

# **Lab4: Object detection (YOLO)**

3099704 AI for Digital Health (2025/2)



# Outline

- **Objective**
- **Material:** Google Colab, Ultralytics HUB, Ultralytics Libraries
- **Lab4.1: YOLOv8n (Ultralytics Hub)**
- Dataset: kvasir dataset (2017)
- Model: YOLOv8 (Ultralytics ,2023)
- Lab4.1: YOLOv8n (Ultralytics Hub) - 5 steps
- Results: Mean IoU = 0.8022
- **Lab4.2: YOLOv8s (Ultralytics)**
- Lab4.2: YOLOv8s (Ultralytics) - 4 steps
- Folder structure
- Results: Mean IoU = 0.8444
- Make it into a video!

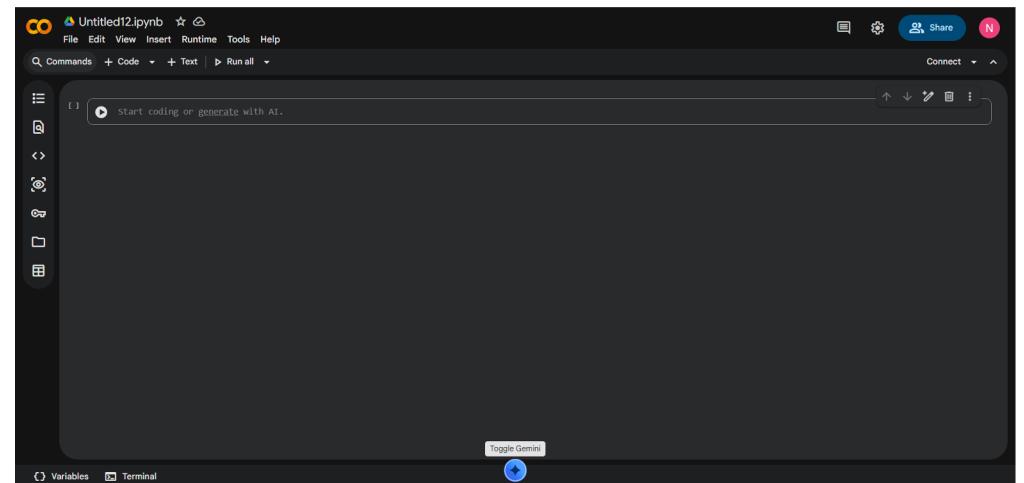
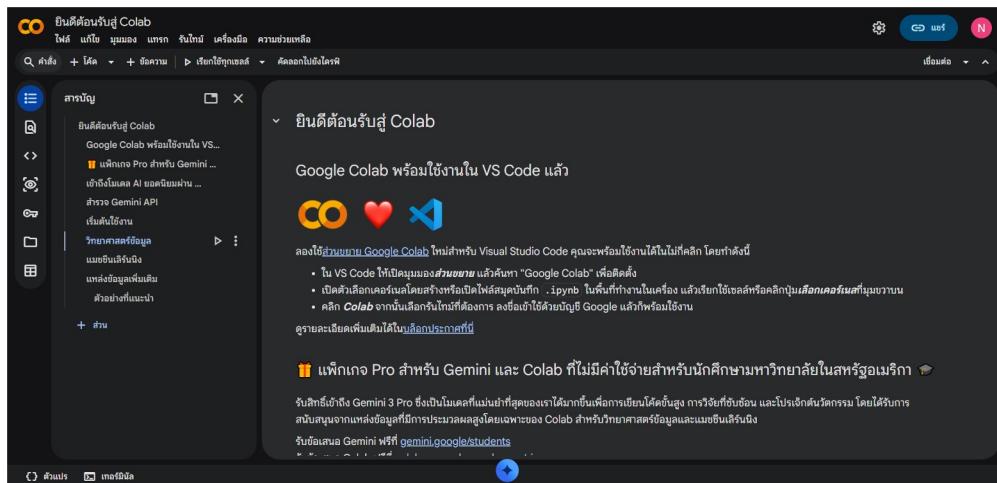
# Objective

- **Create & train** object detection model from ultralytics hub and ultralytics library
- **Inference & evaluate** performance each model



# Material

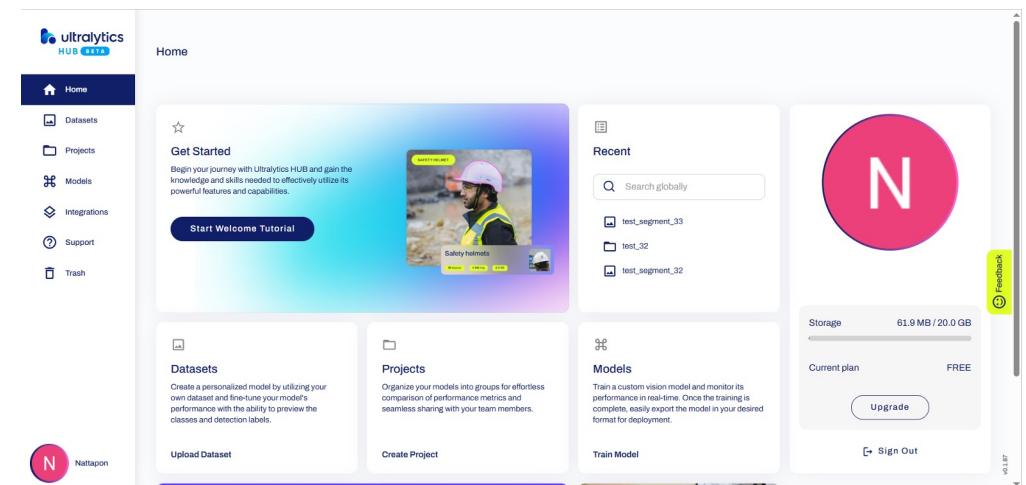
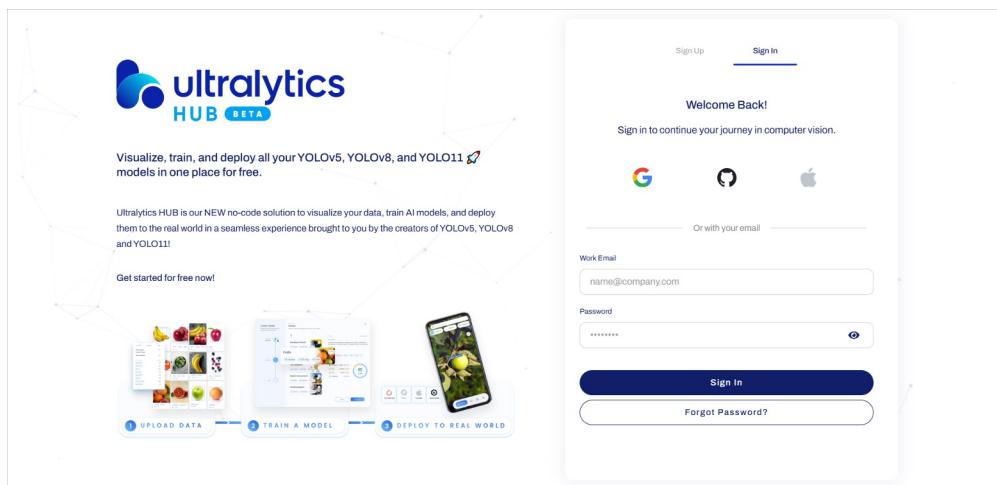
- With **Google Colab**, you don't need to install any software. All you need is a Google account, and you can start using it right away. Simply visit: <https://colab.research.google.com/> or select NEW NOTEBOOK to start a new file.



# Material



- **Ultralytics HUB** is a web-based platform that can be accessed via a browser at <https://hub.ultralytics.com/>, enabling users to upload datasets, train YOLO models, and monitor performance without any local software installation.



# Ultralytics HUB (limitation)



In this lab, we use the Ultralytics HUB free plan, which comes with the following limitations:

- (1) **Inference API** allows up to **100 calls per hour and 1,000 calls per month**
- (2) total dataset **storage** on Ultralytics HUB is limited to **20.0 GB**.



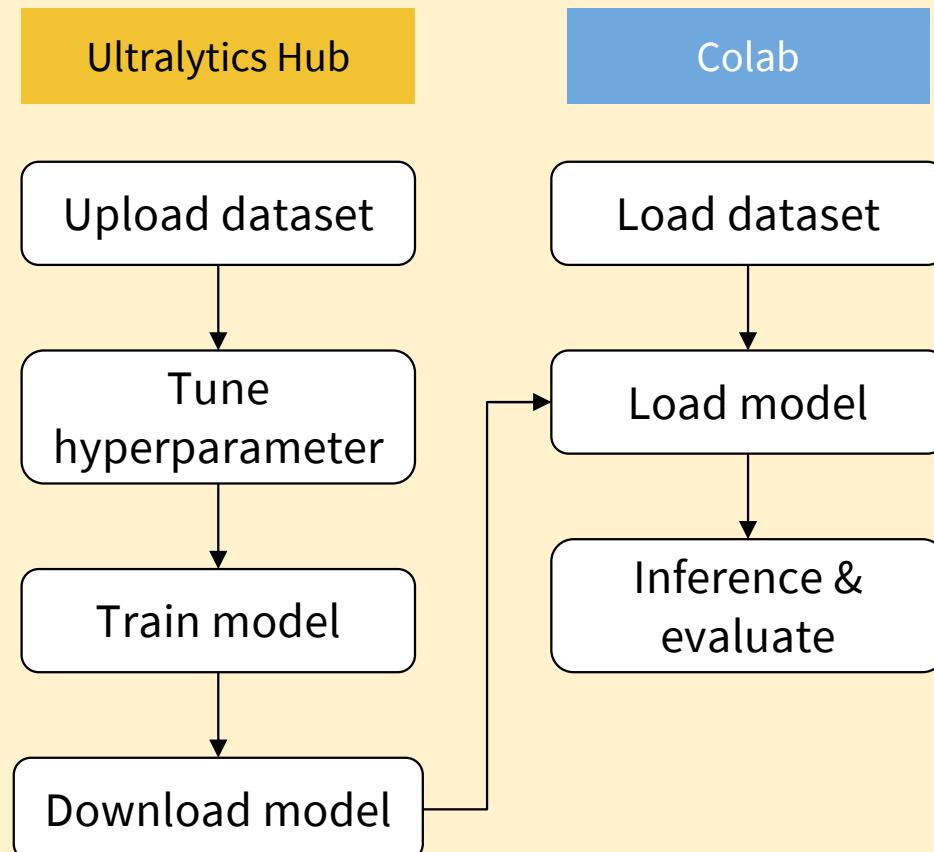
# Material



- **Ultralytics is a Python library for computer vision tasks**, best known for developing and training **YOLO (You Only Look Once)** models.
- It supports common tasks such as **image classification, object detection, and image segmentation**.
- Ultralytics models can be trained and used **on a local computer or on Google Colab**.
- Install the library using **pip**

# Lab4.1: YOLOv8n (Ultralytics Hub)

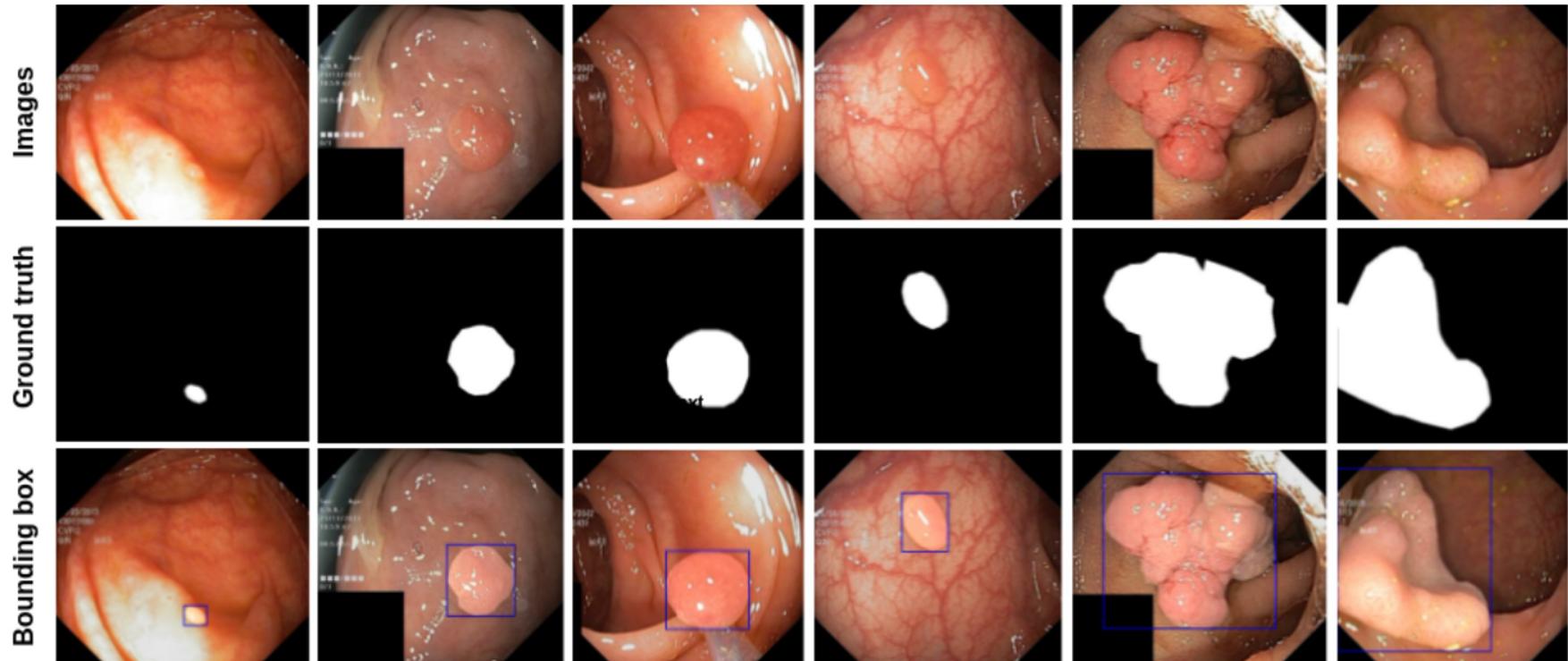
In this lab, you will create polyp detection model (**YOLOv8n**) using **Ultralytics HUB** and then evaluate its performance using a **Google Colab notebook**.



## Dataset: kvasir dataset (2017)

- **Kvasir dataset** consists of 4,000 images, annotated, including 8 classes showing anatomical landmarks, pathological findings or endoscopic procedures in the GI tract.
- The dataset consist of the images with different resolution from 720x576 up to 1920x1072 pixels in JPG format and documents in JSON format
- The dataset was released in 2017 by the **Simula Research Laboratory, Norway.**
- To simplify the experiment, we selected only **500 images** containing polyps and prepared the dataset in a format compatible with YOLO training.

# Dataset: kvasir dataset (2017)



The figure shows the example images, bounding box, and mask from Kvasir-SEG. The white mask shows the area covered by the polyp region, and the background regions contain non-polyp tissue pixels.

## **Model: YOLOv8 (Ultralytics ,2023)**

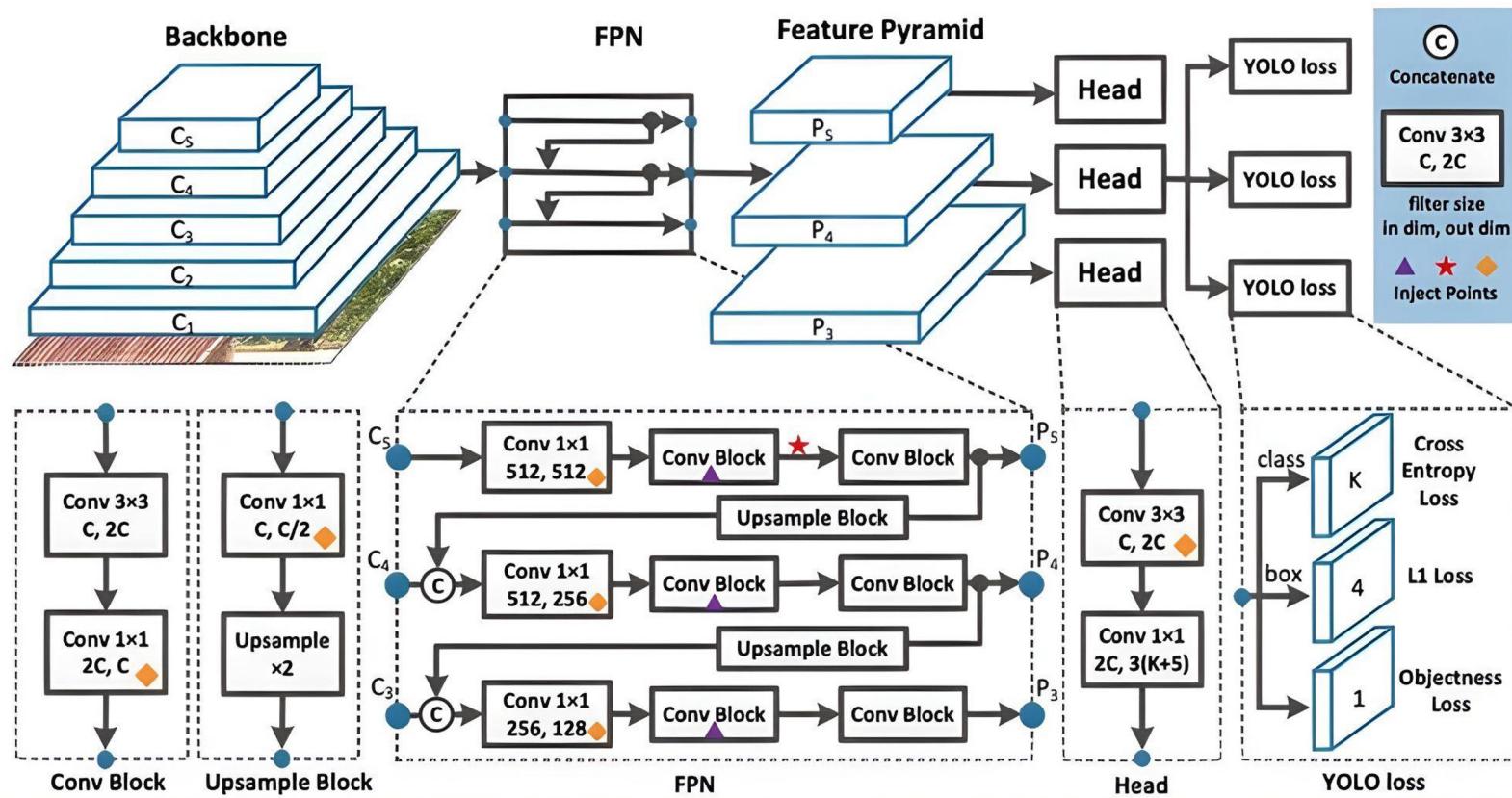
- YOLOv8 was released by Ultralytics on January 10, 2023, offering cutting-edge performance in terms of accuracy and speed.
- These models are designed for various tasks like instance segmentation, pose/keypoints detection, oriented object detection, and classification.

## Model: YOLOv8 (Ultralytics ,2023)

In this lab, we chose to use YOLOv8, which has the following key architectural innovations:

- **Backbone:** Uses CSPNet with improved **C2f modules** for efficient feature extraction and richer representations.
- **Neck:** Combines **FPN + PAN** to fuse multi-scale features, enabling detection of both small and large objects.
- **Head:** Anchor-free and **decoupled** design, separating classification and localization for higher accuracy.
- **Loss Functions:** Employs task-specific losses (e.g., BCE for classification, DFL for bounding box regression).

# Model: YOLOv8 (Ultralytics ,2023)



# Model: YOLOv8 (Ultralytics ,2023)

YOLOv8 is available in multiple model sizes: n (nano), s (small), m (medium), l (large), and x (extra-large). These models share the same overall architecture but differ in the number of channels (width) and repeated blocks (depth). Due to limitations, we use the **nano, small** version, which has the lowest computational cost.

Model	size (pixels)	mAP <sup>val</sup> 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

# **Lab4.1: YOLOv8n (Ultralytics Hub) - Steps**

- 1) Sign in Ultralytic hub
- 2) Download the dataset from GitHub, then upload in Ultralytic hub.
- 3) Create new project
- 4) Train Model (choose dataset)
- 5) Export model (Pytorch version)

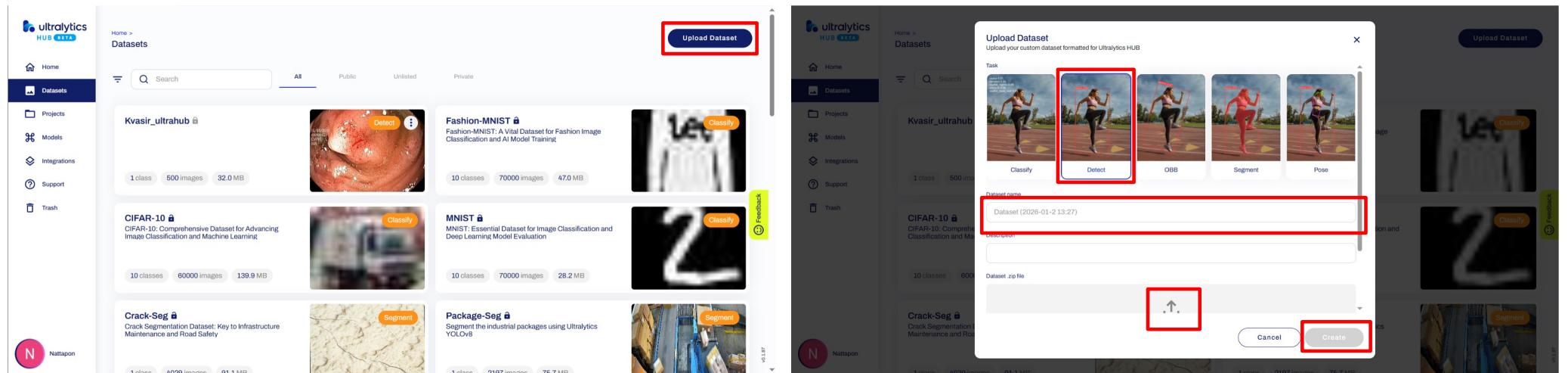
# Lab4.1: YOLOv8n (Ultralytics Hub)

## 1) Sign in Ultralytic hub

The image shows the Ultralytics HUB interface. On the left, there's a landing page with a 'Sign In' button highlighted with a red box. The main part shows the sign-in screen with fields for Work Email and Password, and options for Google or Apple sign-in. To the right is the Home dashboard, which includes sections for Get Started, Datasets, Projects, and Models. It also features a 'Recent' section with a message from 'Hello Nattapon!' and a 'Feedback' button.

# Lab4.1: YOLOv8n (Ultralytics Hub)

2) Download the dataset from [GitHub](#), then upload in **Ultralytic hub**.



# Lab4.1: YOLOv8n (Ultralytics Hub)

2) Download the dataset from [GitHub](#), then upload in **Ultralytic hub**.

train: images/train  
val: images/val  
test: images/test  
  
names:  
0: polyp

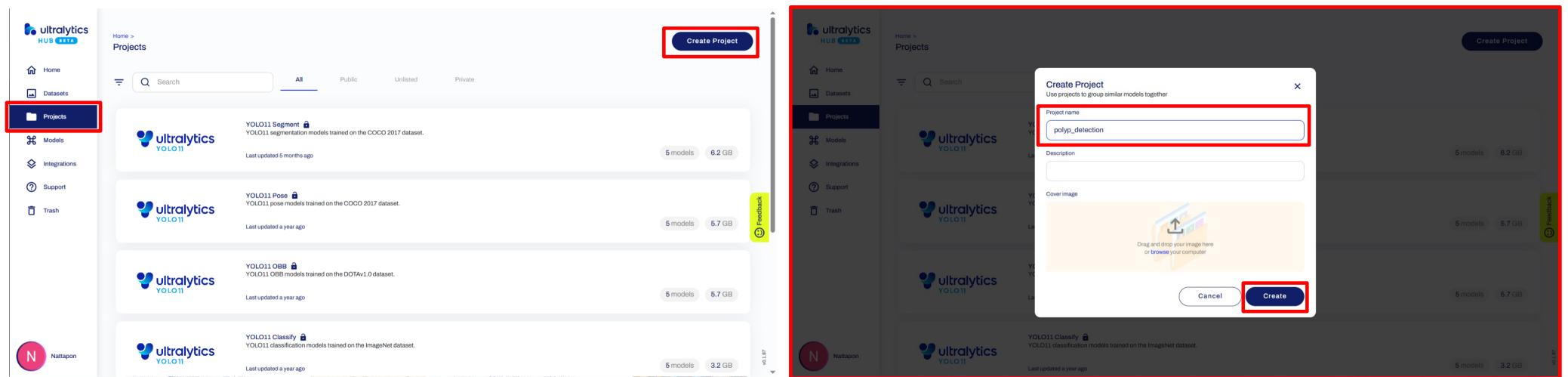
detect/  
  ├── images/  
  │   ├── train/  
  │   └── val/  
  ├── img1.jpg  
  ├── img2.jpg  
  └── test/  
      ├── img3.jpg  
      └── labels/  
          ├── train/  
          └── val/  
          └── test/  
            ├── img1.txt  
            └── img2.txt  
            └── img3.txt  
  └── data.yaml



[0 0.736334 0.502016 0.231511 0.322581]  
<class\_index> <x\_center> <y\_center> <width> <height>

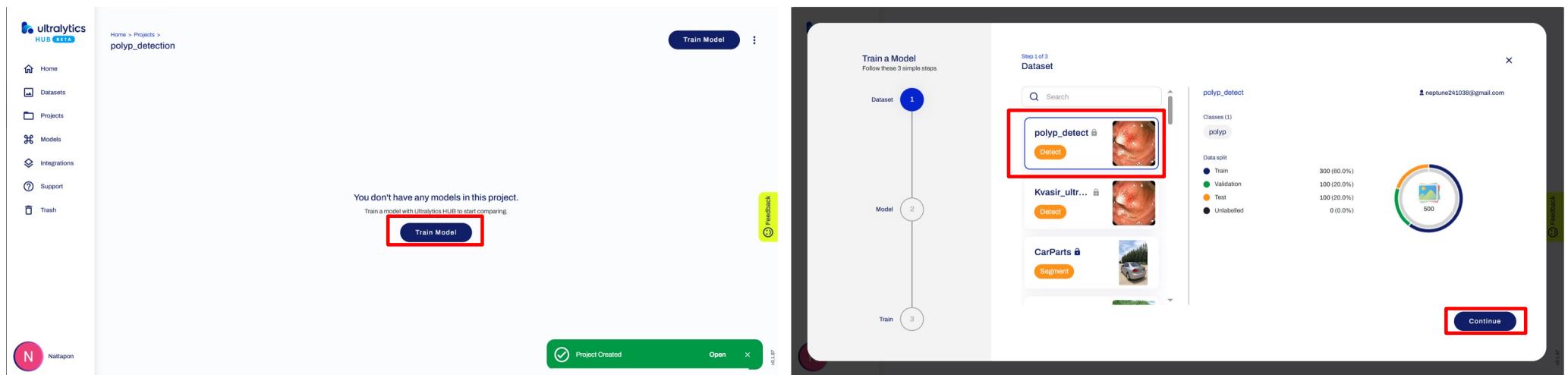
# Lab4.1: YOLOv8n (Ultralytics Hub)

## 3) Create new project



# Lab4.1: YOLOv8n (Ultralytics Hub)

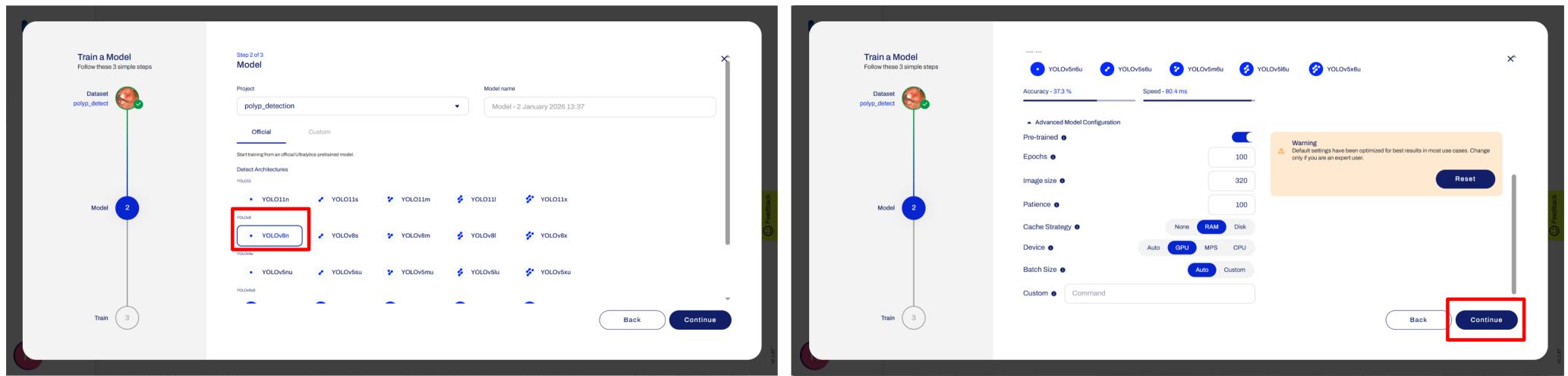
## 4) Train Model (choose dataset)



# Lab4.1: YOLOv8n (Ultralytics Hub)

## 4) Train Model (set configuration)

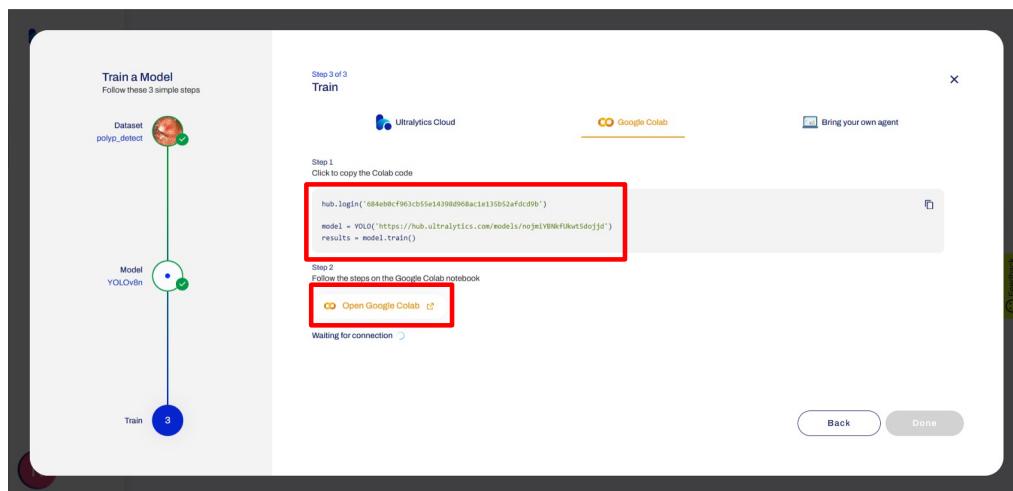
- model: YOLOv8n, Epochs: 100, Image size: 320, Cache: RAM, Device: GPU



# Lab4.1: YOLOv8n (Ultralytics Hub)

## 4) Train Model (training)

- Copy code and run in colab



The screenshot shows a Jupyter Notebook in Google Colab. The title bar says 'Ultralytics HUB'. The notebook has two sections:

- Setup:**  
Pip install ultralytics and dependencies and check software and hardware.  
pip 20.3.246 downloads 147M python 3.8 | 3.9 | 3.10 | 3.11 | 3.12  
Xpip install ultralytics  
from ultralytics import YOLO, checks, hub  
checks() # Verify system setup for ultralytics training  
ultralytics 8.3.99 Python-3.11.11 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)  
Setup complete (2 CPUs, 12.7 GB RAM, 39.6/112.6 GB disk)
- Start:**  
Login with your API key, load your YOLO model, and start training in 3 lines of code!  
hub.login('684eb0cf963cb55e14398d968ac1e135052afcd9b')  
# Load your model from HUB (replace 'YOUR\_MODEL\_ID' with your model ID)  
model = YOLO('https://hub.ultralytics.com/models/nojmiYBnfUkv5dojjd')  
# Train the model  
results = model.train()

# Lab4.1: YOLOv8n (Ultralytics Hub)

## 5) Export model (Pytorch version)

The image consists of two side-by-side screenshots of the Ultralytics Hub web application.

**Left Screenshot (Deployment):** This screenshot shows the 'Train' tab of a model configuration page. A red box highlights the 'Deploy' button at the top right of the configuration panel. Below it is a 'Metrics' chart showing validation set accuracy over training epochs. The chart includes four data series: mAP50(B) (dark blue), mAP50-95(B) (orange), precision(B) (cyan), and recall(B) (magenta).

**Right Screenshot (Export):** This screenshot shows the 'Export' section of the model details page. A red box highlights the 'PyTorch' option, which is listed as a 5.9 MB file. Other export options shown include TorchScript, ONNX, OpenVINO, TensorRT, CoreML, Sony's IMX300, TensorFlow SavedModel, TensorFlow GraphDef, TensorFlow Lite, TensorFlow Edge TPU, PyTorch Pegasus, and NCNN.

# Lab4.1: YOLOv8n (Ultralytics Hub)

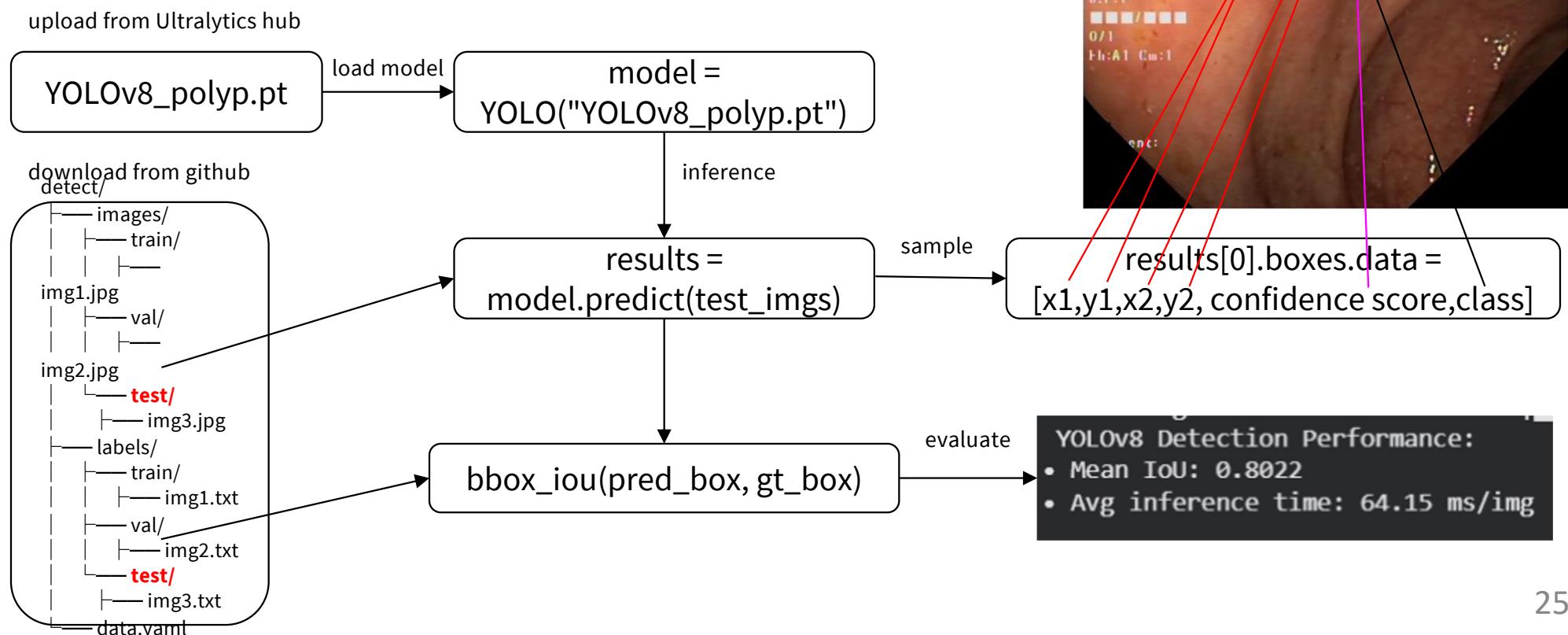
- 5) Open [Lab\\_4\\_1\\_ultralyricshub](#) in Colab, upload the pt. file exported from Ultralytics Hub, and then run the code in the notebook to evaluate the model's performance.

The screenshot shows a Google Colab interface with the following details:

- Title:** Lab\_4\_1\_ultralyricshub.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help, Share, Connect.
- Code Cell:** A code cell containing Python code for setup and library imports. It includes commands like `!pip install ultralytics opencv-python tqdm matplotlib -q` and imports for `os` and `shutil`.
- Text Content:** The notebook describes the purpose of evaluating polyp detection models and outlines a four-step process: Setup, Load Dataset, Load Model, and Inference & Evaluate. The first step, "1) Setup", is currently selected.
- Bottom Navigation:** Variables and Terminal tabs.

# Lab4.1: YOLOv8n (Ultralytics Hub)

## 5) Overview of [Lab\\_4\\_1\\_ultralyricshub](#)



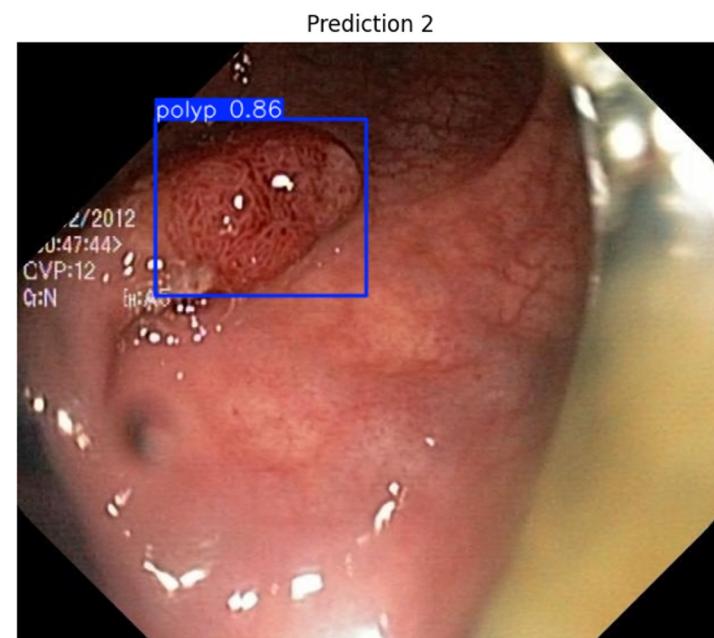
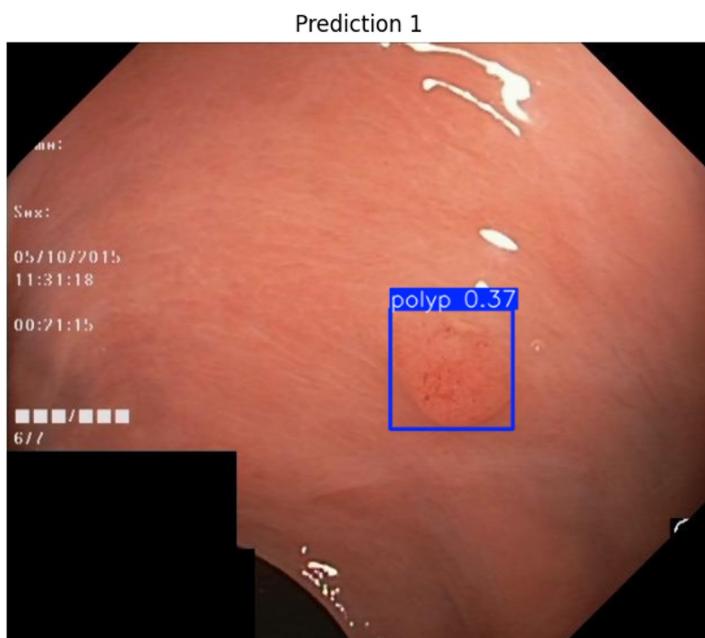
## Lab4.1: YOLOv8n (Ultralytics Hub)

Results may vary between runs due to random seed initialization and hyperparameter tuning; however, the overall performance should be similar to the results shown on this page.

```
Testing 100 test images...
Evaluating YOLOv8 Detection: 100%|██████████| 100/100 [00:06<00:09, 14.81it/s]
YOLOv8 Detection Performance:
• Mean IoU: 0.8022
• Avg inference time: 64.15 ms/img
```

# Lab4.1: YOLOv8n (Ultralytics Hub)

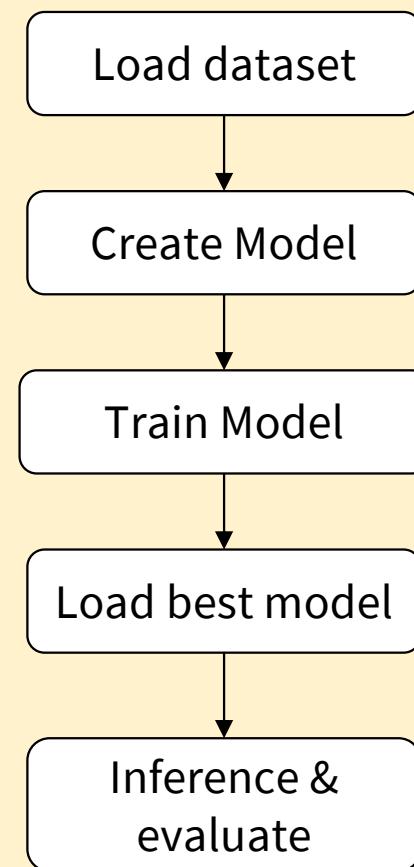
Results may vary between runs due to random seed initialization and hyperparameter tuning; however, the overall performance should be similar to the results shown on this page.



## Lab4.2: YOLOv8s (Ultralytics)

Colab

In this lab, you will create an polyp detection model (**YOLOv8s**) and evaluate its performance using **Ultralytics library** in colab



# Lab4.2: YOLOv8s (Ultralytics) - 4 steps

Run [Lab\\_4\\_2\\_ultralytics.ipynb](#) (in colab)

- 1) Setup
- 2) Load Data
- 3) Train model
- 4) Inference & Evaluate

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** Lab\_4\_2\_ultralytics.ipynb, Save in GitHub to keep changes, File, Edit, View, Insert, Runtime, Tools, Help.
- Toolbar:** Commands, + Code, + Text, Run all, Copy to Drive.
- Header:** Share, Connect 14, N.
- Section:** Lab 4.2 – Polyp detection: YOLO8 (Ultralytics)
- Description:** This notebook is used to evaluate the performance of polyp detection models trained on Ultralytics hub. This example code will consist of:

  1. Setup
  2. Load Dataset
  3. Train Model
  4. Inference & Evaluate

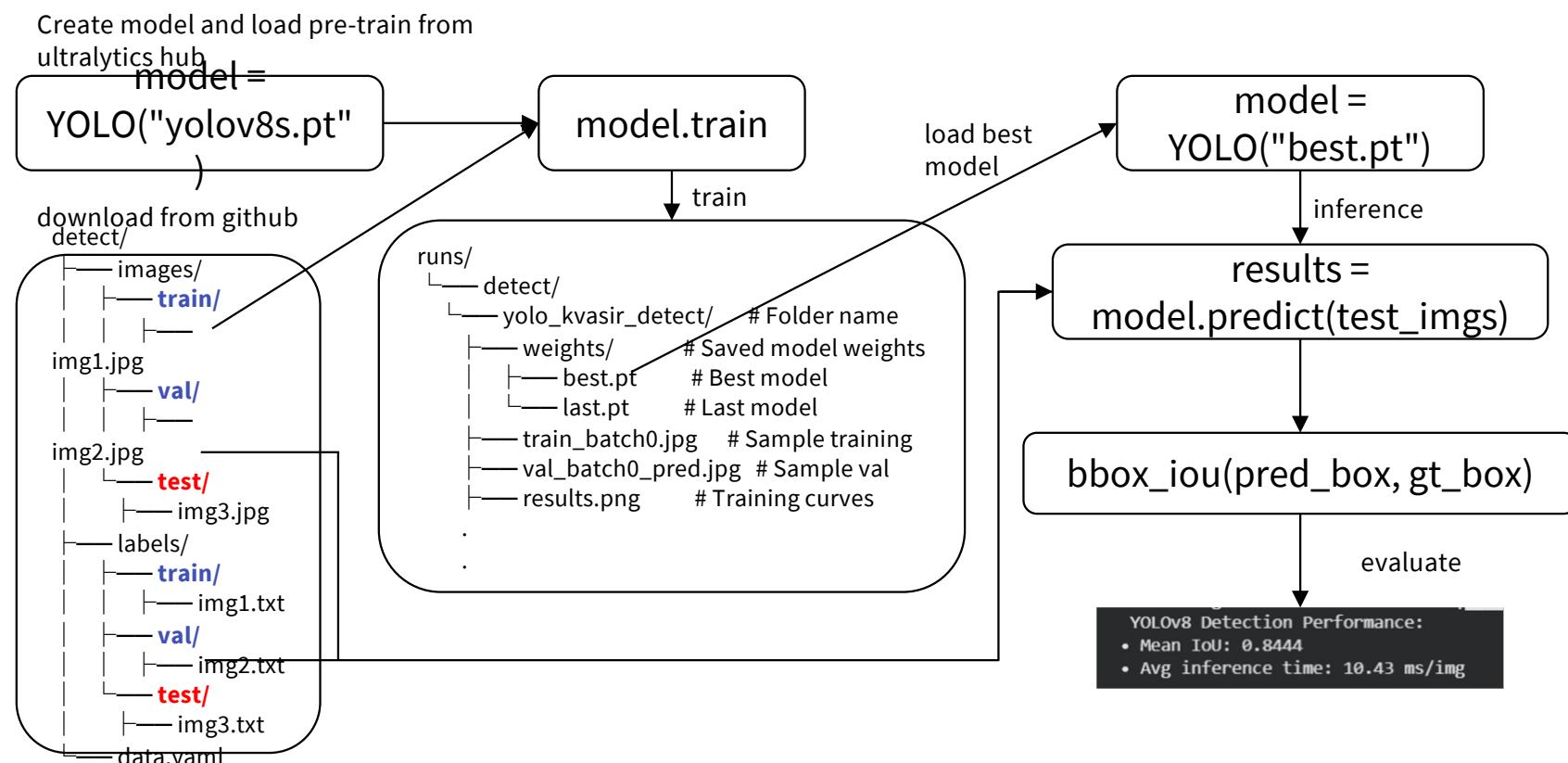
- Section 1: 1) Setup**

The code below install and import all required libraries and defines utility functions that will be used in the rest of this notebook.

```
# download library
!pip install ultralytics opencv-python tqdm matplotlib -q
...
# import library
import os
import shutil
```

# Lab4.2: YOLOv8s (Ultralytics)

Overview of [Lab\\_4\\_2\\_ultralytics.ipynb](#)



## Lab4.2: YOLOv8s (Ultralytics)

Results may vary between runs due to random seed initialization and hyperparameter tuning; however, the overall performance should be similar to the results shown on this page.

```
Testing 100 test images...
Evaluating YOLOv8 Detection: 100%|██████████| 100/100 [00:01<00:00, 89.09it/s]
YOLOv8 Detection Performance:
• Mean IoU: 0.8444
• Avg inference time: 10.43 ms/img
```

## Lab4.2: YOLOv8s (Ultralytics)

Results may vary between runs due to random seed initialization and hyperparameter tuning; however, the overall performance should be similar to the results shown on this page.

