Duality AI - Track for Code Veda Hackathon

Objectives

- **Train a robust object detection model using the provided synthetic dataset to accurately detect and classify space station objects — a critical capability for ensuring operational safety.**
- **Evaluate model performance across challenging scenarios including varied lighting, object angles, and occlusions.**

- **Benchmark and optimize your model to improve accuracy, generalizability, and efficiency for real-world deployment scenarios.**
- **(For bonus points) Create an application that uses this detection model, including how to keep the model up to date using Falcon.**

Hackathon Tasks

1. **AI Engineering**
    - **Train and fine-tune the YOLOv8 object detection model using the generated dataset.**
    - **Evaluate model performance.**
    - **Use optimization techniques to improve accuracy and reduce inference time.**

2. **Documentation G Presenting Final Team Results**

- **Document the workflow, including:**
  - **dataset generation**
  - **model training**
  - **evaluation metrics.**
- **Prepare a structured report and final presentation showcasing findings, results, and insights.**
- **Create visualizations such as confusion matrices and performance graphs to highlight model behavior.**

3. **(Bonus) Create an App**
   - **Create a computer or phone app that uses the model**
   - **Describe the plan for keeping the model up-to-date using Falcon**

Key Deliverables

1. **Trained Object Detection Model**
   a. **Fully trained YOLOv8 model that detects and classifies all three object categories.**

b. **The package must include model weights, scripts, and config files.**

2. **Performance Evaluation G Analysis Report, including but is not limited to:**

a. **mAP Scores (@0.5 IoU) to evaluate model accuracy.**

b. **Confusion Matrix to visualize class-wise performance.**

c. **Failure Case Analysis, highlighting misclassifications and possible improvements.**

Task Instructions

i. AI Engineering:

This section will walk you through:

- ➔ **Setting up your Falcon account**
- ➔ **Downloading and using the dataset**
- ➔ **Preparing your training environment**
- ➔ **Understanding the training workflow (Exercise 3)**
- ➔ **Establishing baseline performance with YOLOv8**

1. **Create a Falcon Account:**

a. **Visit Falcon and sign up for an account if you don't have one**

b. **Once registered, log in to access datasets, ɛxɛrcisɛs, and tools.**

2. **Download the Dataset**

a. **Download the dataset to your local machine from this pagɛ**

   i. **The page also contains useful links**

   ii. **The dataset includes:**

      - **Pre-collected images and YOLO- compatible labels**

         a. **Separated into Train, Val, and Test folders**

- **Sample Training and Prediction scripts**
- **Configuration files (config.yaml)**

3. **Set Up the Training Environment**
    a. **Make sure you have Anaconda installed, and open an Anaconda Prompt window**
    b. **Navigate to the ENV_SETUP sub folder in the Hackthon_Dataset folder**
    c. **Run the setup_env.bat file in the Anaconda Prompt window**
        i. **This will set up an environment called "EDU", containing all the dependencies required to run the training and prediction scripts.**

NOTE- Mac/Linux users, Create a setup_env.sh script with equivalent commands

4. **Understand the Training Workflow using Exercise 3**

    Falcon provides a guided walkthrough, explaining:
    a. **How the YOLOv8 training pipeline is structured**
    b. **What each training script does**
    c. **How to evaluate and visualize results**

5. **Train the YOLOv8 Model on the Sample Dataset**

    Once the EDU environment is ready and the dataset is in place:
    a. **Open an anaconda command prompt or a terminal**
    b. **Navigate to the training and prediction scripts directory**
    c. **Activate the environment by typing '*conda activate EDU*' in the terminal.**

d. **Run the training command: '*python train.py*'**

This will begin training your model and save logs + checkpoints to the runs/ directory.

6. **Establish Benchmark Results with YOLOv8 Model on the sample dataset After training is completed, evaluate your model's performance by running the prediction script (predict.py) in the same command prompt window. This tests its performance on real-world test images and gives you the following:**
   a. **mAP@0.5**
   b. **Precision and Recall**
   c. **Confusion Matrix**
   d. **Predictions**

Use the results as your benchmark, so that later, when you train with newer model settings, you can:

- **Compare performance**
- **Track improvements**
- **Identify where the model struggled (e.g., occlusion or lighting)**

Documentation G Presentation:

Ensure that your team's work is:

- ✔ **Clear and understandable**
- ✔ **Reproducible by others**
- ✔ **Professional and impactful for judges**

1. **Keep it Structured G Organized using this General Structure**

a. **Title G Summary: Clearly state the purpose of the document.**

b. **Step-by-Step Instructions: Use numbered lists or bullet points.**

c. **Diagrams G Visuals: Use flowcharts, tables, and screenshots from the training process and the visuals from the runs/ folder (created after training and prediction is complete)**

i. **Graphs G Charts: Show training loss, accuracy trends, and comparisons.**

ii. **Screenshots: Use images from the runs/ folder after training**

iii. **Confusion Matrix: Highlight model performance across different classes.**

iv. **Before G After Images: Show examples of correct vs. misclassified objects.**

2. **Document Everything, But Keep it Concise**

a. **Record major steps like dataset creation, training process, and evaluation.**

b. **Avoid overly technical language — aim for clarity.**

c. **Use clear, plain language—assume the reader is new to the project.**

3. **Example Entry:**

a. **Task: Model Training on Dataset**

b. **Initial mAP Score:** 42%

c. **Issue Faced: Low recall for "Oxygen Tank" class due to occlusion.**

d. **Solution: Augmented dataset with occlusion examples → New mAP Score: 55%**

4. **Documenting Failure Cases and Solutions**

a. **Include failure case images to illustrate what went wrong and how it was fixed.**

5. **Report Format (8 Pages Max)**

Your Report should be concise, structured, and visually engaging. Use the following storytelling approach:

Problem → Fix→ Results → Challenges → Future Work

Page.No	Section	Description
1	Title	Team name, project name, brief tagline.
2	Methodology	Steps taken while training the model, and fine-tuned results.
3-4	Results & Performance Metrics	mAP scores, confusion matrix, accuracy comparisons.
5-6	Challenges & Solutions	Key obstacles and how they were resolved. *See below for examples
7	Conclusion & Future Work	Final thoughts, and potential improvements.

3. Submission and Final Steps Instructions

i. Necessary Submission Files:

- **A single, Final Packaged Folder that includes all necessary files: ☐ Model training and inference scripts (train.py, predict.py) ☐ YOLO configuration files**

  ☐ runs/directory (containing training logs, outputs, and performance visualizations)

  ☐ Any additional assets or scripts required to test your model

- **A well-structured Hackathon Report (PDF or DOCX) that covers the following:**

  ☐ Methodology: Your training approach and setup

  ☐ Challenges & Solutions: Issues faced and how you overcame them ☐ Optimizations: Techniques used to improve model performance

  ☐ Performance Evaluation: mAP@0.5 score, Confusion matrix, Failure case analysis and observations.

- **A README.md or README.txt that provides:**

  ☐ Step-by-step instructions to run and test your model ☐ How to reproduce your final results

  ☐ Any environment or dependency requirements

  ☐ Notes on expected outputs and how to interpret them

- **(BONUS) A Use Case Application that covers the following:**

- ☐ **A .pdf or .docs file that describes what your application does, how you created the application, and a proposed plan for updating the model as the situation changes over time**
- ☐ **A video of the application working, or the application execution files/links**

Note: You are welcome to use your own object detection models and custom training scripts or notebooks. However, you must train your model exclusively on the dataset provided for this challenge.

Important: Using any of the designated testing images for training purposes is strictly prohibited and will result in disqualification. Please ensure a clear separation between training, validation, and test sets throughout your workflow.

ii. Upload Instructions:

1. **Ensure your submission folder contains all of the above**
2. **Compress everything into a .zip file.**
3. **Upload the zipped folder to a private GitHub repository of your own.**
4. **Complete the submission form (Hєrє) by:**

a. **Reporting your team's final score**

b. **Providing the GitHub repository link**

5. **Finally, add the following reviewers as collaborators:**

a. **Rishikesh Jadhav**

   i. **GitHub Username - Rishikesh-Jadhav**

b. **Syed Muhammad Maaz**

   i. **GitHub Username - Maazsyedm**

c. **Rebekah Bogdanoff**

   i. **GitHub Username - rebekah-bogdanoff**

After Submission:

1. **Sharing Results & Feedback**

a. **After submission, teams can showcase their models and insights.**

b. **Feedback from judges will be provided after evaluation.**

c. **A selected few teams will be invited for a post-hackathon discussion.**

2. **Future Opportunities & Improvements**

a. **Continue exploring advanced topics such as:**

   i. **Self-supervised learning**

   ii. **Domain adaptation**

   iii. **Multi-view detection**

b. **Consider using Falcon to generate your own synthetic data augmentation for better training.**

3. **Stay connected with us via Discord for:**

a. **Future challenges**

b. **Internship and apprenticeship opportunities**

c. **Community events and AI workshops**

NOTE: If you face any issues along the process join our discord channel linked above where we will be providing support to you for the hackathon.