**Project Report: SpaceDebris Detection Application**

**Executive Summary**

The SpaceDebris detection application is designed to address the critical need for identifying space debris in satellite imagery, a growing concern for space agencies and satellite operators due to the increasing risk of collisions in orbit. This project leverages machine learning to classify images as containing debris or not, utilizing a convolutional neural network (CNN) trained on a custom dataset. The application serves as a proof-of-concept, showcasing skills in Platform as a Service (PaaS) deployment, API development and consumption, image processing, and integration with Google Earth Engine. Deployed on Heroku for the backend API and Netlify for the frontend React web interface, the application allows users to upload images and receive predictions, demonstrating a scalable and accessible solution. This Phase 1 completion marks a solid foundation, with plans to enhance the model and expand website functionality in future phases.

**Project Development Phase 1**

**Step 1: Project Conception and Technology Selection**

* **Objective**: The project began with the decision to create a CNN to classify satellite images for space debris detection, addressing the challenge of limited real debris imagery.
* **Milestone**: Selected Python with TensorFlow for model development, Flask for the API, and React for the web interface, integrating Google Earth Engine for image data acquisition.
* **Key Activities**: Identified the use case of debris detection, researched suitable technologies, and outlined a development plan leveraging PaaS platforms (Heroku and Netlify) for deployment.

**Step 2: Data Acquisition with Google Earth Engine**

* **Objective**: Gather satellite imagery to form the basis of the training dataset.
* **Milestone**: Developed code to interface with the Google Earth Engine API, enabling the download of satellite images suitable for debris detection.
* **Key Activities**: Authenticated with Google Earth Engine, wrote scripts to fetch and preprocess raw satellite data, and ensured compliance with API usage limits. This step provided the initial dataset, 888 images, for model training.

**Step 3: Image Modification and Dataset Creation**

* **Objective**: Simulate debris in images due to the scarcity of real debris imagery.
* **Milestone**: Created the add\_debris.py script to modify 50% of the downloaded images, adding simulated debris using labels.csv to insure it matched during training, and compiled a balanced dataset.
* **Key Activities**: Manually adjusted images to mimic debris patterns, validated the modifications, and organized the dataset into training and testing sets (e.g., X\_train.npy, y\_train.npy).

**Step 4: Image Preprocessing and Model Training**

* **Objective**: Prepare the dataset and train a CNN to classify images.
* **Milestone**: Developed preprocessing scripts and the train\_model.py script to train the model, saving it as debris\_classifier.h5.
* **Key Activities**: Preprocessed images to a uniform 128x128 resolution, applied mild data augmentation (e.g., rotation, zoom), and trained the CNN with early stopping and class weighting (1.0 for no debris, 1.9 for debris). Evaluated the model with metrics like accuracy, precision, recall, and F1 score, achieving a functional classifier.

**Step 5: API Development and Encapsulation**

* **Objective**: Encapsulate the trained model in a RESTful API for external access.
* **Milestone**: Created src/api.py using Flask, integrating lazy model loading with a background thread and CORS support.
* **Key Activities**: Implemented image preprocessing, prediction logic, and error handling. Tested locally with curl commands, ensuring the API returned JSON responses (e.g., {"probability": 0.75, "class": 1, "label": "Debris"}). Addressed initial deployment issues (e.g., ValueError, port mismatch) by refining the code and using tensorflow-cpu to reduce slug size.

**Step 6: Web Frontend Development**

* **Objective**: Develop a React-based web interface to utilize the API.
* **Milestone**: Built the frontend directory with SpaceDebrisClassifier.js, enabling image uploads and displaying predictions.
* **Key Activities**: Created a user interface with file input, integrated Axios for API calls to Heroku, and styled with Tailwind CSS. Validated functionality locally with npm start and prepared the build folder for deployment.

**Step 7: Deployment to Heroku and Netlify**

* **Objective**: Deploy the API and web app to PaaS platforms for public access.
* **Milestone**: Successfully deployed the Flask API on Heroku and the React app on Netlify, with the site (https://spacedebris.netlify.app/).
* **Key Activities**: Configured Heroku with Procfile and requirements.txt, resolved slug size issues (e.g., 570.1M to 316.4M) using Git LFS and tensorflow-cpu, and fixed port binding by using $PORT. Installed Netlify CLI, created a new site with netlify sites:create, and deployed the build folder. Tested end-to-end by submitting images and receiving predictions, confirming a solid Phase 1 completion.

**Additional Milestones from Chat History**

* **Environment Migration**: Transitioned development from an ARM to an Intel Windows machine, reinstalling dependencies and resolving version mismatches (e.g., Python 3.11.9 alignment).
* **Debugging and Optimization**: Addressed model loading errors (ValueError), boot timeouts (R10), and dependency warnings, showcasing iterative problem-solving.
* **Tooling**: Utilized Git for versioning and deployment, Visual Studio Code for development, and Git Bash for command-line operations, enhancing workflow efficiency.

**Conclusion**

Phase 1 of the SpaceDebris detection application is successfully completed, delivering a functional prototype that demonstrates PaaS deployment, API development, machine learning model creation, and web integration. The Heroku-hosted API and Netlify-deployed React site provide a scalable platform for debris classification, leveraging Google Earth Engine data and a custom-trained CNN. Future phases will focus on enhancing the model (e.g., improving accuracy with more data) and expanding website functionality (e.g., user accounts, advanced visualization). This project highlights my skills in full-stack development, deployment optimization, and innovative use of satellite imagery.