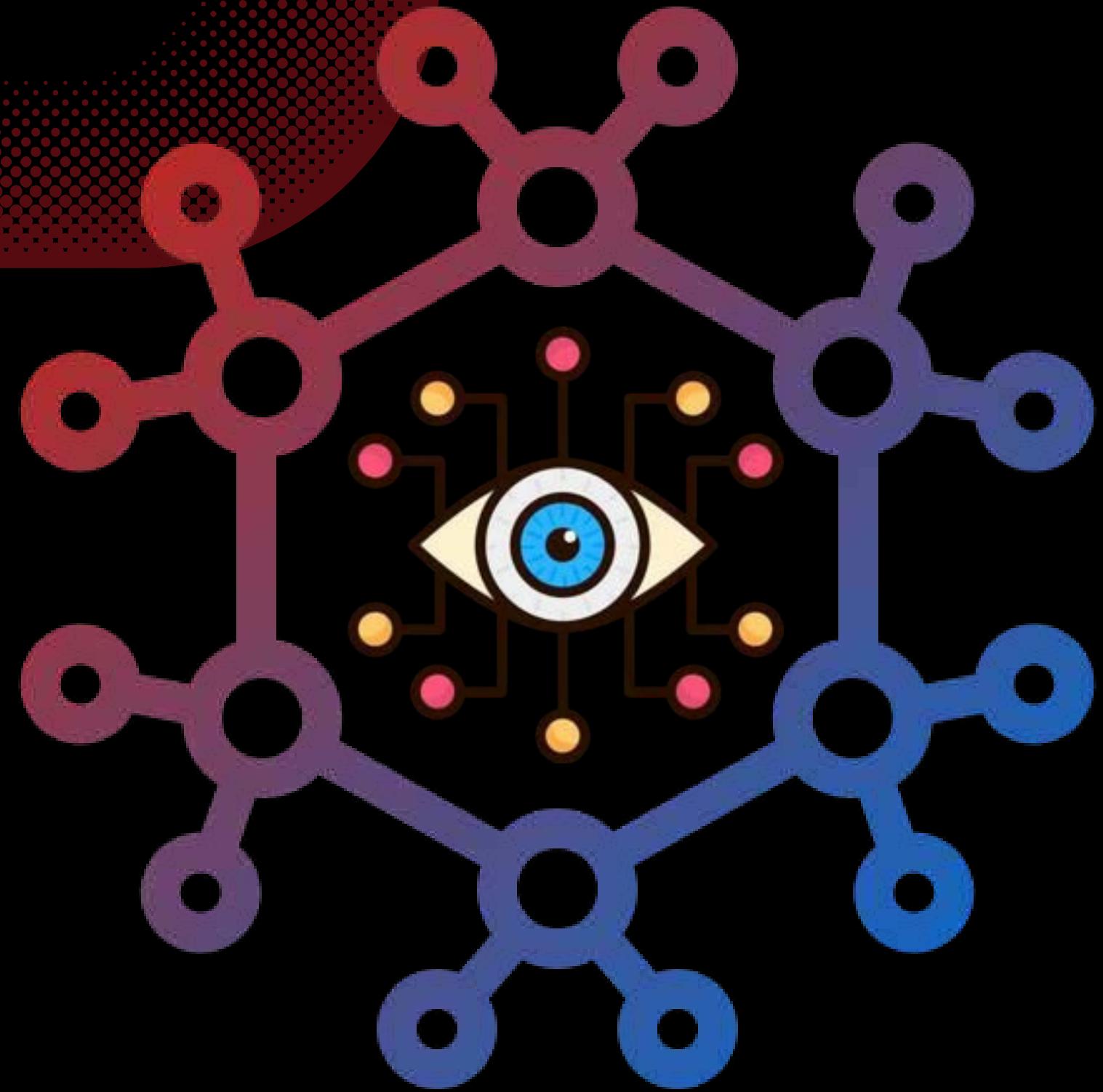




IMPLEMENTACIÓN DE UN SISTEMA DE ALERTA TEMPRANA PARA LA SEGURIDAD FÍSICA EN LA UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE CON VISIÓN ARTIFICIAL

INTEGRANTES
PAOLA FARINANGO
CRISTIAN IDROBO



PROBLEMA

El problema principal es implementar un sistema de seguridad y alerta temprana en la Universidad de las Fuerzas Armadas ESPE, en cumplimiento con la actual ley aprobada de uso y portamiento de armas en el territorio Ecuatoriano. Se busca garantizar la seguridad de la comunidad universitaria mediante el uso de tecnologías de visión artificial y detección, con el objetivo de prevenir la presencia de armas en el campus universitario. Esta medida contribuirá a mantener un entorno educativo seguro y protegido.

HERRAMIENTAS



Roboflow

Plataforma de inteligencia artificial diseñada para facilitar la construcción y despliegue de modelos de visión artificial.



Colab

Es un servicio gratuito de nube alojado por Google para fomentar la investigación sobre Aprendizaje de Máquina e inteligencia Artificial.



YOLOv5

Es el modelo de identificación de objetos más preciso y rápido disponible actualmente.



Anaconda

Es una plataforma de programación y ciencia de datos que simplifica la gestión de paquetes y entornos de desarrollo en Python.



PALABRAS CLAVE

Armas

- Weapon detection
- Firearms recognition
- Gun detection

Uso de armas

- Violent behavior detection
- Aggression detection
- Fighting detection

CARGA DEL DATASET



PROYECTOVISION

Browse [How to Search](#)

Search your images by subject (e.g. "brown dog" or "green apple") [Search](#)

De-Select Select All 0 Images Selected

Filename: Split: **ALL** TRAIN VALID TEST Classes: All Classes Tags: All Sort By: Newest

Armas [Object Detection](#)

3309 2

TRAIN ABBframe00190.j... TRAIN ABbframe00160.j... VALID ABBframe00280.... VALID Abbframe00316.j... TRAIN ABmframe00127.... TRAIN ABbframe00346.... TRAIN ABBframe00346.... TRAIN ABBframe00421.j...

TRAIN ABBframe00421.j... TRAIN ABmframe00277.... TRAIN ABmframe00133... TRAIN ABmframe00211.... TRAIN ABmframe00301.... VALID ABmframe00301.... VALID ABBframe00370.... TEST ABmframe00412...

Images per page: 50 1 - 50 of 3309

ETIQUETADO/ANOTACIÓN



Annotations

Group: armas

CLASSES LAYERS

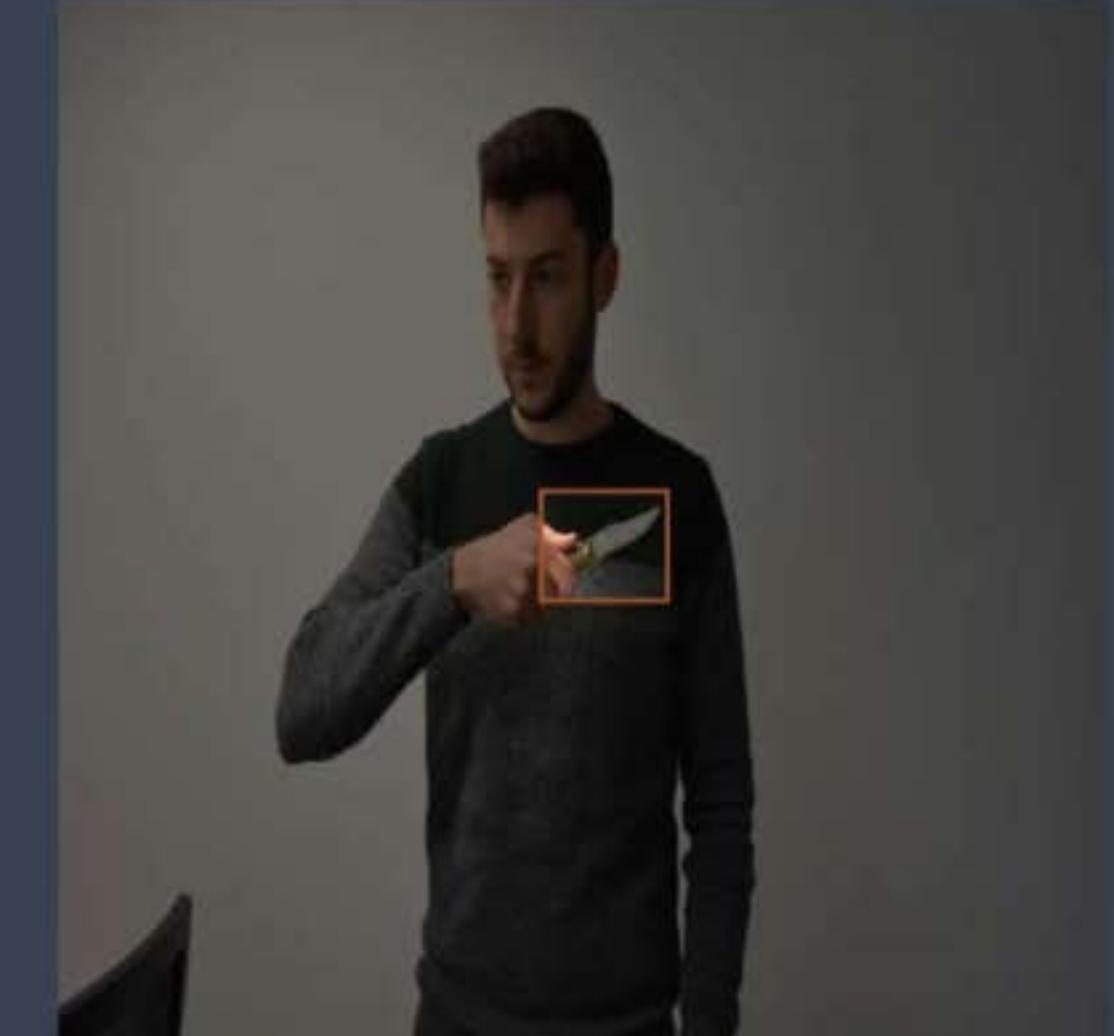
UNUSED CLASSES

1
5
6
Postura Sospechosa
arma
arma

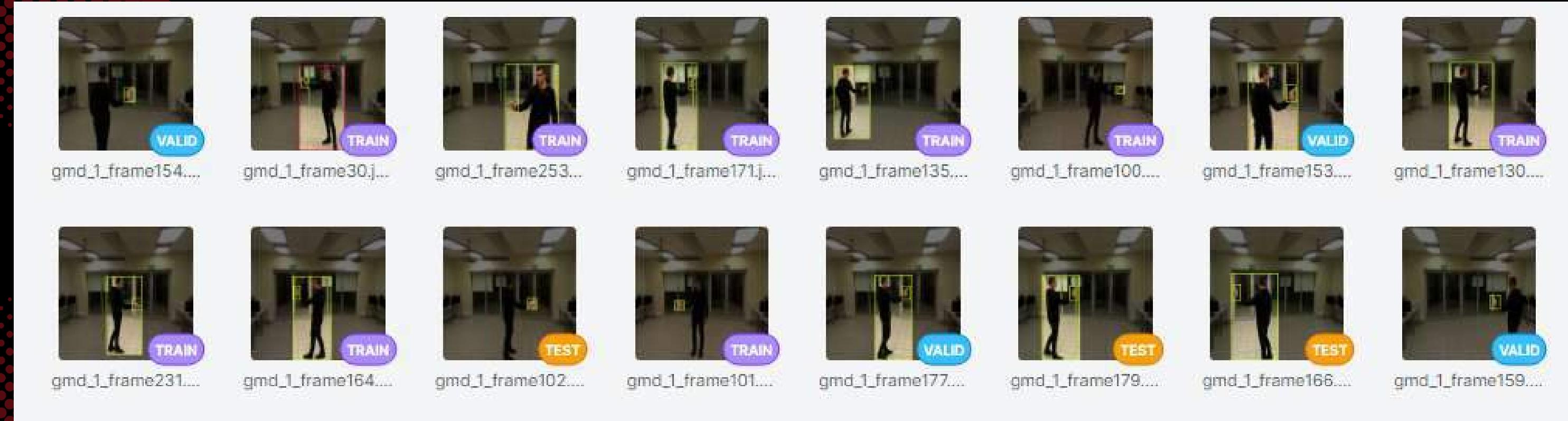
Tags

No Tags Applied

Type and select tags below to add them to the image.



DATOS CARGADOS



División de datos

VALID

Estas imágenes se utilizan para evaluar y ajustar el rendimiento del modelo durante el entrenamiento.

TRAIN

Estas imágenes se utilizan para entrenar el modelo, son para enseñar a reconocer y clasificar los objetos, como personas sospechosas o armas.

TEST

Estas imágenes evalúan el rendimiento final del modelo entrenado, no se utilizan durante el entrenamiento ni validación, se separan para probar la capacidad del modelo para generalizar a datos nuevos y no vistos previamente.

Preprocesamiento

PREPROCESSING	Auto-Orient: Applied Resize: Stretch to 640×640
AUGMENTATIONS	Outputs per training example: 3 Flip: Horizontal Crop: 0% Minimum Zoom, 20% Maximum Zoom Shear: ±15° Horizontal, ±15° Vertical Grayscale: Apply to 29% of images Blur: Up to 1.5px
DETAILS	Version Name: 2023-06-27 12:19am Version ID: 3 Generated: Jun 27, 2023 Annotation Group: armas

Jupyter Terminal Raw URL

Paste this snippet into [a notebook from our model library](#) to download and unzip [your dataset](#):

```
!pip install roboflow
from roboflow import Roboflow
rf = Roboflow(api_key="REDACTED")
project = rf.workspace("proyectovision").project("armas-koñy")
dataset = project.version(2).download("yolov5")
```

Warning: Do not share this snippet beyond your team, it contains a private key that is tied to your Roboflow account. Acceptable use policy applies.

DETECCIÓN DE OBJETOS



Entrenamiento del Modelo

coColab

1

```
#clone YOLOv5 and
!git clone https://github.com/ultralytics/yolov5 #clone repo
%cd yolov5
%pip install -qr requirements.txt #install dependencies
%pip install -q roboflow

import torch
import os
from IPython.display import Image, clear_output # to display images

print(f"Setup complete. Using torch {torch.__version__}({{torch.cuda.get_device_properties(0).name if torch.cuda.is_available() else 'CPU'}})")
```

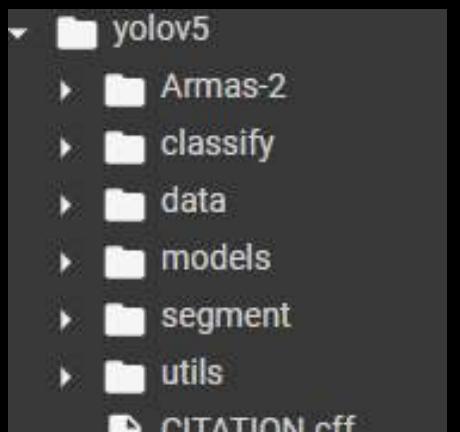
Instalamos Roboflow.

Agregamos el código obtenido del preprocesamiento de roboflow denominado "data.yaml"

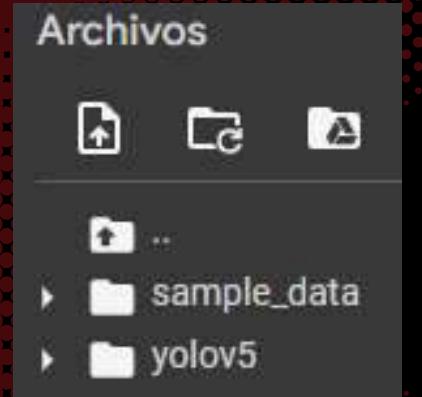
2

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="SzvijpsX3s8ONMLD0Dmm")
project = rf.workspace("proyectovision").project("armas-ko3ny")
dataset = project.version(2).download("yolov5")
```



Clonación de la librería
yolov5 para el
entrenamiento y
dependencias



3

```
!python train.py --img 416 --batch 16 --epochs 100 --data {dataset.location}/data.yaml --weight yolov5s.pt --cache --name yolov5s_results
```

AutoAnchor: 5.05 anchors/target, 1.000 Best Possible Recall (BPR). Current anchors are a good fit to dataset ✓
Plotting labels to runs/train/yolov5s_results/labels.jpg...
Image sizes 416 train, 416 val
Using 2 dataloader workers
Logging results to runs/train/yolov5s_results
Starting training for 100 epochs...

Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
0/99	1.52G	0.1162	0.02168	0.02878	32	416: 100% 16/16 [00:07<00:00, 2.26it/s]
	Class	Images	Instances	P	R	mAP50 mAP50-95: 100% 1/1 [00:01<00:00, 1.43s/it]
	all	28	53	0.99286	0.522	0.99716 0.99243

Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
1/99	1.52G	0.09793	0.02739	0.02427	47	416: 100% 16/16 [00:02<00:00, 7.76it/s]
	Class	Images	Instances	P	R	mAP50 mAP50-95: 100% 1/1 [00:00<00:00, 3.75it/s]
	all	28	53	0.00544	0.641	0.101 0.0435

Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
2/99	1.52G	0.08619	0.02733	0.01974	45	416: 100% 16/16 [00:03<00:00, 4.97it/s]
	Class	Images	Instances	P	R	mAP50
	all	28	53	0.54	0.433	0.145
						0.0719

Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size	
3/99	1.52G	0.07811	0.02558	0.01624	43	416: 100% 16/16 [00:02<00:00, 7.75it/s]	
	Class	Images	Instances	P	R	mAP50	mAP50-95: 100% 1/1 [00:00<00:00, 4.63it/s]
	11	55	55	0.755	0.755	0.755	0.755

Finalmente ejecutamos la linea de entrenamiento

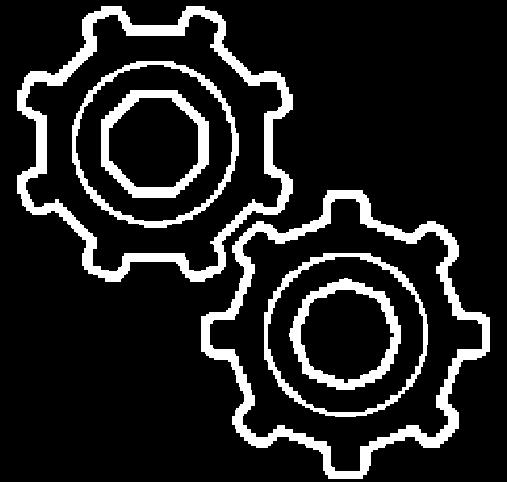
Prueba en imágenes

4

```
!python detect.py --weights runs/train/yolov5s_results/weights/best.pt --img 416 --conf 0.1 --source {dataset.location}/test/images
```

detect: weights=['runs/train/yolov5s_results/weights/best.pt'], source=/content/yolov5/Armas-2/test/images, data=data/coco128.yaml, imgsz=[416, 416], conf_thres=0.4
YOLOv5 🚀 v7.0-187-g0004c74 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)

PRUEBA EN EL ENTORNO LOCAL



INSTALACIÓN DE LIBRERÍAS
Y DEPENDENCIAS



CREACIÓN Y
ENTRENAMIENTO DE LA
RED NEURONAL

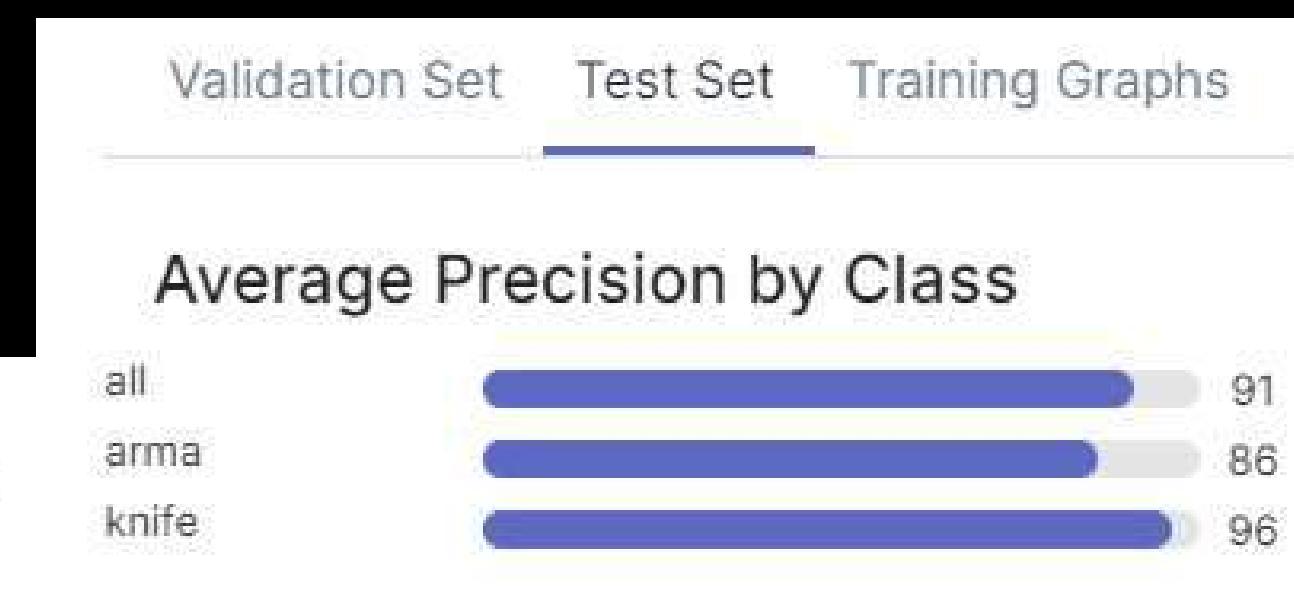
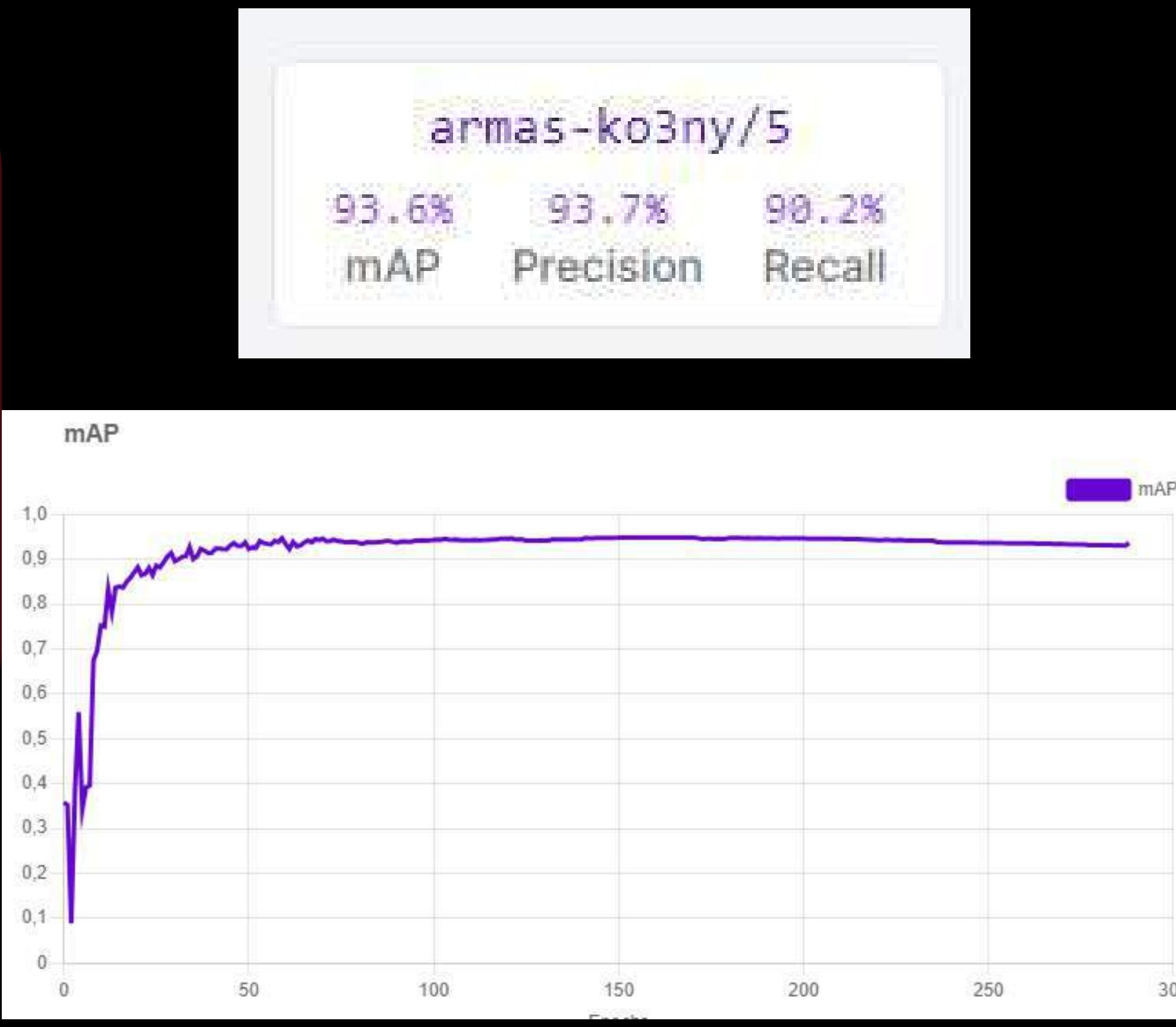


RESULTADOS EN TIEMPO
REAL

RESULTADOS

- Red neuronal entrenada para el reconocimiento en tiempo real de Armas de fuego y armas blancas.
- Implementar un sistema de alerta temprana que notificará al personal de seguridad de la universidad cuando se detecten dichos objetos, permitiendo una respuesta inmediata y adecuada.

Evaluación de conjunto de validación y conjunto de entrenamiento



REFERENCIAS

M. T. Bhatti, M. G. Khan, M. Aslam and M. J. Fiaz, "Weapon Detection in Real-Time CCTV Videos Using Deep Learning," in IEEE Access, vol. 9, pp. 34366-34382, 2021, doi: 10.1109/ACCESS.2021.3059170.

(S/f). Aplicaciones.ai. Recuperado el 13 de junio de 2023, de <https://aplicaciones.ai/roboflow/#:~:text=Roboflow%20es%20una%20plataforma%20de,modelos%20de%20visi%C3%B3n%20por%20ordenador>.

Williams, K. (2021, mayo 24). Why roboflow train? Roboflow Blog. <https://blog.roboflow.com/why-roboflow-train/>

GRACIAS