Automated Segmentation of Sea Turtle Body Parts Using Computer Vision Techniques

Sujan Bharadwaj

Abstract –

This project uses computer vision techniques to segment sea turtle photographs. It achieved this through three different models: U-Net, DeepLabV3, and PSPNet. PSPNet model had the best performance out of the three models.

Keywords - segmentation, SeaTurtleID2022 Dataset, U-Net, DeepLabV3, PSPNet

INTRODUCTION

The segmentation of sea turtles from photographs is crucial for ecological research, enabling effective population monitoring and behavioral analysis and management. Manual segmentation of body parts such as the head, flippers, and carapace are labor-intensive, highlighting the need for automated solutions. This project aims to develop and evaluate different computer vision methods for segmenting sea turtle body parts using the SeaTurtleID2022 dataset. The dataset includes 8,729 images spanning over 13 years, making it suitable for training and testing various deep-learning models. The SeaTurtleID2022 dataset allows us to split the data using an open set split where the data is split into a train, test and validation set based on the year in which the image was captured.

LITERATURE REVIEW

This project leverages several prominent deep learning-based image segmentation models: U-Net, DeeplabV3, and PSP Net, each contributing unique strength to our approach.

U-Net: U-Net, introduced by Ronneberger [1], has become a foundational model in biomedical image segmentation. Its architecture consists of a contracting path to capture context and a symmetric expanding path to achieve precise localization. Skip connections between the contracting and expanding paths allow for detailed feature transfer, enabling better segmentation of small structures. U-Net's effectiveness is enhanced by extensive data augmentation, which helps the model generalize from limited training data. This architecture has proven to be versatile, achieving robust performance across various segmentation tasks by balancing context capture and localization.

DeepLabV3: Chen [2] developed DeepLabV3, which introduces atrous (dilated) convolution to address the limitations of down-sampled feature maps. Atrous convolution allows for dense feature extraction without increasing the number of parameters, thereby maintaining spatial resolution. DeepLabV3 also incorporates Atrous Spatial Pyramid Pooling (ASPP), which samples input features at multiple scales to improve the segmentation of objects appearing in varied sizes. The model's ability to integrate contextual information while retaining high-resolution details makes it a strong candidate for precise segmentation tasks.

PSPNet: PSPNet (Pyramid Scene Parsing Network), proposed by Zhao [3], introduces a pyramid pooling module that aggregates global contextual information at different scales. This approach allows the network to recognize objects in complex scenes by considering both local and global features. The pyramid pooling layer significantly enhances the model's ability to parse scene context, resulting in more accurate segmentation maps, particularly for images with varied backgrounds and object sizes.

SeaTurtleID2022 Dataset: The SeaTurtleID2022 dataset serves as a comprehensive benchmark for this project. Spanning over 13 years and containing 8,729 images of 438 unique turtles, it provides detailed annotations, including body part segmentation masks. This dataset's extensive time span and varied image conditions make it ideal for training models that need to generalize well in real-world ecological studies. [4]

These models and custom implementations provide a balanced approach in understanding how different architectures perform in segmenting sea turtle body parts. The comparative analysis will highlight the strengths and potential drawbacks of each model and offer insights into the most effective techniques for this task.

METHODS

In this project, we have developed and evaluated three primary segmentation methods based on U-Net, DeepLabV3, and PSPNet. Each method was implemented and tested according to the following structured approach:

A. *Data Preparation*

The SeaTurtleID2022 dataset was prepared by splitting images into training, validation, and test sets based on time-aware open-set split. In this type of split, images from specific time durations are allocated to either training, testing or validation. For this project, images taken up to the year 2018 were allocated to the training set, 2019 to the validation set, and 2020 onwards to the test set. According to [5], the open set split closely resembles the real-world settings because the images from a period are in the same set. If the images are split randomly, they often originate from the same observation and thus lead to data leakage, which might result in the overfitting of the model.

A subset of 2,000 images was chosen from the entire dataset for training and testing the models. Annotated images were created using the COCO API, and metadata was used to confirm the split. Data augmentation techniques were applied to the training set to enhance generalization and included flipping, rotation, translation, and brightness adjustments. By performing data augmentation, we are ensuring model is learning equally from all the classes.

Approximately all the images were of the size 2000x1333 pixels and were preprocessed to ensure that all images and masks were resized to 224x224 pixels.

B. *Model Architecture*

UNET Model

The customized U-Net architecture developed for this project aimed to better capture the unique features of sea turtle images and improve the segmentation of sea turtle body parts under diverse lighting and occlusion conditions. The model followed a typical U-Net structure with enhancements to improve feature extraction and detail preservation:

Encoder Path (Contracting Path): The encoder progressively captures high level features by reducing spatial dimensions through max pooling and increasing depth through convolutional layers. Each encoder block consists of:

- Two convolutional layers with a kernel size of 3x3, maintaining spatial dimensions through 'same' padding.
- Batch Normalization after each convolution to stabilize and speed up training
- ReLu activation functions, which adds non-linearity

Max pooling layer after each block to down sample the feature maps, followed by drop out with a rate of 0.3, enhancing generalization and reducing overfitting.

Bottleneck: Situated between the encoder and decoder, the bottleneck is a deeper convolutional layer that learns complex and abstract features. This layer uses an increased number of filters to capture high-level representations, serving as the model's deepest layer and connecting the contracting and expanding paths.

Decoder Path (Expanding Path): The decoder path up samples features maps back to the original resolution for pixel-level predictions. Each decoder block includes:

- Transposed convolution layer for upsampling.
- Skip connections from corresponding encoder layer to retain spatial information lost during down sampling, enhancing the network's ability to localize fine grained features.
- Two Convolutional layers with batch normalization and ReLu activation, ensuring learned features are integrated with encoder information.

Output Layer: A final 1 X 1 convolutional layer with a SoftMax activation outputs the segmented map with pixel-wise class probabilities, allowing the network to distinguish between multiple classes.

DeepLabV3

Implemented with a ResNet-50 backbone, this model uses atrous convolution and the ASPP (Atrous Spatial Pyramid Pooling) module to manage multi-scale feature extraction, aiding in segmenting diverse image regions.

Atrous convolutional, also called the dilation convolution, is used to extract features at specific scale by using dilation rate. Unlike standard convolution, it increases the receptive field without increasing the number of parameters. This is because the dilation leaves the gap between the kernel elements in all the four directions leading to increased receptive field. For example, 3x3 kernel has a receptive field of 5x5.

Atrous Spatial Pyramid Pooling (ASPP): An object in one part of an image might be smaller than the other. To address this, the ASPP module applies parallel atrous convolution with different dilation rates to capture multi-scale features. This ensures the model captures fine-grained details and learns the broader contextual information.

The deeplabv3 model is composed of ASPP module combined with global pooling and the decoder.

The model starts with using ResNet-50 as the backbone. The layer 'conv3_block4_out' having shape (28, 28, 512) from the backbone is used as an input to the ASPP module.

In the ASPP module, parallel atrous convolution is achieved using dilation rates 1, 6, 12, 18 all extracting 256 feature maps. Global average pooling is applied to get information at global level. It takes the average of the entire feature map and outputs an image of size 1x1 with 256 feature maps.

The overall 1280 (256 x 5) features are then concatenated and again passed through 1x1 convolution which is a linear combination of all channels at each spatial position.
This convolution gives the desired number of features that can save computation cost and reduce redundant features to help model avoid overfitting.

Decoder: The output from the ASPP module is then passed on to the decoder module, which performs gradual upsampling of the images. It achieves this using Conv2DTranspose followed by BatchNormalisation and ReLu function.

At the last step of the model, the number of features is reduced to the number of classes to be predicted using a 1x1 convolution. [6]

PSPNet

The PSPNet model is used to enhance semantic segmentation. This model uses a pyramid pooling module to aggregate the features at different-sized pooling operations. These pooling operations are performed on the feature maps obtained from a pre-trained convolution network that acts as a backbone network for our model. Further, these pooled features are concatenated to get more refined features. Then, to better capture contextual details (local and global), we use additional convolution layers. This as an output gives us significantly enhanced segmentation accuracy.

Our custom PSPNet architecture initiates the processing by taking the input image of 224X224X3. This image is then processed through ResNet-50, which is our deployed backbone model, which uses its deep residual connections for learning of intricate patterns efficiently. In this specific implementation, we utilize the output from the conv5_block3_out layer specifically because it serves as a deep feature source, with a semantically rich feature map that captures high-level semantic information that is essential for rate segmentation tasks.

Upon extracting features from the backbone network, the model architecture deploys Pyramid Pooling module. The Pyramid Pooling is used for integrating multi-scale contextual information. This module ensures that we grasp both macro and micro aspects of information by applying parallel pooling operation at different scales to abstract spatial hierarchies of features.

To achieve this, we utilize the feature maps provided by the ResNet-50 backbone and then apply four different pooling scales of 1X1, 2X2, 3X3 and 6X6 to assimilate featured from different spatial extents to capture context at various granularities. The output from each pooling scale is then first passed through a 1X1 convolution layer. This operation helps compress the channel dimension, to maintain efficiency and returns feature maps with reduced depth while retaining critical information. Then we process these outputs from each pooled feature maps back to the same spatial dimension as the backbone's output using our custom Resize Layer function.

The processed feature maps, after resizing, are then concatenated into a single comprehensive feature map. This aggregated feature map consists of detailed local features as well as broader contextual information. This feature map is then processed through a 3X3 convolution with 512 filters, then subsequent batch normalization and ReLU activations to refine the features. Further we integrate additional context into these refined features, using mid-level features from the conv4_block6_out layer from ResNet-50, which ensure high-resolution details are maintained for accurate pixel-level predictions.

The model architecture then shifts focus to up-sampling using skip connections. This is achieved by performing strategic resizing and concatenation with earlier extracted features, as it helps us restore the original resolution of the input while enriching the up-sampled outputs with those of skip connection features to ensure seamless integration. Our model's first skip connection uses the output from the conv3_block4_out layer and merges it with the up-sampled output. This process helps enhance the up-sampled features with detailed spatial information. Then we repeat the same process but now by merging the up-sampled output and features from conv2_block3_out, to fuse more detailed context and further enrich the feature set. At each stage we also use our custom build Resize Layer function, to ensure the dimensions of the up-sampled output aligns with the features from skip connections for efficient integration.

These resized features are then processed through a last step of 64 Filter 3X3 Convolution layer, followed by batch normalization and ReLU activation. This process helps us ensure that the segmentation map is precise, pinpoints the exact segmentation boundaries and has refined segmentation details.

The model architecture concludes with a final 1X1 Convolution with SoftMax activation, that helps us neatly categorize each pixel into its respective segment based on the predefined classes. As a result, we get a detailed segmentation map that matches the input dimensions.

EXPERIMENTAL RESULTS

A. *Experimental Setup*

1. *Hardware and framework*

For the development and evaluation of the segmentation models, Google Colab was used as the processing environment. Specifically, we used the high-performance A100 GPU for faster training and testing of the models. All the segmentation models in this project were implemented using the TensorFlow library in Python.

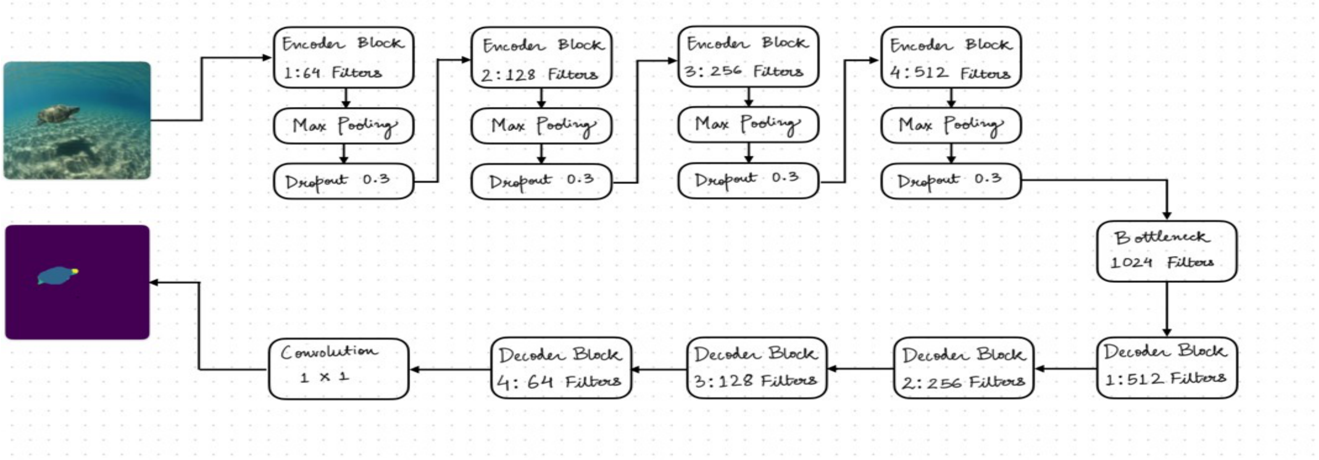

Fig 1: U-Net model architecture

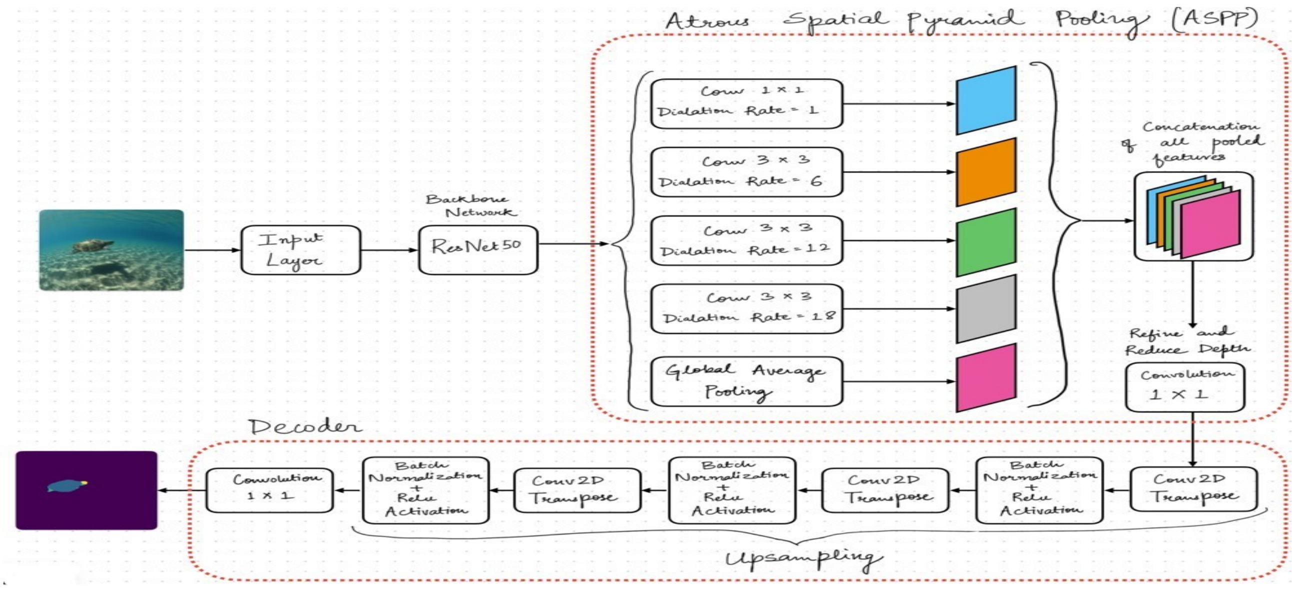

Fig 2: DeepLabV3 model architecture [6]

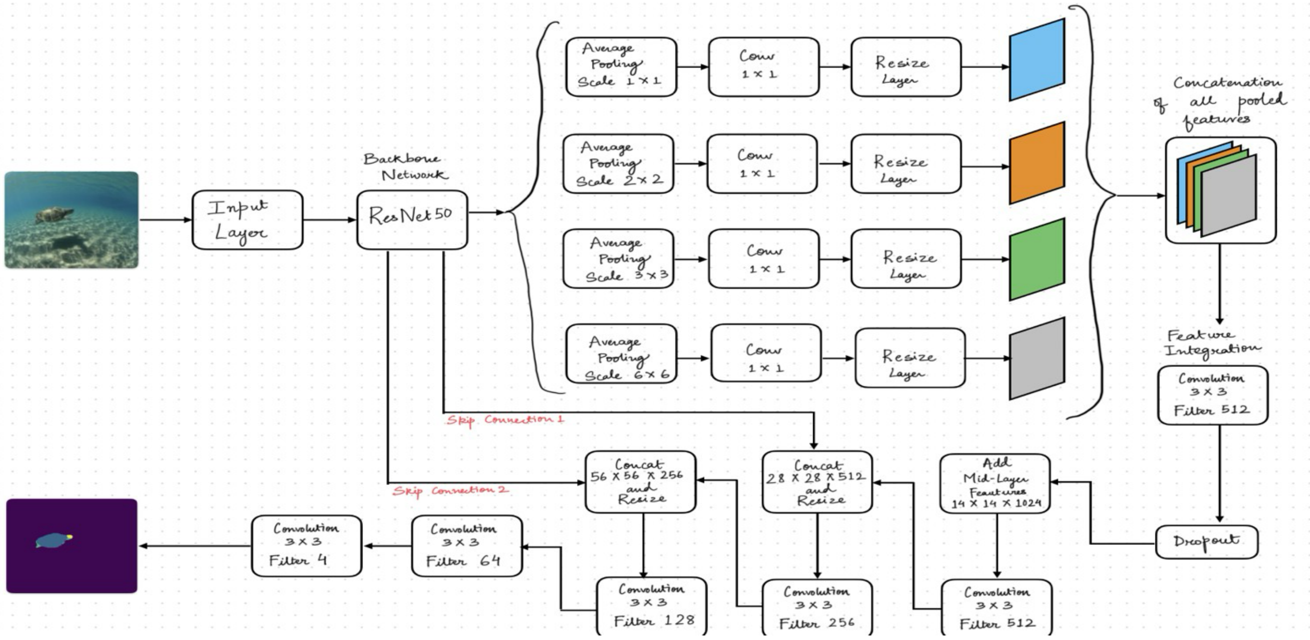

Fig 3: PSPNet model architecture

2. *Pre-training process*

Time-aware open-set splitting method has been employed in this project to divide the dataset into training, validation, and test. As opposed to random splitting, this method is more reliable and realist as it aids the model to generalize across different periods. Images captured between 2010 and 2018 are all considered for training purposes. Similarly, validation images have a timestamp of 2019, and the rest are taken for testing purposes.

A series of preprocessing steps are applied to images and their corresponding masks. All the images and the masks are resized to a fixed dimension of 224x224 pixels. This ensures a good balance between retaining sufficient details and achieving computational efficiency. To improve generalization, data augmentation has been incorporated. Images and masks are randomly flipped and rotated, horizontally and vertically. To adjust to the different lighting scenarios underwater, brightness of the images is varied.

All these preprocessed images are then stored in three separate folders for training, validation and test set images and masks respectively to further retrieve input for the models.

3. *Training process*

UNet

The U-net model has 31,043,716 trainable parameters. It was trained for 30 epochs with a learning rate of 0.0001 using Adam optimizer and sparse categorical cross-entropy as the loss function which is a suitable choice for multi-class segmentation task. To mitigate the overfitting, early stopping was implemented during training.

DeepLabV3

This model, due to its intricate ASPP module was trained for 50 epochs with a default learning rate of 0.001. Training utilized the Adam optimizer and sparse categorical cross-entropy as the loss function. To avoid fluctuations in the validation accuracy of the model, a learning rate scheduler was implemented. A patient level of 3 was selected which reduces the learning rate by a factor of 0.5 if the validation loss does not improve for 3 epochs. This ensures dynamic adaptation to the model performance changes.

PSPNet

This model was set to train with 30 epochs in total. Like u-net, a learning rate of 0.0001 was explicitly defined using Adam optimizer while sparse categorical cross-entropy is chosen as the loss function. This combination of parameters aims to provide smooth convergence of the model.

B. *Results*

1. *Performance metrics*

Model	**Mean IoU** **Backg-round**	**Mean IoU** **Head**	**Mean IoU** **Flipper**	**Mean IoU** **Carap-ace**
UNet	0.9736	0.6783	0.4924	0.5309
Deep LabV3	0.9796	0.7532	0.5069	0.5122
PSPNet	0.9880	0.8560	0.6938	0.7163

Table 1: Models and IOU

- **UNet Model:**

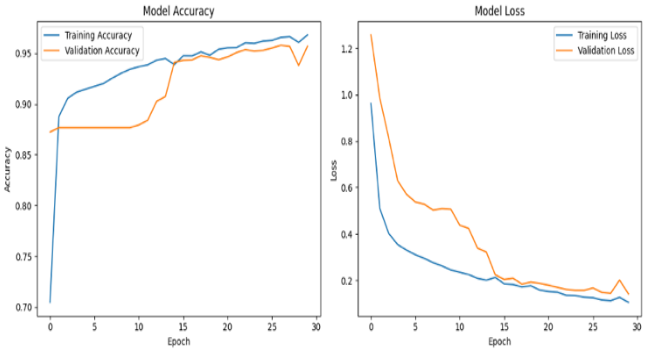

Fig 4: UNet Accuracy vs Epoch and Loss vs Epoch

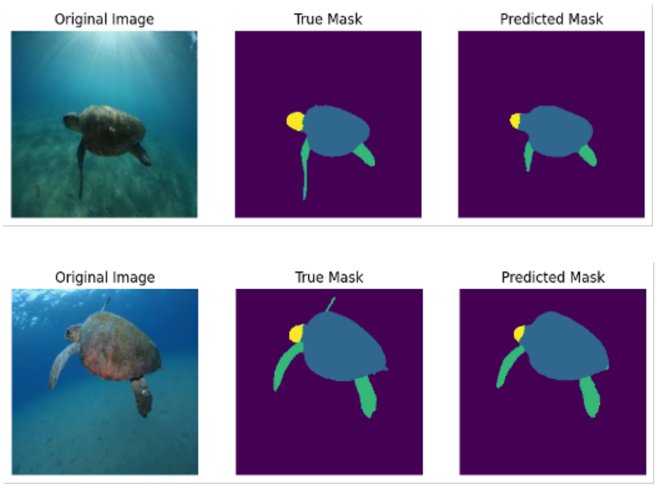

Fig 5: UNet Output

Analysis:

Background Segmentation: The UNet model achieves a high IoU for the background (0.9736), indicating accurate segmentation of background regions.
Head, Flipper, and Carapace Performance: The IoU for the head is moderate (0.6783), whereas the IoU for the flipper (0.4924) and carapace (0.5309) are low, suggesting that the model struggles with accurately identifying these finer details.
Training and Validation Patterns: The accuracy and loss plots indicate stable training with convergence by 30 epochs. However, there remains a noticeable gap between training and validation accuracy, which points to slight overfitting.

- **DeepLabV3 Model:**

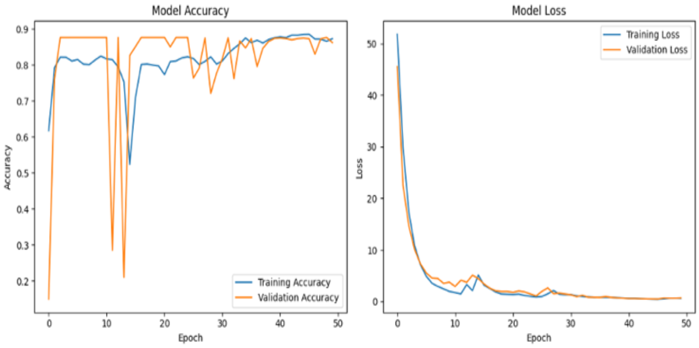

Fig 6: DeepLabV3 Accuracy vs Epoch and Loss vs Epoch

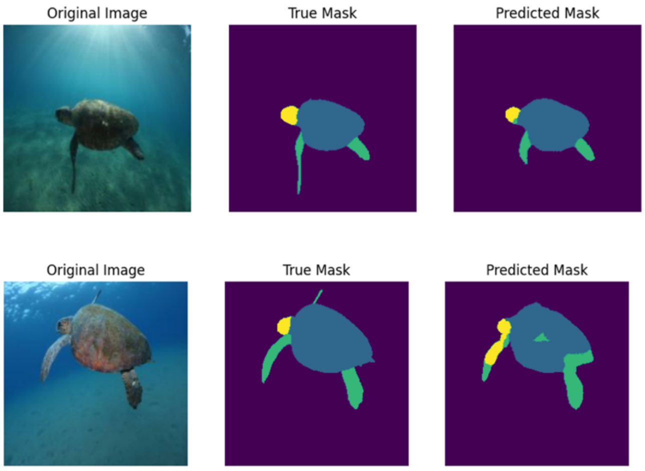

Fig 7: DeepLabV3 Output

Analysis:

Background Segmentation: DeepLabV3 performs well in the background, achieving an IoU of 0.9796, like UNet's performance in this region.
Head, Flipper, and Carapace Performance: It achieves a slightly higher IoU for the head (0.7532) and flipper (0.5069), with a comparable IoU for the carapace (0.5122), showing moderate improvement over UNet, especially for the head.
Overall Performance: The mean IoU is higher than UNet's, indicating better overall segmentation performance across regions.
Training and Validation Patterns: The training and validation accuracy curves fluctuate more than those of UNet, suggesting some sensitivity to batch variations or hyperparameter settings. Despite these fluctuations, the model demonstrates good generalization.

- **PSPNet Model:**

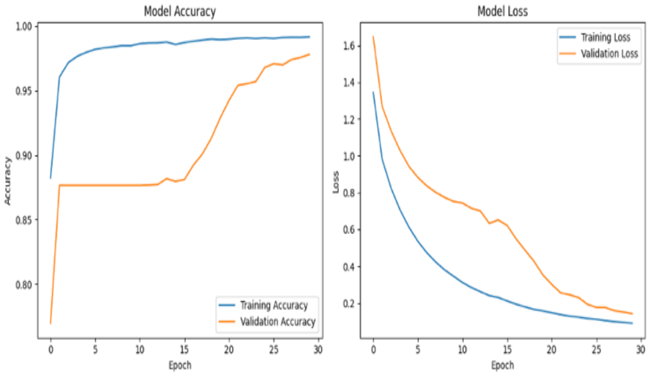

Fig 8: PSPNet Accuracy vs Epoch and Loss vs Epoch

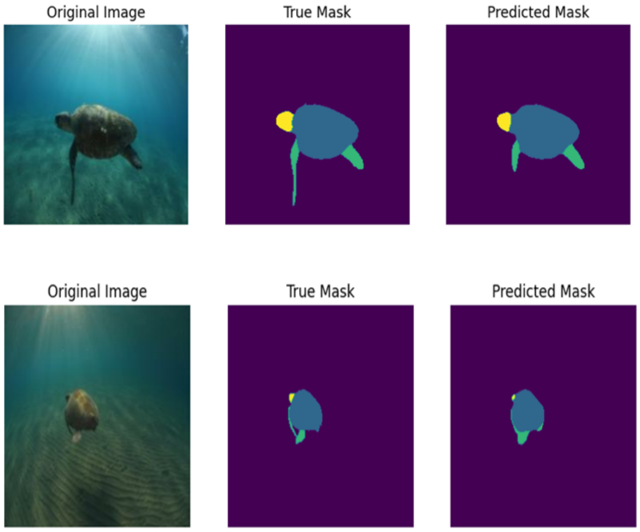

Fig 9: PSPNet Output

Analysis:

Best Overall Performance: PSPNet achieves the highest IoU across all categories, with a mean IoU of 0.9880 for the background, 0.8560 for the head, 0.6938 for the flipper, and 0.7163 for the carapace, making it the most accurate model among the three.

Improvement in Challenging Regions: The model shows significant improvement in IoU for challenging regions like the head, flipper, and carapace, demonstrating its effectiveness in detailed segmentation tasks.

Training Stability: The accuracy and loss curves indicate stable training with minimal fluctuations and a smooth convergence pattern, suggesting effective training and better generalization compared to the other models.

2. Comparison of methods

UNet

Net performs well in segmenting the background but struggles with specific turtle parts, particularly the flipper and carapace, where IoU scores are low. The training and validation curves show stable convergence, though a noticeable gap suggests mild overfitting. UNet is the least effective of the three models for segmenting minute details.

DeepLabV3

DeepLabV3 offers a slight improvement over UNet, especially in the head region, with a moderately higher mean IoU, indicating a more balanced performance. However, its accuracy fluctuates more during training, highlighting sensitivity to data or hyperparameters. DeepLabV3 serves as a good compromise, providing decent segmentation without excessive complexity.

PSPNet

PSPNet achieves the highest IoU across all regions, making it the most accurate model for capturing the details of turtle parts. Its training and validation curves display stable convergence with minimal fluctuations, indicating effective training and strong generalization. PSPNet is the best performer in this segmentation task, especially for challenging regions like the flipper and carapace.

3. Error analysis

UNet

Most errors are found in thin, irregular regions like the flippers and low-contrast areas like the carapace. The UNet model's architecture struggles with capturing minute details, resulting in lower IoU scores for these challenging parts.

DeepLabV3

DeepLabV3 shows slight improvements over UNet but has inconsistencies due to batch sensitivity and fluctuating accuracy during training. It performs better on broader, well-defined regions like the head and carapace but continues to struggle with thin, intricate regions.

PSP Net

PSPNet exhibits the fewest errors, excelling in fine-grained segmentation. Errors are minimal and confined to slight boundary misclassifications, making it the most reliable model for complex segmentation tasks with challenging regions.

4. Model improvements

UNet

To improve UNet's segmentation accuracy, consider adding multi-scale context through dilated convolutions and incorporating attention mechanisms to enhance focus on intricate areas. Using a pretrained backbone and adding deep supervision could improve performance on challenging parts like the flippers and carapace.

DeepLabV3

Improve stability in DeepLabV3 by fine-tuning hyperparameters and adding a lightweight decoder to sharpen boundary segmentation. Increasing the depth of the ASPP module and experimenting with boundary-aware loss functions could help enhance accuracy, especially for thin and complex structures.

PSPNet

To further optimize PSPNet, consider enhancing the Pyramid Pooling Module with additional pooling scales and incorporating attention mechanisms for capturing fine details. Adding auxiliary losses and hybrid loss functions, along with advanced data augmentation techniques, could further boost performance, especially in intricate regions.

Dataset Size as an Improvement

Currently, each model is trained on only a subset of 2000 images out of 8729. Increasing the dataset size could improve the model's performance by providing a broader range of examples, which would help in learning more robust features, particularly for the segmentation of intricate areas like flippers and carapaces.

References

[1] Convolution Networks for Biomedical Image Segmentation: https://link.springer.com/chapter/10.1007/978-3-319-24574-4_28

[2] DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs https://arxiv.org/pdf/1606.00915v2

[3] Pyramid Scene Parsing Network: https://arxiv.org/pdf/1612.01105v2

[4] Wildlife Datasets, *Sea Turtle ID 2022*: https://www.kaggle.com/datasets/wildlifedatasets/seaturtleid2022. [Accessed: Oct. 12, 2024]

[5] L. Adam, V. Čermák, K. Papafitsoros and L. Picek, "SeaTurtleID2022: A long-span dataset for reliable sea turtle re-identification," *Czech Technical University*, 2022.

[6] S. Japsica, C. Maharshi, S. Priya, B. Sujan and B. Anmol, "Semantic Segmentation in Unstructured Environments", COMP9444 – Neural Networks and DeepLearning Project, UNSW, 2024.