# COMP9517 GROUP PROJECT REPORT

Kartikaye Chandhok
zid: z5285022

Andy Wang
zid: z5360671

Jerry Cheng
zid: z5484763

Liangchao Chen
zid: z5494376

Jianning Yu
zid: z5499765

## I. Introduction

In this project, we aim to perform segmentation of sea turtles in underwater images. Segmentation is a crucial computer vision task, particularly in applications involving biological or environmental monitoring, where accurately identifying and distinguishing objects or regions is essential.

Our dataset comprises underwater images of sea turtles, annotated with masks that distinguish each part of the turtle. Before the model implementation process, I first need to process the data carefully, divide the dataset into training set, validation set and test set according to the split_open of the csv file, and save the image path information. Because it is mentioned in the dataset paper that open-set splitting gives a much more realistic performance.

The project workflow includes dataset preprocessing, model training with periodic validation to monitor performance, and final testing on an unseen dataset to evaluate the model's generalization ability. Throughout this process, we employ Intersection over Union (IoU) and mean Intersection over Union (mIoU) as evaluation metrics to quantify the model's accuracy for each segmented class. In order to achieve this, we will compare the IoU of 4 different deep learning methods and an ensemble model and anaylse their results.

## II. Literature Review

The study centered in the research paper is focused on identifying turtles and their various body parts - specifically the head, flippers and carapace. One of the key features of the study is the generation of the dataset comprising over 8700 photographs of 438 individual turtles collected over 13 years. The identification and annotations are done manually by experts who segment relevant body parts to identify unique patterns in each individual – a process that is labor-intensive and not scalable- thus making this a computer vision task which could be automated to an image segmentation or object detection task.

With rapid growth of digital datasets over decades of ecosystem observation has created a demand for automated computer vision methods in the wildlife photography domain, that can perform identification in the face of individual variability. In computer vision with object recognition tasks, one of the most important processes is feature representation.

Early identification techniques relied on local feature descriptors and manually matched patterns to tell individual animals apart. Techniques such as SIFT and Superpoint extracted keypoints and descriptors that align similar markings across images, thus enabling the identification of individuals through their unique patterns. Subsequent applications like Hotspotter further extended this process by matching specific markings through similarity scores between the descriptors. However, these foundational approaches clearly had some limits in treating variable backgrounds and lighting conditions [11].

Image partitioning is the most basic established method based on traditional segmentation methods like thresholding, and K-means clustering which depends on pixel intensity groupings or color-based pixel grouping. However, such techniques often fail in natural and unstructured environments because of overlapping intensities and complicated backgrounds [12].

As the image datasets became increasingly complex, the models had to be adapted to cope with diversified environments, which gave the key to new deep learning segmentation and re-identification techniques that are increasingly accurate and robust. To further improve the individuality of animal encodings, deep learning-based feature extraction models have proven to be very efficient. Metric Learning methods, trained on different loss functions, utilize deep networks to learn high dimensional embeddings encoding how to best separate subjects.

There are two different kinds of segmentation processes associated with deep learning:- instance and semantic. Instance segmentation involves delineating an instance of an object in an image, while semantic segmentation classifies each pixel into a category in an image.

In the context of the study, accurate segmentation significantly enhances the feature extraction phase of animal re-identification. Thus the study discusses several image segmentation processes mostly instance segmentation -Mask R-CNN, Hybrid Task Cascade (HTC), Mask2Former model all supported using ResNet50 and Swin-B transformer using the MMDetection framework. The whole study highlights that re-identification accuracy of the end-to-end system was directly influenced by the performance of segmentation. Segmentation of these regions by the re-identification models reduces noise from backgrounds and irrelevant parts, enabling one to achieve higher accuracy and robustness in distinguishing individual animals.

However, for this project semantic segmentation would be more ideal than instance segmentation as our task is directly related to differentiate different body parts at a pixel level rather than differentiating between individual turtles within a single image, since there is no need to identify multiple instances of turtles within the dataset. FCN's, PSPNet, DeepLab, U-Net, SegNet are some of the models each characterized by their unique features and proven performance in dense segmentation tasks [4].

The models chosen for this project are thus based on this approach and fine tuned for the scope of this project. The

Fully Convolutional Networks (FCN's) as the name suggests is fully convolutional and thus could be trained for pixel classification tasks. Building on FCN we have U-Net - another fully convolutional network based on encoder-decoder architecture which is good at capturing the finer details of the image, especially small spatial features like the head and flippers. Finally DeeplabV3 and DeeplabV3+ uses Atrous Spatial Pyramid Pooling (ASPP) for contextual capturing and learning of the various turtle features. These models and their performance evaluations are discussed in depth further [2][5][8].

## III. METHODS

**Dataset:** We use the SeaTurtleID2022 dataset, which cDataset: We use SeaTurtleID2022 dataset which contains 8729 images divided into training, validation and test sets. There are no duplicate images in the training, validation and test sets. Each image is labelled with detailed information such as head, fin limbs and carapace.

**Data preprocessing:** we resize the images to 512x512 size and convert to tensor format. Masked data was interpolated using nearest neighbour interpolation to maintain label integrity.

**Dataset Classes:** We set up a dataset class that labels head, fin limbs and carapace as separate categories. It requires image path and annotation data to generate the corresponding masks that allow the model to distinguish the location of each part of the image.

**Training process and loss caculation:** During the training process, we used the cross-entropy loss function and the Adam optimiser [6]. In each period, the model learnt to segment the different parts of the turtle by calculating the loss, performing backpropagation and updating the parameters.

**mIoU evaluation:** in the validation and testing phases, we evaluate the segmentation accuracy of the model by calculating the IoU per class per image and the average IoU per class (mIoU) [7]. Specific evaluation results will be shown in the experimental results section, including analyses of success and failure cases.

For this task, we use four models as base models, and we choose build a weighed voting ensemble model base on these four models in order to achieve the best performance in segmenting the SeaTurtleID2022 dataset.

### *A. FCN*

Fully Convolutional Networks (FCNs) are a deep learning architecture specifically designed for image segmentation tasks. They utilize convolutional layers, which makes them highly efficient for dense, pixel-level classification. FCNs can accept input images of any size and produce an output that maintains the spatial structure of the input. This capability allows them to perform tasks such as semantic segmentation, where every pixel in the image needs to be classified. The key techniques used in FCNs include:

*1) **Fully Convolutional Architecture:** By replacing fully connected layers with convolutional layers, FCN retains the spatial structure of the input image. According to the announcement of Long et al. (2015), This design allows the model to generate segmentation outputs with dimensions consistent with the input image and supports inputs of varying sizes [9].*

*2) **Upsampling Techniques**: FCN employs transposed convolutions or bilinear interpolation to upsample low-resolution feature maps back to the original image size. Additionally, skip connections are introduced to combine high-resolution features from shallow layers with semantic features from deeper layers, enhancing the detail accuracy of segmentation results [10].*

*3) **Multi-scale Feature Fusion**: By integrating features from both shallow and deep layers, FCN captures high-resolution spatial details and low-resolution semantic information simultaneously, improving both boundary precision and overall segmentation quality.*

As a cornerstone in the field of image segmentation, FCN advances segmentation performance through its efficient network design and flexible upsampling strategies, paving the way for subsequent models such as U-Net, DeepLab and Mask-RCNN.

### *B. U-net*

U-Net was chosen as a model for this task for its high efficiency of segmentation for small datasets. It is based on fully convolutional neural networks and originally used for biomedical image segmentation, proven to be highly effective in scenarios where data is limited, this is reinforced by *Ronneberger, O., Fischer, P., Brox, T. (2015) which stated "The U-Net achieves significantly better performance than other methods, with a 92% IoU for cell segmentation, surpassing the second-best algorithm at 83%.".[13]* Similarly, for this sea turtle segmentation task, the remaining data after data splitting and cut down of training sample to half due to the lack of computational power matches this condition perfectly, and the high IoU results reflects U-Net's capability of handling segmentation tasks. The model is structured in a symmetric encoder-decoder fashion, with additional skip connections between corresponding layers, this allows the recovery of spatial details, which is a crucial property for segmenting fine structures like turtle flippers and edges. Details of main features and structures are below:

*1) **Encoder-Decoder Structure:** U-Net uses encoder-decoder architecture where encoder is considered as the contraction path and the decoder functions as the expansion path. Encoders consist of a series of convolutional and max-pooling layers, after taking in some variable length sequence as input, it then extracts hierarchical features, and gradually reducing the spatial dimensions while increasing the depth of feature maps and transforms it into a state with a fixed shape. This design has a great capability of capturing context stated, "The u-shaped architecture with a contracting path and an expansive path allows the network to capture context while enabling precise localization."[13] In the same paper it mentioned*

*the structure of the encoder, where each encoder involves two 3x3 convolutions followed by a ReLU activation and a 2x2 max-pooling operation with a stride of 2 for down-sampling. The decoder on the other hand, reconstructs the spatial dimensions of the feature maps to match the input image size. It consists of transposed convolutional layers for up-sampling which allows concatenation of corresponding feature map from the encoder via skip connections to restore fine-grained spatial details.*

*2)* ***Skip Connections:*** *Skip connections directly link the feature maps from the encoder to the decoder, this is typically mentioned by Ronneberger, O., Fischer, P., Brox, T. (2015) " Skip connections provide the expansive path with essential high-resolution features from the contracting path, ensuring that spatial information is preserved even after multiple down-sampling steps."[13] These connections help preserve spatial information lost during down-sampling. In our model, resnet34 is used as the backbone of the U-Net model. The paper Prabakaran, J., Selvaraj, P. (2023) highlighted ResNet's ability to categorizing features while it analyzed the ResNet-50 algorithm with the skip connection principle efficiently identifies and categorizes features, enabling robust predictions even in complex classification problems, and it mentioned skip connections in ResNet help preserve low-level features across layers, ensuring important hierarchical features are not lost for deep architectures.[14] This principle enhances the encoder of U-Net when ResNet-34 or ResNet-50 is used as the backbone, improving feature extraction and representation for segmentation. Moreover, from our group's previous studies, the skip connections also help to mitigate the vanishing gradient problem which is a common issue for deep networks.*

*3)* ***Final Output Layer:*** The final layer in the U-Net model is a 1x1 convolution to match the number of channels with number of segmentation classes. In our experiment with the model, SoftMax activation function is selected to generate per-pixel class probabilities.

### *C. Deeplabv3*

DeepLabV3 was selected for this project due to its effectiveness in high-precision semantic segmentation. The model employs dilated convolution and the ASPP module, a combination introduced by Chen et al. (2017) to address feature resolution loss and the challenge of multi-scale object processing in segmentation tasks[1][5]. It performs 85.7% on the PASCAL VOC 2012 test set and achieves a mIoU of 81.35 on the CitySpaces test dataset[1]. The previous result shows that it has robustness and effectiveness in segmentation tasks.

*1)* ***ASPP and Dilated convolution:*** *The key part ASPP module of deeplabv3 contains several independent dilated convolutional layers, each of which processes the image together with different dilation rates, enhancing the ability to extract multi-scale features. For different parts of the image, such as the shell and head of a turtle, ASPP uses different dilation rates to obtain information. This design allows the model to recognise both the overall structure and finer details, enabling precise segmentation.*

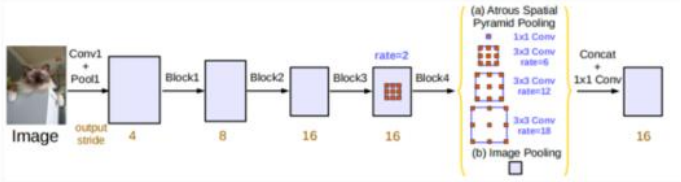



Fig. 1. Dilated Convolution (ASPP) parallel module for image-level feature enhancement. (Chen et al., 2017).

### *D. Deeplabv3+*

Because of Deeplabv3, we know that it has an enhanced version, Deeplabv3+. The paper by Chen et al. (2018) shows that Deeplabv3+ also performs well in image segmentation. What is different from Deeplabv3 is that it contains a decoding module and dilated separable convolution, which improves the accuracy of semantic segmentation [2].

*1)* ***Decoding Module:*** *The decoding module actually recovers some details of the image, especially the details at the edges. It is similar to the decoding module of U-net, and it also combines shallow features with deeper semantic information through "skip connections" to improve segmentation accuracy. Therefore, this decoding block can segment images well in complex scenes with detailed boundaries. In short, the decoding module improves the image resolution and allows the model to better identify edges, which is critical for image segmentation tasks.[2, 8].*

*2)* ***Atrous separable convolution:*** *DeepLabV3+ introduces null separable convolution, which decomposes the standard convolution into depth convolution and point convolution. This decomposition reduces computational cost while maintaining the effectiveness of capturing image contextual information. This innovation allows the model to expand the receptive field, thereby improving its ability to capture finer image details without significantly increasing the computational load [2].*

### *E. Weighted Voting Ensemble Model*

After the completion of the 4 base models, we introduce this weighted voting ensemble model to decide whether using manipulated weights by performance will leverage the strengths of individual models to achieve improved segmentation accuracy. After examining the results shown in part IV below, we decided to use 30% weight for the 3 good performing models including FCN, U-Net and Deeplabv3 and only giving 10% weight for the Deeplabv3+ model due to its relatively bad performance compared to the other 3. We achieve this by saving the model during training phase and use these stored model weights, we run them against the testing set, and using the weight to segment each pixel.

## IV. Experimental Results

### A. Models setup

Before comparing the four models, we tried to make the parameters of each model (e.g. optimiser and learning rate) as consistent as possible in order to better compare the models.

***Pre-training weight and encorder weight:*** *In order to better test the ability of the models themselves, we do not set pre-training weights or encoder weights for any of our four models.*

```
encoder_weights=None (pretrained=False)
```

Fig. 2. The code about pre-training weight/encoder weight

***Loss Function:*** *All four models use a cross-entropy loss function to calculate the training loss.*

```
criterion = torch.nn.CrossEntropyLoss()
```

Fig. 3. The code about loss function

***Optimizer and Learning rate:*** *Four models use Adam optimizer and learning rate is 0.0001.*

```
optimizer = optim.Adam(model.parameters(), lr=0.0001)
```

Fig. 4. The code about optimizer and learning rate

*1)* ***FCN:*** *In the FCN model, we selected ResNet50 as the backbone. To adapt the model to the scenario where the dataset contains only 4 categories (including the background), we modified the final classification layer of the model to ensure that the output dimension matches the requirements of this task, and we did not use pre-training weight.*

```
model = models.segmentation.fcn_resnet50(pretrained=False)
```

Fig. 5. The code about FCN setup

```
model.classifier[4] = torch.nn.Conv2d(512, num_classes, kernel_size=(1, 1))
```

Fig. 6. The code about classifier in FCN

*2)* ***U-net:*** *Different to the other models, ResNet34 is selected as the backbone while keeping the same loss function and optimizer. By selecting resnet34 as the backbone it provides several advantages. Firstly, it reduces the computational complexity, and this is an essential factor for our task as out of memory errors are often met which led to the cut down of training sample (For U-Net, the training data is reduced to half). Furthermore, it is noticed that U-Net has very high IoU values during training and testing phase, and it often converge early at epoch 15ish. Therefore, the use of a less deep model as backbone reduces the risk of overfitting while still benefiting from the relatively deep network's advantages. The loss function and optimizer are maintained as cross entropy for its good properties for classification and segmentation tasks, and Adam with learning rate of 0.0001 is kept from our first trial as we found it reaches a good minima position with just small oscillations. When it comes to our code, U-Net is not available in torch/torchvision, therefore, segmentation model's library was used to directly use the model. We did not choose the encoder weights in U-net.*

```
model = Unet('resnet34', classes=num_classes, activation=None, encoder_weights=None)
```

Fig. 7. The code about U-net setup

*3)* ***Deeplabv3:*** *In the deeplabv3 model, we selected resnet50 as the backbone, and in addition to the same loss function and optimiser, in order to adapt the model to the situation where there are only 4 categories (including background) in this dataset, we modified the final classification layer of the model to ensure that the output dimensions are consistent with this task Consistency and did not choose the pre-training weights.*

```
model = torchvision.models.segmentation.deeplabv3_resnet50(pretrained=False)
```

Fig. 8. The code about Deeplabv3 setup

```
# Modify the category header of the model to fit the number of custom categories
model.classifier[4] = torch.nn.Conv2d(256, num_classes, kernel_size=(1,1))
```

Fig. 9. The code about classifier in Deeplabv3

*4)* ***Deeplabv3+:*** *In the deeplabv3+ model, in addition to the same loss function and optimiser, although the option to set the encoder has been added, we did not set a deeper encoder at the beginning, but chose the default resnet50, mainly for comparison with other models and did not choose the encoder weights.*

```
model = smp.DeepLabV3Plus(
    encoder_name="resnet50",
    encoder_weights=None,
    in_channels=3,
    classes=num_classes
)
```

Fig. 10. The code about Deeplabv3+ setup

### B. Results of mIoU

TABLE I. The final test data set mIoU of all models

| Model name | 'carapace' mIoU | 'flipper' mIoU | 'head' mIoU | Average mIoU |
|---|---|---|---|---|
| FCN | 92% | 84% | 80% | 85% |
| U-net | 92% | 84% | 81% | 86% |
| Deeplabv3 | 92% | 85% | 82% | 86% |
| Deeplabv3+ | 85% | 69% | 69% | 74% |

### C. Successful examples

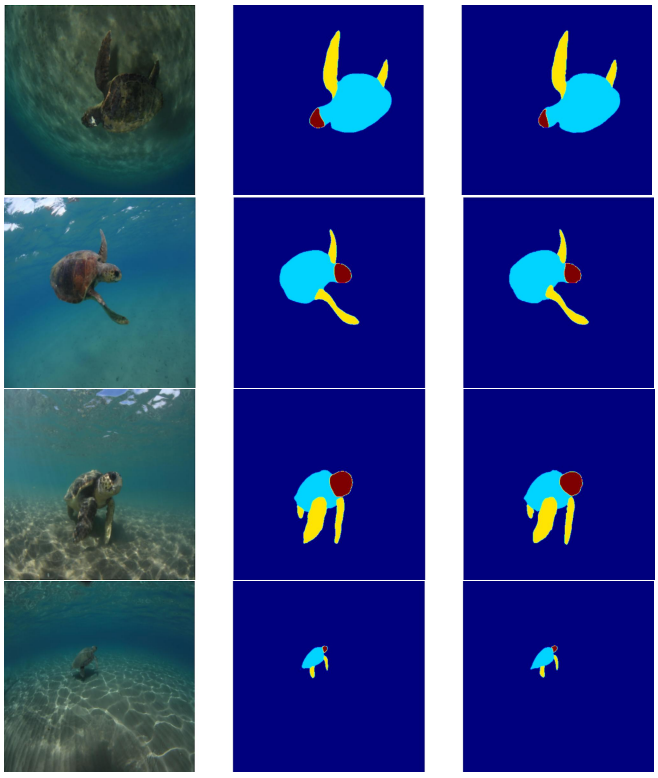


Fig. 11. Some successful visualization results in FCN

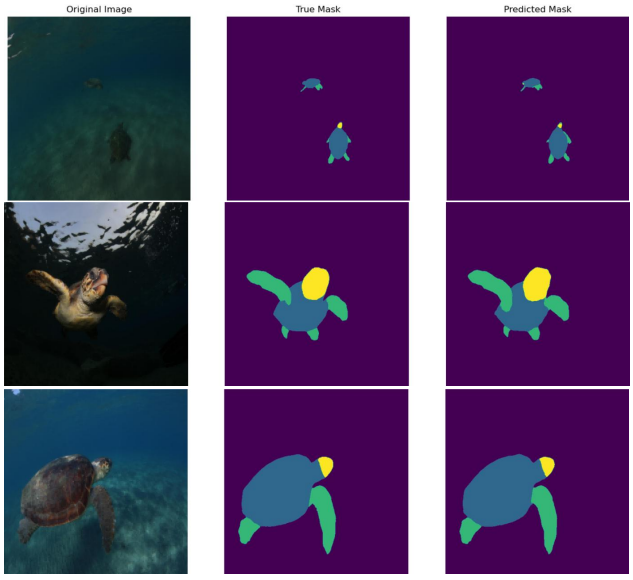



Fig. 12. Some successful visualization results in U-Net

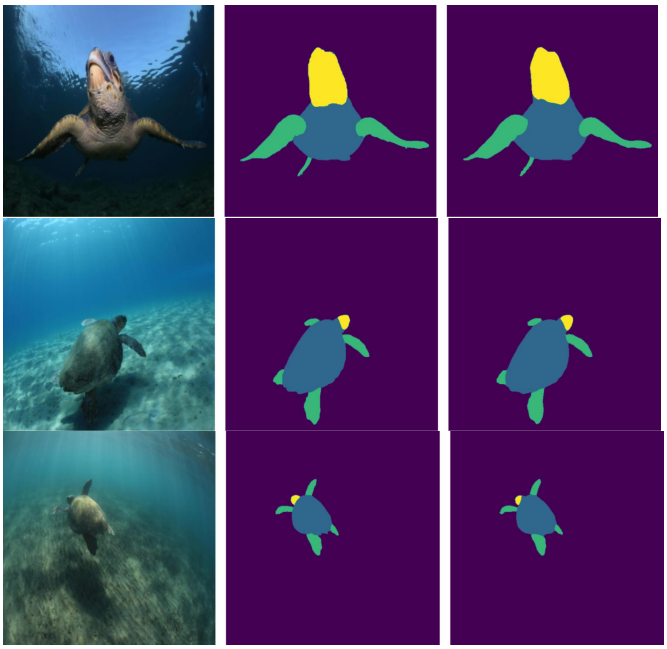


Fig. 13. Some successful visualization results in Deeopabv3

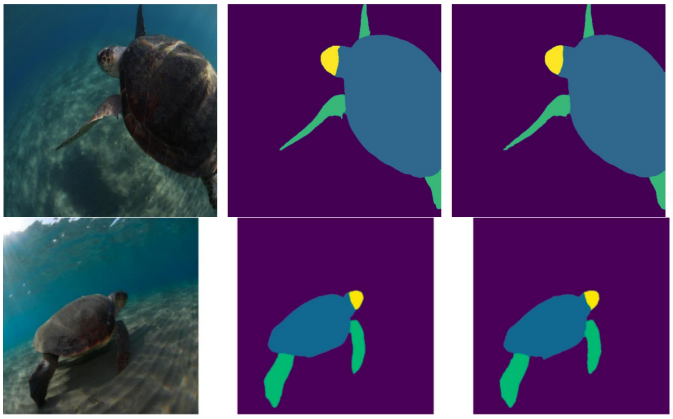
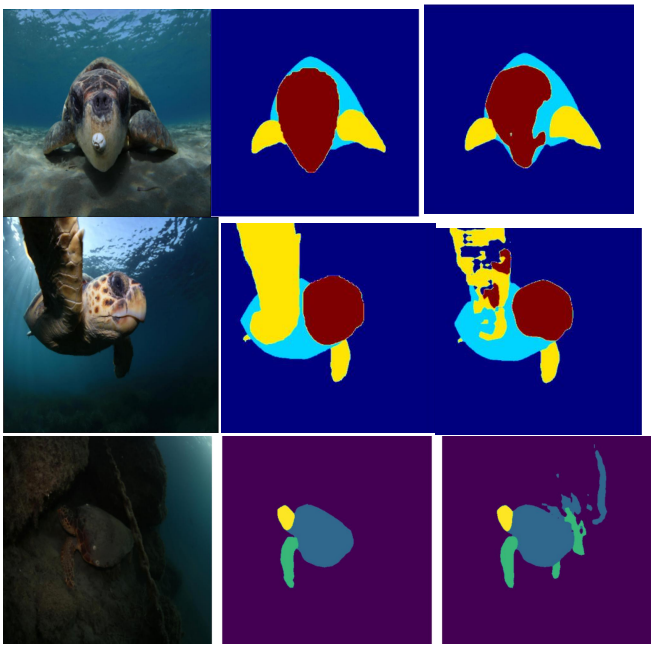


Fig. 14. Some successful visualization results in Deeopabv3+

### D. Failed examples



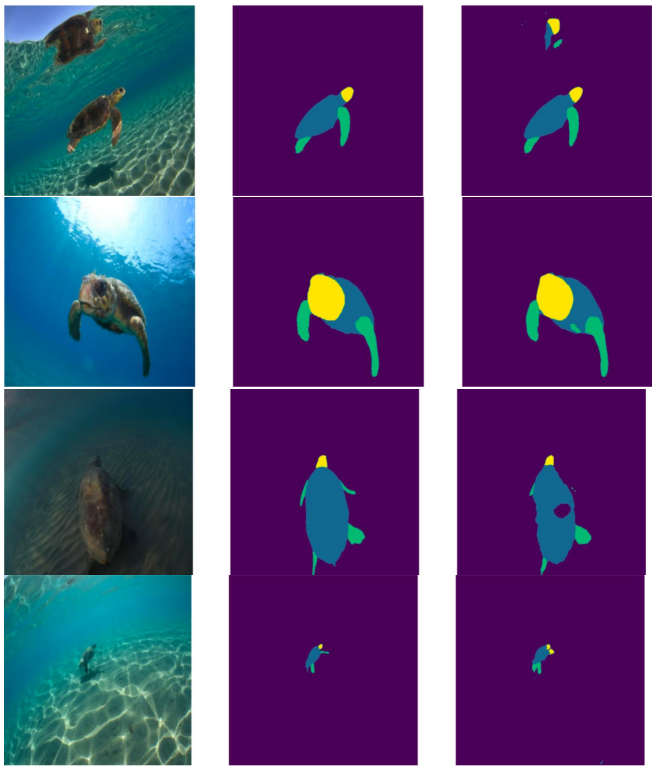


Fig. 15. Some failed visualization results on all models

## V. DISCUSSION

### A. Model Performance

The dataset processing and model setup were performed as in the previous method.

Then during model training, we trained 25 epochs.Finally, we compared the mIoU of the final test dataset of the four models.As shown in Table 1, FCN, U-net and Deeplabv3 all performed well, especially Deeplabv3, which achieved 86% mIoU.

*1) **FCN:** During training, the Training Loss decreased rapidly with the number of epochs, dropping from 0.1081 to 0.0161 within the first 10 epochs, indicating effective convergence on the training dataset. After the 11th epoch, the Training Loss remained relatively stable, but there is a rise at the 20th epoch, and finally reaches 0.0066 at the 25th epoch. In contrast, the Validation Average mIoU exhibited a two-phase pattern: a rapid increase during the early stages (from 0.7591 to 0.8057 in the first 10 epochs) followed by a stabilization phase (from the 11th to the 25th epoch) followed by a drop to 0.7730 at the 20th epoch. This fluctuations may be attributed to over fitting or parameter instability. The final testing result of 0.8543 aligns closely with the validation results 0.8614, indicating good generalization on the test dataset. Overall, the model demonstrates effective convergence during training but still has room for improvement in validation performance, such as by optimizing hyperparameters or enhancing data augmentation strategies.*

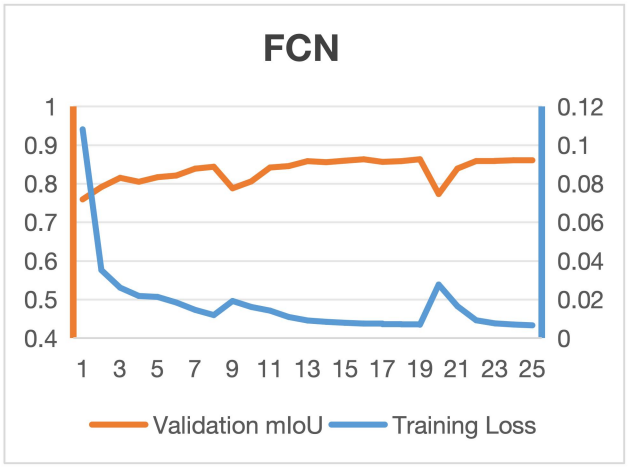
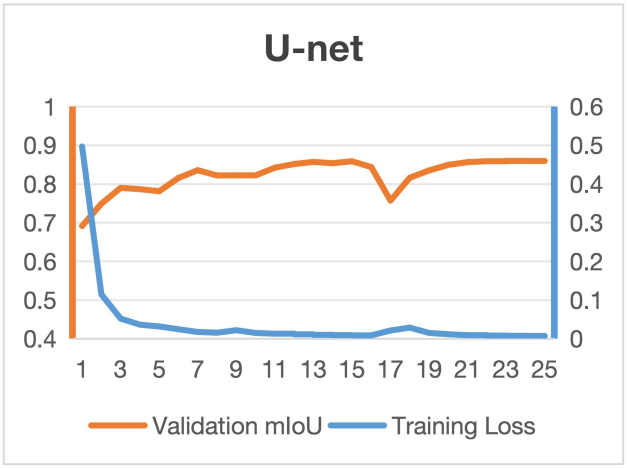



Fig. 1. The line chart of mIoU and loss in FCN

*2) **U-net:** From the result, we can see both deeplabv3 and U-Net achieved a mean IoU of 86%, indicating their high performance against segmentation tasks. When we take a closer look during the training-validation phase IoUs below, we can see the curve looks smooth except at epoch 17 which seems like a large jump. Other than that, the entire curve looks smooth, and the IoU curve converges in early epochs. The overall smooth curve indicates the initilised learning rate is selected well and highlights the effectiveness of U-Net.*




*3) **Deeplabv3:** As can be seen from the line graph, deeplabv3's training process is very good, with a decreasing loss and a very low initial loss, preferably close to 0. The initial validation miou is very high, reaching 0.7+, and improves very quickly in the first few epochs, with only one big swing. Overall, deeplabv3 performs well.*

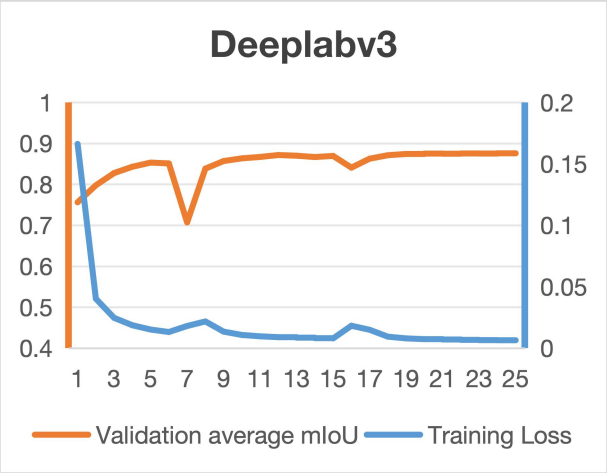



Fig. 16. The line chart of mIoU and loss in deeplabv3

*4) **Deeplabv3+:** From the line graph, we can know that the training process of deeplabv3+ is good, but not as good as the previous three models, the initial l training loss is higher, the model convergence is slower, and it is verified that the improvement of miou is not very significant, and there is a tendency to fluctuate back and forth, so deeplabv3+ is not the best for this task.*

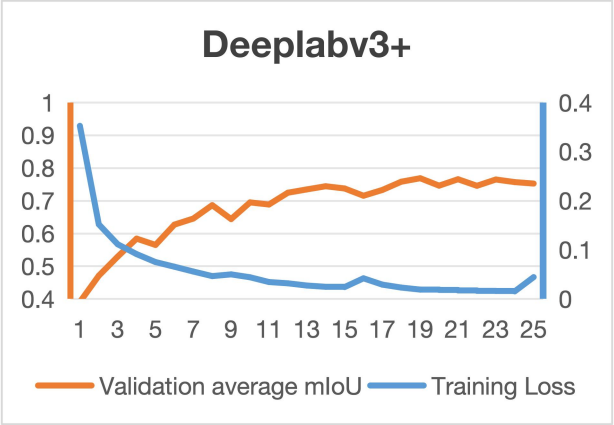



Fig. 17. The line chart of mIoU and loss in deeplabv3+

## B. Problems of Models

For the current successful cases, each model has met the requirements and segmented the three parts of the turtle. However, these models are not perfect and there are still some failure cases. As shown in some failed visualization results, some images are not segmented correctly.

When I processed the mask information of the dataset, I also found that a few mask information was not captured correctly, so it led to some images when the model was trained, and did not learn the correct information.

In this regard, we analyzed the reasons and found that many semantic segmentation models are prone to errors when dealing with object boundaries, especially when the boundaries are fuzzy or complex [3]. Even if the deep learning model has a good segmentation effect inside the object, errors may occur at the edges. The semantic segmentation model may not be able to generalize well in different environments or fields. For example, under different weather and lighting conditions, the performance of the model may decline, resulting in incorrect segmentation [4].

## C. Optimization Deeplabv3+

However, the slightly more complex model Deeplabv3+ did not perform as well as Deeplabv3. We reviewed the data to find out the reason and found that complex models would produce poor results when processing simple tasks. This is because complex models have more parameters that need to be tuned. If the same parameters as before are used, it may not necessarily bring good results, such as learning rate, batchsize, etc., to meet the learning needs of complex models. Therefore, we tuned some parameters of Deeplabv3+, using the following three steps for optimization.

*1) **Image augmentation:** Functions such as HorizontalFlip, Rotate, and ColorJitter are added by changing the transformation function of the training dataset.*

```
# optim transform method for image and mask
transform = A.Compose([
        A.Resize(512, 512),
        A.HorizontalFlip(p=0.5),
        A.Rotate(limit=10, p=0.5),
        A.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1, p=0.5),
        A.Normalize(mean=(0.0, 0.0, 0.0), std=(1.0, 1.0, 1.0)),
        ToTensorV2()
    ], additional_targets={'mask': 'mask'})
```

Fig. 18. The code about image augmentation

*2) **Changing Learing rate:*** Increase the learning rate from 0.0001 to 0.0002.

```
# Define the optimizer (Adam optimizer, learning rate 0.0001 -> 0.0002)
optimizer = optim.Adam(model.parameters(), lr=0.0002)
```

Fig. 19. The code about changing learning rate

*3) **Chaging encoder:** Change the deeper encoder from resnet50 to resnet101 which contains more residual blocks.*

```
model = smp.DeepLabV3Plus(
    # encoder_name="resnet50"
    encoder_name="resnet101",        # use resnet101 to be encoder
```

Fig. 20. The code about changing encoder

## D. Effects of Optimization

TABLE II. AVERAGE MIOU AT DIFFERENT STAGES IN THE DEEPLABV3+ MODEL

| Model name | **First valiation average mIoU** | **Fifth valiation average mIoU** | **Final test average mIoU** |
|---|---|---|---|
| Deeplabv3+ | 39% | 56% | 74% |

TABLE III. AFTER EACH STEP OF THE OPTIMIZATION METHOD, AVERAGE MIOU AT DIFFERENT STAGES IN THE DEEPLABV3+ MODEL.

| **Optim method** | **First valiation average mIoU** | **Fifth valiation average mIoU** | **Final test average mIoU** |
|---|---|---|---|
| Step1. Image augmentation | 34% | 59% | 75% |
| Step2. Learning rate change to 0.0002 | 41% | 58% | 79% |

| | | | |
|---|---|---|---|
| Step3. resnet50 change to resnet101 | 40% | 60% | 79% |

Specifically, we chose the image augmentation method [2], but we only enhanced the images of the training set, and the validation set and test set remained unchanged, in order to allow the model to adapt to different lighting conditions, different perspectives, and changes in different directions during training.

Then we chose to change the learning rate, because we found that the original Deeplabv3+ model converged slowly, so we increased the learning rate to increase the training speed [2].

Finally, we changed the deeper decoder to include more residual blocks, which can learn richer and more complex features [2], thereby improving the final test mIoU. According to Table 2 and Table 3, we can see that our final experimental results have achieved that through parameter tuning, the final performance of the Deeplabv3+ model is just better than the original one.

### E. Weighted Voting Ensemble

TABLE IV. THE FINAL TEST DATA SET MIOU OF ALL MODELS

| Model name | 'carapace' mIoU | 'flipper' mIoU | 'head' mIoU | Average mIoU |
|---|---|---|---|---|
| FCN | 92% | 84% | 80% | 85% |
| U-net | 92% | 84% | 81% | 86% |
| Deeplabv3 | 92% | 85% | 82% | 86% |
| Deeplabv3+_optim | 89% | 75% | 74% | 79% |
| Weighted Voting | 93% | 86% | 82% | 87% |

From the results, we can see the weighted voting ensemble model achieved the highest mIoU of 87% between all the models, this suggest that the ensemble model is successfully leveraging the strengths while mitigating weaknesses of each model, adapting complementary strength of the 4 models and leading to a higher performance. Furthermore, the robustness is also improved as it reduces the errors of single model by diversifying the risk.

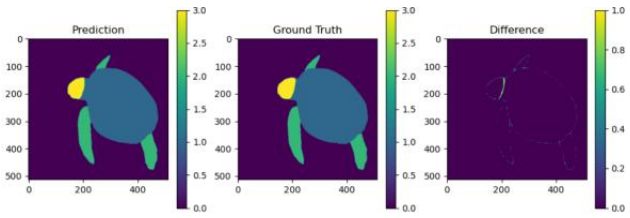



Fig. 21. Ensemble model sample

The figure above reinforces the point that diversification reduces risk of making large errors unless more than 1 high weighted model makes the same mistake. Instead, the errors only focus on the edges of body parts since that is the region where these models make tiny different judgments.

## VI. CONCLUSION

### A. Compare models

Among the four models, we compared the loss reduction rate of each model through a line graph, and compared the convergence speed of each model through each verification miou, and finally compared the miou of the test set.

FCN has a strong advantage for this segmentation task mainly because of its simple structure and effective design, which can extract more detailed information through skip connections. Moreover, the up-sampling method of FCN combines shallow boundary features and deep semantic information, which makes it more accurate in fine segmentation and edge precision, especially when dealing with sea turtle images in complex backgrounds. U-net and FCN have a similar structure and also have the above functional modules

Deeplabv3 also has a simple structure, using Atrous Convolution and ASPP cores to handle details for training, and the final task performance is also very good.

However, for the slightly more complex Deeplabv3+, the task performance is not so good. Although the core parts of deeplabv3+ and deeplabv3 are roughly the same, it adds a decoder module. The increase in parameters also increases its learning requirements, so the original parameters cannot meet its learning requirements, so the final result will be relatively poor.

### B. Innovative adjustment

We tuned the parameters of the deeplabv3+ model.

First, we modified the transformation function to achieve image augmentation. Then, because we know from the line graph that deeplabv3+ converges slowly, I increased the learning rate.

Finally, I also chose to replace the deeper decoder resnet101. We found that image augmentation did not achieve the expected effect, although it improved the final effect. However, increasing the learning rate made the model converge faster, and miou reached 58% in the fifth verification, an increase of 2%. After the third step of tuning, the test set miou increased by 5%.

Another attempt we have made is using the weighted ensemble model upon the 4 base models we have and assigned weights by examining the performance. From the results, we can see that the weighted ensemble achieved the best performance which indicates its capabilities of adapting the strengths while mitigating weaknesses of each model. Typically, it achieved a higher IoU for all three body parts, which highlighted its ability to reduce the risk of single model making errors. Overall, our weighted voting ensemble model achieves very good performance with 87% MioU by minimizing large scale errors and reduces misclassifications to minor boundary-level inaccuracies.

### C. Advice for future

In the future, if we can further optimize the Deeplabv3+ model, we can improve the decoder and try to introduce more advanced decoder designs, such as U-Net-style skip connections or dual-branch structures, to better

integrate detailed information in the encoder. This design can enhance the model's ability to learn boundaries and details.

We can also introduce multi-scale feature fusion modules, such as Feature Pyramid Network (FPN) or Attention Pyramid Network (APN), to achieve more effective information fusion between features at different scales.

About adaptive loss functions, the conventional cross-entropy loss function may be insufficient for segmentation tasks with clear boundaries. We can try to introduce boundary-aware loss functions or use multiple loss functions in combination to improve the segmentation accuracy of the model in detail and boundary areas. Especially when segmenting small objects or targets with complex boundaries, an adaptive loss function may significantly improve the performance of the model.

## REFERENCES


[1] Chen, L. C. (2017). Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587.

[2] 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).

[3] Kirillov, A., Wu, Y., He, K., & Girshick, R. (2020). Pointrend: Image segmentation as rendering. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 9799-9808).

[4] Mottaghi, R., Chen, X., Liu, X., Cho, N. G., Lee, S. W., Fidler, S., ... & Yuille, A. (2014). The role of context for object detection and semantic segmentation in the wild. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 891-898).

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

[6] Kingma, D. P. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[7] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. International journal of computer vision, 88, 303-338.

[8] Xie, E., Yao, X., & Zhou, X. (2018). "DeepLabV3+: Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

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

[10] Chen, Q., Xu, J., & Koltun, V. (2017). Fast image processing with fully-convolutional networks. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2516–2525.

[11] Lu, S., Ding, Y., Liu, M., Yin, Z., Yin, L., & Zheng, W. (2023). Multiscale Feature Extraction and Fusion of Image and Text in VQA. International Journal of Computational Intelligence Systems, 16.

[12] Wu, X., Sun, C., Zou, T., Li, L., Wang, L., & Liu, H. (2020). SVM-based image partitioning for vision recognition of AGV guide paths under complex illumination conditions. Robotics and Computer-Integrated Manufacturing, 61, 101856.

[13] Ronneberger, O., Fischer, P., Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., Wells, W., Frangi, A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science(), vol 9351. Springer, Cham. https://doi.org/10.1007/978-3-319-24574-4_28

[14] Prabakaran, J., Selvaraj, P. (2023). Implementation of ResNet-50 with the Skip Connection Principle in Transfer Learning Models for Lung Disease Prediction. In: Reddy, V.S., Prasad, V.K., Wang, J., Rao Dasari, N.M. (eds) Intelligent Systems and Sustainable Computing. ICISSC 2022. Smart Innovation, Systems and Technologies, vol 363. Springer, Singapore. https://doi.org/10.1007/978-981-99-4717-1_2