

**Universidad Peruana de Ciencias Aplicadas**



**INFORME DE TF**

**CURSO: Procesamiento de Imágenes**

**Carrera de Ciencias de la Computación**

**Sección: 307**

<b>Alumnos</b>	
<b>Código</b>	<b>Nombres y apellidos</b>
U202214090	Francesca Nicole Bances Torres
U202118582	Marcos Aaron Bedia Torres
U201616851	Lizbeth Teresita Olivera Alvarez

**2025-1**

## **CONTENIDO**

<b>CONTENIDO.....</b>	<b>2</b>
<b>1. Descripción del caso de uso:.....</b>	<b>3</b>
<b>2. Descripción del conjunto de datos (dataset):.....</b>	<b>4</b>
<b>3. Propuesta de Modelización:.....</b>	<b>4</b>
Tabla comparativa en nano y small:.....	5
<b>4. Modelización.....</b>	<b>7</b>
<b>5. Publicación de resultados.....</b>	<b>8</b>
<b>6. Conclusiones.....</b>	<b>13</b>
<b>7. Referencias bibliográficas.....</b>	<b>14</b>

# 1. Descripción del caso de uso:

En años recientes, el incremento de la violencia y del crimen organizado en Perú ha sido preocupante. Cada vez se reportan más incidentes de homicidios, extorsiones y crímenes perpetrados con armas, lo que provoca temor en la sociedad peruana. De acuerdo con ComexPerú, el índice de homicidios aumentó en un 74% entre 2019 y 2024, mientras que las extorsiones se incrementaron más del 438% en el mismo intervalo de tiempo. Esto no solo evidencia una crisis de seguridad, sino también un aumento en la desconfianza hacia las autoridades responsables de resguardar la seguridad pública. Ante esta circunstancia, resulta imprescindible buscar alternativas que contribuyan a potenciar la vigilancia y a evitar actos delictivos.

En este contexto, nuestro trabajo busca ofrecer una propuesta tecnológica que ayude a abordar este importante problema. El objetivo es crear un sistema para la identificación automática de armas (como pistolas, cuchillos, rifles o granadas) utilizando técnicas de procesamiento de imágenes y aprendizaje automático. Aunque la idea general contempla la integración en sistemas de videovigilancia, en este proyecto en particular solo se trabajó con imágenes estáticas debido a la disponibilidad y facilidad de anotación de datos.

Este caso de uso no solo se justifica por la necesidad de incrementar la seguridad de los ciudadanos, sino también por el posible efecto beneficioso de las tecnologías de visión computacional en la prevención del crimen. La automatización de la detección de armas en espacios públicos podría facilitar una reacción más ágil de las autoridades, disminuir el tiempo de respuesta frente a circunstancias de riesgo, y funcionar como factor disuasorio para posibles delincuentes.

Asimismo, este proyecto representa una oportunidad educativa para aplicar conocimientos de procesamiento digital de imágenes, detección de objetos, entrenamiento de modelos de aprendizaje y evaluación de desempeño en contextos del mundo real.

## 2. Descripción del conjunto de datos (dataset):

El dataset a utilizar es “Test Computer Vision Project”, disponible en la plataforma Roboflow. Este conjunto de datos fue diseñado para proyectos de visión por computadora, específicamente la detección de armas como granadas, rifles, pistolas y cuchillos.

Dataset: Test Computer Vision Project

Origen: Roboflow Universe

Cantidad total de imágenes: 4666

Cantidad de imágenes por clase:

- Grenade: 1413
- Rifle: 1226
- Pistol: 1010
- Knife: 845

## 3. Propuesta de Modelización:

El objetivo es crear un sistema inteligente capaz de detectar armas (pistolas, rifles, cuchillos, granadas, etc.) en tiempo real a partir de imágenes estáticas. El objetivo es maximizar la precisión y velocidad para alertar rápidamente a las autoridades y prevenir actos delictivos.

### 1. Selección de Modelos

Se evaluaron diferentes versiones de los modelos YOLO, considerando la especialización en detección rápida y precisa, con especial atención en variantes ligeras y de alta precisión para contextos en los que el hardware puede ser limitado.

### 2. ¿Qué son los modelos Small y Nano?

Los modelos Small (s) y Nano (n) de YOLO representan variantes reducidas y ligeras de las arquitecturas originales, diseñadas principalmente para funcionar en dispositivos con recursos limitados, como smartphones, sistemas embebidos o hardware con capacidades de procesamiento modestas.

### 3. Propósito común:

Estos modelos están optimizados para ofrecer un equilibrio entre velocidad y precisión, priorizando tiempos de inferencia rápidos y cargas computacionales bajas. Sin embargo, eso puede venir acompañado de una caída en precisión comparado con versiones más grandes.

## Tabla comparativa en nano y small:

Característica	Nano (n)	Small (s)
Tamaño (px)	Generalmente, tamaño de entrada 640x640 píxeles	También 640x640 píxeles, pero con mayor capacidad de procesamiento comparado con Nano
Precisión (mAP@50-95%)	Menor, típicamente en el rango de 37-40%	Mejor que Nano, en torno a 44-45%
Velocidad (ms, CPU ONNX)	Muy rápida (aproximadamente 3 ms)	Rápida, pero un poco menos que Nano
Parámetros (M)	Menos de 10 millones	Mayor que Nano, generalmente alrededor de 11 millones
FLOPs (B)	Muy ligera, menor uso de cálculos	Algo más pesada pero aún ligera
Adecuado para	Dispositivos con recursos extremadamente limitados y necesidad de alta velocidad	Hardware con algo más capacidad, con mejor equilibrio velocidad-precisión

### ¿Por qué escoger uno u otro?

- Nano (n): Ideal para aplicaciones donde la velocidad de inferencia es crítica y los recursos son limitados, como cámaras embebidas o dispositivos IoT. La precisión puede ser menor, pero permite un procesamiento casi en tiempo real.
- Small (s): Mejor opción si se dispone de algo más de capacidad computacional y se busca un balance entre rapidez y precisión. Es el más recomendable en contextos donde aún se requiere un rendimiento decente, pero con mayor confiabilidad en la detección.

## Tabla comparativa: Modelos Small (s) y Nano (n) de YOLOv5, YOLOv8 y YOLOv12

Modelo	Tipo	Velocidad (ms, CPU ONNX)	Ventajas	Desventajas
YOLOv5n	Nano (n)	~3.0 ms	Súper rápido, adecuado para hardware muy limitado; bajo consumo energético.	Menor precisión en detecciones; menos robusto a condiciones complejas.

<b>YOLOv5s</b>	Small (s)	~5.5 ms	Buen equilibrio entre velocidad y precisión; versátil para muchas aplicaciones.	Requiere hardware algo más potente que la versión Nano; menor precisión comparada con modelos más grandes.
<b>YOLOv8n</b>	Nano (n)	~1.2 ms	La más rápida de YOLOv8, ideal para tareas en tiempo real en hardware muy restringido.	Menor precisión; menos capacidad para detectar objetos pequeños o en condiciones difíciles.
<b>YOLOv8s</b>	Small (s)	~1.2 ms	Muy rápida, buena precisión para uso en tiempo real; adecuado para sistemas con recursos moderados.	Menor precisión en objetos pequeños comparado con modelos más grandes.
<b>YOLO12n</b>	Nano (n) (versión personalizada)	~2.6 ms	Alta velocidad, muy eficiente; adaptado para hardware muy limitado y aplicaciones en tiempo real.	Precisión menor que modelos más grandes; menos robusto ante escenarios complejos.
<b>YOLO12s</b>	Small (s) (versión personalizada)	~9.3 ms	Mejor precisión que Nano, buena velocidad; adecuado para hardware con capacidad moderada.	Más recursos que Nano, menor velocidad comparado con YOLOv8s y YOLOv5s.

Modelo	Ventajas	Desventajas	Uso recomendado en este proyecto

<b>YOLO v8s (Small)</b>	<ul style="list-style-type: none"> <li>- Balance adecuado entre precisión y velocidad.</li> <li>- Alta capacidad de detección, baja tasa de falsos negativos</li> <li>- Se adapta bien para tareas en tiempo real en hardware moderado.</li> <li>- Precisión (mAP) de aproximadamente 44.9%.</li> </ul>	<ul style="list-style-type: none"> <li>- Mayor tamaño y consumo computacional comparado con Nano.</li> <li>- Requiere hardware con capacidad media.</li> </ul>	Se recomienda para detección confiable en condiciones reales donde la seguridad y precisión son prioritarios.
<b>YOLO v8n (Nano)</b>	<ul style="list-style-type: none"> <li>- Muy rápido y eficiente en hardware muy limitado.</li> <li>- Menor tamaño de modelo, menor consumo de recursos.</li> <li>- Útil para plataformas con recursos muy restringidos.</li> </ul>	<ul style="list-style-type: none"> <li>- Menor precisión y capacidad de detección.</li> <li>- Mayor riesgo de falsos negativos, especialmente en tareas críticas como detección de armas.</li> <li>- Tasa de falsos negativos puede afectar la confiabilidad del sistema.</li> </ul>	Se descarta en este trabajo por su menor precisión, aunque podría considerarse en escenarios donde la velocidad y recursos sean el principal criterio.

Basándonos en los resultados de las métricas de evaluación, se decidió utilizar YOLOv8s ya que combina una buena precisión (44.9%) y una velocidad de inferencia adecuada (1.20 ms en CPU ONNX), satisfaciendo los requisitos para detectar armas en escenarios en tiempo real con hardware moderado.

#### **¿En qué se diferencian los modelos Small y Nano?**

El modelo Nano se dejó de lado debido a su menor capacidad de detección en comparación con Small, lo cual puede ocasionar mayor cantidad de falsos negativos en situaciones críticas, aunque en algunas aplicaciones de hardware muy limitado puede ser considerado.

## 4. Modelización

Se entrenaron tres versiones del algoritmo YOLO: YOLOv5s, YOLOv8s y YOLOv12s. El objetivo del entrenamiento de estas tres versiones fue evaluar cual ofrece un mejor rendimiento en términos de precisión, rapidez de entrenamiento y calidad de detección, teniendo en cuenta las restricciones computacionales disponibles.

El proceso de entrenamiento incluyó los siguientes pasos:

1. **Descarga y preparación de datos:** Utilizando la API de Roboflow, se descargó el conjunto de datos "test" en los formatos compatibles con cada versión del modelo.
2. **Entrenamiento de modelos:**

Para asegurar una comparación equitativa, se entrenó cada modelo en su versión small (s), lo que permitió mantener tiempos de entrenamiento razonables.

- Para YOLOv5s, se utilizó el script train.py proporcionado en el repositorio oficial <https://github.com/ultralytics/yolov5>.
- Para YOLOv8s y YOLOv12s, se empleó la biblioteca ultralytics, que proporciona una interfaz unificada y simplificada para el entrenamiento.

En todos los casos se utilizaron los mismos hiperparámetros:

- Tamaño de imagen: 800x800 píxeles
- Tamaño de batch: 16
- Número de épocas: 100

3. **Evaluación de resultados:** Tras finalizar el entrenamiento de cada versión, se descargaron y comprimieron las carpetas runs, que contienen los modelos entrenados y las métricas recolectadas durante el proceso.

4. **Problemas detectados con modelos más pesados (como YOLOv12):**

- o Lentitud: No permiten procesos en tiempo real en hardware con recursos limitados.
- o Consumo alto de recursos: Mayor uso de GPU/CPU, menos eficiente para despliegues en dispositivos embebidos.
- o Dificultad en entrenamiento: Mayor tiempo de entrenamiento y ajuste, con menor rapidez para iterar.

Link al repositorio GitHub: <https://github.com/Hato959/CC235-TP-TF-2025-1>

## 5. Publicación de resultados

A continuación, se presentan los resultados obtenidos para cada uno de los modelos entrenados, utilizando las métricas estándar de evaluación en tareas de detección de objetos: Precisión, Recall, mAP@0.5 y mAP@0.5:0.95.

**Precisión:** La precisión mide la proporción de predicciones positivas que fueron realmente correctas. Es decir, cuántas de las detecciones realizadas por el modelo corresponden efectivamente a objetos verdaderos. Su valor varía entre 0 y 1, donde 1 indica una precisión perfecta.

**Recall:** El recall mide la capacidad del modelo para encontrar todos los objetos reales en la imagen. Es la proporción de objetos correctamente detectados con respecto al total de objetos existentes. Su valor varía entre 0 y 1.

**mAP:** El AP se calcula para cada clase por separado y representa el área bajo la curva de precisión vs. recall. Esto permite evaluar el desempeño del modelo específicamente para cada categoría de objeto. No se calcula combinando todas las clases, ya que eso ocultaría el rendimiento individual de cada una.

- **mAP@0.5:** Calcula el mAP considerando que una detección es correcta si la superposición (IoU) entre la predicción y la anotación real es de al menos 0.5 (50%).
- **mAP@0.5:0.95:** Calcula el mAP promediando los resultados obtenidos para distintos umbrales de IoU (de 0.5 a 0.95 en pasos de 0.05). Es una métrica más

estricta y completa, ya que evalúa tanto la precisión de la localización como la correcta clasificación.

### Comparación de modelos:

Modelo	Precision	Recall	mAP@0.5	mAP@0.5:0.95
<b>YOLOv5s</b>	0.73969	0.66662	0.69362	0.42625
<b>YOLOv8s</b>	0.78656	0.72473	0.78009	0.54554
<b>YOLOv12s</b>	0.75515	0.70118	0.75901	0.53217

En el contexto del problema (detección de armas), el Recall se prioriza sobre la Precisión, ya que:

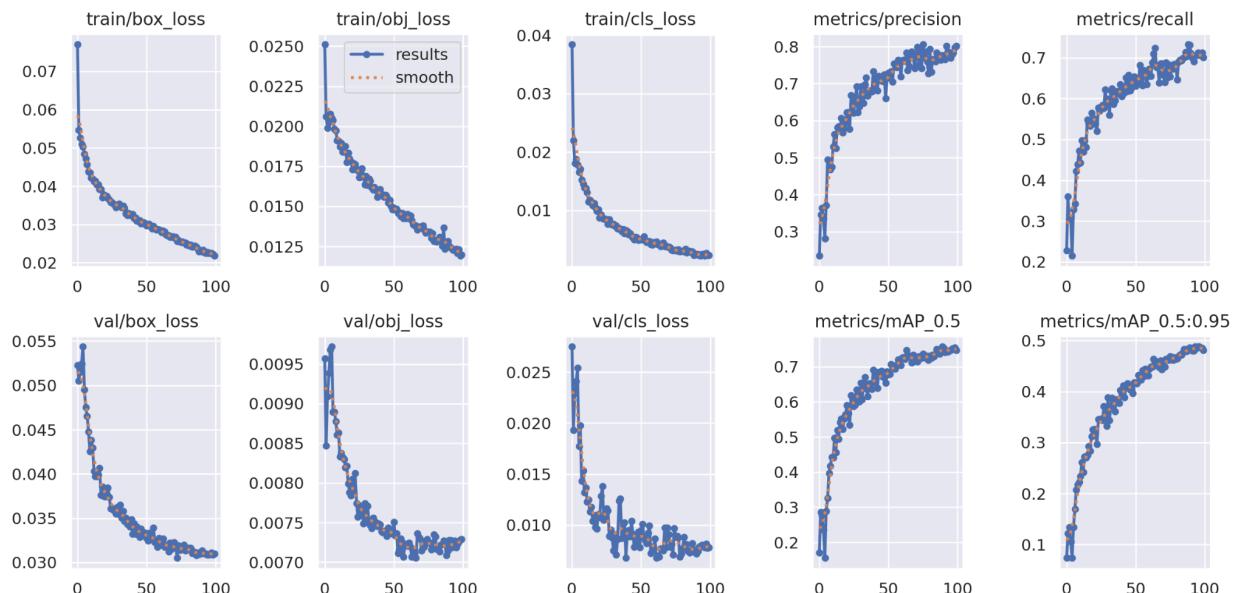
- Un falso negativo (no detectar un arma real) representa un riesgo alto.
- Un falso positivo (detectar erróneamente un objeto como arma) puede gestionarse manualmente sin poner en riesgo la seguridad.

Con base en esta prioridad, YOLOv8s se destaca como el mejor modelo:

- Tiene el mayor Recall (0.72473), lo que significa que detecta más armas reales.
- También presenta los mejores resultados en mAP, indicando un sólido desempeño general.
- Aunque su Precisión es ligeramente inferior a la de YOLOv5s, el equilibrio general de sus métricas lo convierte en el candidato más confiable.

### Resultados del entrenamiento por versión

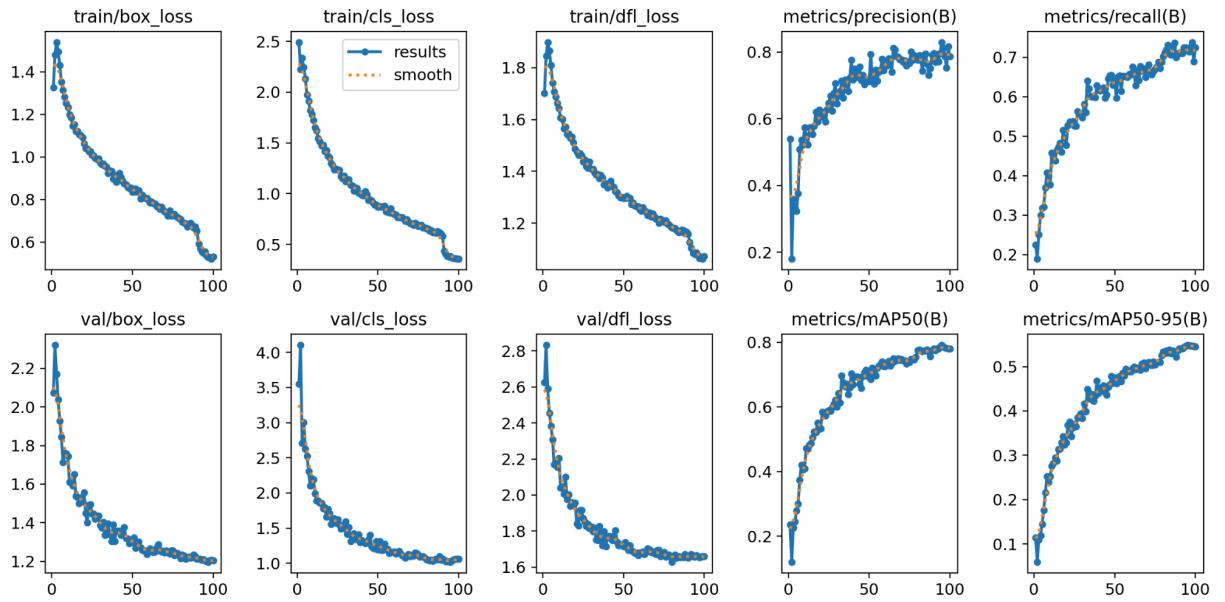
#### - YOLOv5



Las curvas muestran una disminución constante de las pérdidas (box\_loss, obj\_loss, cls\_loss) tanto en entrenamiento como en validación, lo que indica que el modelo aprendió de forma efectiva sin signos de sobreajuste. Además, las métricas de precisión y recall aumentan progresivamente, alcanzando valores de 0.8 y 0.7 respectivamente. La métrica mAP@0.5 se estabiliza cerca de 0.8, mientras que

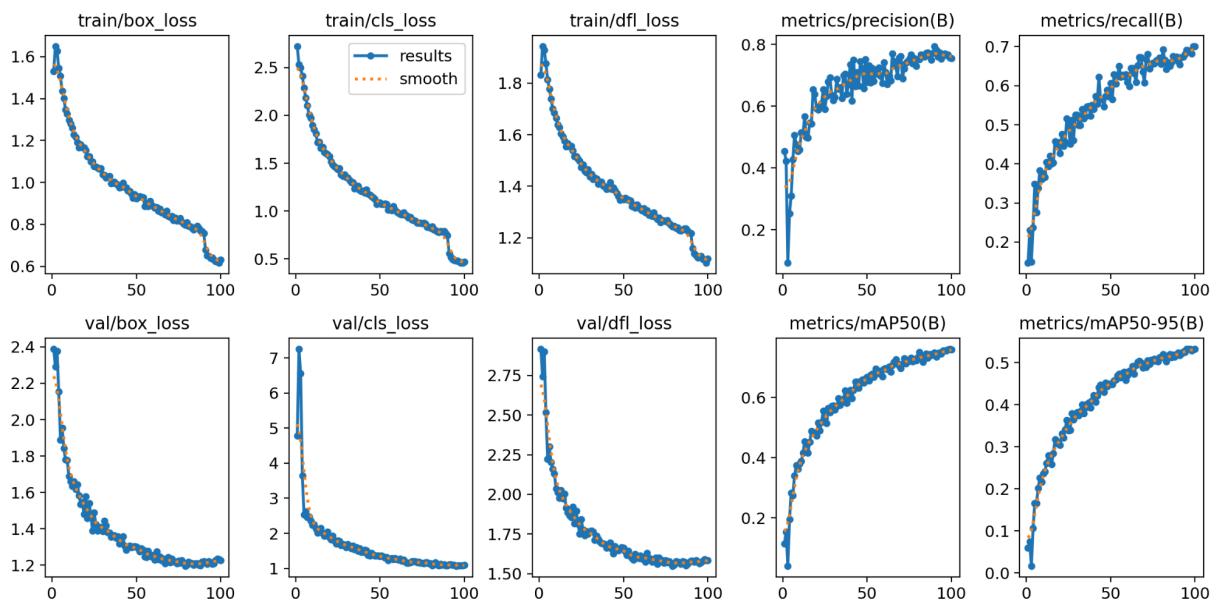
mAP@0.5:0.95 alcanza ~0.48, reflejando un buen desempeño general en la detección y clasificación de objetos.

### - YOLOv8



Durante el entrenamiento del modelo de detección de objetos, se observó una disminución constante en las pérdidas de entrenamiento y validación (box\_loss, cls\_loss, dfl\_loss), lo que indica una mejora progresiva en la precisión de las predicciones. Asimismo, las métricas de evaluación (precision, recall, mAP50, mAP50-95) mostraron un aumento sostenido, alcanzando valores satisfactorios al final del proceso (por ejemplo, mAP50-95 llegó a 0.55). No se evidenció sobreajuste, ya que las curvas de validación siguen una tendencia similar a las de entrenamiento. En conjunto, estos resultados reflejan un entrenamiento exitoso y un modelo con buen rendimiento general.

### - YOLOv12



El entrenamiento muestra una reducción constante en las pérdidas de entrenamiento y validación, especialmente en box\_loss, cls\_loss y dfl\_loss, lo que indica una mejora progresiva en la precisión de las predicciones. Aunque la pérdida de clasificación en validación (val/cls\_loss) presenta un pico inicial, luego se estabiliza y disminuye, lo que sugiere un ajuste adecuado del modelo. Las métricas de evaluación también mejoran de forma sostenida: la precisión y recall alcanzan valores cercanos a 0.8 y 0.7 respectivamente, mientras que el mAP50-95 llega a 0.52. Estos resultados indican un entrenamiento efectivo, sin evidencia de sobreajuste.

### Comparación del batch con los labels y batch de predicción:

A continuación se muestran ejemplos visuales generados durante el entrenamiento de los modelos. Estas imágenes permiten observar cómo responde el modelo frente a las imágenes del conjunto de entrenamiento o validación.

#### YOLOv5



A continuación se presentan imágenes comparativas entre el batch de etiquetas reales (batch\_label, izquierda), que corresponden a las anotaciones proporcionadas por el dataset, y el batch de predicciones (batch\_pred, derecha), generadas por el modelo después del entrenamiento.

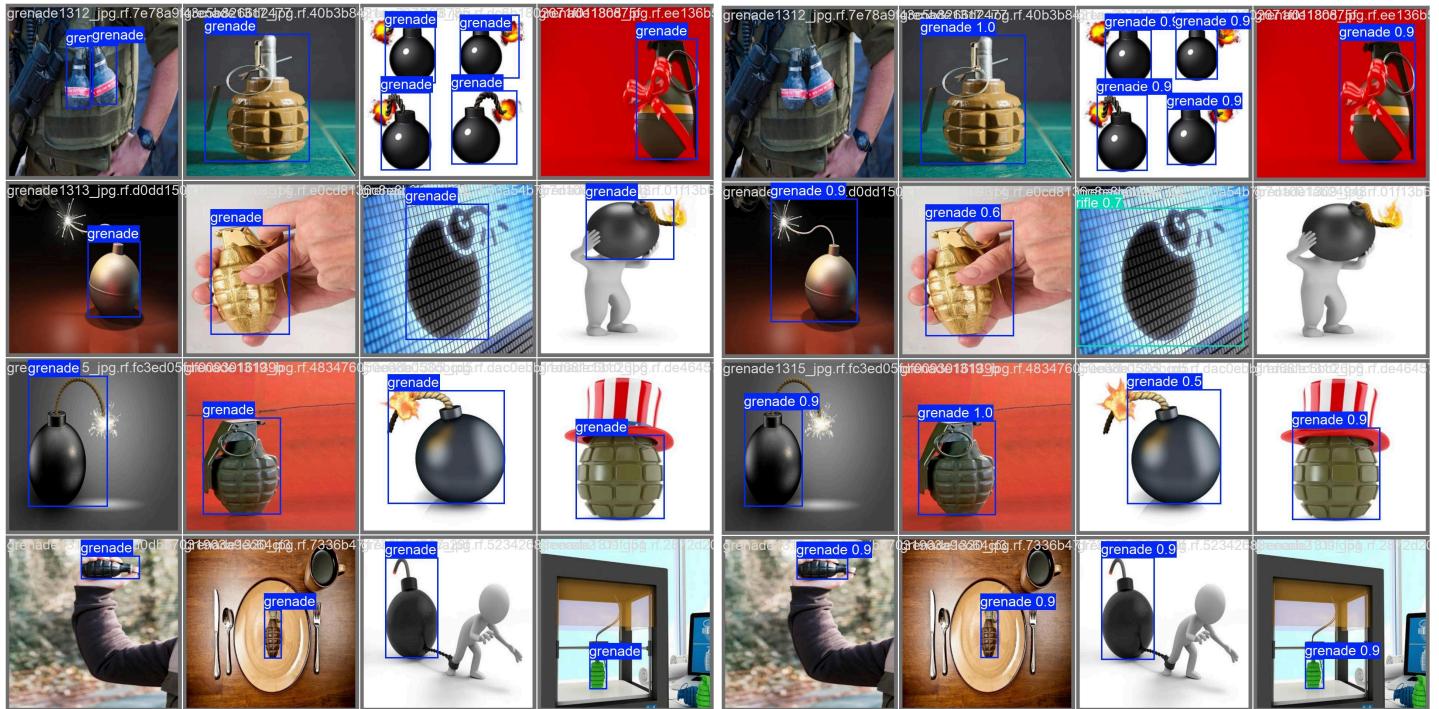
Se puede observar que, en algunos casos, el modelo presenta confianza baja (confidence score), lo que indica una menor certeza en la detección. Además, hay imágenes en las que el modelo no logra identificar correctamente el objeto, omitiendo algunas armas presentes en la escena.

## YOLOv8



En el caso del modelo YOLOv8, se observa que el confidence score es considerablemente alto, manteniéndose entre 0.9 y 1.0 en la mayoría de las predicciones. Sin embargo, a pesar de esta alta confianza, el modelo presenta algunos falsos positivos, identificando erróneamente ciertos objetos como granadas o pistolas cuando en realidad no lo son. Aun así, en términos generales, las predicciones del modelo YOLOv8 son más precisas y consistentes en comparación con YOLOv5, logrando detectar una mayor cantidad de armas verdaderas y con mayor seguridad.

## YOLOv12



Finalmente, en el caso del modelo YOLOv12s, se observa que el confidence score presenta una mayor variabilidad, oscilando entre 0.6 y 0.9. Además, en algunas imágenes no logra detectar correctamente el arma, lo que indica una menor consistencia en sus predicciones en comparación con YOLOv8s.

## 6. Conclusiones

- El desarrollo y entrenamiento de modelos de detección de armas mediante visión por computadora demuestra ser una solución efectiva y viable para mejorar los sistemas de seguridad en diversos escenarios.
- La precisión y fiabilidad del sistema dependen en gran medida de la calidad y cantidad de los datos utilizados en el entrenamiento, destacando la importancia de contar con un conjunto de datos representativo y bien anotado.
- La implementación de modelos como YOLOv8s permite una detección en tiempo real, contribuyendo significativamente a la prevención y respuesta rápida ante situaciones de riesgo en ámbitos de seguridad pública y vigilancia.
- La comparación entre las versiones Nano y Small de los modelos YOLO permitió identificar que YOLOv8s ofrece el mejor equilibrio entre precisión, velocidad y eficiencia, cumpliendo con los requerimientos del sistema propuesto. Aunque modelos como YOLOv8n destacan por su rapidez y bajo consumo de recursos, su menor precisión representa un riesgo en contextos sensibles como la detección de armas. Por lo tanto, YOLOv8s fue seleccionado como el modelo más adecuado, garantizando una detección confiable y en tiempo real en dispositivos con capacidad moderada.
- YOLOv8s demostró ser el modelo más adecuado para el contexto del problema, dado que obtuvo el mayor Recall (0.72473), métrica prioritaria en tareas de

detección de armas. Esto significa que tiene mayor capacidad para detectar armas reales, minimizando los falsos negativos, los cuales representan un riesgo significativo para la seguridad.

- En las imágenes comparativas, se observó que YOLOv8s realiza detecciones más precisas y con mayor confianza, aunque aún presenta algunos falsos positivos. Sin embargo, estos son aceptables dentro del contexto del problema, ya que pueden corregirse manualmente sin comprometer la seguridad. Por lo tanto, YOLOv8s se posiciona como el modelo más confiable y efectivo para la tarea de detección de armas, considerando tanto el análisis cuantitativo de métricas como la evaluación cualitativa de las predicciones.

## 7. Referencias bibliográficas

ashish. (2022). test Dataset [Dataset]. Roboflow Universe.  
<https://universe.roboflow.com/ashish-cuamw/test-y7rj3>

Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767.  
<https://arxiv.org/abs/1804.02767>

Redmon, J., & Zhang, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767.  
<https://arxiv.org/abs/1804.02767>

LearnOpenCV. (2022, agosto 9). *Mean Average Precision (mAP) in Object Detection*. LearnOpenCV.  
<https://learnopencv.com/mean-average-precision-map-object-detection-model-evaluation-metric/>

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Advances in Neural Information Processing Systems (NeurIPS).  
<https://arxiv.org/abs/1506.01497>

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. CVPR.  
<https://arxiv.org/abs/1512.03385>

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D & Rabinovich, A. (2015). Going deeper with convolutions. CVPR.  
<https://arxiv.org/abs/1409.4842>