Semantic Segmentation for Wildscences dataset

Gordon Lam
School of Computer Science & Engineering
Faculty of Engineering,
University of New South Wales
Sydney, Australia
z5060305@ad.unsw.edu.au

Alex Zhao
School of Computer Science & Engineering
Faculty of Engineering,
University of New South Wales
Sydney, Australia
z5237250@ad.unsw.edu.au

Junkai Wu
School of Computer Science & Engineering
Faculty of Engineering,
University of New South Wales
Sydney, Australia
z5237534@ad.unsw.edu.au

Pranav Bhatnagar
School of Computer Science & Engineering
Faculty of Engineering,
University of New South Wales
Sydney, Australia
z5340772@ad.unsw.edu.au

***Abstract*— In order for autonomous vehicles to navigate safely and accurately in natural environments, it is important that they can recognize the different types of scenarios and objects they may encounter along the way. However, unlike semantic segmentation in urban settings, natural environment pose a unique challenge due to the vast variability in objects detected. Within the original Wild Scene research study, several semantic segmentation techniques were explored including the use of Neural networks and Transformer based models. In our report, we endeavor to explore two additional methodologies – UNet CNNs and a more traditional machine learning method – Random Forests to make an evaluation along against benchmark models seen in the original Wildscene papers. We propose usage of smaller samples for the benchmarking that aims to preserve the original distribution of the Wildscene data and an approach to conduct a fair comparison of approach.**

Keywords—Semantic Segmentation, Performance Evaluation, Benchmarking, Datasets for Robotic Vision, Convolutional Neural Networks, Image Segmentation, Random Forests.

INTRODUCTION

As autonomous vehicles become increasingly ubiquitous, the need to provide safety and efficiency through self-navigation in different scenarios is gradually increasing. One of the challenges that such vehicles face is the recognition and accurate perception of several situations and objects around the path. For instance, a vehicle needs to be more cautious while moving on sand or mud, or around large groups of trees, than when moving on smooth gravel or asphalt in open areas. Moreover, and more importantly, water is a major obstacle that should be avoided as much as possible (Grigorescu, S., Trasnea, B., Cocias, T., & Macesanu, G. (2020)).

Navigation in natural environments is quite different from navigation within urban settings. The urban landscape is well-defined, clearly identifiable with roads and buildings. Most features in a natural terrain are, however, irregular and unstructured, thereby contributing to higher complexity in perception and decision-making. Therefore, advanced perception systems designed for autonomous vehicles should be able to interpret the fine details of the world for successful self-navigation in such kinds of environments.

First, fine-grained semantic segmentation is achieved towards the complete understanding of scenes. This basically entails assigning a specific class label to each pixel of the images taken by cameras mounted on the vehicle in order to come up with a detailed map of the environment, as indicated by Long, J., Shelhamer, E., & Darrell, T. (2015). By comprehending what every pixel represents, be it sand, mud, trees, or even roads, the vehicle makes informed decisions on how to proceed.

Influenced by the very latest developments in deep learning, semantic segmentation has of late returned much more accurate results using convolutional neural networks, as indicated by Chen et al (2018). However, implementation of these techniques in the complexities of natural, unstructured environments remains a great challenge to the field of computer vision and autonomous navigation.

The paper below, therefore, permits an in-depth comparison of semantic segmentation methods for autonomous navigation in natural scenes based on the very latest source of 'WildScenes: A Benchmark for 2D and 3D Semantic Segmentation in Large-scale Natural Environments'. In this paper, we present a comparison of deep-learning architectures, such as UNet and traditional machine learning methods like random forests, with respect to the baseline models and methods laid down in the WildScenes benchmark. We evaluate different techniques that deal with unstructured terrain challenges. Ultimately, our aim is to further this effort in progress toward enabling safe and efficient autonomous navigation along the broad spectrum of challenging landscapes.

A. DataSet

We make use of the WildScenes dataset, a holistic bimodal dataset designed to perform semantic segmentation tasks in natural scenes, as introduced in the WildScenes paper. This dataset contains 9,306 high-resolution images captured by repeated traversals across two scenes at several different temporal scales, therefore covering most of the environmental and seasonal variability.

It is further augmented with exact 2D semantic labels derived from human annotation. One key property of the dataset is its support for semantic domain adaptation research; this has been guaranteed by including environments that are geographically very apart and their repeated traversal with a six-monthly time gap. This dataset can become a very good benchmark for semantic scene understanding in natural environments due to the availability of carefully curated train-validation-test sets, optimized to ensure there is a balanced distribution of class labels, including dedicated splits for domain adaptation experiments.

In our adaptation of the dataset, we aimed to preserve the original distribution by design through uniform sampling over the symlinked splits defined by the WildScenes creators. In our approach, the overall training and testing sample space was reduced to 30% of the data size using a random shuffle for reasons of reducing space complexity and time to train the models.

Due to several technical constraints on data acquisition and transferral (caused by the large dataset size of Wildscenes2d), the preservation of class uniformity could not be perfectly realised in practice. Even after increasing sampling size after and performing a secondary shuffle, the models did not further improve it is predictive power. We discuss this in more detail in the experimental results. This would, in any case, not affect our primary research objective of comparative analysis for the different methodologies using the same consistent dataset with a small bias. This will help in keeping the integrity of our comparative study since all methods will be run on the same dataset, and the validity of the inter-method performance assessments will be preserved despite class-specific biases that might exist.

LITERATURE REVIEW

B. Semantic Segmentation

One of the most important tasks in computer vision is semantic segmentation, which involves assigning a class label to every pixel in an image. According to Garcia-Garcia et al., it refers to "the process of partitioning an image into semantically meaningful parts, and classifying each part into one of the pre-determined classes". Unlike image classification or object detection, semantic segmentation provides a detailed understanding of the scene structure by generating dense pixel-wise classifications.

The importance of semantic segmentation lies in the capability of the former to provide detailed scene understanding, which is crucial in applications like autonomous driving, medical image analysis, and remote sensing. As noted by Minaee et al., "Semantic segmentation has a wide range of applications, from scene understanding to medical image analysis to computational photography".

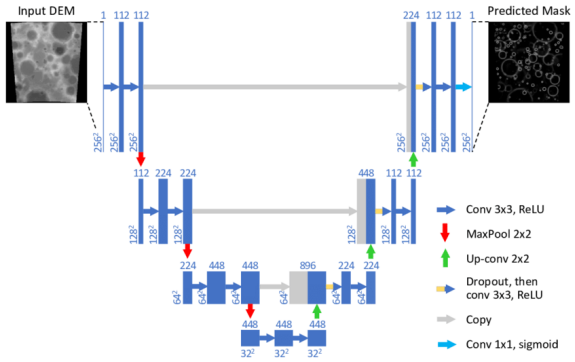
Semantic segmentation poses a challenge because it requires the incorporation of local and global context while maintaining spatial precision. In fact, algorithms need to be developed that can process multiscale information for the accurate segmentation of objects of all sizes. Furthermore, these algorithms are meant to gain invariance regarding light, pose, and occlusion. Long et al. contributed to this challenge: "Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation".

Identify applicable funding agency here. If none, delete this text box.

C. *Convolutional Neural Networks*

Recently, state-of-the-art approaches to semantic segmentation have been dominated by CNNs, particularly those based on the U-Net architecture. U-Net, as proposed by Ronneberger et al. (2015), is symmetric in structure, having an encoder-decoder path. On one branch, the encoder captures increasingly abstract features, and on the other, the decoder recovers spatial information. One of the important innovations brought about by the U-Net is the **skip connections**, which concatenate feature maps from the encoder to corresponding layers in the decoder to recover information that may be lost during the up-sampling process. Figure 2.1 depicts this architecture where-in

FIGURE 2.1 EXAMPLE UNET ARCHITECTURE USED IN "LUNAR CRATER IDENTIFICATION VIA DEEP LEARNING" SHOWING SYMMTRIC NETWORK SHAPE



In this respect, the good performance of U-Net in semantic segmentation may be related to several factors. According to Çiçek et al., "The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization". This way, it is clear how U-Net captures low-level information about details and high-level semantic information, which turns out to be useful to obtain an accurate segmentation map.

The effectiveness of skip connections in U-Net has also been validated by numerous studies, showing significant improvements in segmentation accuracy by preserving spatial details and improving training dynamics. For instance, Huang et al. (2017) use a variation of skip connections by connecting each layer to every other layer in a feed-forward fashion in the model now known as Dense Convolutional Networks (DenseNets). This architecture demonstrates that skip connections can improve feature reuse and gradient flow, leading to enhanced learning efficiency and accuracy.

Later research has based its foundation on the architecture of U-Net. For instance, Zhou et al. proposed UNet++, which introduced nested and dense skip connections that bridge the semantic gap between encoder and decoder features. That is what allows more flexible feature fusion and shows improved performance on medical image segmentation tasks.

D. *Random Forest Segmentation*

Although CNNs now form the default approach to many semantic segmentation tasks, Random Forests offer a rival methodology with their own strengths. Random Forests are an ensemble learning method which builds many decision trees and then outputs the mode of the classes for classification tasks. Shotton et al. illustrated the use of Random Forests in semantic segmentation in their paper on Semantic Texton Forests.

Random Forests have several possible advantages for semantic segmentation. According to Schroff et al., "Random forests have been shown to be efficient classifiers for multi-class problems". They are intrinsically suitable for multi-class issues, and their applicability as such to segmentation tasks for multiple object categories follows immediately. Their ensemble nature lends robustness to Random Forests

against overfitting, which may turn out especially useful when working with limited training data.

One of the intrinsic advantages of random forests is computational efficiency. Both the training and inference are relatively easy to parallelize and could result in faster processing times than deep neural networks, especially over smaller datasets or weak computing devices. Another major advantage of Random Forests is that they offer some interpretability, an area where deep learning approaches are usually lacking.

Both CNN-based methods, such as U-Net, and Random Forests have some advantages concerning semantic segmentation. Thus, choosing one or the other, or a hybrid model bridging between them, will be based on the dataset size, computational resources, the need for interpretability, and the specifics of the segmentation task.

METHOD BASELINE

As the aim of this research will be a comparative analysis of different methods, a baseline measure was introduced to guide the model evaluation plan. In this research we utilise the Wildscene codebase to provide a boiler plate to build on and test other CNN models such as UNet. Within the Wildscene repository, pre-existing model configurations using modified backbones from the mmsegmentation framework was used to configure the deeplabv3 baseline for benchmarking. As mentioned in the original Wildcenes paper, DeepLabv3 was also chosen as the traditional CNN architecture for benchmarking (Strudel et al. (2021); Li et al. (2022); Cheng et al. (2021)).

METHOD – UNET CNN

Implementation of the UNet CNN framework utilises the MMsegmentation framework for training and evaluation. By design, UNet requires configuration over the encoding path (contracting) and decoder path (expansive) to capture information for classes at all levels. Because of this, the UNet Neural network is able to understand context of objects and perform precise localization to determine the boundary points between them at pixel level.

A. Data preprocessing configuration

Due to the variance in colour intensity, the image RGBs have been minimally processed as consistent with the Wildscenes creator using image normalisation and conversion to RGB format to standardize the colour distribution across all images which is crucial for helping neural networks learn more effectively. In 2015, Ioffe and Szegedy noted that normalization of each input layer significantly improved the training of deep neural network by reducing internal covariate shift. Our configurations used the same normalisation values as per Wildscene models to better represent the actual population statistics (assuming we had used all Wildscene samples).

Mean = [123.675, 116.28, 103.53],
Std = [58.395, 57.12, 57.375]

FIGURE 4.1 NORMALISATION DOES NOT CHANGE THE IMAGE ITSELF

Apart from this, the training process will also incorporate elements of basic image processing such as random cropping and random flipping to ensure the model performance can robustly capture minute details in the image.

B. Model Configuration

The UNet configuration utilises the mmsegmention backbone for UNet with customised configurations suited for our needs. In terms of feature extraction, key configuration settings include:

1) Set num_stages = 5

This parameter defined the number of resolution levels in the network to help capture features at different scales such as objects of different sizes and complexities(e.g. trees vs cluster of leaves) as supported by Fu, Hengrong, et all (2021). At each stage, we begin with 64 base channels (or filters) for the image as indicated by O. Ronneberger (2015) to provide a good balance between performance and computational efficiency

2) Convolutional Layers = 2 (across encoder and decoder)

We select 2 convolutional layers on each stage for both the encoder (to learn the complex hierarchical features from input image) and decoder (to reconstruct the segmentation map by combining features from encoder with the skip connection). We can increase this, if the level of information captured is still insufficient as networks with more convolutional layers are shown to improve performance in semantic segmentation.(J.Long, E.Shelhamer and T.Darrell, 2015) but we are keeping it minimal due to limitations in computing resources.

Between each stage of the encoder, a max-pooling layer is enabled to enable down-sampling to occur to reduce the dimensionality of the images and capture higher level features.

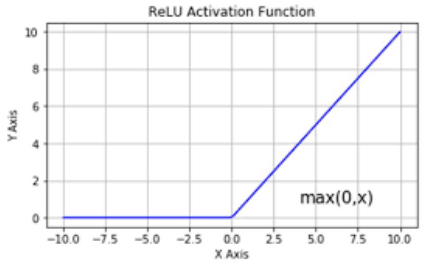
3) Setting upsample_cfg=dict(type='InterpConv')

This setting effectively allows for the network to resize feature during the decoder up sampling stage but preserve the spatial details usually lost to when the image is down sampled (which we will allow for all stages).

C. *Activation function*

A ReLu activation function was selected to introduce non-linearity into the network Normalization but overall is known to perform well in image recognition and segmentation tasks (Krizhevsky et al. 2012)

FIGURE 4.2 RELU ACTIVATION FUNCTION, NEARAL NODE ACTIVATES ON POSITIVE WEIGHTED SUM VALUE OF X



D. *Classes*

It should be noted that the UNet Model is originally trained over the Wildscene allocation with 19 classes. Post our training processes, we have remapped several of the outcome classes down to 15 produce the final evaluation metrics.

E. *Hyperparameter Tuning for Model Training*

For our initial setup we will use a learning rate of 0.01, though we will vary thing amount. The training rate will help us converge faster or slower but setting at a bad value may lead to the model stuck on a local minima. We will need to configure and tune this value

The Iteration rates will also be tuned to configure speed and computational memory. We start with a value of 80,000 but change that as required.

METHOD – RANDOM FOREST

F. *Data Pre-processing*

The WildScenes dataset involves pictures taken in a very bright scenario – causing some objects such as trees to reflect light onto the camera and appear much brighter. This is an issue when trying to segment the image because the difference in color pixels is much lower between different segmentation classes – leading to increases in misclassification.

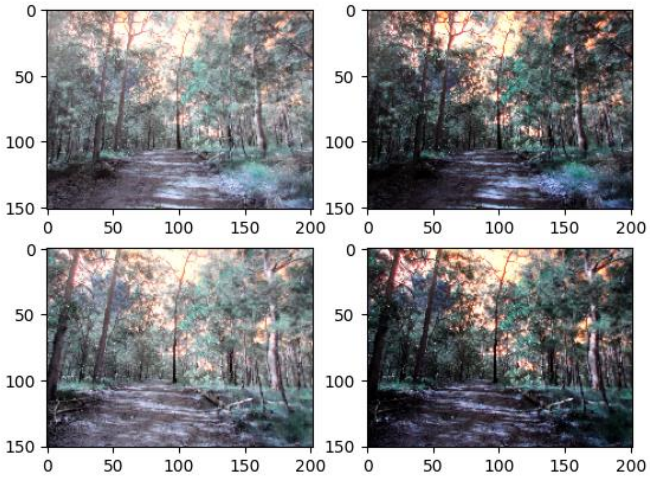
FIGURE 5.1 EXAMPLE OF AN BRIGHT IMAGE

To solve this, we used applied gamma correction on the image.

$$x' = c \cdot x^{\gamma} ,$$

Where x is the value of a pixel, x' is the new pixel value, and c is the normalising factor to ensure x'stays in the range $[0,255]$.Through fine tuning, we chose $\gamma = 1.9$ to widen the gap between pixel color values of pixels that are more white.

FIGURE 5.2 BEFORE AND AFTER GAMMA CORRECTION



One more thing we did was reduce the dimensions of the image – due to time constraints.

G. *Extracting features*

In order to train a random forest we need to first extract features for our model to fit to the labels.

We chose to extract 5 features for each pixel – 3 color channels and its (x, y) position in the image.

Thus given we have n images of size (a, b) we obtain our training vector:

$$(n, a, b, 5) \rightarrow (n \cdot a \cdot b, 5)$$

Similarly, for our labels, we extract 3 features for each pixel – its classification (an integer in range $[1,18]$) and its (x, y) position in the mask. Similarly, we obtain our label vector

$$(n, a, b, 3) \rightarrow (n \cdot a \cdot b, 3)$$

H. Fitting

For our RF model, we tune the following hyperparameters:

n_estimators – 150
max_depth – 13

Other parameters remain as default.

Due to time contraints, we chose n_estimators, the number of trees, only to be 150.
We also limit the max_depth of the tree to reduce overfitting of our model.

Our model takes in the training vector, and attempts to predict the correct label vector.

$$(n \cdot a \cdot b, 5) \rightarrow (n \cdot a \cdot b, 3)$$

Then to retrieve our label mask from the output vector, we reshape the vector back into a list of images:

$$(n \cdot a \cdot b, 3) \rightarrow (n, a, b, 3)$$

I. Smoothing

However, the output of our segmentation usually contains a lot of noise and because we are classifying each pixel one by one, it does not contain a smooth boundary. To solve this problem, we apply smoothing filter to our output from the RF model.

Note that the exact value of each pixel in our mask images are important – changing a pixel value from 1 to 2 will cause a misclassification. Therefore we need a smoothing filter that only changes pixel values to the exact value of a pixel in its vicinity.

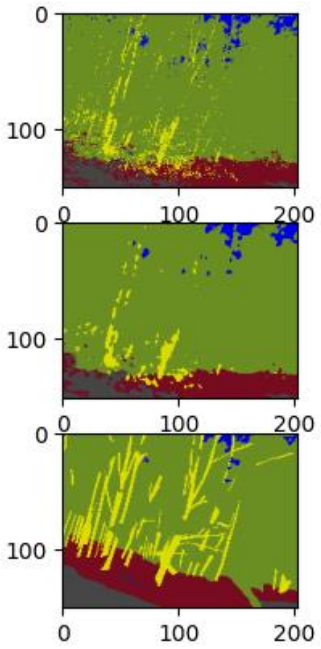
For this we chose the mode filter:

$$x' = \text{Mode}(x, h)$$

Where x' is the new pixel value, x the original value, h the width of our filter, and Mode() the function that returns the mode of values within a h radius around x .

Due to the smaller size of our images, we chose $h = 1$, but as the size of the image increases, so will the radius of our filter.

FIGURE 5.3 TOP IMAGE IS ORIGINAL MASK, MIDDLE IMAGE IS AFTER APPLYING MODE FILTER, BOTTOM IS TARGET LABEL



METHOD – RANDOM FOREST

A. Measurement Goal

As explained by the Wildscene creators, the original wildscenes2d dataset were originally split into train,test and validation groups in a class uniform manner. Despite the MNAR loss in samples which resulted in poor segmentation performance in several classes, the tested methods were able to yield results that were consistent across all 3

methods tested. In consequence we can use two measures to provide a consistent measure of comparison across all 3 groups:

1) *Mean Intersection over Union (mIoU)*

$$\text{Mean IoU} = \frac{1}{C}\sum_{c=1}^{C} \text{IoU}_c$$

$$\text{IoU}_c = \frac{|A_c \cap B_c|}{|A_c \cup B_c|} = \frac{TP_c}{TP_c + FP_c + FN_c}$$

, where
c = all classes.
TP = True Positive
FP = False Positive
FN = False Negative
The mIoU metric is commonly used to evaluate semantic segmentation tasks. The goal of the metric is to measure the overlap between the predicted segmentation against the ground truth segmentation against all each class.

2) *Adjusted Intersection over Union*

Due to the issue with MNAR data loss, we introduce another measure that adjusts the mean intersection over union (mIoU). The adjustment will look to exclude IoU for classes where there is 0 intersection. The purpose of the adjusted mIoU is to provide a layer of evaluation for classes that could be recognised (defined arbitrarily as IoU > 0.1)

3) *Other measures*

Accuracy is the ratio of the number of correctly predicted samples to the total number of samples. Mean Accuracy, often referred to as MAcc, is the average accuracy across all classes. It is especially useful for evaluating models on imbalanced datasets, where some classes may be underrepresented.

TABLE I. INDIVIDUAL CLASS IOU ACROSS MODELS

Class	*Baseline model (DeepLapv3)*	*U-Net CNN*	*Random Forest*
Bush	*0.0*	*0.0*	*0.0*
Dirt	*34.42*	*48.69*	*51.17*
Fence	*Nan*	*Nan*	*0.0*
Grass	*23.95*	*50.04*	*21.49*
Gravel	*0.0*	*0.0*	*0.0*
Log	*0.0*	*0.0*	*0.0*
Mud	*0.0*	*0.0*	*0.0*
Object	*0.0*	*0.0*	*0.0*
Other-terrain	*0.0*	*0.0*	*nan*
Rock	*0.0*	*0.0*	*0.0*
Sky	*0.0*	*60.22*	*59.31*
Structure	*14.76*	*0.0*	*0.0*
Tree-foliage	*74.44*	*78.65*	*61.43*
Tree-trunk	*30.74*	*47.17*	*11.02*
water	*nan*	*Nan*	*0.0*
MIoU	*13.72*	*21.91*	*14.6*
Adjusted mIoU	*35.662*	*56.954*	*40.884*

4) *Results U-net CNN*

With the given base model from mmsegmentation, in this progress, the U-net CNN network would be training and testing with the following parameters: batch size, learning rate, and max iteration. Because of the local device limitation, the training and testing would run on colab. To reduce the running time and prevent the out-of-memory issue of Cuda, the batch size was set to 2. The following table will show the individual class IoU of the iteration that generates the max mIoU or the latest iteration during each training period.

TABLE II. INDIVIDUAL CLASS IOU + AACC FOR UNET TRAINING ACROSS DIFFERENT TRAINING PARAMETERS

	Lr = 0.01, iter=72k, with 5% dataset		*Lr=0.001, iter=40k, with 5% dataset*		*Lr=0.01, iter=20k, with 30% dataset*	
Class	*Iter=48k*	*Iter=72k*	*Iter=28k*	*Iter=40k*	*Iter=14k*	*Iter=20k*
Bush	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*
Dirt	*48.69*	*36.98*	*55.65*	*47.84*	*48.73*	*36.86*
Fence	*Nan*	*Nan*	*Nan*	*Nan*	*Nan*	*Nan*
Grass	*50.04*	*38.05*	*44.98*	*45.35*	*41.13*	*37.79*
Gravel	*0.0*	*1.43*	*8.88*	*0.0*	*0.0*	*0.0*
Log	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.02*
Mud	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*
Object	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*
Other-terrain	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*
Rock	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*
Sky	*60.22*	*51.92*	*43.05*	*39.08*	*64.51*	*57.06*
Structure	*0.0*	*1.0*	*0.0*	*0.0*	*0.0*	*0.0*
Tree-foliage	*78.65*	*75.8*	*77.24*	*77.41*	*75.74*	*75.33*
Tree-trunk	*47.17*	*46.34*	*42.36*	*47.65*	*41.99*	*38.17*

water	*Nan*	*Nan*	*Nan*	*Nan*	*Nan*	*Nan*
MIoU	*21.91*	*19.35*	*20.93*	*19.79*	*20.93*	*18.86*
aAcc	*80.16*	*77.84*	*78.3*	*78.65*	*77.8*	*76.54*

5) *Random Forest*

For our dataset – we have 720 images for training, and 191 images for testing.

We run our testing dataset through our Random Forest model to retrieve our predicted labels. Then we apply our mode filter to test both methods.

The model was tested by applying a mask that sets all pixels except a certain class our predicted labels and target labels. Then for each class, we calculate the mean IoU over all the images. The following Table shows the individual class IoU for our final labels for both methods.

TABLE III. INDIVIDUAL CLASS IOU WITH AND WITHOUT MODE FILTER

Class	*Random Forest (without mode filter)*	*Random Forest (with mode filter)*
Bush	*0.0*	*0.0*
Dirt	*50.22*	*51.17*
Fence	*0.0*	*0.0*
Grass	*21.76*	*21.49*
Gravel	*0.0*	*0.0*
Log	*0.0*	*0.0*
Mud	*0.0*	*0.0*
Object	*0.0*	*0.0*
Other-terrain	*nan*	*nan*
Rock	*0.0*	*0.0*
Sky	*56.45*	*59.31*
Structure	*0.0*	*0.0*
Tree-foliage	*61.56*	*61.43*
Tree-trunk	*15.56*	*11.02*
water	*0.0*	*0.0*
MIoU	*14.66*	*14.6*
Adjusted	*41.07*	*40.884*

DISCUSSION

6) *U-net CNN*

According to Table 2, a learning rate (lr) of 0.01 with 48,000 iterations and 20% of the dataset yields the best overall performance, achieving the highest Intersection over Union (IoU) of 21.91 and accuracy of 80.16%. Conversely, a lr of 0.001 with 28,000 iterations and the same dataset size performs reasonably well, with an IoU of 20.93 and an accuracy of 78.3%, indicating that a lower lr with fewer iterations can still be effective. Increasing the dataset to 30% reaches a relatively good performance with fewer iterations.

Classes such as dirt, grass, sky, tree foliage, and tree trunk demonstrate consistent and higher performance across different configurations, highlighting the model's effective learning capabilities for these categories. In contrast, other classes like bush, gravel, log, mud, object, other-terrain, rock and structure consistently exhibit zero or low IoU, indicating significant challenges in learning and segmentation for these categories. With high iterations or low learning rates, the model seems to start paying attention to the challenging classes while the IoU is still low for these classes.

Several factors may cause the challenging class to have 0 IoU. Firstly, the dataset used for training may have serious class imbalance issues. The dataset may have very few instances of these classes compared to others, which makes it hard for the model to recognize them effectively. Since only a few portions of the dataset are used, there might not be enough examples of these classes in the training set. Besides, these classes might be inherently more difficult to segment due to their appearance, size, or context within the images. Moreover, the model may be too simple to capture the complexities of these classes.

7) *Random Forest*

According to table 3, classes such as dirt, sky, and tree foliage have demonstrate higher accuracy between both methods, and classes such as tree-trunk and grass struggle (with accuracies of 21.49% and 11.02%) and are often misclassified. Furthermore, Classes such as other classes like bush, fence, gravel, log, mud, object, rock, structure and water consistently show an accuracy of 0%, indicating significant challenges to these classes as well.

Similar to the CNN method, class imbalance plays a large factor in the misclassification of the classes with 0 IOU. However another factor is that due to time and device constraints, the images have been downscaled to run faster for the random forest implementation. Thus, we are losing 99% of pixels (downscaling by factor of 10 in both dimensions), and lose a lot of information – which can cause our random forest to interpret the classes with 0 IOU as well as tree trunk and grass as simple noise.

Furthermore, we are only using a small subset of the dataset as well, which leads to a lack of occurrences for the misclassified classes. For the tree-trunk specifically, because we use rgb colours as part of our feature extraction, it is rather hard to classify trees when some trees appear "white" due to the suns reflection and some "black" from their natural colour.

Comparing the two methods – with and without mode, the method without the filter yields a higher adjusted accuracy of 41.07% vs 40.884%. Comparing between individual classes, the largest difference was the tree trunk, dropping the accuracy from 15.56% to 11.02%. Some factors that may have caused this is the small image size – and thus our mode filter (although currently width 1) is still large relative to our image size. However, although not using a mode filter yields a higher accuracy, it may still be preferred to smooth out the boundaries for real life use cases of segmentation.

8) Other Remarks

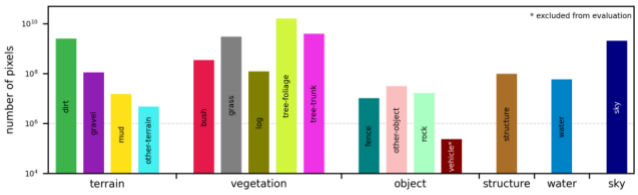
From a performance perspective, the UNet model when operationalised using CUDA for training, runs modestly well. It should be noted that training across all stages typically took several hours and increased linearly while increasing sample size and varying training parameters across different iteration values. The random forest model was initially built to leverage only CPU processors. For modest sample size about 40% of what was used to train UNet and Deeplabv3. During upscaling to leverage the larger dataset (30%), model fitting suffered from poor scalability where it often ran out of memory or suffered from run-time disconnection.

Despite this, the performance of the model has shown modest results and doing almost as well as the UNet model.
In the future we propose several enhancements that would uplift the integrity of this comparative study. Firstly, more time would be required to validate the data distribution, thought the wildscene creators had attempted to flatten the distribution of the classes, the model itself is still inherently biased as per the figure below. Because of this, some considerations may be needed as to how to oversample data where it is underrepresented as this is affecting the ability of both deeplabv3, Unet and even random forest models for different samples of the Data.

Secondly, there is opportunity for further parameter optimisation to improve the model further, only a few of options were explored but one potential is to adjust the filter size to be even smaller to be larger to and add more layers in the UNet network to capture more depth of feature.

For the random forest model there was little opportunity to tune the model as it was built on CPU and training even on small datasets usually took hours. By rebuilding on CPU, the model can be better tuned for an even fairer comparative analysis in this study.

FIGURE 7.1 DISTRIBUTION OF CLASSES FROM WILDSCENE DATA



CONCLUSION

Using the U-net CNN model for semantic segmentation, more than 75% of the pixels can be recognised correctly (for high frequency classes with IoU > 0), while the model cannot effectively recognize the rest of the pixels, which contain the challenging classes. These classes get low Intersection over Union, which may be caused by class imbalance, insufficient training data, segmentation challenges, and model complexity.

The random forest model in comparison was less performant all across the board, however, was able to provide results in the same classes where Unet model was also able to. In future, if the model could be rebuilt to leverage GPU, the model could equate or even exceed the performance of UNet.

To address the challenges in semantic segmentation, maintain a lower learning rate and greater iterations. Increase the dataset size beyond 30% and balance class representation using techniques like oversampling and data augmentation. Implement class-weighted loss and transfer learning to focus on underrepresented classes. Use cross-validation and analyse confusion matrices for better evaluation and model refinement.

REFERENCES

[1] K. Vidanapathirana, J. Knights, S. Hausler, M. Cox, M. Ramezani, J. Jooste, E. Griffiths, S. Mohamed, S. Sridharan, C. Fookes, and P. Moghadam, "WildScenes: A Benchmark for 2D and 3D Semantic Segmentation in Large-scale Natural Environments," *The International Journal of Robotics Research*, vol. XX, no. X, pp. 1-15, 2023, doi: 10.1177

[2] Grigorescu, S., Trasnea, B., Cocias, T., & Macesanu, G. (2020). A survey *of* deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3), 362-386.

[3] Chen, X., Milioto, A., Palazzolo, E., Giguère, P., Behley, J., & Stachniss, C. (2021). SuMa++: Efficient LiDAR-based semantic SLAM. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1-7). IEEE.

[4] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3431-3440).

[5] Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 801-818).

[6] A. Garcia-Garcia et al., "A Review on Deep Learning Techniques Applied to Semantic Segmentation," arXiv:1704.06857, 2017. [Online]. Available: https://arxiv.org/abs/1704.06857

[7] S. Minaee et al., "Image Segmentation Using Deep Learning: A Survey," IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 2, pp. 479-500, 2021.

[8] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in MICCAI, 2015, pp. 234-241.

[9] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261-2269.

[10] J. Shotton, M. Johnson, and R. Cipolla, "Semantic Texton Forests for Image Categorization and Segmentation," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, USA, 2008, pp. 1-8, doi: 10.1109/CVPR.2008.4587506.

[11] A. Silburt, M. Ali-Dib, C. Zhu, A. Jackson, D. Valencia, Y. Kissin, D. Tamayo, and K. Menou, "Lunar Crater Identification via Deep Learning," *Icarus*, vol. 317, 2018, doi: 10.1016/j.icarus.2018.06.022.

[12] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

[13] Strudel R, Garcia R, Laptev I and Schmid C (2021) Segmenter: Transformer for semantic segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 7262–7272.

[14] Fu, Hengrong, et al., "A Survey on Deep Learning for Semantic Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 5, pp. 1604-1622, 2021.

[15] Ö. Çiçek et al., "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation," in MICCAI, 2016, pp. 424-432.

[16] Z. Zhou et al., "UNet++: A Nested U-Net Architecture for Medical Image Segmentation," in DLMIA, 2018, pp. 3-11.

[17] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431-3440.

[18] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2008, pp. 1-8.

[19] F. Schroff, A. Criminisi, and A. Zisserman, "Object Class Segmentation using Random Forests," in Proc. Brit. Mach. Vis. Conf., 2008, pp. 54.1-54.10.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems (NIPS)*, vol. 25, pp. 1097-1105, 2012.