COMP9444 Project Summary Report - DeepMinders

Semantic Segmentation in Unstructured Environments

z5444107	Aryan Wadhawan
z5208799	Jonas Macken
z5361805	Man Ching Melvin Chan
z5520430	Dimas Putra Anugerah
z5464720	Zirui Wang

1. Introduction

The aim of this study is to repurpose existing encoder-decoder Convolutional Neural Networks used for image segmentation tasks for the RUGD dataset.

Our study could improve image segmentation to help drones identify objects in complex environments, crucial for tasks like disaster relief and agriculture. Training on the RUGD dataset enables drones to better navigate tough areas and reach people faster. Data augmentation techniques such a cropping and resizing, CutMix, and a novel approach called "StickerMix" were used to balance the dataset's distribution. Additionally, we incorporate advanced neural network modules to investigate the combined impact of these techniques in addressing the unique challenges of the RUGD dataset.

2. Literature Review

This section will highlight the key findings from two literature reviews that were used to establish the baseline models employed in this study, as well key challenges that must be addressed.

2.1. A RUGD Dataset for Autonomous Navigation and Visual Perception in Unstructured Outdoor Environments: A review

This paper [1] reviews the Robot Unstructured Ground Driving (RUGD) dataset and identifies the key challenges that deep learning modules must address for this image segmentation class. Specifically, the RUGD dataset differs from traditional image segmentation tasks in structured urban environments. This is as unstructured, unmarked terrains presents challenges that the conventional models cannot solve, attributed to three key features; **Blurred frames:** Most scenes contain no discernible geometric edges or vanishing points, **Irregular boundaries:** 8 distinct terrains are traversed *(dirt, grass, sand, water, rock bed, gravel, mulch, and asphalt)* which has no regular separation points; **Illumination changes:** From the harsh shadows of outdoor environment

This paper subsequently introduces the traditional encoder-decoder networks used for urban environments in the context of the Robot Unstructured Ground Driving (RUGD) dataset. The models investigated for this report are,

a) ResNet50 encoder with deep dilation + PSPNet
b) ResNet50 encoder with deep dilation + PSPNet with deep supervision

These convolutional neural networks are used for traditional structured environment datasets, and their performance on the RUGD data were assessed in this paper. Although this provides a reasonable starting point for image segmentation tasks, it does not highlight the optimisation of these models through techniques such as hyper parameter tuning or data augmentation. Therefore, these limitations will be addressed in this study, compared with three key metrics introduced by the paper, which are discussed below.

Mean IOU Accuracy: This represents the Mean Intersection-over-Union (Mean IoU) accuracy. Mean IOU is average IOU across all 25 segmentation classes, with the IOU for each class computed as

$$\frac{True\ Positive}{True\ Positive + False\ Positives + True\ Negatives}.$$

Pixel Accuracy: Percentage of correctly classified pixels across all classes

Mean Pixel Accuracy: Percentage of correctly classified classes, averaged across the 25 semantic categories

2.2. Challenges and Solutions for Autonomous Ground Robot Scene Understanding and Navigation in Unstructured Outdoor Environments: A Review

This paper [2] highlights the limitations of the RUGD dataset and inspires novel approaches to address them. It also outlines the importance of sensor fusion with multi-modal data for achieving high segmentation performance in unstructured environments, a challenge for our dataset, which consists solely of RGB data. The paper emphasises leveraging data augmentation to simulate different data types and applying innovative architectural tweaks to help models better perceive and understand complex scenes with limited data. These insights significantly influenced the methods we adopted in our project.

3. Methods and Models

First, three baseline models were developed and evaluated on the dataset. Next, the best-performing of these baselines was selected, and various optimization methods, such as hyperparameter tuning, architecture enhancements (ASPP modules), class-aware loss weighting, and multiple data augmentation techniques, were trialed. Finally, the optimization methods that gave performance improvements were combined into a final model. This section will explore the details of each of these steps.

3.1. Baseline Models

3.1.1. ResNet50 + UNet

U-Net is a well-established architecture for many semantic segmentation tasks, including those in complex outdoor scenes. It is a fully convolutional network with encoder and decoder blocks, compressing images down to lower dimensional representations before reconstructing them to full size for pixel-wise predictions. It also uses skip connections, where images in the encoder block are copied and fed into the corresponding step in the decoder block. This is done to help the model retain spatial information, meaning crucial details aren't lost during the upsampling process, which is particularly important in complex unstructured scenes like those in our dataset.

3.1.2. ResNet50 with Dilated Convolutions + PSPNet

Pyramid Scene Parsing Network is a semantic segmentation model that uses a novel pyramidal pooling module after the encoder. This module consists of several convolutional layers with varied sizes that pools the feature maps, creating new

feature maps that adds global and local context. The new feature maps are interpolated to match the output size and concatenated with the original feature maps, increasing global context information and hence pixel-level accuracy. A dilated convolution value of 8 is used, which is the gaps between each stride in the convolution stage. This allows the model to get a wider context of what is happening in the image.

3.1.3. ResNet50 with Dilated Convolutions + PSPNet with Deep Supervision

This baseline model incorporates an additional deep supervision module into the convolutional network introduced in section 3.1.2. Deep supervision increases information flow from the encoder to the decoder by adding additional skip connections. This technique also attaches an additional loss function to the outputs of the hidden layer, enabling the gradients to be directly back-propagated to the earlier layers of the network. Two skip connections were added to the second and third layer of the PSPNet with 2D convolutional layer. The definition of the auxiliary outputs can be seen in the attached code notebook.

3.2. Hyperparameter Tuning

Hyper parameter tuning was done for the key parameters of learning rate, batch number, momentum, and weight decay. Original parameters were taken from the RUGD paper and are as follows: learning rate = 0.01, batch number = 8, momentum = 0.9, weight decay = 0.001. Random search was used to establish baseline trends to see how the hyper parameters can be adjusted for optimisation.

3.3. ASPP Module

Atrous Spacing Pyramidal Pooling (ASPP) modules is used to pool multiple convolution layers with dilated strides into a single input for the next layer. This allows the convolutional neural network to identify key features in the image at a larger context. This is visualized in Figure 1 below.

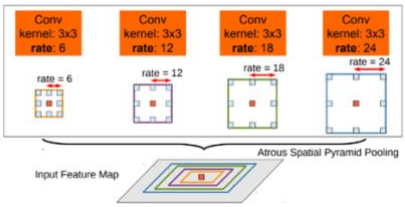

Figure 1. Diagram of a ASPP module [2]

ASPP modules of 3, 4, and 5 convolutional layers was tested with a dilation rates set of $[6, 12, 18, 24]$. This module is added to the input of the encoder.

3.4. Data Augmentation

3.4.1. Cropping and Resizing

This data rebalancing method involves locating connected regions of rare classes, generating minimum bounding boxes around them, upsizing them to full-sized images, and adding them into the training set.

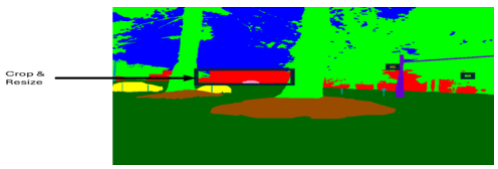

Figure 2. Data augmentation through cropping and resizing

3.4.2. CutMix

The original definition of CutMix involves pasting a part of another image at the same location to improve the model's generalization and balance class distribution. However, due to the scarcity of certain classes, we decided to try dividing the image into smaller patches and randomly pasting extracted rare classes within these patches to better balance the distribution. Initially, we only extracted rectangular segments containing rare classes, which didn't yield a significant effect. After removing the extra portions, we achieved a more balanced distribution, as shown in the image below.

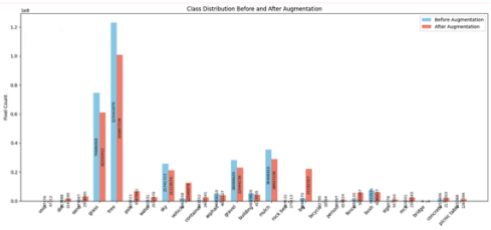

Figure 3. Class distribution before and after cutmix data augmentation

3.4.3. StickerMix

Extending upon CutMix, we introduced a novel method, StickerMix, to more effectively balance class distributions. StickerMix involves first locating connected regions of rare classes, as discussed in Section 3.4.3, saving them in a 'sticker book'. Then, we paste some number of stickers of desired classes in random locations on copies of existing images in the training set, and add these new sticker-augmented copies to the training set. By selecting the appropriate classes and the right number of stickers pasted per image, class distributions should balance and overall model performance should improve.

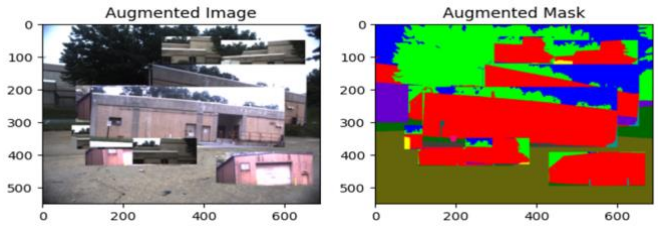

Figure 4. Novel stickmix approach to rebalance class distributions

3.5. Class-Aware Loss Weighting

To address the big class imbalance, weights can be assigned to each class when the loss is calculated after each batch. Increasing the weights of low frequency classes would increase their importance. We evaluated several formulas to determine class weights.

4. Experimental Setup

The RUGD dataset is split into training, validation, and testing sets according to the predefined folder splits outlined in the dataset paper. All training is performed using the ADAM optimizer and the cross-entropy loss function. During training and validation, we monitor the loss and accuracy for each epoch and plot their progression to analyze performance. Three metrics are used to evaluate a method: pixel accuracy, mean pixel accuracy, and mean IoU across all classes on the

test dataset. The method with the highest overall score across all metrics is selected.

First, three baseline models are trained, and the best-performing model on the test is selected as the foundation for the final model. For all experiments, the learning rate is set to 0.001, the maximum number of epochs is 50, the batch size is 8, and early stopping is employed. Afterward, we experimented with various optimizations, including hyperparameter tuning (learning rate, momentum, batch size, and weight decay), integrating the ASPP module, applying data augmentations, applying CutMix, StickerMix, and class-aware loss weighting. Each optimization is evaluated on 256 x 256 images and 4 epochs to save time. Finally, all effective optimizations are combined into the final model configuration, which is trained for up to 50 epochs with early stopping.

5. Results

This section will display the results of the investigations done in this study. Effective optimization strategies were implemented into the final model to determine the significance of these improvements for the unstructured image segmentation class. The impacts of each module will also be discussed.

5.1. Baseline results

5.1.1. ResNet50 + UNet

	Mean IOU (%)		Pixel Accuracy (%)		Mean Pixel Accuracy (%)	
	Validation	Test	Validation	Test	Validation	Test
ResNet50 + UNet	17.23	13.93	90.09%	73.70	20.20%	17.26

5.1.2. ResNet50 with deep dilation + PSPNet

	Mean IOU (%)		Pixel Accuracy (%)		Mean Pixel Accuracy (%)	
	Validation	Test	Validation	Test	Validation	Test
ResNet50 + PSPNet	15.43	13.62	88.95	72.28	17.99	16.58

5.1.3. ResNet50 with deep dilation + PSPNet with deep supervision

	Mean IOU (%)		Pixel Accuracy (%)		Mean Pixel Accuracy (%)	
	Validation	Test	Validation	Test	Validation	Test
ResNet50 + UNet	33.2	10.39	85.7	67.5	15.22	13.33

5.2. Hyper-Parameter Tuning

Using random search the first 10 iterations are highlighted below, and the outcome showed that iterations were more optimised with higher weight decay, lower learning rate, and lower momentum. These results are shown below in Figure 5.

Iterations	1	2	3	4	5	6	7	8	9	10
Learning Rate	0.01	0.01225	0.0763	0.0675	0.0757	0.0348	0.5539	0.008478	0.19219	0.00995
Batch_size	8	2	3	8	7	13	16	13	5	5
Momentum	0.9	0.275	0.6767	0.025	0.025744	0.00275	0.8363	0.61	0.51	0.51
Decay	0.001	0.003	0.009	0.00348	0.002176	0.003	0.006883	0.003	0.004575	0.003475
Results										
Pixel Accuracy	60.57%	53.73%	38.89%	47.74%	51.45%	49.38%	35.63%	56.96%	35.63%	51.36%
Mean Pixel Accuracy	21.46%	21.85%	13.54%	18.76%	22.24%	17.41%	11.24%	23.07%	11.34%	19.84%
Mean IOU	7.83%	6.43%	2.31%	4.23%	5.14%	4.35%	1.43%	7.05%	1.43%	5.23%

Figure 5. Hyper parameter optimisation and its impacts on the key metrics

Therefore, the hyper parameters were tuned to a learning rate = 0.002, batch number = 10, momentum = 0.7, and Weight decay = 0.003. These produced significant improvements to the baseline model which is shown

	Pixel Accuracy	Mean Pixel Accuracy	Mean IOU
Baseline	60.57 %	21.46%	7.83%
Optimised hyper parameters	66.62%	25.01%	8.85%

Figure 6. Key metrics measured for fine-tuned hyper parameter

It is hypothesised that the decrease in learning rate improves the testing results of the image segmentation task as it slows down and prevents the overfitting of the model to the training data. As a 6% increase in pixel accuracy represents significant improvements in performance, the tuned hyper parameters will be included in the final model.

5.3. ASPP Modules

The results for the application of ASPP to the ResNet50+UNet model are presented in Table 1 below.

Table 1. Results for ASPP modules used in ResNet50 + UNet model

Number of convolutional components	Mean IOU	Pixel Accuracy	Mean Pixel Accuracy
Baseline (0)	13.82%	73.69%	23.88%
3	12.17%	71.45%	27.07%
4	13.14%	74.55%	28.21%
5	12.28%	74.11%	27.10%

Therefore, this study shows that 4 convolutional layers produced the highest performing model. This is as using 3 dilated convolutional layer underfits the data, while 5 overfits the data. The hyper parameters of dilation rates and number of convolutional models is a topic of discussion that can be expanded from this report. Due to significant increase in mean pixel accuracy, ASPP modules with 4 convolutional layers was used in the final model.

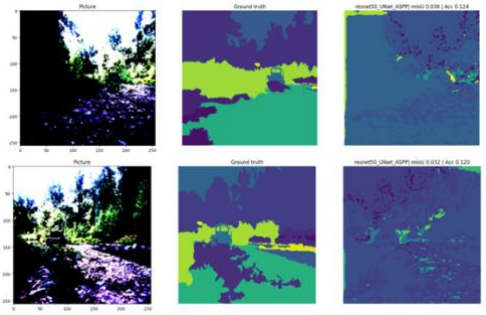

Figure 7. Examples of image segmentation by the ResNet50 + ASPP and UNet model

Figure 7 above shows that the model can clearly identify the borders of the background and foreground in the image but lacks the ability to identify the internal classes correctly. This may be due to the under-representation of other classes in the training data that can be addressed with data augmentation methods.

5.4. Data Augmentation

5.4.1. Cropping and Resizing

While successfully rebalancing class distributions, this approach involved adding distorted images to the training set, forcing the model to learn unrealistic features, and ultimately lead to a decrease in model performance. So, this method wasn't adopted in our final model.

5.4.2. CutMix

The Cutmix we used required pasting rare classes 1,000 times per image, making training and data processing very slow.

Considering that the online training platform we used did not provide sufficient training time to support the implementation of CutMix, we ultimately decided not to apply it in the final model.

5.4.3. StickerMix

Applying StickerMix to a single class, Buildings (Class 12), increased its distribution in the dataset and resulted in significant improvements in class-specific mIoU and accuracy in the test set. Specifically, using StickerMix to increase Building representation from 5% to 20% increased its IoU from 10.83% to 21.60%, and increased its accuracy from 18.65% to 26.65%.

Class 12 (Building)		
% of Total Pixels	IoU (%)	Accuracy (%)
~5	10.83	18.65
~20	21.60	26.56

Figure 8. Results of stickmix class rebalancing

5.5. Class-Aware Loss Weighting

	Pixel Accuracy	Mean Pixel Accuracy	Mean IoU	Delta
Baseline	61.13%	24.37%	9.26%	
Inverse Weighting	64.26%	20.93%	10%	+0.43%
Recall Loss	63.69%	22.38%	9.57%	+0.88%
Logit Inverse Weighting	63.48%	23.79%	10.02%	+2.53%

$$W_i = \log\left(1 + \frac{total\ pixel\ counts}{class_i\ pixel\ counts}\right)$$

Overall, the best performance improvement obtained was by using logit inverse weighting formula. Pixel accuracy and mean IOU by 3.11% but decreased mean pixel accuracy by 0.58%.

5.6. Final Model

Models	Mean IOU (%)	Pixel Accuracy (%)	Mean Pixel Accuracy (%)
Baseline	14.65	73.81	17.87
Final model	10.47	66.5%	23.05

Figure 9. Testing performance of final model

It is shown that the overall testing performance of the final model decreases. This is hypothesized to be attributed to the fact that hyper parameter tuning is specific to each model. Therefore, hyper parameters need to be tuned with the same method as this paper in future works. Furthermore, the interaction of each individual module is not investigated fully.

However, it was found that the training and validation losses are closer together, which indicates a better fitting of the data with the final model.

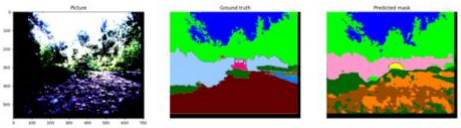

Figure 10. Image segmentation performed with the final model

From the image segmentation results, it can be seen that the final model is able to identify the complex border geometries such as in ASPP. However, the internal class classifications are poor. This can be fixed in future works by combining this final model with sticker mix to improve the semantic segmentation performance of the model,

6. Conclusions

6.1. Novelty and Contributions

Our contributions focused on enhancing baseline performance through hyperparameter tuning, architectural improvements such as integrating ASPP modules, and implementing class-aware loss weighting. While the combination of these methods led to a slight reduction in overall accuracy, each approach individually demonstrated significant improvements to the baseline performance. Additionally, we conducted an in-depth exploration of various data augmentation techniques, emphasizing their limitations when applied to tasks in unstructured environments.

As part of our efforts, we introduced *StickerMix*, a novel method designed to improve class-specific performance in unstructured environments. This approach is both effective and easily adaptable, enabling models to be fine-tuned for environments with a high prevalence of certain classes. Future potential for *StickerMix* includes using it for automated class balancing and thus dynamically addressing class imbalances.

6.2. Limitations and Further Works

A big limitation of our final model is that it can still not accurately assign labels even though it can identify segment boundaries quite well. This is especially the case for less frequent classes. In the case of the ground scenes, the model often mistakes the ground materials, such as logs for mulch, and rock for bush. Another limitation is the limited GPU access, which led to downscaling data and using fewer epochs when evaluating methods.

For future works, we would like to explore using automated pixel count statistics to determine how many stickers each class to paste in the StickerMix method, instead of randomly assigning stickers. This aims to create a flatter class distribution and address imbalance dynamically. Another future area to explore is using a learnable GAN that produces accurate class images. Finally, a poly learning rate could be tested during training to reduce overfitting.

7. References

[1] Wigness. M, Eum. S, Rogers III. G. J, Han. D, Kwon. H. 2020. A RUGD Dataset for Autonomous Navigation and Visual Perception in Unstructured Outdoor Environments. http://rugd.vision/pdfs/RUGD_IROS2019.pdf

[2] Wijayathunga, L.; Rassau, A.; Chai, D. Challenges and Solutions for Autonomous Ground Robot Scene Understanding and Navigation in Unstructured Outdoor Environments: A Review. Appl. Sci. 2023, 13, 9877. https://doi.org/10.3390/app13179877

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, Alan L. Yuille. 2016. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. https://paperswithcode.com/paper/deeplab-semantic-image-segmentation-with-deep