

# Classification and Segmentation of UXO using hand-crafted features and CNNs

**Ahmed Alketbi**

Computer Vision and Robotics Research Institute (VICOROB), University of Girona, Spain.  
Autonomous Robotics, Technology Innovation Institute (TII), United Arab Emirates.

**Abstract:** There is a strong need for technologies to detect, identify and map underwater military munitions and other forms of unexploded ordnance (UXO). Such technologies are crucial for the planning and the execution of remediation actions, such as the removal of the explosives. Recent research efforts focused on using optical images instead of traditional acoustic sensors. One of such efforts, led by the UdG, focused on detecting ammunition in the seabed and classifying reefs using MATLAB-based algorithms. Over time, however, the technology and tools used in the study became outdated. The main goal of this study was twofold: first, to recreate the work using Python libraries, and second, to explore how neural network models could be used for similar tasks. Transitioning from MATLAB to Python posed challenges due to the lack of detailed documentation in the original work, the different behavior of Python libraries compared to MATLAB, and the reduced size of the munition dataset. Despite all that, initial results from using neural networks seemed promising. This study emphasizes the current options and performance machine learning tech offers to tackle UXO detection and segmentation with limited data.

Index Terms: UXO detection, 2.5D features, elevation map, underwater object classification, fusion of 2D and 2.5D features, limited data CNNs.

## 1. Introduction:

Remote sensing has entirely revolutionized our perception and interaction with the world. They enable us to track disasters, monitor deforestation and urban development, and provide insights into vast areas that were previously unimaginable. One fascinating application of remote sensing is its use in classifying delicate ecosystems such as coral reefs and detecting possible threats to those ecosystems.



Figure 1: Seabed munitions from World War II bombs in Tuvalu waters [19]

The ocean floor, often called the Earth's frontier, holds a treasure trove of secrets ranging from shipwrecks to remnants of past wars. Among these there are hazards such as discarded munitions known as "Underwater unexploded Ordnance (UXOs)". Detecting these munitions is crucial for ensuring marine traffic and preserving the delicate balance of our ecosystems. Similarly, coral reefs, often referred to as the "rainforests of the sea," play a role in maintaining biodiversity. It is imperative that we can accurately classify and monitor these reefs in the face of global climate challenges.

There is a growing urgency in accurately detecting and mapping underwater military munitions. These munitions pose significant navigational and safety hazards, especially given that historical data about their locations can be vague and sometimes inaccurate. Traditional methods, including acoustic or metal detection techniques, have their limitations in scope and precision. While optical imagery of seabeds offers a potential solution with its unparalleled spatial resolution, there's a historical challenge: extracting actionable data from these images has been labor-intensive.

A study closely related to our work, titled "Improved Supervised Classification of Underwater Military Munitions Using Height Features Derived from Imagery" [2] made significant advancements by utilizing MATLAB to tackle these challenges. This research stood out not only for its approach but also for producing invaluable results with exceptional accuracy. However, as technology continues to evolve and new programming paradigms emerge, old implementations need to be revisited and updated. The work was developed in 2014 in the framework of a US research project whose consortium included the UdG. The goal of the project was to develop and test a set of processing tools for the detection and classification of underwater marine munitions from optical imagery alone, and to test the benefit of using 3D information as a complement to using texture alone. This work served as the starting point to our work presented in this paper.

Transitioning from MATLAB to Python has become an increasingly strategic decision in computational and engineering domains, especially when versatility and scalability are paramount. First and foremost, Python, being open-source, fosters a broader community-driven ecosystem, which encourages swift advancements and a myriad of specialized libraries. This not only enhances the developmental experience but also promotes more democratized access to cutting-edge tools. When combined with Docker [21], Python's modular nature becomes even more pronounced. Docker containers encapsulate Python applications in a consistent environment, ensuring reproducibility, isolation, and portability across different computing setups without the overhead of traditional virtual machines. This lightweight characteristic is especially pivotal for applications on underwater vehicles. These vehicles, given their stringent power and computational constraints, demand efficient software ecosystems. Python's elegant architecture, combined with Docker's isolation capabilities, ensures that underwater vehicles can process data and execute algorithms in real time without being bogged down by the more cumbersome requirements of environments like MATLAB. In essence, recreating an implementation from MATLAB to Python not only unlocks a more agile and adaptive development landscape but also aligns seamlessly with the evolving demands of modern underwater exploration and automation. The main goals of this study were twofold: to bring this groundbreaking work into the Python environment and to assess the potential of neural networks in these tasks.

Embarking on such a project was not without its difficulties. Switching from MATLAB to Python presented challenges due to differences in how libraries functioned and incomplete documentation from the original work. Additionally, the limited size of the ammunition dataset

posed constraints when training models using neural networks that typically perform better with larger datasets.

Despite these challenges, the project provided insights and highlighted areas for the current possibilities and options that modern machine learning offers to classify and segment UXOs.

This thesis is organized subsequently: Section 2 delves into the contemporary literature surrounding the varied technologies examined for task execution. Section 3 delineates the procedural framework adopted, the tools employed, and their integration for task accomplishment. Section 4 elucidates and critically analyzes the outcomes derived from this work. Conclusively, Section 5 encapsulates the results drawn from the study, while Section 6 offers a perspective on the limitations faced. Section 7 gives insights for further future work.

## **2. State of the Art**

The utilization of satellite imagery for the purpose of underwater object classification has emerged as a significant area of study in recent years. The overarching goal of these endeavors is to harness the capabilities of satellite imagery to accurately detect and classify submerged objects despite the challenges posed by underwater environments.

The pioneering work "Automated Underwater Object Classification using Optical Imagery" by Shihavuddin [1] comprehensively explored the challenges of using optical imagery in aquatic terrains. Factors such as unpredictable underwater light propagation and water turbidity were discussed. Despite these inherent challenges, Shihavuddin managed to pave the way for advanced classification methodologies tailored specifically for optical images, emphasizing various image enhancement techniques.

Building upon this foundational knowledge, the research titled "Supervised Classification of Underwater Military Munitions Using Height Features Derived from Imagery" [2] honed in on the niche area of military munition detection. This study demonstrated the immense potential of using height features extracted from satellite imagery. A robust model for munition detection in underwater environments was proposed by harnessing these features and integrating them with MATLAB.

The study "Automated Detection of Underwater Military Munitions Using Fusion of 2D and 2.5D Features From Optical Imagery" [3] proposes a novel approach, introducing an automated classification system that fuses both two-dimensional (2D) and two-and-a-half-dimensional (2.5D) features. The 2D features focus on texture, while the 2.5D ones encapsulate the geometry of the seabed terrain. To extract these 2.5D features, a unique 3D texture model of the seabed is generated, facilitating the computation of an elevation map. This methodology was put to the test using the Ordnance Reef dataset, obtained off Waianae, Hawaii. This dataset, comprising images of 5-inch shells and seafloor background, brought forth challenges like marine growth on the shells (biofouling), making the differentiation between shells and their surroundings a complex task.

As technology has advanced, so too have the methods and tools available for researchers. There has been a noticeable shift towards more contemporary machine learning solutions, especially the adoption of Convolutional Neural Networks (CNNs). CNNs, with their capacity for image recognition and classification, have shown significant promise in many domains, including underwater object classification. The architecture of CNNs, characterized by their hierarchical layer structure, is particularly suited for tasks where spatial hierarchies and patterns play a critical

role, making them a potentially viable solution for the challenges presented by underwater imagery.

However, while CNNs and other machine learning approaches have gained traction and shown success in various scenarios, their application to the specific challenge of classifying underwater munitions and other objects using satellite imagery remains an area ripe for exploration and refinement. This transition to leveraging machine learning, specifically CNNs, and migrating from traditional platforms like MATLAB to more flexible programming environments such as Python underscores the evolving nature of the field and the continuous pursuit for enhanced accuracy and efficiency.

The fusion of texture and geometric information in the realm of underwater munition detection presents a promising avenue. Preliminary results indicate that this combination enhances the precision of detection, paving the way for safer underwater navigation. While the early outcomes are encouraging, it is crucial to delve deeper, refining the methodologies, and testing them on broader datasets to ensure their efficacy in real-world scenarios.

### **3. Methodology**

Our research methodology was divided into two parts. The primary objective was to replicate the procedures outlined in the study "Automated underwater object classification using optical imagery " [1] but within the Python ecosystem. The second aspect involved exploring network architectures to achieve and potentially exceed the results obtained in previous research while addressing the challenge of having a considerably small dataset for deep learning.

#### **3.1. Project Work Plan**

To tackle the project in an effective manner, the following work plan was devised to avoid potential obstacles and make the most of the timeline:

- 1) Study the original work and research the current state-of-the-art.
- 2) Examine the original MATLAB implementation.
- 3) Learn and install all the possible libraries needed to recreate the original work in Python.
- 4) Implementation of the original work following the structure and steps suggested in the figures in section 3.2
- 5) Testing and comparing results with the original work.
- 6) Exploring the potentials of CNNs by implementation and testing.
- 7) Document the results and the work for future use and publication.

#### **3.2. Original Work Pipelines**

The Original MATLAB work [1] followed a standard machine-learning pipeline shown in Figures 2 and 3 below.

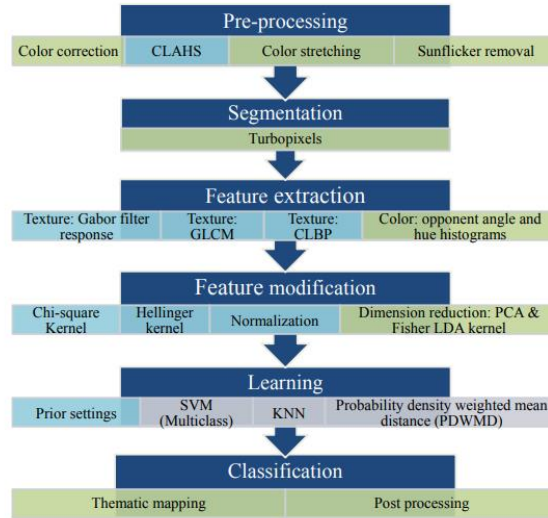


Figure 2: Original Work Pipeline as shown in [1]

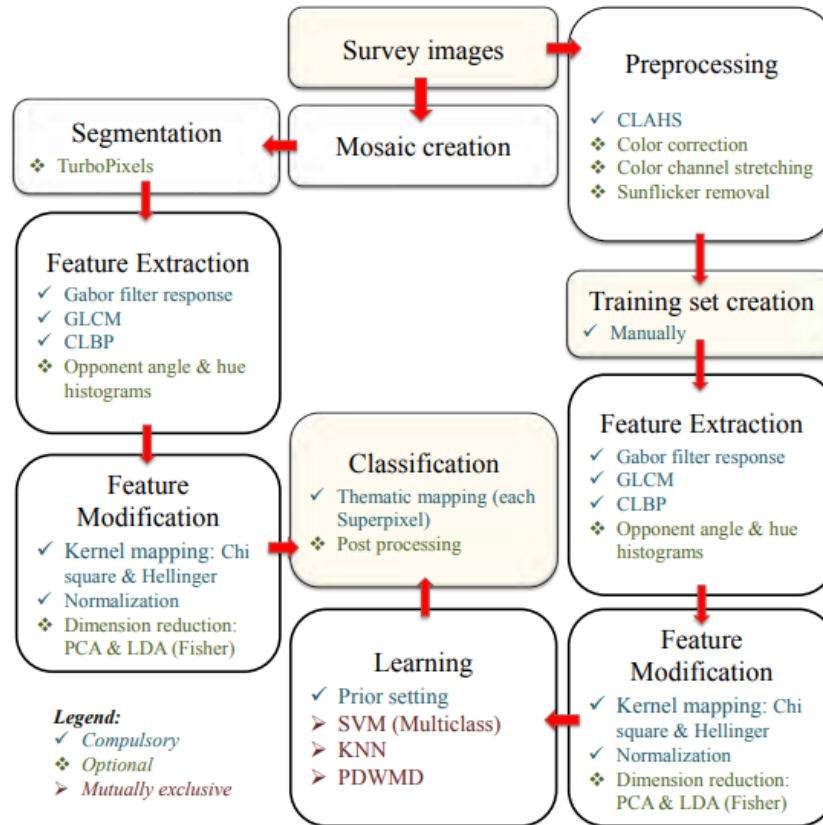


Figure 3: Complete pipeline with mosaic images processing as shown in [1]

Figure 3 shows the complete pipeline when we classify and segment a seabed image. The proper pipeline creates a model for us that can later be used for classification in the left pipeline. The left pipeline segments the image using turbo pixel to create a mosaic that can be applied with the model from the right pipeline to classify and visualize the results using thematic mapping.

### 3.3. Libraries Needed

To replicate the original work, we ended up using the following libraries due to their current reliability and future stability.

- 1) OpenCV [5] is an open-source computer vision and machine learning software library.
  - a) Used for CLAHS, Grayscale image reading, Color correction, and most preprocessing requirements.
- 2) Numpy [4]: It provides support for working with arrays and matrices and mathematical functions to operate on these data structures.
  - a) Efficient and optimized storage of features and labels in data structures.
  - b) Applying matrices and large-scale mathematical calculations.
  - c) Used to replicate the original work 3D feature extractions.
- 3) Scikit: Massive libraries with convenient tools to perform image feature extractions/modifications and machine learning purposes.
  - a) Skimage [7]: Provided implementations for GLCM and other feature extraction operations as well as an alternative to MATLAB's superpixels segmentation called SLIC.
  - b) Sklearn [6]: Provided PCA application and many classifiers for model learning.
- 4) Pytorch [12]: open-source computing library that facilitates efficient matrix computations, possesses automatic differentiation capabilities, and provides tools for parallel computation necessary for CNNs.
- 5) Tensorflow [13]: open-source software library optimized for high-performance numerical computations, especially excelling in handling Convolutional Neural Networks (CNNs) while also offering a comprehensive platform for various machine learning and scientific applications.

### 3.4. Datasets Used

Numerous datasets played a role in the origins work investigation, some of which were also utilized in the study;

- 1) EILAT & EILAT 2 [20]: Both datasets contain coral reefs of Eilat

Name	Number of Classes	Number of Samples	Sample Resolution	Color	Size
EILAT	8	1123	64x64	Yes	Small
EILAT2	5	303	256x256	Yes	Small

Figure 4: General Description of EILAT and EILAT2 Datasets

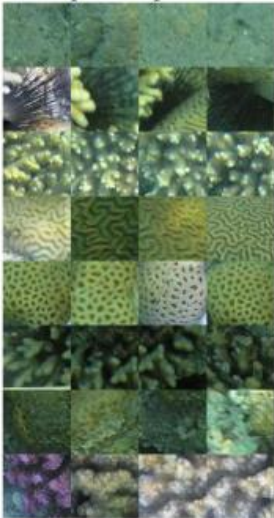
Examples of patches	Class	Number	% in dataset
	Sand	87	7.7%
	Urchin	80	7.1%
	Branch I	23	2.0%
	Brain coral	160	14.2%
	Favid coral	200	17.8%
	Branch II	216	19.2%
	Dead coral	280	24.9%
	Branch III	77	6.8%

Figure 5: Subset of EILAT Dataset taken from [1]

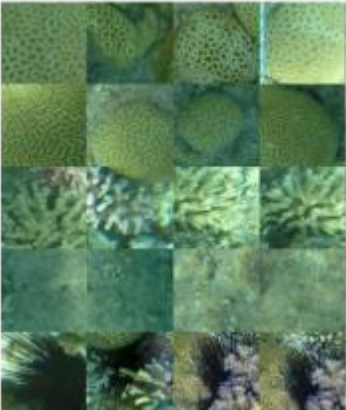
Examples of patches	Class	Number	% in dataset
	Favid coral	89	29.4%
	Brain coral	71	23.4%
	Branching coral	49	16.2%
	Sand	80	26.4%
	Urchin	14	4.6%

Figure 6: Subset of EILAT 2 Dataset taken from [1]

- 2) Primary Dataset: This dataset is found and used by Shihavuddin [1]. It is already preprocessed, and we don't have the original Bomb Dataset. However, the coral reef dataset is a preprocessed EILAT 2 dataset.

Name	Number of Classes	Number of Samples	Sample Resolution	Color	Size
Primary Dataset	5	1123	64x64	Yes	Small
Primary Bomb Dataset	2	10681	128x128	yes	Medium

Figure 7: General Description of the Primary and Primary Bomb Datasets



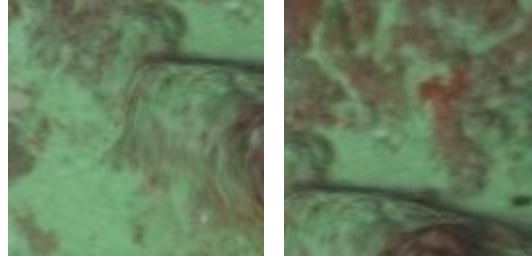


Figure 8: Sample Primary Bomb Images from [1]

### 3) Dataset For CNN use:

From the Primary Bomb Dataset (782 Images), masks were created manually in order to be utilized in classification and segmentation using a CNN. Because some images were preprocessed, only 642 images were clear enough to create a mask. Thus, the final dataset to be used in CNN contained 642 preprocessed images.

### 3.5. Preprocessing & Feature Extraction

To prepare our datasets for analysis, we subjected them to a preprocessing procedure;

- 1) Color Correction: Ensuring representation of underwater objects.
- 2) CLAHS [15]: Contrast Limited Adaptive Histogram Equalization, a vital technique that enhanced contrast in our image datasets.
- 3) Contrast Stretching: Maximizing the range of images to highlight details that may have been unnoticed.
- 4) Sunflicker Removal: Originally intended to reduce interference from sun reflections but ultimately not implemented.

For segmentation. The best option was to replace the "turbo pixels" method from the original implementation with the SLIC segmentation library in Python because it uses the same algorithm in MATLAB's turbo pixels.

To extract features, we used a variety of techniques:

- 1) Gabor Filter Response [14]: A particularly useful method for analyzing textures.
- 2) GLCM [16]: (Gray Level Co-occurrence Matrix) It evaluates the relationship between pixels to describe image texture.
- 3) CLBP [17]: (Completed Local Binary Pattern) This technique was used to characterize texture.
- 4) Color Histograms: These charts display the distribution and subtle variations of colors within images.
- 5) 2.5D Features: We integrated height features, which are believed to improve accuracy in detecting munitions.

After extracting the features, we further refined them through the following:

- 1) Kernel Mapping: Adjusting the features to align with the requirements of machine learning algorithms.
- 2) Normalization: Ensuring that all features were on a harmonized scale before training our model.
- 3) Dimension Reduction using PCA, Simplifying the feature space to enhance efficiency.

For the learning process, the following is a series of classifiers in the original work:

- 1) SVM (Support Vector Machine) [6]: a method that searches for the best-dividing line or, in higher dimensions, a hyperplane to distinguish data into two classes.



- 2) KNN (K Nearest Neighbors) [6]: an algorithm where an object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

- 3) PDWMD (Probability density-weighted mean distance) [18]

Lastly, for visualization, Trained thematic mapping helped facilitate classification by translating model predictions into actionable conclusions.

### 3.6. Implementation of Neural Networks

Furthermore, we also employed a U-net [8] inspired architecture for binary segmentation of UxOs, modified to suit our dataset. U-Net [8] was originally designed for semantic segmentation of biomedical images with a lack of sufficiently large datasets for training. Its symmetric encoder-decoder design with skip connections preserves high-resolution features and merges them with deeper contextual insights. The implemented model follows an encoder-decoder structure with concatenating skip connections, as shown in Figure 9. Bilinear interpolation was used for upsampling operations, whereas Average pooling was used for downsampling operations. Batch Normalization [10] and a ReLU [11] non-linearity were applied after all convolutions. Additionally, to minimize the number of parameters while enabling multi-scale feature extraction, multiscale depthwise-separable convolutions inspired by Rajani et al. [9] were utilized instead of full convolutions in encoder and decoder blocks.

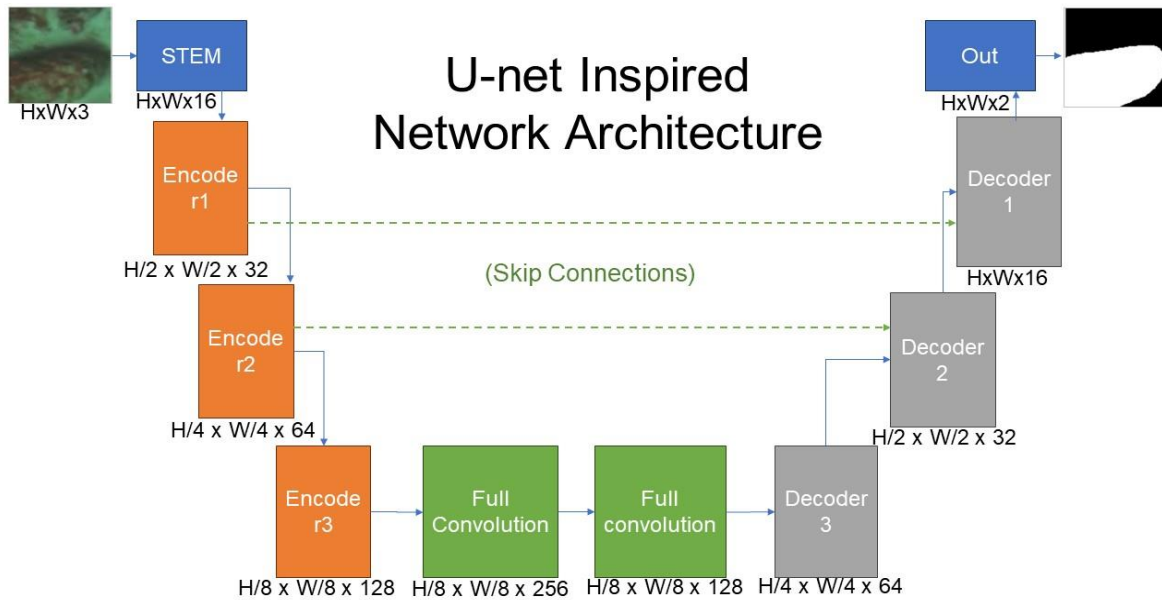


Figure 9: Visualization of the U-net[8] inspired network that was implemented and tested in this thesis

### 3.7. Challenges

We encountered some challenges when transitioning from MATLAB to Python. The original MATLAB code, although it worked well, lacked comments. It also had confusing variable names and an overall complex organizational structure. Converting this code accurately into Python required effort and careful interpretation. The size constraint of the UXO dataset made annotations harder to produce and limited the viable options of NNs.

#### 4. Results

While certain study aspects did not align with our initial expectations, we encountered unexpected yet insightful findings in other areas. All contributed to the purpose of this research of comparing classification and segmentation approaches of UXOs. To assess our approach, we used the same testing metrics in the original work. We employed 10-fold cross-validation, wherein 90% of the image patches served as training data and 10% as testing data across ten iterations. This method of cross-validation minimizes the risk of over-fitting and potential result bias.

As for the performance evaluation of the U-net-inspired neural network, we used mean IOU graphs and loss graphs in both training and validation. We also utilized visual tests to see how the results compare to the ground truth masks.

##### 4.1. Original Reimplementation Results vs. Original Work in Coral Reef Classification

This test aims to replicate and compare some documented tested pipelines in the original work. These results report our finding when we fixed GLBP + GLCM + Gabor + Hue as our feature extraction methods. Also, Used PCA and normalization as our feature modification methods. Finally, we used a KNN model as our classification method. We changed the Preprocessing approach in each step to see if our finding would match the original work.

##### ELIAT Dataset:

Test Pipeline	Original MATLAB Work Accuracy	Python Reimplementation Accuracy
No preprocessing	90.7%	45.5%
CLAHS	92.9%	76.2%
Color correction + CLAHS	NA	71.9%
Color correction	NA	60.8%
Color stretching	67.1%	48.6%
Color Stretching + CLAHS	91.4%	76.4%
Color correcting and stretching + CLAHS	NA	66.4%

Figure 10: Table comparing MATLAB implementation [1] results with this reimplementation results in EILAT Dataset

**ELIAT2 Dataset and Primary Dataset:**

Test Pipeline	Original MATLAB Work Accuracy (ELIAT2 Dataset)	Python Reimplementation Accuracy (Primary Dataset)
No preprocessing	80.1%	49.1%
CLAHS	87.4%	84%
Color correction + CLAHS	NA	82.4%
Color correction	NA	67.9%
Color stretching	62.9%	55.6%
Color Stretching + CLAHS	81.7%	84.4%
Color correcting and stretching + CLAHS	NA	74.4%

Figure 11: Table comparing MATLAB implementation [1] results with this reimplementation results in EILAT2 Dataset and Primary Dataset ( a preprocessed EILAT2 Dataset)

#### **4.2. Original Reimplementation Results vs. Original Work in Bomb Classification using 2D and 2.5D Features**

Utilizing the pipeline from section 4.1 and combining the recalculated 2.5D height features from the corresponding elevation maps, the following results were obtained:

- **2D Features only:** Overall accuracy is 92.2%, with UXOs identification at 10.6%.
- **2.5D Features only:** Overall accuracy reaches 93.8%, and UXOs identification significantly increases to 45.3%.
- **Combined 2 + 2.5D Features:** The model shows an overall accuracy of 92.3% and UXO identification at 11.8%.

Diving further into the results, we can predict that the model will not give us high accuracies in identifying UXOs due to the previous results without the 2.5D Features. Because of the vast class imbalance (roughly 10k background images to 1k Bomb images), we have high model accuracy, but the actual effectiveness in identifying UXOs shined when using 2.5 D features only. While the pipelines are the same as the original work, the individual parameters of each step are not tuned like the original work. Due to the lack of documentation in the original work, moving on to investigate the performance of CNNs was a better use of time instead of tweaking parameters furthermore.

### 4.3. CNN Results using images only

Our U-net inspired neural network vastly outperformed the MATLAB reimplementation in several key aspects. The model delivered significantly higher mIOU (mean Intersection over Union) and showcased superior processing speed, making it a more efficient alternative. The ease of tweaking and optimizing the model parameters rendered it a more adaptable approach compared to the MATLAB reimplementation. Initial results of CNNs proved to be promising with bomb classification tasks. The best results were obtained using the following parameters:

CNN Run Parameter	Value
# of Epochs	300
Batch Size	16
Learning Rate	$1 \times 10^{-3}$
Convolution Type	Multi separable
Downsampling Type	Average pooling
Upsampling Type	interpolate
Skip Connections Type	concatenate

Figure 12: Best Overall Learning parameters using the U-net inspired CNN without elevation maps.

The model had 773,874 total parameters with a mean IOU accuracy of 85.4342%.

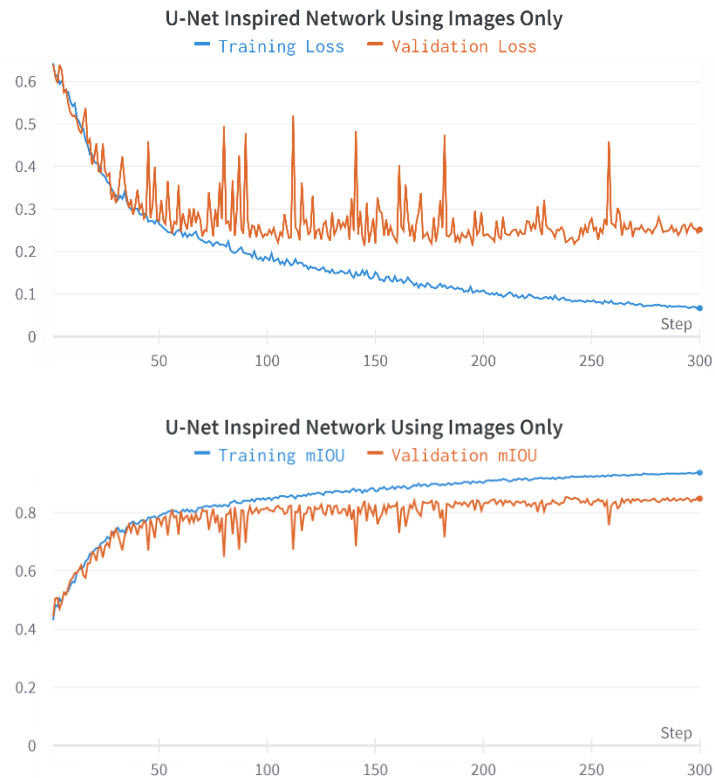


Figure 13: Graphs showing the mean IOU and Loss of training and validation of the best model without using elevation maps.

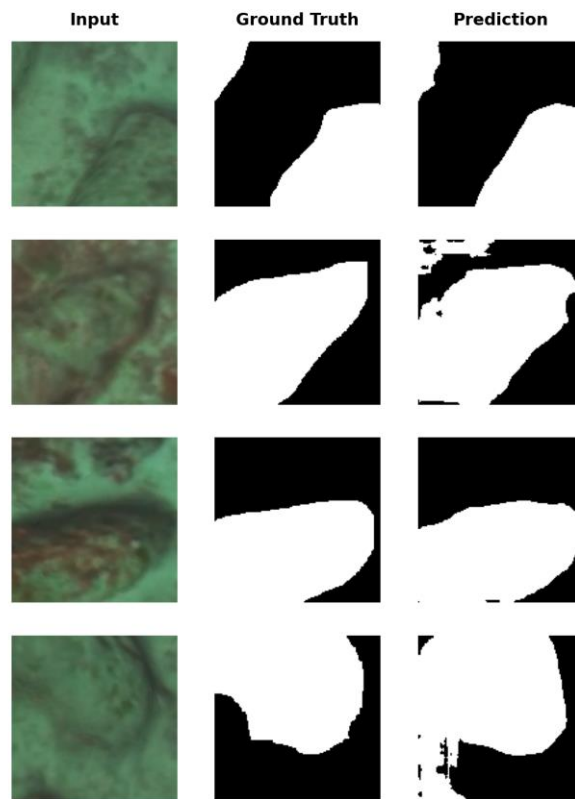


Figure 14: Visualization of the accuracy of the current best overall model without using elevation maps.

The model performed extremely well despite the small dataset (642 images). With further tuning of the network and taking measures to avoid overfitting, it can achieve higher accuracy. As shown in Figure 13, the validation loss graph starts fluctuating, which can signify overfitting/model memorizing training data.

#### 4.4. CNN Results using Images and Elevation Maps

Further tests were conducted to investigate the benefit of elevation maps on in the neural network. The following parameters were used for two runs, one with elevation maps and one without:

CNN Run Parameter	Value
# of Epochs	100
Batch Size	16
Learning Rate	$1 \times 10^{-4}$
Convolution Type	Multi separable
Downsampling Type	Average pooling
Upsampling Type	interpolate
Skip Connections Type	concatenate

Figure 15: Best Overall Learning parameters using the U-net inspired CNN with elevation maps.

Adding elevation maps gave significantly higher accuracies with 774,018 parameters and a mean IOU of 99.75%.

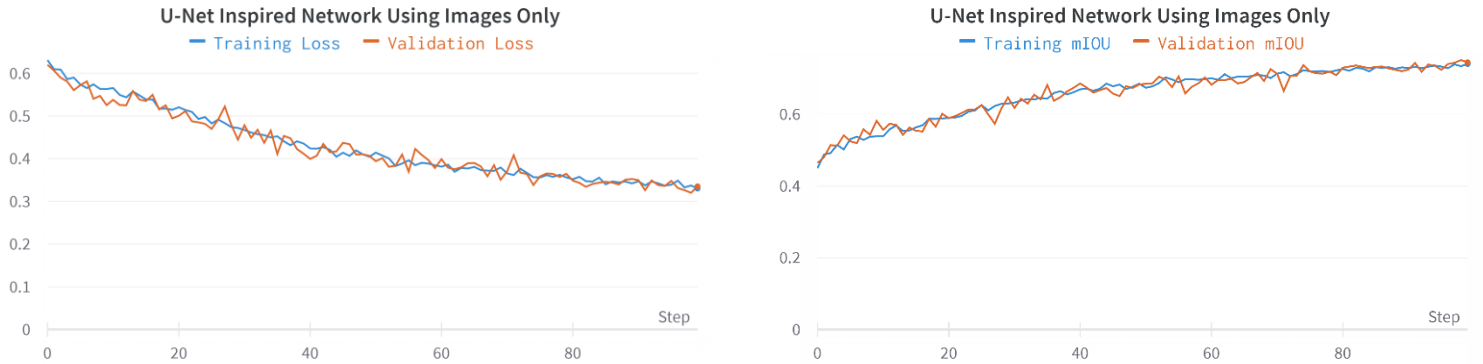


Figure 16: Graphs showing the mean IOU and Loss of training and validation of the best model with elevation maps without supplying the elevation maps

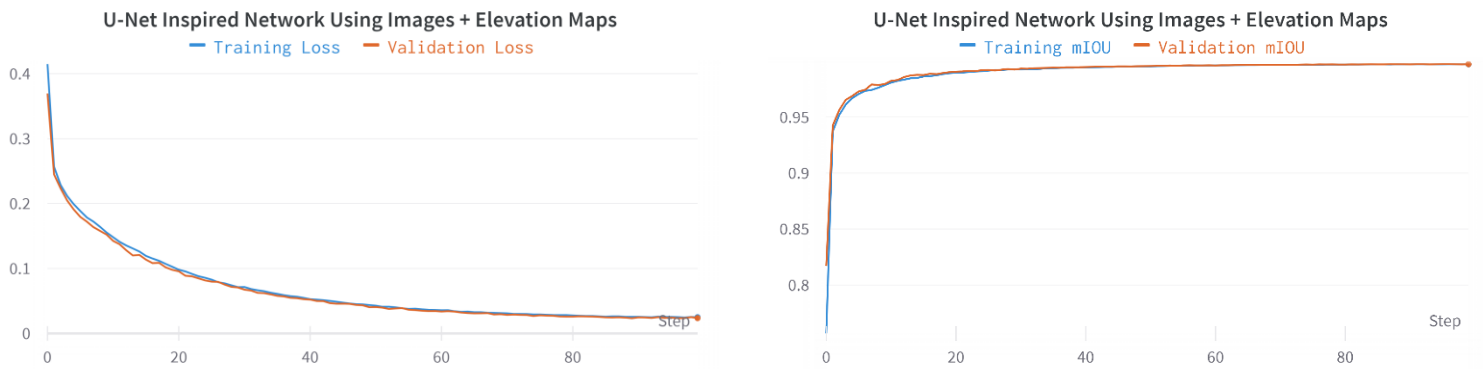


Figure 17: Graphs showing the mean IOU and Loss of training and validation of the best model with elevation maps with supplying the elevation maps

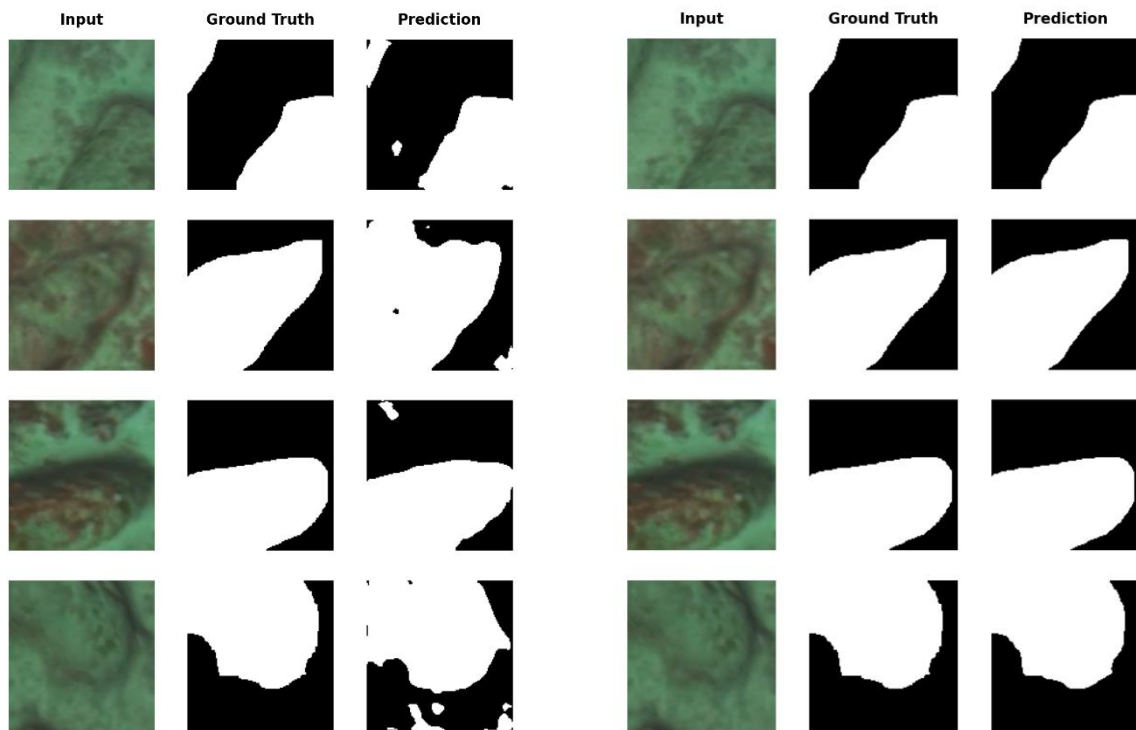


Figure 18: Visualization of the accuracy without using elevation maps ( Left) and with using elevation maps (Right)

The results of using elevation maps give promising insights; choosing this direction as the optimal way to tackle the UXO detection problem in the seabed seems to be the best since we got an accuracy of almost 100%. However, further investigation must be done. It is possible that the model is overfitting and that it would not perform as well when introducing more classes.

## 5. Conclusion

The process of replicating and improving the work titled "Automated Detection of Underwater Military Munitions Using Fusion of 2D and 2.5D Features From Optical Imagery" [3] was both enlightening and challenging. This endeavor shed light on the complexities involved in transitioning between programming environments from MATLAB to Python. Despite the obstacles encountered in deciphering the codebase, significant progress was made.

By employing both classifiers and newer neural network architectures, we gained an understanding of their strengths and potential drawbacks. Although the Python environment posed its challenges with library-specific behaviors, the preliminary results are promising, especially the ones from the neural networks.

One key lesson learned from this undertaking is the reaffirmation of how vital it is to keep code well documented and organized. The initial difficulties faced in comprehending the MATLAB implementation highlight the importance of documentation, which contributes to research projects longevity and reproducibility.



## 6. Limitations

There were many limitations that affected the results of this thesis work. Among them are:

- a. Creating a KNN model with class weights seems to be straightforward in MATLAB. However, upon investigating many Python KNN implementations, no library supported this functionality.
- b. The Bomb Dataset that was used in this thesis work was found among the files of the original work. It appears to be preprocessed already, which might have altered some of the results.
- c. Lack of documentation in the original work made it hard to replicate and to test for bottlenecks. There is a very large number of parameters that can be tweaked in each step of the pipeline.

## 7. Future Work

Looking forward, there are avenues for further exploration:

1. Data Augmentation and GANs: To overcome the limitation of a dataset for munitions, we can consider using Generative Adversarial Networks (GANs) to generate additional synthetic data. This can potentially improve the accuracy and robustness of the models.
2. Exploring Advanced Neural Architectures: Building upon the success of UNET, it would be worthwhile to explore architectures specifically designed for image segmentation tasks. This may lead to results in our research.
3. Enhancing Preprocessing Techniques: As we did not implement sun-flicker removal in this phase, incorporating and optimizing this technique could significantly enhance image quality, subsequently improving classification accuracy.
4. Thorough documentation: It is essential to document all developments and iterations based on the lessons learned from this project. This will significantly benefit research projects. Provide a smoother transition for researchers building upon this work.
5. Collaborating with Marine Biology Experts: Since our work also has applications in classifying reefs, collaborating with experts in biology can provide valuable insights into refining classification metrics and enhancing real-world applicability.

## 8. References

- [1] A. S. M. Shihavuddin, "Automated underwater object classification using optical imagery," M.S. thesis, Departament d'Arquitectura i Tecnologia de Computadors, Universitat de Girona, Girona, Spain, 2014.
- [2] Gleason, Arthur & Gracias, Nuno & Shihavuddin, Asm & Schultz, Greg & Gintert, Brooke. (2015). Improved Supervised Classification of Underwater Military Munitions Using Height Features Derived from Optical Imagery. 10.23919/OCEANS.2015.7404580.
- [3] A. S. M. Shihavuddin, N. Gracias, R. Garcia, R. Campos, A. C. R. Gleason, and B. Gintert, "Automated Detection of Underwater Military Munitions Using Fusion of 2D and 2.5D Features From Optical Imagery," Marine Technology Society Journal, vol. 48, no. 4, pp. 61-71, 2014.
- [4] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," Computing in Science & Engineering, vol. 13, no. 2, pp. 22-30, 2011. [Online]. Available: <https://www.numpy.org/>

- [5] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000. [Online]. Available: <https://opencv.org/>
- [6] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011. [Online]. Available: <https://scikit-learn.org/>
- [7] S. van der Walt et al., "scikit-image: image processing in Python," PeerJ, vol. 2, p. e453, 2014. [Online]. Available: <https://scikit-image.org/>
- [8] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. Springer, Cham, 2015, pp. 234–241.
- [9] H. Rajani, N. Gracias, and R. Garcia, "A convolutional vision transformer for semantic segmentation of side-scan sonar data," Ocean Engineering, vol. 286, no. Part 2, pp. 115647, 2023. [Online]. Available: <https://doi.org/10.1016/j.oceaneng.2023.115647>.
- [10] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," Proc. of the 32nd International Conference on Machine Learning, Lille, France, 2015.
- [11] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," Proc. of the 27th International Conference on Machine Learning, Haifa, Israel, 2010.
- [12] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [13] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems, vol. 32, pp. 8026-8037, 2019.
- [14] R. Porter and N. Canagarajah. Robust rotation-invariant texture classification: wavelet, gabor filter and gmrf based schemes. Vision, Image and Signal Processing, IEE Proceedings -, 144[3]:180 –188, Jun 1997. 57, 80
- [15] Zuiderveld, K. 1994. Contrast limited adaptive histogram equalization. San Diego, CA, USA: Academic Press Professional, Inc Graphics gems IV. pp. 474-85.
- [16] Shihavuddin, A., Gracias, N., Garcia, R., Gleason, A., & Gintert, B. 2013. Image based coral reef classification and thematic mapping. Remote Sensing. 5(4):1809-41. <http://dx.doi.org/10.3390/rs5041809>.
- [17] Ojala, T., Pietikainen, M., & Harwood, D. 1996. A comparative study of texture measures with classification based on featured distributions. Pattern Recogn. 29(1):51-9. [http://dx.doi.org/10.1016/0031-3203\(95\)00067-4](http://dx.doi.org/10.1016/0031-3203(95)00067-4).
- [18] Stokes, M., & Deane, G. 2009. Automated processing of coral reef benthic images. Limnol Oceanogr-Meth. 7:157-68.

[19] A. Russell, "A bomb found at Nanumea in Tuvalu," The Detail, Newsroom, Oct. 7, 2022. [Online]. Available: <https://www.newsroom.co.nz/podcast-the-detail/the-deadly-remnants-of-a-war-that-wont-go-away>.

[20] Y. Loya. The coral reefs of eilat past, present and future: Three decades of coral community structure studies. In Coral Health and Disease, pages 1–34. Springer Berlin Heidelberg, 2004. 125, 131

[21] Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. Linux Journal, 2014(239), 2.