Semantic Segmentation on WildScene: UNet, SegNet, DeepLabV3+, and SVD Tuning

C. Chen, T. Xu, X. Li, Y. Qu, Y. Ma
School of Computer Science and Engineering
University of New South Wales
Sydney, NSW

1. Introduction

In this section, we explore semantic segmentation and the WildScenes 2D dataset, focus on natural environment challenges, and find strategies to solve data imbalance and pre-processing needs.

1.1. Understanding of Semantic Segmentation Task

Semantic segmentation, which involves classifying each pixel in an image into predefined categories, has received increasing attention in computer vision. It becomes one of the most useful intelligent applications , particularly in autonomous vehicles. Autonomous vehicles recognize the surrounding environment, including roads, pedestrians, and vehicles, and make decisions accordingly. Thus, autonomous vehicles can better understand complex driving environments, improving safety and efficiency. Advanced algorithms and models such as UNet, SegNet, and DeepLabV3+ are widely employed in semantic segmentation.

However, these methods are often difficult to process and generalize to different environments in real time. Unstructured environments, such as natural and large-scale open areas, pose even greater challenges. Semantic segmentation in natural environments faces many difficulties, including lighting variations, shadow blurring, and diverse natural textures.

1.2. Understanding of WildScenes Dataset

For our group project, we used the WildScenes 2D dataset, which consists of 9306 high-resolution images (2016 × 1512). Each of the images is annotated with accurate 2D semantic labels, dividing the scene into 18 natural-scene classes. Our dataset provides pixel-level labels, offering a high level of detail compared to those with bounding boxes or annotations for entire objects. After post-processing, any class with fewer than 10^6 pixels was excluded from evaluation. We merged pole into other object, merged asphalt into other-terrain, and excluded vehicle from evaluation-leaving an evaluation set of 15 classes for 2D semantic segmentation (Vidanapathirana et al., 2023). Our data analysis reveals a significant imbalance in the dataset, with some classes dominating while others are much less frequent. For instance, 'tree-foliage' accounts for 56.78% of the dataset, whereas 'pole' makes up only 0.01%.

The intensity of natural lighting conditions makes it difficult to distinguish between similar terrains and objects, such as dirt, mud, and gravel. This dataset contains many instances of glare and shadows, making segmentation challenging and requiring pre-processing. Another major problem is the branches, leaves, and bushes overlap and blur boundaries.

To improve model performance, We use Singular Value Decomposition (SVD) as a fine-tuning and a combination of loss functions.

2. Literature Review

There are many traditional and emerging methods of image segmentation. From the simplest thresholding, and clustering to watershed transformation and neural networks (Basavaprasad & Ravindra, 2014).

Traditionally, there are primary 2 methods. One is Edge-Based segmentation, using 1st order Derivative or 2nd Order Derivative, like the Prewitt operator, Sobel operator and the Laplacian operator. Another is Region-Based segmentation, using Region Growing, Splitting, and Merging or Watershed Transformation (Sujata Saini, 2014). However, most of these techniques are not suitable for noisy environments (Pal & Pal, 1993), which is characteristic of our dataset.

With the increasing need to segment real and complex pictures and the invention of neural network models like convolutional neural networks, it has become the most popular methods to do image segmentation (Ghosh, Das, Das, & Maulik, 2019). In recent years, more and more types of neural network methods have been developed. One of the most important methods is the Encoder-Decoder method in CNNs. This method is represented by UNet. The encoder part of UNet is used to gradually reduce the spatial resolution and extract features, with convolutional and pooling layers. The most famous encoders are VGG and ResNet, using multiple convolutions in sequences or Residual learning frameworks (Simonyan & Zisserman, 2014) (He, Zhang, Ren, & Sun, 2016), for image classification. The decoder part is used to gradually recover the resolution and positional information, merging the low-level and

high-level features in the encoder with upsampling and skip connections (Ronneberger, Fischer, & Brox, 2015). Based on the concept of Encoder-Decoder, some other models have been created with their own features, like SegNet, PSPNet, and DeepLabV3+. In SegNet, it removed the fully connected layers of VGG16 and used Max-Pooling Indices to save space and maintain sufficient accuracy, making it easier to train and having a faster processing speed than other models (Badrinarayanan, Kendall, & Cipolla, 2017a). The main difference between SegNet and UNet is that SegNet does not use Skip Connections, which UNet uses to transfer encoder features directly to the decoder. In PSPNet and DeepLabV3+, they used Pyramid Pooling Module, Dilated Convolution, and Atrous Spatial Pyramid Pooling to detect multiple versions of features at different resolutions, which enable them get obtain information from different scales (Zhao, Shi, Qi, Wang, & Jia, 2017) (Chen, Zhu, Papandreou, Schroff, & Adam, 2018).

Besides the Encoder-Decoder based methods in CNNs, Regional Proposal methods, Recurrent Neural Networks, and Semi and Weakly Supervised Methods can also be used in image segmentation. R-CNN, a method using the Regional Proposal method, combined Region Proposals with CNN, which significantly improved target detection accuracy but can occupy enormous resources during training and testing because the feature extraction, classification, and regression are all separated from each other (Girshick, Donahue, Darrell, & Malik, 2014). ConvLSTM, a recurrent neural network that can be used for Spatio-Temporal prediction. However, we need the input and the prediction target to be Spatio-Temporal sequences, which are unavailable in our dataset (Shi et al., 2015). For semi and weakly-supervised methods, as we have the segmentation label data and semi and weakly-supervised methods give significantly worse results than fully-supervised methods (Papandreou, Chen, Murphy, & Yuille, 2015), we decided not to use them.

At the same time, the performance of the model is also very important. One method to speed up the training process and increase the accuracy is combining different loss functions (Dickson, Bosman, & Malan, 2022) (Li, He, Li, & Shen, 2021). For semantic segmentation, we typically use Categorical Cross-entropy Loss (Cox, 1958), Intersection over Union Loss , Dice Loss (Sørensen, 1948), Focal Loss (Lin, Goyal, Girshick, He, & Dollár, 2017), and among others. Another method is to use Singular Value Decomposition (SVD) to do the parameter-efficient fine-tuning (PEFT). With SVD, we can introduce a novel framework termed subspace tuning, which can help us to encapsulate all known PEFT methods under a unified theory, by doing subspace tuning, subspace reconstruction, subspace extension, and subspace combination (Si, Yang, & Shen, 2024), with lower computational cost and space consumption.

3. Methods

3.1. Motivation

Initially, the resolution of our images (1512×2016) was significantly higher compared to the common resolutions (around 400×400) in highly cited papers. To save on computations, we wanted to scale down the resolution but were concerned about losing too much information after downsampling. Therefore, we realized that encoder-decoder based methods would best suit our needs. We sought an invertible 'operation' to easily reconstruct the original information before encoding.

We initially explored using various kernels, such as Gaussian, heat, and Poisson kernels, inspired by kernel methods and real analysis. The idea was to approximate a 'bad' function with a 'good' function using a 'good' kernel, as defined in real analysis (Stein & Shakarchi, 2009). However, these methods are computationally expensive, often requiring the computation of high-order derivatives via Taylor series. Given our limited computational resources as students, we had to compromise.

Ultimately, we accepted some loss of information during the encoding-decoding process but opted for bilinear interpolation to minimize this loss compared to using max-pooling, as demonstrated in Badrinarayanan's work (Badrinarayanan et al., 2017a).

We selected an encoder-decoder architecture, particularly the UNet, SegNet and DeepLabV3+ models, due to its ability to preserve spatial information throughout the network. This is crucial for tasks like semantic segmentation where labels are pixel-specific rather than region-specific, as in the Wild-Scene Dataset. This architecture also reduces computational load through convolutional layers, making it efficient for large-scale image analysis.

Encoder-decoder based methods like UNet, SegNet, and DeepLabV3+ are advantageous for their hierarchical feature extraction and precise localization capabilities through skip connections. These connections help retain fine-grained spatial details, which are essential for accurate segmentation. In contrast, fully connected networks or traditional convolutional neural networks without an encoder-decoder structure often fail to maintain the necessary spatial details.

3.2. Pre-processing

Preprocessing is an important aspect of improving the performance of image segmentation. After having a detailed look at the dataset, we decided to split the dataset and try to do some preprocessing to solve the following questions in the dataset.

3.2.1. Dataset Splitting. In the dataset, there are more than 9000 high-resolution images (2016×1512). With this volume and resolution, it's hard to train a model in a limited time duration. Therefore, we decided to take a random stratified sample of 50% of the images from the entire dataset and use it for all our models.

3.2.2. Frequency research. After a brief look at the label of the dataset, we found out that the dataset's classes are serious imbalanced. For example, the "fence" class only showed 0.04% pixels of the whole dataset and caused catastrophic predictions to all models in the WildScenes paper (Vidanapathirana et al., 2023). So, we decided to try to use a weighted cross entropy loss function to solve this problem and increase the performance of the models.

No.	Class	Percentage	Pixel Count
01	asphalt	<0.01%	124,981
02	dirt	8.93%	2,422,149,709
03	mud	0.06%	15,047,893
04	water	0.21%	58,185,431
05	gravel	0.34%	91,940,545
06	other-terrain	0.01%	3,968,625
07	tree-trunk	13.99%	3,793,706,766
08	tree-foliage	56.78%	15,400,343,509
09	bush	1.22%	330,763,422
10	fence	0.04%	10,189,508
11	structure	0.32%	86,517,870
12	pole	0.01%	3,481,029
13	vehicle	<0.01%	234,928
14	rock	0.06%	15,997,697
15	log	0.44%	120,107,589
16	other-object	0.10%	27,368,672
17	sky	6.94%	1,882,958,614
18	grass	10.54%	2,859,725,628

TABLE 1. PIXEL LEVEL CLASSES FREQUENCY RESEARCH

No.	Class	Percentage	Picture Count
01	asphalt	0.26%	23
02	dirt	86.64%	7,709
03	mud	3.05%	271
04	water	4.73%	421
05	gravel	3.97%	353
06	other-terrain	0.71%	63
07	tree-trunk	99.76%	8,877
08	tree-foliage	99.94%	8,893
09	bush	22.09%	1,966
10	fence	2.17%	193
11	structure	7.38%	657
12	pole	3.94%	351
13	vehicle	0.28%	25
14	rock	4.93%	439
15	log	45.45%	4,044
16	other-object	16.35%	1,455
17	sky	93.11%	8,285
18	grass	91.78%	8,167

TABLE 2. PICTURE LEVEL CLASSES FREQUENCY RESEARCH

But after defining the inverse class frequency according to the result of our frequency research to the loss function, we found out that although some low-frequency classes' IoU increased slightly, the mIoU decreased more.

As the weighted cross entropy loss function brought a negative impact on our mean measurement score (mIoU), we decided not to use it. We think this might be because the weighted loss function caused the model to overfit the minority class, resulting in reduced generalization ability on the test set.

3.2.3. White Balance. In the dataset, we can see a lot of pictures don't have a good white balance. A lot of them are skewed toward purple or blue. So, we decided to apply white balance directly to all the pictures in the dataset before it is used for training or testing, to avoid the internal inconsistencies within the dataset.

Figure 1. White Balance Demonstration

3.2.4. Resolution. Reducing image resolution in model training can significantly reduce the computational costs and speed up training, as we don't have much time and hardware resources, we decided to cut the height and width of all the dataset into 1/4 of the original, although reducing the resolution may cause information lost.

3.2.5. Normalization. Normalization can accelerate convergence and mitigate gradient vanishing and exploding. So we applied normalization for all the image before put it into the model.

3.2.6. Flares, Shadow and Lighting conditions. From the dataset, we can find out that many of the pictures have purple flares caused by sunlight, almost all the pictures have shadows and all the objects in the pictures are in different lighting conditions. Because flares and shadows can influence the objects' color and shape, and different lighting conditions can cause one class to have too many different colors, we think it's necessary to try to solve it to obtain the correct segmentation. For flares, we manually pick out the picture with purple flares and exclude them from training. For shadows and lighting conditions, we tried to align the brightness of the light and dark parts of the image to the mean part of it using 9-means clustering, to decrease the influence of the shadow. The result is as follows:

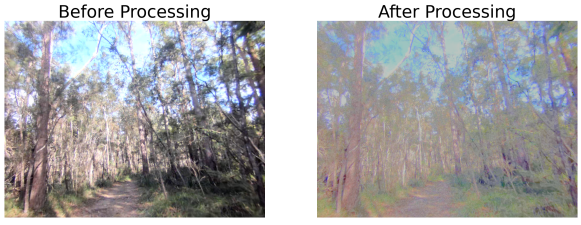

Figure 2. Shadow and Lighting Processing Demonstration

In the processing, firstly, we eliminated all pixels' RGB values all lower than 10 and used the Navier-Stokes-based method to refill that area, to avoid severe color drift after increasing brightness. Then, as mentioned before, we used 9-means clustering on the grayscale of the picture to get the mask of different

brightness levels. After that, we realign the brightness of different brightness levels into the mean part of the picture. And finally, regenerate the picture with different brightness delta with Gaussian blurred mask, for better gradient between different parts.

However, after a small-scale test using 5% of the data in the dataset, comparing the mIoU of using or not using this processing, there is no significant difference. As this processing will slow the training of the model, we decided not to use it. We think it might be because, after this processing, the color difference between the shadow and the object is still not emulated. So, there is no significant change in mIoU.

3.3. Deep learning models

3.3.1. DeepLabV3+. DeepLabV3+ (Chen et al., 2018) is an advanced semantic segmentation model designed to enhance the performance of its predecessors in the DeepLab family by incorporating an encoder-decoder architecture. The encoder component utilizes a deep convolutional neural network, such as ResNet-101, as its backbone for feature extraction. This backbone is augmented with an Atrous Spatial Pyramid Pooling (ASPP) module at the end, which is crucial for capturing multi-scale contextual information. The ASPP module achieves this by employing atrous (dilated) convolutions at various rates, allowing the model to effectively enlarge the receptive field without increasing the number of parameters or computational cost.

In DeepLabV3+, as the decoder part we extracts low-level features from the intermediate layers of ResNet-101 and processes them through a convolution layer to adjust the channel number, the decoder component is responsible for improving the spatial resolution of the feature maps produced by the encoder. This is accomplished by combining the low-resolution feature maps from the encoder with high-resolution feature maps from earlier layers in the network. Specifically, the low-resolution feature maps output by the encoder are concatenated with intermediate high-resolution feature maps, typically from a middle layer of the backbone network like ResNet. This fusion is followed by a series of convolutional operations and upsampling processes, which progressively refine the feature maps to restore spatial details. The final output of the decoder is a prediction map that matches the input image's size, providing detailed and accurate segmentation results.

The key techniques that enable the superior performance of DeepLabV3+ include the use of atrous convolutions and the ASPP module. Atrous convolutions are pivotal in expanding the receptive field, allowing the model to capture more extensive contextual information without adding extra parameters. The ASPP module, which integrates multiple atrous convolutions at different scales along with global average pooling, further enhances the model's ability to capture diverse contextual cues from the input image. The encoder-decoder structure of DeepLabV3+ allows the model to extract robust features and then refine these features to produce high-resolution segmentation maps, making it highly effective for tasks requiring precise delineation of object boundaries.

It is worth noting that during our implementation of DeepLabV3+, we experimented with both pretrained and non-pretrained versions of the model. Our observations indicated that the pretrained model achieved an approximate 0.1 improvement in the mean Intersection over Union (mIoU), demonstrating the significant advantage of using pretrained models.

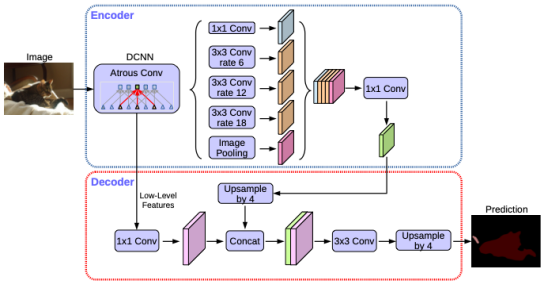

Figure 3. Structure of DeepLabV3+ (Chen et al., 2018)

3.3.2. Unet. UNet (Ronneberger et al., 2015) employs an encoder-decoder structure where the encoder path captures context through downsampling, and the decoder path performs upsampling to generate a high-resolution output. By combining low-level features from the encoder with the corresponding upsampled features in the decoder through skip connections, UNet maintains spatial resolution and detail, which is critical for pixel-wise predictions. The following picture shows the structure of our UNet:

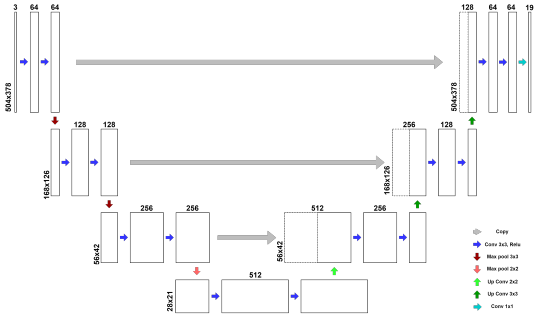

Figure 4. Structure of Our UNet

We used three encoder layers to balance resolution reduction and computational feasibility, selecting a resolution of 1512/4 × 2016/4, inspired by similar high-citation works like SegNet. Skip connections were implemented to preserve spatial context by linking encoder and decoder layers. The downsampling kernel sizes(3, 3, 2 for each encoder layers) were determined by the prime decomposition of 2016 and 1512, where $2016 = 2^5 \times 3^2 \times 7$, $1512 = 2^3 \times 3^3 \times 7$ and the upsampling kernel sizes(2, 3, 3 for each

decoder layers) were set to mirror the downsampling kernel size to ensure consistent spatial transformations. For upsampling, we chose bilinear operations due to their simplicity and effectiveness in maintaining spatial coherence.

However, after training the UNet model with 50% of the data, we observed that the mean Intersection over Union (mIoU) was approximately 38%, using a learning rate of 0.0003 and a batch size of 8. This result was not satisfactory. Consequently, we considered using the UNet model as our backbone for further tuning. We opted for Singular Value Decomposition (SVD) for this tuning process due to its favorable properties. This will be introduced in next section.

3.3.3. SegNet. Essentially, SegNet is an encoder-decoder type deep learning network. The encoder part has 16 layers, including the following layers:

1) 1st and 2nd layer: convolution layers consist of 64 filters each;
2) 3rd and 4th layer: convolution layers consist of 128 filters each;
3) 5th, 6th and 7th layer: convolution layers consist of 256 filters each;
4) 8th till 13th layer: convolution layers have 512 filters each.

Each convolution block is followed by a max pooling layer in the encoder part.
The previous part is followed by the decoder part which starts with an upsampling layer followed by convolution blocks in the reverse order, and finally ending with a softmax layer.
But in our SegNet model, we used the previous 34 layers of VGG16 as the encode block.

The advantages of Using VGG16 as encoder
When used as the encoder in SegNet, VGG16's ability to extract detailed and meaningful features significantly improves the quality of segmentation. This leads to more precise segmentation results, especially in complex scenes. Additionaly, one of the primary advantages of using VGG16 in SegNet is the ability to leverage pre-trained weights. VGG16 has been extensively trained on large-scale datasets such as ImageNet, capturing a wide array of features applicable to various visual tasks. By initializing SegNet with these pre-trained weights, we can accelerate the training process and achieve the convergence faster.

Evaluation about SegNet :

Strengths :

1) Efficient Memory Usage:
   SegNet uses a special decoder technique that reuses the max-pooling indices from the corresponding encoder layers, reducing the memory requirements.

2) Fast Inference:
   SegNet enables faster inference times, which is beneficial for real-time applications.

3) High-Resolution Segmentation:
   Due to using the max-pooling indices, SegNet can produce high-resolution segmentation maps, which is crucial for tasks requiring more details.

Weaknesses :

1) Performance on Complex Scenes:
   SegNet struggles with highly complex and cluttered scenes where the context is vital for accurate segmentation task. Like in this WildScene dataset, the information are complex and have many disturbing features such as flares and shadows.

2) Comparative Accuracy:
   Although SegNet is efficient, it often lags behind more recent architectures like DeepLabV3+ and UNet, particularly on challenging benchmarks.

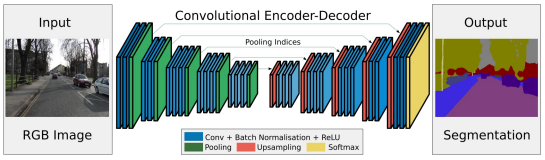

Figure 5. Structure of SegNet
(Badrinarayanan, Kendall, & Cipolla, 2017b)

3.4. Methods improvements applied to models

3.4.1. SVD Tuning. The following theorem and statements below reveal the relationship between τ and its adjoint transformation τ^*, exhibiting symmetrical behavior. Hence, during our tuning stage, we only applied linear transformations to the left singular vectors. This approach ultimately adjusts the singular values, thereby refining the weight matrix. We observed that after tuning, our model could recognize more details and perform better on minority classes, such as identifying artificial structures like cables.

Another advantage is that once the rank r of the operator τ is known, computation can be reduced since for $i > r$, the result is zero. However, we did not apply this in our work due to time constraints. Future work will focus on limiting the way we update the linear transformation to make it less computationally expensive by leveraging its mathematical properties and bound estimations.

Theorem 1. *(Roman, Axler, & Gehring, 2005) Let U and V be finite-dimensional inner product spaces over $\mathbb{C}$ or $\mathbb{R}$ and let $\tau \in L(U, V)$ have rank r. Then there is an ordered orthonormal basis $\mathcal{B} = (u_1, \ldots, u_r, u_{r+1}, \ldots, u_n)$ of U and an ordered orthonormal basis $\mathcal{C} = (v_1, \ldots, v_r, v_{r+1}, \ldots, v_m)$ of V with the following properties:*

1) $\mathcal{B}_r = (u_1, \ldots, u_r)$ *is an orthonormal basis for* $\ker(\tau)^\perp = \mathrm{im}(\tau^*)$
2) $(u_{r+1}, \ldots, u_n)$ *is an orthonormal basis for* $\ker(\tau)$
3) $\mathcal{C}_r = (v_1, \ldots, v_r)$ *is an orthonormal basis for* $\ker(\tau^*)^\perp = \mathrm{im}(\tau)$
4) $(v_{r+1}, \ldots, v_m)$ *is an orthonormal basis for* $\ker(\tau^*)$
5) *The operators* τ *and* τ^* *behave "symmetrically" on* $\mathcal{B}_r$ *and* $\mathcal{C}_r$*, specifically, for* $i \leq r$*,*

$$\tau(u_i) = s_i v_i$$

$$\tau^*(v_i) = s_i u_i$$

where $s_i > 0$ *are called the singular values of* τ*. The vectors* u_i *are called the right singular vectors for* τ *and the vectors* v_i *are called the left singular vectors for* τ*.*

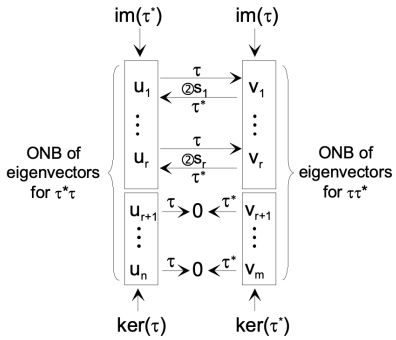

Figure 6. Illustration of Theorem 1 (Roman et al., 2005)

The matrix version of this theorem leads to the well known *singular value decomposition* of a matrix. The matrix of τ under the ordered orthonormal bases $\mathcal{B} = (u_1, \ldots, u_n)$ and $\mathcal{C} = (v_1, \ldots, v_m)$ is

$$[\tau]_{\mathcal{B},\mathcal{C}} = \Sigma = \mathrm{diag}(s_1, s_2, \ldots, s_r, 0, \ldots, 0)$$

Given any matrix $A \in \mathcal{M}_{m,n}$ of rank r, let $\tau = \tau_A$ be multiplication by A. Then $A = [\tau_A]_{\mathcal{E}_n,\mathcal{E}_m}$ where $\mathcal{E}_n$ and $\mathcal{E}_m$ are the standard bases for U and V, respectively. By changing orthonormal bases to $\mathcal{B}$ and $\mathcal{C}$ we get

$$A = [\tau_A]_{\mathcal{E}_n,\mathcal{E}_m} = M_{\mathcal{C},\mathcal{E}_m}[\tau_A]_{\mathcal{B},\mathcal{C}} M_{\mathcal{E}_n,\mathcal{B}} = P\Sigma Q^*$$

where $P = M_{\mathcal{C},\mathcal{E}_m}$ is unitary (orthogonal for $F = \mathbb{R}$) with ith column equal to $[v_i]_{\mathcal{E}_m}$ and $Q = M_{\mathcal{E}_n,\mathcal{B}}$ is unitary (orthogonal for $F = \mathbb{R}$) with ith column equal to $[u_i]_{\mathcal{E}_n}$.

We assume the existence of an optimal weight matrix $\mathbf{W}^*$ (Ding et al., 2023; Si et al., 2024). In the context of functional analysis (Bühler & Salamon, 2018), Most of Banach spaces exhibit the approximation property, if they fit two conditions. However some of Banach spaces have notable counterexamples (Enflo, 1973; Szankowski, 1978). We aim to transform our current weight matrix $\mathbf{W}$ to reduce the loss, effectively making the transformed weight $\phi(\mathbf{W})$ approach the optimal weight $\mathbf{W}^*$, as shown below:

$$\min_{\phi} \ell(\mathbf{W}^*, \phi(\mathbf{W})),$$

where ℓ measures the difference between the two matrices.

Given that the weight matrix $\mathbf{W}$ can be decomposed using singular value decomposition (SVD), which means $\mathbf{W} = P\Sigma Q^*$, our focus is on transforming one of its components. Simply multiplying by a scalar won't alter the direction, so to mitigate the influence of an inferior weight matrix, as seen in pre-trained UNet models, we chose a linear transformation ϕ on the left singular vectors P. Notably, applying linear transformation on both left and right singular vector, scaling up singular values are other feasible ways too. Our choice is motivated by its 'symmetry' as shown in previous theorem and simplicity.

3.4.2. Combined Loss Function. We also explored defining our own loss function, hoping it would help our model achieve faster convergence and avoid local minima. Inspired by active boundary method (Wang et al., 2022), we revisited the work on active contours without edges by Tony F. Chan and Vese (Chan & Vese, 2001). Our motivation was that if we could first identify all object boundaries, we could fill different 'colors' inside these boundaries, with further refinement learned by our neural networks. In their paper, they defined an energy function as follows:

$$\begin{aligned} &F_\epsilon(c_1, c_2, \phi) \\ &= \mu \int_\Omega \delta_\epsilon(\phi(x,y)) |\nabla \phi(x,y)| \, dx\, dy \\ &\quad + \nu \int_\Omega H_\epsilon(\phi(x,y)) \, dx\, dy \\ &\quad + \lambda_1 \int_\Omega |u_0(x,y) - c_1|^2 H_\epsilon(\phi(x,y)) \, dx\, dy \\ &\quad + \lambda_2 \int_\Omega |u_0(x,y) - c_2|^2 (1 - H_\epsilon(\phi(x,y))) \, dx\, dy, \end{aligned}$$

where c_1 and c_2 are given by

$$\begin{cases} c_1(\phi) = \text{average}(u_0) \text{ in } \{\phi \geq 0\} \\ c_2(\phi) = \text{average}(u_0) \text{ in } \{\phi < 0\}, \end{cases}$$

u_0 is the given image and H is Heaviside function and δ_0 is the one-dimensional Dirac measure, and defined, respectively, by

$$H(z) = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{if } z < 0, \end{cases} \qquad \delta_0(z) = \frac{d}{dz} H(z).$$

H_ϵ is any $C^2(\Omega)$ regularization of H, and $\delta_\epsilon = H'_\epsilon$. F_ϵ is the associated regularized functional hence the associated Euler-Lagrange equation for the unknown function ϕ can be computed. See further reference(Chan & Vese, 2001).

We implemented this approach, and as demonstrated, it was remarkably effective at capturing accurate boundaries using intensity and differential geometrical invariants. However, when we incorporated

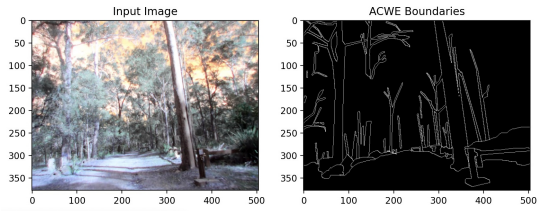

Figure 7. Boundaries Detected by Chan's Algorithm

this energy function into our loss function and tested it on a toy dataset (5 pictures), the performance was worse than before, failing to meet our initial expectations. We suspect this method led our model to learn class-wise instead of pixel-wise, potentially causing it to be trapped in local minima. Due to time constraints, we suspended this exploration, but it remains a focus for future work.

Ultimately, we used a combined loss function (Li et al., 2021) comprising cross-entropy loss, dice loss, and mIoU loss. Unlike dice loss and mIoU loss, which compute loss based on specific regions, cross-entropy computes loss pixel-wise. Since our primary metric is mIoU, we included an mIoU term in our combined loss function. Thus, our model updates in two ways: pixel-wise (improving accuracy) and region-based (potentially exploring new classes). This dual approach is the main reason why our results using the combined loss function are better than using only cross-entropy loss.

4. Experimental results

For all three models, we used one-quarter of the original image resolution as our training and testing image size. Approximately half of the original dataset (4437 images) was used for our training, validation, and test sets, with the training set comprising 4437 * 0.8 * 0.8 images, the validation set 4437 * 0.8 * 0.2 images, and the test set 4437 * 0.2 images. Additional setup details can be found in Table 3 and Table 4.

We used accuracy, mean Intersection over Union (mIoU), and Dice coefficient to assess model performance. For the number of epochs, we set the max epoch to 100-300 and employed an early stopping strategy. The final number of epochs is shown in Table 3.

In Figure 8, we present the results of our experiments with the three models on the WildScenes dataset and compare the classes of the three models. Notably, the DeepLabV3+ model with pre-trained weights outperformed the other two models across almost all classes.

From frequency analysis, we can see that the long-tailed distribution of category frequencies in natural environments results in lower IoU scores for rare classes, skewing the performance. In our training and testing datasets, the most common samples are dirt, tree-trunk, tree-foliage, sky, and grass. Correspondingly, all models perform well on these classes, with each model achieving an IoU above 50%. The 5 least common classes are mud, other-terrain, fence, rock, and other-object, which also have the poorest performance, with almost all IoU scores below 20%. In the predictions of the SegNet model, some of these classes are not detected at all. This demonstrates that the low IoU for rare classes during testing is due to class imbalance, resulting in a lack of training samples for the network to learn from. However, despite bush being relatively frequent, all three models predict this class with an IoU below 10%. On the other hand, although other-terrain is rare, models other than SegNet achieved some results, with IoU scores of 18%(DeepLabV3+) and 14%(UNet) respectively.

Another observation is that the differences between the three models are not significant for most classes. For the most common classes, tree-foliage, the results of the three models are 84%, 83%, and 83% respectively. However, for rare classes, the differences between the models become substantial. The model with the highest mIoU, DeepLabV3+, outperforms the other two models on rare classes, indicating that rare classes are an area for future exploration. It is also likely that models would require more samples to learn the features of rare classes if pre-training is not performed.

5. Discussion

In general, as illustrated in Figure 8, all models achieved mean Intersection over Union (mIoU) comparable to the baseline, despite utilizing only 50% of the data and significantly fewer epochs compared to the baseline (Vidanapathirana et al., 2023). However, Figure 8 also highlights that all models underperformed in classes such as mud, bush and rock, attributable to class imbalance. The frequency analysis in Section 3.2.2 shows that classes like mud, bush, and rock constitute only 0.06%, 1.22%, and 0.06% of the total pixel data, respectively. Moreover, shadows and flares in our dataset may hinder model performance during training. In contrast, dominant classes such as tree-foliage, tree-trunk, and grass, with pixel frequencies of 56.78%, 13.99%, and 10.54%, achieved significantly higher IoU scores as displayed in Figure 8. Notably, even though tree-trunk and grass have lower overall frequencies, they each appear in over 90% of the pixels in a single image, leading to their relatively high IoU scores.

Additionally, DeepLabV3+ outperformed SegNet and UNet overall. This superior performance can be attributed to the use of a pre-trained model in DeepLabV3+, which was trained on a larger and less imbalanced dataset, thus enhancing model performance. This is evident as the non-pre-trained version of DeepLabV3+ achieved a lower mIoU score. DeepLabV3+ also integrates both feature extraction and an encoder-decoder process, which likely contributes to more efficient training and better performance, even when both DeepLabV3+ and SegNet used pre-trained models. SegNet and UNet displayed similar mIoU scores due to their analogous structures, despite the modification of using VGG16 as

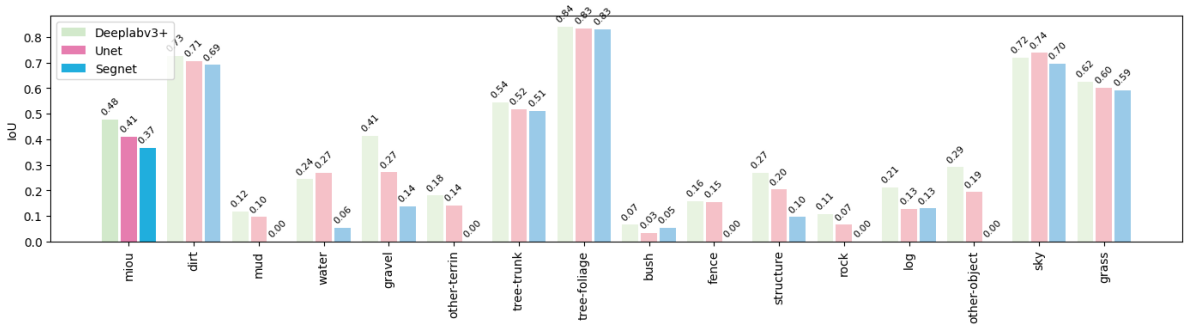

Figure 8. IoU Comparison of 3 Models

	Pre-trianing	Parameter size	Learning rate	Batch size	Number of epochs	optimizer	weight decay
DeepLabV3+	Yes	232.68 M	0.0003	4	42	Adam	1e-5
Segnet	Part of	63.366M	0.001	4	200	SGD	1e-5
Unet	No	107.75M	0.00035	8	24	Adam	1e-5

TABLE 3. EXPERIMENTAL SETUP AND PARAMETER CONFIGURATION DETAILS

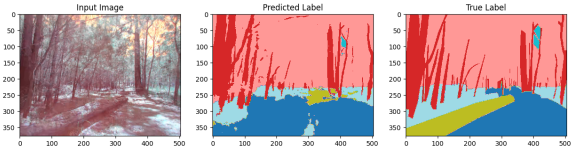

Figure 9. DeepLabV3+ without Pretraining

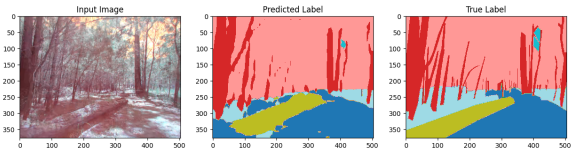

Figure 10. DeepLabV3+ with Pretraining

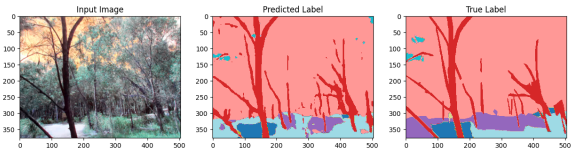

Figure 11. SegNet Prediction using 50 Percent Data

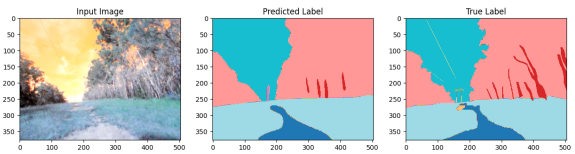

Figure 12. Unet using 50 Percent Data

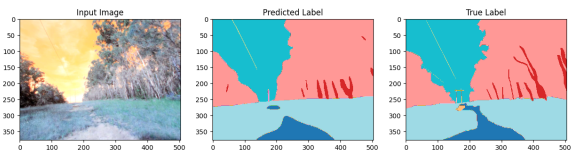

Figure 13. Unet after SVD Tuning using 50 Percent Data

	GPU	Operating system	Deep learning framework
DeepLabV3+	1x RTX 4090	Linux	Pytorch
Segnet	1x RTX 4090	Linux	Pytorch
Unet	1x A100	Windows	Pytorch

TABLE 4. MODEL ENVIRONMENT

an encoder in SegNet. However, UNet has a larger number of total parameters (over 100 MB) compared to SegNet (around 60 MB). In terms of memory storage, SegNet is preferable since it achieves similar results but with less memory usage. Nonetheless, after applying SVD tuning to UNet, it outperformed SegNet. Due to time constraints, we only applied SVD tuning to UNet.

For UNet, after tuning, our model achieved a mean Intersection over Union (mIoU) of nearly 41% and recognized finer details than before. This improvement was achieved using only 50% of the Wildscence Dataset and fewer epochs compared to the baseline paper (Vidanapathirana et al., 2023), without relying on any pre-trained models from larger and better-balanced datasets.

However, our UNet model has its shortcomings, beyond the larger number of parameters, that need to be addressed in future work. A significant issue is that the training log indicates overfitting to the training data, especially after SVD tuning, despite the increase in mIoU. This overfitting might be due to the combined loss function used during tuning. While the mIoU loss decreased, the other two terms (cross-entropy loss and dice loss) increased.

Additionally, the singular value decomposition (SVD) is generally not unique (Bisgard, 2020) (if $n < m$), which may hinder our model's learning process. Addressing these issues and further refining our approach will be the focus of future research.

6. Conclusion

In this project, we use three different models to deal with the segmentation task, DeepLabV3+,

SegNet and UNet. Besides the basic model structure, we added more techniques to improve the result of these three models. List as follows:

1) We applied the white balance to all pictures as part of our data-preprocessing. And this work help to avoid the internal inconsistencies successfully.

2) We used the self-defined combined loss function, which takes the loss of dice, accuracy and mIoU into consideration. And this work has increase the mIoU of three models and improve the accuracy of the segmentation.

3) We used the SVD tuning (most simple one) to deal with the classes which has a small sample size, adding it to UNet, and the mIoU evaluation result with SVD tuning of UNet has increased compared to the one without SVD tuning.

4) We also tried to use the pre-trained weight in SegNet and DeepLabV3+, and this acceralated the training procession and also led to a early achievement of convergence. In addition, the mIoU results of these two models have also increased a lot, cause the pre-trained model has been trained based on a large amount of data, the result will be more robust and accurate.

Apart from the one showing above, we also tried some other methods aiming to improve the result of our models, listed as follows:

1) We tried to deal with the flares, shadow and lighting condition, but the result did not improved significantly.

2) We tried the active contour, which can detected the edges of the objects in the image pretty precise. But when adding it into the model, the performance of the model did not improved a lot.

Future Work

As for the future improvement, here are some future work might improve the models' performance:

1) Computational optimization:
We can explore invertible kernels or dimensionality reduction techniques, like KPCA (Kernel Principal Component Analysis), which reduces computation while retaining spatial information.

2) Model performance enhancement:
We can try to address some shadows and flares in the images as a step of data preprocession, and also refine predictions with post-processing techniques like Genetic algorithms and CRFs.
Besides, we can also consider more additional convolutional layers to be added into the suitable models for better feature extraction.

3) Data diversity and balance:
Cause this dataset faced the problem of classes imbalance, so we can apply some techniques like GANs for data generation and also apply class-specific weights to tackle imbalance.

4) Subspace-decomposition:
For future work, we can also explore recent advancements in subspace decomposition to enhance feature representation and improve model accuracy. This will provide more robust and discriminative features by effectively capturing the underlying structure of the data, potentially leading to significant performance gains in our models.

5) Use Self-Attention Mechanism:
The self-attention mechanism can directly relate different parts of the input sequence, making the network more effective for long sequences, and it will improve the accuracy of segmentation. So it is possible that if we apply this technique, the model will be more robust with some disturbing issues.

Acknowledgments

We would like to extend our gratitude to the following individuals and resources that significantly contributed to our work:

- The recommendation algorithm that led us to discover the subspace tuning paper.
- Chongjie Si, the author of the subspace tuning paper, for responding to our inquiries, providing invaluable insights on tuning, and recommending relevant papers and useful experiences in machine learning.
- Dr. Dong Gong and Dr. Erik Meijering, for their suggestions on updating our GPUs, offering insightful advice on encoder-decoder methods, and confirming the validity of our efforts.
- Dr. Michael Bain, for encouraging us to explore more transformations when applying SVD tuning.
- The tutors in COMP9417 and COMP9517, for their helpful responses to our queries.
- Every teammate in our group, for their dedication to our work and the many late nights spent advancing our project.

References

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017a). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, *39*(12), 2481–2495.

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017b). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(12), 2481-2495. doi: 10.1109/TPAMI.2016.2644615

Basavaprasad, B., & Ravindra, S. H. (2014). A survey on traditional and graph theoretical techniques for image segmentation. *Int. J. Comput. Appl*, *975*, 8887.

Bisgard, J. (2020). *Analysis and linear algebra: the singular value decomposition and applications* (Vol. 94). American Mathematical Soc.

Bühler, T., & Salamon, D. A. (2018). *Functional analysis* (Vol. 191). American Mathematical Soc.

Chan, T. F., & Vese, L. A. (2001). Active contours without edges. *IEEE Transactions on image processing*, *10*(2), 266–277.

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).

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *20*(2), 215–232.

Dickson, M. C., Bosman, A. S., & Malan, K. M. (2022). Hybridised loss functions for improved neural network generalisation. In *Pan-african artificial intelligence and smart systems* (p. 169–181). Springer International Publishing. Retrieved from http://dx.doi.org/10.1007/978-3-030-93314-2_11 doi: 10.1007/978-3-030-93314-2_11

Ding, N., Qin, Y., Yang, G., Wei, F., Yang, Z., Su, Y., ... others (2023). Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, *5*(3), 220–235.

Enflo, P. (1973). A counterexample to the approximation problem in banach spaces. *Acta Mathematica*, *130*(1), 309–317.

Ghosh, S., Das, N., Das, I., & Maulik, U. (2019). Understanding deep learning techniques for image segmentation. *ACM computing surveys (CSUR)*, *52*(4), 1–35.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 580–587).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Li, X., He, M., Li, H., & Shen, H. (2021). A combined loss-based multiscale fully convolutional network for high-resolution remote sensing image change detection. *IEEE Geoscience and Remote Sensing Letters*, *19*, 1–5.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the ieee international conference on computer vision* (pp. 2980–2988).

Pal, N. R., & Pal, S. K. (1993). A review on image segmentation techniques. *Pattern recognition*, *26*(9), 1277–1294.

Papandreou, G., Chen, L.-C., Murphy, K. P., & Yuille, A. L. (2015). Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of the ieee international conference on computer vision* (pp. 1742–1750).

Roman, S., Axler, S., & Gehring, F. (2005). *Advanced linear algebra* (Vol. 3). Springer.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–miccai 2015: 18th international conference, munich, germany, october 5-9, 2015, proceedings, part iii 18* (pp. 234–241).

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., & Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, *28*.

Si, C., Yang, X., & Shen, W. (2024). See further for parameter efficient fine-tuning by standing on the shoulders of decomposition. *arXiv preprint arXiv:2407.05417*.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sørensen, T. (1948). *A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons*. Munksgaard in Komm. Retrieved from https://books.google.com.au/books?id=rpS8GAAACAAJ

Stein, E. M., & Shakarchi, R. (2009). *Real analysis: Measure theory, integration, and hilbert spaces*. Princeton University Press.

Sujata Saini, K. A. (2014). A study analysis on the different image segmentation techniques. *International Journal of Information Computation Technology*, *4*, 1445-1452.

Szankowski, A. (1978). Subspaces without the approximation property. *Israel Journal of Mathematics*, *30*, 123–129.

Vidanapathirana, K., Knights, J., Hausler, S., Cox, M., Ramezani, M., Jooste, J., ... others (2023).

Wildscenes: A benchmark for 2d and 3d semantic segmentation in large-scale natural environments. *arXiv preprint arXiv:2312.15364*.

Wang, C., Zhang, Y., Cui, M., Ren, P., Yang, Y., Xie, X., … Xu, W. (2022). Active boundary loss for semantic segmentation. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 36, pp. 2397–2405).

Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). Pyramid scene parsing network. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2881–2890).