**CODE :**

!nvidia-smi

import os

HOME = os.getcwd()

print(HOME)

!pip install -q git+https://github.com/THU-MIG/yolov10.git

!pip install -q supervision roboflow

!mkdir -p {HOME}/weights

!wget -P {HOME}/weights -q https://github.com/THU-MIG/yolov10/releases/download/v1.1/yolov10n.pt

!ls -lh {HOME}/weights

import os

os.environ["https://universe.roboflow.com/national-yang-ming-chiao-tung-university-jvm0d/cancer-cell-box"] = "4LwPEB7bW61h7s5xoSSU" # replace with your actual key

!mkdir {HOME}/datasets

%cd {HOME}/datasets

!pip install -q roboflow

from google.colab import userdata

from roboflow import Roboflow

import os

from dotenv import load\_dotenv, find\_dotenv

\_= load\_dotenv(find\_dotenv())

ROBOFLOW\_API\_KEY = os.environ["https://universe.roboflow.com/national-yang-ming-chiao-tung-university-jvm0d/cancer-cell-box"]

rf = Roboflow(api\_key=ROBOFLOW\_API\_KEY)

project = rf.workspace("national-yang-ming-chiao-tung-university-jvm0d").project("cancer-cell-box")

version = project.version(2)

dataset = version.download("yolov8")

%cd {HOME}

!yolo task=detect mode=train epochs=10 batch=32 plots=True \

model={HOME}/weights/yolov10n.pt \

data={dataset.location}/data.yaml

!yolo task=detect mode=val \

model=/content/runs/detect/train/weights/best.pt \

data=/content/datasets/Cancer-cell-box-2/data.yaml

import subprocess

# Run the validation command

result = subprocess.run(

[

"yolo", "task=detect", "mode=val",

"model=/content/runs/detect/train/weights/best.pt",

"data=/content/datasets/Cancer-cell-box-2/data.yaml"

],

capture\_output=True, text=True

)

# Parse mAP@50 from the validation output

output = result.stdout

map\_50 = None

for line in output.splitlines():

if line.strip().startswith("all"): # Match the 'all' line

columns = line.split()

try:

map\_50 = float(columns[5]) # mAP50 is in the 6th column (index 5)

except (IndexError, ValueError):

print("Error extracting mAP50.")

break

# Calculate accuracy

if map\_50 is not None:

accuracy = map\_50 \* 100

print(f"Accuracy: {accuracy:.2f}%")

else:

print("mAP@50 not found in the validation output.")

!ls {HOME}/runs/detect/train/

from IPython.display import Image, display

%cd {HOME}

Image(filename=f'{HOME}/runs/detect/train/confusion\_matrix.png', width=600)

%cd {HOME}

Image(filename=f'{HOME}/runs/detect/train/results.png', width=600)

from ultralytics import YOLOv10

import supervision as sv

model = YOLOv10(f'{HOME}/runs/detect/train/weights/best.pt')

dataset = sv.DetectionDataset.from\_yolo(

images\_directory\_path=f"{dataset.location}/valid/images",

annotations\_directory\_path=f"{dataset.location}/valid/labels",

data\_yaml\_path=f"{dataset.location}/data.yaml"

)

bounding\_box\_annotator = sv.BoundingBoxAnnotator()

label\_annotator = sv.LabelAnnotator()

import random

random\_image = random.choice(list(dataset.images.keys()))

random\_image = dataset.images[random\_image]

results = model(source=random\_image, conf=0.25)[0]

detections = sv.Detections.from\_ultralytics(results)

annotated\_image = bounding\_box\_annotator.annotate(

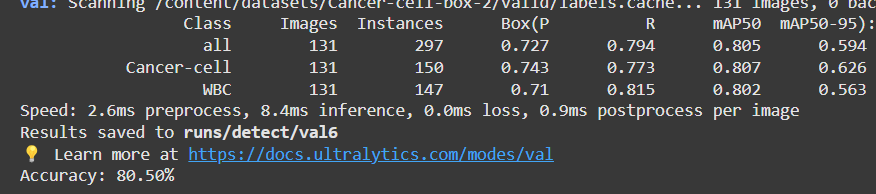
scene=random\_image, detections=detections)

annotated\_image = label\_annotator.annotate(

scene=annotated\_image, detections=detections)

sv.plot\_image(annotated\_image)

**OUTPUT :**

****

**REPORT :**

**Objective:**

The objective of this assignment is to train and evaluate a YOLOv10-based object detection model on a custom dataset containing images of cancer cells and white blood cells (WBC). The model aims to accurately detect and classify objects within these images, with a specific focus on evaluating the model’s performance using mAP (mean Average Precision) metrics at different IoU (Intersection over Union) thresholds.

**Data Preprocessing:**

The dataset used for training and evaluation consists of labeled images of cancer cells and WBC. The preprocessing steps involve:

* **Downloading Dataset:** The dataset is obtained from Roboflow and includes images and their corresponding labels.
* **Label Format Conversion:** The dataset is downloaded in YOLO format, with each image having a corresponding .txt label file containing object bounding boxes in YOLO format (class id, x\_center, y\_center, width, height).
* **Training/Validation Split:** The data is split into training and validation sets, with 131 images for validation.

**Model Architecture:**

The model used in this assignment is **YOLOv10**. The architecture of YOLO models typically consists of the following components:

* **Backbone:** A deep convolutional network that extracts feature maps from input images. YOLOv10 uses the CSPDarknet backbone.
* **Neck:** A PANet (Path Aggregation Network) that enhances feature representation by combining information from different layers of the backbone.
* **Head:** A detection head that produces predictions for bounding box coordinates, objectness scores, and class probabilities for each anchor box at multiple scales.

The hyperparameters for training the model are:

* **Model used:** yolov10n.pt (YOLOv10 nano variant)
* **Number of epochs:** 10
* **Batch size:** 32
* **Plots:** Enabled for visualization of training and validation results.
* **Learning rate and optimizer:** YOLOv10 uses the SGD optimizer by default with a warm-up phase and cosine learning rate scheduler.
* **Confidence threshold for predictions:** 0.25

**Training Details:**

The model was trained for 10 epochs with a batch size of 32 on the custom dataset containing images of cancer cells and WBC. The dataset was preprocessed into the required YOLO format, with bounding boxes and labels provided for each object in the images.

Training took place on a Tesla T4 GPU, providing fast processing due to its CUDA-accelerated operations. YOLOv10's efficient architecture ensures that the training process is relatively quick, even for complex datasets.

During training, plots were generated to monitor metrics such as loss and performance (mAP). The training focused on learning the best weights for detecting cancer cells and WBCs in images.

**Evaluation:**

The model's performance was evaluated on the validation dataset using the following metrics:

* **Precision (P):** Precision measures the fraction of relevant instances among the retrieved instances.
* **Recall (R):** Recall measures the fraction of relevant instances that have been retrieved over the total amount of relevant instances.
* **mAP@50 (mean Average Precision at IoU threshold 0.50):** This metric evaluates the model’s ability to detect objects with a high degree of overlap (IoU ≥ 50%).
* **mAP@50-95:** This evaluates the model over multiple IoU thresholds (from 0.5 to 0.95) to give a more robust indication of model accuracy.

**Results:**

The model achieved the following performance on the validation dataset:

* **Precision (P):** 0.727
* **Recall (R):** 0.794
* **mAP50 (mean Average Precision at IoU threshold 0.50):** 0.805
* **mAP50-95 (mean Average Precision over IoU range from 0.50 to 0.95):** 0.594

This indicates that the model is performing well, with high precision and recall, and a solid mAP50 score of 0.805. The mAP50-95 score indicates that while the model performs well at IoU = 0.50, it could be further improved when considering stricter overlap criteria.

**Accuracy Calculation:**

Based on the mAP50 score of 0.805, the accuracy was calculated as:

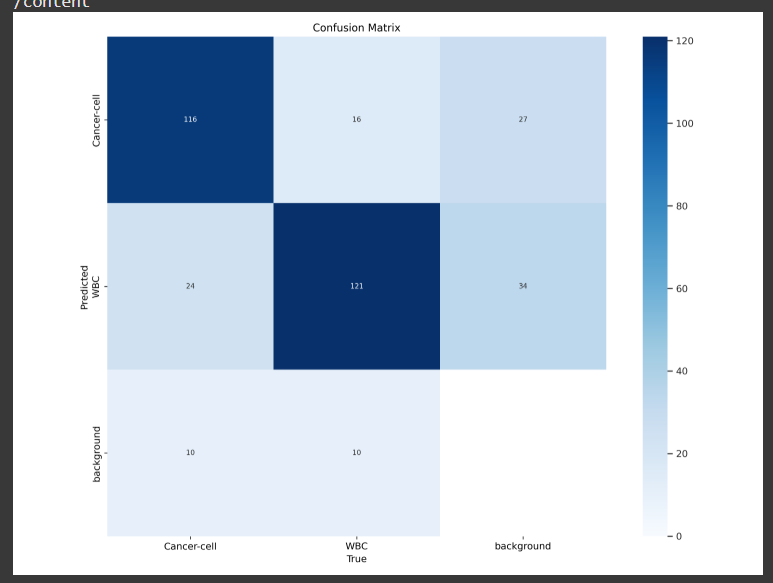
* **Accuracy:** 80.5%

**Insights and Conclusions:**

* The YOLOv10 model has demonstrated good performance on the validation dataset, with an mAP50 of 0.805, which suggests that the model can effectively detect objects within the images at the IoU threshold of 50%.
* While the model is performing well, further fine-tuning of hyperparameters, increasing training epochs, or using data augmentation techniques might improve performance, especially for stricter overlap thresholds.
* Visualizations such as confusion matrices and annotated predictions can help in further analyzing and improving model accuracy.

**Visual Outputs:**

* **Confusion Matrix:**



* **Results Visualization:**

