Deep-sea Biota Classification from Remote Operated Vehicle Images Through Efficient Deep Learning Models

Kazi Shaila Meraz*, Nuzhat Tahsin*, R. M. Alvi Amin, Rashedur M. Rahman
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
kazi.meraz@northsouth.edu, nuzhat.tahsin@northsouth.edu, alvi.amin@northsouth.edu, rashedur.rahman@northsouth.edu

***Abstract*—The deep sea has always been a mystery to mankind. High water pressure and poor lighting make exploration difficult. Advancements in robotics and remotely operated vehicle technology have enabled further research on the deep sea and its diverse biota. However, a lack of light restricts the collection of high-quality labeled images. A lightweight image classification model could help by automatically labeling new images directly on the Remotely Operated Vehicles (ROV) using low-powered hardware. In this paper, we train different models on the DeepDive dataset, which consists of deep-sea ROV biota images from the Great Barrier Reef. We found the best balance between performance and model size with EfficientNetB0, achieving an F1 score of 86.06% and 90.66% accuracy with 5.3 million parameters and a size of 21.4 megabytes.**

***Index Terms*—Deep Learning, Deep-sea biota, Multiclass Classification, Image Classification, EfficientNetB0, Lightweight Models**

I. Introduction

Earth's most vast, enigmatic, and biologically rich ecosystems are found in the deep sea. Ninety percent of the ocean is considered deep, with depths surpassing 656 feet (200 meters). Mankind has explored space more than the ocean. Only a quarter of the ocean has been mapped at high resolution. Just a fraction has been surveyed and explored [1]. Life forms in the deep sea are diverse. There are about 250,000 known species, and many more remain to be discovered.

At least two-thirds of the world's marine species are still unidentified [2]. Autonomous underwater vehicles (AUVs) are a major breakthrough in deep-sea exploration. ROVs and AUVs generate high-quality image and video data from diverse ocean ecosystems. These state-of-the-art vehicles are expensive, but technical obstacles remain. The underwater environment is the largest challenge in collecting quality data. As a result, a range of models and systems is increasingly used. These models focus on Underwater Image Enhancement (UIE), restoration, classification, or object detection. There is still a need to develop and optimize systems. Challenges such as imbalanced datasets, low-quality images, degradation, and small dataset sizes remain under-addressed. Our work is based on trial-and-error with a small dataset to determine the best data preprocessing. This ensures quality training data and supports a system with optimized classification models while maintaining generalization.

*These authors contributed equally to this work.

Deep-sea images are severely distorted due to the underwater environment. Li et al.(2025) developed a highly effective underwater image enhancement(UIE) model to prevail over such challenges. They aimed for color accuracy, visual clarity, and structural details. DeepSeaNet, their proposed framework, enhanced the UDnet framework in integrating Vision Transformer (ViT) as the feature extractor. To produce more realistic images and to ensure structural consistency, they added the MCOLE score module in the multi-component loss function. The model was evaluated using multiple datasets, where it achieved avant garde results. It presented a 20%-40% improvement over baseline models with an average SSIM of 0.901 and a maximum PSNR of 28.96 dB [3].

Following the same principle, Lopez-Vazquez et al. (2023) improved the automated classification of deep-sea species. They also accepted that pre-processing is a crucial step when working with underwater images. They designed a two-stage pipeline. The residual convolutional neural network was trained to improve image quality overall from crawler footage. The improved dataset was then used to train classifiers. With an accuracy of 79.44% on the enhanced dataset, their best performing model was a Deep Neural Network [4].

The detecting architecture or classifiers need to be robust to handle such complexities. Long et al. (2025) focused on this issue. They aimed to create an object detection architecture for ecological monitoring in polymetallic nodule fields. Their novel architecture, the BM-YOLO architecture, improves the existing YOLOv8 by incorporating a deformable convolutional backbone, a transfer module, and a varifocal loss function to address the class imbalance prevalent in most deep-sea datasets. Their new POBM dataset was used to train this specialized model. It outperformed existing models with an mAP50 of 85.9% [5].

Feng and Jin (2024) proposed CEH-YOLO with a High-Order Deformable Attention module integrated, an enforced Spatial Pyramid Pooling-Fast module, and to refine predictions, a Composite Detection module. The DUO and UTDAC2020 datasets were used to evaluate the model. The model achieved state-of-the-art mAP scores for both lightweight designs (88.4%) and real-time speed (87.7%) [6].

Saravanan Vadivazhagan (2025) also aimed for robustness as well as classification accuracy. They used bio-inspired optimization to refine their model. Improved the existing Enhanced YOLOv5 (EY5) architecture with bio-inspired Bearded

Dragon Optimization (BDO) algorithm and developed BeardY-OLO. Evaluated using the LSUI dataset, it achieved a classification accuracy of 79.87% and an F1-score of 79.81% [7].

The quality of data is vital for the success of any architecture. This was understood by Iyer et al. (2025). They worked to develop a coherent and accurate system for automating the detection and quantification of deep-sea benthic life. They created the DeepSea dataset. It is a collection of 2,825 curated images with 20,076 annotated instances across 15 morphospecies classes from North Atlantic deep-sea environments. They combined AdamW and SGD optimizers as well as data transformations to increase robustness. Their model achieved a mAP50 of 0.84 [8].

Reddy et al. (2025) used many of these elements and presented a comprehensive framework for fish species detection and classification. Among the evaluated YOLO models(v8-v12), YOLOv9 performed the best with an average precision of 0.9236 and F1-score of 0.9120 on raw images [9].

This paper has been divided into different sections. The "Proposed System" section presents a detailed representation of the proposed system including the dataset, augmentations, the synthesis done on it, and the models that were trained on it. The "Result and Analysis" section covers the performance metrics and sizes of different models with the rationale behind deciding on a superior model. The "Conclusion" covers some of our limitations and how this work can be extended in the future.

II. Proposed System

We propose a deep learning multi-classification system using the Deepdive dataset, captured by ROVs in deep-sea environments. This system classifies marine organisms and substrates, including corals, sponges, and algae, to assist in ecological surveillance and habitat mapping. We use various data augmentation techniques during training and increase the size of the training set by augmenting it.

A. Dataset

The Deepdive dataset was used for our study. It was captured by Remotely Operated Vehicles (ROVs) and is publicly available. It contains 3,994 images of deep-sea biota, split into 60:20:20 for training, validation, and testing. Each folder contains 33 subfolders, each representing a different benthic class. The training set contains 2,667 images, the validation set 667, and the test set 660. Folder organization under sub-folders makes it easier to select specific classes for model training. All sub-folders have uniform names for management.

To create a uniform, structured set of images for classification, the data were preprocessed by excluding classes with fewer than 15 samples to reduce extreme imbalance, leaving 33 individual classes. Even after censoring, the data remains imbalanced, with a largest-to-smallest class ratio of approximately 33:1. To balance representation, images were partitioned randomly into the training, validation, and test sets. All images were converted to a uniform resolution of 250 × 250 pixels via bilinear interpolation, and pixel values were normalized from the original [0, 255] range to [0, 1] for compatibility and uniformity with deep learning classification models.

Fig. 1. Sample images from the dataset

B. Dataset Preprocessing

We decided to re-split the dataset to ensure balanced image distribution across the training, validation, and test subsets. Even though the authors tried to balance the dataset. Still, the dataset had a severe class imbalance and inconsistent image counts across folders. As a result, this could bias the model toward the majority classes. To ensure a fair evaluation, we re-split the dataset using stratified random sampling, ensuring that each subset contains images from all 33 classes. We have split the combined dataset into 70% for training, 15% for validation, and 15% for testing. After re-splitting, the dataset contains 2,697 training images, 578 validation images, and 578 test images. This strategy ensures a balanced dataset split, enabling accurate model training and unbiased performance estimation across all classes.

We balanced the training data set by ensuring that each class contained exactly 1000 images. We used an augmentation function that introduces variation via random horizontal flipping, rotations of 0°, 90°, 180°, or 270°, and small random color enhancements to create different augmented images. For each class, all original images were copied to the output folder, then the number of available images was counted, and the augmentation was repeated in a loop until 1000 images were obtained. The process ensures class balance, preventing model bias against large classes and improving training stability and performance. Fig. 2 shows a comparison of sample images before and after balancing the training data. The first displays the original, imbalanced image counts per class, and the second shows the augmented, balanced version. This illustrates how the data augmentation helps to create a more uniform class distribution.

Fig. 2 shows a comparison of sample images before and after balancing the training data. The first one displays the original, imbalanced image counts per class and the second one shows the augmented and balanced version. This illustrates how the data augmentation helps to create a more uniform class distribution.

To further augment the training set, we applied different approaches. We combined data augmentation techniques, including random flips along the vertical and horizontal axes, rotations (0°, 90°, 180°, 270°), random resizing with cropping

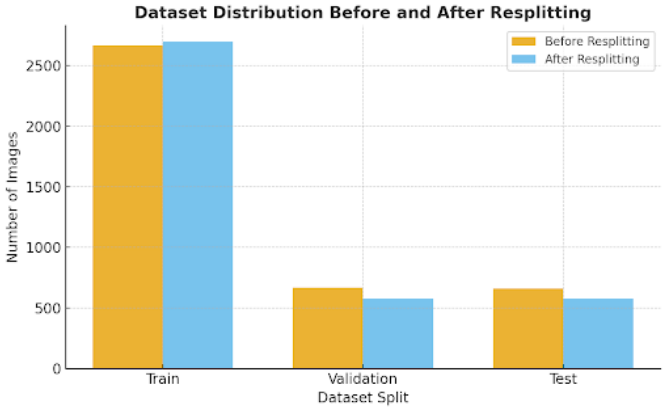


Fig. 2. Dataset Distribution

TABLE I
DATA AUGMENTATION TECHNIQUES AND PARAMETERS

Technique	Parameter
Random crop + resize	Output size = 128 × 128, Scale = (0.8, 1.0)
Random horizontal flip	Probability $p = 0.5$
Random vertical flip	Probability $p = 0.5$
Random rotation	Degrees = 0, 90, 180, 270
Brightness jitter	Factor = 0.8–1.2
Contrast jitter	Factor = 0.8–1.2
Saturation	Factor = 0.8–1.2
CutMix patch	Patch size = random 30% of image area

to a uniform 128×128 resolution, and color jitter with brightness, contrast, and saturation modifications.

Additionally, a CutMix-based method was applied. Such augmentations increased intra-class diversity, reduced overfitting, and ensured equal sample counts across all classes, ultimately improving model generalization. The generated dataset was verified to ensure that all classes had the desired number of samples.

C. Applied Deep Learning Models

EfficientNet-B0: EfficientNet-B0 is a baseline model which is trained using ImageNet-1k. EfficientNet-B0 was developed using neural architecture search using MBConv blocks, squeeze, and excitation optimization to optimize accuracy and efficiency. This model has about 5.5 million parameters.

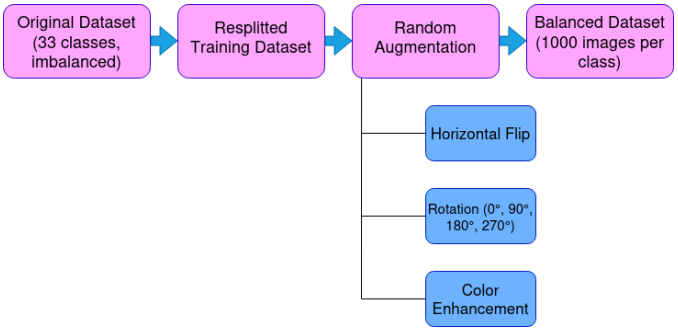


Fig. 3. Class Imbalance Handling Pipeline

EfficientNet-B0 uses significantly fewer parameters and computations compared to traditional CNNs while offering state-of-the-art accuracy. It is an excellent option for mid-sized dataset because of its efficiency and scalability. To ensure faster training the model was resized to 128×128 without sacrificing sufficient resolution for feature extraction.

ResNet18: ResNet18 is a deep residual learning-based convolutional neural network of 18 layers. It effectively solves the vanishing gradient problem, allows gradients to flow directly through identity mappings and enables deeper networks to train efficiently by utilizing skip connections. This model is a powerful tool for image classification tasks and can be further extended to more complex versions like ResNet34, ResNet50, etc. According to stability and generalization power, ResNet architectures are strong baselines for image classification tasks. Use of ResNet-18 allowed one to verify how increased depth influences performance on the given dataset, especially when the number of samples per class is limited.

ResNet34: It is a member of the Residual Networks family. The skip connections allow the network to bypass one or more layers. This mitigates the vanishing gradient problem for very deep networks. The model, as its name suggests, has 34 layers and contains about 21.8 million parameters. This offers balance between performance and resource usage. It is an extension of ResNet18 architecture. According to stability and generalization power, ResNet architectures are strong baselines for image classification tasks. Use of ResNet-34 allowed one to verify how increased depth influences performance on the given dataset, especially when the number of samples per class is limited. Resnet-34 is a robust, interpretable model which can generalize well across image domains.

MobileNetV2: MobileNetV2 uses inverted residuals and linear bottlenecks in a bid to maximize efficiency by reducing multiply-add operations. It consists of depth wise separable convolutions and narrow intermediate layers, thus enabling high accuracy with low computational complexity.MobileNetV2 is commonly used for tasks like image classification, object detection, and semantic segmentation. MobileNetV2 is more suitable to be deployed on resource-constrained or real-time systems, e.g., edge or mobile devices. It was selected as it is light in terms of architecture and fast in inference, making it a viable for efficient model deployment in the future.

DenseNet-121: For addressing the challenges like vanishing gradients, feature reuse, and parameter efficiency in deep learning, DenseNet-121 is a commonly used model which is a variant of the Dense Convolutional Network. It performs very well on tasks like image classification, object detection, and semantic segmentation. Each layer of DenseNet121 receives input from all preceding layers. DenseNet121 has 121 layers, including dense blocks and transition layers, which makes it efficient computationally without losing accuracy. The DenseNet-121 model excels in the task of learning detailed visual information, especially in cases where inter-class variation is subtle.

DenseNet169: DenseNet169 is the extension of densenet-

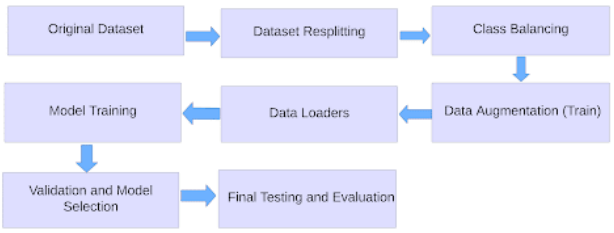


Fig. 4. Complete Pipeline Overview

121.By increasing the number of dense blocks and transition layer, densenet121 shows deeper hierarchical feature extraction. The dense connectivity pattern stays stable which provides enhanced gradient flow and improved generalization. To check whether the additional dense layers improved classification accuracy or not, the deeper structure like DenseNet169 was chosen. Its enhanced capacity makes it suitable for datasets which contain intricate and overlapping visual features.

We applied different deep learning models under the unified training pipeline to evaluate their performance. By using ImageNet pretrained weights, all models were initialized using the same data preprocessing, augmentation, and optimization methods.This consistent setup guaranteed that performance arose primarily from architectural differences rather than from training inconsistencies. We fine tuned each model by using the Adam optimizer with a learning rate of 1e-4, cross-entropy loss, batch size of 32, and early stopping on validation loss to prevent overfitting.

III. Results and Analysis

We implemented transfer learning through training a total of 6 foundational models spanning 4 different architectures on our dataset, focusing on smaller models for optimal deployability on a wide range of ROVs (Remote Operated Vehicles), even those with less on-board computational resources. In this section, we will describe the performance of the models and the reasoning behind deciding on EfficientNetB0 to be the best model as well as the convergence behavior. In all of our evaluations we use macro values so that classification performance is valued equally for all classes.

A. Inference Performance Analysis

Precision: Precision is the measure of how many predictions of a model were true out of all the predictions for a particular class. It is calculated as :

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

where:

- TP = True Positives (correctly predicted positive instances)
- FN = False Positive (negative instances incorrectly predicted as positive)

Macro precision: Macro precision is the average precision across all classes regardless of class imbalance. This is done by giving equal weights to each class during averaging. Macro precision is calculated as:

$$\text{Macro Precision} = \frac{1}{C}\sum_{i=1}^{C}\text{Precision}_i \tag{2}$$

where:

- C = number of classes
- Precision_i = precision for class i

The trained models **Resnet18, DenseNet121, EfficientNetB0, MobileNetv2, Resnet34, DenseNet169** achieved macro precision scores of **83.41%, 87.82%, 86.76%, 85.95%, 84.65%, 84.38%** respectively on the test set. We can see that **DenseNet121** was the most precise in classifying the biota which is closely followed by **EfficientNetB0**, trailing by just **1%**.

Recall: Recall is used to measure how many actual instances of a class are being recognized or classified by the model. Calculation of recall is done by :

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

where:

- TP = True Positives (correctly predicted positive instances)
- FN = False Negatives (positive instances incorrectly predicted as negative)

Macro Recall: Macro recall is the equal weighted average of recall across all of the classes. It is calculated as:

$$\text{Macro Recall} = \frac{1}{C}\sum_{i=1}^{C}\text{Recall}_i \tag{4}$$

where:

- C = number of classes
- Recall_i = recall for class i

The highest macro recall score achieved is **87.77%** by **EfficientNetB0**, leading by **1.5%** from **DenseNet169** which has a macro recall score of **86.21%**. **Resnet18, DenseNet121, MobileNetv2, ResNet34** achieved scores of **84.60%, 86.08%, 85.98% and 85.83%** respectively.

F1 Score: F1 score is used to quantify the balance between precision and recall of a class. It is the harmonic mean of the two which is calculated as:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

where:

- $\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$ (proportion of predicted positives that are correct)

- Recall = $\frac{TP}{TP+FN}$ (proportion of actual positives correctly identified)
- TP = True Positives, FP = False Positives, FN = False Negatives

Macro F1 Score: The equal weighted average of F1 score for all the classes is known as the Macro F1 score which is defined as:

$$\text{Macro F1} = \frac{1}{C}\sum_{i=1}^{C}\text{F1 Score}_i \tag{6}$$

where:

- C = number of classes
- F1 Score$_i$ = F1 score for class i

EfficientNetB0 leads in terms of Macro F1 score with a score of **86.06%**. The rest of the models, **Resnet18, DenseNet121, MobileNetv2, ResNet34 and DenseNet169** are trailing with Macro F1 scores of **82.64%, 85.63%, 84.07%, 84.40% and 84.15%** respectively.

For model performance, we have chosen the Macro F1 score to be the deciding factor since our dataset is severely imbalanced and the Macro F1 scores captures the models' classification performance better than just the accuracy score. Accuracy score can be heavily skewed by a higher number of samples in the common classes which all of the models do a good job of classifying.

For our goal of having models be potentially hosted on ROVs (Remote Operated Vehicle), we also need to prioritise models that are lightweight, resource efficient and have fast inference performance. We have tested models of varying parameter counts and architectures to decide what type of model performs well while having a small footprint. From Table III we can see that the smallest model out of the 6 models is **MobileNetv2** with a parameter count of **3.5 million** and size of **13.6MB**. However, its macro F1 score trails behind the second smallest model here, **EfficientNetB0** by **2%**. The **EfficientNetB0** model has **5.3 million** parameters and a footprint of **21.4MB**, but performs the best in terms of macro F1 score. We think that **EfficientNetB0** strikes a good balance between being performant and being less resource intensive.

ROC-AUC: ROC-AUC or Receiver Operating Characteristics-Area Under Curve represents the area under the ROC curve which plots true positive rate and false positive rate of classes at various thresholds. This helps us measure a model's ability to differentiate between classes. The ROC-AUC score is calculated by:

$$\text{ROC-AUC} = \int_0^1 \text{TPR}(\text{FPR})\, d\text{FPR} \tag{7}$$

where:

- TPR (True Positive Rate) = Recall = $\frac{TP}{TP+FN}$
- FPR (False Positive Rate) = $\frac{FP}{FP+TN}$
- TP = True Positives, FN = False Negatives, FP = False Positives, TN = True Negatives

Macro ROC-AUC: This is the weighted average of the ROC-AUC score for all the classes. This is calculated by:

$$\text{Macro ROC-AUC} = \frac{1}{C}\sum_{i=1}^{C}\text{ROC-AUC}_i \tag{8}$$

where:

- C = number of classes
- ROC-AUC$_i$ = ROC-AUC for class i (typically computed using one-vs-rest)

The **EfficientNetB0** achieved a very high ROC-AUC score of **99.3%** signifying the model's ability to distinguish between classes. Fig. 5 visualizes the ROC-AUC scores at various thresholds of the top 6 classes in terms of sample count in the test set.

TABLE II
PERFORMANCE METRICS OF DIFFERENT MODELS

Model	Accuracy	Precision (Macro)	Recall (Macro)	F1 (Macro)
ResNet18	0.9187	0.8341	0.8460	0.8264
DenseNet121	0.9118	**0.8782**	0.8608	0.8563
EfficientNetB0	0.9066	0.8676	**0.8777**	**0.8606**
MobileNetV2	0.9066	0.8595	0.8598	0.8407
ResNet34	0.9187	0.8465	0.8583	0.8440
DenseNet169	**0.9273**	0.8438	0.8621	0.8415

TABLE III
MODEL PARAMETER COUNT AND SIZE

Model	Parameters (Millions)	Size (MB)
ResNet18	11.7	46.0
DenseNet121	8.0	32.3
EfficientNetB0	5.3	21.4
MobileNetV2	3.5	13.6
ResNet34	21.8	87.3
DenseNet169	14.1	57.4

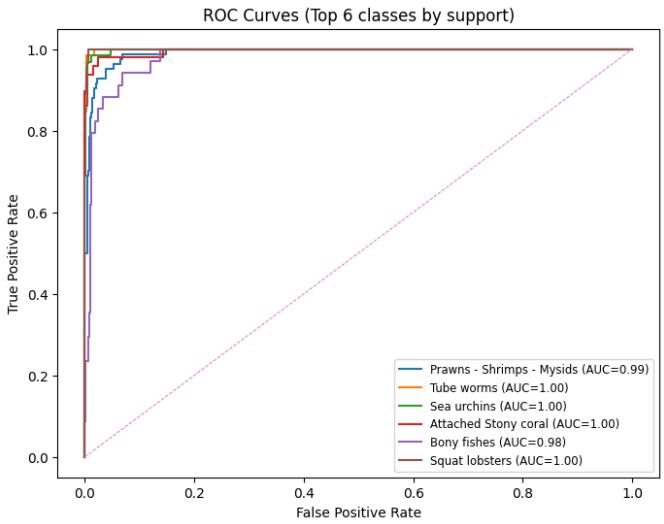
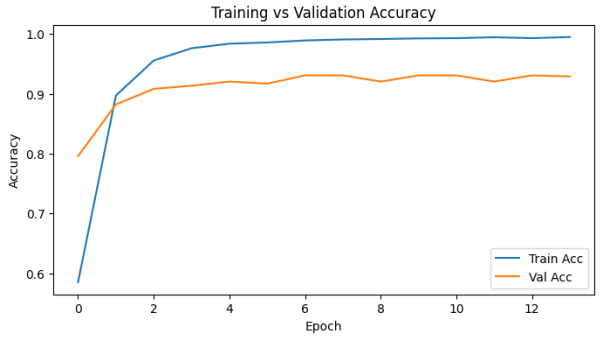
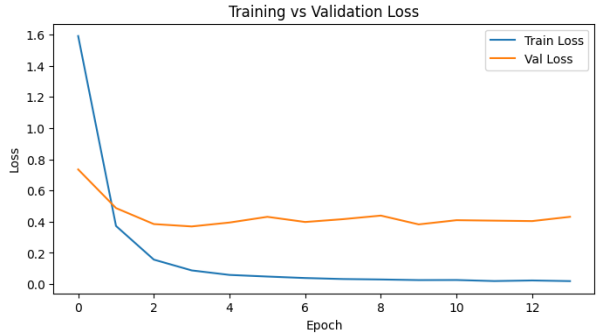


Fig. 5. ROC-AUC of EfficientNetB0



(a)



(b)

Fig. 6. Training Convergence. (a) Accuracy, (b) Loss

B. Convergence Analysis

The EfficientNetB0 model reaches convergence very early in the training loop as evident from Fig. 6. The classification performance on the validation set reaches a plateau after the second epoch. The model continues training until epoch 14 as we have set the patience for our early stop mechanism to be 10. Since, in the past 10 epochs the validation loss does not improve, the best model weights are restored and training is halted.

This behavior can be attributed to the limited availability and variability of images in the dataset. The model learns what it could very early on from the dataset.We can see in Fig. 6, any training afterwards causes overfitting as seen from the accuracy and loss differences.

IV. Conclusion

This study presented a deep-learning-based multi-class classification system for the identification of deep-sea biota. This was done via training the models using the Deepdive dataset. The dataset, although lacking in some respects, was an excellent choice for this work, as it included images of various species. We developed a robust pipeline that addressed the challenges we faced, such as class imbalance. There was extensive use of data augmentation techniques. Our approach to systematically processing data before training ensured that the models were trained on high-quality data. The ImageNet dataset is used to train the foundational models for our transfer learning. The lack of rare species data in the ImageNet dataset also makes it harder to fully capture the features of the marine organisms, resulting in poor classification performance for those species. The species' environment is also not considered when classifying. In the future, our work will focus on overcoming the stated limitations and also experimenting with several key methodologies. We plan to address the class imbalance in our dataset and improve the visual quality using image enhancement methods. We intend to incorporate Explainable AI (XAI) methods to make our system more transparent, rather than simply acting as a black box. This will improve model interpretability, helping the user make valuable decisions. Depending on data availability, we also plan to include environmental metadata to make the system more robust. Then, to improve deployability, we will use knowledge distillation techniques to produce smaller, faster models with minimal loss of accuracy. The successful implementation of systems like ours will pioneer new approaches and take this discipline in a new direction. These tools will become indispensable in the near future for assisting researchers with large-scale habitat mapping and identification, thereby contributing to the preservation and understanding of unexplored deep-sea ecosystems.

References

[1] NOAA Ocean Exploration, "Why ocean exploration matters." https://oceanexplorer.noaa.gov/why-exploration-matters/, Aug. 2024. Published: August 13, 2024.

[2] C. Paşca Palmer, "Marine biodiversity and ecosystems underpin a healthy planet and social well-being," *UN Chronicle*, vol. LIV, May 2017.

[3] J. Li, Y. Ouyang, H. Wang, D. Wu, and Y. Pan, "Deepseanet: An efficient uie deep network," *Electronics*, vol. 14, no. 12, pp. 2411–2411, 2025.

[4] V. Lopez-Vazquez, J. M. Lopez-Guede, D. Chatzievangelou, and J. Aguzzi, "Deep learning based deep-sea automatic image enhancement and animal species classification," *Journal of Big Data*, vol. 10, no. 1, 2023.

[5] G. Long, W. Song, X. Liu, Z. Fang, J. An, K. Liu, Y. Huang, and X. He, "Automated recognition of deep-sea benthic megafauna in polymetallic nodule mining areas based on deep learning," *Ecological Informatics*, vol. 90, p. 103319, 2025.

[6] J. Feng and T. Jin, "Ceh-yolo: A composite enhanced yolo-based model for underwater object detection," *Ecological Informatics*, vol. 82, p. 102758, 2024.

[7] P. Saravanan, "Efficient underwater image classification with enhanced yolov5 and bearded dragon optimization (beardyolo)," *Journal of Theoretical and Applied Information Technology*, vol. 103, no. 12, 2025.

[8] K. H. Iyer, C. M. Marnor, D. W. Schmid, and E. H. Hartz, "Detecting and quantifying deep sea benthic life using advanced object detection," *Frontiers in Marine Science*, vol. 11, 2025.

[9] G. Joshita Reddy, K. Reddy, S. Ruthvik Athota, S. J. Narayanan, B. Perumal, and G. Kumar Nayak, "Deepseavision: Enhanced detection and classification of underwater species," *IEEE Access*, vol. 13, pp. 173347–173367, 2025.