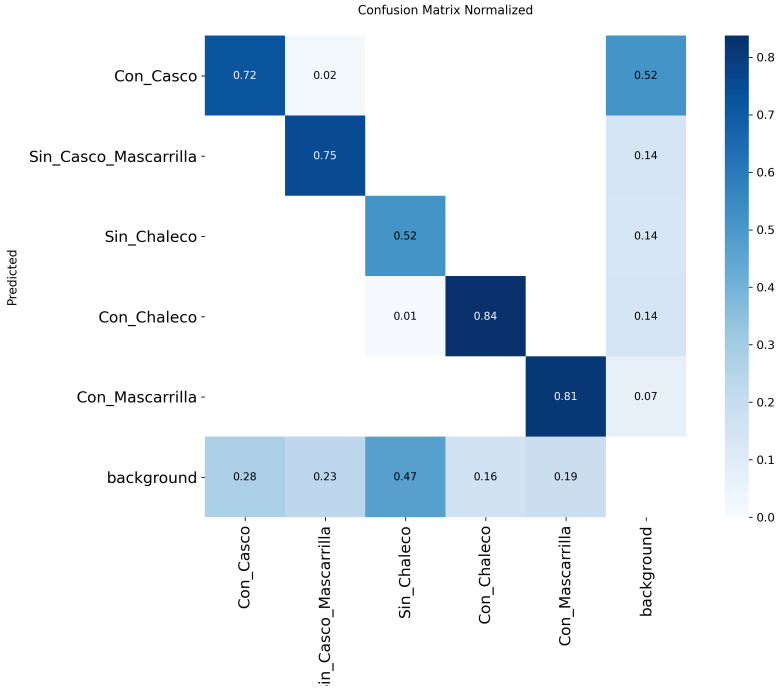



**Figura 29.** Matriz de confusión posterior al ajuste de etiquetas.

## 15.9. Herramientas utilizadas

### 15.9.1. Entorno de desarrollo PyCharm

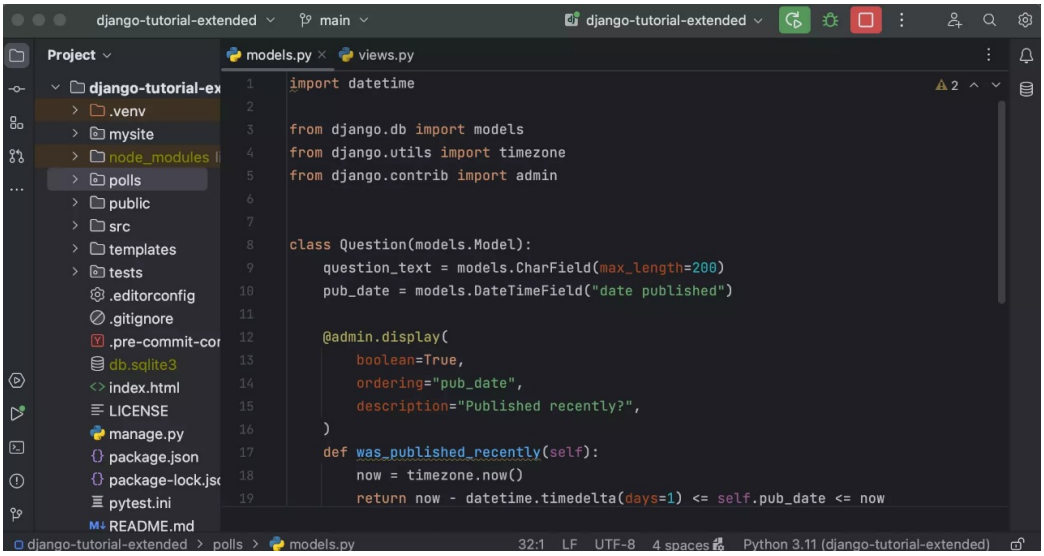



**Figura 30.** Captura del entorno de desarrollo utilizado (PyCharm).

# CAPÍTULO 16

## Glosario

***bounding box*** Rectángulo delimitador que indica la posición de un objeto dentro de una imagen. 11, 13, 14, 22, 26, 32

***cuda*** Plataforma de cómputo paralelo desarrollada por NVIDIA que permite utilizar la GPU para acelerar operaciones intensivas en el entrenamiento de modelos. 17

***cudnn*** Biblioteca de NVIDIA optimizada para acelerar operaciones de redes neuronales profundas, especialmente convoluciones, sobre GPU. 17

***framework*** Conjunto estructurado de herramientas y librerías diseñado para facilitar el desarrollo y la implementación de aplicaciones o modelos. 12, 17

***overfitting*** Fenómeno en el que un modelo de aprendizaje automático se ajusta excesivamente a los datos de entrenamiento, perdiendo capacidad de generalización y mostrando bajo rendimiento en datos nuevos. 15, 24

***pytorch*** Framework de aprendizaje profundo utilizado para construir y entrenar redes neuronales en CPU o GPU. 13, 17

***ultralytics*** Organización responsable de desarrollar los modelos modernos de la familia YOLO y herramientas relacionadas. 27, 36

**aumento de datos** Técnica que genera variaciones de imágenes existentes para incrementar la diversidad del conjunto de entrenamiento y mejorar la generalización del modelo. 14, 15

**conjunto de datos** Colección organizada de imágenes y etiquetas empleada para entrenar, validar y evaluar modelos de aprendizaje automático. 10, 15, 18–20, 25, 28

**gpu** Unidad de procesamiento gráfico diseñada para realizar operaciones matemáticas intensivas y acelerar el entrenamiento de modelos de visión por computadora. 17

**json** Formato de intercambio de datos basado en texto utilizado para almacenar información estructurada como anotaciones de imágenes. 13, 22, 23

**yolo** Arquitectura de detección de objetos optimizada para realizar predicciones rápidas en tiempo real. 39