# **Visión por Computadora**

# Machine Learning

NOMBRE: Jarot Osiel Fierro Concha

CARRERA: Ingeniería en Informática

# Introducción

El objetivo principal de este proyecto es evaluar y comparar el desempeño de diferentes versiones del modelo YOLOv8 para tareas de detección de objetos, específicamente en un dataset de imágenes de mouses. El uso de modelos como YOLOv8 es fundamental en el ámbito del aprendizaje profundo debido a su capacidad para identificar objetos de manera eficiente y precisa. En este contexto, se seleccionaron dos versiones del modelo, una intermedia (YOLOv8m.pt) y otra avanzada (YOLOv8x.pt), para analizar sus diferencias en rendimiento y consumo de recursos. Además, este proceso resalta la importancia de una correcta configuración del dataset y de los parámetros de entrenamiento. Por último, el trabajo busca identificar ventajas y limitaciones de cada modelo para proponer mejoras en futuros proyectos similares.

POR DERECHO A LICENCIAS LAS IMÁGENES NO SE PUEDEN SUBIR, favor de seguir las indicaciones para hacerlo funcionar en caso de necesitarlo.

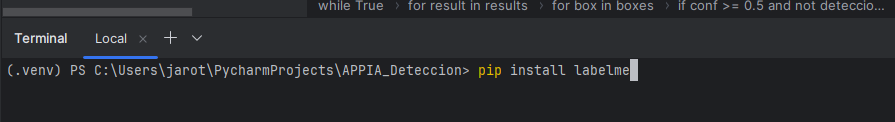
# Desarrollo

**Recolección de Imágenes**

El primer paso consistió en la recolección de imágenes a través de la extensión de Download All Images de chorme, después se realizó limpieza de las imágenes con formato extraño, para este proyecto se utilizaron solo 80 imágenes en total (60 para train y 20 para val).

Luego se creó un proyecto en pycharm con la versión de Python 3.8, es importante que sea esa.

**Etiquetado con Labelme**



Captura de pantalla de un celular

Descripción generada automáticamente

Se instala labelme (se upgradea pip), y etiquetan las imágenes y se le agregan las etiquetas necesarias

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica

Descripción generada automáticamente

El resultado son todas todas las etiquetas de las imágenes en formato .json

**Preparando los archivos**

Texto

Descripción generada automáticamente

Instalamos labelme2yolo para transformar los archivos json a txt y así poder entrenar el modelo

Interfaz de usuario gráfica, Texto

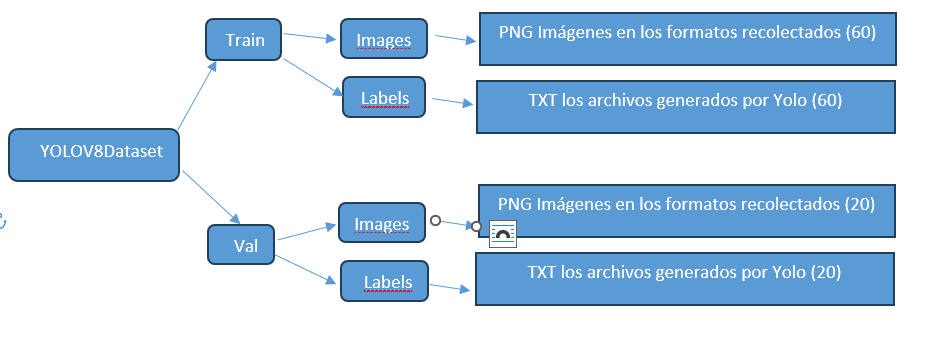
Descripción generada automáticamente

Indicamos la ruta de donde se encuentran las imágenes y los archivos txt, al ejecutar este comando se creará una carpeta llamada Yolov8Dataset

Forma, Logotipo, nombre de la empresa, Flecha

Descripción generada automáticamente

Dentro de la carpeta Yolo debe estar la siguiente estructura:

****

Los txt deben ser exclusivamente de las imágenes seleccionadas para cada carpeta, cada imagen tiene un txt especifico, para las 60 imágenes debe tener 60 archivos txt exactos de las 60 imágenes seleccionadas, así mismo para la otra carpeta.

**train**

Captura de pantalla de un celular

Descripción generada automáticamente

**val**

Captura de pantalla de un celular

Descripción generada automáticamente

**Configuración del Modelo**

El primer paso consistió en la configuración del archivo dataset.yaml, donde se definieron las rutas de los conjuntos de entrenamiento y validación.

Texto

Descripción generada automáticamente

Se seleccionaron dos versiones del modelo YOLOv8: YOLOv8m.pt (modelo intermedio) y YOLOv8x.pt (modelo avanzado). Esta elección se justificó considerando un balance entre complejidad y desempeño, así como la capacidad de evaluar cómo el modelo más avanzado aprovecha mejor los recursos disponibles.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Entrenamiento**

El entrenamiento se llevó a cabo en un proyecto desarrollado en PyCharm, donde se configuró un entorno con Python 3.8 y las bibliotecas necesarias, como Labelme y YOLO. Los archivos JSON generados por Labelme fueron convertidos a formato TXT compatible con YOLO. Para cada modelo, se ejecutó un entrenamiento de 5 épocas, especificando un tamaño de imagen adecuado (imgsz) y un tamaño de lote (batch) acorde al hardware disponible. La implementación se realizó en un equipo con CPU, lo que permitió optimizar los tiempos de procesamiento. Este proceso incluyó ajustes continuos para garantizar la estabilidad del entrenamiento y la obtención de resultados comparables entre las versiones del modelo.

Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Para el segundo modelo solo se cambia donde dice model a yolov8x.pt

**Comparación de Resultados**

Los resultados obtenidos evidenciaron diferencias significativas entre las versiones del modelo YOLOv8. El modelo intermedio (YOLOv8m.pt) mostró un buen desempeño con tiempos de entrenamiento más cortos y un menor uso de recursos. Por otro lado, el modelo avanzado (YOLOv8x.pt) generó problemas a la hora de entrenar el modelo, las causas fueron desconocidas ya que se empleó el mismo proceso para los dos, este segundo modelo presentaba problemas a la hora de identificar el objeto, captando cualquier cosa como “mouse”, además tuvo un mayor tiempo de entrenamiento y requerimientos de hardware, por lo que la neurona no aprendió a identificar la imagen, no se logró identificar el por qué, pero buscando en foros e internet una de las razones podría ser que el modelo YOLOx8x.pt al ser más preciso también necesita más imágenes y de más calidad y más pixeles.

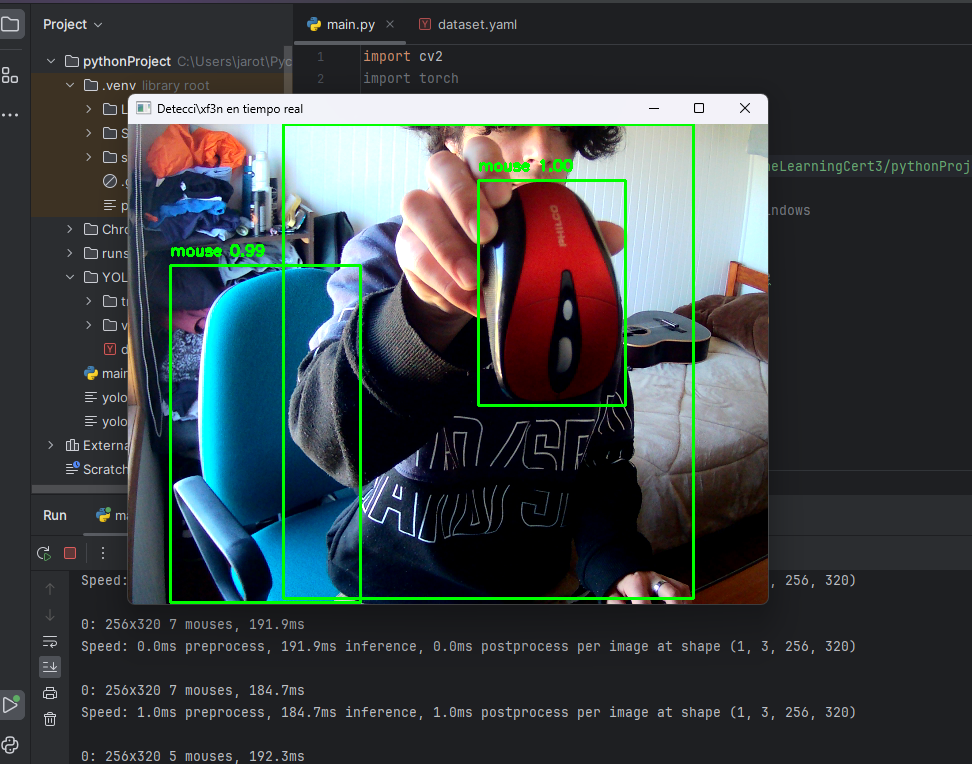
Esta comparación resaltó la relación directa entre la complejidad del modelo y su desempeño, así como la importancia de elegir el modelo adecuado en función de los recursos disponibles y los objetivos específicos del proyecto.

## Modelo 1: YOLOv8m.pt

Interfaz de usuario gráfica

Descripción generada automáticamente

## Modelo 2: TOLOv8x.pt



# Conclusiones

El análisis entre las versiones YOLOv8m.pt e YOLOv8x.pt reveló resultados inesperados. Aunque el modelo avanzado (YOLOv8x.pt) se esperaba que fuera más preciso, terminó mostrando un rendimiento inferior al modelo intermedio (YOLOv8m.pt). Esta discrepancia puede atribuirse a la cantidad limitada de imágenes y además investigando el modelo avanzado, hubo algunos sitios en donde recomendaban utilizar imágenes de resolución más grandes para el entrenamiento, lo que posiblemente afectó la capacidad del modelo avanzado para generalizar correctamente. Por otro lado, YOLOv8m.pt manejó de manera más eficiente los datos disponibles, logrando detectar los objetos de manera más precisa. Este caso resalta la importancia de contar con un dataset amplio y variado al trabajar con modelos más complejos.