Detección de Equipo de Seguridad con Técnicas Avanzadas de Aprendizaje Profundo

Arturo Cristián Díaz López
Instituto Tecnológico y de Estudios Superiores de Monterrey

25-Nov-2024

Resumen

Este proyecto desarrolla un modelo de detección de objetos en tiempo real basado en deep learning, diseñado para monitorear el cumplimiento del uso de equipo de seguridad en entornos laborales. El modelo es capaz de distinguir entre la presencia y la ausencia de chalecos de seguridad en imágenes capturadas en ambientes de trabajo, facilitando una supervisión constante y automatizada de la seguridad en áreas de producción. La intención de esta investigación es crear una solución fácilmente integrable en sistemas de monitoreo CCTV, con el objetivo de asegurar que el personal cumpla con las políticas de seguridad establecidas. Al implementar esta solución, se espera contribuir a la reducción de riesgos laborales y mejorar el cumplimiento de normas de seguridad en tiempo real.

1 Introduction

En muchos sectores industriales y de construcción, el uso adecuado de equipo de seguridad es fundamental para minimizar riesgos laborales y proteger la integridad física de los trabajadores. Sin embargo, la supervisión manual del cumplimiento de las normas de seguridad, como el uso de chalecos de protección, suele ser un proceso costoso y propenso a errores, especialmente en entornos grandes o de alta actividad. La falta de cumplimiento en el uso de equipo de seguridad representa una amenaza para la salud y seguridad de los empleados y, a su vez, puede generar repercusiones legales y financieras para las organizaciones.

El avance en el aprendizaje profundo ha abierto nuevas oportunidades para abordar problemas de supervisión mediante la automatización. En particular, el uso de modelos de deep learning permite analizar imágenes de video en tiempo real, lo que hace posible detectar en segundos si un trabajador porta el equipo de seguridad requerido. Este proyecto se centra en el desarrollo de un modelo de detección de objetos entrenado para identificar el uso de chalecos de seguridad en ambientes laborales. Esta solución tiene el potencial de integrarse fácilmente en sistemas de cámaras de seguridad (CCTV), ofreciendo un monitoreo constante y preciso sin necesidad de intervención humana. Al implementar esta tecnología, las empresas pueden reducir significativamente los riesgos laborales y mejorar el cumplimiento de normas de seguridad, contribuyendo a un entorno de trabajo más seguro y eficiente.

Para resolver esta problemática, el proyecto implementa una solución basada en el reconocimiento de bounding boxes, lo cual permite identificar y localizar objetos dentro de una imagen. Esta tecnología permite que el modelo no solo clasifique si un chaleco de seguridad está presente, sino también que determine la ubicación del objeto en tiempo real. Esto es especialmente útil en entornos industriales o de construcción, donde la supervisión de la seguridad en todas las áreas es fundamental.

2 Dataset

La calidad y estructura de un dataset son elementos esenciales para el entrenamiento efectivo de un modelo de aprendizaje profundo. En proyectos de visión por computadora, contar con un conjunto de datos bien etiquetado y representativo garantiza que el modelo aprenda a identificar patrones y características relevantes en las imágenes. En este proyecto, el objetivo es detectar la presencia o ausencia de chalecos de seguridad en entornos laborales, por lo que se utilizó un dataset con imágenes donde pueden apreciarse tanto trabajadores portando chalecos de seguridad como aquellos que no cumplen con esta norma. Este dataset contiene una diversidad de imágenes tomadas en distintos escenarios laborales, como sitios de construcción y entornos industriales, lo cual permite entrenar un modelo robusto capaz de identificar correctamente la presencia del equipo de seguridad en condiciones visuales variadas.

2.1 Fuente

La fuente del set de datos utilizado en este proyecto es Roboflow, una plataforma en línea que simplifica la creación y gestión de datasets para proyectos de visión por computadora. Roboflow ofrece una amplia gama de datasets y modelos pre-entrenados de código abierto, lo que permite a los desarrolladores acceder a recursos valiosos para sus proyectos. El dataset seleccionado puede ser consultado en el enlace `https://universe.roboflow.com/roboflow-universe-projects/safety-vests/dataset/7`.

Figura 1: *Ejemplo de imagen proveniente del set de datos.*

2.2 Estructura

El dataset utilizado para el desarrollo del modelo contiene lo siguiente:

- Un total de 3,900 imágenes con resoluciones de entre 600 y 300 píxeles, representando una amplia variedad de entornos laborales, condiciones de iluminación y ángulos de cámara.
- 3,900 archivos de texto que contienen las anotaciones, cada uno describiendo las coordenadas precisas de los bounding boxes que indican la presencia o ausencia de chalecos de seguridad en las imágenes.

Estas anotaciones proporcionan información esencial para el modelo, permitiéndole aprender a detectar y localizar el equipo de seguridad en diversas condiciones. La anotación adecuada de cada imagen asegura que el modelo pueda diferenciar entre trabajadores que portan el chaleco de seguridad y aquellos que no, generando una herramienta confiable para su posterior implementación en sistemas de monitoreo en tiempo real.

Este dataset fue dividido en conjuntos de entrenamiento, validación y prueba, con un balance entre las clases para evitar sesgos en la predicción. Las imágenes se preprocesaron y redimensionaron a 640x640 píxeles para estandarizar la entrada del modelo y optimizar su rendimiento.

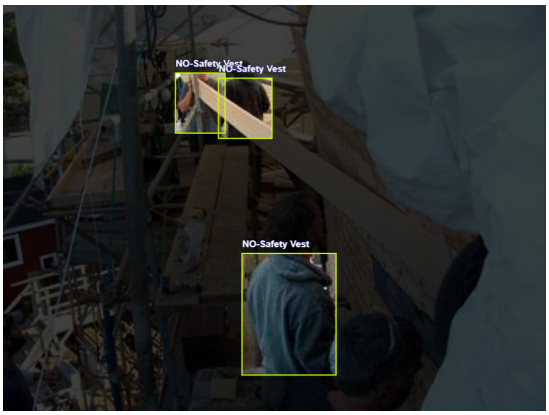

Figura 2: *Imagen con anotaciones usada en el set de entrenamiento.*

3 Modelo

El modelo desarrollado para este proyecto se basa en técnicas avanzadas de aprendizaje profundo, específicamente en el ámbito de la detección de objetos. La detección de objetos permite identificar y localizar instancias de interés dentro de una imagen, en este caso, la presencia de chalecos de seguridad en entornos laborales.

3.1 Arquitectura

YOLO

YOLO, que significa "You Only Look Once", es una arquitectura revolucionaria en la detección de objetos que ha transformado el campo de la visión por computadora. Su diseño se basa en la idea de realizar la detección en una única pasada a través de la red, lo que contrasta con métodos más tradicionales que operan en múltiples etapas. Esta característica permite que YOLO produzca resultados de detección de objetos en tiempo real, lo que es esencial para aplicaciones donde la velocidad es crítica, como en la vigilancia y el monitoreo de seguridad.

La arquitectura original de YOLO introduce un enfoque que divide la imagen en una cuadrícula, donde cada celda de la cuadrícula es responsable de predecir las bounding boxes y las clases de los

objetos que caen dentro de esa celda. Esto no solo optimiza el proceso de detección, sino que también permite que el modelo considere el contexto global de la imagen en lugar de analizar pequeñas regiones de manera aislada. Esta capacidad de ver la imagen en su totalidad le permite a YOLO realizar detecciones más coherentes y precisas. Con la evolución de la tecnología y la demanda de mejores rendimientos, se han desarrollado varias versiones de YOLO, siendo YOLOv5 una de las más destacadas. YOLOv5 es una implementación de YOLO que ha sido optimizada para mejorar tanto la velocidad como la precisión en la detección de objetos. Utiliza una serie de innovaciones en su arquitectura, incluyendo la incorporación de capas adicionales y la mejora de los algoritmos de post-procesamiento.

La implementación de YOLOv5 en este proyecto se centra en la detección de chalecos de seguridad en imágenes de entornos laborales. Al aprovechar su rapidez y precisión, el modelo se convierte en una herramienta valiosa para las empresas que buscan asegurar el cumplimiento de las normas de seguridad y contribuir a un entorno de trabajo más seguro. La capacidad de integrar YOLOv5 en sistemas de monitoreo CCTV permite un seguimiento constante y automatizado del uso del equipo de seguridad, lo que facilita la identificación de riesgos y mejora la seguridad general del lugar de trabajo.

¿Cómo funciona?

La arquitectura de YOLOv5 se basa en la idea de realizar detección de objetos en una sola pasada a través de la red neuronal. YOLOv5 utiliza una estructura de red neuronal que combina varias capas convolucionales para extraer características relevantes de las imágenes de entrada. La red se compone de tres etapas principales: la extracción de características, la detección de objetos y la post-procesamiento de las predicciones.

1. **Backbone** (Extracción de características): La primera etapa de YOLOv5 implica la utilización de una red de extracción de características, que se basa en un diseño eficiente que permite reducir el tamaño de la imagen a medida que se extraen características más complejas.
2. **Neck** (Detección de objetos): Después de la extracción de características, YOLOv5 divide la imagen en una cuadrícula de celdas, donde cada celda es responsable de detectar objetos en su área correspondiente. Para cada celda, el modelo predice múltiples bounding boxes y las probabilidades asociadas a cada clase. Esto incluye la posición y el tamaño de la bounding box, así como una puntuación de confianza que indica la probabilidad de que un objeto esté presente en esa celda.
3. **Head** (Predicciones y Post-Procesamiento): En esta parte, se realizan las predicciones finales. Aquí, se generan los bounding boxes y las probabilidades de clase para los objetos detectados.

¿Por qué YOLO?

YOLO es conocido por su velocidad y eficiencia, lo que permite realizar detecciones en tiempo real; esta característica es crucial para la supervisión constante o fotograma a fotograma. A pesar de su rapidez, el modelo no sacrifica precisión en la identificación de objetos, adaptándose con efectividad a diversas escalas y posiciones. La arquitectura de una sola etapa de YOLO simplifica su implementación, facilitando la integración en sistemas existentes, como cámaras de vigilancia CCTV. Además, el modelo es robusto ante condiciones de iluminación variables y escenarios complejos, garantizando resultados confiables en una amplia gama de situaciones.

3.2 Entrenamiento

El proceso de entrenamiento del modelo se llevó a cabo utilizando el repositorio de YOLOv5 de Ultralytics (`https://github.com/ultralytics/yolov5`), que proporciona una implementación optimizada y fácil de usar de la arquitectura YOLO. Para comenzar, se configuró el entorno de desarrollo, asegurando que todas las dependencias necesarias estuvieran instaladas, lo que incluyó librerías como PyTorch y OpenCV. En particular, se configuró el entorno para aprovechar la aceleración de hardware a través de GPU, utilizando CUDA para maximizar la eficiencia y velocidad del entrenamiento. Esto permitió manejar de manera más efectiva los cálculos intensivos requeridos por los modelos de deep learning. Una vez listo el entorno, se preparó el dataset, asegurando que las imágenes y sus correspondientes anotaciones en formato YOLO estuvieran correctamente organizadas en sus respectivas carpetas de entrenamiento, pruebas y validación.

El comando utilizado para entrenar el modelo:

```
python train.py --img 640 --batch 4
--epochs 50 --data ../data.yaml
--weights yolov5s.pt --device 0
```

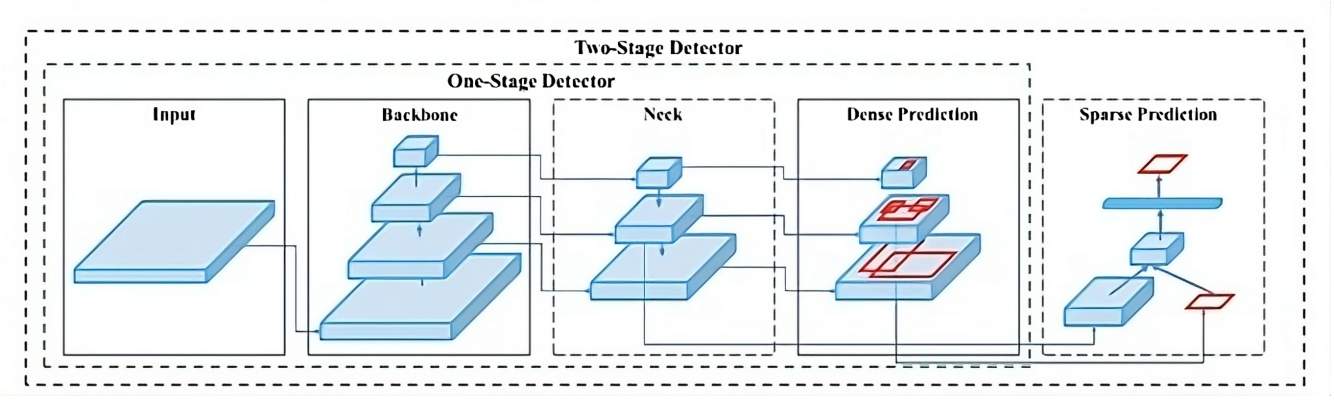

Figura 3: *Arquitectura del proceso de detección de objetos.*

En este comando, el parámetro –img 640 especifica que todas las imágenes se redimensionarán a un tamaño de 640x640 píxeles antes de ser alimentadas al modelo. Este tamaño es óptimo para el equilibrio entre precisión y velocidad, permitiendo al modelo procesar las imágenes de manera eficiente. El parámetro –batch 4 establece el tamaño del lote a 4, lo que significa que el modelo procesará cuatro imágenes a la vez antes de actualizar los pesos, un enfoque que puede ayudar a gestionar la memoria durante el entrenamiento, especialmente cuando se utiliza una GPU. El número de épocas se define con –epochs 50, lo que indica que el modelo se entrenará durante 50 ciclos completos a través del conjunto de datos, permitiendo suficiente tiempo para que el modelo aprenda las características relevantes de las clases objetivo. El parámetro –data ../data.yaml especifica la ubicación del archivo YAML que contiene la configuración del conjunto de datos, incluyendo las rutas de las imágenes y las anotaciones. Además, el uso de –weights yolov5s.pt permite comenzar el entrenamiento con un modelo pre-entrenado, lo que facilita una mejor convergencia al aprovechar el conocimiento previo adquirido en un conjunto de datos amplio. Finalmente, –device 0 indica que el entrenamiento debe realizarse en la primera GPU disponible.

Durante el proceso de entrenamiento, se monitorearon métricas clave, como la pérdida del modelo y la precisión en el conjunto de validación, lo que proporcionó información valiosa sobre el desempeño del modelo a medida que se entrenaba. Una vez completado el entrenamiento, se generaron modelos finales, incluidos el mejor modelo basado en la métrica de precisión y el modelo final entrenado.

Hiperparámetros

Los hiperparámetros definidos para realizar el entrenamiento fueron:

- lr0: 0.01
- lrf: 0.01
- momentum: 0.937
- weight_decay: 0.0005
- warmup_epochs: 3
- warmup_momentum: 0.8

Algunos otros fueron el umbral de Intersección sobre la Unión (IoU), indicado por el parámetro `iou_t`, establece un valor mínimo que determina cuándo una detección se considera correcta en función del grado de superposición con el objeto real. Un umbral bajo aumenta el número de detecciones, mientras que un valor más alto hace que el modelo sea más riguroso. El parámetro `fliplr` controla la probabilidad de voltear las imágenes horizontalmente durante la augmentación de datos, configurado aquí en 0.5 para que la mitad de las imágenes se voltee aleatoriamente. Por otro lado, la técnica `mosaic` de augmentación permite combinar varias imágenes en una sola, diversificando las variaciones en las escenas de entrenamiento y contribuyendo así a una mejora en la generalización del modelo. Finalmente, se emplearon aumentaciones adicionales mediante la biblioteca `albumentations`, aplicando de forma aleatoria desenfoques, conversiones a escala de grises, y ajustes de contraste, entre otros, para incrementar la robustez del modelo frente a distintas condiciones visuales en las imágenes. Para el proceso de optimización del modelo, se utilizó el optimizador SGD (Stochastic Gradient Descent) con una tasa de aprendizaje inicial (`lr`) de 0.01. El SGD es ampliamente utilizado en entrenamientos de deep learning por su efectividad en la convergencia y control de los gradientes, especialmente en tareas de visión por computadora. Adicionalmente, se configuraron diferentes grupos de parámetros con distintos valores de `weight decay`. En este caso, los

pesos sin decaimiento fueron 57, mientras que 60 pesos aplicaron un `weight decay` de 0.0005.

4 Resultados

4.1 Descripción de las métricas

Para evaluar el desempeño del modelo, se utilizaron métricas ampliamente reconocidas en tareas de detección de objetos, como precisión (*precision*), *recall* y F1-score. Estas métricas permiten medir la eficacia del modelo en identificar correctamente la presencia de chalecos de seguridad en imágenes de prueba. En este contexto:

- **Precisión**: Indica la proporción de detecciones correctas (chalecos identificados) frente al total de detecciones realizadas. Es importante para evitar falsas alarmas.
- **Recall**: Mide la capacidad del modelo para encontrar todos los objetos de interés (chalecos de seguridad) presentes en las imágenes. Es crucial en aplicaciones donde es importante minimizar el riesgo de omitir elementos relevantes para la seguridad.
- **mAP@0.5**: Esta métrica, conocida como *mean Average Precision* al 50 % de *IoU*, evalúa la precisión promedio del modelo considerando que las predicciones correctas deben alcanzar al menos un 50 % de solapamiento (IoU) con los objetos reales. Es una métrica comúnmente utilizada para evaluar la precisión en tareas de detección de objetos.
- **mAP@0.5:0.95**: Similar al mAP@0.5, esta métrica evalúa la precisión promedio del modelo, pero calcula el promedio de IoUs desde el 50 % hasta el 95 % en intervalos del 5 %. Este rango permite obtener una medida más exigente del desempeño del modelo, ya que evalúa su precisión en una gama más amplia de solapamientos posibles, proporcionando una visión más completa de la calidad de las predicciones.

4.2 Resultados cuantitativos

Los resultados obtenidos durante el entrenamiento y evaluación del modelo se resumen en la siguiente tabla. El modelo se evaluó en un conjunto de datos de prueba, reportando las métricas de precisión, *recall* y F1-score.

- **Precisión**: 0.8816
- **Recall**: 0.8392
- **mAP@0.5**: 0.9068
- **mAP@0.5:0.95**: 0.5634

4.3 Visualización de detecciones

Para ilustrar los resultados del modelo, se desarrolló un script en Python que carga el modelo entrenado utilizando PyTorch. Este script permite realizar predicciones sobre un conjunto de imágenes de prueba, evaluando así la capacidad del modelo para identificar la presencia de chalecos de seguridad en entornos laborales. Al ejecutar el script, se procesan las imágenes y se generan visualizaciones que incluyen las predicciones del modelo, mostrando los cuadros delimitadores (bounding boxes) alrededor de los chalecos detectados. Además, se superponen etiquetas que indican la clase predicha, así como su valor de confianza en predicción del 0 al 1. Esta representación gráfica no solo permite verificar la efectividad del modelo, sino que también proporciona una comprensión más clara de cómo se comporta en situaciones del mundo real.

Figura 4: *Imagen con anotaciones predichas por el modelo entrenado.*

5 Tuning

El proceso de refinamiento del modelo se dividió en dos iteraciones principales, donde cada iteración implementó ajustes específicos en los hiperparámetros, arquitectura y datos de entrenamiento con el objetivo de mejorar el desempeño general del modelo.

5.1 Iteraciones

A continuación, se describen las características principales de cada etapa en los modelos:

Modelo 1: Vainilla y entrenamiento default

Inicialmente, se realizó un entrenamiento *vainilla* como primera iteración y se utilizaró la arquitectura *YOLOv5s* con hiperparámetros predeterminados para establecer una línea base de desempeño. Este modelo fue entrenado utilizando 3,505 imágenes del dataset original bajo los siguientes parámetros:

- **Arquitectura:** *YOLOv5s* (small).
- **Hiperparámetros:** Predeterminados en la configuración de YOLOv5.
- **Entrenamiento:** 50 épocas con un tamaño de lote de 4 imágenes.

Modelo 2: Segunda iteración

Durante esta iteración, se buscó realizar una mejora al modelo inicial, por lo que se realizó un refinamiento a algunos hiperparámetros y se ajustaron los pesos iniciales en el entrenamiento del modelo.

- **Arquitectura:** *YOLOv5l* (large).
- **Hiperparámetros customizados:**
    1. **lr0:** Se disminuyó el *learning rate* inicial a 0.005 para facilitar una convergencia más estable.
    2. **lrf:** Se incrementó el *learning rate* final a 0.01 para lograr ajustes más precisos en las últimas etapas del entrenamiento.
    3. **iou_t:** Se aumentó el umbral de detección a 0.3 para hacer las predicciones más estrictas.
- **Entrenamiento:** 50 épocas con un tamaño de lote de 4 imágenes.

Modelo 3: Tercera iteración

Finalmente, se buscó realizar una mejora adicional al modelo de la segunda iteración. Para lograrlo, se integró un segundo dataset que introdujo aproximadamente 1,200 imágenes completamente nuevas, con el objetivo de ampliar la capacidad del modelo para generalizar mejor frente a las condiciones de una imagen real de CCTV.

El entrenamiento de esta iteración utilizó un enfoque de *transfer learning*, aprovechando los pesos refinados generados en el modelo de la segunda iteración. Además, para acelerar el proceso de entrenamiento y preservar los conocimientos previamente adquiridos, se congelaron las primeras 10 capas del modelo (el *backbone*). Esto permitió concentrar la optimización únicamente en las capas superiores, responsables de las tareas de detección específicas.

- **Arquitectura:** *YOLOv5l* (large).
- **Pesos iniciales:** *best.pt* (Mejores pesos del entrenamiento del segundo modelo).
- **Dataset:** Combinación de 3,505 imágenes originales y 1,200 imágenes nuevas.
- **Hiperparámetros:** Predeterminados.
- **Entrenamiento:** 25 épocas con un tamaño de lote de 4 imágenes.

5.2 Resultados

Los resultados y métricas obtenidas en validación por los tres modelos se pueden apreciar en la siguiente tabla.

	Modelo 1	**Modelo 2**	**Modelo 3**
Arquitectura	YOLOv5s	YOLOv5l	YOLOv5l
Pesos iniciales	COCO	COCO	best.pt
Épocas	50	50	25
Conjunto de datos (número de imágenes)	3,505	3,505	4,705
Capas congeladas	-	-	Backbone
Box Loss	0.0249	0.0233	0.0222
Object Loss	0.0145	0.0149	0.0161
Precisión	88.16 %	90.37 %	86.94 %
Recall	83.92 %	88.09 %	83.78 %
mAP@0.5	90.68 %	92.41 %	89.24 %
mAP@0.5:0.95	56.34 %	59.17 %	58.32 %

Cuadro 1: *Comparación de métricas y resultados entre las iteraciones del modelo.*

Tercer Modelo

Las métricas del tercer modelo indican un posible caso de sobreajuste. Aunque se buscaba mejorar la generalización al integrar un segundo conjunto de datos que incrementó el tamaño total del dataset a 4,705 imágenes, los resultados muestran una discrepancia entre las pérdidas de entrenamiento y las métricas de validación.

Por ejemplo, la pérdida en validación para el ajuste de objetos (box_loss), inicia con un crecimiento durante aproximadamente 3 épocas, y después muestra una tendencia a la baja. Esto indica que el modelo generaliza con éxito las medidas esperadas de un objeto o caja. La gráfica siguiente muestra el detalle de este valor a lo largo del entrenamiento.

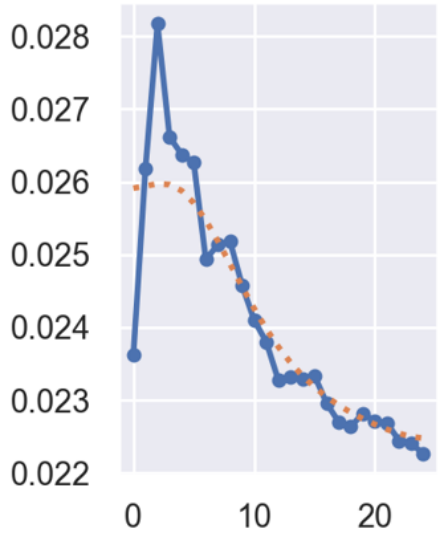

Figura 5: *Pérdida en validación para el ajuste de cajas (box_loss).*

Por otra parte, la pérdida en validación para la detección de objetos (obj_loss), muestra una tendencia creciente hacia el final del entrenamiento, lo que sugiere que el modelo pierde capacidad de generalización en la tarea de detectar la presencia de objetos. Este comportamiento se puede observar en la siguiente figura.

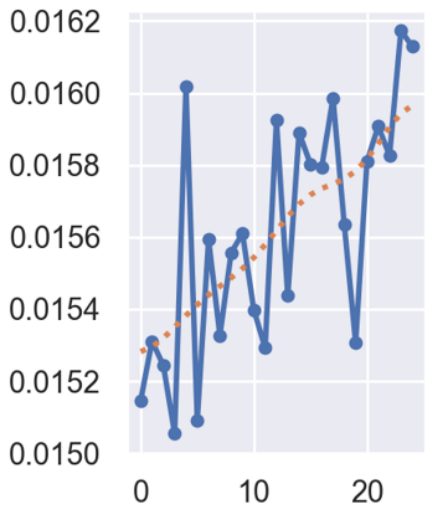

Figura 6: *Pérdida en validación para la detección de objetos (obj_loss).*

Finalmente, las métricas de evaluación también evidencian este sobreajuste. La precisión y el recall muestran fluctuaciones significativas sin una clara tendencia de mejora.

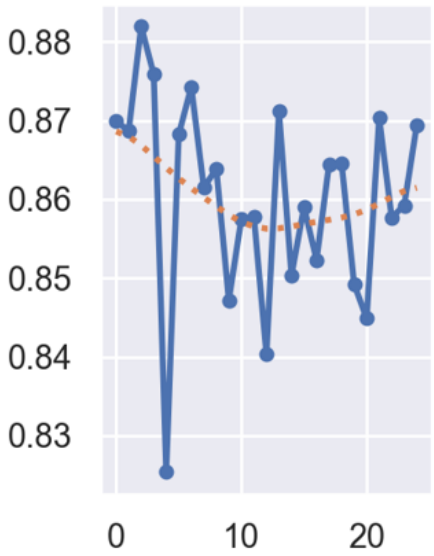

Figura 7: *Precisión del modelo a lo largo del entrenamiento.*

Particularmente, el recall presenta una tendencia descendente a lo largo del entrenamiento, lo que indica que el modelo gradualmente pierde capacidad para detectar todos los objetos relevantes en las imágenes de validación.

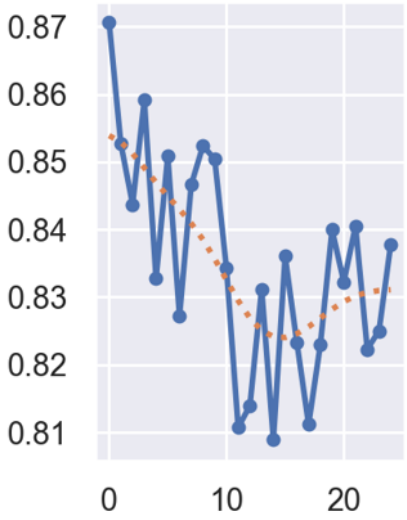

Figura 8: *Recall del modelo a lo largo del entrenamiento.*

Para mitigar este sobreajuste, se podrían considerar las siguientes estrategias en futuras iteraciones del modelo:

- Implementar técnicas adicionales de regularización como dropout o weight decay más agresivo.
- Ajustar la arquitectura del modelo para reducir su capacidad y evitar el sobreajuste

- Introducir early stopping basado en las métricas de validación
- Explorar técnicas de balanceo de clases si existe desbalance en el conjunto de datos

Segundo Modelo

El segundo modelo muestra una mejora significativa en el rendimiento general, evidenciado tanto por las métricas de pérdida como por los indicadores de precisión. Con la implementación de la arquitectura YOLOv5l, se logró una mejor capacidad de generalización y estabilidad durante el entrenamiento respecto al modelo vainilla.

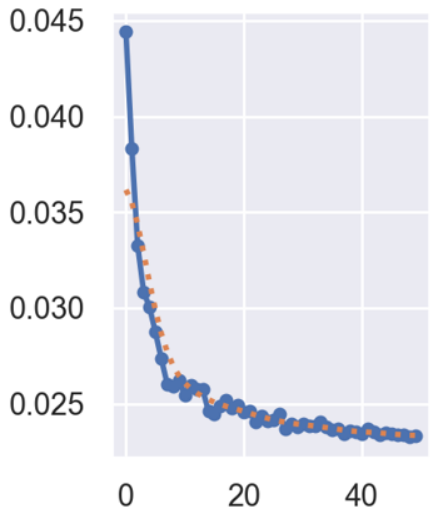

Figura 9: *Pérdida en validación para el ajuste de cajas (box_loss) en el segundo modelo.*

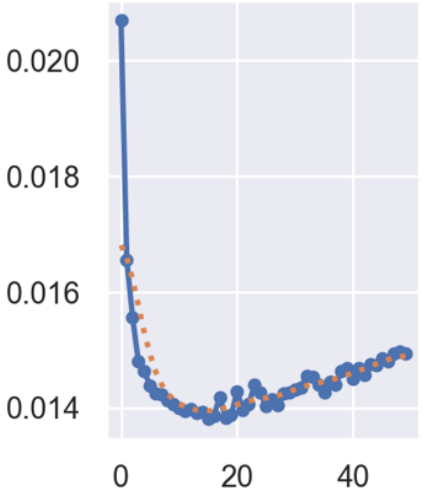

Figura 10: *Pérdida en validación para la detección de objetos (obj_loss) en el segundo modelo.*

La pérdida de validación para la detección de objetos (val/obj_loss) muestra una tendencia decreciente hasta aproximadamente la época 20.

Métricas de evaluación reflejan esta mejora en el rendimiento. La precisión alcanza y mantiene valores superiores al 90 %, con una tendencia ascendente estable.

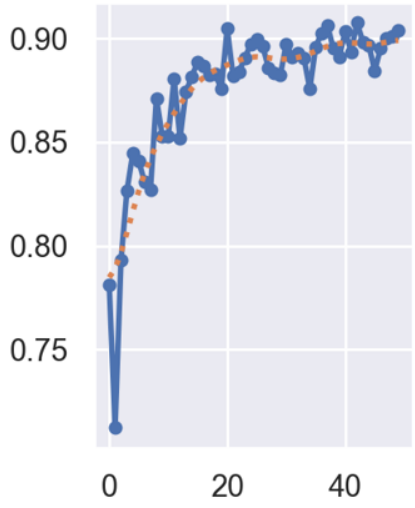

Figura 11: *Precisión del segundo modelo a lo largo del entrenamiento.*

El recall muestra una mejora sustancial, alcanzando un 88.09 %, lo que representa un incremento de más de 4 puntos porcentuales respecto al modelo anterior.

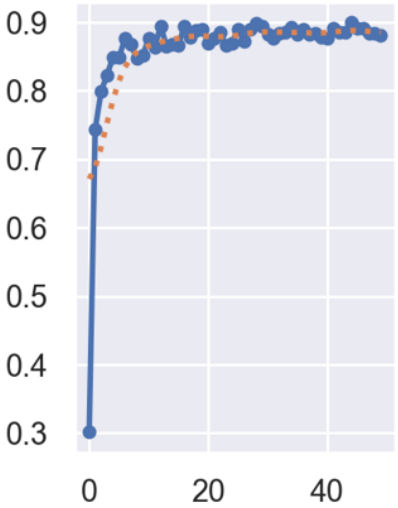

Figura 12: *Recall del segundo modelo a lo largo del entrenamiento.*

Es importante destacar que estas mejoras se obtuvieron manteniendo el mismo conjunto de datos de 3,505 imágenes, lo que sugiere que la arquitectura más robusta de YOLOv5l permitió un mejor aprovechamiento de las características presentes en los datos originales de entrenamiento. La estabilidad en las métricas de validación y la ausencia de oscilaciones pronunciadas indican que el modelo logró un buen balance entre el ajuste a los datos de entrenamiento y la capacidad de generalización.

6 Conclusiones

Esta investigación demuestra que la implementación de un modelo de detección de objetos basado en la arquitectura YOLOv5 puede ser una solución efectiva para supervisar el uso de equipo de seguridad en entornos laborales. Los resultados obtenidos evidencian un rendimiento notable en términos de precisión, recall y F1-score, lo que resalta la viabilidad de esta tecnología para mejorar la seguridad en el trabajo. A medida que las organizaciones continúan buscando maneras de optimizar sus procesos de seguridad, este modelo presenta una opción prometedora para automatizar la supervisión y garantizar el cumplimiento de las normativas de seguridad.

7 Bibliografía

[1] Roboflow. *Dataset: Safety Vests*. https://universe.roboflow.com/roboflow-universe-projects/safety-vests/dataset/7

[2] Roboflow. *Dataset: Check B Computer Vision*. https://universe.roboflow.com/shicong-liu-gvsji/check_b

[3] Ultralytics. *YOLOv5: A State-of-the-Art Object Detection Architecture*. https://github.com/ultralytics/yolov5

[4] Ultralytics. *Ultralytics Train Custom Data*. `https://docs.ultralytics.com/yolov5/tutorials/train_custom_data/`

[5] Ultralytics. *Comprehensive Guide to Ultralytics YOLOv5*. https://docs.ultralytics.com/yolov5/

[6] Ultralytics. *Ultralytics YOLOv5 Architecture*. `https://docs.ultralytics.com/yolov5/tutorials/architecture_description/`

[7] Bochkovskiy, Alexey, Wang, Chien-Yao, and Liao, Hong-Yuan Mark. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. https://arxiv.org/pdf/2004.10934

[8] Roboflow. *What is YOLOv5? A Guide for Beginners*. https://blog.roboflow.com/yolov5-improvements-and-evaluation/

[9] Tsang, Sh. *Reading: C3 — Concentrated-Comprehensive Convolution (Semantic Segmentation)*. https://sh-tsang.medium.com/reading-c3-concentrated-comprehensive-convolution-semantic-segmentation-5b6dd3fb46b2

[10] Tsang, Sh. *Brief Review: YOLOv5 for Object Detection*. https://sh-tsang.medium.com/brief-review-yolov5-for-object-detection-84cc6c6a0e3a

[11] Hui, Jonathan. *mAP (mean Average Precision) for Object Detection*. https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173