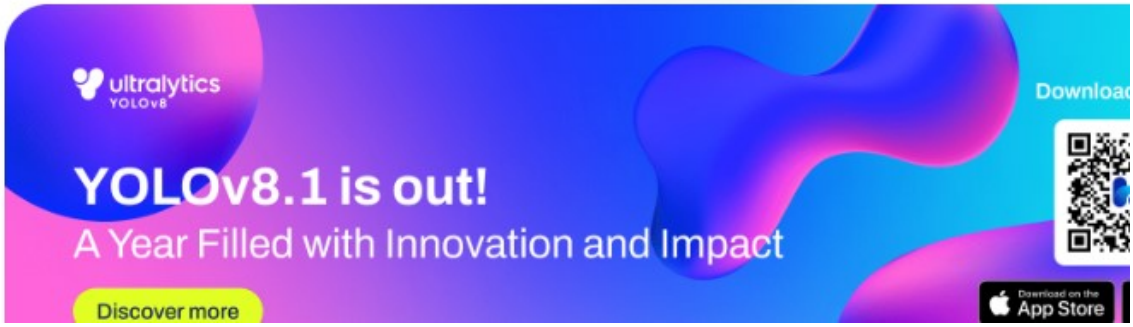
Capítulo 1: Conociendo YOLO (You Only Look Once)



Machine Learning _YOLO

Toda la documentación está disponible en :

https://docs.ultralytics.com/es/

1. Introducción

YOLO, acrónimo de "You Only Look Once" (Solo Miras Una Vez), representa un sistema innovador en el ámbito de la detección de objetos, diseñado para procesar imágenes en tiempo real. Se distingue de otros sistemas de detección de objetos por tratar la detección como un único problema de regresión, prediciendo directamente las cajas delimitadoras y las probabilidades de clases de imágenes completas en una sola evaluación. Esta característica le permite alcanzar velocidades y precisión elevadas, lo cual es esencial para aplicaciones que necesitan procesamiento en tiempo real como el análisis de vídeo, los vehículos autónomos y los sistemas de vigilancia.

La arquitectura de YOLO incluye múltiples capas, con capas convolucionales encargadas de procesar la imagen de entrada. Emplea técnicas como la normalización por lotes y el abandono para mejorar el rendimiento y prevenir el sobreajuste. Un modelo típico de YOLO redimensionará la imagen de entrada a un tamaño estándar (por ejemplo, 448x448 píxeles) y luego la procesará a través de sus capas para predecir objetos y sus ubicaciones dentro de la imagen **(Learn R, Python & Data Science Online).**

Ultralytics ha jugado un papel crucial en el desarrollo y mantenimiento de los modelos YOLO, incluida la versión más reciente, YOLOv8. Esta versión continúa el éxito de las iteraciones anteriores, introduciendo características y mejoras para un rendimiento, flexibilidad y eficiencia mejorados. YOLOv8 no solo es rápido y preciso, sino también adaptable a una amplia gama de plataformas de hardware, desde dispositivos de borde hasta APIs en la nube, lo que lo hace adecuado para diversas aplicaciones en diferentes dominios (Home).

Desde su primera liberación en 2015, YOLO ha experimentado varias actualizaciones, cada una introduciendo mejoras en velocidad, precisión y funcionalidad. Desde YOLOv1 hasta la última YOLOv8, cada versión ha contribuido a la evolución del modelo, abordando limitaciones de versiones anteriores e introduciendo innovaciones como predicciones a múltiples escalas, mejores clasificadores de respaldo y nuevos algoritmos para la ampliación de datos y funciones de pérdida **(Learn R, Python & Data Science Online) (PJREDDIE).**

Para su uso práctico, los modelos de YOLO pueden ser entrenados en conjuntos de datos personalizados, y hay modelos preentrenados disponibles para una implementación rápida. La capacidad de YOLO para realizar detecciones en tiempo real en webcams o archivos de vídeo lo hace increíblemente útil para tareas de detección de objetos en vivo (PJREDDIE).

La documentación de Ultralytics proporciona una guía completa para utilizar YOLOv8, cubriendo la instalación, predicción, entrenamiento en conjuntos de datos personalizados y realización de diversas tareas de visión como segmentación, clasificación y estimación de poses (Home). Este amplio soporte facilita tanto a usuarios nuevos como experimentados la implementación de YOLO en sus proyectos, aprovechando sus capacidades para satisfacer sus necesidades específicas.

YOLO (You Only Look Once) no se clasifica como un modelo de aprendizaje auto-supervisado. En cambio, es un ejemplo prominente de un sistema de detección de objetos basado en aprendizaje supervisado. YOLO utiliza un conjunto de datos etiquetado, donde cada imagen de entrenamiento ha sido marcada con cajas delimitadoras alrededor de los objetos y

clasificada en categorías específicas. Esto permite que el modelo aprenda a identificar y localizar objetos dentro de imágenes nuevas basándose en este entrenamiento previo.

El aprendizaje supervisado, como el que utiliza YOLO, depende de datos previamente etiquetados por humanos para aprender a realizar tareas de clasificación o regresión. Esto difiere del aprendizaje auto-supervisado, que no requiere etiquetas explícitas para cada ejemplo de datos durante el entrenamiento, permitiendo al modelo aprender patrones y características directamente de la estructura no etiquetada de los datos (Blogthinkbig.com) (abdatum).

YOLO es particularmente conocido por su velocidad y precisión en la detección de objetos en tiempo real, lo que lo hace ideal para aplicaciones que requieren un procesamiento rápido de imágenes o video, como la vigilancia de seguridad, los sistemas de vehículos autónomos y muchas otras aplicaciones de visión por computadora (Learn R, Python & Data Science Online) (PJREDDIE).

2. Utilidades del sistema.

2.1 ¿De dónde salen los datos de las máquinas operativas ?

La lista COCO, o más formalmente el conjunto de datos COCO (Common Objects in Context), es un recurso ampliamente utilizado en el campo de la visión por computadora, especialmente en tareas de detección, segmentación y reconocimiento de objetos en imágenes. COCO proporciona un conjunto de datos rico y extenso diseñado para el desarrollo y la evaluación de algoritmos de visión por computadora.

Características Principales

Imágenes Diversas: COCO contiene más de 330,000 imágenes, las cuales están etiquetadas con más de 2.5 millones de instancias de objetos pertenecientes a 80 categorías diferentes, como personas, vehículos, animales y muebles.

Annotaciones Ricas: Cada imagen en COCO ha sido meticulosamente anotada para incluir no solo las cajas delimitadoras que identifican la ubicación de los objetos dentro de las imágenes, sino también segmentaciones precisas que describen los contornos exactos de los objetos. Esto hace que COCO sea particularmente valioso para tareas que requieren un nivel de detalle fino, como la segmentación de instancias y la segmentación semántica.

Contexto de Objetos: Una característica distintiva de COCO es su enfoque en capturar objetos en su contexto natural. Las imágenes a menudo presentan escenas complejas con múltiples objetos interactuando de maneras significativas, lo que ayuda a desarrollar algoritmos capaces de comprender mejor el contexto y la relación entre objetos en una escena.

Usos en Investigación y Desarrollo

Los investigadores y desarrolladores utilizan COCO para una variedad de propósitos:

Entrenamiento y Evaluación: Es común utilizar COCO para entrenar modelos de detección de objetos desde cero o ajustarlos (finetuning) a partir de modelos preentrenados. Además, su amplio conjunto de datos de validación y pruebas permite una evaluación rigurosa del rendimiento del modelo.

Desarrollo de Nuevas Tecnologías: Las características únicas de COCO impulsan la innovación en técnicas avanzadas de visión por computadora, incluyendo la mejora de la precisión de la detección de objetos, la comprensión del contexto visual y el desarrollo de modelos capaces de realizar segmentaciones detalladas.

Competencias y Desafíos: COCO ha sido la base para numerosas competencias y desafíos, como los organizados en las conferencias ICCV y ECCV, donde investigadores de todo el mundo compiten para desarrollar algoritmos con el mejor rendimiento en las tareas de visión por computadora soportadas por el conjunto de datos COCO.

El conjunto de datos COCO continúa siendo una herramienta invaluable para el avance de la investigación en visión por computadora, facilitando

tanto el desarrollo tecnológico como la evaluación objetiva de los progresos en este campo.

2.2 Como puedo usar Yolo en sus versiones para el análisis de objetos??

YOLOv8 es la iteración más reciente en la serie YOLO de detectores de objetos en tiempo real de Ultralytics. Ofrece rendimiento de vanguardia en términos de precisión y velocidad, mejorando las versiones anteriores de YOLO con nuevas características y optimizaciones. YOLOv8 presenta una arquitectura avanzada, es ancla-libre para una detección más eficiente, y equilibra óptimamente la precisión con la velocidad. Hay varias versiones de YOLOv8 diseñadas para tareas específicas en visión por computadora, como YOLOv8n y YOLOv8s, que están optimizadas para diferentes requisitos de rendimiento, asegurando un alto rendimiento y precisión en una variedad de aplicaciones.

3. Script de iniciación:

Instalamos ultralytics y con el fichero yolov8n lo ponemos en nuestro entorno de trabajo.

1º Desacargar archivo Yolov8n =>
https://docs.ultralytics.com/models/yolov8/#performance-metrics

2º Abrir visual Code Studio e instalar ultralytics

Pip Install ultralytics

3º Copiamos el código y ejecutamos

```
from ultralytics import YOLO

# Instancia del modelo YOLO con los pesos especificados
pesos_yolo = "C:/Users/Usuario/Desktop/YoloV8/yolov8n.pt"
model = YOLO(pesos_yolo)

try:
    # Realiza la detección de objetos en el flujo de video de la cámara del computador
    results = model(source=0, show=True, conf=0.3, save=True)
except Exception as e:
    print("Ocurrió un error al realizar la detección de objetos:", e)
```

Script 1

Este algoritmo debería abrir una ventana que muestra la salida de la detección de objetos en tiempo real utilizando la cámara del ordenador como entrada de video. La detección de objetos se realiza utilizando el modelo YOLOv8 con los pesos especificados. Los objetos detectados se mostrarán en la ventana, y la salida visual también se guardará en un archivo según lo especificado por el parámetro

El archivo yolov8n.pt es un archivo de pesos pre-entrenados para el modelo YOLOv8. Este archivo contiene los parámetros aprendidos por el modelo durante el proceso de entrenamiento en un conjunto de datos específico. En el caso de YOLOv8, estos pesos se utilizan para inferir las ubicaciones y las clases de los objetos en las imágenes durante la detección de objetos.

3.Script de traking:

Para crear una representación visual de cómo funciona YOLOv8 como modelo de entrenamiento de aprendizaje automático, podríamos imaginar un proceso donde el modelo analiza imágenes para identificar y

clasificar objetos. Dado que YOLO (You Only Look Once) es conocido por su rapidez y precisión en la detección de objetos en tiempo real, el dibujo podría ilustrar las etapas de procesamiento de una imagen desde la entrada hasta la identificación de objetos con etiquetas y cuadros delimitadores.

Imaginemos un flujo donde una imagen entra en el sistema, se procesa a través de varias capas de la red neuronal convolucional (CNN) de YOLOv8, y luego emite la imagen original con cuadros delimitadores alrededor de los objetos detectados, cada uno con una etiqueta que indica la clase del objeto (por ejemplo, persona, coche, semáforo). También podemos incluir elementos visuales que representen la mejora en la precisión y velocidad de YOLOv8 en comparación con versiones anteriores, como YOLOv4 o YOLOv5.

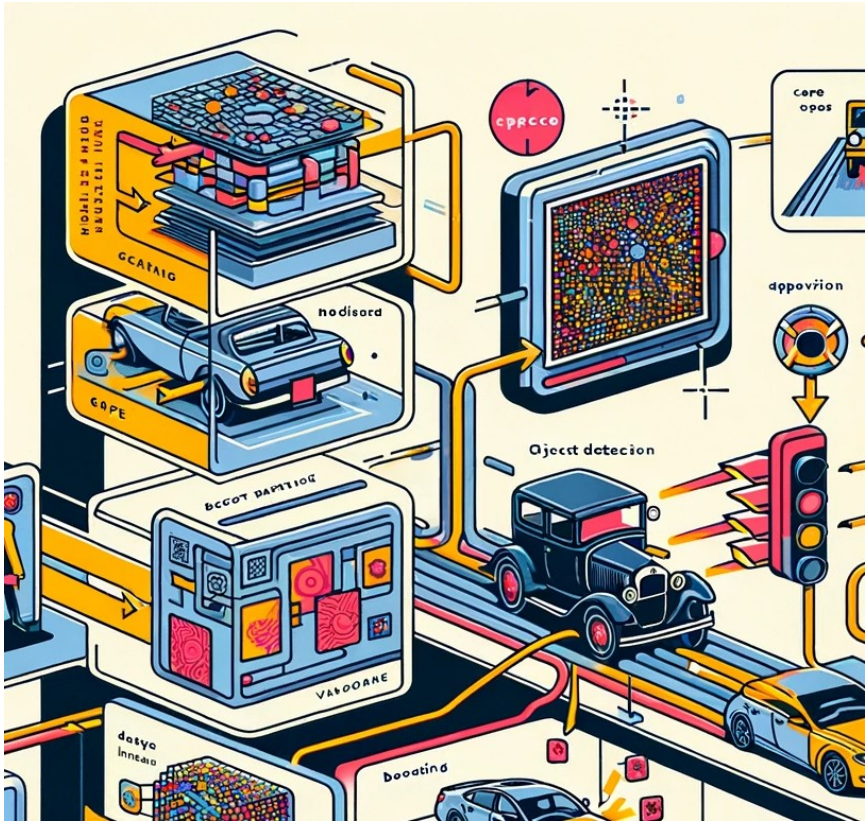


Figura 1:Representación de funcionamiento de machine learning

https://docs.ultralytics.com/es/modes/track/

```
from ultralytics import YOLO

# Configure the tracking parameters and run the tracker
model = YOLO('yolov8n.pt')
results = model.track(source="0", conf=0.3, iou=0.5, show=True)
```

Script 2

Este código utiliza el modelo YOLOv8 para realizar seguimiento de objetos en tiempo real utilizando la cámara del ordenador como fuente de video. Los objetos detectados se mostrarán en una ventana, y el seguimiento se ejecutará con los parámetros especificados.

Reto del aprendizaje :

https://docs.ultralytics.com/es/datasets/detect/coco/#sample-images-and-annotations

Enunciado:

El script antes propuesto anteriormente muestra el rastreo de que es una persona ,piensa que ha sido entrenad con la lista Coco.txt, como podrías cambiar este script para que rastrease otro objeto de dicha lista.

Objetivo :

Hacer un script similar al segundo ,que nos perimta elegir el objeto para realizar el tracking del script

Ayuda:

https://github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/datasets/coco.yaml

```
# Ultralytics YOLO 🚀, AGPL-3.0 license
# COCO 2017 dataset https://cocodataset.org by Microsoft
# Documentation: https://docs.ultralytics.com/datasets/detect/coco/
# Example usage: yolo train data=coco.yaml
# parent
# ├── ultralytics
# └── datasets
#     └── coco  ← downloads here (20.1 GB)

# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ..]
path: ../datasets/coco # dataset root dir
train: train2017.txt # train images (relative to 'path') 118287 images
val: val2017.txt # val images (relative to 'path') 5000 images
test: test-dev2017.txt # 20288 of 40670 images, submit to https://competitions.codalab.org/competitions/20794

# Classes
names:
  0: person
  1: bicycle
  2: car
  3: motorcycle
  4: airplane
  5: bus
  6: train
  7: truck
  8: boat
  9: traffic light
  10: fire hydrant
  11: stop sign
  12: parking meter
  13: bench
  14: bird
  15: cat
```

16: dog
17: horse
18: sheep
19: cow
20: elephant
21: bear
22: zebra
23: giraffe
24: backpack
25: umbrella
26: handbag
27: tie
28: suitcase
29: frisbee
30: skis
31: snowboard
32: sports ball
33: kite
34: baseball bat
35: baseball glove
36: skateboard
37: surfboard
38: tennis racket
39: bottle
40: wine glass
41: cup
42: fork
43: knife
44: spoon
45: bowl
46: banana
47: apple
48: sandwich
49: orange
50: broccoli
51: carrot
52: hot dog
53: pizza
54: donut

```
  55: cake
  56: chair
  57: couch
  58: potted plant
  59: bed
  60: dining table
  61: toilet
  62: tv
  63: laptop
  64: mouse
  65: remote
  66: keyboard
  67: cell phone
  68: microwave
  69: oven
  70: toaster
  71: sink
  72: refrigerator
  73: book
  74: clock
  75: vase
  76: scissors
  77: teddy bear
  78: hair drier
  79: toothbrush

# Download script/URL (optional)
download: |
  from ultralytics.utils.downloads import download
  from pathlib import Path

  # Download labels
  segments = True  # segment or box labels
  dir = Path(yaml['path'])  # dataset root dir
  url = 'https://github.com/ultralytics/yolov5/releases/download/v1.0/'
  urls = [url + ('coco2017labels-segments.zip' if segments else
'coco2017labels.zip')]  # labels
  download(urls, dir=dir.parent)
  # Download data
```

```
  urls = ['http://images.cocodataset.org/zips/train2017.zip',  # 19G, 118k images
          'http://images.cocodataset.org/zips/val2017.zip',  # 1G, 5k images
          'http://images.cocodataset.org/zips/test2017.zip']  # 7G, 41k images (optional)
  download(urls, dir=dir / 'images', threads=3)
```

Script 3