Semantic Segmentation task: Automated Sea Turtle Segmentation

1st Xuda Chen
zID: z5527738
Email: z5527738@ad.unsw.edu.au

2nd Zhihong Jin
zID: z5613826
Email: z5613826@ad.unsw.edu.au

3nd Wenyu Yin
zID: z5516670
Email: z5516670@ad.unsw.edu.au

4nd Yizhuo Xu
zID: z5460384
Email: z5460384@ad.unsw.edu.au

5nd Hao Zhu
zID: z5503328
Email: z5503328@ad.unsw.edu.au

***Abstract*—In this study, we employ a diverse set of semantic segmentation models, including both traditional clustering techniques (Mean-shift and K-means) and advanced deep learning architectures (U-Net, DeepLabV3 and DeepLabV3+). Additionally, we have developed some innovative models (DeepLabV3Plus_MobileNetV2, DeepLabV3Plus_VIT, DeepLabV3Plus_VIT2 and DeepLabV3Plus_VIT3) that incorporate unique architectural modifications to further enhance segmentation performance. This paper describes the design and implementation details of each model.**

I. Introduction

In wildlife research, identifying individual animals in images is essential but often requires manual effort from specialists, making the process labor-intensive and time-consuming. Advances in computer vision offer promising solutions to automate and streamline this work.

Our project specifically focuses on segmenting various parts of sea turtles, including the shell, flippers, and head, using both traditional machine learning and neural network models. By exploring, comparing, and refining these approaches, we aim to enhance the accuracy and efficiency of automated animal identification, providing valuable insights into the potential of modern computer vision techniques for wildlife research.

The dataset utilized in this study originates from Kaggle and focuses on sea turtles, naming SeaTurtleID2022 [1], it's an extensive resource for research on sea turtle identification and re-identification, containing 8,729 images of 438 unique individuals. The images vary in size, with common dimensions like 2000x1333 and 2000x1500 pixels, offering high-resolution visuals under diverse environmental conditions. It includes four main files: images, annotations.json, metadata.csv, and metadata_splits.csv. The annotations.json file is COCO API-compatible, categorizing images into three segments: The whole turtle, flippers, and head. Metadata files provide details such as capture date, timestamp, unique identifiers, and split strategies. The dataset features both open-set and closed-set splits. Open-set splitting offers realistic performance evaluation by placing different images of the same turtle into various subsets (train, validation, and test), mimicking real-world scenarios where new individuals are encountered. Closed-set splitting, on the other hand, keeps all images of the same turtle within one subset. This structure makes SeaTurtleID2022 highly useful for developing and evaluating advanced computer vision models in wildlife research.

We employed Mean-shift [2] and K-means [3] as representative traditional machine learning methods. While traditional techniques can perform adequately on simpler tasks, they reveal significant limitations when faced with complex scenes. Specifically, these methods struggle with large variations in lighting, complex boundaries and shapes, and are sensitive to parameter settings. Additionally, they lack adaptability to diverse datasets and suffer from poor computational efficiency and scalability. In contrast, deep learning methods based on neural networks demonstrate greater adaptability and robustness in complex scenarios.

To assess the potential of deep learning, we implemented neural network models based on U-Net [16], DeepLabV3 [5] and DeepLabV3+ [7] architectures to detect sea turtles in the images and compared their performance with the two traditional methods. Our results highlight the superiority of neural network models in handling this task.

Building on DeepLabV3+, we developed several enhanced models, including DeepLabV3+ with MobileNetV2, DeepLabV3+ with Vision Transformer (ViT), and two advanced variants, DeepLabV3Plus_ViT2 and DeepLabV3Plus_ViT3. The last three models were progressively designed to address specific challenges and improve performance.

The DeepLabV3Plus_ViT model (see Fig. 3) uses a Vision Transformer (ViT) backbone to leverage the transformer's strength in capturing global features. To address ViT's limitations in low-level detail processing, DeepLabV3Plus_ViT2 (see Fig. 4) incorporates additional convolutional layers to enhance low-level feature extraction. Further refining the approach, DeepLabV3Plus_ViT3 (see Fig. 1) employs a lightweight version of ViT with an 8x8 patch size. The smaller patches enable better feature representation, enhancing the model's ability to capture finer details and improving segmentation performance, especially in scenarios requiring high precision.

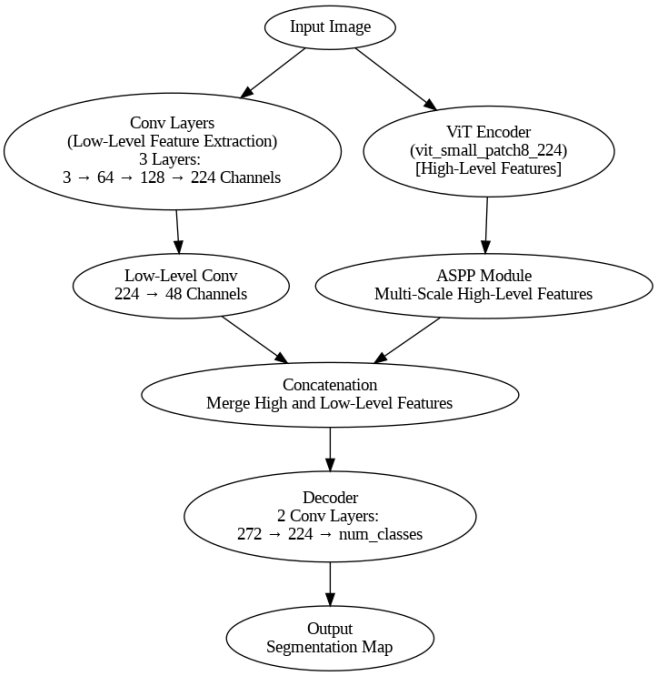

Fig. 1. The structure of our DeepLabV3Plus_VIT3 model.

II. Literature Review

Semantic segmentation is a fundamental task in computer vision that involves dividing an image into regions that correspond to a particular object class. This section reviews key technologies and architectures that have influenced advances in image segmentation, focusing on both traditional and modern deep learning methods.

A. Traditional Machine Learning Techniques

1) Mean-Shift Segmentation: Mean-shift is a mode-seeking clustering algorithm that has been widely applied in computer vision. Comaniciu and Meer [2] introduced a robust feature space analysis method that iteratively shifts data points toward areas of higher density. In image segmentation, Mean-shift can effectively group pixels based on color and spatial proximity. However, it relies on manually engineered features, limiting its effectiveness for complex, multi-class segmentation tasks where adaptive learning is crucial.

2) K-Means Clustering: K-means, one of the most commonly used clustering algorithms, partitions data into k clusters by minimizing intra-cluster variance, as formalized by Linde et al. [3]. This algorithm is simple and effective but struggles with complex image structures. Its performance depends heavily on the initial number of clusters and lacks the ability to capture nonlinear patterns, making it unsuitable for fine segmentation tasks, such as distinguishing multiple anatomical parts of a turtle.

B. Deep Learning Techniques

1) U-Net Architecture: The U-Net model, introduced by Ronneberger et al. [16], is a convolutional neural network designed for biomedical image segmentation. Its encoder-decoder architecture is capable of extracting hierarchical features, which allows it to efficiently segment images with limited data. Skip connections help retain spatial information and improve the model's ability to capture detail. U-Net laid the foundation for many subsequent advances in segmentation, especially those applications that required both global context and local details.

2) DeepLabV3 and DeepLabV3+: DeepLabV3, developed by Chen et al. [5], introduced the concept of atrous (or dilated) convolutions to increase the receptive field while preserving spatial resolution. The Atrous Spatial Pyramid Pooling (ASPP) module captures features at multiple scales, enhancing the model's ability to handle objects of varying sizes. This architectural innovation has become a standard for semantic segmentation, known for balancing the need for contextual information and spatial precision.

DeepLabV3+ [7] further improved upon DeepLabV3 by incorporating a decoder module to refine the segmentation results. The decoder module effectively sharpens object boundaries, addressing a key limitation of previous versions. These improvements make DeepLabV3+ a robust choice for complex segmentation tasks, where accurate delineation of object contours is crucial.

3) DeepLabV3Plus-PyTorch Project: VainF's [8] GitHub project explored the use of different backbones for the DeepLabV3+ model. This project demonstrated the impact of using various pre-trained models as feature extractors by providing a flexible implementation of DeepLabV3+. This work inspired our own experiments with different backbones, such as MobileNetV2 and Vision Transformers (ViT), to investigate how architectural choices influence segmentation performance.

4) MobileNetV2: MobileNetV2, presented by Sandler et al. [9], is a lightweight convolutional neural network that uses inverted residuals and linear bottlenecks to reduce computational complexity. It has been widely adopted in applications requiring efficient processing, showcasing the importance of balancing model performance with computational resource requirements.

5) Vision Transformer (ViT): Dosovitskiy et al. [10] introduced the Vision Transformer (ViT), which applies self-attention mechanisms to images, treating them as sequences of non-overlapping patches. ViT has demonstrated impressive performance in image recognition tasks by capturing long-range dependencies, setting a new benchmark for image classification. Its architecture departs from traditional convolutional models, relying entirely on transformers for feature extraction.

However, its application to segmentation tasks has been limited by its lack of local feature extraction capabilities. Researchers, such as Wu et al. [11], have explored integrating convolutions with transformers to address these shortcomings, leading to hybrid models that combine the strengths of both architectures. These hybrids, such as CvT (Convolutional Vision Transformers), aim to improve segmentation performance by capturing both global context and fine-grained details.

Additionally, the use of smaller patches (see Fig. 2), as explored by Chen et al. [12], has been shown to enhance spatial resolution, making transformers more effective for dense prediction tasks like segmentation.

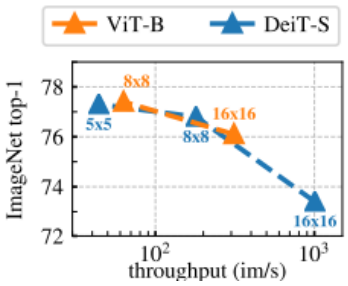

Fig. 2. **Effect of Patch Size.** [12] k-NN evaluation as a function of the throughputs for different input patch sizes with ViT-B and ViT-S(DeiT-S). Models are trained for 300 epochs.

III. Method

In this study, we employ a diverse set of semantic segmentation models, including both traditional clustering techniques and advanced deep learning architectures. Additionally, we have developed some innovative models that incorporate unique architectural modifications to further enhance segmentation performance. This section describes the design and implementation details of each model.

A. Traditional Machine Learning Methods

1) Mean Shift Segmentation: Mean Shift [2] is a non-parametric clustering algorithm used for unsupervised segmentation. It iteratively shifts data points towards regions of higher density using kernel density estimation. In our implementation (based on [13]):

- **Preprocessing**: The input image is converted from RGB to the LAB color space to improve perceptual clustering. The LAB color features are then flattened.
- **Bandwidth Estimation**: The bandwidth, a critical hyperparameter, is estimated using the 'estimate_bandwidth' function from Scikit-learn, which automatically adjusts based on the specified quantile and number of samples.
- **Segmentation**: The Mean Shift model clusters the feature space and assigns each pixel to a cluster, producing a segmented image.

2) K-Means Segmentation: K-Means [14] is another widely used unsupervised clustering technique that partitions data into k clusters by minimizing intra-cluster variance. In our approach (based on [15]):

- **Preprocessing**: Similar to Mean Shift, the image is converted to LAB space, and features are reshaped for clustering.
- **Clustering**: The OpenCV implementation of K-Means is utilized with criteria based on maximum iterations and convergence accuracy. The algorithm is run multiple times to achieve robust segmentation.

B. Deep Learning Methods

1) U-Net Method: U-Net [16] is a fully convolutional neural network (CNN) architecture commonly used for biomedical image segmentation. Our implementation (based on [17]) includes:

- **Encoder**: A series of downsampling convolutional blocks that capture semantic features at multiple scales.
- **Decoder**: Upsampling layers that refine the segmentation mask using skip connections from corresponding encoder layers, which provide fine-grained details.
- **Output**: The final output layer uses a 1x1 convolution to map features to the desired number of classes.

2) DeepLabV3 Method: DeepLabV3 [5] employs atrous (dilated) convolution and an Atrous Spatial Pyramid Pooling (ASPP) module to capture multi-scale context efficiently. Our implementation (based on [6]) uses:

- **Backbone**: A pre-trained ResNet-50 model with modified dilated convolutions for dense feature extraction.
- **ASPP**: Multiple atrous rates are used to process the high-level features, improving performance on objects of varying sizes.

3) DeepLabV3+ Method: DeepLabV3+ [7] extends DeepLabV3 by adding a decoder module, enhancing segmentation accuracy, especially around object boundaries. Our implementation (based on [8]) uses:

- **Encoder**: A pre-trained ResNet-50 extracts both low-level and high-level features.
- **Low-Level Feature Processing**: The low-level features are reduced using a 1x1 convolution to ensure compatibility with the upsampled high-level features.
- **Decoder**: Combines ASPP features with low-level features and refines the segmentation map.

C. Innovative Methods

Inspired by [8], we used the architecture of DeepLabV3+ and tried different models as the backbone. We then made further improvements based on DeepLabV3+ with ViT as the backbone.

1) DeepLabV3+ (MobileNetV2 Backbone): We modify the DeepLabV3+ architecture by using MobileNetV2 [9] as a lightweight, efficient backbone. This setup reduces computational complexity while maintaining competitive performance:

- **Feature Extraction**: Uses MobileNetV2's intermediate layers for low-level and high-level features.
- **ASPP and Decoder**: Similar to the ResNet-50 variant but optimized for MobileNetV2's output channels.

2) DeepLabV3+ (ViT Backbone): This model adapts the DeepLabV3+ architecture to use a Vision Transformer (ViT) [10] backbone.(see Fig. 3) ViT captures global context more effectively, which is crucial for dense prediction tasks:

- **High-Level Features**: Extracted using the ViT model, reshaped into a spatial format for segmentation.
- **Low-Level Features**: Dummy features are used as placeholders, given ViT's lack of spatially detailed representations [10].
- **ASPP and Decoder**: Standard components are used to refine the segmentation output.

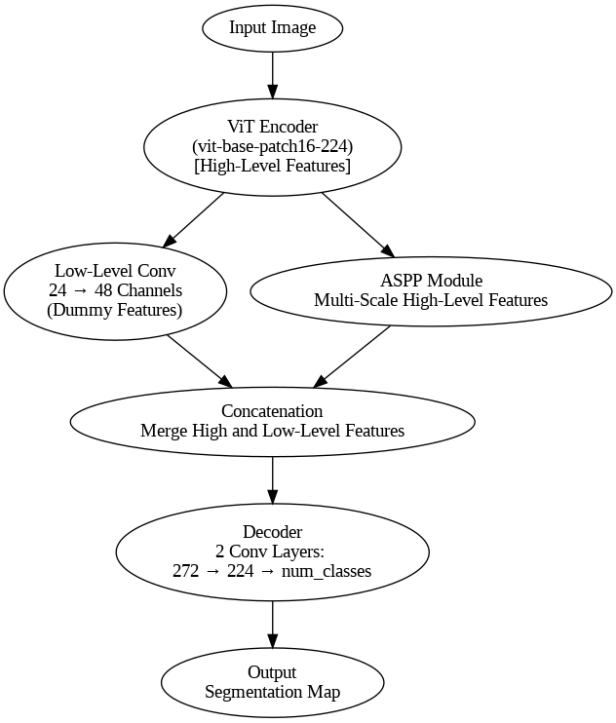

Fig. 3. The structure of our DeepLabV3Plus_VIT model.

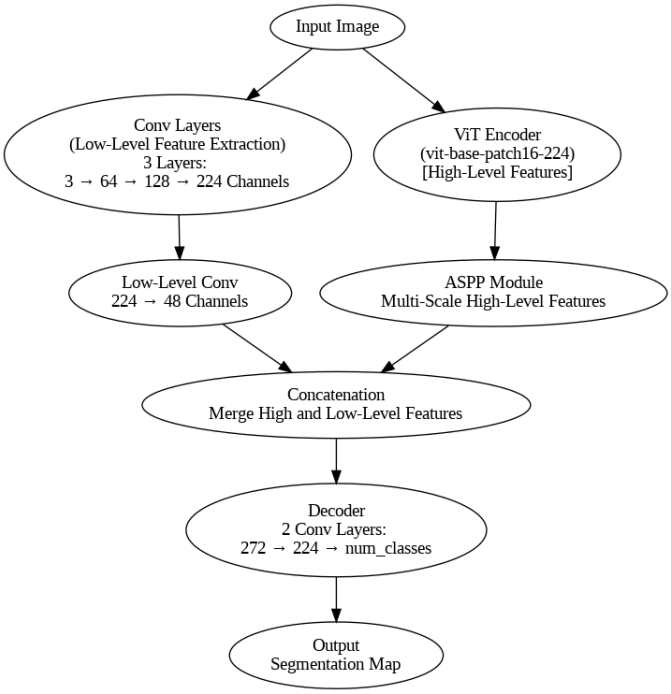

Fig. 4. The structure of our DeepLabV3Plus_VIT2 model.

3) DeepLabV3Plus_VIT2: An enhancement over the previous ViT-based model, DeepLabV3Plus_VIT2 introduces convolutional layers for better low-level feature extraction [11]. (see Fig. 4) Key difference from the previous ViT-based model:

- **Low-Level Features**: Meaningful features are extracted from the input image by a series of additional convolutional layers at the beginning of the model, providing fine-grained details that improve segmentation accuracy.

4) DeepLabV3Plus_VIT3: Based on the comparison in the paper [12], ViT with small versions and 8x8 pixel patches tends to strike a better balance between performance and efficiency (see Fig. 2), significantly improving performance while maintaining relatively fast processing speeds. This variant further optimizes DeepLabV3Plus_VIT2 architecture by using the 'vit_small_patch8_224' model from the TIMM library as backbone (see Fig. 1). Key differences from the previous DeepLabV3Plus_VIT2 model:

- **The Smaller Patches**: Processes finer 8x8 pixel patches, enhancing spatial resolution.
- **The Smaller Architecture**: The smaller, more efficient ViT model balances performance and computational cost, making it suitable for resource-constrained environments.

IV. Experimental Results

A. Preprocessing Steps

Our experiments were conducted using the PyTorch framework, following rigorous steps to ensure reproducibility:

- **Reproducibility:** We ensured experiment reproducibility by setting a consistent random seed for random, numpy, and PyTorch, enforcing fixed algorithms with torch.backends.cudnn.deterministic = True and torch.backends.cudnn.benchmark = False , using the AMSGrad variant of the Adam optimizer and setting shuffle=False in the DataLoader.
- **Data Splits:** We split the dataset into training, validation, and test subsets using the open-set approach specified by the creators in the metadata of the SeaTurtleID2022 dataset [1]. According to the authors, open-set splitting provides a more realistic evaluation of model performance compared to closed-set or random splitting. To speed up training and testing, we used only half of the data for each of the training, validation, and test subsets.
- **Preprocessing Pipeline:** We designed preprocessing functions to standardize the input image sizes, mask functions to ensure accurate label annotations, and a custom TurtleDataset class to efficiently generate images and corresponding labels.

B. Training Procedure

Traditional Machine Learning Models:

- For traditional machine learning models, we manually tuned hyperparameters on the training set to achieve optimal performance:
  - *Mean-Shift:* quantile = 0.2, n_samples = 50
  - *K-means:* k = 4, attempts = 10, max_iter = 100

  These hyperparameters were subsequently used for testing on the test set. Hyperparameter tuning was performed

TABLE I
MIOU RESULTS FOR ALL MODELS

Method Type	Model	Shell mIoU	Flippers mIoU	Head mIoU	Min mIoU	Average mIoU
Traditional ML	Mean-shift	0.0137	0.0494	0.0223	0.0137	0.0285
Traditional ML	K-means	0.0999	0.0299	0.0306	0.0299	0.0535
Neural Network	U-Net	0.6062	0.5374	0.5034	0.5034	0.5490
Neural Network	DeepLabV3	0.8755	0.7092	0.7326	0.7092	0.7724
Neural Network	DeepLabV3Plus	0.8704	0.7307	0.7107	0.7107	0.7706
Neural Network	DeepLabV3Plus_MobileNetV2	0.8525	0.6955	0.6789	0.6789	0.7423
Neural Network	DeepLabV3Plus_VIT	0.8269	0.5720	0.5919	0.5720	0.6636
Neural Network	DeepLabV3Plus_VIT2	0.8671	0.7125	0.7059	0.7059	0.7618
Neural Network	DeepLabV3Plus_VIT3	0.9010	0.7775	0.7803	0.7775	0.8196

exclusively on the training set to prevent data leakage and maintain the integrity of the evaluation.

Deep Learning Models:

- To ensure a fair comparison between different deep learning models, we kept training parameters consistent across all models:
  - *Batch Size:* 16
  - *Target Size:* (256, 256), except for models using Vision Transformer (ViT) backbones, which were limited to (224, 224) [10] due to architectural constraints.
  - *Number of Classes:* 4
  - *Learning Rate:* 1×10^{-4}
  - *Number of Epochs:* 30
- **Loss Function:** CrossEntropyLoss was used to optimize model performance [18].
- **Optimizer:** Adam [19].
- **Activation Function:** ReLU [20]
- **Batch Normalization:** BatchNorm2d layer [21].

During training, we only saved models that achieved the lowest validation loss. We also plotted training curves, including both training and validation loss curves, as well as mean Intersection over Union (mIoU) [22] for each class.(see Fig. 5)

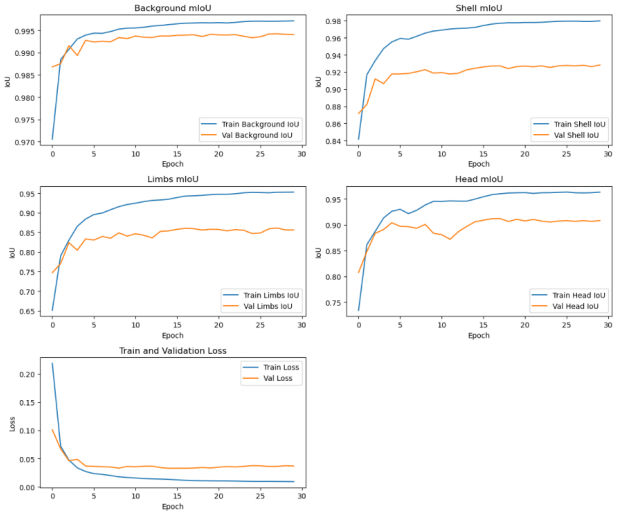

Fig. 5. Training Process Curves for the DeepLabV3Plus_VIT3 Model (Loss, mIoU).

C. Testing Procedure and Results

Performance Evaluation:

- We assess the performance of each method by computing the mean Intersection over Union (mIoU) [22] for each class within the image. For each test image, we generate a pixel-level predicted mask and compare it with the ground truth mask. The IoU for each class is then calculated from these masks. We then sum the IOU values for each class across all images to calculate the mIoU for each class, using this mIoU as the model's performance metric on the test set. We also computed the minimum mIoU and the average mIoU for the three main turtle anatomical parts (shell, flippers, and head). Table I summarizes the results.

Qualitative Analysis:

- We visually examined the segmentation performance of all models on images, selecting some representative samples showing either decent or weak performance, and output their predicted masks to visually assess their segmentation performance.:
  - *High-Quality Images:* In cases where the turtle is prominently visible in the image, most models performed well. Figure 6 showcases examples where segmentation results were accurate and detailed.
  - *Challenging Images:* In more complex scenarios, such as images with multiple turtles, small turtles, or cluttered backgrounds, segmentation quality declined across all models. Figure 7 illustrates examples where performance was suboptimal, highlighting areas for future improvement.

V. DISCUSSION

A. Selection of Minimum mIoU as the Primary Evaluation Metric

In evaluating the performance of our turtle segmentation models, we used minimum mean Intersection over Union (min mIoU) as our main metric. Unlike average mIoU, which provides an overall performance snapshot, min mIoU focuses on the model's weakest performance across all classes. This ensures consistent segmentation across all categories—shell, flippers, and head—rather than excelling only in the easier classes. By prioritizing min mIoU, we aim to create a more

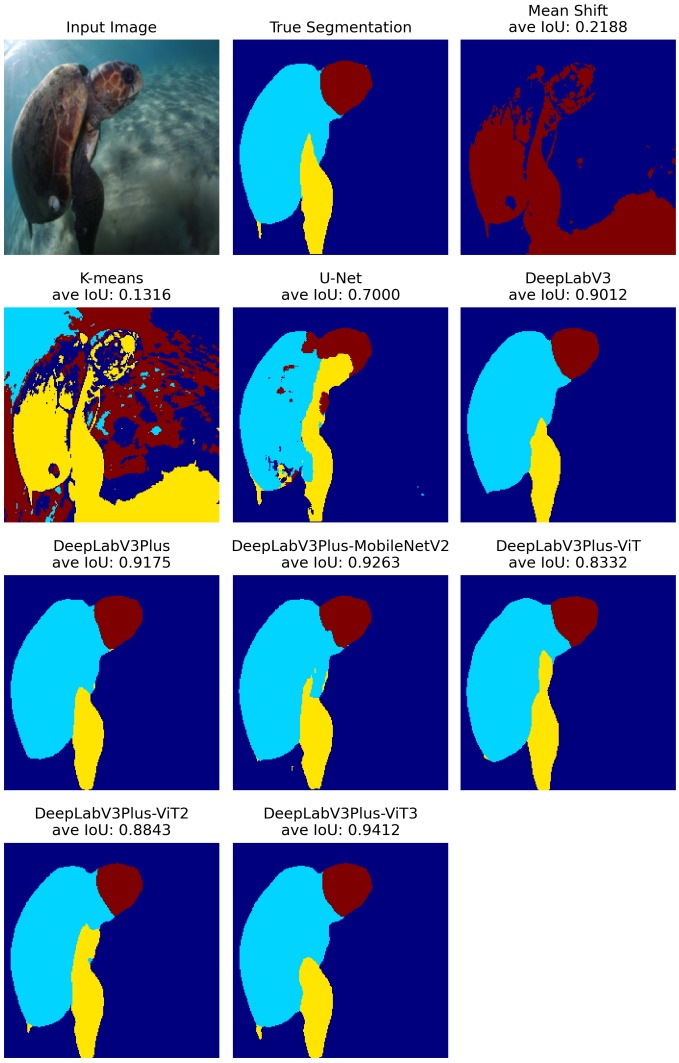

Fig. 6. Segmentation results produced by each model for the image where the turtles are prominently visible and easily distinguishable.

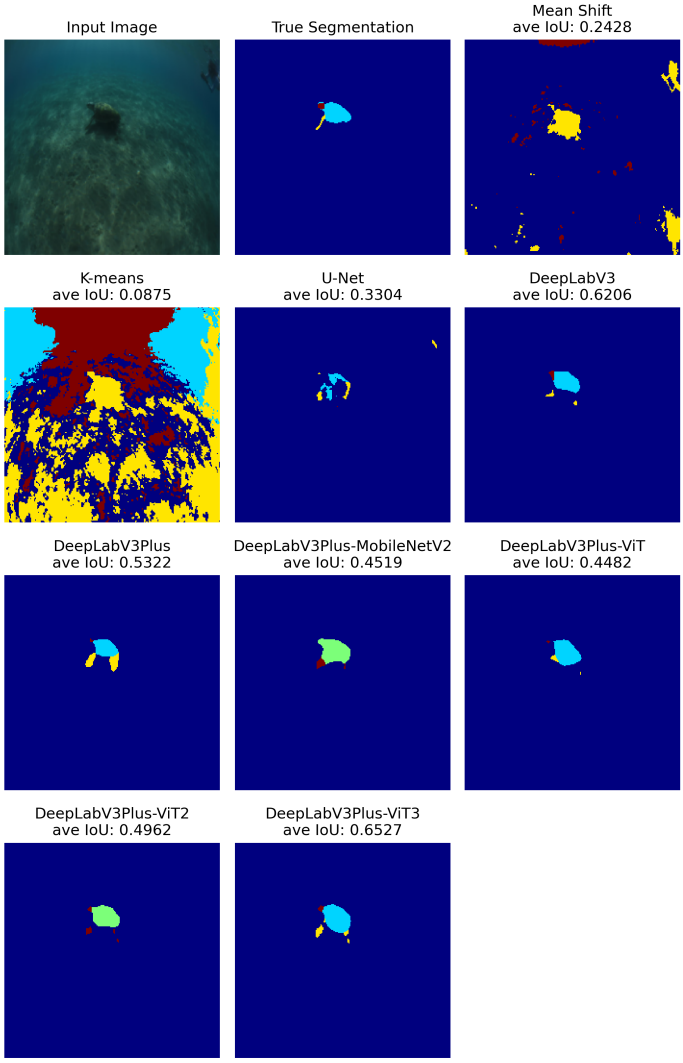

Fig. 7. Segmentation results produced by each model for the challenging image.

robust model, enhancing segmentation reliability across turtle anatomy.

The overall performance ranking based on min mIoU (According to the Table I) is shown in Table II:

TABLE II
PERFORMANCE RANKING BASED ON MIN MIOU

Rank	Model	min mIoU
1	DeepLabV3Plus_VIT3	0.7775
2	DeepLabV3Plus	0.7107
3	DeepLabV3	0.7092
4	DeepLabV3Plus_VIT2	0.7059
5	DeepLabV3Plus_MobileNetV2	0.6789
6	DeepLabV3Plus_VIT	0.5720
7	U-Net	0.5034
8	K-means	0.0299
9	Mean-shift	0.0137

B. "High-quality" images and "challenging" images

Based on the performance of our models on various specific images, we observed a clear distinction between "high-quality" and "challenging" images. Generally, images in which the sea turtles occupy a large portion of the frame, feature only one turtle, and have bright, clear lighting are categorized as "high-quality." Our models consistently perform well on these images, delivering accurate segmentation results (see Fig.6). On the other hand, images where the turtles are small, there are multiple turtles, or the lighting is dim and the scene is less distinct are considered "challenging." All models tend to struggle with these challenging images, resulting in lower segmentation accuracy. (see Fig.7)

We evaluate a model's performance by considering its results on both "high-quality" images and "challenging" images comprehensively. This balanced assessment allows us to

understand how well a model performs under ideal conditions as well as in more difficult scenarios, ensuring a thorough evaluation of its segmentation capabilities.

C. Model Performance Analysis

1) Traditional vs. Deep Learning Methods:

Traditional Machine Learning Methods: Mean-shift scored a min mIoU of 0.0137 and K-means achieved a min mIoU of 0.0299 (see Table II), highlighting their inadequacy for complex, multi-class segmentation. As shown in the images with better segmentation results (see Fig.6), traditional methods like Mean-shift and K-means can only distinguish the whole turtle from the background. The primary limitation lies in their reliance on handcrafted features, which fail to capture the intricate details needed for precise segmentation. Mean-shift struggles with predefined cluster numbers, often only distinguishing foreground and background, while K-means specifies four clusters but remains insufficient for diverse turtle features.

Deep Learning Methods: Deep learning models, including U-Net and DeepLabV3 variants, significantly outperformed traditional methods due to their ability to learn complex features [27]. They excel in segmenting clear images with prominently visible turtles ((see Fig.6)). Unlike traditional methods, which rely on manually engineered features and are sensitive to variations in lighting and shape, deep learning models automatically extract hierarchical features from the data [28], capturing both global context and fine-grained local details [5]. This allows them to handle diverse and complex segmentation tasks more effectively, even in the presence of challenging conditions like overlapping objects, varying scales, and intricate boundaries [5].

2) Detailed Comparison Within the DeepLabV3 Series:

U-Net vs. DeepLabV3 Series: U-Net's min mIoU of 0.5034 contrasts sharply with DeepLabV3's 0.7092 (see Table II), highlighting the effectiveness of DeepLabV3's advanced architecture [5]. This significant difference underscores the superior performance of DeepLabV3 in capturing complex features and providing more accurate segmentation results. The advanced design of DeepLabV3, which includes atrous spatial pyramid pooling and deeper network layers, allows it to better handle the intricacies of the segmentation task compared to the more traditional U-Net architecture.

DeepLabV3Plus and Variants:

- DeepLabV3Plus model showed slightly better performance than DeepLabV3 model, thanks to the addition of a refined decoder module that better delineates object boundaries [7].
- To speed up training, we replaced the ResNet-50 encoder with MobileNetV2, resulting in the DeepLabV3Plus_MobileNetV2 model. Although this change decreased min mIoU (0.6789) (see Table II), it significantly improved training efficiency, highlighting MobileNetV2's suitability for applications requiring faster inference. [9]
- To further enhance segmentation, we integrated a Vision Transformer (ViT) as the encoder in DeepLabV3Plus_VIT model, leveraging ViT's ability to model global information effectively [10]. However, the performance unexpectedly dropped, as indicated by lower mIoU scores (0.5720) (see Table II). The analysis revealed that while ViT excels in capturing long-range dependencies, it lacks the ability to extract local details effectively.
- To address this limitation, we developed DeepLabV3Plus_VIT2 model, incorporating additional convolutional layers to improve low-level feature extraction. This modification led to better segmentation results (0.7059) (see Table II), as the model could now capture both global context and local details more efficiently.
- We then refined the model further by reducing the ViT patch size from 16x16 to 8x8, resulting in DeepLabV3Plus_VIT3 model. Smaller patches allow the model to capture finer details, enhancing local feature representation. Consequently, DeepLabV3Plus_VIT3 model achieved the highest min mIoU (0.7775) (see Table II), demonstrating improved performance, particularly in scenarios where precise segmentation is crucial. Notably, even in images where turtles are small (see Fig.7), DeepLabV3Plus_VIT3 outperformed other models.

D. Reasons for Model Failures

As discussed earlier, the primary reason for the failure of traditional machine learning methods lies in their reliance on handcrafted features, which fail to capture the intricate details required for precise segmentation. In contrast, deep learning models have better performance, among them, DeepLabV3Plus_VIT3 has demonstrated the best performance. This model effectively combines global and local features, enabling it to capture critical information in images better than other deep learning models. Consequently, it achieves superior performance on challenging images. However, despite these advancements, DeepLabV3Plus_VIT3 still struggles with particularly difficult images (see Fig.8).

From Fig.8, it is evident that the primary reasons for model failure include poor lighting conditions (images being too dark), very small turtle size within the frame, and the presence of multiple turtles in the same image. Fundamentally, the root cause lies in the structural limitations of the model contribute to these failures. Despite DeepLabV3Plus_VIT3's improved ability to capture both global and local features, achieving an optimal balance between global context and local detail extraction remains a persistent challenge.

VI. Conclusion

A. Project Summary

1) Method Comparison: Our project demonstrates that deep learning approaches significantly outperform traditional machine learning methods for turtle segmentation. Traditional methods like Mean-shift and K-means exhibited significantly

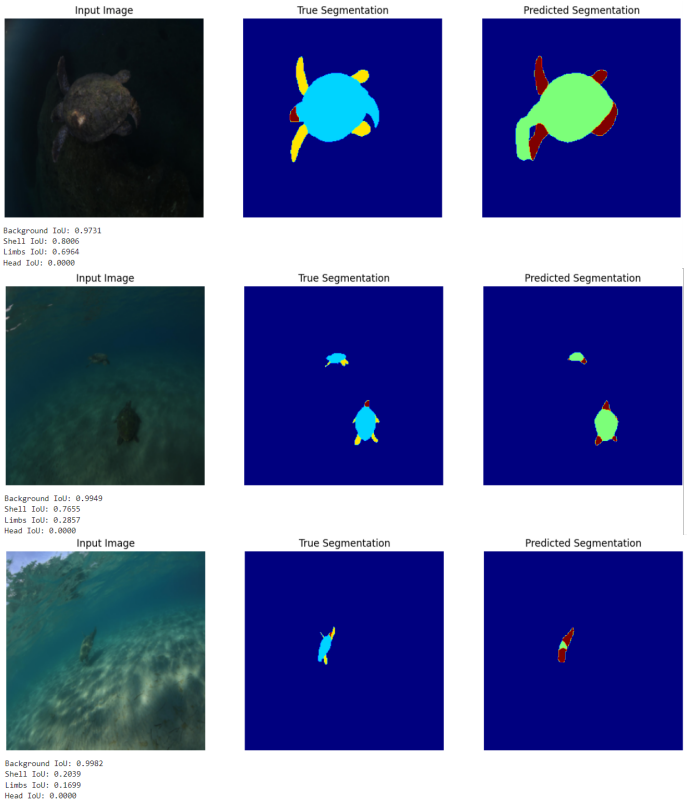

Fig. 8. Segmentation results produced by DeepLabV3Plus_VIT3 for the particularly difficult images.

lower min mIoU scores, highlighting their inadequacy for handling the complexity and multi-class nature of the segmentation task. In contrast, deep learning models, especially DeepLabV3Plus variants, achieved higher segmentation accuracy, validating the superior feature extraction and learning capabilities of these models.

2) Development of the Best Performing Model: DeepLabV3Plus_VIT3 emerged as the best-performing model, achieving the highest min mIoU of 0.7775 (see Table II). This model integrates a Vision Transformer with smaller patch sizes (8x8) and additional convolutional layers for better low-level feature extraction. These enhancements enable effective global contextual and local feature capture, resulting in superior segmentation accuracy across turtle anatomical parts.

3) Model Performance Analysis:
High-Quality Images (Large Turtle Presence): In images where turtles occupy a significant portion of the frame, all deep learning models performed well. DeepLabV3Plus_VIT3 excelled in segmenting the head, flippers, and shell, demonstrating the model's efficacy in handling clear segmentation tasks.
Challenging Images (Small or Multiple Turtles): In complex scenarios involving multiple or smaller turtles, performance generally declined. While DeepLabV3Plus_VIT3 performed better than its counterparts, segmentation quality still suffered, highlighting the need for further model enhancements.

4) Project Limitations: Despite its advancements, DeepLabV3Plus_VIT3 still struggles with images that have poor lighting conditions or feature multiple or small turtles. Challenges such as overlapping instances and significant scale variations negatively impact segmentation accuracy. Further enhancements are required to improve the model's robustness and performance in these complex scenarios.

B. Future Work

1) Further Model Optimization:
Exploring Smaller Patch Sizes: Investigate smaller patch sizes (e.g., 5x5) within the Vision Transformer to capture fine-grained details and improve segmentation in complex scenes.
Incorporating Additional Convolutional Layers: Add more convolutional layers to enhance low-level feature extraction and improve segmentation precision for closely situated or overlapping turtle parts.

2) Data Augmentation and Expansion:
Diversifying the Dataset: Expand the dataset to include varied images with multiple turtles, different sizes, and poses to improve generalization.
Advanced Data Augmentation: Use techniques like rotations, scaling, flipping, and occlusion to increase diversity and model robustness. [23]

3) Model Lightweighting:
Optimizing Computational Efficiency: Apply model compression methods [24](e.g., pruning, quantization) to reduce the computational footprint, enabling real-time processing without significant accuracy loss.

4) Integration of Additional Technologies:
Combining Segmentation with Object Detection: Use object detection to localize turtles before segmentation, enhancing accuracy by focusing on detected regions. [25]
Multi-Modal Learning: Incorporate additional sensor data, like depth or infrared imaging, to distinguish turtles from complex backgrounds. [26]

REFERENCES

[1] L. Adam, V. Čermák, K. Papafitsoros, and L. Picek, "SeaTurtleID2022: A long-span dataset for reliable sea turtle re-identification," *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Waikoloa, HI, USA, 2024, pp. 7131–7141, doi: 10.1109/WACV57701.2024.00699. [Online]. Available: https://ieeexplore.ieee.org/document/10484106

[2] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002. [Online]. Available: https://ieeexplore.ieee.org/document/1000236

[3] Y. Linde, A. Buzo, and R. M. Gray, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, 1980. [Online]. Available: https://ieeexplore.ieee.org/document/1056489

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *arXiv preprint arXiv:1505.04597*, 2015. [Online]. Available: https://arxiv.org/abs/1505.04597

[5] L. Chen et al., "DeepLabV3: Rethinking Atrous Convolution for Semantic Image Segmentation," *arXiv preprint arXiv:1706.05587*, 2017. [Online]. Available: https://arxiv.org/abs/1706.05587

[6] Official PyTorch implementation and documentation: DeepLabV3_ResNet50. [Online]. Available at: https://pytorch.org/vision/stable/models/generated/torchvision.models.segmentation.deeplabv3_resnet50.html. [Accessed: Month Day, Year].

[7] L. Chen et al., "DeepLabV3+: Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," *arXiv preprint arXiv:1802.02611*, 2018. [Online]. Available: https://arxiv.org/abs/1802.02611

[8] VainF, "DeepLabV3Plus-PyTorch," GitHub Repository. [Online]. Available: https://github.com/VainF/DeepLabV3Plus-Pytorch

[9] M. Sandler et al., "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *arXiv preprint arXiv:1801.04381*, 2018. [Online]. Available: https://arxiv.org/abs/1801.04381

[10] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *arXiv preprint arXiv:2010.11929*, 2020. [Online]. Available: https://arxiv.org/abs/2010.11929

[11] B. Wu et al., "CvT: Introducing Convolutions to Vision Transformers," *arXiv preprint arXiv:2103.15808*, 2021. [Online]. Available: https://arxiv.org/abs/2103.15808

[12] X. Chen et al., "Emerging Properties in Self-Supervised Vision Transformers," *arXiv preprint arXiv:2104.14294*, 2021. [Online]. Available: https://arxiv.org/abs/2104.14294

[13] Scikit-learn Documentation: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MeanShift.html

[14] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1967, pp. 281-297.

[15] OpenCV Documentation: https://docs.opencv.org/3.4/d1/d5c/tutorial_py_kmeans_opencv.html

[16] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *arXiv:1505.04597*, 2015.

[17] PyTorch U-Net Project: https://github.com/milesial/Pytorch-UNet/tree/master/unet

[18] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org

[19] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations (ICLR)*, 2015. [Online]. Available: https://arxiv.org/abs/1412.6980

[20] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010. [Online]. Available: http://www.cs.toronto.edu/~hinton/absps/reluICML.pdf

[21] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015. [Online]. Available: https://arxiv.org/abs/1502.03167

[22] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010. [Online]. Available: https://link.springer.com/article/10.1007/s11263-009-0275-4

[23] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1-48, 2019. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0

[24] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *arXiv preprint arXiv:1510.00149*, 2015. [Online]. Available: https://arxiv.org/abs/1510.00149

[25] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137-1149, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7485869

[26] T. Baltrušaitis, C. Ahuja, and L. P. Morency, "Multimodal Machine Learning: A Survey and Taxonomy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 423-443, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8237881

[27] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436-444, 2015. [Online]. Available: https://www.nature.com/articles/nature14539

[28] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431-3440. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2015/papers/Long_Fully_Convolutional_Networks_2015_CVPR_paper.pdf