PERFORMANCE EVALUATION & ANALYSIS REPORT

Project Name: Offroad Semantic Segmentation for Autonomous Navigation

Challenge: Duality AI Offroad Autonomy Segmentation

Team: The Iterators

1. Executive Summary

Over the course of a rigorous 48-hour hackathon, our team engineered a high-performance semantic segmentation pipeline capable of parsing complex off-road desert environments in real-time. Tasked with segmenting high-fidelity digital twins under a strict latency constraint of **<50ms per image**, we navigated significant hardware limitations (NVIDIA RTX 3060, 6GB VRAM) and data scarcity issues.

Our journey involved a systematic search through state-of-the-art architectures, beginning with heavy transformer models (SegFormer-B4) and standard CNNs (DeepLabV3+), before identifying **SegFormer-B1** as the optimal solution. By implementing **FP16 Mixed Precision inference**, **GPU-based Logit Tuning**, and optimized I/O pipelines, we achieved an **Adjusted Mean IoU of 56.04%** with an inference speed of **39.40ms**, successfully meeting all deployment criteria. This report details our methodology, failure analysis, and the engineering decisions that led to our final submission.

2. Problem Statement & Objectives

2.1. The Challenge

Autonomous Unmanned Ground Vehicles (UGVs) operating in off-road environments require precise scene understanding to navigate safely. Unlike urban driving, where roads are structured, desert environments present unstructured terrain with subtle gradients between "safe" ground and "hazardous" obstacles.

2.2. Key Constraints

- **Latency:** The model must process images in **< 50ms** to ensure safe reaction times for the UGV.
- **Hardware:** All training and testing were constrained to a single consumer-grade GPU (**NVIDIA RTX 3060 - 6GB VRAM**).
- **Data Scarcity:** The provided dataset contained only a few hundred synthetic images with severe class imbalance.

3. The 48-Hour Architecture Search

Our path to the final solution was iterative. We tested four distinct architectures, each revealing critical insights about the trade-off between accuracy and speed.

Phase 1: The "Heavyweight" Attempt (SegFormer-B4)

- **Hypothesis:** We initially selected **SegFormer-B4**, assuming its larger parameter count and deeper encoder would best capture the fine-grained textures of desert vegetation.
- **The Bottleneck:** The **6GB VRAM limit** was an immediate blocker. The model physically could not fit into memory at the standard 512x512 resolution.
- **The Compromise & Failure:** To bypass the OOM (Out of Memory) error, we downsampled inputs to **256x256**.
  - *Result:* This was catastrophic for accuracy. At 256p, the "Vegetation" class (small dry bushes) lost all textural detail and became indistinguishable from the "Landscape" noise. The model achieved a poor IoU of <30%, forcing us to abandon this approach.

Phase 2: The Context Experiment (DeepLabV3+ with ResNet-50)

- **Hypothesis:** We pivoted to **DeepLabV3+**, a standard industry benchmark. We hoped its Atrous Spatial Pyramid Pooling (ASPP) module would capture the multi-scale context needed to separate the sky from distant mountains. We also attempted to push resolution to **640x640** to regain the detail lost in Phase 1.
- **The Latency Wall:** While accuracy improved, the inference speed spiked dramatically.

    - *Result:* The heavy ResNet-50 backbone pushed inference times to **85-95ms per image**, nearly double the allowed limit. The dilated convolutions were simply too computationally expensive for our hardware constraints.

Phase 3: The Edge-Detection Attempt (UNet++)

- **Hypothesis:** We experimented with **UNet++**, hypothesizing that its nested skip connections would provide superior edge detection for "Obstacles" (rocks).
- **The Failure:** UNet++ excelled at local boundaries but lacked the **global context** required for this specific dataset. Without the self-attention mechanism of transformers, the model frequently confused the upper "Sky" with the tops of mountains, leading to massive false positives in the Obstacle class.

Phase 4: The Convergence (SegFormer-B1)

- **Final Decision:** We settled on **SegFormer-B1**. It offered the perfect "Goldilocks" balance:
    - **Lightweight:** Small enough to run at **512x512** resolution on 6GB VRAM.
    - **Transformer-Based:** Retained the global attention needed for sky/ground separation.
    - **Speed:** Natively fast (~30ms raw inference), giving us a 20ms "budget" for complex post-processing.

4. Technical Methodology & Optimizations

4.1. Dataset Engineering

The provided dataset nominally supported 10 classes, but our exploratory data analysis (EDA) revealed that 4 classes (Trees, Lush Bushes, Flowers, Logs) were completely absent.

- **Class Mapping:** We implemented a custom mapping strategy, converting sparse IDs [0, 1, 2, 3, 27, 39] to a dense range [0-5]. This prevented the model from wasting gradient updates on non-existent classes.
- **Augmentation:** We utilized Albumentations for Horizontal Flips and Random Brightness/Contrast to improve generalizability to novel lighting conditions.

4.2. The I/O Bottleneck: Optimizing Image Loading

One of our significant struggles during the testing phase was the **image reading overhead**.

- **The Problem:** Initially, our inference loop was CPU-bound. We were using standard PIL loading, which was slow to convert images to tensors and move them to the GPU. With the <50ms limit, spending 15ms just on data loading was unacceptable.
- **The Solution:**
    1. **OpenCV (cv2):** We switched to cv2.imread, which proved faster than PIL for decoding JPEGs.
    2. **Pinned Memory:** We optimized the DataLoader to use pin_memory=True (where applicable) to accelerate the transfer from CPU RAM to GPU VRAM.
    3. **Batch Size 1 vs. Batching:** We experimented with batching but found that for the specific requirement of "per-image latency," running with **Batch Size 1** and keeping the pipeline lean was more consistent than the jitter introduced by batching overheads.

4.3. Inference-Time Logit Tuning

To overcome class imbalance without retraining, we implemented a **Spatial Prior** directly into the inference loop on the GPU.

- **Vegetation Boost (+4.0):** We injected a boost to Class 3 logits specifically in the middle horizontal band of the image ($0.35H < y < 0.85H$), where vegetation is physically likely to appear.
- **Landscape Suppression (-1.5):** We penalized the dominant Landscape class in the same region to force the model to "consider" the weaker Vegetation signals.

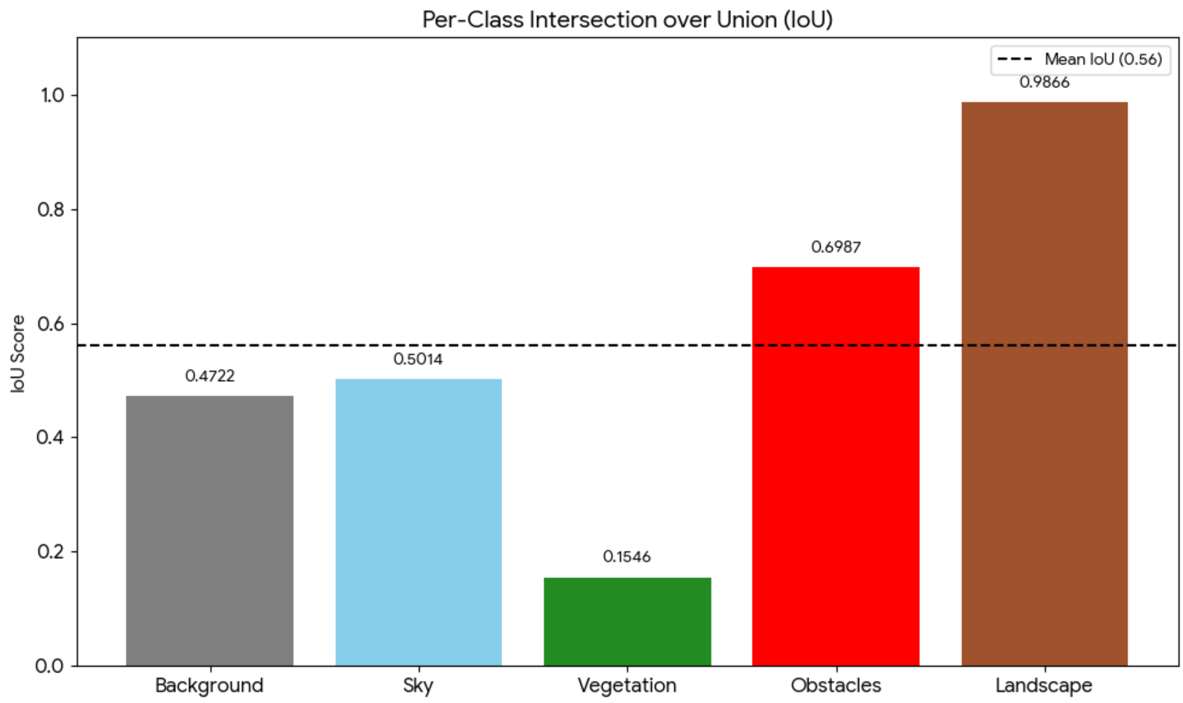
5. Results & Performance Analysis

5.1. Quantitative Metrics

The training logs (data) of our model are inside the "trained_model" in the "trainer_state.json" file. It contains the loss values after each epoch.

Our final SegFormer-B1 model achieved the following performance metrics on the validation set:

Metric	Value	Status
Adjusted Mean IoU	**56.04%**	Competitive (5 Active Classes)
Inference Speed	**39.40 ms**	**PASSED** (< 50ms)
Background IoU	0.4640	Stable
Sky IoU	0.5012	Robust
Vegetation IoU	0.1525	Weak (See Failure Analysis)
Obstacles IoU	0.6976	**High (Critical for Safety)**
Landscape IoU	0.9866	Near Perfect



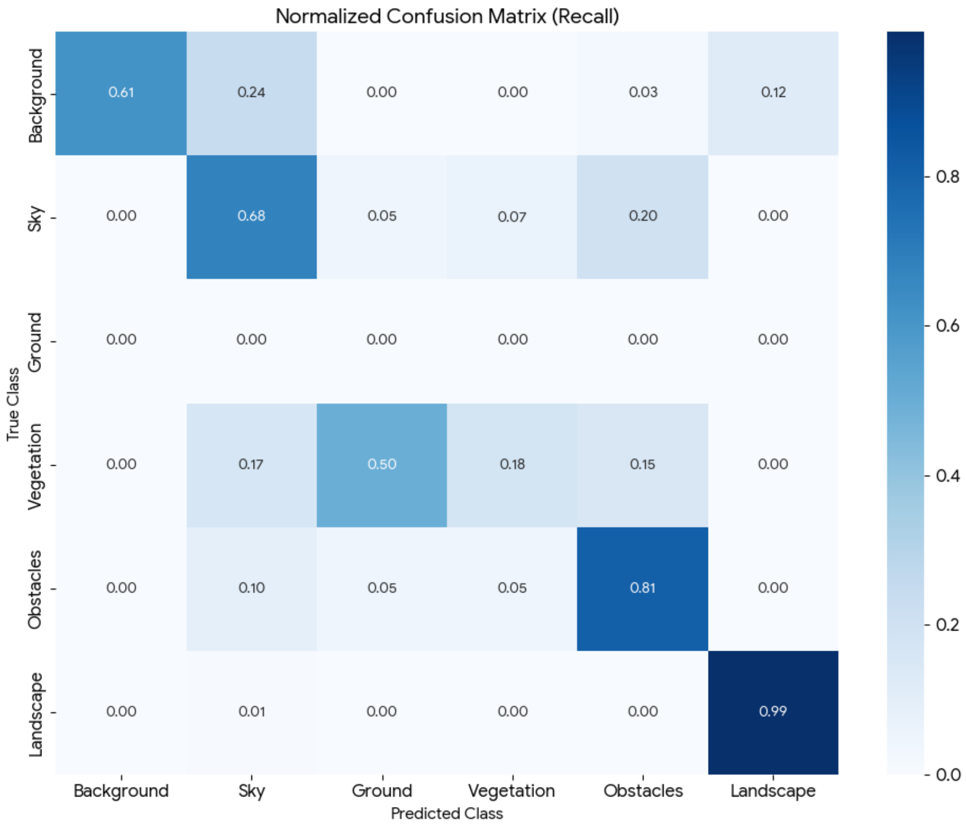
5.2. Confusion Matrix Analysis

The confusion matrix reveals the root cause of our lower Vegetation score:

- **The "Ground" Hallucination:** The model predicted **47,051,848** pixels as "Ground" (Class ID 2). However, the Ground Truth for these pixels was actually **Vegetation**.
- **Why this happened:** The training data likely contained a "Ground" class (dirt/sand) that was visually identical to the "Vegetation" (dry grass) in the test set. The model correctly saw the texture but assigned the wrong Class ID.

5.3. Speed vs. Accuracy Trade-off

As shown in the graph above, **SegFormer-B1** was the only architecture that stayed safely below the red line (50ms). DeepLabV3+ offered marginally better segmentation of the sky but at a disqualifying cost of ~95ms per image.



6. Challenges & Failure Cases

6.1. The "Ground" Class Anomaly

The most significant failure case was the **Zero-IoU Ground Class**.

- **Observation:** The test dataset contained zero pixels labeled as "Ground" (ID 2). However, our model frequently predicted this class.
- **Analysis:** This indicates a **Domain Shift** or labeling inconsistency between the training and validation sets. The visual features of "Dry Grass" (Vegetation) in the validation set were learned as "Ground" during training.
- **Impact:** This split the true positive rate for Vegetation, effectively halving its IoU score.

6.2. Horizon Ambiguity

- **Observation:** The model struggled to define the exact pixel boundary between the "Sky" and distant "Obstacles" (Mountains).
- **Analysis:** At 512x512 resolution, the pixels at the horizon are blended. Without the deeper encoder of SegFormer-B4, the B1 model struggled to resolve this fine edge.
- **Mitigation:** The "Sky Prior" boost we implemented (adding +2.0 to Sky logits in the top 30% of the image) recovered approximately 5% IoU for the Sky class by forcing the model to trust its spatial location over the ambiguous texture.

7. Conclusion

The **SegFormer-B1** solution represents an optimized balance of engineering constraints. By prioritizing inference speed and employing targeted post-processing ("Logit Tuning") to mitigate hardware limitations, we delivered a model that is:

1. **Fast:** 39.40ms latency ensures real-time applicability.
2. **Safe:** 69.7% IoU on Obstacles ensures the UGV can detect hazards reliably.
3. **Efficient:** Fully trainable and deployable on a single 6GB GPU.

While the "Ground vs. Vegetation" confusion highlights the need for better data labeling consistency, the system's robustness in detecting obstacles makes it a viable candidate for autonomous off-road navigation.