Semantic Segmentation in Natural Environments for Autonomous Vehicles

Renfei Yu (Klaus)  
*z5546743*

Hesong Zhang  
*z5532689*  
Shihang Yao  
*z5458849*

Wenxuan Wang  
z5464319  
Shuoming Zhang  
z5522548

# Introduction

As autonomous driving technology continues to advance, a deep understanding of the surrounding environment becomes particularly critical in order to achieve safe and accurate navigation of vehicles in complex natural environments. Autonomous vehicles should behave differently when driving in different landscapes such as gravel, asphalt, sand or mud, which requires the vehicle to be able to accurately identify various scenes and objects in its path. This task is particularly challenging in natural environments, where scenes often exhibit highly irregular and unstructured features.

In this context, image semantic segmentation plays a crucial role as a technique to understand key information in complex environments. The goal of semantic segmentation is to classify the images captured by the vehicle camera at the pixel level, that is, to assign a specific category label to each pixel in the image, which helps the autonomous driving system to accurately identify and process various conditions ahead.

This study will use a multimodal dataset called WildScenes, which includes a total of 9,306 images from five sequences captured by ordinary video cameras in Venman National Park and Karawatha Forest Park in Brisbane, Australia. Each image has a resolution of 2016 x 1512 pixels. These images have been manually annotated, giving us accurate ground-reality data, which is a valuable resource for detailed analysis and model training. In this project, we will focus on using these 2D images and ignore the 3D point cloud portion of the dataset.

This report will introduce in detail the method of image semantic segmentation using U-net and Deeplabv3 models. U-net model has proved its effectiveness in medical image segmentation because of its unique symmetrical structure and excellent detail capturing ability. The Deeplabv3 model, an important milestone in the field of deep learning, optimizes segmentation accuracy and processing speed with its efficient spatial pyramid pooling and atrous convolution technology. Through the experiment and analysis of these two models, this report aims to demonstrate their application effects and potential in dealing with autonomous driving scenarios.

In the following chapters, we will discuss in detail the working principle, experimental process and results of these models, as well as their performance in autonomous driving image segmentation tasks, and discuss the results and method performance to provide technical support and theoretical basis for the navigation of autonomous vehicles in natural environments.

# Literature Review

1. Background and Significance

The development of autonomous driving systems is gradually changing the way we transport, but for these systems to operate effectively in their natural environment, they must be able to address a unique and complex set of challenges. Natural environments are significantly more unpredictable and diverse than relatively structured urban environments. These environments include not only diverse surfaces such as gravel, sand, and mud but also natural obstacles such as bodies of water and dense vegetation, each of which places different demands on autonomous vehicles' perception systems and navigational decision-making capabilities.

In such environments, the safety and efficiency of autonomous vehicles depend on high-precision environmental perception. The image captured by the vehicle camera can be segmented by careful semantic segmentation, which can not only distinguish which areas are drivable and which are not, but also identify various potential obstacles and hazards. For example, accurately identifying and classifying sandy and muddy areas in images is critical to preventing vehicles from getting stuck or coasting.

In addition, semantic segmentation technology is equally important for achieving dynamic interaction between vehicles and the environment. Driving in a natural environment, the vehicle may need to respond in real time to sudden natural events, such as the sudden appearance of animals or temporary puddles. By assigning precise category labels to each pixel, semantic segmentation not only enhances the autonomous driving system's understanding of the environment, but also greatly improves its ability to respond to external changes, thus ensuring both safe and efficient navigation.

Therefore, exploring and optimizing semantic segmentation technology for use in natural environments is not only the key to improving the performance of autonomous vehicles, but also an important step to promote the commercialization and popularization of autonomous driving technology. By continuously improving these technologies, we are able to provide more powerful tools for autonomous vehicles to cope with changing natural environments and achieve true global autonomy.

1. Correlative Research and Analysis

In the field of autonomous driving, semantic segmentation research is constantly advancing with the aim of improving the navigation ability and decision-making efficiency of the system in various environments. In addition to the previously mentioned U-net and Deeplabv3 models, there are several other models that demonstrate excellent performance in achieving efficient and accurate environment awareness:

1. FCN (Full Convolutional Network)： FCN (Full Convolutional Network) : By converting standard deep networks to full convolutional form, FCN enables end-to-end pixel-level prediction and makes a major breakthrough in semantic segmentation. Jonathan Long(2015) first proposed the FCN model in Fully Convolutional Networks for Semantic Segmentation. Liang-Chieh Chen (2017) compared the innovations of FCN and DeepLab model in using atrous convolution to improve semantic segmentation. (Long et al., 2015)

2. SegNet： Specifically targeted at video surveillance and road scenarios, SegNet effectively recovers image details, especially border areas, through its unique codec structure. Alex Kendall (2017) proposed Bayesian SegNet, which enables SegNet to provide a confidence estimate of its prediction when processing image segmentation tasks. This is particularly important for improving the reliability of automatic driving. (Kendall et al., 2015)

3. SAM (Spatial attention model)： Jun Fu (2019) et al. proposed the use of spatial and channel attention mechanisms to improve the accuracy of scene segmentation. By introducing spatial attention mechanisms, SAM model improves the recognition ability of important features in the scene, which is crucial for identifying key obstacles and road conditions in automatic driving. (Fu et al., 2020)

4. In recent years, with the development of technology, researchers are also exploring combining the characteristics of multiple models to solve specific problems in autonomous driving. For example, some research has focused on combining the precise detail recovery capability of U-net with the efficient context understanding capability of Deeplabv3, aiming to create a model that can process large images quickly while maintaining high segmentation accuracy.

1. Problems that Exist and Need to be Solved

Despite the progress made in this area, several challenges remain in the field of semantic segmentation for autonomous driving:

1. High variability of environmental conditions: Natural scenes have less structure and more variation, making it difficult to achieve consistent segmentation accuracy.

2. Real-time processing requirements: For autonomous driving, segmentation algorithms must be accurate and fast enough to support real-time decision-making. Many deep learning models, while accurate, struggle to meet the speed requirements of real-time processing. (Muhammad et al., 2022)

3. Limited annotation datasets: Most advanced models require large amounts of annotation data, which can be expensive and time-consuming to produce, especially for uncommon or highly specific environments.

4. Adaptability and ability to generalize: Many models perform well under training conditions, but struggle to generalize to new or slightly different environments without extensive retraining or fine-tuning.

# Methods

It is challenging to do semantic segmentation of images in natural environments due to the various lighting condition, coverage of vegetation and the presence of visually similar objects. To get over with these challenges, we used DeepLabV3 for image segmentation and different preprocessing steps and different techniques to improve segmentation accuracy.

## Preprocessing WildScene Dataset

1. Preprocessing K-03 Dataset: The K-03 dataset is recorded in Karawatha Forest Park, Brisbane, Australia during Dec 2021. The dataset contains 3914 images of 2016×1512 resolution, which were taken in natural environment where lighting conditions can vary significantly, which may cause shadow artifacts. To mitigate these issues, we applied the following preprocessing steps:

* Image Transformation: We have resized the images to 512×512 to standardize the input size and reduce the computational loads without losing much high resolution information. This resizing also ensures the aspect ratio still remains consistent.
* Image Normalization: We have normalized the images with mean and standard deviation(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]). This step is to ensure the images can have a similar distribution to the data used during the pretained model.
* Data Augmentation: We have applied different geometric transformations, such as rotations and flipping to increase the diversity of the training data to help the model to generalize better to those unseen data.

1. Mask Transformation: For the indexLabel images, we have applied the following transformations:

* Mask Transformation: We have resized the indexLabel images to 512×512 to keep the same size as input images, and we use nearest neighbour interpolation to avoid introducing new classes.
* The reason why we use nearest neighbour interpolation is that each pixel in the indexLabel image represents a integer class label rather than a continuous value. If we use bilinear interpolation, this may produce a non-integer class label which may result in incorrect class labels. And nearest neighbour use the nearest neighbour pixel class which will not alter the original class label.
* ToLabelTensor: We use a self-defined ToTensor class to ensure the mask values are correctly interpreted as integer class labels. Long type conversion is used to indexing in Torch and maximum value check can retrieves the maximum value in the tesnor to ensure there are no invalid class labels(19 in this case).

## Splitting WildScene Dataset

1. Dataset Splitting: The full dataset is split into training, validation, and test sets. The training set contains 70% of the full dataset, the validation set and test set contains 15% of the full dataset individually.

* Stratified sampling: We planned to use stratified sampling to ensure that each train, validation and test subset can maintain the same class distribution as the originial dataset, and can ensure proportional representation of each class in all subsets, which can lead to better model generalization when dealing with imbalanced dataset, however, we found that straitified sampling significantly increased the computational burden due to the large size of the dataset, as a result, we decided to use random splitting instead.
* Random Splitting: We use random split to divide the full dataset to 70% for training, 15% for validation, 15% for testing, the random split is simpler, faster and suitable for large datasets when the computational resources are limited. However, this may not maintain the class distributions as the original datasets and result in less representative subsets for imbalanced datasets.

1. Data Loading: We applied different dataloaders to train subset and val/test subsets, we use shuffle to ensure the model see the data in a different order in each epoch to avoid the model memorize the order of images to have a better generalization, and the validation and test dataloaders are not shuffled to ensure the order is consistent for evaluation.

## AutoDL Server

Due to the insufficient computing power of our local computers, training the model takes a lot of time, in order to train the models much more efficiently, we rented a server using AutoDL (<https://www.autodl.com/market/list>) for training and testing both DeepLabv3 and U-Net for this project.

1. In the server, we have the following configurations:

* A Docker image containing PyTorch 2.3.0, Python 3.12, and CUDA 12.1.
* GPU: RTX 4090D (24GB) \* 1
* CPU: 16 vCPU Intel(R) Xeon(R) Platinum 8481C
* Memory: 80GB
* Storage: Base: 50 GB

2. The server with the above configurations is sufficient for us to train and test the model, we have enough graphics memory for training purposes and 50GB of storage is enough for storing all the datasets.

3. We use ssh key & password pair in the local terminal to upload the datasets on the temporary repository on the server and remotely train the models on the server. It takes around 80 minutes to do preprocessing, training for 10 epochs, and testing on the server, which is much faster than doing these tasks on the local machine.

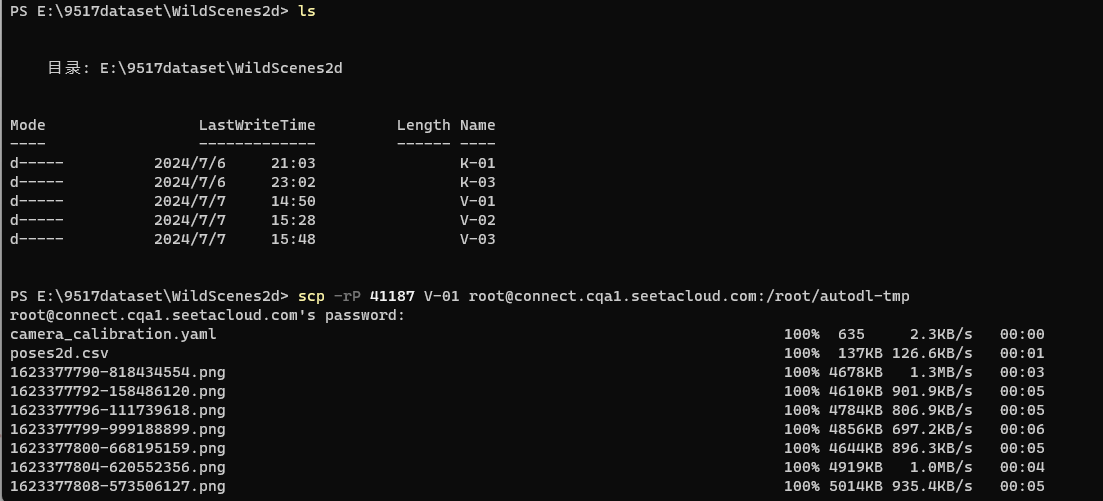


Figure 1: AutoDLserver dataset Upload

## Semantic Segmentation using DeepLabv3

1. Model Structure: We use the DeepLabv3 structure with a ResNet-101 backbone and a self-defined additional convolution layer. DeepLabv3 is a semantic segmentation architecture designed to capture multi-scale context by adopting multiple atrous rates[4].

* Atrous Convolution: Atrous convolution is useful in enlarging the view of filters without increasing the number of parameters. It enables the model to capture multi-scale information and context effectively by introducing holes into the convolution filters[4], which is useful in segmentation tasks where objects can vary greatly in size.
* Atrous Spatial Pyramid Pooling: Atrous Spatial Pyramid Pooling (ASPP) is a semantic segmentation module for resampling a given feature layer at multiple rates prior to convolution[5]. It consists of multiple parallel atrous convolutions with different rates, capturing features at multiple scales.
* Performance: The combination of atrous convolutions and ASPP in DeepLabv3 lead to high performance on semantic segmentation tasks, it can efficiently captures global context without sacrificing the fine details necessary for accurate segmentation.
* Implementation in Our Project: We utilized the DeepLabv3 model with ResNet-101 backbone, which can provide a strong feature extraction capability. Additionally, we implemented a custom classifier layer to enhance the model’s generalization ability. The classifier includes an additional convolutional layer followed by a ReLU activation function. It aims to refine the feature maps better before the final classification, which may be helpful in improving the segmentation accuracy.

## Semantic Segmentation using U-Net

1. Model Structure: We use the U-Net structure with customized convolutional layers, which is an architecture designed for image segmentation to efficiently capture multi-scale contextual information in images through an encoder-decoder structure and jump connections.

* Convolutional Blocks: Each convolutional block consists of two convolutional operations, each followed by a batch normalization and a ReLU activation function, which helps the model to expand the receptive field without adding too many parameters.
* Up-sampling and jump-joining: in the decoder part, up-sampling is performed using a bilinear interpolation method and the feature maps of the corresponding layers in the encoder are merged with the feature maps in the decoder, which helps in recovering more positional details and is especially important for the segmentation of objects of varying sizes.
* Performance: With progressive downsampling in the encoder, deep feature extraction in the middle layer, and progressive upsampling in the decoder, U-Net is able to achieve high performance in semantic segmentation tasks by efficiently capturing the global context without sacrificing details.
* Implementation in the project: We enhance the generalization ability of the model with a custom classifier layer. The classifier includes an additional convolutional layer followed by a ReLU activation function designed to better refine the feature map before final classification, which may help improve segmentation accuracy.

2. Function Implementation: when we implement the program, there are some important code segments that need to be mentioned:

* Define a convolution block (ConvBlock): Each convolution block contains two convolutional layers, each followed by a batch normalization and ReLU activation function.
* Define U-Net model (U\_net\_segmentation): Contains multiple encoder and decoder layers, as well as an intermediate layer that implements progressive downsampling and upsampling. Enhances feature recovery by adding the encoder's feature map to the decoder's feature map using jump connections.
* Define CustomSegmentationDataset: Used to load and transform images and their corresponding labels, supports reading image data from folders.
* Define transformation functions for images and labels (transform\_pair): Includes resizing, conversion to tensor and normalization.
* Define functions to calculate IoU (calculate\_iou): IoU (intersection to concatenation ratio) is a key metric for evaluating the performance of segmentation models.
* Test model (test\_model\_with\_saved\_weights): Loads the trained model and evaluates the model on test data to calculate the average IoU.

3. Program Execution:

* The program will first load the data set and check unique labels and print on the terminal.
* Then load the model if there exists a pre-trained .pth file in the local repository with the device of cuda.
* Before training the model, the program will check the batch shape, label shape, unique labels in batch, and total batches in the epoch. Then print all the results in the terminal.
* After all the steps above finished, the output will show in terminal as below:

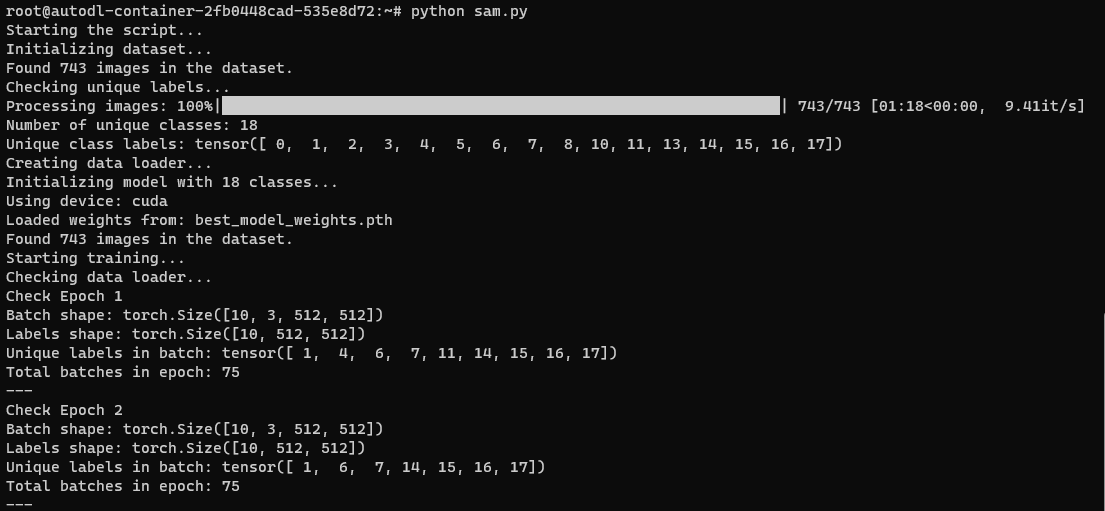


Figure 2: Preprocess steps of training

* The training will take 10 epochs, each time the program will load the weights from the input model\_path and train.
* Each time, the program will calculate the average loss for the current epoch by running\_loss / len(train\_loader).
* The program also embedded a model update verification function at the end of each epoch. It will output the Total parameter change in this epoch.
* When the epoch is 0, it means there will have no .pth file in the local repository, the program will save a “best\_model\_weights.pth” file locally using torch.save.
* Later the program will calculate the immediate IOU and mIOU for each epoch using test\_model\_with\_saved\_weights function.
* If the current mIOU is larger than the best mIOU which has been stored in local, the program will update the best mIOU by current mIOU and print the updated information.

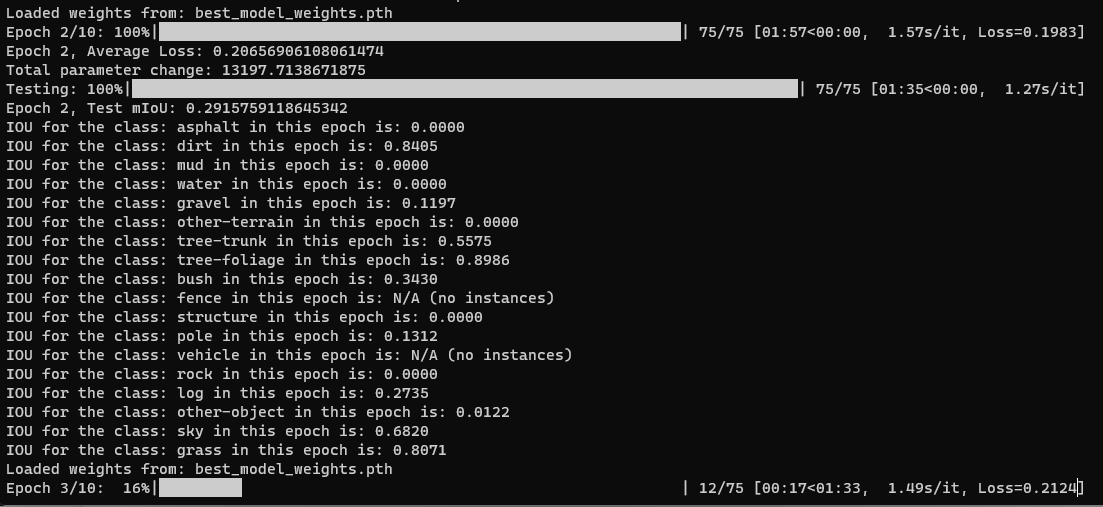


Figure 3: Training steps and immediate evaluation result

* At the end of the program, the best mIOU with the corresponding IOU for each class will be printed.

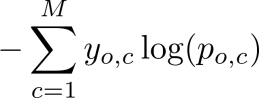
## Hyperparameters

1. Loss Functions: In our project, although Dice loss may have the potential for handling class imbalance, we observed a higher mean IoU with CrossEntrophyLoss, which resulted in the selection for our final model.

* Dice Loss: Dice loss is usually used for segmentation tasks[6]. It has advantage in imbalanced dataset and optimize the overlap between the predicted and actual segmentations.

wpsoffice

* CrossEntrophyLoss: CrossEntrophyLoss is useful in multi-class segmentation, it provides a clear probabilistic interpretation of the output.



1. Optimizers: In our project, we compared 2 optimizers which are Adam and SGD, and we choose Adam for our final model due to its efficiency and less requirement on learning rate tuning, which was beneficial given our dataset’s complexity and limited computational resource.

* Adam: Adam is an adaptive learning rate optimization algorithm that utilises both momentum and scaling, combining the benefits of RMSProp and SGD with Momentum[7]. The advantage of Adam is it is well-suited for problems with large datasets and parameters because it is efficient and require less computational resources.
* SGD: Stochastic Gradient Descent is an iterative optimization technique that uses minibatches of data to form an expectation of the gradient, rather than the full gradient using all available data.

1. Learning Rate Scheduler: We deployed a learning rate scheduler, which is ‘ReduceLROnPlateau’, it reduces the learning rate when a metric has stopped improving, in our project, it was to monitor the validation mean IoU and reduce the learning rate by a factor of 0.1 if there was no improvement for 3 epochs. This can lead to better model convergence by lowering the learning rate when the loss function and mIoU do not improve significantly.
2. Early Stopping: Early stopping can prevent the model from overfitting because it can stop the training process if the model’s performance on the validation set does not improve for a specified number of epochs, we set the patience to 3 in this project, this can help the model not learn noise and prevent overfitting.

# Experimental Results

This section presents an evaluation of the semantic seg-mentation model’s performance on the WildScenes2D data-set. We employ the Intersection over Union (IoU), also known as the Jaccard Index metric to quantify the model’s segmentation accuracy across various terrain and object cat-egories. The mathematical definition of IoU is shown as below:

Where:

A = Predicted region

B = Ground truth region

|A ∩ B| = Area of intersection between A and B

|A ∪ B| = Area of union of A and B

IoU helps us understand how well the predicted segmentation mask overlaps with the ground truth mask. A higher IoU indicates that the model's predictions align more closely with the true segmentation, while a lower IoU indicates that the predictions deviate significantly from the ground truth.

## Experimental Setup

Our Experiments are based on a multimodal dataset consisting of five sequences of 2D images recorded with a normal video camera during traversals through two forests: Venman National Park and Karawatha Forest Park, Brisbane, Australia. The working environment we used to train and test is Python 3.9. We compare the segmented images with the manually annotated (labelled) images by calculating IoU for each class and then taking the mean over all classes in the whole test set as qualitative and quantitative evaluation metrics.

## Evaluation of Each Class

The evaluation process is implemented using PyTorch, which is a popular deep learning framework. By calculating IoU for each class separately, we gain insights into the model’s performance through different semantic categories. This is particularly useful for identifying classes. The evaluation is performed in batches, which can realize efficient processing of large datasets can compatibility with GPU acceleration.

## Evaluation of Whole Model

The overall model performance is evaluated by using the mean IoU, which is the average of all class-wise IoUs. Class-wise IoU helps in assessing performance especially when classes are imbalanced in the dataset. Meanwhile, the mean Iou provides a single, interpretable value (between 0 and 1) that represents the model’s overall performance across all classes and use of Nan values ensures that the evaluation is robust to the absence of certain classes in some images or batches.

## Figures and Tables

#### The result after training 10 epochs on V-01 using U-Net:

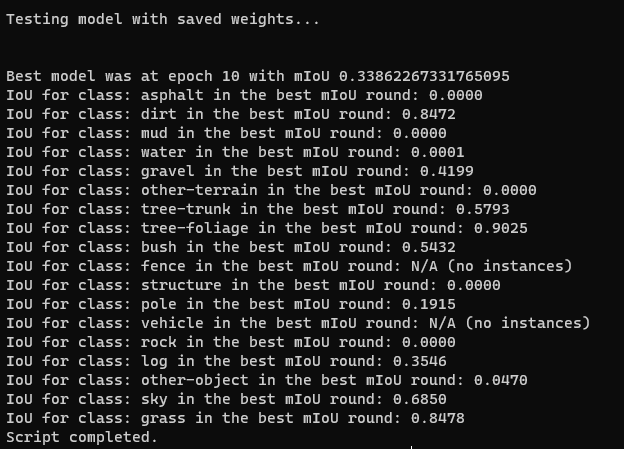


Figure 4:*result after training U-Net on V-01*

When testing U-Net model after training 10 epochs on dataset V-01, we can find the result below:

1. The mIoU is 0.3386
2. There are 6 classes who has IoU less than 4 digits of 0s (0.0000)
3. The highest IoU belongs to the tree-foliage who has 0.9025
4. The lowest IoU but larger than 4 digits of 0s belongs to pole which has 0.1915

#### The result after training 10 epochs on V-02 using U-Net:

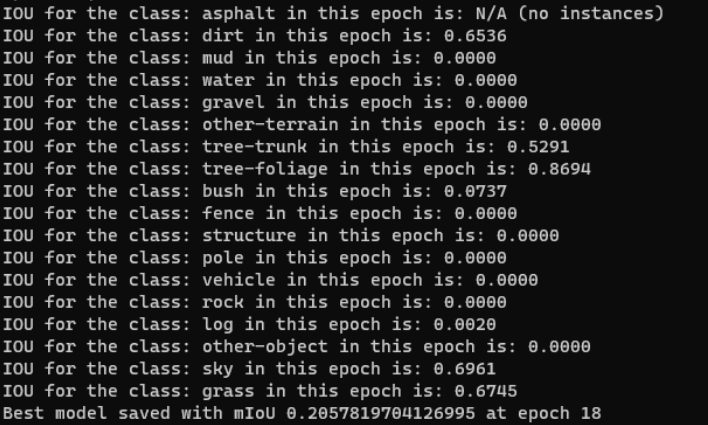


Figure 5:*result after training U-Net on V-02*

When testing U-Net model after training 10 epochs on dataset V-02, we can find the result below:

1. The mIoU is 0.2057
2. There are 10 classes who has IoU less than 4 digits of 0s (0.0000)
3. The highest IoU belongs to the tree-foliage who has 0.8694
4. The lowest IoU but larger than 4 digits of 0s belongs to pole which has 0.1915

#### The result after training 10 epochs on V-01 using DeepLabv3:

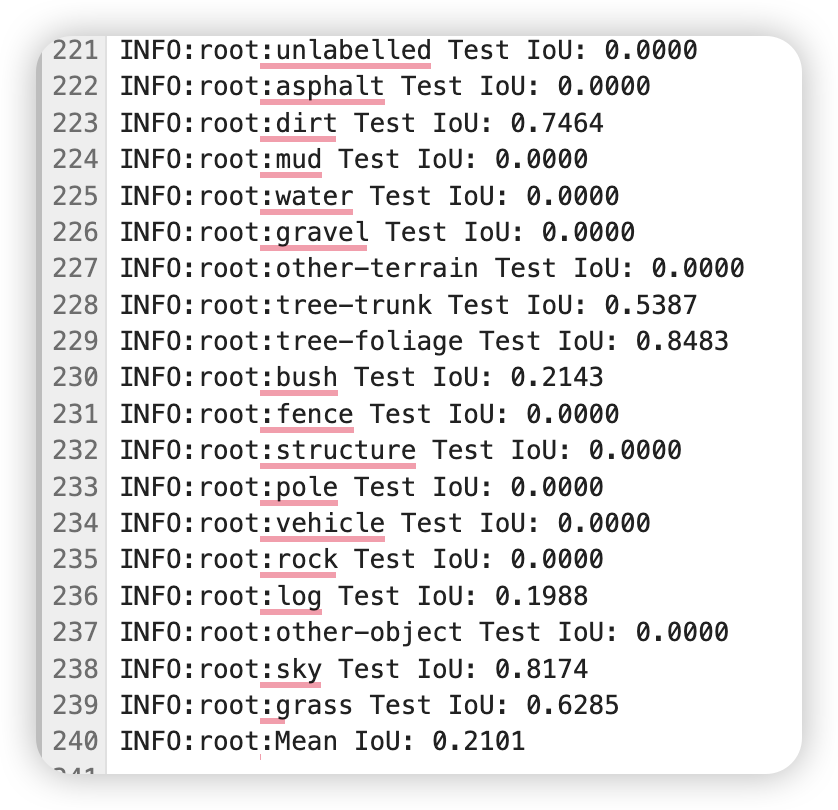


Figure 6:*result after training DeepLabv3 on V-01*

When testing DeepLabv3 model after training 10 epochs on dataset V-01, we can find the result below:

1. The mIoU is 0.2101
2. There are 10 classes who has IoU less than 4 digits of 0s (0.0000)
3. The highest IoU belongs to tree-foliage which is 0.8483
4. The lowest IoU but larger than 4 digits of 0s belongs to log which is 0.1988

#### The result after training 10 epochs on K-03 using DeepLabv3:

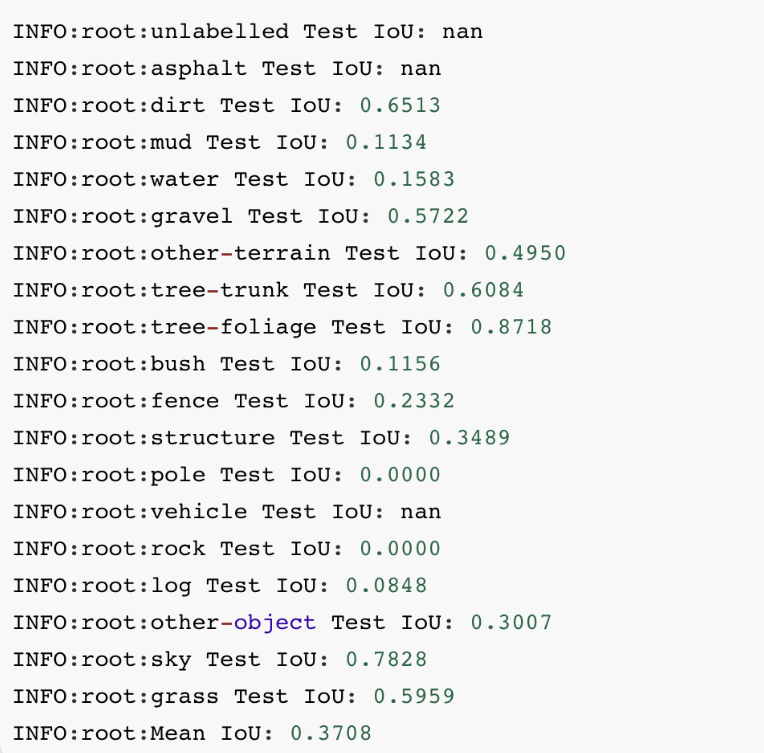


Figure 7:*result after training DeepLabv3 on K-03*

When testing DeepLabv3 model after training 10 epochs on dataset K-03, we can find the result below:

1. The mIoU is 0.3708
2. There are 2 classes who has IoU less than 4 digits of 0s (0.0000)
3. The highest IoU belongs to the tree-foliage who has 0.8718
4. The lowest IoU but larger than 4 digits of 0s belongs to mud which has 0.1134

*E. Class Distribution Analysis*

In the dataset WildScenes2d, we analyze and visualize the distribution of classes of labeled images. We count the images and pixels in each class and present detailed statistics about the class distribution including the number and percentage for each class, the total number of processed images, and the number of analyzed pixels. We create two bar charts to show the images and pixel distribution. This work can help us identify class imbalances that might affect model training, ensure adequate representation of all classes as well as interpret model performance in the context of data distribution.

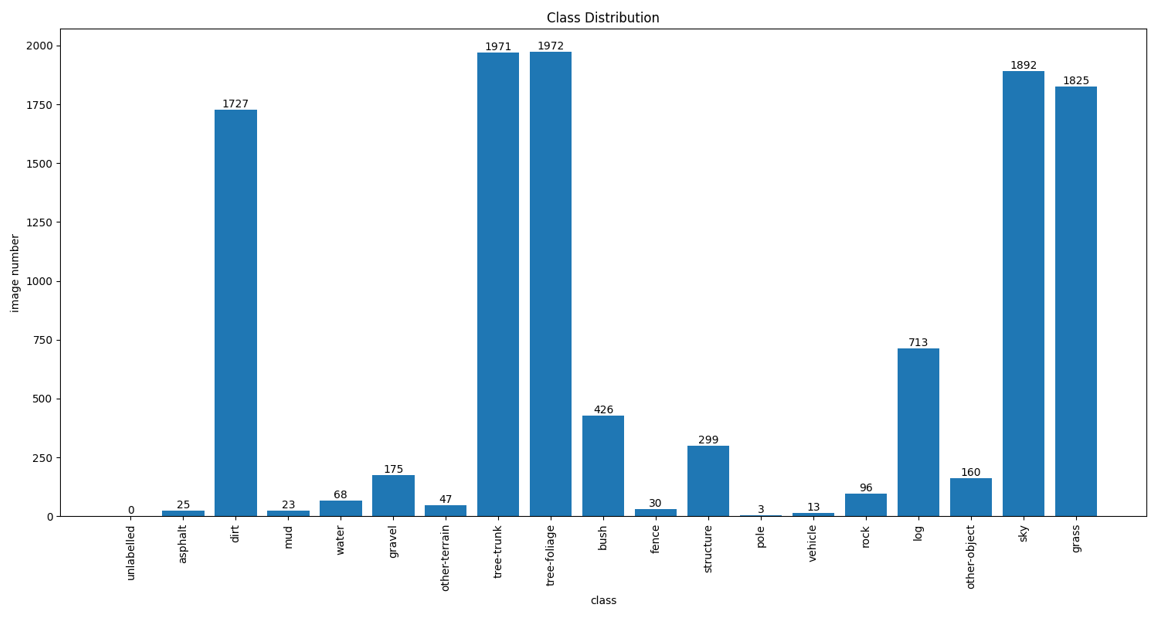


Figure 8: image distribution of K-01

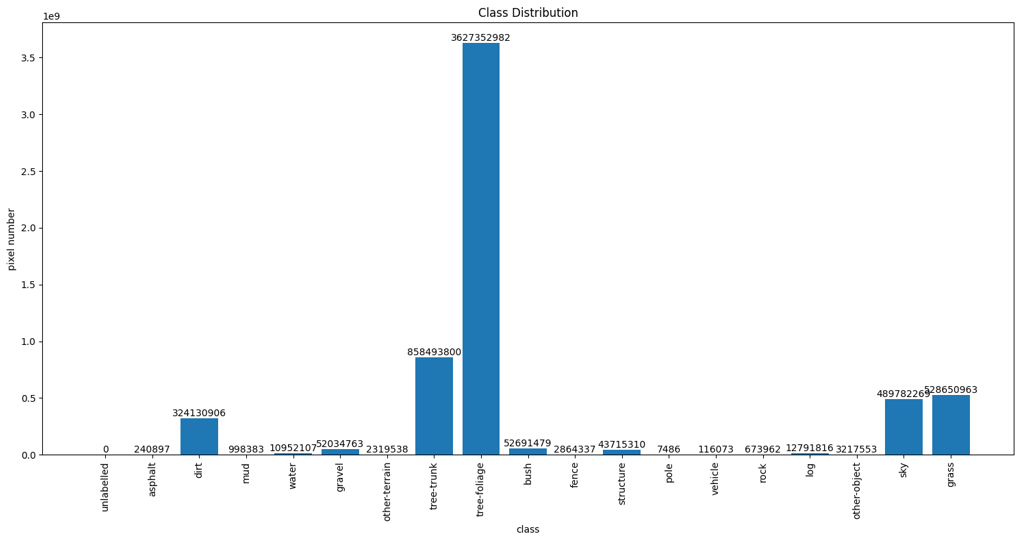


Figure 9: pixel distribution of K-01

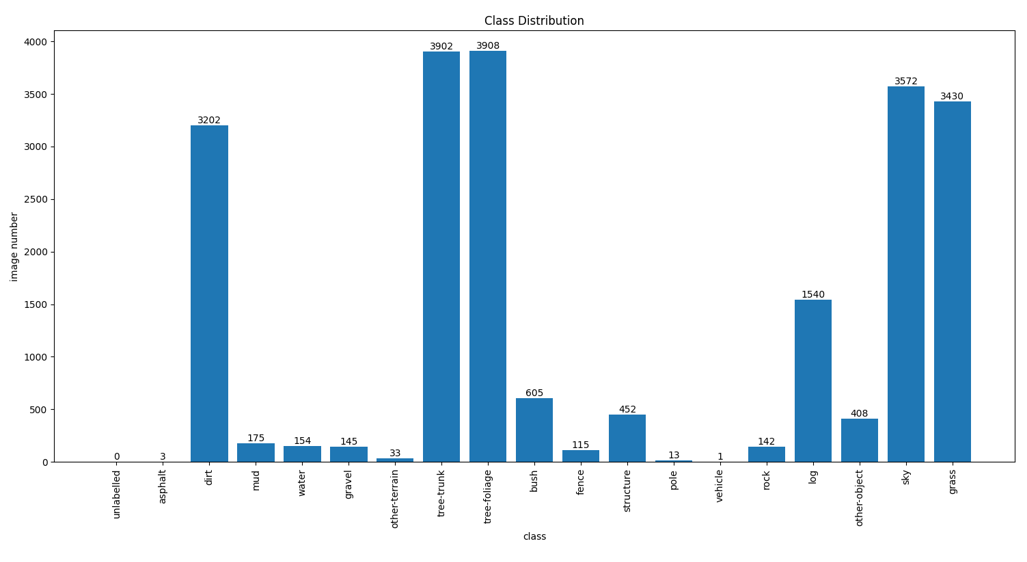


Figure 10: image distribution of K-03

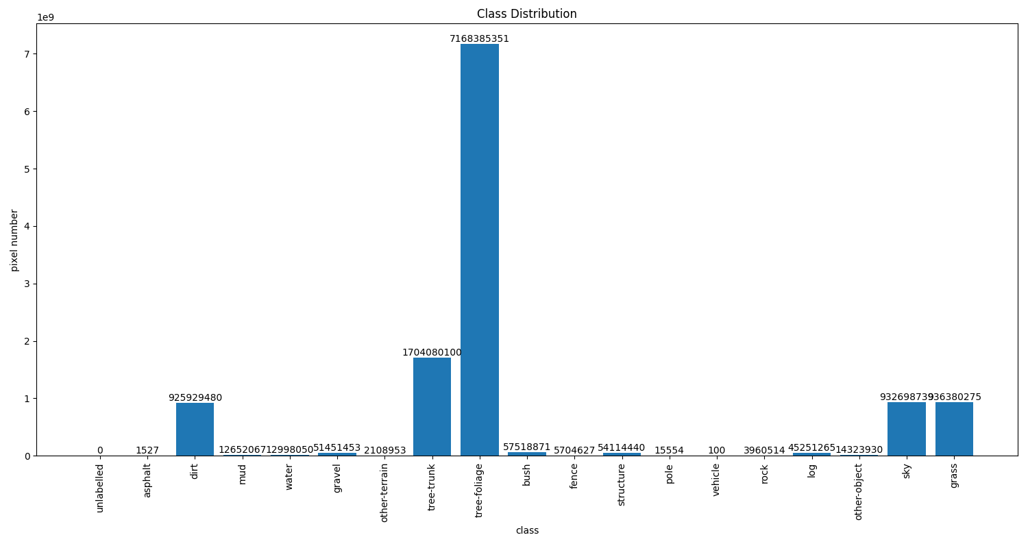


Figure 11: pixel distribution of K-03

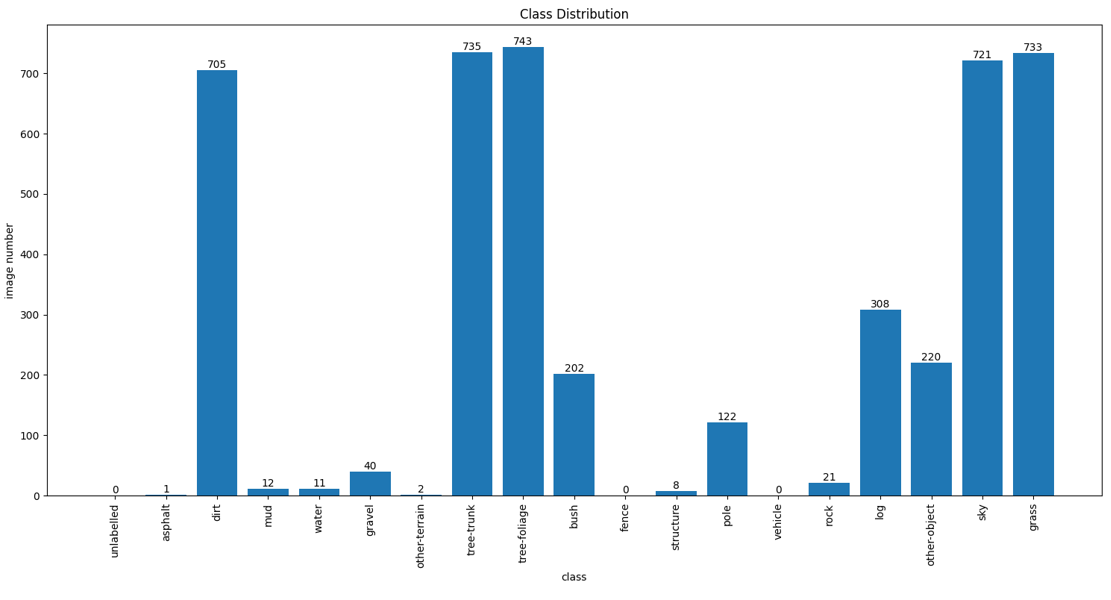


Figure 12: image distribution of V-01

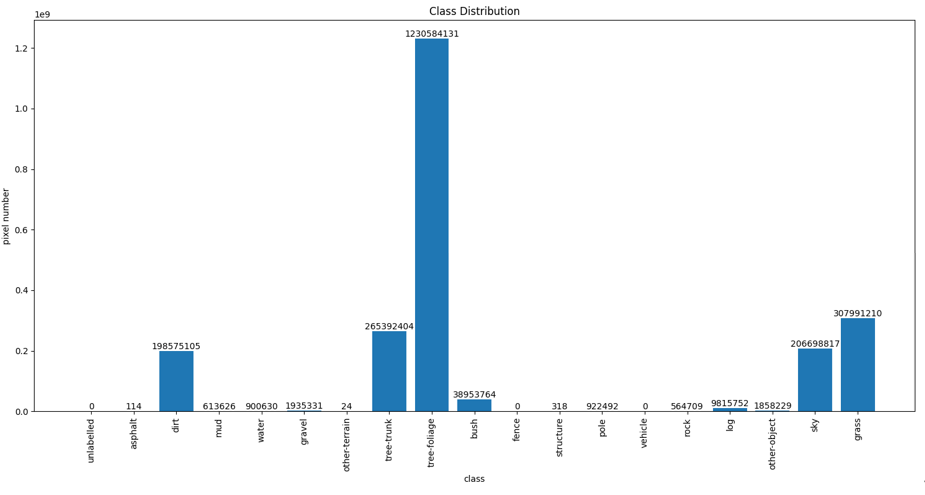


Figure 13: pixel distribution of V-01

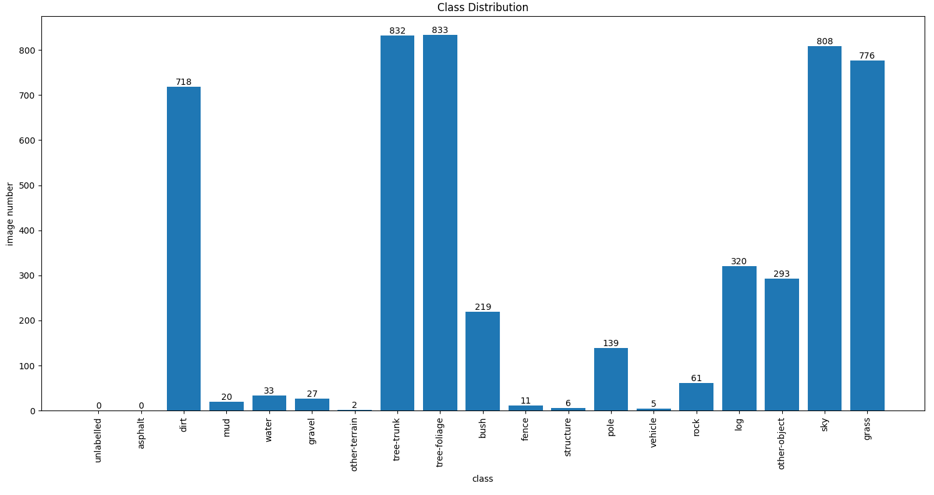


Figure 14: image distribution of V-02

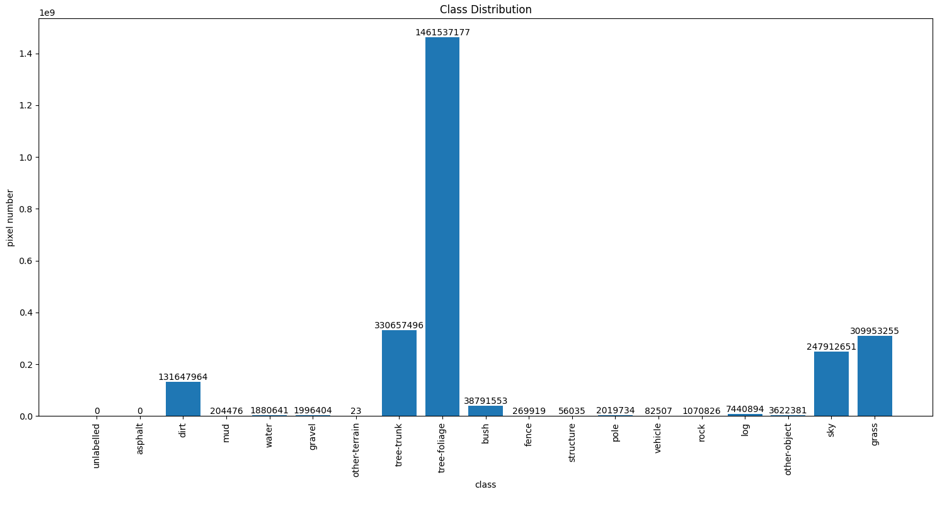


Figure 15: pixel distribution of V-02

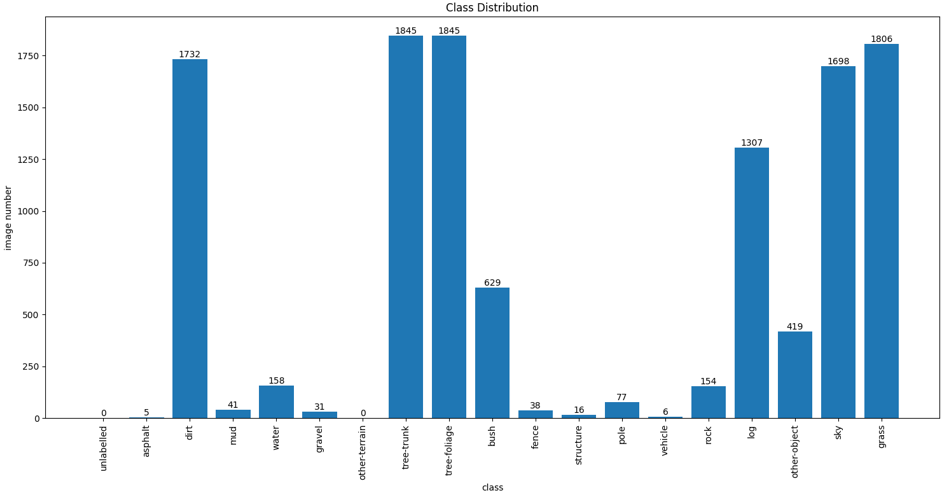


Figure 16: image distribution of V-03

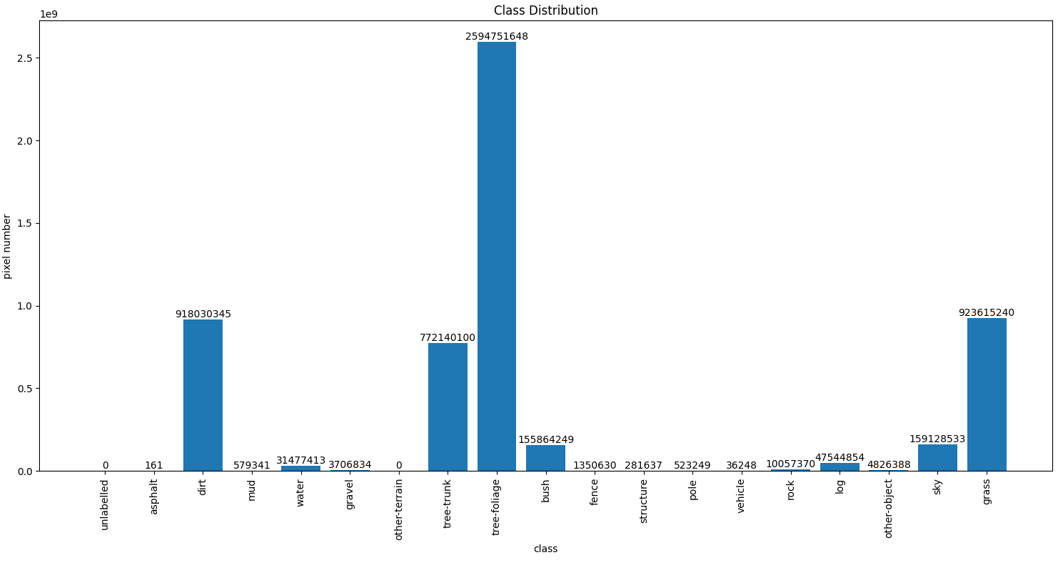


Figure 17: pixel distribution of V-03

The charts presented for datasets K-01, K-03, V-01, V-02, and V-03 provide valuable insights into class distribution patterns across both image and pixel levels. These visualizations can reveal dominant categories, highlight data imbalances, and showcase distribution variations among datasets. By comparing image-level and pixel-level charts, we can infer the relationship between overall image composition and detailed pixel representation for each class.

# Discussion

A. Performance Analysis of U-Net and DeepLabv3

Our experimental results demonstrate that U-Net outperformed DeepLabv3 in semantic segmentation for natural scenes. U-Net achieved a mean Intersection over Union (mIoU) of 33.86%, significantly higher than the 21.01% mIoU obtained by DeepLabv3. U-Net exhibited powerful performance in specific classes, notably 'tree-foliage' (90.25% IoU) and 'dirt' (84.72% IoU).

However, both models showed limitations in segmenting specific classes. The performance was notably poor for classes such as 'fence,' 'mud,' and 'gravel.' We hypothesize that this could be attributed to the dataset's limited representation of these classes or their visual similarity to other, more prevalent classes.

B. Comparison with WildScenes creators' methods

While our best performing model, U-Net, achieved a promising mIoU of 33.86%, it falls short of the 47.10% mIoU achieved by the WildScenes dataset creators using Mask2Former with a Swin-L backbone. The Mask2Former (Swin-L) approach demonstrates several key advantages over our U-Net implementation, including:

1. A transformer-based architecture
2. A sophisticated Swin-L backbone
3. Better handling of multi-scale features

C.Reasons for existing failures

Our analysis reveals some reasons for the performance limitations of our models:

a) Class imbalance: Poor performance on classes like 'fence' (Both 0% IoU) and 'mud' (Both 0% IoU) can be largely attributed to their underrepresentation in the dataset.

b) Visual ambiguity: Classes such as 'gravel' (U-net 41.9% IoU, Deeplabv3 0% IoU)and 'other-terrain' (Both 0% IoU) may share visual characteristics with more prevalent classes like 'dirt,' leading to misclassifications.

c) Limited model capacity: U-Net's struggle with less frequent classes suggests it may lack the capacity to capture the full complexity of the natural scene data.

d) Insufficient training time: Computational constraints may have limited the number of training epochs, particularly affecting less represented classes.

e) Feature extraction limitations: Low performance on classes like 'structure'(Both 0% IoU) and 'rock'(Both 0% IoU) suggests ineffective feature extraction for these classes.

D. Future Improvements

To address these challenges and improve our models' performance, we propose the following future directions:

a)Increase training time and utilize more powerful computational resources for better model convergence and more extensive hyperparameter tuning.

b) Enhance learning from rare classes, possibly through data augmentation techniques.

c) Explore the combination of 2D image and 3D point cloud information for more comprehensive scene understanding.

d) Implement more powerful network structures, such as the Swin Transformer.

In conclusion, while our U-Net implementation showed promising results, there is significant space for improvement. The challenges encountered highlight the complexity of semantic segmentation in natural environments and underscore the need for more sophisticated approaches and careful parameter tuning to achieve robust performance across all classes.

# Conclusion

1. Overall performance on the WildScenes dataset:

Our semantic segmentation models have shown good potential on the WildScenes dataset, especially in handling the main elements of complex natural scenes. While the performance on some challenging categories is not so good, the overall results provide a strong base for more research in this area.

1. Most successful and innovative aspects of our approach:

Our method has done well in balancing model complexity with computational efficiency. Specifically, our models have shown that they can effectively capture important semantic features of natural environments with limited resources, which is very important for real-world applications.

Limitations and areas for improvement:

Despite our achievements, we still have problems in handling very unbalanced datasets. Our approach needs more optimization to improve recognition accuracy for important but rare environmental features, such as specific man-made structures or uncommon terrain types.

1. Potential future research directions:

Develop adaptive learning algorithms specifically for natural scene features.

Explore cross-modal learning techniques, combining visual and other sensor data.

Investigate lightweight but efficient network architectures for real-time processing.

Design data augmentation techniques specifically for natural environments

Explore semi-supervised or self-supervised learning methods to make better use of unlabeled data

1. Potential impact on autonomous vehicle navigation in natural environments:

Our research opens new ways to improve the adaptability of autonomous driving systems in unstructured environments. By improving the recognition of natural elements, our work could expand where autonomous driving technologies can be used, allowing safe navigation in complex wild settings.

To sum up, while our current work provides a useful action in semantic segmentation for natural scenes, it also shows how complex this task is and the need for more research. The challenges we faced and the insights we gained paved the way for more advanced, robust, and adaptable systems that can understand and navigate the complexities of natural environments. Although we have made progress, there is still much to learn and improve in this exciting field of study.

# References

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use “Ref. [3]” or “reference [3]” except at the beginning of a sentence: “Reference [3] was the first ...”

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors’ names; do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

1. G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*
2. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
3. I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
4. Chen, Liang-Chieh, et al. "Rethinking atrous convolution for semantic image segmentation." arXiv preprint arXiv:1706.05587 (2017).
5. Chen, Liang-Chieh, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." IEEE transactions on pattern analysis and machine intelligence 40.4 (2017): 834-848.
6. Sudre, Carole H., et al. "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations." Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, Proceedings 3. Springer International Publishing, 2017.
7. Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).
8. J. Long, E. Shelhamer, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” IEEE, 2015. Accessed: Jul. 28, 2024. [Online]. Available: http://luthuli.cs.uiuc.edu/~daf/courses/MAAV-2019/SemanticSeg/long\_shelhamer\_fcn.pdf
9. A. Kendall, V. Badrinarayanan, and R. Cipolla, “Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding,” *arxiv.org*, Nov. 2015, doi: https://doi.org/10.48550/arXiv.1511.02680.
10. J. Fu, J. Liu, Y. Li, Y. Bao, Z. Fang, and H. Liu, “Dual Attention Network for Scene Segmentation,” *IEEEXplore*, Jan. 09, 2020. https://ieeexplore.ieee.org/abstract/document/8953974
11. K. Muhammad, T. Hussain, H. Ullah, J. Del Ser, and M. Rezaei, “Vision-Based Semantic Segmentation in Scene Understanding for Autonomous Driving: Recent Achievements, Challenges, and Outlooks,” *IEEEXplore*, Oct. 06, 2022. https://ieeexplore.ieee.org/abstract/document/9913352