Duality AI - Space Station Hackathon

1. Overview

Duality AI is excited to present the Space Station Hackathon, a challenge designed to test cutting-edge AI training techniques to train a model for Object Detection in a space station environment. Participants will train a model using a synthetic dataset from Duality AI's digital twin simulation platform- Falcon. Each team will work to achieve the most accurate and precise model by adjusting training parameters and processes. Along the way, participants will learn how Duality AI uses high-quality synthetic data to train AI models in unseen or difficult-to-access environments, such as a space station. Through industry-proven techniques and tools, we can train an AI model in tasks such as object detection using only synthetic data. Through this competition, participants will develop novel AI training skills, enhance their portfolio, build connections with Duality AI and other participants, and have the opportunity to win prizes and recognition!

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.

Importance of Digital Twins in Space Station Simulation

Digital twins are highly realistic virtual representations of real-world systems. This challenge uses data from a digital twin space station environment generated within Falcon. Digital twins, such as the space station, allow students, researchers, and AI enthusiasts to train a model in an environment that they otherwise might not be able to in the physical world - whether because it is difficult to access, not yet in existence, or limited in its availability. Using synthetic data from Falcon, users can:

- Simulate real-world conditions (e.g., lighting, occlusions).
- Train AI models without relying on real-world data collection.
- Enable reproducible and scalable experimentation.

Hackathon Structure

Data Overview

- Participants will work with a dataset generated from FalconEditor of a simulated space station scene featuring:
    - Varying lighting conditions
    - Occlusions and object overlap
    - Diverse camera angles and distances
- Target Object Categories:
    - Toolbox
    - Oxygen Tank
    - Fire Extinguisher

Participants will process synthetic data from the simulation, train an AI model to classify these objects, and optimize detection accuracy under realistic constraints.

i. Hackathon Tasks

To maximize efficiency, we recommend dividing responsibilities based on these roles. Proper delegation will help streamline the workflow and ensure high-quality results across the technical and presentation components.

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 & 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

ii. Key Deliverables

At the end of the hackathon, teams must submit:

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 & 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.

iii. Prizes and Rewards

We are excited to offer the following prizes and recognition to participants in the Duality Track - Space Station Hackathon:

- First Place: $200
- Second Place: $100
- Top 10 Teams: Participation Certificate

These prizes are awarded based on the performance of your object detection model, with a focus on accuracy and innovation. We encourage all teams to strive for excellence and showcase their skills in AI model training and optimization.

iv. Judging Criteria

Teams will be evaluated on a 100-point scale, based on the following criteria:

a. Model Performance

mAP@0.5 (Mean Average Precision at IoU=50%) .....80 Points

- Measures the accuracy of object detection and classification.
- Higher mAP scores indicate better model performance in detecting and classifying objects correctly.

b. Performance Report Clarity

Structured Findings & Detailed Reporting.....20 Points

- Well-organized documentation of the methodology, challenges, and solutions.
- Clearly outlined steps, including dataset creation, model training, and evaluation.

c. (Bonus) Use Case Proposal

Proposed Application .....(Bonus) 15 Points*

**Partial completion can still give you partial points. You can only score a maximum of 100 points for this Hackathon challenge, but you can earn 15 bonus points that will be added to your final score UP TO 100 total points.*

- Create an application that uses this model
- Include how Falcon can be used to keep the model up-to-date with the real world use

Teams must balance technical performance with clear and professional documentation to maximize their score.

---

2. 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, exercises, and tools.

2. **Download the Dataset**
    a. Download the dataset to your local machine from this page
        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)

---

ii. Documentation & Presentation:

Ensure that your team's work is:

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

1. **Keep it Structured & Organized using this General Structure**
    a. **Title & Summary:** Clearly state the purpose of the document.
    b. **Step-by-Step Instructions:** Use numbered lists or bullet points.
    c. **Diagrams & 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 & 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 & 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.

iii. (Bonus) Use Case Proposal:

This is an optional bonus task that can replace any of the required points. It can push the score up to, but not above, 100 total points.

1. **Create an application that uses this object detection model and runs either on your computer or phone**
    a. Show how this model could be used to address a real-world problem or need.
    b. Create a video of the application, or share the application files in your submission
2. **Include how Falcon can maintain the model**
    a. As part of the application, describe how Falcon could continuously be used to update the model as the real-world problem changes.
        i. For example: If the detected object changes its look or if a new object gets introduced to the environment that causes confusion for the model, how can Falcon be used to address the new changes?

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 (Here) 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

iii. 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.

---

4. Common Issues and Troubleshooting

FAQ's

1. **Will setup_env.bat Work on Both Windows & Mac?**
   - No, The setup_env.bat script is designed primarily for **Windows** environments.
   - Alternatively, for Mac/Linux use create a setup_env.sh shell script equivalent that installs all required packages.
2. **My training process is too slow. What can I do?**
   - Here are a few tips to improve training speed:
     i. Reduce the batch size in your training configuration.
     ii. Close any unused applications or background processes to free up system resources.
     iii. If using GPU, monitor GPU usage with tools like nvidia-smi.
3. **How should I manage data transfer between team members or for submission?**
   - Why backup your data:
     i. Share results between teammates

    ii. Backup project files and checkpoints
  - We recommend using cloud storage platforms such as Google Drive, Dropbox, OneDrive, git
4. **Where can I find the training and prediction scripts?**
  - Located inside the dataset folder: (train.py, predict.py)
5. **What if my model is not improving?**
  - For additional tips to improve performance refer Here

Support and Communication

1. **Join the official Duality Community Discord Server for:**
  a. Real-time help
  b. Announcements
  c. Live Q&A with organizers
  d. Invite Link Here

---

5. Glossary and Metric Benchmarks

Term	Definition
Digital Twin	A virtual replica of a real-world object or system.
Multi-Class Object Detection	Identifying and classifying multiple objects in an image/video.
mAP (Mean Average Precision)	Primary metric for evaluating object detection models.
IoU (Intersection over Union)	Measures how well a predicted bounding box overlaps with the ground truth.
Bounding Box	A rectangle defining an object's position in an image.

NMS (Non-Maximum Suppression)	A technique to remove duplicate detections of the same object.
Class Imbalance	When some object categories have significantly fewer samples than others.
Ground Truth	Manually labeled data specifying correct object locations and classes.
mAP (Mean Average Precision @ 0.5 IoU)	Measures how well the model detects objects (avg. over all classes Expected Benchmark: 40-50% (Baseline)
Precision & Recall	Precision = What percentage of the positive predictions are correct. Recall = What percentage of the total object present were correctly detected. Expected Benchmark: > 70% for best models
Training Loss	Lower loss = better training. If loss plateaus too high, the model may be underfitting. If the loss starts increasing, the model may be overfitting Expected Benchmark: Should steadily decrease
Inference Speed	Time taken to predict per image Expected Benchmark: < 50ms per image

We appreciate your hard work and look forward to reviewing your project! Feel free to share your work on LinkedIn and show the world your real-world problem solving skills in action!

Tag DualityAI to:

- Celebrate your team’s efforts and creativity.
- Get noticed by industry experts and recruiters

— The Duality AI Team