Group Name - CodeVelocity
Project Name - Offroad Semantic Scene Segmentation

Summary-

Offroad Semantic Scene Segmentation Using Synthetic Data

In this project, we developed a semantic segmentation model to classify desert scene elements at the pixel level using a synthetic dataset generated from Falcon's digital twin platform. The goal was to train a robust computer vision model capable of accurately identifying terrain and environmental features in off-road desert environments.

The model was trained to segment the following classes:

- **Trees**
- **Lush Bushes**
- **Dry Grass**
- **Dry Bushes**
- **Ground Clutter**
- **Flowers**
- **Logs**
- **Rocks**
- **Landscape**
- **Sky**

Each pixel in an image is classified into one of these categories, enabling fine-grained scene understanding. This type of segmentation is essential for off-road autonomy systems, where unmanned ground vehicles must understand terrain composition for navigation and obstacle avoidance.

We trained the model using the provided synthetic training dataset and evaluated performance on unseen desert images to test generalization. The primary evaluation metric was Intersection over Union (IoU), which measures how well the predicted segmentation overlaps with the ground truth masks.

Throughout the project, we:

- **Established a baseline model**
- **Tuned training parameters such as epochs and learning rate**
- **Evaluated performance using IoU and loss metrics**
- **Analyzed failure cases to understand model limitations**
- **Documented improvements through controlled experimentation**

The final model demonstrates improved segmentation performance through systematic optimization and structured experimentation, while maintaining reproducibility and clarity in documentation.

Methodology

1. Dataset Preparation

We used the synthetic desert dataset generated from Falcon's digital twin platform. The dataset consists of RGB images and their corresponding pixel-wise segmentation masks. Each mask contains labeled regions corresponding to predefined desert scene classes such as trees, rocks, sky, dry grass, and other terrain elements.

The dataset was organized into training and validation subsets. Before training, we verified the folder structure to ensure correct loading of image–mask pairs.

2. Data Preprocessing

To improve model training stability and convergence, the following preprocessing steps were applied:

- **Resizing images to a fixed resolution**
- **Normalizing pixel values to standard ranges**
- **Converting segmentation masks into numerical class labels**
- **Applying tensor transformations for model compatibility**

This ensured consistency in input dimensions and reduced computational overhead during training.

3. Model Architecture

We implemented a semantic segmentation architecture suitable for pixel-level classification. The model follows an encoder–decoder structure:

- **The encoder extracts hierarchical feature representations from the input image.**

- **The decoder upsamples the extracted features to produce a segmentation mask matching the original image resolution.**

The final output layer produces class probabilities for each pixel across all desert scene categories.

4. Training Procedure

The model was trained using supervised learning, where input images were paired with ground truth masks.

Key training parameters included:

- **Loss Function: Cross-Entropy Loss (for multi-class segmentation)**
- **Optimizer: Adam optimizer**
- **Learning Rate: Tuned experimentally**
- **Epochs: Multiple training cycles over the dataset**
- **Batch Size: Adjusted based on hardware capability**

During each epoch:

1. **The model predicted segmentation masks.**
2. **The loss between prediction and ground truth was computed.**
3. **Backpropagation updated model weights.**
4. **Performance metrics were recorded.**

5. Evaluation Metric

The primary evaluation metric was Intersection over Union (IoU).

IoU measures the overlap between the predicted segmentation and the actual ground truth mask for each class. It is calculated as:

IoU = (Area of Overlap) / (Area of Union)

Higher IoU indicates better segmentation accuracy.

We monitored:

- **Per-class IoU**
- **Mean IoU (mIoU)**
- **Training and validation loss curves**

6. Model Optimization

To improve performance, we experimented with:

- **Increasing number of training epochs**
- **Learning rate adjustments**
- **Data normalization improvements**
- **Hyperparameter tuning**

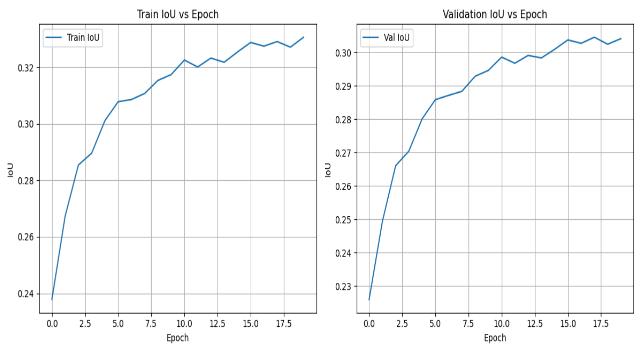
Performance improvements were validated using IoU comparisons across experiments.

7. Hardware and Environment

The model was trained using a Python-based deep learning framework within a local development environment. Training was executed on available CPU/GPU resources depending on system compatibility.

Results -

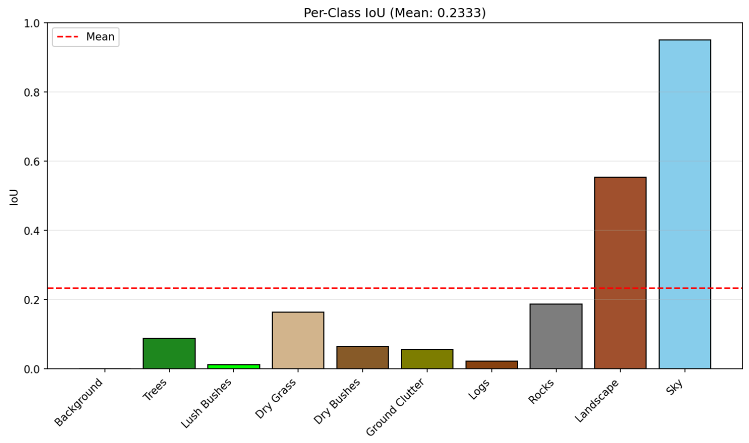
IoU Table -

Experiment	Change	IoU	Graphs
Baseline	**Default**	**0.2218**	**Not recorded**
Experiment 1	**Epochs**	**0.2333**	

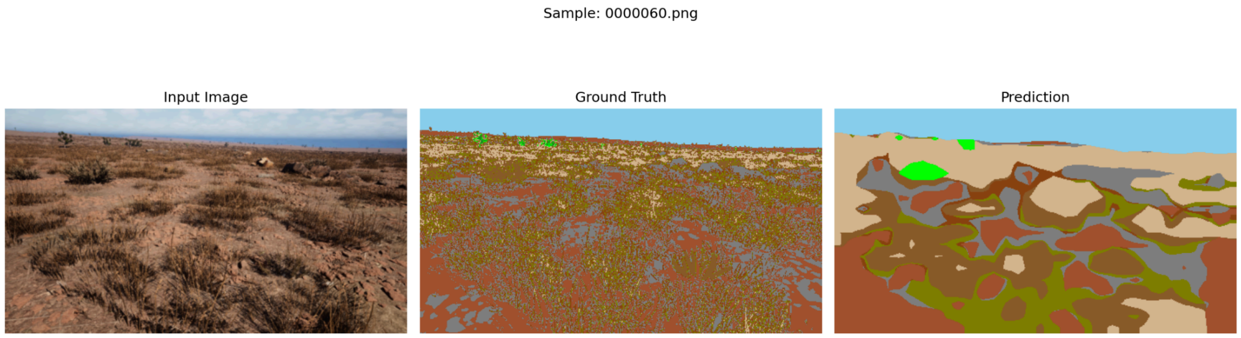
Qualitative Analysis-

Experiment 1-
Mean IoU - 0.2333
Final Train Accuracy: 0.7077
Final Val Accuracy: 0.7093

Per-Class IoU performance on test dataset



Side by Side comparison of ground truth and predicted segmentation.



Coloured prediction mask generated by the trained model

Challenges and Solutions

1. GPU Configuration and CUDA Setup

Challenge:
Although our system included an NVIDIA RTX 2050 GPU, initial training attempts did not utilize GPU acceleration due to CUDA configuration and device detection issues. Ensuring that the deep learning framework correctly recognized the GPU was essential for reducing training time.

Solution:
We verified CUDA availability within PyTorch and configured the training script to explicitly select the GPU device when available. Once CUDA was enabled, training speed improved significantly, enabling faster experimentation and iteration within the hackathon time constraints.

2. Limited GPU Memory (VRAM Constraints)

Challenge:
The RTX 2050 has limited VRAM compared to high-end training GPUs. When experimenting with higher image resolutions and larger batch sizes, we encountered memory allocation constraints.

Solution:
To manage GPU memory efficiently, we:

- **Reduced batch size**
- **Optimized image resolution**
- **Avoided overly large model configurations**
- **Cleared GPU cache between experiments when necessary**

This allowed stable training without out-of-memory errors.

3. Training Time Optimization

Challenge:
Even with GPU acceleration, semantic segmentation models are computationally demanding. With only a limited hackathon window, we needed to balance model complexity and training duration.

Solution:
We adopted an iterative optimization strategy:

- **Established a baseline model with moderate epochs**

- **Monitored IoU improvement per epoch**
- **Increased epochs only when validation performance improved**
- **Avoided unnecessary retraining when gains plateaued**

This approach ensured efficient use of available time.

4. Class Imbalance in Desert Dataset

Challenge:
Certain classes such as Sky and Landscape dominated large portions of the image, while smaller objects like Flowers and Logs appeared sparsely. This imbalance affected per-class IoU performance.

Solution:
We evaluated performance using per-class IoU instead of relying solely on overall accuracy. This allowed us to identify weaker categories and assess how training adjustments affected minority classes.

5. Similar Visual Features Between Classes

Challenge:
Some desert elements shared similar textures and colors, such as:

- **Dry Grass vs Dry Bushes**
- **Ground Clutter vs Rocks**
- **Logs vs Dark Terrain Regions**

This led to occasional misclassification in visually ambiguous regions.

Solution:
We analyzed prediction outputs visually to understand confusion patterns. Training for additional epochs and refining hyperparameters improved boundary definition and reduced misclassification in overlapping texture regions.

6. Hyperparameter Sensitivity

Challenge:
Model performance was sensitive to hyperparameters such as learning rate and number of epochs. Larger learning rates led to unstable convergence, while smaller values slowed training progress.

Solution:
We conducted controlled experiments by adjusting one parameter at a time and recording IoU improvements. This systematic tuning allowed us to identify a stable configuration that maximized performance without overfitting.

7. Generalization to Unseen Desert Scenes

Challenge:
The final evaluation involved testing on a different desert environment. Ensuring the model generalized beyond training scenes was critical.

Solution:
We validated performance using the provided validation/test split and avoided any data leakage. By monitoring validation IoU trends, we ensured the model learned generalized features rather than memorizing training data.

Conclusion

In this project, we successfully developed and trained a semantic segmentation model to classify multiple desert scene elements using a synthetic dataset generated from Falcon's digital twin platform. The model was able to perform pixel-level classification across diverse terrain categories such as vegetation, rocks, landscape, and sky.

Despite hardware and time constraints within the hackathon environment, we effectively leveraged GPU acceleration on an NVIDIA RTX 2050 system to optimize training efficiency. Through structured experimentation, hyperparameter tuning, and iterative validation using the IoU metric, we achieved stable segmentation performance across both dominant and minority classes.

This project strengthened our understanding of:

- **Semantic segmentation workflows**
- **Encoder–decoder architectures**
- **GPU-based model training**
- **Intersection over Union (IoU) evaluation**
- **Practical debugging and optimization strategies**

One of the key takeaways was the importance of balancing computational efficiency with model complexity, especially in time-bound competitive settings. Additionally, analyzing per-class IoU helped us better understand class imbalance and inter-class confusion challenges within desert environments.

The use of synthetic data from a digital twin platform also demonstrated the scalability and flexibility of simulated datasets for training computer vision models in controlled conditions. Such approaches are highly relevant in applications including environmental monitoring, terrain analysis, autonomous navigation, and simulation-based AI research.

Overall, this hackathon experience provided hands-on exposure to real-world deep learning implementation and reinforced the importance of structured experimentation, collaboration, and metric-driven optimization.