COMP9517: Computer Vision

A report in Semantic Segmentation Methods for Autonomous Vehicle Navigation in Natural Environments

UNSW AUSTRALIA

Team vison_group6

z5098152 Muxin Qiao

z5261247 Jingwen Yang

z5408868 Po-Hsun Chang

z5438698 Xuewei Zhuang

z5514915 Tianyi Dou

2024 Term 2

1. Introduction

The safety and accuracy of autonomous vehicle navigation in natural environments rely heavily on their ability to recognize and respond to various scenarios and objects they may encounter. Due to the complexity, unscriptural and unstructured formation found in natural environments, fine-grained semantic segmentation of images captured by the vehicle's cameras is essential for scene understanding. The pixel data extrapolated from the images captured will allow the determination of its natural surroundings including and not limited to terrain data, road conditions and surrounding vegetation. This project aims to develop and compare different computer vision methods for semantic segmentation in these challenging natural environments.

Given the complexity and variability of natural terrains, our approach focuses on developing a model that is capable of accurate segmentation. Natural environment contains a large number of artefacts. Development a model that can accurate segmentation under the natural terrains that is complexity and variability can be challenging. Therefore, we utilised two advanced deep learning methods which is U-Net and Deeplabv3 along with implementing tradition computer vision techniques to try address these challenges. U-Net was complemented under Tensorflow and Pytorch to see which have better performance. As such, by combing tradition methods with advanced deep learning approaches, a hybrid system is created to try maximises segmentation accuracy and efficiency. These methods both aim to help improve the vehicle's ability to navigate safely through diverse and unpredictable natural landscapes. Ultimately, this will contribute to the broader application of autonomous systems in various highly populated environments or environments contain many artefacts.

WildScenes dataset's characteristics is another area that was investigated and analysed. The dataset contains 9,301 2D images of size 2,016 x 1,512 pixels capturing various natural environments. Due to its considerable size, processing the entire dataset would be computationally, resource intensive and time-consuming. Therefore, utilising data sampling, certain subsets of images were selected as representatives of the overarching dataset. This process involved analysing the distribution of different classes and features within the dataset and selecting the least commonly distributed artefacts and utilising these landmark artefacts to further concatenate and analyse to ensure that our subset captures the diversity, complexity and uniqueness of the original dataset.

2. Literature Review

Semantic segmentation is done by labelling each pixel of the image according to the object class it belongs to. It is considered as the process of pixel-label image recognition and a further improvement of image classification ([1] Hao S.,2020). As image classification only assigns a single label to an entire image, while semantic segmentation involves dense pixel-wise classification base on the model provided.

DeepLabv3 was utilised due to its performance of semantic segmentation tasks. DeepLabv3 primarily improves in Atrous Spatial Pyramid Pooling (ASPP) and the use of atrous convolution and as such is utilised to refine the ResNet model. The ASPP in DeepLabv3 integrates image-level feature connections, a 1x1 convolution, and three 3x3 atrous convolutions with varying rates. This kind of unique structure allows the model to capture multi-scale contextual information effectively. Batch normalisation will apply on each parallel convolution layer to helps improves stability and performance ([2] Chen L C., 2018). By implementing Deeplabv3 along with cascaded ResNet module directly to intermediate feature maps rather than the final confidence maps, it can enhance the model's ability to capture detailed features at multi scales ([3] Chen L C., 2017).

U-Net was also found to be instrumental in completing dataset analysis. It is one of the earlier developed image segmentation models utilised in computer vision analysis. U-Net is constructed using a dual-path architecture, consisting of a contraction path (encoder) and an expanding path (decoder). These components enable U-Net to capture all the context effectively and achieve precise localisation. The encoder path is constructed by a traditional stack of convolutional and max pooling layers that is designated to down sample the input image and extract high-level

features from it. The decoder path shares the similar pattern as the encoder path, using transposed convolutions to up sample the features and refine the segmentation output. This can allow U-Net to be an end-to-end fully convolutional network (FCN) that is capable to accepting any size images without needing the dense layers ([4] Ronneberger O., 2015). Base on the previous works that have built in convolution networks, U-Net approach have showcasing it multi-functionality in different applications.

We realised that U-Net can effectively integrate both global and local features of the image and allows for comprehensive analysis. After each convolutional layer, feature maps are concatenated with subsequent layers, directly transforming information to the corresponding decoder layers for further analysis. In the initial convolutions, the feature maps retain high-resolution details of the image (textures, specific shapes, roughness, etc) which is extremely helpful as the details contribute to filtering the natural environment images. As the convolution and pooling operations progress, the feature maps at the deepest layer of the U-Net capture the global context of the detail image, including the distribution of the image label classes. Through successive up sampling layers, the information from different levels will merge and allow for multi-scale information fusion and resulting in accurate segmentation outputs.

3. Methods

3.1 Data Sampling

There are numerous approaches for data sampling that could be utilised for our application. However, the chosen sampling techniques provides a more unified and methodical approach. This method involves filtering through the original 9,301 images using colour information to establish 19 total classes. These classes allow us to identify and effectively count various artefacts present in the dataset. The images' colour information was then analysed to effectively differentiate between various key elements. These elements are then filtered, concatenated, and annotated to generate a frequency distribution table, shown in Figure 1, to illustrate the distribution of these classes

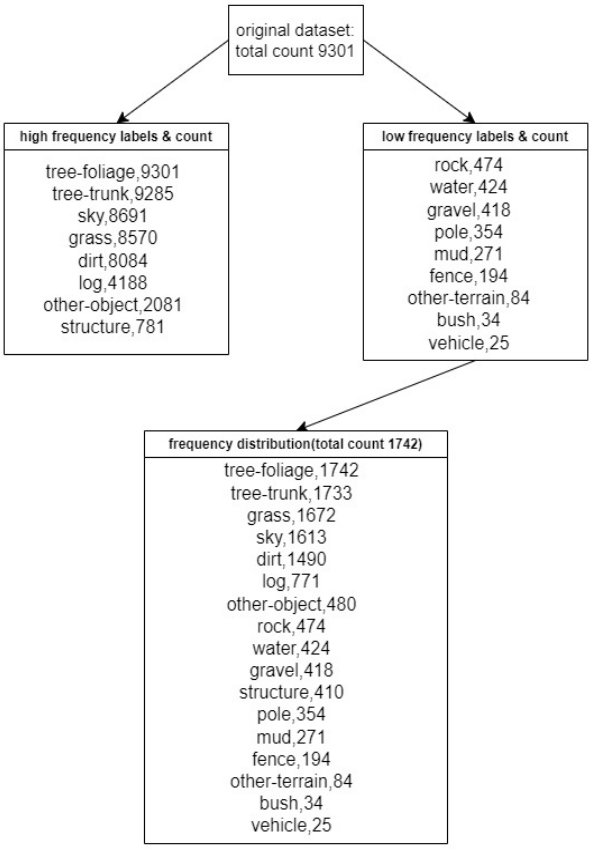

Figure 1. Data types of distribution

As per Figure 1, the number of images containing different colour information and classes vary incredibly with some only appearing a miniscule number of times (low frequency labels- less than 500 times) to appearing in most images (high frequency labels). Due to the uniqueness of the low frequency labels, 9 classes were focused on primarily. From these 9 classes, 1742 subset images were then further analysed and separated into the previously established 19 classes. These 1742 images are the main images that were focused on.

3.2 Data Preprocessing

Grayscale mask preprocessing was utilised by first converting the images to grayscale. By accomplishing this, the images are simplified, and the computational load/time is reduced. It also allowed for focus to be shifted onto intensity variations rather than colour. Each pixel in the grayscale image represents the intensity value, which is essential for the analysis of specific artefacts. Then, the cross-entropy loss function was implemented to the data preprocessing in line with multi-class segmentation and loss function

selection. The images were resized to 128x128x3 using RGB channels and the weight and height of the images CNN layer were integrated to trying to minimise the time spent on training. Label preprocessing was then conducted by encoding each label image to a specific colour map with corresponding colour palettes. The dimensions of encoded label images were resized to 128x128x1 before the comparison with colour palettes (categorised into 19 classes). After the predicted mask being generated, the image was then decoded back into RGB channels. A PIL package was also implemented to help process data as an import function.

3.3 Hyperparameters

Iterative training was implemented to optimise the normal hyperparameters (learning rate and epoch size). A learning rate of 0.001 was selected to ensure the fastest, most stable approach was utilised. Given the dataset size of 1,742 samples, and after splitting the data into training and validation sets with an 80/20 ratio, the training set was comprised of 1,393 samples. To further optimise the batch normalisation the batch size was chosen such that each batch calculation is efficient and balanced. As such, the batch size was the divisor of the training set to avoid batches with a single remainder sample. The normalisation process can be disrupted by this and hence, reducing the model performance. A batch size of 10 was selected as under the consideration of the need for computational efficiency and constraints that is imposed by batch normalisation. The selected batch size will strike a balance between the processing and the requirements from batch normalisation. Therefore, this ensures that the batches can be evenly divided and result in stable training dynamics and optimal utilisation of computational resources. In result, the training process will remain both effective and efficient.

3.4 Deeplabv3+ResNet

Deeplabv3 followed the traditional encoder-decoder structure to complete the semantic segmentation problem (Chen L C., 2018). In the down sampling procedure, it concatenates through 5 convolution layers to get one result layer. In the decoder procedure, using this generated layer to concatenate the original image after one convolution layer to complete the first up sampling procedure. By doing this, a predicated mask is generated.

3.5 U-Net

Similarly, the U-Net method had also utilised the ResNet structure for it encode. It is taking the advantage from pre-train weights to extract rich feature representations. There is total 5 stages of the encoder path from the ResNet architecture. The stages start with an initial sequence of 3 convolution layers, then followed by a progressively deeper convolutional block. The capture of high-level features and reduction in spatial dimensions in down sampling process have happened at the same time. Similarly to the encoder structure, the decoder path will progressively restore the spatial dimensions of the features maps through the utilisation of the up-sampling layers. Through each up-sampling stage, the features will be concatenated with corresponding encoder features to ensure the spatial information is preserved throughout the network. The resolution of the feature maps increased with the employ of decoder up sampling layers and the convolution blocks will then refine their features.

We utilised a loss function that integrates Dice loss and Cross-Entropy loss to enhance segmentation performance. This approach leverages the benefit from both losses, promotes accurate pixel classification and ensures good overlap between predicted and true segmentation. During the forward pass, the function adjusts the dimensions of the target labels and converts to the appropriate data type. The losses for predicted label images and original label images are computed and the sum of both losses will return. This can make sure the model has beneficial from the pixel-wise accuracy and interspace overlap accuracy from both losses. Once the targe labels ensure its dimensions is correct through the calculation of dice loss, it will then be converted into a one-hot encoded format to match the shape of the predicted label images. The predicted label images will then process using SoftMax function to obtain probability distribution. The overlap area between the predicted label images and the original label image will be measured to calculate the dice coefficient. The dice loss will then derive from the dice coefficient and

hence, it encourages the model to maximise overlap.

3.6 Evaluation Metrics

To evaluate the performance of this method, we use the Intersection over Union (IoU) metric which measures the similarity between the predicted mask and the ground truth. The IoU is calculated using the Jaccard index (Figure 2). The mean IoU is derived by averaging the IoU values across all classes.

$$Jaccard = \frac{Intersection(A,B)}{Union(A,B)}$$

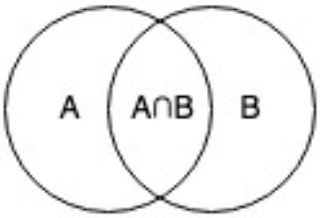

Figure 2. Jaccard Index

In this context, TP (True Positive), FP (False Positive), and FN (False Negative) denote the number of pixels that are correctly predicted, incorrectly predicted, and missed, respectively. C represents the total number of classes, and IoU_c is the IoU for class c ([6] M. D. Sulistiyo et al., 2018).

$$IoU_C = \frac{TP_c}{TP_c + FP_c + FN_c}$$

$$mean\ IoU = \frac{1}{c}\sum_{c} IoU_c$$

3.7 Demo Software Implementation

Due to submission file size limitations, the training checkpoint file has been uploaded to Google Drive. When running the demo software, it will automatically download the best model checkpoint file. Once the download is complete, it can import any images from the WildScenes2D dataset to check the predicted mask. The software interface, implemented using Python's Tkinter, includes a 'Clear' button that allows clearing the current result and importing a new image for prediction ([6] Python Software Foundation).

4. Results

4.1 Deeplabv3 + ResNet

In Figures 3 and 4, the training and validation losses stabilise after 12 epochs. The difference between the training loss and validation loss remains around 0.17, suggesting a minor overfitting issue. As the training loss decreases, the training IoU increases, indicating that the cross-entropy loss effectively captures the necessary information for multi-class segmentation problems.

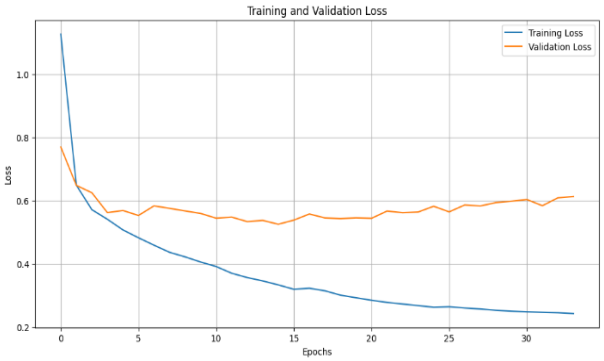

Figure 3. Training and Validation Loss for DeepLabV3

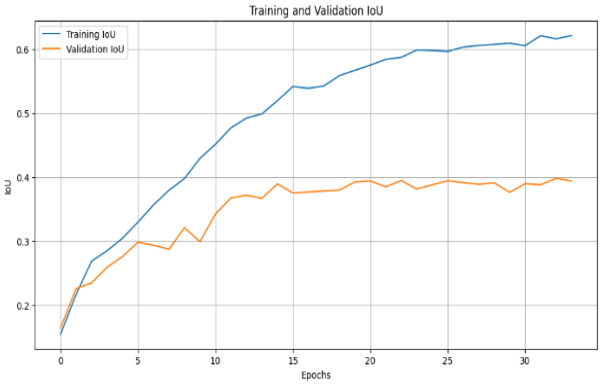

Figure 4. Training and Validation IoU for DeepLabV3

The effect plot for the DeepLabv3+ResNet model (Figure 5) results shows three images from the test loader. The model performs well in predicting large areas accurately; however, thin trees and tiny leaves lead to significant overlaps, indicating challenges in segmenting finer details.

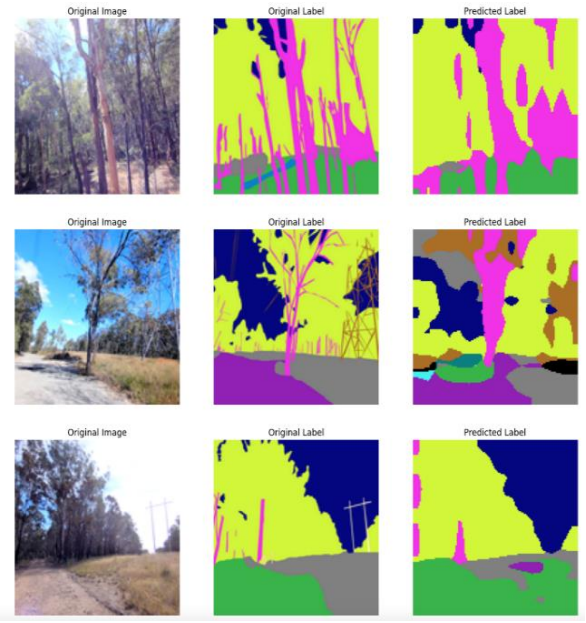

Figure 5. Effert Plot of DeepLabV3

4.2 U-Net

We used Pytorch and Tensorflow to execute the U-net Method.

- Pytorch

  With the use of Pytorch, during the first 10 epochs, both the validation loss and training loss trended towards 0.8, and the gap between the two losses gradually decreased. However, the training loss experienced a slight increase around epoch 8(Figure 6). The training IoU improved from 0.25 to 0.37 during the same period. Similarly, the validation IoU showed a slight upward trend but also experienced an unusual dip around epoch 8 (Figure 7).

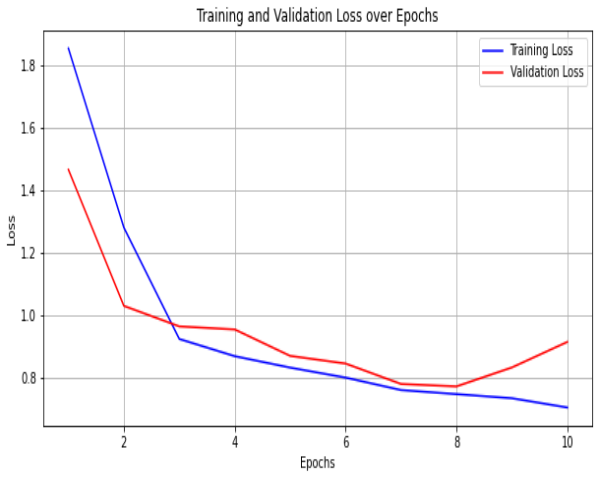

Figure 6. Training and Validation Loss for U-Net (Pytorch)

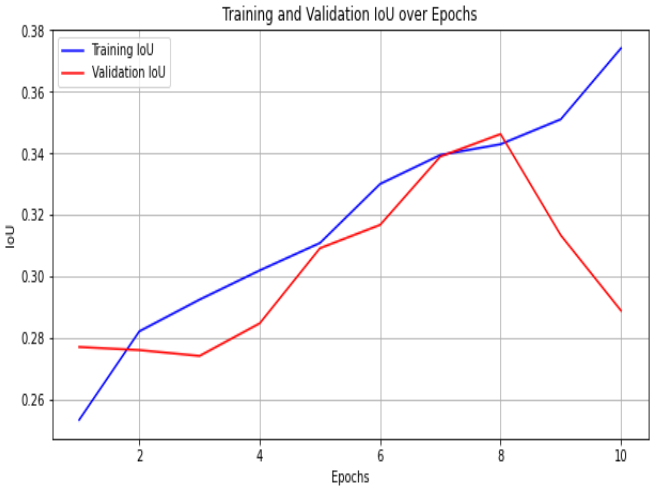

Figure 7. Training and Validation IoU for U-Net (Pytorch)

The predicted label image in effect plot for U-Net (Pytorch) shows approximately 70% similarity with the original label image (Figure 8).

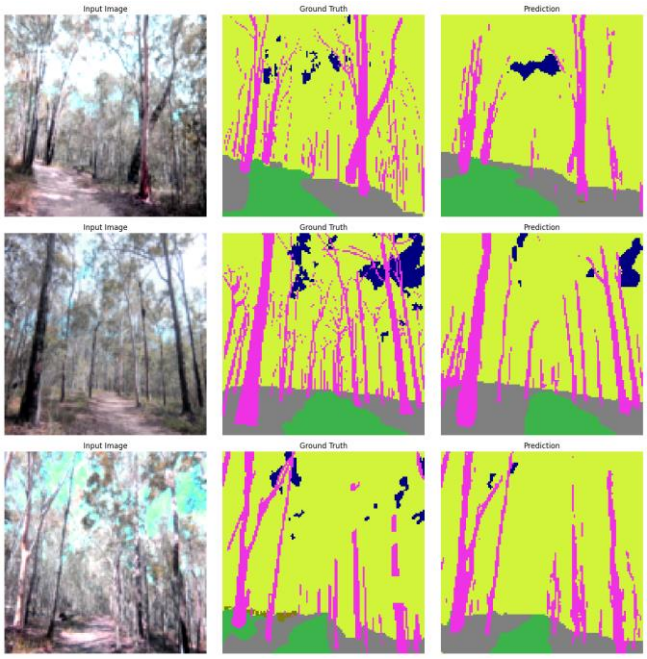

Figure 8. Effert Port of U-Net (Pytorch)

- Tensorflow

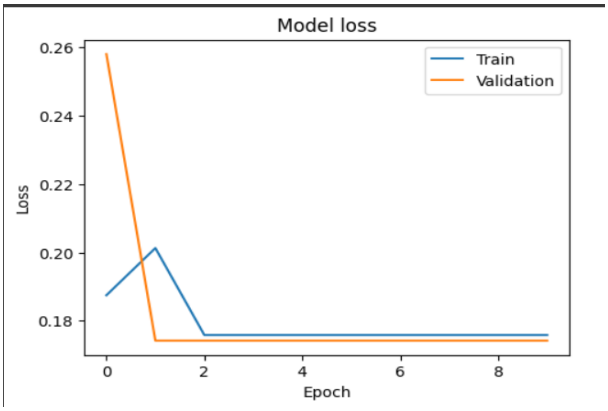

Figure 9. Training and Validation Loss for U-Net (Tensorflow)

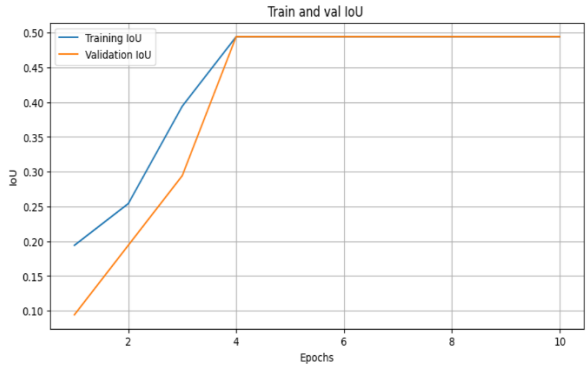

Figure 10. Training and Validation Loss for U-Net (Tensorflow)

5. Discussion

5.1 Analysis of multi-classes segmentation

The overfitting problem that occurred in the Deeplabv3 model could be caused by a variety of factors. One of the main factors could be that the natural environment is too complex as it contains many artifacts. A large amount of variability could make it difficult for the model to handle. The encoder structure can generate more than 100,000 features by capturing a large number of details and patterns. In addition, since we did not include any dropout layers to eliminate less relevant features, this could cause noise to be mis-segmented as key categories. Hence, the noise and irrelevant features could be retained the overemphasized in the model.

Another potential reason for overfitting could be the lack of merging of similar categories. Since some similar categories in our dataset (e.g. asphalt and other terrain types) are distinct, it increases the complexity of the model. However, if we combine similar categories, the segmentation task can be simplified and make the model unable to distinguish between different parts correctly. This is because the generalization ability of the model can be improved by ensuring that the model can learn more and fewer category-specific features.

Additionally, due to time constraints, we were unable to perform extensive parameter tuning of the weighted loss. It is important to set an appropriate weighted loss to penalize high frequency pixels while rewarding low frequency pixels in our balanced dataset. This helps address the class imbalance problem and ensures that the model is not biased towards more frequent classes. This could severely limit our current approach.

5.2 Analysis of U-Net model (Tensorflow & Pytorch)

The performance of the U-Net model under Pytorch shows that over the first 10 epochs, the loss gradually decreases and the IoU steadily increases. One possibility is that the validation set encounters relatively more challenging examples during this epoch, causing a temporary drop in performance. This could include images with complex features or less common artifacts that the model has not yet learned to accurately segment. It could also be an indication that the model begins to overfit the training data at this point, where the model begins to memorise the training data rather than learning to generalise from it. This raises concern because it could limit the model's ability to perform well on unseen data. Overfitting can be caused by the model excessively memorizing specific features and noise in the training set and failing to learn the underlying patterns.

However, the U-Net model's performance does not appear as more significant with Tensorflow (Figure 9 & Figure 10). The training loss remained consistent around 0.17, while the validation loss was slightly lower at 0.17. This indicates that the model is not overfitting and is performing similarly on both training and validation datasets. The Union (IoU) is approximately 0.49 both for training and validation. As a result, the U-Net model can be seen to be more relevant for segmentation tasks as

it measures the overlap between the predicted and ground truth masks. The IoU result of being around 0.494 indicates that the model is moderately effective at segmenting the images, but there is potential for further optimization and improvement.

5.3 Comparison

For the models that we have implied, this section will provide a comprehensive analysis of their performance metrics, including accuracy, training time, inference time, convergence by referring to the dataset's loss and Intersection over Union (IoU). By examining these metrics in detail, we can better understand the strengths and weaknesses of each model.

	DeepLabv3	U-Net (Pytorch)	U-Net (Tensorflow)
Training Loss	0.24	0.34	0.17
Validation Loss	0.6	0.83	0.16
Training IoU	0.62	0.55	0.49
Validation IoU	0.39	0.40	0.49

Figure 11. Loss and IoU of DeepLabv3, U-Net with Pytorch and Tensorflow

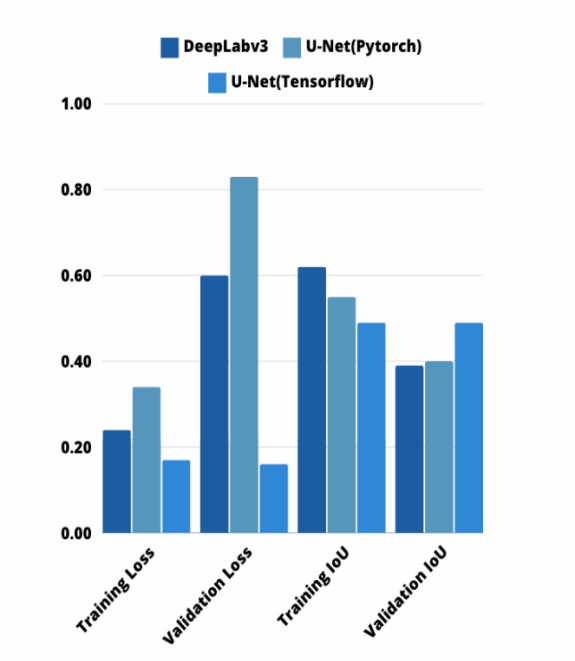

Figure 12. The graph of loss and IoU of DeepLabv3, U-Net with Pytorch and Tensorflow

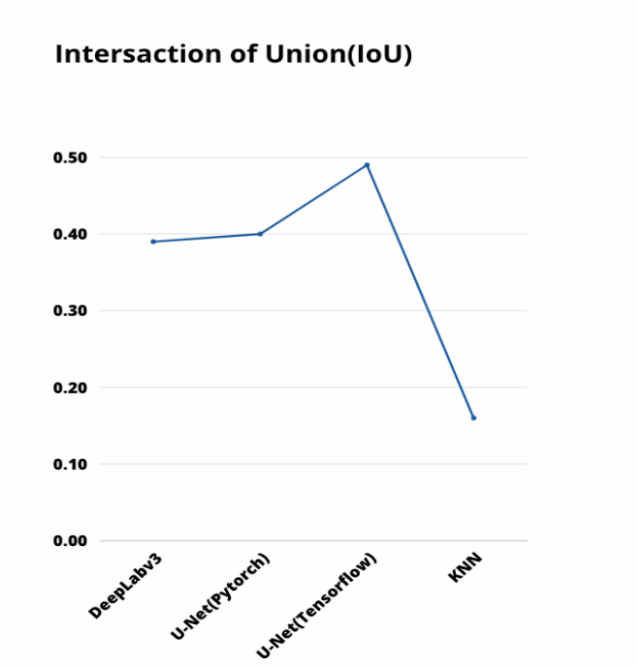

Figure 13. IoU of DeepLabv3, U-Net with Pytorch and Tensorflow, and KNN

As the above two figures show, the performance trends of DeepLabv3 and U-Net with Pytorch are the same. For Loss, the training loss of both these two models is significantly lower than the validation loss, and U-Net with Pytorch shows a larger gap. Regarding IoU, the training set results are higher than the validation one, but the training IoU of DeepLabv3 is slightly higher than the Pytorch one. However, for U-Net with TensorFlow, both the training and validation sets have almost the same loss and IoU, and the loss is lower than the Deeplabv3 and U-Net (Pytorch). The validation IoU is the highest among these three methods. The data used to generate Figures 12 and 13 represent the results after multiple rounds of training. An interesting observation is that the IoU for U-Net with TensorFlow stabilizes after several training iterations, whereas the IoU for DeepLabv3 continues to improve gradually. This shows that the DeepLabv3 model is more suitable for processing large-scale data sets, demonstrating its robustness and ability to continuously learn and improve.

In terms of trend, DeepLabv3 outperforms U-Net in IoU on most datasets, especially when dealing with complex scenes and multi-scale features. The main reason is the design of its ASPP module and dilated convolution. These architectural advantages of Deeplabv3 make it particularly effective at

handling complex segmentation tasks, whereas other models such as U-Net may find it difficult to maintain high performance. ASPP and dilated convolutions work together can help ensure that Deeplabv3 can adapt to diverse and multi-scale natural environment scenes, providing accurate segmentation results in a variety of applications ([7] Y. Wang et al., 2024).

In terms of training time, with the use of CUDA, U-Net's relatively shallow network structure allows for a faster training speed, with each epoch taking roughly 4-5 minutes in Pytorch. On the other hand, DeepLabv3 requires significantly more time due to its complex network architecture and multi-scale processing capabilities. For the dataset of around 1,700 images, each epoch of training DeepLabv3 takes about 30 minutes. U-Net reached convergence in fewer training cycles, while DeepLabv3 took a long time to train, but it showed better accuracy when dealing with large-scale datasets. In terms of inference time, these two models are similarly in time consuming.

For model convergence, in Figure 13, U-Net shows a significant convergence after the first few rounds of training and remains stable afterward, while DeepLabv3 displays a trend of gradual convergence during the training process, which is beneficial for capturing more complex multi-scale information. In contrast, Deeplabv3 shows a more gradual convergence trend throughout the training process.

In addition, a traditional machine learning model K-Nearest Neighbors (KNN) was trained as a comparison group. In this method, due to equipment limitations, this model was trained by the preprocessed dataset which is around 20% of the whole dataset. In this model, the PCA dimensionality reduction technique is utilised, so the training time is extremely short. However, since the principal component was set to 50 during PCA dimensionality reduction, the original image dimension 9144576 needs to be reduced to 50. Although dimensionality reduction filters out noise and redundant information, a lot of detailed information is also lost in the process, resulting in a decrease in classification performance. On the other hand, PCA has high computational complexity and occupies a huge amount of memory which exceeds the device load. Especially when processing an image set of about 1800, the memory occupied is over several hundred GB. Therefore, the IoU of this method is only 0.164 which is relatively lower than those two methods mentioned above displayed in Figure 11. This comparison further highlights the superiority of deep learning models such as U-Net and Deeplabv3, which can better handle the complexity and variability of large-scale image datasets while maintaining high accuracy and performance.

6. Conclusion

In this report, we describe the utilized method in detail for training including U-Net, DeepLabv3 models, and a traditional KNN model, for semantic segmentation in natural environments. Improving the ability of autonomous vehicles to accurately understand and navigate through diverse terrains is our goal. Additionally, this report compares those three models.

The literature review highlighted the strengths of U-Net and DeepLabv3 in semantic segmentation. U-Net's encoder-decoder structure and skip connections effectively capture spatial information, making it suitable for high-resolution image segmentation. DeepLabv3's use of Atrous Spatial Pyramid Pooling (ASPP) and Atrous convolutions captures multi-scale contextual information, essential for complex environments.

Our experiments showed that DeepLabv3 has the best trend of the results, indicating better segmentation performance. However, it required longer training times and more computational resources. On the other hand, U-Net trained faster and converged quickly, which is suitable for limited resources. The traditional machine learning model KNN, which utilized PCA for dimensionality reduction, shows a significantly low IoU.

Additionally, as mentioned above, the main criteria used are the loss and IoU of the training and validation sets. However, there are several limitations to IoU. When the two bounding boxes do not overlap, the IoU value is 0. This means that if IoU is used as a loss function for training, the model cannot learn how to adjust the position of the bounding box through gradient descent ([8] H. Rezatofighi et al., 2019). Consequently, if the predicted bounding box and the real bounding box do not overlap, it may affect the IoU results.

Therefore, in the future, adopting Generalized Intersection over Union (GIoU) as an evaluation criterion may provide a more accurate assessment of the model. By using GIoU, we can overcome some of the inherent limitations of IoU and achieve a higher level of accuracy and effectiveness in our semantic segmentation tasks.

In conclusion, U-Net and DeepLabv3 are both prove to be effective for semantic segmentation with their distinct advantages. U-Net is efficient and quick to train, suitable for limited resources. On the other hand, DeepLabv3 supports complex scenarios with a higher computational cost. Future work should focus on parameter adjustment to enhance performance and generalization and ultimately further enhance model segmentation accuracy in diverse real-world scenarios.

Reference

[1] Hao S, Zhou Y, Guo Y. A brief survey on semantic segmentation with deep learning[J]. Neurocomputing, 2020, 406: 302-321.

[2] Chen L C, Papandreou G, Schroff F, et al. Rethinking atrous convolution for semantic image segmentation[J]. arXiv preprint arXiv:1706.05587, 2017.

[3] Chen L C, Papandreou G, Kokkinos I, et al. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs[J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 40(4): 834-848.

[4] Ronneberger O, Fischer P, Brox T (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". arXiv:1505.04597 [cs.CV].

[5] Python Software Foundation. "Tkinter: Python Interface to Tcl/Tk." Python Documentation. Accessed July 28, 2024. https://docs.python.org/3/library/tkinter.html.

[6] Sulistiyo M D, Kawanishi Y, Deguchi D, et al. Attribute-aware semantic segmentation of road scenes for understanding pedestrian orientations[C]//2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2018: 2698-2703.

[7] Wang, Y., Yang, L., Liu, X. et al. An improved semantic segmentation algorithm for high-resolution remote sensing images based on DeepLabv3+. Sci Rep 14, 9716 (2024). https://doi.org/10.1038/s41598-024-60375-1

[8] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., & Savarese, S. (2019). Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, 658-666. https://doi.org/10.1109/CVPR.2019.00075