



# **Universidad Tecnológica Nacional**

## **Facultad Regional Rosario**

**Soporte  
Ciclo Lectivo 2024**

**TPI  
Sistema Detección PPE  
Grupo 08**

### **Alumnos:**

Chiara, Agostina	48843	<a href="mailto:agosschiara25@gmail.com">agosschiara25@gmail.com</a>
Matteucci, Andrea	49419	<a href="mailto:andymatteucci2003@gmail.com">andymatteucci2003@gmail.com</a>
Tulian, Laura	46904	<a href="mailto:lau.tulian01@gmail.com">lau.tulian01@gmail.com</a>

### **Docentes:**

Torres, Juan Ignacio  
Castagnino, Mario

<b>Introducción</b>	<b>2</b>
<b>Objetivo</b>	<b>2</b>
<b>Metodología</b>	<b>3</b>
<b>Requerimientos</b>	<b>3</b>
Funcionales	3
No Funcionales	4
<b>Reglas de Negocio</b>	<b>4</b>
<b>Caso de Uso</b>	<b>5</b>
<b>Modelo de dominio</b>	<b>6</b>
<b>Imágenes</b>	<b>6</b>
Resultados entrenamiento del modelo	6
Funcionamiento detección	9
<b>Stack Tecnológico</b>	<b>10</b>
<b>Documentación</b>	<b>11</b>
<b>Código Fuente</b>	<b>11</b>

# Introducción

En ámbitos laborales como la construcción y la industria manufacturera donde la seguridad es primordial, el uso adecuado de equipo de protección personal (EPP) por parte de los trabajadores y operarios es crucial para minimizar los riesgos de accidentes. Como resultado, se posibilita la disminución del costo económico, social y ético generado por dichos accidentes laborales.

El EPP regulado para una cierta actividad puede incluir elementos como guantes, gafas o calzado de seguridad, tapones para los oídos u orejeras, cascos, respiradores, o monos, chalecos y trajes de cuerpo completo, entre otros.



Este proyecto se centra en desarrollar un sistema automatizado que utiliza técnicas de visión por computadora con OpenCV y el modelo de detección de objetos YOLO para monitorear el uso correcto de EPP, específicamente nos centraremos en cascos, chalecos, orejeras, lentes de seguridad y botas, en los trabajadores.

## Objetivo

El objetivo principal de este proyecto es implementar un sistema que pueda analizar imágenes y videos en tiempo real para detectar si los trabajadores están usando el equipo de protección requerido. Esto permitirá mejorar la supervisión de las normas de seguridad, reducir los incidentes y asegurar un ambiente de trabajo más seguro.

# Metodología

El sistema utiliza el modelo YOLO, entrenado para reconocer clases específicas de EPP, ya mencionadas. La detección se realiza en tiempo real mediante el uso de OpenCV, lo que permite la captura y procesamiento continuo de imágenes y videos de las zonas de trabajo.

## 1. Entrenamiento del Modelo:

- Para entrenar el modelo YOLO, se utilizó una combinación de Grounding DINO y Roboflow.
  - **Grounding DINO** se empleó para generar etiquetas de alta calidad y localizar de manera precisa los elementos de equipamiento de protección personal en las imágenes.
  - **Roboflow** se utilizó para gestionar, aumentar y preprocesar el conjunto de datos, asegurando una variedad adecuada de imágenes y un etiquetado coherente. Facilitó la creación de un conjunto de datos robusto, incluyendo técnicas de aumento de datos como rotación, escalado y ajustes de iluminación, para mejorar la capacidad del modelo para generalizar en diversas condiciones del entorno laboral.

## 2. Implementación del Sistema:

- Permite la detección tanto en imágenes en vivo capturadas desde cámaras en tiempo real, como en imágenes estáticas y vídeos previamente grabados. Si se detecta que un trabajador no lleva algún elemento de seguridad, el sistema está configurado para activar una alarma, alertando inmediatamente a los supervisores de seguridad. Además, el sistema puede capturar y guardar una imagen del momento exacto de la infracción, proporcionando evidencia visual que puede ser utilizada para análisis posteriores o para la implementación de medidas correctivas

## 3. Resultados y Beneficios:

- El sistema proporciona un monitoreo continuo y automatizado, eliminando la necesidad de inspecciones manuales constantes.
- Mejora el cumplimiento de las normas de seguridad, disminuyendo los riesgos de accidentes laborales.
- Permite la generación de reportes y estadísticas sobre el uso del EPP, ayudando en la toma de decisiones y la implementación de medidas correctivas.

# Requerimientos

## Funcionales

- El sistema debe permitir que los usuarios se autentiquen mediante un formulario de inicio de sesión, con campos para ingresar un nombre de usuario y contraseña.

- El sistema debe permitir que nuevos usuarios se registren creando una cuenta, indicando su nombre de usuario y contraseña.
- El sistema debe validar que el nombre de usuario no esté duplicado.
- El sistema debe validar las credenciales y, si son correctas, redirigir al usuario a la pantalla principal. Si no son válidas, debe mostrar un mensaje de error.
- El sistema debe permitir la detección de objetos de seguridad (casco, guantes, chaleco, etc.) en tiempo real utilizando la cámara del dispositivo.
- El sistema debe permitir cargar imágenes desde el dispositivo para realizar la detección de objetos de seguridad.
- El sistema debe permitir la carga de videos para procesar y detectar objetos de seguridad en cada fotograma.
- El sistema debe mostrar mensajes de error claros en caso de fallos al abrir la cámara, archivos no soportados, o problemas al procesar el video.

## No Funcionales

- Las contraseñas de los usuarios deben estar cifradas para garantizar la seguridad de las credenciales.
- El sistema debe ser capaz de integrarse con diferentes tipos de cámaras y resoluciones sin necesidad de reescribir el código base.
- La interfaz debe ser intuitiva, permitiendo a usuarios no técnicos manejar el sistema con facilidad.
- Debe funcionar en diferentes versiones de Python (preferentemente 3.8+) y ser compatible con sistemas operativos como Windows, Linux, y macOS.

## Reglas de Negocio

- Los usuarios deben registrarse proporcionando un nombre de usuario único y una contraseña válida.
- Las contraseñas deben ser cifradas y almacenadas de forma segura.
- Si un usuario intenta registrar un nombre de usuario ya existente, el sistema debe rechazar el registro e informar al usuario que el nombre ya está en uso.
- Solo los usuarios con credenciales válidas pueden acceder a la funcionalidad principal de la aplicación. El sistema debe validar el nombre de usuario y la contraseña, y restringir el acceso si no se cumple esta regla.
- El sistema debe priorizar la detección de elementos de seguridad (ej. cascos y guantes) en cada fotograma.
- El sistema no debe procesar archivos que no sean imágenes o videos con los formatos especificados (.jpg, .jpeg, .png, .mp4, .avi).

# Caso de Uso

**Código y Nombre del CASO DE USO:** CURS01 - Detección de PPI en imagen estática/video.

## Dimensiones de clasificación:

Nivel	Estructura	Alcance	Caja	Instanciación	Interacción
Resumen	Sin estructurar	Sistema	Negra	Real	Semántico

**Meta del CASO DE USO:** Detectar los elementos de seguridad.

## ACTORES

Primario: Usuario

**PRECONDICIONES (de sistema):** El modelo YOLO está cargado correctamente. El usuario inició sesión.

**DISPARADOR:** El usuario selecciona la opción de subir un archivo para la detección.

## CAMINO BÁSICO:

1. El usuario selecciona la opción de subir un archivo para la detección. El sistema abre una nueva ventana permitiendo seleccionar un archivo.
2. El usuario elige una imagen/vídeo. El sistema recibe el archivo y comienza a procesarlo.
3. Una vez completada la detección, el sistema muestra los resultados de la misma.

## CAMINOS ALTERNATIVOS:

**2.<Posterior>** El formato del archivo no está soportado:

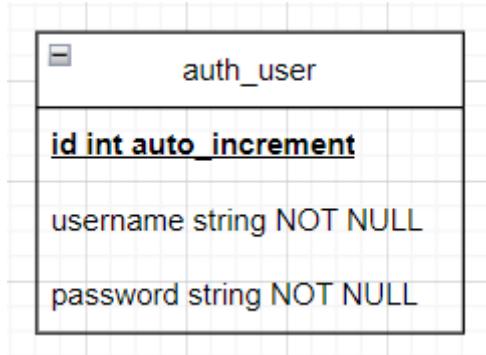
**2.a.1** El sistema informa que el formato seleccionado no es válido. Vuelve al paso 1.

## POSTCONDICIONES (de sistema):

**Éxito:** Se procesaron las imágenes y se mostraron los resultados.

**Fracaso:** Las imágenes no pudieron ser procesadas.

# Modelo de dominio



## Imágenes

### Resultados entrenamiento del modelo

Es necesario conocer los conceptos de precisión y recall para entender las siguientes curvas, ya que son conceptos bastante conocidos en el mundo del machine-learning y la inteligencia artificial. Estos conceptos nacen de una serie de indicadores que contabilizan si las predicciones de un clasificador binario (positivo o negativo) se han realizado correctamente:

- **TP (True positives)**: ejemplos positivos clasificados correctamente.
- **FP (False positives)**: ejemplos negativos clasificados incorrectamente como positivos.
- **TN (True negatives)**: ejemplos negativos clasificados correctamente.
- **FN (False negatives)**: ejemplos positivos clasificados incorrectamente como positivos.

Con estas cuatro métricas, podemos calcular los ratios de precisión y recall.

- ❖ **Precisión:** la precisión es el ratio o porcentaje de clasificaciones correctas de nuestro clasificador. En otras palabras, de todo lo que nuestro clasificador clasifica como positivo, correcta o incorrectamente (TP + FP), cual es el ratio de clasificaciones correctas.

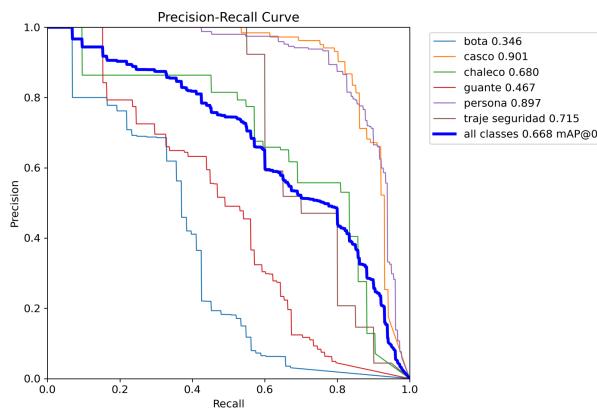
$$precision = \frac{TP}{TP + FP}$$

- ❖ **Recall:** el recall o sensibilidad de nuestro modelo es el ratio de positivos detectado en el dataset por nuestro clasificador. En otras palabras, de todos los positivos reales de nuestro dataset, detectados o no (TP + FN), cual es el ratio de positivos detectados.

$$recall = \frac{TP}{TP + FN}$$

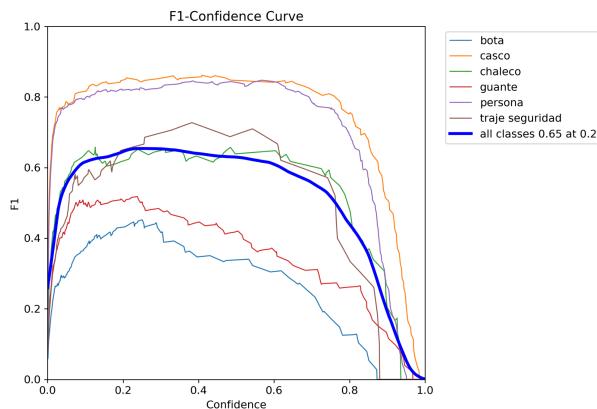
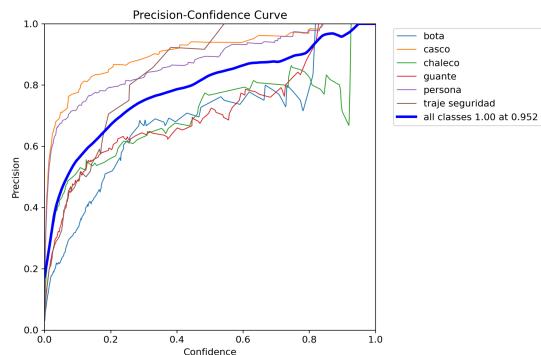
Con estos conceptos en mente, podemos analizar y entender mejor las curvas que se mostrarán a continuación. Estas curvas fueron obtenidas a partir del entrenamiento de nuestro dataset utilizando YOLO.

### Precision-Recall Curve

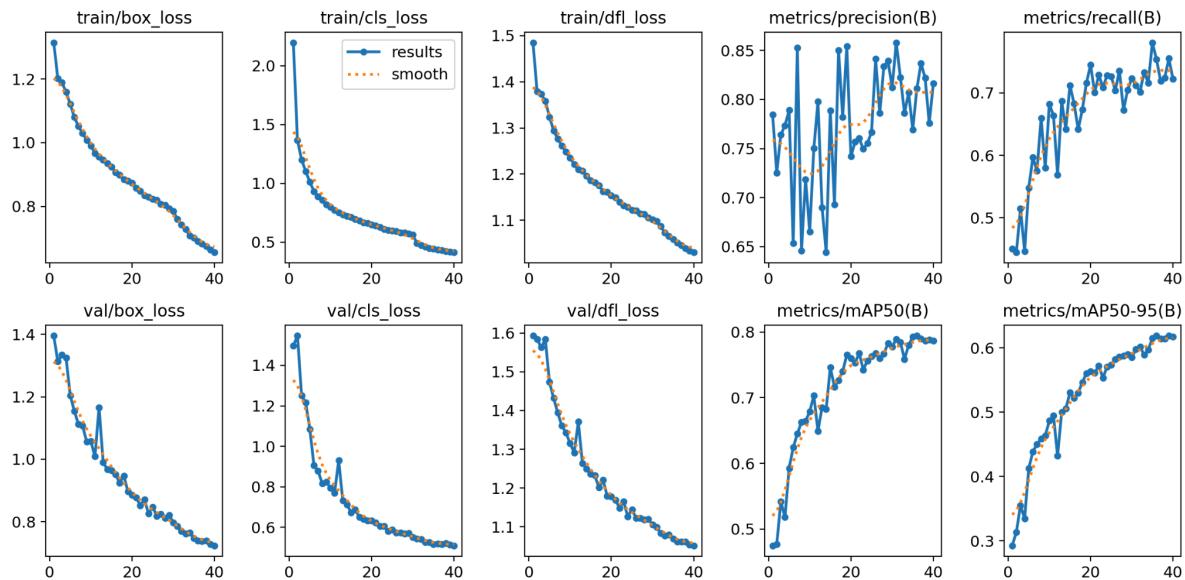
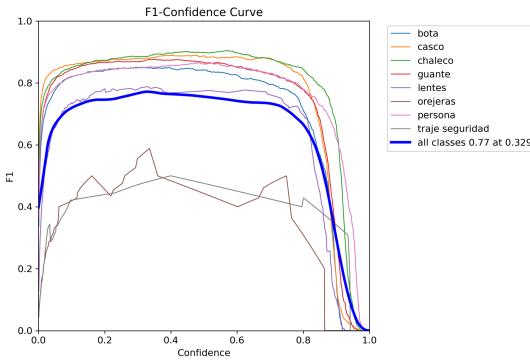


Esta curva evalúa cómo varía la precisión en función del umbral de confianza seleccionado. A medida que aumenta el umbral de confianza, el modelo hace predicciones más seguras, pero podría reducir el número total de objetos detectados, lo que afecta el balance entre precisión y recall.

La curva PR es el resultado de dibujar la gráfica entre la precisión y el recall. Esta gráfica nos permite ver a partir de qué recall tenemos una degradación de la precisión y viceversa. Lo ideal sería una curva que se acerque lo máximo posible a la esquina superior derecha (alta precisión y alto recall). En nuestro caso, el promedio es de 0.668, cuanto más se acerque su valor a 1, mejor será nuestro modelo.



La curva de F1 muestra cómo varía la métrica F1 (una combinación armónica de precisión y recall) según el umbral de confianza. El F1 es útil cuando se busca un equilibrio entre precisión y recall. Un valor de F1 alto implica que el modelo logra una buena detección con un equilibrio entre los dos.



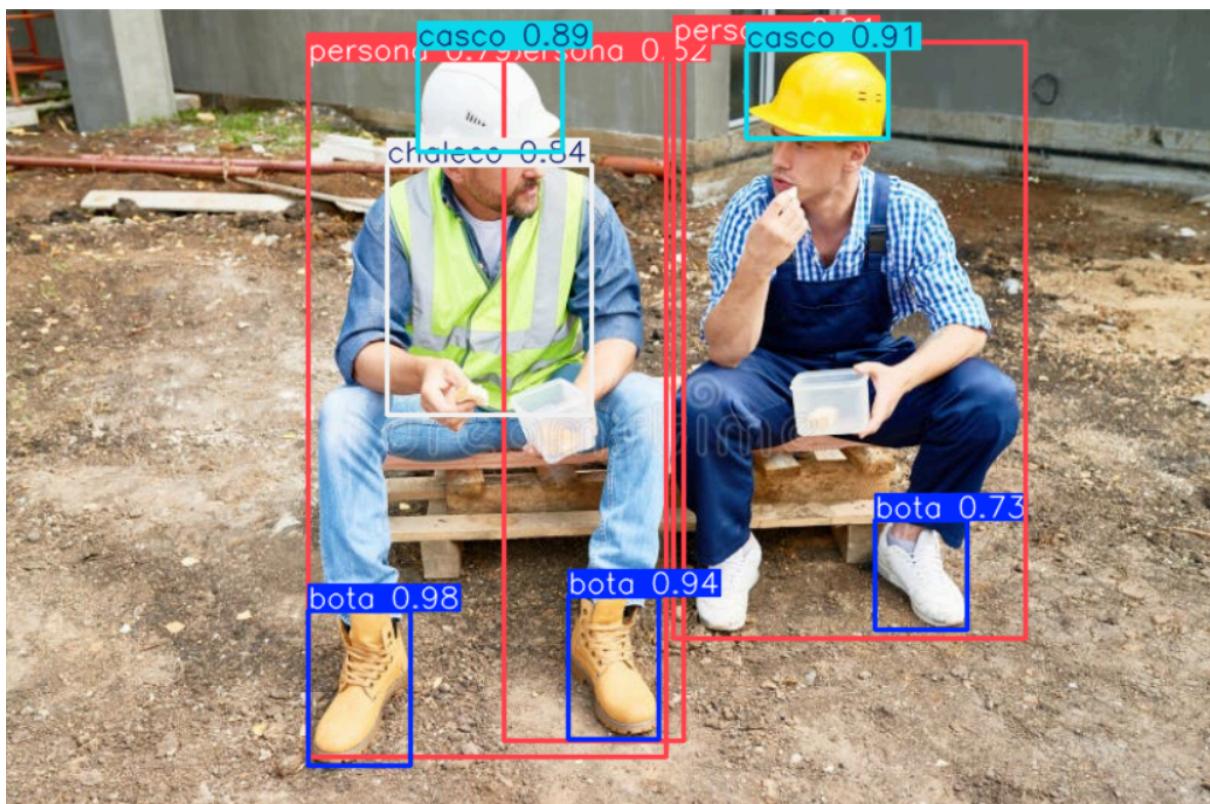
- **Train/Box Loss:** Esta métrica indica el error en la predicción de las cajas delimitadoras durante el entrenamiento. Un valor menor significa que el modelo está mejorando en la precisión espacial de las cajas predichas, ajustándose mejor a los objetos.
- **Train/Cls Loss:** La pérdida de clasificación durante el entrenamiento refleja el error en la identificación de las clases de los objetos detectados. Un valor menor indica que el modelo está clasificando los objetos con mayor precisión.
- **Train/DFL Loss:** La "Distribution Focal Loss" (DFL) mejora la precisión de las cajas delimitadoras mediante una refinación de la distribución de las predicciones de coordenadas. Una pérdida más baja significa que el modelo está ajustando mejor las predicciones de las cajas a los objetos.
- **Metrics/Precision (B):** La precisión en la validación mide cuántas de las predicciones del modelo son correctas en comparación con todas las predicciones positivas. Un valor alto indica que el modelo tiene pocos falsos positivos.
- **Metrics/Recall (B):** El recall durante la validación refleja cuántos de los objetos verdaderos fueron correctamente detectados por el modelo. Un valor alto indica que el modelo está identificando la mayoría de los objetos presentes.
- **Val/Box Loss:** Esta métrica es similar al train/box\_loss, pero calculada durante la validación. Un valor bajo muestra que el modelo predice correctamente las posiciones de las cajas para datos no vistos.

- **Val/Cls Loss:** La pérdida de clasificación en la validación refleja la capacidad del modelo para identificar correctamente las clases en nuevos datos. Una pérdida baja indica que el modelo generaliza bien en sus predicciones de clase.
- **Val/DFL Loss:** Similar a la pérdida DFL en el entrenamiento, pero aplicada a los datos de validación. Un valor más bajo indica que el modelo mejora en ajustar las cajas a objetos en datos no vistos.
- **Metrics/mAP50 (B):** El "Mean Average Precision" a un umbral de IoU del 50% mide la precisión promedio del modelo para detectar objetos correctamente en la validación. Un valor alto muestra que el modelo tiene buen rendimiento en la localización de objetos.
- **Metrics/mAP50-95 (B):** Esta métrica es una versión más rigurosa del mAP que utiliza umbrales de IoU que varían entre 50% y 95%. Un valor alto refleja la capacidad del modelo para localizar objetos con precisión en una amplia gama de umbrales.

Básicamente, con estos gráficos podemos ver cómo cambian las diferentes pérdidas durante el entrenamiento, y cómo se comportan en el conjunto de validación después de cada “epoch”.

## Funcionamiento detección





# Stack Tecnológico

## Lista completa de librerías:

certifi	2024.8.30	matplotlib	3.9.2	pyxnat	1.6.2
cffi	1.17.1	mpmath	1.3.0	PyYAML	6.0.2
charset-normalizer	3.3.2	networkx	3.2.1	rdflib	7.0.0
ci-info	0.3.0	nibabel	5.2.1	requests	2.32.3
click	8.1.7	nipype	1.8.6	scipy	1.13.1
colorama	0.4.6	numpy	1.26.4	seaborn	0.13.2
configobj	5.0.9	opencv-python		setuptools	49.2.1
configparser	7.1.0	4.10.0.84		simplejson	3.19.3
contourpy	1.3.0	packaging	24.1	six	1.16.0
cryptography	43.0.1	pandas	2.2.3	SQLAlchemy	2.0.35
cycler	0.12.1	pathlib	1.0.1	sympy	1.13.3
etellemetry	0.3.1	pillow	10.4.0	tkfontawesome	0.2.0
filelock	3.16.1	pip	20.2.3	tksvg	0.7.4
fonttools	4.54.0	prov	2.0.1	torch	2.4.1
fsspec	2024.9.0	psutil	6.0.0	torchvision	0.19.1
greenlet	3.1.1	py-cpuinfo	9.0.0	tqdm	4.66.5
httplib2	0.22.0	pycparser	2.22	traits	6.3.2
idna	3.10	pydot	3.0.1	typing-extensions	4.12.2
importlib-resources	6.4.5	PyMuPDF	1.24.10	tzdata	2024.2
isodate	0.6.1	PyMuPDFb		ultralytics	8.2.100
jinja2	3.1.4	1.24.10		ultralytics-thop	2.0.8
kiwisolver	1.4.7	pyparsing	3.1.4	urllib3	2.2.3
looseversion	1.3.0	python-dateutil		zipp	3.20.2
lxml	4.7.1	2.9.0.post0			
MarkupSafe	2.1.5	pytz	2024.2		

## Documentación

- <https://docs.python.org/3/library/tkinter.html>
- <https://docs.opencv.org/4.x/index.html>
- <https://docs.ultralytics.com/>

## Código Fuente

<https://github.com/lauratulian/PPE-Detection-Python>