**TEAM–** NeuroWave.

**Documentation of how the Animal type detection Challenge code was run.**

### **Overview**

This documentation provides a step-by-step guide on how to run the YOLOv8 object detection model. The model was trained to detect specific animal types, and this guide will help you set up the environment, specify data paths, and execute the code.

### **Prerequisites**

Before running the code, one should ensure that they have the necessary Python environment set up with the required libraries. One can install all dependencies using the requirements.txt file provided.

Below are the requirements as listed in the requirements file;-

ultralytics==8.0.20

torch==2.0.1

numpy==1.23.0

pandas==1.5.0

matplotlib==3.6.0

scikit-learn==1.2.0

opencv-python==4.8.0

seaborn==0.12.0

pillow==9.5.0

To install all the requirements at once, one should run the command below;-

**!pip install -r requirements.txt**

**Specifying Data Paths**

**os.makedirs('animal\_type\_dataset', exist\_ok=True)**

**This code creates directory animal\_type\_dataset in your current working dataset if it does not exist.**

**!git clone -b animal\_type\_detection\_dataset https://github.com/MVet-Platform/M-Vet\_Hackathon24.git ./animal\_type\_dataset**

**This code clones the specified repository contents into the created animal\_type\_datset directory.**

**%cd /kaggle/working/animal\_type\_dataset**

**This makes the animal\_type dataset folder our current working directory.**

os.makedirs('yolo\_dataset/train/labels', exist\_ok=True)

os.makedirs('yolo\_dataset/train/images', exist\_ok=True)

This code creates directories yolo\_dataset as a sub directory in animal\_type\_datset directory as well as train directory as a sub directory to yolo\_dataset sub directory as well as labels and images subdirectories.

This is also done for the valid and test images.

Within the animal\_type\_dataset directory, the script expects three subdirectories:

* train: This directory should contain the training images and labels sud directories for the YOLO model.
* valid: This directory should contain the validation images and labels sub directories for the YOLO model.
* test: This directory should contain the test imagesfor which you want to predict animal types.

with open('yolo\_dataset/data.yaml', 'w') as file:

This code creates a data.yml file for yolo configuration under yolo\_dataset sub directory.

os.listdir("runs/detect/train/")

This creates a run sub directory with other subdirectories.

**Configured Wandb**

We configured Wandb in our project for better tracking of our machine learning experiments.

**Data Preparation**

-We downloaded the dataset by cloning it into our kaggle workspace.

-We proceeded with loading the dataset.

-We then did some data pre-processing that involved steps of proportion of the class, we searched for duplicates in the dataset and we actually removed the duplicates.

- We made a function that draws bounding boxes around the images for better image concentration. Then we converted the bounding box coordinates from VOC (Visual Object Classes) format to YOLO (You Only Look Once) format.

**Data Splitting**

* We split our dataset into train and validation sets. Our train set took 70% and the validation set took 30%.
* We defined a list of labels containing cow, pig and goat.
* We defined a function for oversampling the minority pig and goat classes which we got to know about by checking for class distributions.

**Model Training and Validation**

* We trained our model using a pre-trained YOLO version 8 medium model with hyperparameters as listed below;-
* epochs=25, imgsz=640, plots=True, lr0=0.01, lrf=0.01, batch=32, weight\_decay=0.0001, augment=True
* We applied techniques like augmentation, L2 Regularization using the weight\_decay parameter, setting initial and final learning rate, increasing batch size as well as increasing model complexity to yolov8 medium for improved performance.
* We then reviewed results collected during training. We proceeded with model validation and then model predictions.

**Model Testing**

* We carried out model predictions with reference to the unique image files provided in the sample\_submission.csv file.
* Here we specified our path to the test images as below;

results = model.predict(source='yolo\_dataset/test/images', conf=0.25, save=True, project=f"runs/detect", name="predict", exist\_ok=True)

-Finally we created our submission csv file that had columns;- filename, class, confidence, xmin, ymin, xmax, and ymax for evaluation.