Egyptian Talents Academy

Giza Branch

Space Debris Detection Project

Submitted By:

Ramez Milad Zakaria

Ibrahim Wageh

Hassan Anees

Motawea Mohamed

Supervised by:

Dr. Mahmoud Farag

2024-2025

Space Debris Detection

Timeline

- **Introduction**
- **Dataset and Hyperparameters**
- **Features**
- **Libraries & Methods**
- **Accuracy & Loss**
- **Conclusion**
- **Project Problems**

Introduction

Space debris has become a growing problem in Earth's orbit over the past century. It includes broken satellite parts, old, unused satellites, discarded rocket stages, tiny paint chips, and even frozen fuel droplets. According to NASA, there are over 27,000 tracked pieces of debris in orbit, but millions of smaller, untracked fragments also exist. These objects travel at speeds of up to 28,000 km/h (17,500 mph), fast enough to damage or destroy active satellites and spacecraft.

Due to their high speeds, even small debris can damage satellites, leading to chain reactions of collisions. The diverse shapes and sizes of space debris make detection challenging using conventional methods.

This project explores how deep learning can help track and detect space debris more effectively. Small satellites equipped with cameras and sensors could be used to locate and monitor debris, improving space safety. Better tracking would help space agencies predict potential collisions and protect satellites. In the future, AI-powered systems could even assist in capturing and removing space junk to keep Earth's orbit clear and safe.

Dataset

The dataset contains images of space with space debris. The images are of size 256x256 in JPG format. The bounding boxes are in "bloxes" with the columns as "ImageID" and "bloxes" containing a list in [xmin, xmax, ymin, ymax] format.

Sample Row

Copy

```
ImageID  bloxes

3824092  [[134,65,69,98],[144,172,266,295]]
```

The boxes are in string format but can be converted into a Python list using the literal_eval function from the ast Python library.

Data Splitting

Train

- **20000 samples**: This CSV file contains the ImageID column and the bloxes column, which contains bounding boxes in a list.

Val

- **2000 samples**: This CSV file contains the ImageID column and the bloxes column, which contains bounding boxes in a list.

Test

- **5000 samples**: This ZIP file contains space debris images used to evaluate the model's performance.

Libraries

Main Libraries

- **Detectron2**: Object detection framework for Faster R-CNN.
- **OpenCV**: Image processing, loading images, and drawing bounding boxes.
- **PyTorch**: Deep learning computations and model execution.
- **NumPy**: Numerical computations.
- **Matplotlib**: Visualization.
- **TensorBoard**: Visualizing training loss, accuracy, and performance.

Detectron2 Library

Detectron2, developed by Facebook, is the main library used in this project. It is built on PyTorch and provides pre-trained models for tasks like object detection, instance segmentation, keypoint detection, and panoptic segmentation. Detectron2 is optimized for speed, flexibility, and scalability, making it ideal for training custom object detection models with datasets in COCO format.

Key Features of Detectron2

- Provides pre-trained models for transfer learning.
- Uses GPU acceleration for fast training and inference.
- Supports custom datasets and COCO-style annotations.

- Supports the Faster R-CNN algorithm for lower training time and higher accuracy.

Methods

R-CNN Model

- **R-CNN (Regional Convolutional Neural Network)** is a deep learning model for object detection that uses a two-step process:
    1. **Region Proposal**: Selects potential object locations using Selective Search and proposes them as regions.
    2. **CNN Feature Extraction & Classification**: Uses a Convolutional Neural Network (CNN) to classify each proposed region and refine bounding boxes.

R-CNN: Regions with CNN Features

1. **Input image**
2. **Extract region proposals (~2k)**
3. **Compute CNN features**
4. **Classify regions**

Hyperparameters

Training Hyperparameters

- **Batch size per iteration**: Number of images processed at once.
- **Initial learning rate**: Controls step size for weight updates.
- **Total training iterations**: Defines how long the model trains.
- **Model backbone**: ResNet-50.
- **Model depth**: Number of hidden layers in the network (50 in this case).

Accuracy & Loss

Accuracy

Detectron2 evaluates model performance using COCO evaluation metrics, which include:

- **mAP (Mean Average Precision)**: Measures overall accuracy across different Intersection over Union (IoU) thresholds (strictest metric).
- **AP@50 (AP at IoU > 50%)**: Measures accuracy when predictions overlap at least 50% with ground truth.

Loss

The loss is shown to be **0.06**, indicating good model performance.

Problems Faced

- **Detectron2 Library**: The main library used in this project (Detectron2) could not run locally and was only functional on the cloud (Google Colab).
- **Real-time Detection**: We could not deploy real-time detection for the project as cloud platforms do not support it. Instead, we wrote a function to take a photo and predict on it.

Conclusion

In this project, we developed a space debris detection system using Faster R-CNN and Detectron2, demonstrating how AI can help identify and track debris to prevent potential collisions. Our model continuously improved, as reflected in the decreasing classification loss. AI-driven solutions like this are crucial for ensuring safer space operations, protecting satellites, and contributing to long-term sustainability in space exploration.

Thank You