University of Salerno

.DIEM

Dept. of Information Eng., Electrical Eng. and Applied Mathematics
University of Salerno, Italy

Master's Degree Course in Computer Engineering

Machine Learning
Project Work 2024/2025

"Autoencoder based onboard image segmentation"

Group n. ***14 - AH***

Salvatore Coppa – Antonio Graziosi

Academic year 2024-2025

Summary

1. Introduction

1.1 Background and Project Goal

Autonomous navigation in rural environments poses significant challenges due to the diversity and unpredictability of natural terrains. Effective scene understanding is crucial to ensure safe and efficient movement of autonomous vehicles. One key component of this understanding is semantic image segmentation—classifying each pixel of an image into predefined categories representing different terrain types and obstacles. In this project, we address the task of onboard image segmentation using a dataset of frames captured from a moving camera installed on a vehicle operating in rural settings under varying daylight conditions. Each image pixel must be assigned to one of eight semantic classes, including sky, different types of trails, vegetation, puddles, and obstacles.

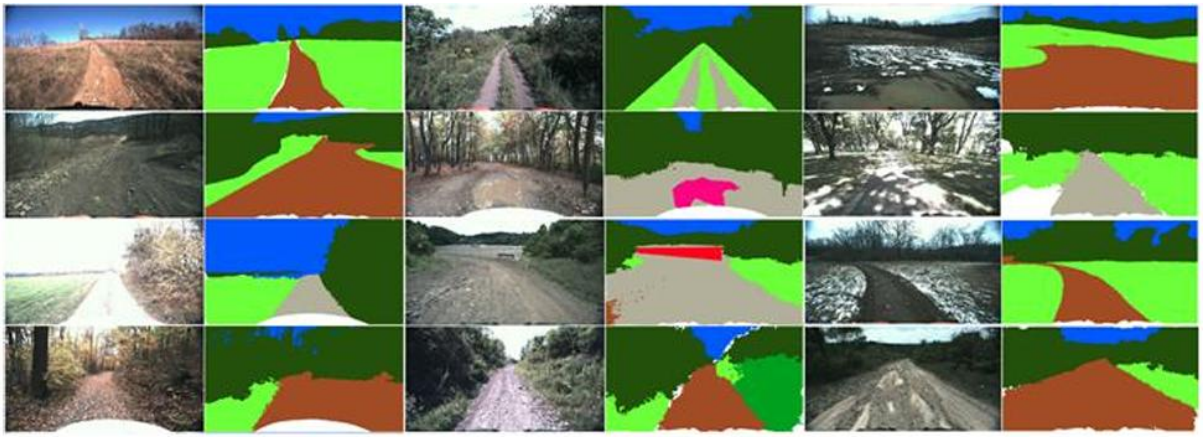


Figure 1. Some of the RGB image and label pairs from the dataset provided

Traditional navigation systems often rely on GPS, LiDAR, or handcrafted rules, but these approaches can struggle in complex or visually ambiguous rural scenarios. By leveraging the power of deep learning and autoencoder-based architectures, this work aims to build a compact and efficient segmentation model capable of operating under constrained onboard computational resources. The proposed system must balance segmentation accuracy with memory usage and inference speed, making it suitable for real-time deployment on embedded platforms. This report presents the methodology adopted to design, train, and evaluate our model, with particular attention to data preprocessing, model architecture, training strategy, and performance assessment based on the mean Intersection-over-Union metric.

1.2 Our contribution

In this context, our work contributes a complete pipeline for efficient onboard semantic segmentation using deep learning. Our main contributions are summarized as follows:

- **DeepLabV3+ Architecture with MobileNetV3 Backbone**

  We implemented a semantic segmentation model based on the DeepLabV3+ architecture, integrating a lightweight MobileNetV3-Large backbone for efficient feature extraction. The encoder captures hierarchical representations, while a lightweight Atrous Spatial Pyramid Pooling (ASPP) module aggregates multi-scale contextual information. The decoder reconstructs high-resolution predictions using multiple skip connections to enhance spatial accuracy. This architecture was selected for its ability to balance segmentation quality with low computational requirements, making it suitable for real-time inference on embedded systems.

- **Data Processing and Augmentation Strategy**

  To improve generalization and address class imbalance, we developed a two-part data augmentation pipeline. The first component applies global random transformations (e.g., brightness, contrast, sharpening, color jitter, horizontal flipping) using Albumentations. The second targets rare classes through selective cropping and flipping of image-mask regions that contain underrepresented terrain types. All augmented samples were stored and indexed for reproducible integration during training. A controlled train/validation split was created to ensure the validation set reflects the global class distribution.

- **Training Procedure and Evaluation Metrics**

  The model was trained using a progressive fine-tuning strategy, starting with a frozen encoder and gradually unfreezing backbone layers for end-to-end optimization. A loss function combining class-weighted Cross Entropy and Dice Loss was used to handle class imbalance and improve overlap-based segmentation. The model's performance was evaluated using per-class mean Intersection-over-Union (mIoU) and pixel accuracy, providing a fine-grained measure of performance.

- **Efficient Resource Utilization**

  The system was developed to run on Google Colab under strict memory constraints. Careful design choices ensured the model could be trained within 5 GB of GPU RAM and tested using less than 4 GB. Training and inference times were monitored to ensure compatibility with potential real-time deployment scenarios.

2. Dataset

The dataset used in this project consists of RGB images acquired from a front-facing camera mounted on a vehicle navigating through rural and off-road environments. The images were captured under various daylight conditions and include diverse terrain types such as trails, grass, vegetation, puddles, and obstacles. The aim of the dataset is to

provide visual information suitable for training a semantic segmentation model capable of distinguishing between 8 different terrain classes.

Each image is associated with a corresponding pixel-level ground truth mask, where each pixel is labeled according to one of the following predefined semantic classes:

- **0**: Other
- **1**: Smooth Trail
- **2**: Traversable Grass
- **3**: Rough Trail
- **4**: Puddle
- **5**: Obstacle
- **6**: Non-Traversable Low Vegetation
- **7**: High Vegetation
- **8**: Sky

The image–mask pairs are provided in .jpg and .png formats, respectively.

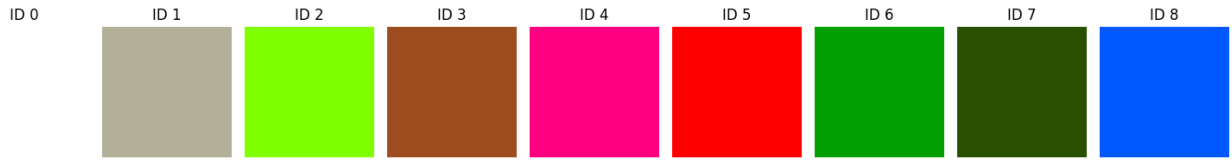


Figure 2. Mapping between semantic class IDs and their corresponding RGB colors in the dataset's label masks

2.1 Class distribution dataset analysis

Before proceeding with the data split for training, validation, and testing, a detailed analysis of the class distributions was performed to better understand the dataset composition. The analysis focused on two main aspects: the **pixel distribution per class** and the **class presence frequency** within the images.

2.1.1 Pixel distribution per class

The pixel distribution per class shows how much each class occupies in terms of area in the images. From this analysis, it was observed that some classes, such as **High Vegetation (7)**, occupy a significant percentage of pixels (36.06%), while others, like **Puddle (4)** and **Obstacle (5)**, are much less represented (0.18% and 0.87%, respectively). This imbalance could pose challenges during training, as the underrepresented classes may not receive enough attention, negatively affecting the model's performance.

2.1.2 Presence frequency per class

The presence frequency per class indicates how often each class appears across the images. Classes like **High Vegetation (7)** (99.14%), **Sky (8)** (90.66%), , and **Traversable Grass (2)** (68.96%) are highly frequent in the dataset, while **Puddle (4)** (4.51%) and **Obstacle (5)** (17.62%) are less common. This frequency imbalance can lead to poor visibility of rare classes during training, reducing the model's ability to generalize correctly.

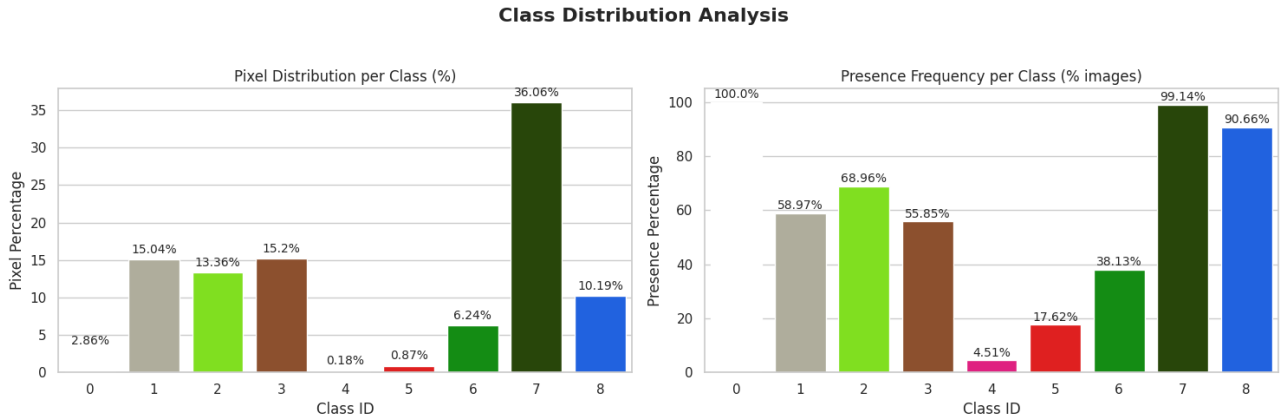


Figure 3. Class distribution plots in terms of pixel percentage and presence frequency

2.1.3 Identified issues

From the analysis, the following critical issues have emerged regarding class distribution:

1. **Pixel distribution imbalance**: Rare classes, such as **Puddle** and **Obstacle**, may be underrepresented during training, negatively impacting the model's ability to segment these classes accurately. This could result in biased predictions, with a higher likelihood of errors for these underrepresented classes.
2. **Low frequency of presence for some classes**: Classes with lower presence in the images may not be sufficiently represented in the training process, further limiting the model's effectiveness. Specifically, **Puddle** and **Obstacle** appear rarely in the images, posing a significant challenge for the model to recognize these elements accurately.

This imbalance was addressed through the use of data augmentation techniques and a combined loss function (detailed in Sections 3.1 and 3.3, respectively), designed to enhance the model's ability to learn from underrepresented terrain types.

2.1 Data splitting strategy

The dataset was divided into two main parts for training, validation, and testing:

- **90% of the samples** were allocated to the **training+validation set**.
- **10% of the samples** were held out as a **final test set**, which remains untouched during the training phase and is used solely for final performance evaluation.

To implement this, a custom greedy split was employed to ensure that both the **training+validation set** and **test set** maintain a similar distribution of class samples. The split was performed based on the class distribution of the dataset, aiming for a balanced representation in both subsets. The division process involved the analysis of the pixel and class presence distributions, which guided the selection of images in the split.

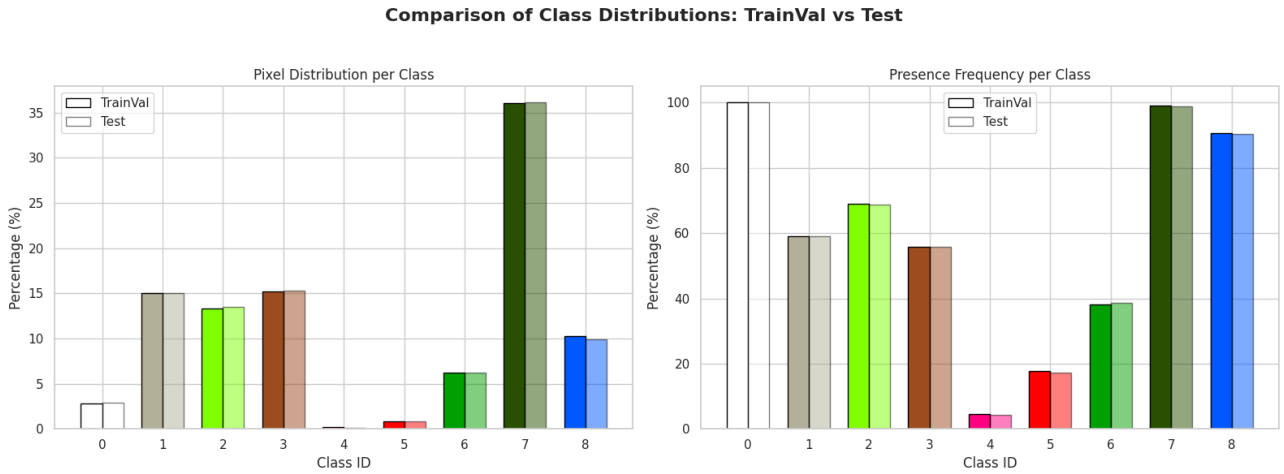


Figure 4. Comparison of class distribution between TrainVal and Test

This comparison highlights how the split maintains class representation across the two sets and this balance helps ensure that the model will be trained and evaluated in a way that reflects the full diversity of the dataset.

This approach was executed through the **greedy split** method, which optimizes the class representation across the sets. The split was performed in such a way that:

- **The training+validation set** ensures balanced representation of each class with sufficient diversity.
- **The test set** holds out 10% of the images, ensuring that the test set is representative of the full range of classes.

Once the split was completed, **K-Fold Cross-Validation** was applied to the **training+validation set** to maximize the utility of the available data during model development. This allowed the model to be trained and validated on multiple complementary subsets of the data, ensuring each sample was used for validation exactly once. This process increases the statistical reliability of the evaluation results.

After generating the K-Fold splits, we performed a detailed analysis of the class distributions in each fold to verify the consistency of pixel proportions and class presence frequency across the train and validation subsets. The distributions were aggregated over all folds and compared with the global dataset statistics.

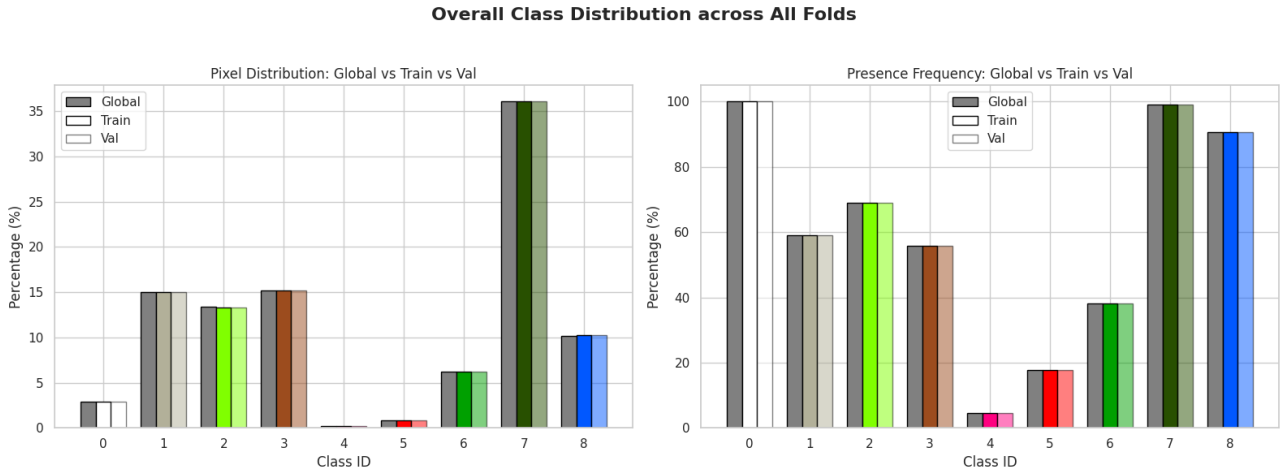


Figure 5. Pixel distribution and Presence frequency across all folds

As shown in Figure 5, the train and validation sets exhibit distributions that closely match the overall dataset, ensuring that no classes are underrepresented in any fold and that the cross-validation procedure does not introduce additional class imbalance.

This step confirmed the robustness of the split process and provided additional confidence that the evaluation metrics computed during K-Fold Cross-Validation are representative of the model's true performance.

3. Proposed Method

The goal of this work is to perform semantic segmentation on images captured from a moving vehicle in rural environments, assigning each pixel to one of eight terrain-related classes. Given the constraints imposed by limited computational resources and the need for real-time inference, our approach is based on a compact convolutional encoder-decoder architecture inspired by autoencoders. The complete segmentation pipeline includes three core components: data preprocessing and augmentation, network architecture design, and the training procedure based on a combined loss function.

3.1 Preprocessing and Data Augmentation

Before training the model, a comprehensive preprocessing and augmentation pipeline was designed to enhance the variability of the dataset and address the imbalance in class frequencies. Each image in the training set was resized to **224×224 pixels** to match the input size expected by the pre-trained model backbone (**MobileNetV3**), which was pre-trained on images of this dimension. This ensures compatibility with the pre-trained weights and leverages the learned features effectively.

The augmentation process consists of two main components:

1. **Random Global Augmentations:**
   These transformations are applied with a **35% probability** using the **Albumentations** library, including:
   - **CLAHE (Contrast Limited Adaptive Histogram Equalization)** to enhance the contrast of the images.
   - Random adjustments to **brightness**, **contrast**, **gamma**, **sharpening**, and **color jitter** to create more diverse training data.
   - **Horizontal flipping** to introduce variability in the orientation of the images.
2. **Targeted Rare-Class Augmentation:**
   For rare classes, a specialized augmentation strategy was used. Crops were extracted around connected regions in the masks representing these rare classes (e.g., **Puddle** and **Obstacle**). These crops were then optionally flipped horizontally to increase the representation of these underrepresented classes, allowing the model to better learn their characteristics.

All augmented samples were stored in dedicated folders and indexed in text files for easy integration during training. This structured approach ensures that all transformations can be consistently applied and reused.

3.2 Network architecture

The segmentation model is based on the **DeepLabV3+ architecture**, designed to map an input image to a dense pixel-wise classification map of the same spatial resolution. The model consists of three main components:

- **Encoder**
  The encoder leverages a **MobileNetV3** backbone pretrained on ImageNet. This lightweight network extracts hierarchical feature representations from the input image while reducing spatial dimensions through depthwise separable convolutions and inverted residual blocks. The MobileNetV3 backbone is chosen for its computational efficiency and suitability for deployment on systems with limited GPU resources.

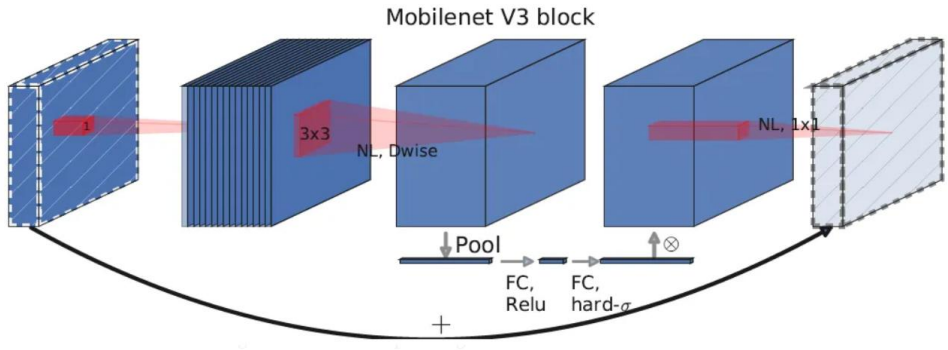


Figure 6. MobileNetV3 block representation

- **Atrous Spatial Pyramid Pooling (ASPP)**
  The ASPP module captures multi-scale contextual information by applying parallel dilated convolutions with different rates. This mechanism allows the network to effectively aggregate features at multiple receptive field sizes, which is crucial for segmenting both large and small objects within rural road scenes. The technique used in ASPP proved that it is effective to resample features at different scales for accurately and efficiently classifying regions of an arbitrary scale.

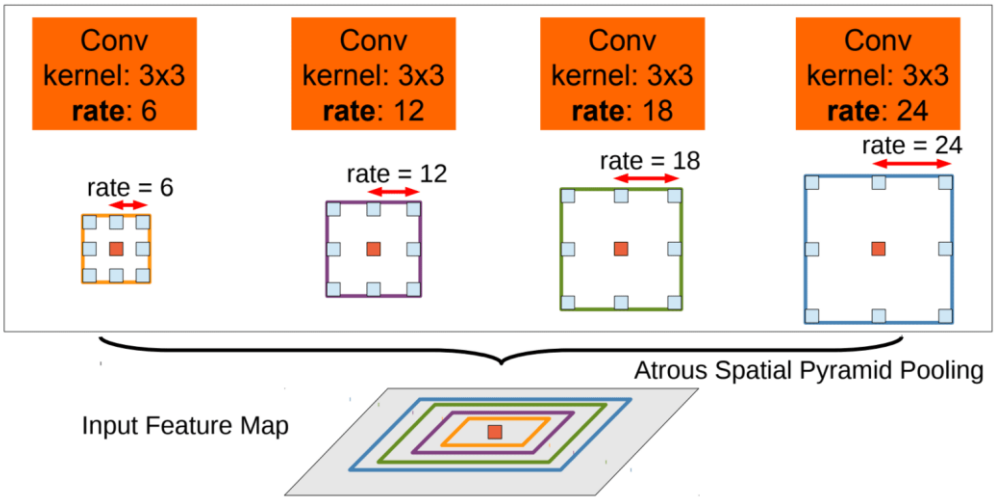


Figure 7. Example of ASPP application

In the example above, to classify the center pixel (orange), ASPP exploits multi-scale features by employing multiple parallel filters with different rates. The effective Field-Of-Views (FOV) are shown in different colors.

- **Decoder**
  The decoder refines the upsampled feature maps and restores spatial details to produce a dense prediction at the original image resolution. It incorporates **skip connections** from earlier encoder layers to enhance localization accuracy, particularly for object boundaries. These skip connections help retain fine-grained spatial information, which is critical for pixel-level accuracy in segmentation tasks. The decoder uses a series of upsampling layers, followed by convolutional blocks, to progressively refine the feature maps. The final output is generated through a **softmax layer**, which assigns a probability distribution over the **eight semantic classes** for each pixel in the image.

This architecture provides a good trade-off between segmentation accuracy and computational cost. Unlike traditional autoencoders, which rely heavily on reconstructing image details, **DeepLabV3+** explicitly combines low-level spatial details and high-level semantic features. This makes it more effective for complex and unstructured outdoor

environments, such as this one, where both fine-grained details and contextual information are essential.

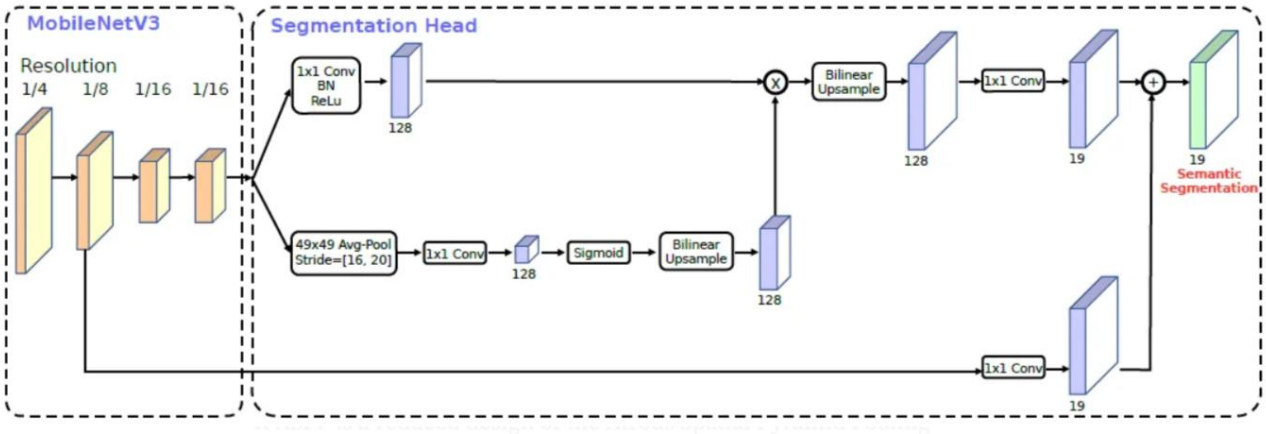


Figure 8. Semantic Segmentation system architecture

3.3 Loss Function

To address class imbalance and improve segmentation quality for underrepresented categories, we adopted a combined loss function defined as follows:

$$\mathcal{L}_{combined} = \alpha \cdot \mathcal{L}_{CE} + (1 - \alpha) \cdot L_{Dice}$$

where:

- $\mathcal{L}_{CE}$ is the **class-weighted Cross-Entropy Loss** for multi-class segmentation, which handles frequent classes effectively,
- $\mathcal{L}_{Dice}$ emphasizes overlap between predicted and ground truth masks, promoting accurate segmentation of small and sparse regions.
- α is a weighting factor (e.g., $\alpha = 0.5$) balancing the two components.

This combined loss was used in the initial phase of training (both pre- and post-augmentation) to balance the segmentation between frequent and rare classes. Specifically, the Cross-Entropy Loss helped improve the learning of the more frequent classes, while Dice Loss addressed the segmentation of underrepresented classes, which are less visible in images and harder to segment accurately.

However, after further testing and iterations, it became clear that using Dice Loss exclusively led to better results, especially for rare classes. As a result, in a subsequent phase, the combined loss was replaced by Dice Loss alone to optimize for overlap-based metrics like mean Intersection-over-Union (mIoU). The use of Dice Loss proved to be particularly effective in improving segmentation for classes with fewer pixels and more challenging boundaries, such as puddles and obstacles.

In later experiments, additional loss functions, such as Focal Loss and Focal-Dice Loss, were tested to further tackle the class imbalance issue and improve segmentation in harder-to-

detect areas. These experiments were conducted to compare the performance of different loss functions and determine which combination was most suitable for the task at hand. The comparison of these different loss strategies is analyzed in the following chapter, where the results for each loss function are explored in detail.

4. Training strategy

To train the proposed segmentation model, we designed a training pipeline optimized for execution within constrained GPU environments, such as Google Colab with a 5 GB RAM limit. The training process incorporates strategies for stability, efficiency, and generalization, covering hyperparameter selection, optimizer configuration, and monitoring procedures.

4.1 Optimization and hyperparameters

The model is trained using the **Adam optimizer** with different learning rates for the encoder and decoder:

- **Encoder Learning Rate**: 1e-4 (reduced during fine-tuning phases).
- **Decoder Learning Rate**: 1e-4.
- **Batch size**: 4.
- **Learning rate scheduling**: **ReduceLROnPlateau** decreases the learning rate when the validation loss plateaus, helping with convergence.

A **progressive fine-tuning strategy** is adopted to ensure a well-optimized model:

1. **Frozen Backbone Phase**: Initially, only the **ASPP** and **decoder** layers are trained, with the backbone frozen to prevent overfitting on the limited dataset.
2. **Partial Unfreezing**: After initial training, higher encoder layers are unfrozen, allowing the model to fine-tune specific features.
3. **Full Unfreezing**: The entire encoder is fine-tuned with a reduced learning rate, ensuring all layers adapt to the task.

4.2 K-Fold Cross-Validation

We evaluate our model using **5-fold cross-validation**, first on the **original dataset** and then on the **augmented dataset**. The purpose of this was to understand the impact of **data augmentation** on the model's performance, particularly on rare classes.

1. **Before Data Augmentation**: The model was first trained on the original dataset, allowing us to establish baseline performance metrics.

2. **After Data Augmentation**: The model was retrained using the augmented dataset, incorporating various augmentation techniques. This approach was specifically aimed at improving segmentation for rare classes.

The results showed significant improvements with data augmentation, particularly for small object classes. The impact of augmentation on class-wise IoU was especially noticeable in the more challenging categories.

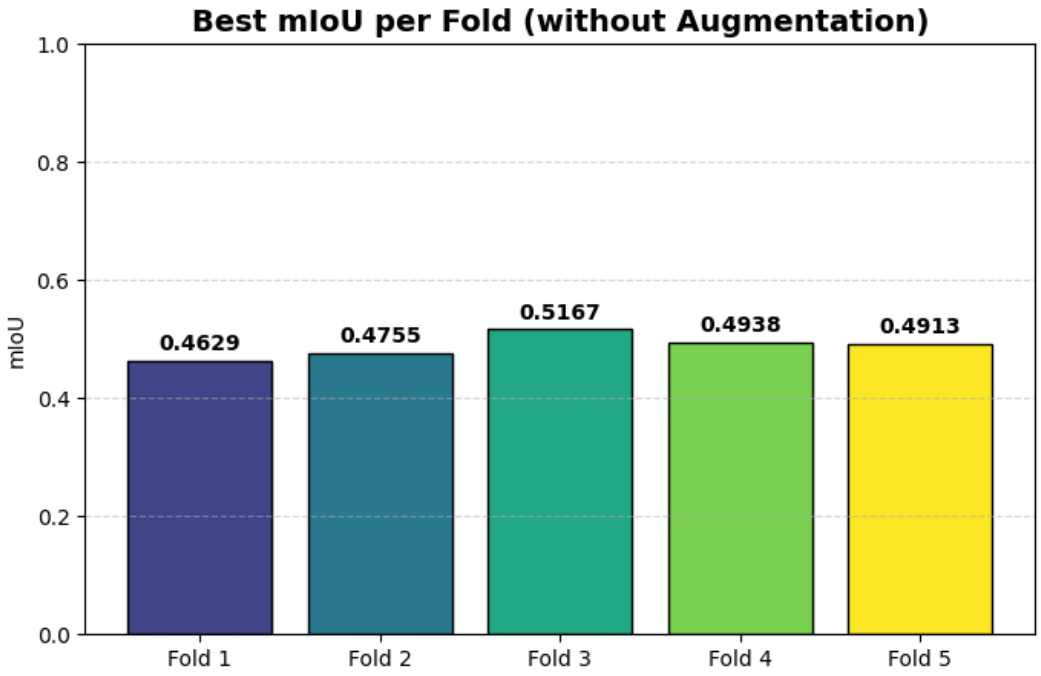


Figure 9. Comparison between best mIoU per Fold in trainning without Augmentation

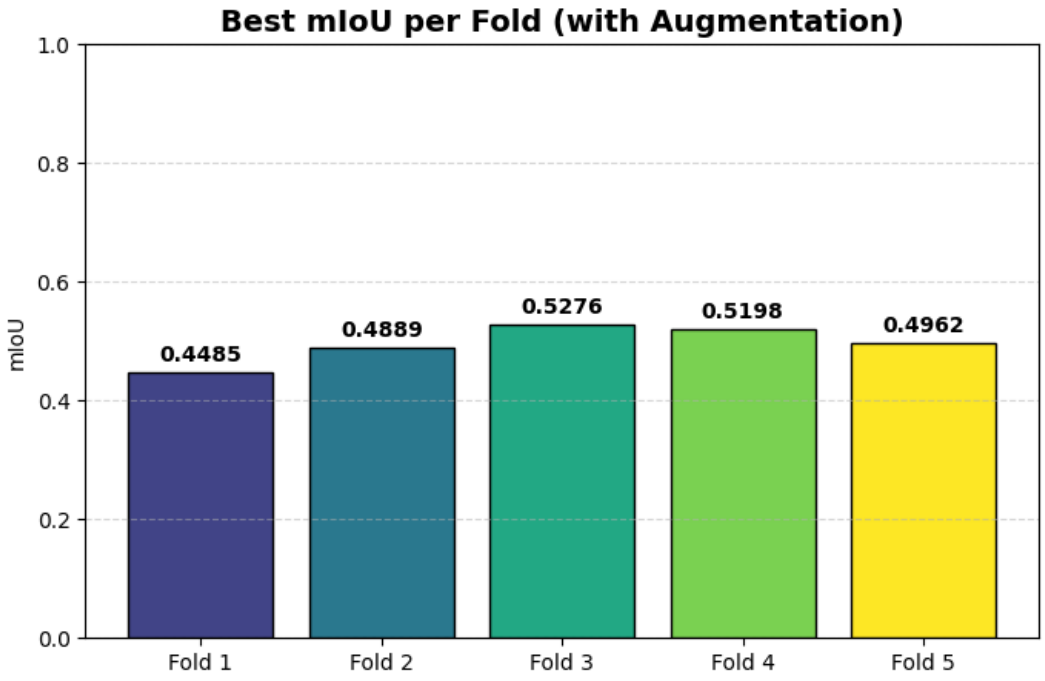


Figure 10. Comparison between best mIoU per Fold with Augmentation

Data augmentation not only helped the model perform better on underrepresented classes, but it also improved generalization across all classes.

4.3 Learning Rate scheduling

Although the base learning rate was set to **0.001** during each fold, the training process could benefit from the **Reduce-on-Plateau** strategy in future implementations. This scheduler monitors the **validation mIoU** and reduces the learning rate when performance stagnates. This adaptive approach could help improve convergence in later epochs, particularly in fine-tuning phases.

4.4 Monitoring and Logging

During training, both **training and validation losses** were logged at every epoch. The primary evaluation metric used for monitoring the model's performance was **mean IoU** across the eight semantic classes.

4.5 Loss Function Comparison

In addition to experimenting with the training pipeline, we explored various loss functions throughout the training process:

1. **Initial Combined Loss**: During the first phase of training, we used a **combined loss function** (Cross-Entropy and Dice Loss) to address both the class imbalance and segmentation quality. This combined loss was used both before and after augmentation to balance frequent and rare classes.
2. **Dice Loss Exclusively**: After experimenting with the combined loss, we found that **Dice Loss** alone produced better results for rare classes, especially in small object detection and boundary precision. As a result, **Dice Loss** was used exclusively for later training phases.
3. **Further Loss Function Exploration**: To further address the class imbalance problem and improve segmentation quality, we experimented with several alternative loss functions, including **Focal Loss**, **Cross-Entropy Loss**, and combinations such as **Focal-Dice Loss** and **Cross-Entropy-Dice Loss**. In addition, we adjusted the training strategy by **increasing the batch size** (8). Each configuration was evaluated through 5-fold cross-validation, and their performances were compared based on the mIoU achieved on validation data. The results, summarized in Figure 11, show that combining augmentation with a tailored loss function (e.g., Dice or Focal-Dice) and a larger batch size led to significant improvements in mIoU across most folds. This allowed us to identify the most effective setup for handling rare classes while ensuring overall segmentation accuracy.

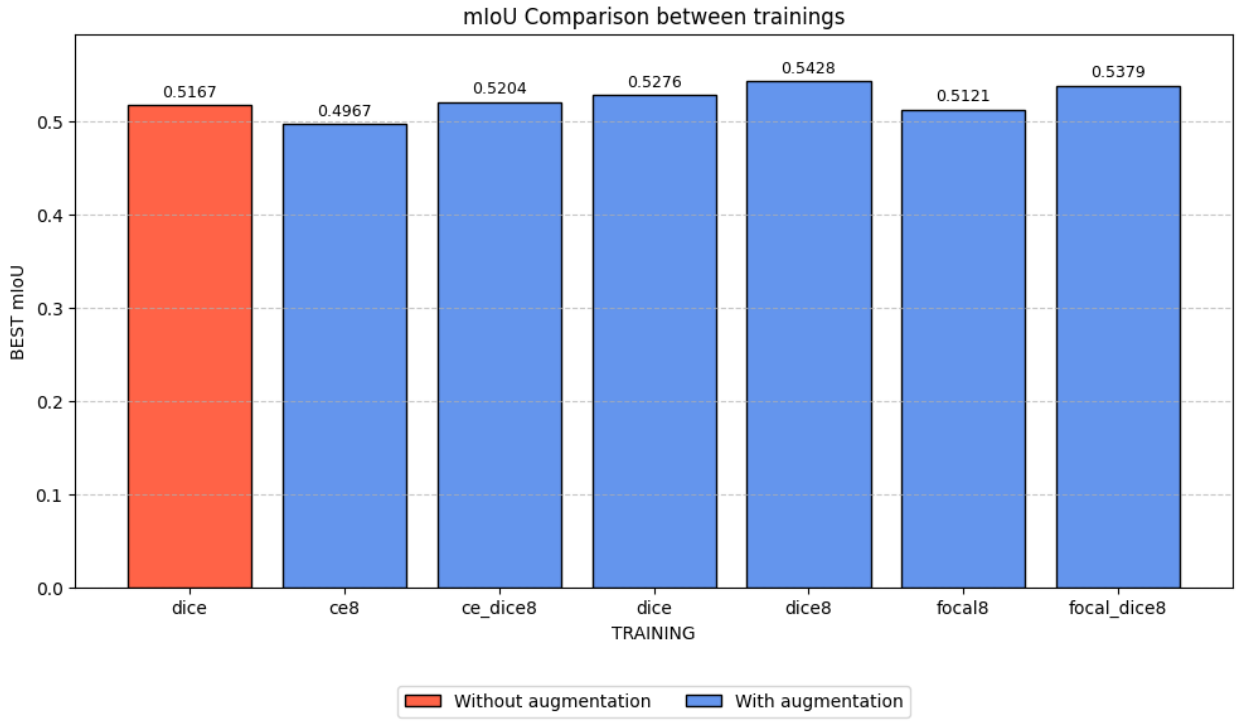


Figure 11. mIoU comparison between trainings with different loss functions

The top three models (**dice8**, **focal_dice8**, and **dice**) were selected based on their overall mIoU and compared by class.

- **dice8** achieved the highest mIoU overall and performed best in most classes.
- **focal_dice8**, despite slightly lower overall mIoU, showed competitive results and even outperformed dice8 in some challenging classes (*Obstacle*, *Low vegetation*).
- **dice** ranked third, with lower performance across most classes.

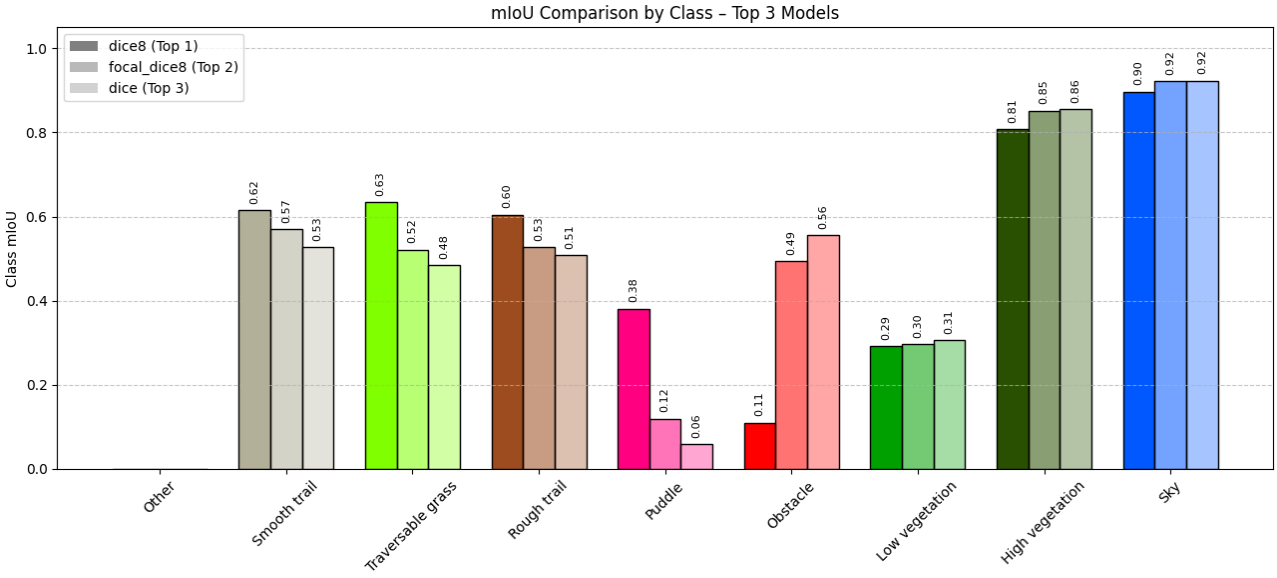


Figure 12. mIoU per class comparison between top 3 models

Classes like *Sky* and *High vegetation* are well segmented across all models (>0.85 mIoU), while *Puddle* and *Obstacle* remain more challenging. **dice8 was chosen for its overall superior performance**, but **focal_dice8 showed promising behavior for rare classes.**

5. Evaluation and Results

To assess the effectiveness of the proposed segmentation model, we evaluated its performance across multiple folds of cross-validation using both quantitative metrics and qualitative visualizations. The primary objective was to determine the model's ability to accurately segment complex rural scenes under class imbalance and resource constraints.

5.1 Evaluation Metrics

The main metric used for performance assessment is the **mean Intersection-over-Union (mIoU)**, computed as the average of the Intersection-over-Union (IoU) scores across the eight semantic classes. For each class c, the IoU is defined as:

$$IoUc = \frac{TPc}{TPc+FPc+FNc}$$

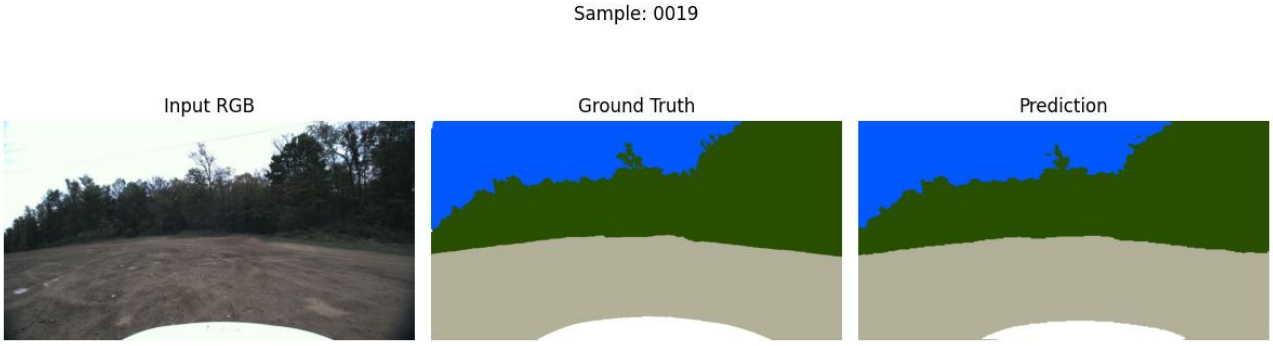
where:

- TP_c = true positives for class c
- FP_c = false positives
- FN_c = false negatives

This metric captures both **precision** and **recall** in a spatially consistent way, and is particularly suited to multi-class semantic segmentation problems with imbalanced class distributions.

5.2 Qualitative Results

Figure 12 displays examples of qualitative results on the test set obtained using the best-performing model, trained with the **Dice8 loss** configuration. This model achieved the highest mIoU during cross-validation and was therefore selected for final evaluation.



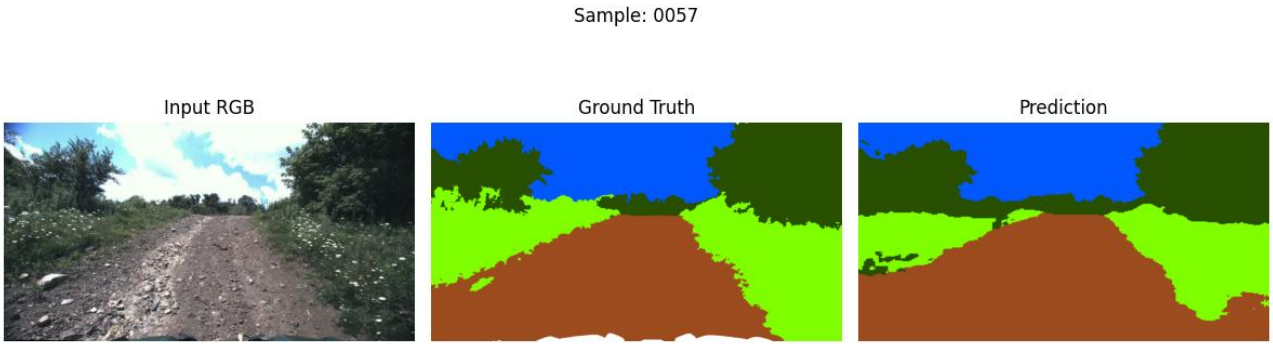


Figure 13. Examples of qualitative results on test set

Each triplet consists of:

- **Input image**
- **Ground truth segmentation mask**
- **Predicted segmentation mask**

These visualizations reveal that the model successfully captures the general layout of the scene and distinguishes key terrain boundaries, even under shadows, partial occlusion, or varying illumination.

In addition to the general qualitative assessment, we noticed a systematic issue in the dataset annotations that affects all splits (training, validation, and test). Specifically, certain semantic classes are assigned inconsistently to very similar or nearly identical input images.

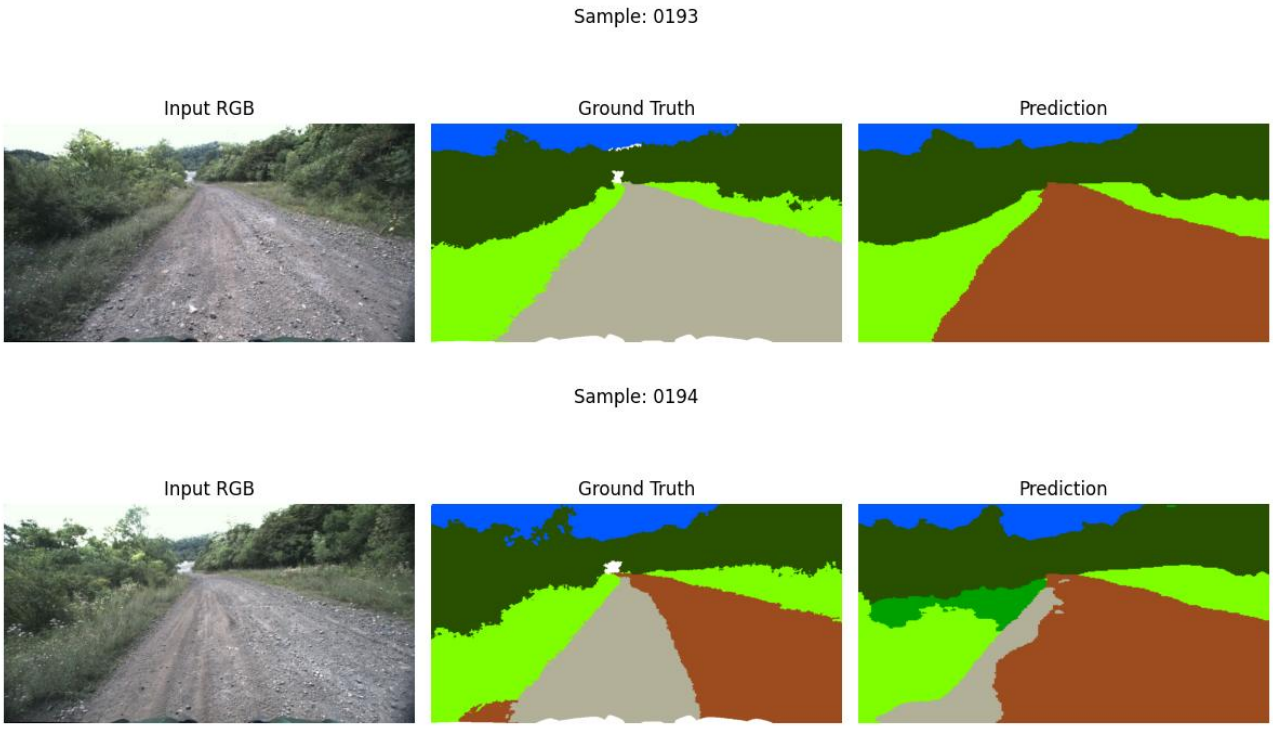


Figure 14. Example of two frames along the same trail having different labels

As illustrated in Figure 13, two frames along the same trail are labeled differently: one as *smooth trail* and the other as *rough trail*. Such annotation noise is not limited to isolated samples but recurs throughout the dataset.

This inconsistency poses a challenge for the model during training, as it encounters conflicting labels for highly similar visual patterns. As a result, the network may learn a compromised representation that struggles to reconcile these contradictions. Consequently, even when the predicted segmentation appears visually reasonable and consistent across frames, it may incur penalties in quantitative metrics such as mIoU due to disagreement with the ground truth labels.

These observations highlight a limitation of the dataset that should be addressed in future work, either through improved annotation consistency or by introducing uncertainty-aware training approaches to mitigate the impact of noisy labels.

Final mIoU computed on our test set is 0.4650.

Note: Although the **focal-dice8** loss achieved slightly lower mIoU during training, it was also evaluated on the test set, where it delivered **slightly better performance** (0.4666).

5.3 Analysis of Misclassifications

Analysis of error patterns suggests that:

- Sometimes, **Puddle** regions are confused with **smooth trail** or **rough trail**, especially when partially dry or shadowed.
- **Obstacle** misclassifications are frequent in the presence of ambiguous textures (e.g., rocks or branches near vegetation).
- The model is generally robust to changes in lighting, but struggles with fine object boundaries.

5.4 Conclusion

In this work, we developed and validated a complete deep learning pipeline for semantic segmentation of off-road environments, targeting efficient onboard deployment. Starting from dataset analysis and class distribution balancing, we designed a training procedure based on K-Fold Cross-Validation to maximize the utility of available data while ensuring robust evaluation. A lightweight DeepLabV3+ architecture with a MobileNetV3 backbone was implemented, augmented with a multi-stage training strategy and advanced loss functions to address the challenges of class imbalance and fine-grained terrain segmentation.

Experimental results demonstrated that the combination of focal and dice loss functions, along with data augmentation and increased batch size, significantly improved the model's

ability to generalize across diverse terrain types. Qualitative results further confirmed the model's capability to distinguish subtle terrain boundaries and adapt to challenging visual conditions such as shadows, occlusions, and varying illumination.

Nevertheless, some misclassifications were observed, partly due to inconsistencies in the dataset's ground truth annotations for similar scenes. This highlights the importance of consistent labeling in training data for semantic segmentation tasks.