Methods in Segmenting Sea Turtle Body Parts

Sarvesh Wanzare
z5327562

Jason Mai
z5623602

Nykolas Rekasius
z5624841

Pramila Jangid
z5494002

Jason Lim
z5359171

I. Abstract

For this project we used the SeaTurtleID2022 dataset to create semantic, multiclass segmentations of turtle images. We experimented with multiple deep learning frameworks for turtle segmentation.

Our dataset comprised 8,729 underwater photographs of sea turtles, split into 5,303 training images (60.75%), 1,118 validation images (12.81%), and 2,308 test images (26.44%). Our implementation handled unique challenges in underwater imagery, including light attenuation, color distortion, and variable visibility conditions that particularly affect segmentation tasks.

First, we set up a multiclass U-Net to establish a baseline segmentation. We then explored background removal as a preprocessing method to see if it would help the model learn turtle features. We used the true masks to remove the backgrounds from the turtle images in order to establish a baseline segmentation for images with blacked out backgrounds. We then created a pipeline with one U-Net that removed the background and a second U-Net that segmented the turtle.

In addition to U-Net based approached, we also experimented with a Mask R-CNN framework. The method involved training separate models on each of the segmentation classes. Each model was then used to predict a segmentation for its respective class. The predicted segmentations were then combined to create a multiclass segmentation for each turtle.

Our baseline multiclass U-Net performed the best, achieving a mean intersection over union of 0.8600. The true mask removed background outperformed the U-Net removed background, achieving mean intersections over union of 0.8080 and 0.7348, respectively.

Under the separate instance segmentation MRCNN approach, a mean intersection over union of 0.7161 was achieved. The segmentation of the turtle body yielded the highest intersection of union at 0.86, followed by the flippers at 0.7249 and the head at 0.5635.

II. Introduction

Turtle populations are declining, with 52% of species listed as threatened and 20% of species listed as critically endangered [1]. Monitoring turtle populations is an important aspect of conservation. Computer vision techniques can be used to automate the monitoring of turtles. Specifically, identification and segmentation techniques can be used to determine populations and assess turtle health.

Underwater image segmentation presents unique technical challenges that significantly impact model performance. These challenges include light attenuation and scattering effects, which cause color distortion and reduced contrast. Variable visibility conditions, refraction effects, and suspended particles further complicate the segmentation task. The accurate segmentation of turtle body parts is crucial for health assessment, as it enables automated monitoring of physical conditions like shell damage, flipper injuries, and overall body condition. These indicators are vital for conservation efforts and population health tracking.

The SeaTurtleID2022 dataset includes images of sea turtles and associated annotations including segmentations that identify the body, head, and flippers of each turtle. Our project focuses on developing novel methods of segmenting specific body parts of sea turtles from a large dataset. The dataset is structured following the Common Objects in Context (COCO) format, and includes annotations in JSON format.

This paper presents a plethora of methodologies that are developed by utilising pre-existing models like U-Net and Masked R-CNN and modifying them slightly to develop our own methodologies like a multi-stage segmentation using U-Net, background removal U-Net pipelined with instance segmentation U-Net, and an independent segmentation using Mask R-CNN. The results of these models is evaluated quantitatively using the Jaccard Score (also known as the Mean Intersection Over Union (mIoU) score). Shortcomings and findings from these models are later discussed.

III. Literature Review

A. Recent Advances in Underwater Image Segmentation

Recent developments in transformer-based architectures have significantly advanced the field of underwater image segmentation. Pavithra and Denny (2024) [2] introduced a hybrid architecture combining Swin Transformer with ConvMixer in a SwinUNet Architecture, which demonstrated superior performance in challenging underwater conditions. Their approach effectively addressed the persistent challenges of underwater imagery, particularly in low-visibility environments, outperforming traditional models such as YOLO v8 and Pix2Pix GAN on the Semantic Segmentation for Underwater Imagery (SUIM) dataset.

The challenge of marine animal segmentation has been further addressed by Kim and Park (2024) [3] through their innovative Parallel Semantic Segmentation Network (PSS-net). Their work specifically tackles two major challenges in underwater segmentation: low-light conditions and the segmentation of marine animals with protective coloration. By employing parallel processing for foreground and background segmentation, enhanced with attention mechanisms, their model achieved impressive results with a mean intersection over union of 87% and structure similarities of 97.3%.

B. Marine Life Segmentation Challenge

Underwater image segmentation presents unique technical challenges that significantly impact model performance. These challenges, as highlighted in recent literature [2], include, light attenuation and scattering effects, variable visibility conditions, low-light environments,

protective coloration of marine animals, complex underwater backgrounds, [3] specifically addressed these challenges through their parallel processing approach, demonstrating that separate handling of foreground and background elements can significantly improve segmentation accuracy in challenging underwater conditions.

C. Consideration of ML approaches

There are two main methods of image segmentation - machine learning and deep learning. Machine learning approaches tend to be less computationally intensive than deep learning approaches, but also less accurate.

We considered various machine learning based methods for image segmentation. We researched HOG for shape detection, thresholding for color and intensity detection, and SIFT for feature detection. We also researched various classifiers including: SVM, decision trees, random trees, and lightGBM.

However, we decided to use deep learning. A crucial advantage to deep learning is that it does not require expert knowledge on the subject matter of the images. While machine learning requires knowledge of turtles to know what kinds of features to extract, deep learning uses generic convolutions to extract information from images.

D. Previous work in Mask R-CNN model for image segmentation

Mask R-CNN is a versatile instance segmentation framework derived from Faster R-CNN [1]. The Mask R-CNN sits on top of a backbone whose primary function is related to extracting features from different scales from images and building a feature map out of them. There are multiple algorithms for backbones available for feature extraction from multiple images. The most popular ones are VGG [2], Inception [3] or ResNet [4]; the backbone network usually adopts a deep convolutional network [5]. The extracted features from the ResNet backbone are then coupled with a Region Proposal Network (RPN) which consists of regions that likely have the object in question and fed into the ROI Align layer that aligns the features extracted with the ResNet backbone with the proposed regions in the RPN network. The Mask Head branch then generates the segmentation masks for each region proposal using the output from the ROI Align layer to successfully predict the instance segmentation mask [5].

Due to its versatility and high accuracy, Mask R-CNN has been used in a number of applications ranging from plant species detection [6] to underwater creature detection [7]. In [7], the proposed solution was to integrate MSRCR with Mask R-CNN to recognise underwater creatures. MSRCR was used to preprocess the image data since underwater imaging generally consists of noise, colour attenuation, bluish-green color tones, and bright spots [7]. To combat this problem, the authors in [7] used MSRCR in order to enhance the picture quality. The Mask R-CNN model was pre-trained on COCO dataset and transfer learning approach was used to train the model on the underwater dataset. The results showed a mIoU of 94.84% and a precision of 97.46%. The proposed result performed 10% better than the mask R-CNN method, 15% better than YOLOv3, and 24% better than SSD in terms of the mIoU of the algorithms. This result, although impressive, was likely due to the subset size being very small compared to the SeaTurtleID2022 dataset that is being used in this paper; this likely introduced a bias which is why the results were so high. This paper shed light on the significance of image preprocessing and the high accuracy of Mask R-CNN, albeit the dataset was very small.

The authors in [6] used a ISC-MRCNN approach to segment and classify plant leaf images. The images in the dataset were pre-processed using the ISC algorithm to obtain the foreground of plant leaf images and then uses MRCNN to fuse the feature maps of varying depths in order to train the network on more detailed features [6]. The experimental results showed that the average precision of the ISC-MRCNN compared to traditional Masked R-CNN was 1.89% more under different thresholds [6]. The authors also proposed another method in which a Support Vector Machine (SVM) was used to replace softmax and then an Adaptive Chaotic Particle Swarm Algorithm is employed to optimise detection performance of overlapping objects. Results indicated a 1.59% improvement over traditional Mask R-CNN methods when the mIoUs of the two algorithms were compared.

E. Previous work in U-Net model for image segmentation

When researching deep learning methods we considered various frameworks. We opted to use U-Net, as it offers accurate results while being significantly faster than other frameworks. The U-Net model works in two stages: downsampling and upsampling. The encoder extracts increasingly complex information as the resolution of the image decreases. The decoder then restores spatial information lost in downsampling using information passed from the same resolution in the encoding phase using skip connections.

Further research was conducted to find a novel approach that could increase U-Net accuracy on our data set. We were intrigued by [8] on preprocessing breast cancer images for deep convolutional neural networks. They referenced [9] that used a different background removal technique and improved their accuracy from 95.42% to 98.34% compared to segmentation without preprocessing.

[8] lined up with our interest in background removal to preprocess images for segmenting using deep learning. However, we determined novel ways to apply the approach. First, the paper outlines a preprocessing method but leaves the application to a deep CNN as an area of further study. Second, the approach has not yet been tested on animals including turtles. Third, the proposed application does not use a U-Net based deep CNN architecture. Fourth, the approach uses a combination of rolling ball, Huang's fuzzy, and morphological transformations for background removal rather than a separate neural network.

This final distinction is particularly important because it ensures a system that is flexible across domains. Traditional preprocessing techniques are specific to the given data and often require subject matter knowledge. By using two deep learning models, both the preprocessing (background removal) stage and the segmentation stage can effectively be trained using labeled data regardless of domain.

IV. METHODS

A. Data Preparation

The COCO format and API provide several key tools and functions useful for loading, viewing, and analyzing annotations and images. The COCO dataset contains categories for the annotations that map category IDs to parts

of the turtle for segmentation with category_id values 1 for "turtle," 2 for "flippers," and 3 for "head."

With coco.loadImgs(image_id) and coco.getAnnIds, we retrieve and visualise specific images alongside their segmentation masks. The coco.annToMask() function enables us to generate binary masks for each segmentation category, which then allows us to view parts of the turtle in isolation. For example, pixels representing the turtle body are assigned to one mask, while the flippers and head are assigned to others.

For a given image, each of its binary masks are scaled by the category_id value to assign pixel values for each class—1 for the turtle body, 2 for the flippers, and 3 for the head—and combined with precedence, overlaying the turtle, then flippers, then head giving us the entire segmentation mask without overlapping values. Pixels in areas that do not overlap with any object annotations are automatically set to 0, marking them as background. This preprocessing step allows us to create a dataset of segmentation masks where each pixel is assigned to a specific class.

The dataset is divided into training, validation, and test sets using the provided metadata_splits.csv file. This file's 'split_open' column specifies which subset each image belongs to, ensuring an open-set split as done in the original paper. This preprocessing is essential for preparing our training, testing, and validation data for our deep learning models.

This process for mask preparation and data partitioning is used for all of our models including for multiclass segmentation, as well as for background removal in some of our methods.

B. *Multiclass U-Net Model*

We began by training and testing a simple U-Net model on the provided dataset of images and masks. To aid in stable training, reduce internal covariate shift, accelerate training and prevent overfitting, we added Batch Normalisation layers in the U-Net [11]. Additionally, we applied dropout layers with a rate of 0.2 to further prevent overfitting by randomly deactivating a subset of neurons during each training iteration [12]. We used cross-entropy loss and Adam optimiser as they are commonly used for CNN training [13]. Our cross-entropy loss was also weighted by class to address biases such as the background often having significantly more pixels than smaller classes like the flippers. We set the learning rate as 1e-4 set to reduce on plateau by a factor of 0.5 and a patience of 3. Lastly, we employed early stopping with a patience of 7 to prevent overtraining, stopping the training process once the validation loss stopped improving for several epochs [14]

The U-Net was trained using masks that included 4 classes: background, body, fin, and head. The model used multiclass segmentation to predict a segmentation for all four classes at together. All U-Net methods resized input images to 128x128 resolution. This first model serves as a baseline for subsequent approaches.

C. *Background Removal Using True Masks → U-Net*

For the second model, we created a dataset where backgrounds were removed, replaced with black pixels, using the true mask labels from the annotations. Training a U-Net on these modified images and masks allowed us to study the impact of clean background removal on segmentation accuracy. This approach is motivated by the idea that a simplified background can help the model focus more directly on the turtle itself, potentially improving its ability to recognise and differentiate turtle parts. This method simulates a pre-processed dataset with perfect background removals, offering insights on performance under controlled conditions.

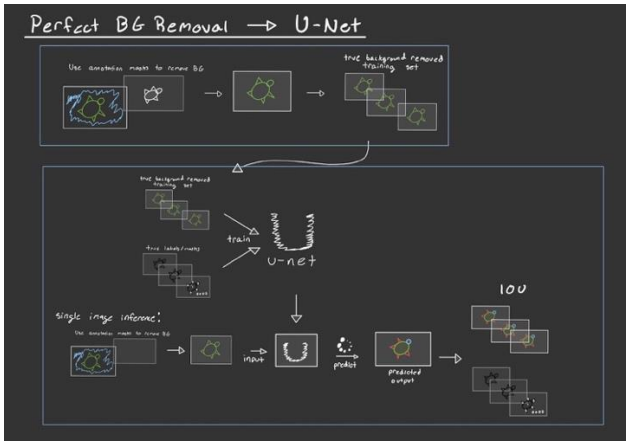

Figure 1: Background Removal Using True Masks → U-Net

D. *Multi-Stage Segmentation - Background Removal Using U-Net → U-Net*

In the real world, having access to precise background masks for background removal is not always feasible. Therefore, to build upon our approach, we developed a multistage pipeline where a U-Net model was first trained to perform background removal, learning to identify and mask the background from the available true masks from the annotations. This first stage model is then applied to remove the backgrounds from the test images to simulate a realistic scenario where true background labels are unavailable. The output of these test images then become our input training images for a second U-Net alongside the true segmentation masks to train the model to perform multiclass segmentation on turtle images with their backgrounds imperfectly removed by the first stage U-Net.

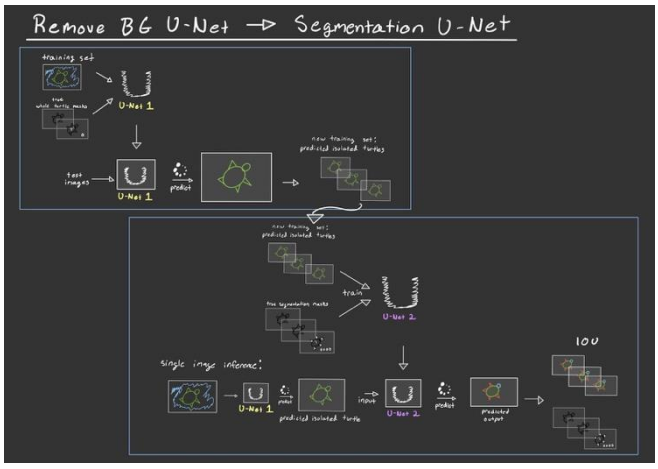

Figure 2: Mutli-Stage Segmentation

E. *Separate Mask R-CNN segmentation*

This paper aimed to explore the feasibility of implementing a traditional Masked R-CNN method on multiple classes separately. This method is implemented by treating all three of the classes of the turtle (head, flippers, and carapace) as separate classes and then using the masked R-CNN algorithm on said classes. The hyperparameters for each of the three models were determined after carefully considering the trade-off between the accuracy and the time

taken to complete the model. Since the proposed method was computationally going to take 3x as long as a traditional method, it was determined that obtaining an accurate model was more important than the training time. Therefore, The hyperparameters for the three segmentation models are:

Epochs: 100

Minimum confidence: 0.8

Image dimensions: 512x512

Learning rate: 0.00015

Once the models were trained on their respective classes, the models were tested on an image in which they predicted the masks of the three body parts. With the mask predictions obtained, the final prediction was made on the image by superimposing all the masks predicted by the respective models on the final image.

V. Results

A. *Multiclass U-Net Model*

The first method involved training a multiclass U-Net on the complete training set without any additional preprocessing. Training lasted 33 epochs over 421 minutes. The best validation loss was at epoch 30 with a value of 0.1439. The training loss reached 0.0703. The IoU scores were as follows:

- Background: 0.9892
- Turtle Body: 0.8698
- Flippers: 0.7344
- Head: 0.8464
- Mean IoU: 0.8600

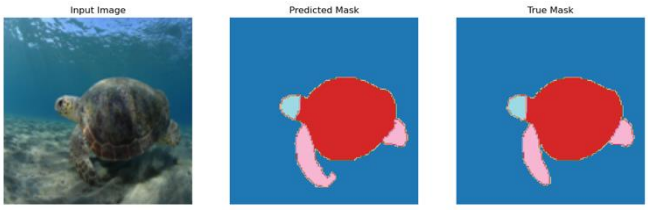

Figure 33: U-Net on Image aAFboMSSvO

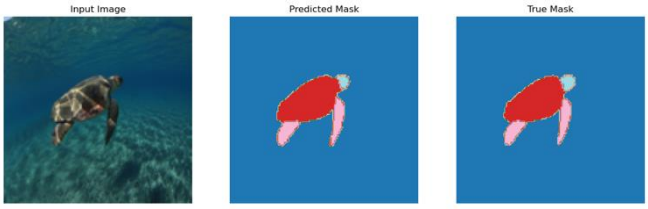

Figure 44: U-Net on Image BBrpGkYbDR

B. *Background Removal Using True Masks → U-Net*

For the second method, the background was removed by setting it to black pixels based on the provided segmentation masks before training. Training lasted 16 epochs over 212 minutes. The best validation loss was at epoch 10 with a value of 0.1713. The training loss reached 0.1310. The IoU scores were as follows:

- Background IoU: 0.9972
- Turtle Body IoU: 0.8118
- Flippers IoU: 0.7414
- Head IoU: 0.6815
- Mean IoU: 0.8080

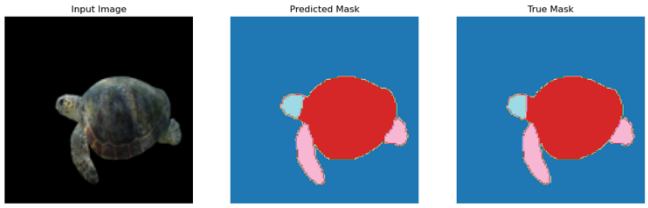

Figure 55: Perfect BG Removal U-Net on Image aAFboMSSvO

Figure 66: Perfect BG Removal U-Net on Image BBrpGkYbDR

C. *Multi-Stage Segmentation - Background Removal Using U-Net → U-Net*

The third method utilised a multi-stage approach with a two-step U-Net process. Initially, a U-Net was trained to remove the background, simulating a real-world scenario where true background masks are unavailable. The output of this background removal model was then used to further train a second U-Net on body part segmentation.

Whole Turtle U-Net: This model was trained to recognize the whole turtle. Training lasted 29 epochs over 329 minutes. The best validation loss was at epoch 25 with a value of 0.0347. The training loss reached 0.0257. The IoU scores were as follows:

- Background: 0.9860
- Turtle Body: 0.9113
- Mean IoU: 0.9487

Figure 77: Predicted Background Removal using U-Net on image AAFBOMSSVO

Figure 88: Predicted Background Removal using U-Net on image BBrpGkYbDR

Turtle Part Segmentation U-Net: Using the background-removed images generated by the initial model, this model focused on segmenting the turtle, head and flippers. Training lasted 19 epochs over 54 minutes. The best validation loss

was at epoch 12 with a value of 0.3065. The training loss reached 0.2540. The IoU scores were as follows:

- Background IoU: 0.9817
- Turtle Body IoU: 0.7423
- Flippers IoU: 0.5594
- Head IoU: 0.6558
- Mean IoU: 0.7348

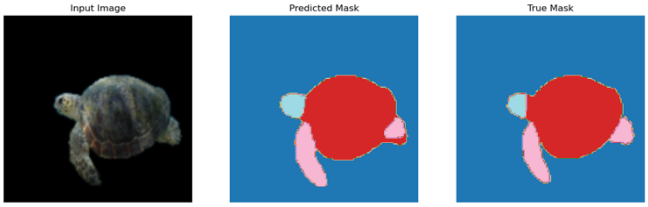

Figure 99: Multi-Stage Segmentation U-Net on Image aAFboMSSvO

Figure 1010: Multi-Stage Segmentation U-Net on Image BBrpGkYbDR

D. Independent Instance Mask R-CNN segmentation

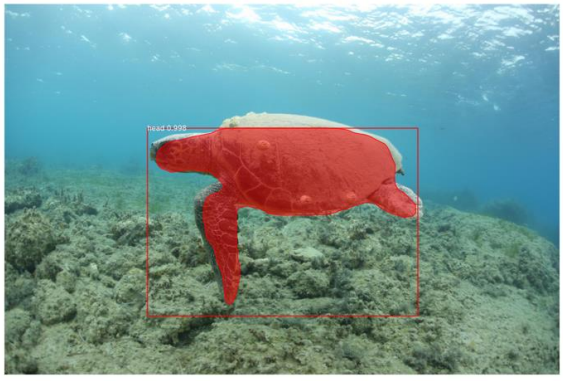

Figure 11: Prediction Result of the Mask-RCNN on Turtle Body

The Mask R-CNN segmentation is achieved through superimposing the masks for the head, turtle body, and flippers, as shown in Figure 11. However, as demonstrated in Figure 11, the segmentation filters did not yield fully accurate results due to overlapping and overfitting between the masks. The reason for the overfitting can be contributed by the imbalanced training data for the head and unique poses of the turtle.

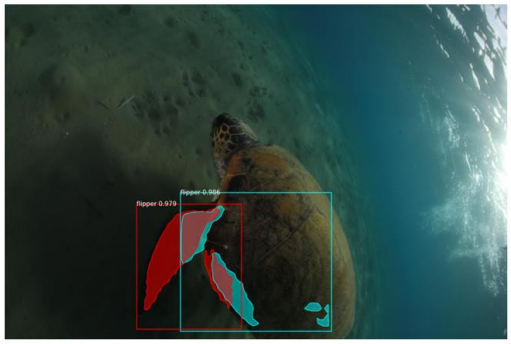

Figure 112: Incorrect segmentation of the flippers

As shown in Figure 12, the model has accurately identified the flippers with high intersection over union of 0.979 and 0.986. However, notably a small component of the turtle's body has been incorrectly segmented in blue as a flipper.

Overall, Mask R-CNN demonstrates higher performance in detecting and segmenting the turtle's body and flippers, reflected by the higher intersection over union values compared to the head. However, segmentation inaccuracies are caused by limited and imbalanced training data and potential overfitting of segmentation filters for components such as the head. As the size and visibility play a significant role in accurate segmentation.

Additional training and fine-tuning, combined with data augmentation and domain-specific pre-training, will significantly enhance the model's performance, to improve head segmentation accuracy in maintaining greater consistency across different images. Additionally, evaluating the model's segmentation capabilities using Precision and Recall, alongside IoU, will provide a more comprehensive assessment of its performance.

VI. Discussion

A. Comparing U-Net Approaches

Across the U-Net methods, the multiclass U-Net model yielded the best IoU results. Sample images (pictured: aAFboMSSvO and BBrpGkYbDR) were used to qualitatively evaluate the segmentation performance for each method, providing insight into how well the models distinguished between different turtle parts. Removing the background with true masks performed slightly worse, possibly because the model was not necessarily 'ignoring' black background pixels for each image. The extra context of background details might also be more important to the model's ability to segment the turtle.

Multiclass U-Net Model

We observe the multiclass U-Net Model messes up on Image aAFboMSSvO where the flipper might blend in with the floor in terms of the colors and shapes present in the turtle scales and shadows on various surfaces (fig. 3). In this case it seems as though the model confuses a section of the background that is similar in appearance to the turtle. Since backgrounds tend have various colors, shapes, and textures, there is a chance that certain sections can appear turtle-like to the U-Net model.

Background Removal Using True Masks → U-Net

Removing the background, the second model does not repeat the same mistake on Image aAFboMSSvO as there is no background to cause the theorized confusion. However, the lack of background context opens up new challenges. Without the context of the background, this model is now susceptible to errors like those observed in Image BBrpGkYbDR where the shapes caused by shadows on the shell might look like segments of the head, possibly explaining the misprediction (fig. 6).

Multi-Stage Segmentation - Background Removal Using U-Net → U-Net

The multi-stage segmentation approach using a U-Net for background removal followed by U-Net for segmenting the turtle parts demonstrated the lowest performance amongst the U-Net approaches. This likely stems from the compounded inaccuracies introduced by using two model predictions.

In fig 10 the model predicted flippers that were larger than in the actual mask. This could be explained by an imperfect background removal leaving excess background around the fins of the turtle. As a result, the second U-Net may consider the false positive background pixels as fin pixels, since they are more similar to the adjacent fins rather than the black removed background.

Another example of where this model failed is when the first-stage U-Net for background removal mislabeled a body part as background, essentially blacking that pixel out. When it came time for the second-stage U-Net to train using the true masks, it was incorrectly told that a black pixel that was supposed to be a body part was a body part.

As the above errors represent, the second U-Net was trained on images that included both blacked out turtle pixels and non-blacked out background pixels. As such, the model resulted in "hallucinations" mislabelling certain turtle pixels as background and vice versa.

We considered a pipeline in which the second U-Net was trained on images with the background removed using the true masks. However, the model would be very rigid, as it would learn that only blacked out pixels could be background pixels. As a result, it would likely mislabel any incorrectly removed pixels from our first U-Net. We decided to stick with our approach, as despite the hallucinations the model was more flexible regarding imperfect background removals.

B. Independent Instance Mask R-CNN segmentation

This methodology worked very well in predicting the body and the head of the turtles, with the confidence level being around 95% on an average. This is because these body parts were not occluded and had good visibility for the most part. There was some undershoot in predicting the body of the turtles; this was likely due to the high number of epochs that the model was trained on in addition to the steps per epoch being 100 which might have resulted in some undershoot.

There are multiple images in which the masked R-CNN for the body is successfully segmenting the body from the background. One such image in which the model performed a very accurate picture is given in Figure 13 below. This result was likely due to less noise in the image, it does fail to segment the flipper on the top but that could be due to the occlusion in the flipper instance.

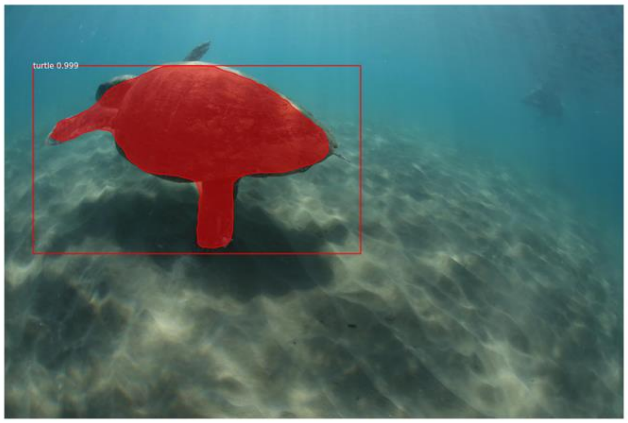

Figure 1312: Better Prediction of Body Mask R-CNN model on Turtle Body

However, the algorithm had some overfitting for segmenting the flippers of the turtles. This is evident from the poor mIoU scores of the flippers in comparison to the head and the body of the turtles. This is likely due to the multiple occlusions prevalent in the flippers, since the Mask R-CNN algorithm works by creating boundary boxes for multiple regions for feature extractions, a limitation of that is that the method is not able to create a boundary box for instances that are occluded/partially visible.

In one instance, the model correctly predicted the flipper within the red bounding box, as shown in Figure 14, but displayed higher confidence in the incorrect segmentation, as seen in the turquoise bounding box in Figure 12. This is likely due to the image being scaled down to a size of 512x512 which would have resulted in a loss of information so when this mask was applied on the original image with those dimensions, it resulted in an inaccurate mask.

Due to the same resizing dilemma, the prediction of the flippers on the turtle was considerably smaller than the actual size of the flipper in the image.

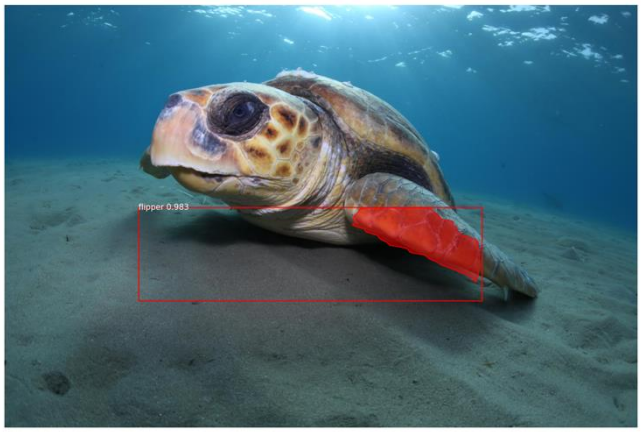

Figure 13: Small mask prediction compared to actual size

The final result of combining the masks of the three segments of the turtle resulted in a very good segmentation for an image in the test set, as shown in Figure 15. This was highly contributed to the low learning rate and high epochs and steps per epoch for our model. Additionally, the noise is the picture is very less and most parts of the turtle are very visible in this image which is also a contributing factor to the accuracy of this mask.

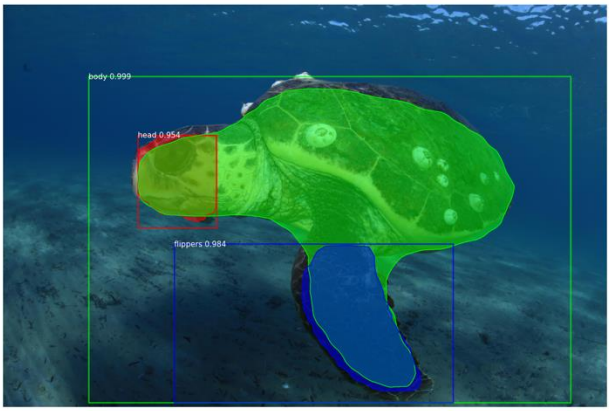

Figure 14: result of combining all the masks

Although the resultant prediction segments the different parts of the turtle, the mask prediction of the body overlaps the segmentation masks of the head and flippers. This is one major drawback that has been observed in this methodology- since each of the models only have an idea that they are the only class that has to be segmented, the model will overshoot the prediction onto other classes that may be predicted for segmentation as well.

VII. FUTURE SCOPE - DEEPLABV3

Our current implementations leverage two distinct architectural approaches: U-Net's encoder-decoder framework and Mask R-CNN's two-stage detection-segmentation pipeline. The U-Net architecture, as implemented in our study, consists of a contracting path that captures context through successive pooling operations and an expansive path that enables precise localisation through transposed convolutions. The skip connections between these paths combine high-resolution features from the contracting path with upsampled features, enabling precise segmentation of turtle features. Our implementation enhanced this architecture with batch normalization layers to reduce internal covariate shift and dropout layers (rate = 0.2) to prevent overfitting, achieving a mean IoU of 0.8600 on the multiclass segmentation task.

The Mask R-CNN implementation, following a different paradigm, utilised a ResNet backbone for feature extraction coupled with a Region Proposal Network (RPN). This two-stage approach first generates region proposals, which are then refined through ROI Align layers before mask generation. Our separate instance segmentation approach using Mask R-CNN achieved varying performance across different turtle components (body: 0.86 IoU, flippers: 0.7249 IoU, head: 0.5635 IoU), highlighting both the strengths and limitations of the instance-based approach in underwater conditions.

While both architectures demonstrated promising results, they exhibit specific limitations in handling the unique challenges of underwater turtle segmentation. U-Net's fixed receptive field can struggle with varying turtle scales, while Mask R-CNN's region-based approach sometimes leads to overlapping predictions in our separate instance implementation. DeepLabV3 [17] emerges as a compelling future direction, particularly when integrated with ResNet backbone architectures. The model's fundamental strength lies in its atrous (dilated) convolutions that systematically aggregate multi-scale contextual information without losing spatial resolution. The ResNet backbone, with its residual learning framework, enables training of substantially deeper networks while mitigating the vanishing gradient problem through identity mappings [18]. DeepLabV3's architecture incorporates parallel atrous convolutions at multiple sampling rates (6, 12, and 18) within the ASPP module, effectively capturing objects and contextual information at multiple scales. This is particularly crucial for underwater imagery where light attenuation and scattering effects can alter feature appearances at different depths. The model's cascaded arrangement of atrous convolutions creates an effective receptive field that grows exponentially with layer depth, while maintaining computational efficiency through factorized convolutions [19]. Furthermore, the integration of batch normalization in both the ASPP module and ResNet backbone facilitates faster convergence and better generalization [20]. When compared to traditional encoder-decoder architectures like U-Net, DeepLabV3's explicit handling of multi-scale features through parallel dilated convolutions offers more robust feature extraction, particularly beneficial for segmenting turtle features at various scales and orientations in underwater conditions.

A. Implementation with DeepLabV3

The implementation utilized DeepLabV3 with a ResNet-50 backbone from torchvision.models.segmentation, configured for binary segmentation of turtle flippers against the background. The model was implemented using PyTorch's deep learning framework, leveraging its neural network modules (torch.nn) for architecture definition and optimisation utilities (torch.optim) for training. Our preprocessing pipeline, built using torchvision.transforms, incorporated normalisation transforms with ImageNet statistics (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]) to standardise input features and improve training stability. Spatial consistency was maintained through bilinear interpolation of input images and nearest-neighbor interpolation of ground truth masks to 512×512 resolution, with the latter chosen specifically to preserve hard boundaries in segmentation masks. The model architecture was modified at the classifier layer, replacing the final convolution to output two channels (background and flipper classes) instead of the default DeepLabV3 configuration, implemented through a Conv2d layer with 256 input channels and kernel size of 1 for pixel-wise classification.

The training pipeline utilised PyTorch's DataLoader with pin_memory=True for optimized data transfer to GPU and a batch size of 4 to balance memory constraints with training stability. The model was optimized using Adam optimiser with a learning rate of 1e-4, chosen for its adaptive learning rate properties and momentum-based updates, particularly beneficial for the varying features in underwater imagery. Cross-Entropy loss was selected as the optimisation criterion, appropriate for binary segmentation tasks as it effectively handles class imbalance between flipper and background pixels. Training was conducted over 15 epochs on 2000 training and 2000 validation images, with model checkpointing implemented to save the best-performing model based on validation IoU.

B. Results with DeepLabV3

The model demonstrated remarkable learning progression, starting with a validation IoU of 0.6316 in epoch 1 and

steadily improving to reach a peak validation IoU of 0.8061 by the final epoch, while training IoU improved from 0.6212 to 0.9170. This consistent improvement in both training and validation metrics suggests robust feature learning without significant overfitting. The model's generalization capabilities were further validated on the test set, achieving an impressive final test IoU of 0.8492, surpassing both the training and validation performance. The implementation's strong performance with limited training data underscores the effectiveness of DeepLabV3's atrous spatial pyramid pooling module in capturing multi-scale contextual information even with constrained training resources. The stable convergence pattern, evidenced by the gradual improvement in validation IoU over 15 epochs, suggests that the chosen hyperparameters and architecture effectively captured the complex features of underwater turtle anatomy.

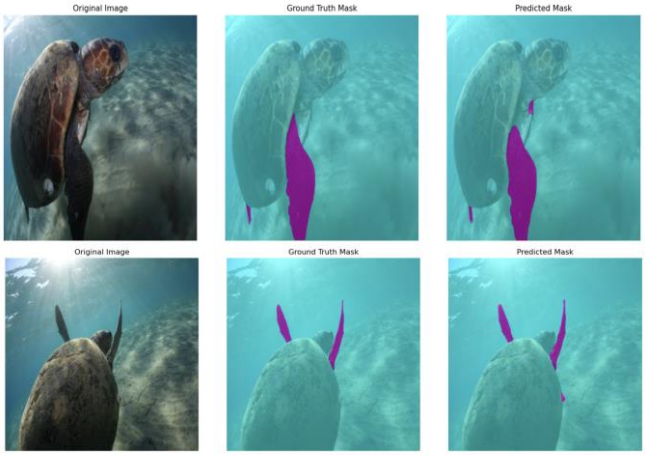

Figure 16: Segmented flippers using DeepLabV3

VIII. Conclusion

When thinking about ways to reduce background noise in order to improve turtle segmentation performance, we hypothesized that using a method like YOLO to create a bounding box that we could then hone in on would a) achieve our goal of reducing the background and b) also reduce the total image resolution allowing for easier computations. However, we discovered that YOLO models are not trained to identify turtles. The bounding boxes YOLO would give us were technically false positives for other objects, i.e. predicting a reasonable bounding box but identifying a turtle as a bird. Hence, we suspected it would be inaccurate to use these bounding boxes.

Additionally, the turtles appear at different scales in the images. As such, using bounding boxes to crop into the turtles would result in images that were differently sized. However, deep learning models such as U-Net require consistent image sizes. Adding padding to make the cropped images uniform would negate much of the benefits to cropping them in the first place.

As part of U-Net methods, we wanted to explore already well-developed and existing methods of background removal for our multi-stage segmentation model such as Segment Anything Model (SAM). In addition to being computationally expensive, we discovered that the masks that SAM returns are formatted as dictionaries containing information about each detected mask in an image. There was no way to automatically identify which mask was which, let alone pick out any single mask or combination of masks that represented the turtle or background to use for our task of separating turtle from background. The proposed method of using a U-Net for background removal worked well leaving room for exploration of the effect of compounding inaccuracy or our theorised idea of “hallucinations” leading to the poor multi-stage segmentation.

The proposed method of independent semantic segmentation using Masked R-CNN did not perform as well as a traditional Masked R-CNN. The main reason for that was that training the models independently led to predictions for the body overlapping other instances. A method was proposed during development to use the Cellpose algorithm to perform instance segmentation separately and then stitch the masks back together but that could not be done in this paper due to the complexity of the task and was deemed infeasible as the cellpose algorithm works best for classifying mainly circular objects in images.

Our exploration of DeepLabV3 revealed compelling advantages for underwater segmentation tasks, particularly in its efficient utilization of limited training data. The model's progression from initial validation IoU of 0.6316 to a final test IoU of 0.8492 demonstrates not just high accuracy, but also strong generalization capabilities. This performance was achieved through careful optimization of the preprocessing pipeline and model architecture, rather than relying on extensive data augmentation or complex ensemble methods. The effectiveness of parallel dilated convolutions in handling underwater imagery suggests promising applications beyond turtle segmentation, potentially extending to other marine species monitoring where data collection is similarly challenging. While our implementation focused on flipper segmentation, the architecture's robust performance indicates potential for multi-class segmentation of other anatomical features. Future research could explore adaptive normalization techniques specifically designed for underwater imagery variations and investigate the integration of attention mechanisms to better handle extreme poses and lighting conditions.

Future work is required in enhancing these underwater images so models like U-Net, Masked R-CNN, and DeepLabV3 can perform better.

IX. References

[1] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137-1149, Jun. 2017.

[2] S. Pavithra and J. C. M. Denny, "An efficient approach to detect and segment underwater images using Swin Transformer," Results in Engineering, vol. 23, p. 102460, 2024, doi: 10.1016/j.rineng.2024.102460.

[3] Y. H. Kim and K. R. Park, "PSS-net: Parallel semantic segmentation network for detecting marine animals in underwater scene," Frontiers in Marine Science, vol. 9, 2022, doi: 10.3389/fmars.2022.1003568.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *arXiv:1409.1556*, 2014, [online] Available: http://arxiv.org/abs/1409.1556.

[5] I S. Christian, L. Wei, J. Yangqing, S. Pierre, R. Scott, A. Dragomir, et al., "Going deeper with convolutions", *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1-9, 2015.

[6] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition", *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 770-778, Jun. 2016.

[7] *Petru Potrimba. (Aug 9, 2023). What is Mask R-CNN? The Ultimate Guide. Roboflow Blog: https://blog.roboflow.com/mask-rcnn/ (accessed Nov. 13, 2024).*

[8] X. Yang et al., "Instance Segmentation and Classification Method for Plant Leaf Images Based on ISC-MRCNN and APS-DCCNN," in IEEE Access, vol. 8, pp. 151555-151573, 2020, doi: 10.1109/ACCESS.2020.3017560

[9] S. Song, J. Zhu, X. Li and Q. Huang, "Integrate MSRCR and Mask R-CNN to Recognize Underwater Creatures on Small Sample Datasets," in *IEEE Access*, vol. 8, pp. 172848-172858, 2020, doi: 10.1109/ACCESS.2020.3025617.

[10] A. R. Beeravolu, S. Azam, M. Jonkman, B. Shanmugam, K. Kannoorpatti and A. Anwar, "Preprocessing of Breast Cancer Images to Create Datasets for Deep-CNN," in *IEEE Access*, vol. 9, pp. 33438-33463, 2021, doi: 10.1109/ACCESS.2021.3058773.

[11] Tavakoli, Nasrin & Karimi, Maryam & Norouzi, Alireza & Karimi, Nader & Samavi, Shadrokh & Soroushmehr, S.M.Reza. (2019). Detection of abnormalities in mammograms using deep features. Journal of Ambient Intelligence and Humanized Computing. 14. 1-13. 10.1007/s12652-019-01639-x.

[12] Hunter J. Howell, Richard H. Legere, David S. Holland, Richard A. Seigel; Long-Term Turtle Declines: Protected Is a Verb, Not an Outcome. Copeia 1 September 2019; 107 (3): 493–501. doi: https://doi.org/10.1643/CH-19-177.

[13] LearnOpenCV: Batch Normalization in Deep Networks. https://learnopencv.com/batch-normalization-in-deep-networks/

[14] Srivastava et al., Dropout: A Simple Way to Prevent Neural Networks from Overfitting. JMLR, 2014 https://colab.research.google.com/github/d2l-ai/d2l-en-colab/blob/master/chapter_multilayer-perceptrons/dropout.ipynb

[15] UNSW Computer Vision Lecture Slides, COMP9517 Deep Learning Part 1, Term 3, 2024

[16] Machine Learning Mastery: A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks URL: https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/

[17] Chen, L. C., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587.

[18] 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).

[19] 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).

[20] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456).