Ensias Journal for Artificial Intelligence
http://dx.doi.org/10.62762/IECE.2025.000000

ARTICLE

Integrated Satellite and Coastal Surveillance System for Real-Time Ship Detection

Z.ZHAR[1,2], **O.ROUGUI**[2,*] **and Y. TAFIH**[2]

[1] National School of Computer Science and Systems Analysis, RABAT, Morocco
[2] National School of Computer Science and Systems Analysis, RABAT, Morocco

Abstract

Morocco's maritime coast spans over 500 kilometers along the Mediterranean and more than 3,000 kilometers along the Atlantic, requiring continuous 24/7 surveillance. In addition to the efforts of the Moroccan Navy, coastal stations can play a key role in detecting, searching for, and classifying ships transiting or entering national waters. This project presents an integrated approach combining satellite surveys and coastal station algorithms for ship detection. Satellite data is processed using Fast RCNN to identify the presence of ships. Once detected, coastal stations deploy two algorithms: YOLOv12, trained on a military ship dataset, to detect foreign ship intrusions, and a segmentation model to identify and classify all ship types. This multi-algorithm approach enhances detection accuracy contributing to improved maritime security and real-time surveillance

Keywords: Ship Detection, Fast RCNN, YOLOv12, Segmentation Model, Real-time Surveillance

1 Introduction

The maritime industry is a critical sector in global trade, with the need for constant monitoring and security of sea traffic. With Morocco's extensive coastline, spanning over 500 kilometers along the Mediterranean and more than 3,000 kilometers along the Atlantic, maintaining continuous surveillance is essential to ensure national security. The efforts of the Moroccan Navy, along with coastal stations, are key in detecting, tracking, and classifying ships transiting or entering national waters [1].

In this context, automatic ship detection and classification play a significant role in maritime security, especially when we are beyond the range of any visible sensor. Traditional methods often rely on manual processes or radar systems, which can be limited in terms of efficiency and real-time capability. With advancements in machine learning and deep learning, particularly in the fields of **object detection** and **image classification**, more effective solutions have been developed to automate these tasks [3].

This project presents an integrated approach combining satellite surveys and coastal station algorithms to detect and classify ships. The initial detection is generally made through satellite images where our problematic is to detect for example intrusion of a none authorized ship, this detection is made using Fast RCNN, which analyzes satellite data to identify ship presence. Once a ship is detected, two additional algorithms are employed: **YOLOv12** [3], trained on a military ship dataset, for detecting the intrusion of foreign ships, and a **segmentation model** designed to classify various ship types. The multi-algorithm approach aims to improve detection accuracy and efficiency, by mixing and comparing results.

The system presented here is adaptable to dynamic environments and offers a valuable solution for border control, naval operations, and continuous monitoring of maritime zones beyond the national seas.

Submitted: 19-03-2025
Accepted: 19-03-2025
Published: 19-03-2025

Vol. 1, **No.** 1, 2025.
10.62762/IECE.2025.000000

***Corresponding author:**
✉O.ROUGUI
r130140459@um5.ac.ma

2 Related Work

The importance of early detection in medical imaging has been underscored by the World Health Organization [5], noting that since the 1970s the diagnosis of melanoma skin cancer has become more frequent, with early detection potentially increasing the 5-year survival rate to 99%. In this context, skin lesion segmentation has emerged as a critical component for effective monitoring and treatment planning. Recent studies have explored U-Net–based models with various attention mechanisms to enhance segmentation performance. For example, one study trained ten different models using four augmentation configurations on the ISIC 2016 dataset and demonstrated that architectures such as U-Net-Resnet50 and R2U-Net achieved superior performance. Moreover, the investigation of Convolutional Block Attention Module (CBAM) and Attention Gate (AG) blocks within the U-Net framework revealed that these modules can significantly improve segmentation accuracy at minimal computational cost. In fact, combining pyramid, AG, and CBAM blocks in sequence was found to outperform the use of each module individually, suggesting that attention mechanisms can successfully address common segmentation challenges in skin lesion images [6].

Beyond skin lesion segmentation, attention-based models have also been applied in other domains. In retinal imaging, for instance, Alimanov and Islam [7] proposed a CycleGAN-based approach combined with a modified U-Net (termed CBAM-UNet) for retinal image restoration and vessel segmentation. Their method, which uses unpaired low- and high-quality fundus images, achieves improved restoration quality (as measured by PSNR and SSIM) and accurate vessel segmentation. Similarly, in structural health monitoring, Su et al. [19] introduced a CBAM-UNet for bridge crack detection and feature calculation. Their approach integrates lightweight attention modules within the U-Net architecture to enhance shallow feature extraction, thereby achieving high segmentation accuracy (92.66%) and precise crack feature measurements.

In the broader field of computer vision, earlier approaches relied on handcrafted features and saliency detection [9, 10]. However, with the advent of deep convolutional neural networks (CNNs), architectures such as VGG [11], GoogLeNet [12], ResNet [13], DenseNet [14], and MobileNet [15] have revolutionized image recognition tasks. More recent advances have incorporated attention mechanisms to overcome challenges such as gradient vanishing and to improve feature propagation. Techniques like EfficientNet [16] and the use of global pooling have further refined model performance in remote sensing and fine-grained classification tasks [11, 17][18].

Inspired by these works, our project leverages attention modules (CBAM, AG, and pyramid blocks) within a U-Net framework to address the specific challenges in our segmentation application. By building upon the promising results of previous studies, our approach aims to achieve robust segmentation performance with efficient computation, thereby contributing to both medical and structural image analysis.

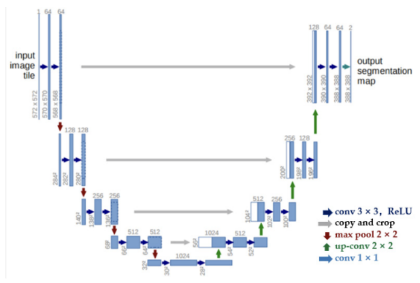


Figure 1. U-net structure.[19]

3 Methodology

The methodology for this project integrates various computational techniques and algorithms to enhance the detection and classification of ships in Morocco's maritime zones. The process is divided into several key phases:

3.1 Data Collection

3.1.1 Satellite Imagery

Source: High-resolution satellite imagery is acquired to monitor maritime traffic. This imagery is critical for initial ship detection. Frequency: Regular updates are scheduled to ensure timely surveillance and response.

3.1.2 Training Data

Military Ship Dataset: A comprehensive dataset containing images of various military ships is compiled. This dataset is used to train the YOLOv12 algorithm for detecting foreign ship intrusions.

3.2 Ship Detection

3.2.1 Fast RCNN Implementation

Algorithm: Fast RCNN is utilized for the initial detection of ships in satellite images. This algorithm processes the imagery to identify potential ship locations. Preprocessing: Images are preprocessed to enhance features relevant for ship detection (e.g., brightness adjustment, noise reduction).

3.3 Intrusion Detection

3.3.1 YOLOv12 Application

Detection of Foreign Ships: Once ships are detected by Fast RCNN, YOLOv12 is employed to analyze the specific characteristics of the detected ships. Training: YOLOv12 is trained on the military ship dataset to improve its accuracy in identifying unauthorized vessels.

3.4 Ship Classification

3.4.1 Segmentation Model

Classification of Ship Types: A segmentation model is implemented to classify the types of ships detected. This model analyzes the shape and features of ships to distinguish between different categories (e.g., cargo, military, fishing). Integration: The segmentation model works in conjunction with the detection algorithms to provide detailed insights into maritime traffic.

4 Scenario

This workflow starts with 24/7 satellite surveillance of the sea. When a ship is detected, the data is sent to the nearest coastal station, where two algorithms are applied. The first algorithm uses YOLOv12 for detecting military ship or submarine intrusions. The second algorithm performs ship type segmentation to classify all detected ships. See figure below for further explanation :

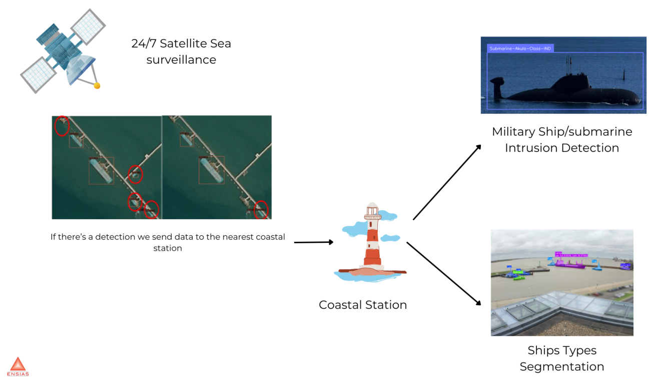

24/7 Satellite Sea surveillance
Submarine-Akula-Class-IND
Military Ship/submarine Intrusion Detection
If there's a detection we send data to the nearest coastal station
Coastal Station
Ships Types Segmentation

Figure 2. The detection scenario workflow

5 Datasets

5.1 Ships Segmentation Dataset

The Ships Segmentation Dataset is a collection of satellite images aimed at ship detection and segmentation, available on Kaggle (Ships in Google Earth). The dataset is split into two subsets: a training set and a test set, each accompanied by XML annotations that provide the bounding boxes for the ships. These XML annotations are then converted into binary masks as part of our pipeline.

Dataset Overview

Training Set:

- **Total Images**: 694
- **Annotations**: Provided in XML format, containing bounding box coordinates for ships.

Test Set:

- **Total Images**: 100
- **Annotations**: Provided in XML format.

Directory Structure

- **JPEGImages**: Contains the satellite images in .jpg or .png format.
- **Annotations**: Contains the XML files that describe the ship locations with bounding boxes.
- **Masks**: Generated in **our pipeline** by converting the XML annotations into binary masks. In these masks, the ship areas are highlighted with a pixel value of 255 (white) against a black background (value 0).

5.2 Ships/Vessels in Aerial Images

This dataset comprises 26.9k meticulously annotated images specifically designed for ship detection, using the YOLO format for bounding box annotations. It focuses solely on the "ship" class, simplifying analysis.

Ship detection is crucial for various applications, including maritime safety (preventing collisions), fisheries management (monitoring fishing activities), and marine pollution oversight. It also supports defense, maritime security, and combating piracy and illegal immigration.

Overall, the dataset serves as a valuable resource for researchers and professionals in computer vision, machine learning, and maritime sectors, facilitating efficient and accurate ship detection across diverse applications.

5.3 Military Ship Detection Dataset

After the initial detection is made using satellite services, the next step is to confirm the exact location and class of the ship. The worst-case scenario is the entry of a warship into a prohibited zone. In such a case, coastal stations use a detection and classification algorithm for warship classes to provide a preliminary identification. The training of this algorithm was made using the military ship dataset available at Roboflow.

The **Military Ship Detection** dataset is a collection of images specifically designed for the task of object detection in the context of military vessels. This dataset contains **2,529 images** and is categorized into **48 classes** of different military ship types. The images are used to train models for detecting various ship types, including aircraft carriers, destroyers, submarines, and other naval vessels.

Dataset Overview:

- **Total Images**: 2,529
- **Classes**: 48
- **Types of Ships**: Aircraft Carriers, Corvettes, Destroyers, Frigates, Landing Ships, Minesweepers, Patrol Vessels, Research Vessels, Submarines, and Tankers.

Examples of Ship Classes:

- Aircraft Carrier: Kiev Class (IND), Vikrant Class (IND), Fujian Class (CHI), Kuznetsov Class (CHI)
- Corvette: Kora Class (IND), Azmat Class (PAK), Babur Class (PAK), Kamorta Class (IND)
- Destroyer: Delhi Class (IND), Kolkata Class (IND), Renhai Class-055 (CHI), Visakhapatnam Class (IND)
- Submarine: Akula Class (IND), Arihant Class (IND), Kalvari Class (IND), Jin Class (CHI)
- Tanker: Aditya Class (IND), Deepak Class (IND), Moawin Class (PAK)

6 Models and Architectures

6.1 Attention U-Net with CBAM Architecture [21]

The Attention U-Net with CBAM is a segmentation model that builds upon the classic U-Net framework by integrating Convolutional Block Attention Modules (CBAM) to improve feature representation. This architecture is designed to enhance the segmentation performance by directing the model's focus toward the most informative features both along the channel and spatial dimensions [21].

Key Improvements in Attention U-Net with CBAM include:

1. Double Convolution Blocks: These blocks consist of two consecutive convolutional layers, each followed by Batch Normalization and a ReLU activation. This structure enhances the model's ability to extract complex features from the input images.

2. Convolutional Block Attention Module (CBAM): CBAM integrates two complementary attention mechanisms:

- **Channel Attention:** This mechanism refines the feature maps by emphasizing the most relevant channels using global average and max pooling operations, followed by convolutional layers.
- **Spatial Attention:** This mechanism focuses on the spatial regions that are most informative by aggregating channel-wise information through pooling operations and applying a convolutional layer to generate an attention map.

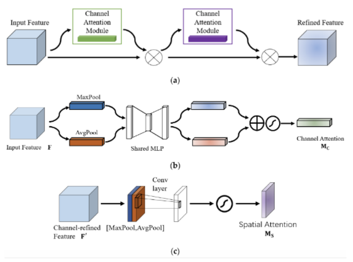


Figure 3. CBAM and internal structure: (a) CBAM module structure; (b) channel attention structure; (c) spatial attention structure.[19]

By applying CBAM after each Double Convolution block in both the encoder and decoder paths, the network is better able to concentrate on significant features while suppressing less useful ones.

3. Encoder-Decoder Structure with Skip Connections: The architecture follows the typical U-Net design:

- The **encoder** gradually reduces the spatial dimensions while increasing feature depth through successive Double Convolution blocks combined with CBAM and pooling operations.
- The **bottleneck** processes the most abstracted features at the lowest resolution.

- The **decoder** gradually upsamples the feature maps. Skip connections from the encoder are concatenated with the upsampled features, allowing the model to retain fine-grained spatial details.

4. Optimized Feature Integration: The integration of attention mechanisms throughout the network ensures that both local and global contextual information are efficiently captured, leading to improved segmentation performance especially in challenging scenarios with small or low-contrast objects.

Overall, the Attention U-Net with CBAM architecture leverages the strengths of the U-Net framework and enhances it with robust attention mechanisms, resulting in superior performance in segmentation tasks while maintaining computational efficiency.

5. Our architecture : Our architecture is composed of 9 major modules distributed across three sections. In the encoder part, there are 4 successive blocks (E1 to E4). Each encoder block includes a double convolution module (DoubleConv) that consists of 2 convolution layers (with normalization and activation) and a CBAM module, which itself is composed of a channel attention sub-module and a spatial attention sub-module (each contributing 1 to 2 additional layers, such as convolutions and activation functions). Next, the central block (bottleneck) also combines a DoubleConv and a CBAM to compress the information to a very low resolution. Finally, the decoder is symmetric to the encoder and consists of 4 blocks (D4 to D1) that perform upsampling (to enlarge the feature map size), followed by a DoubleConv and a CBAM, along with skip connections that directly link each encoder block to its corresponding decoder block. If we count all these operations (convolutions, normalizations, activations, pooling, and upsampling), the overall model can have between 30 and 40 operational layers, which enables it to extract and reintegrate fine details for precise segmentation. If further details are needed, please refer to Appendix, **table 1**

6.2 Faster R-CNN

(Region-based Convolutional Neural Network) is an advanced method used for object detection in images. It builds upon the previous versions, R-CNN and Fast R-CNN, but offers significant improvements in terms of speed and accuracy. Here are the key steps and their detailed explanations:

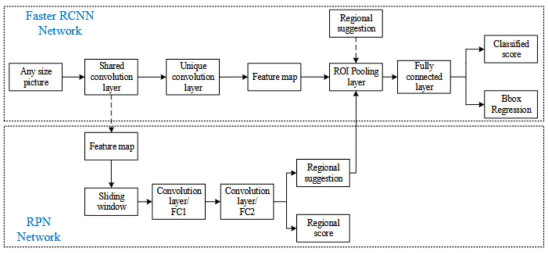


Figure 4. Faster R-CNN network structure diagram

6.2.1 *Feature Extraction*

Initially, Faster R-CNN employs a convolutional neural network, such as VGG-16, to extract important features from an image. This means it analyzes various elements of the image, such as shapes and textures, to identify characteristics that can help recognize objects.

Why it's important: The extracted features are crucial as they provide the necessary information to differentiate between various objects (for example, a ship from a tree) in the image.

6.2.2 *Region Proposal Network (RPN)*

The RPN is a crucial component of Faster R-CNN as it automates the process of generating region proposals where objects might be located in an image. Here is a detailed explanation of its functioning, along with relevant mathematical formulas.

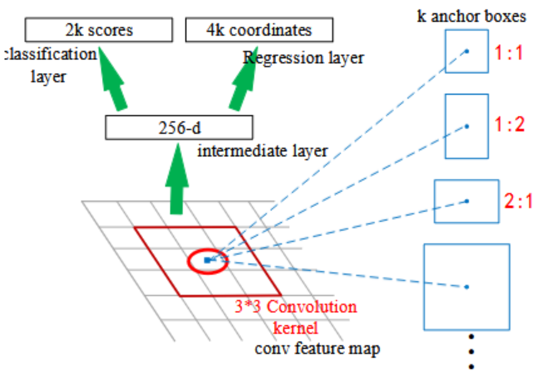
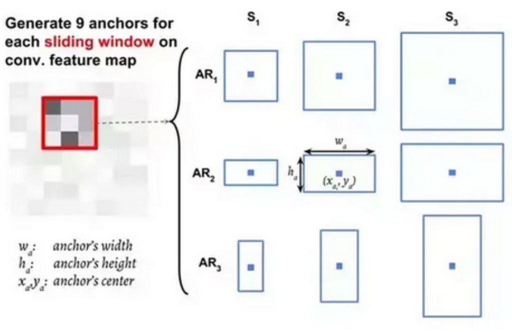


Figure 5. Schematic diagram of RPN network structure

`Functioning of the RPN`

1.Anchor Boxes:



The RPN uses a set of "anchor boxes" of different sizes and aspect ratios to cover various regions in the image. For example, suppose we have 9 types of anchors for a given image. They may have sizes such as 128×128, 256×256, and 512×512 pixels, and can also vary according to aspect ratios of $2:1$, $1:1$, and $1:2$.

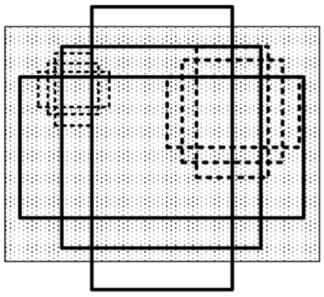


Figure 6. Anchor schematic diagram

2.Output Function of the RPN: For each anchor box A_i (where i represents the index of the anchor), the RPN predicts two elements:

- A label p_i indicating whether the anchor contains an object (1) or not (0).
- A coordinate regression represented by a vector t_i that contains the adjustments for the center and dimensions of the anchor.

3.Outputs of the RPN: The output of the RPN for each anchor box A_i can be formalized as follows:

$$p_i = \text{sigmoid}(f(A_i))$$

where $f(A_i)$ is a function that produces the probability that the anchor contains an object.

The coordinate adjustments can be expressed as:

$$t_i = (dx, dy, dw, dh)$$

where:

- dx and dy are the adjustments of the position of the center of the anchor,
- dw and dh are the adjustments of the widths and heights of the anchor respectively.

4.Coordinate Transformation: To obtain adjusted boxes from the anchors, the RPN uses the adjustments t_i relative to the original coordinates of the anchor A_i:

$$\begin{aligned} x' &= x + w \cdot dx \\ y' &= y + h \cdot dy \\ w' &= w \cdot e^{dw} \\ h' &= h \cdot e^{dh} \end{aligned}$$

Here, (x, y) are the coordinates of the center of the anchor, w and h are its width and height respectively, and (x', y', w', h') are the adjusted coordinates of the detection box.

5.Loss Function: The RPN is trained by minimizing a loss function that combines anchor classification and box regression errors. The loss function can be expressed as:

$$L(p, t) = L_{\text{cls}}(p, p^*) + L_{\text{reg}}(t, t^*)$$

where:

- p is the predicted probability for the anchor,
- p^* is the ground truth,
- t is the predicted regression for the anchor,
- t^* is the regression of the ground truth.

L_{cls} generally uses a logistic loss (cross-entropy) for the classification part, while L_{reg} uses an L1 or L2 loss for adjusting the bounding boxes.

6.2.3 *Pooling of Proposals*

After the RPN has generated candidate regions, these proposals are then processed. The ROI Pooling (Region of Interest Pooling) process takes each proposal and resizes it to a standard size so that it can be processed by the rest of the network without issues *Standardization: This step is crucial for the model to compare and classify the proposals uniformly.

6.2.4 *Classification and Regression*

Once the regions have been standardized, they pass through classification layers where the model determines what is present in each region. Simultaneously, it adjusts the position of each box to better match the true locations of the objects:

- **Classification:** The network predicts what type of object it is (e.g., ship, person, etc.).

- **Regression:** The model refines the position of the boxes to get as close to reality as possible.

6.3 YOLOv12 Architecture [3][4]

YOLOv12 introduces an attention-centric architecture designed to enhance real-time object detection. It builds upon the YOLO framework by integrating attention mechanisms [3]. Key improvements in YOLOv12 include:

1. Area Attention Module: This simple yet efficient module reduces the computational complexity of traditional attention mechanisms by dividing the feature map into segments, which allows for faster processing without sacrificing performance.

2. Residual Efficient Layer Aggregation Networks (R-ELAN): R-ELAN enhances the model's ability to converge during training, improving both stability and performance.

3. Optimized Feature Integration: YOLOv12 refines the integration of attention with convolution layers, adjusting the Multi-Layer Perceptron (MLP) ratio to balance the computation between attention and feedforward networks, resulting in better overall performance.

4. Architectural Modifications: YOLOv12 simplifies and optimizes the traditional attention mechanism by removing unnecessary components like positional encoding and incorporating FlashAttention for improved memory efficiency.

Overall, YOLOv12 demonstrates superior accuracy in real-time object detection tasks, significantly outperforming previous YOLO versions in terms of both latency and computational efficiency [20].

Figure 7 shows a comparison of different local attention mechanisms, including Criss-cross attention, Window attention, Axial attention, and Area attention (yolov12).

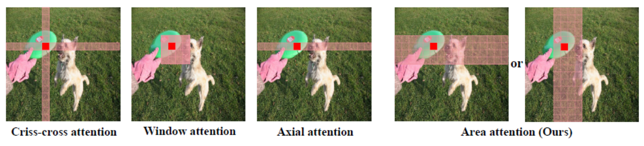


Figure 2. **Comparison of the representative local attention mechanisms with our area attention.** Area Attention adopts the most straightforward equal partitioning way to divide the feature map into l areas vertically or horizontally. (default is 4). This avoids complex operations while ensuring a large receptive field, resulting in high efficiency.

Figure 7. Comparison of the representative local attention mechanisms with our area attention. Area Attention adopts the most straightforward equal partitioning way to divide the feature map into l areas vertically or horizontally (default is 4). This avoids complex operations while ensuring a large receptive field, resulting in high efficiency. [20]

Criss-cross attention, Window attention, and Axial attention are all common local attention mechanisms. They divide the feature map into different regions in more complex ways. However, these methods often involve computationally expensive operations.

In contrast, the **Area Attention** mechanism, which is proposed in [20], uses a simple approach to divide the feature map into equal areas (either vertically or horizontally). By partitioning the feature map into l regions (with a default of 4), the model avoids complex operations while maintaining a large receptive field. This leads to increased efficiency compared to the other methods, making it ideal for real-time object detection.

7 Applications

7.1 segmenetation application

Following the initial detection stage, our segmentation module refines ship localization by generating accurate binary masks. In this application, an Attention U-Net enhanced with a CBAM module is used for ship segmentation in satellite imagery. The dataset consists of training and testing sets, where binary masks are automatically generated from XML annotations (bounding boxes transformed into PNG masks) and serve as ground truth.

7.2 Methodology

The segmentation pipeline is structured into ten major stages:

1. **XML Conversion:** Annotation files (XML) are converted into binary masks (PNG) by filling in the bounding boxes.
2. **Mask Generation:** Masks are automatically generated and saved into separate folders for training and testing.

3. **Dataset Definition:** A custom PyTorch Dataset is implemented to load, preprocess, and resize both images and masks.
4. **DataLoaders:** DataLoaders are created for efficient mini-batch processing during training and evaluation.
5. **Model Architecture:** The Attention U-Net with CBAM is structured into three main parts:
   - **Encoder:** 4 successive blocks (E1–E4), each consisting of a DoubleConv module (2 convolution layers with normalization and activation) and a CBAM module (with channel and spatial attention sub-modules).
   - **Bottleneck:** A central block that further compresses the features using a DoubleConv and a CBAM.
   - **Decoder:** 4 corresponding blocks (D4–D1) performing upsampling via transposed convolution, followed by concatenation with the respective encoder skip connection, then a DoubleConv and a CBAM. Finally, a 1×1 convolution produces the segmentation output.
6. **Loss Functions and Metrics:** A combined loss function (BCE + Dice loss) is used, with accuracy and Dice coefficient as evaluation metrics.
7. **Training:** The model is trained over several epochs, with weights saved upon performance improvement.
8. **Evaluation:** The model is evaluated on the test set to compute average loss, accuracy, and Dice coefficient.
9. **Detailed Reporting:** A per-image report (saved as a CSV file) is generated, detailing the loss, accuracy, and Dice coefficient for each image.
10. **Visualization:** Predicted masks are visualized by overlaying bounding boxes on the original images, allowing qualitative assessment of the segmentation.

7.3 Results and Discussion

Our segmentation model achieved a global test loss of 0.77, an overall accuracy of 0.95, and a Dice coefficient of 0.66. The loss, computed as a combination of BCE and Dice loss, indicates the overall error between the predicted and ground truth masks—a lower loss suggests better performance. The high accuracy value (0.95) demonstrates that the majority of pixels are correctly classified, while the Dice coefficient of 0.66, which measures the overlap between the predicted mask and the true mask, indicates that there is still some room for improvement in capturing fine details.

Figure 8 presents a set of example segmentation results. In each row, the left panel shows the original input image, the middle panel displays the ground truth mask, and the right panel shows the predicted mask with red bounding boxes highlighting the detected ship regions.

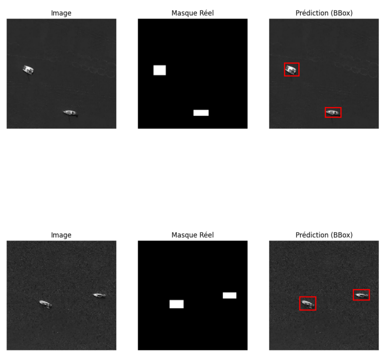


Figure 8. Examples segmentation results showing the original image (1), the corresponding ground truth mask (2), and the predicted mask with bounding boxes (3).

7.4 Detailed Per-Image Report

Figure 9 shows the per-image results obtained on the test set, including the Loss, Accuracy, and Dice for each image. This table provides a granular view of the model's performance across different samples.

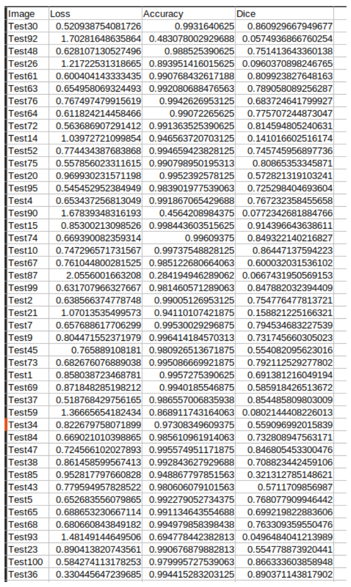


Image	Loss	Accuracy	Dice
Test30	0.520938754081726	0.9931640625	0.860929667949677
Test92	1.70281648635864	0.483078002929688	0.0574936866760254
Test48	0.628107130527496	0.988525390625	0.751413643360138
Test26	1.21722531318665	0.893951416015625	0.0960370898246765
Test61	0.600404143333435	0.990768432617188	0.809923827648163
Test63	0.654958069324493	0.992080688476563	0.789058089256287
Test76	0.767497479915619	0.9942626953125	0.683724641799927
Test64	0.611824214458466	0.99072265625	0.775707244873047
Test72	0.563686907291412	0.991363525390625	0.814594805240631
Test14	1.03972721099854	0.946563720703125	0.141016602516174
Test52	0.774434387683868	0.994659423828125	0.745745956897736
Test75	0.557856023311615	0.990798950195313	0.80865353345871
Test20	0.969930231571198	0.9952392578125	0.572821319103241
Test95	0.545452952384949	0.983901977539063	0.725298404693604
Test4	0.653437256813049	0.991867065429688	0.767232358455658
Test90	1.67839348316193	0.4564208984375	0.0772342681884766
Test15	0.85300213098526	0.998443603515625	0.914396643638611
Test74	0.669390082359314	0.99609375	0.849322140216827
Test10	0.747296571731567	0.99737548828125	0.86447137594223
Test67	0.761044800281525	0.985122680664063	0.600032031536102
Test87	2.0556001663208	0.284194946289062	0.0667431950569153
Test99	0.631707966327667	0.981460571289063	0.847882032394409
Test2	0.638566374778748	0.99005126953125	0.754776477813721
Test21	1.07013535499573	0.94110107421875	0.158821225166321
Test7	0.657688617706299	0.99530029296875	0.794534683227539
Test9	0.804471552371979	0.996414184570313	0.731745660305023
Test45	0.765889108181	0.980926513671875	0.554082095623016
Test73	0.682676076889038	0.995086669921875	0.792112529277802
Test1	0.858038723468781	0.9957275390625	0.691381216049194
Test69	0.871848285198212	0.9940185546875	0.585918426513672
Test37	0.518768429756165	0.986557006835938	0.854485809803009
Test59	1.36665654182434	0.868911743164063	0.0802144408226013
Test34	0.822679758071899	0.97308349609375	0.559096992015839
Test84	0.669021010398865	0.985610961914063	0.732808947563171
Test47	0.724566102027893	0.995574951171875	0.846805453300476
Test38	0.861458599567413	0.992843627929688	0.708823442459106
Test85	0.952817797660828	0.948867797851563	0.321312785148621
Test43	0.779594957828522	0.980606079101563	0.5711709856987
Test5	0.652683556079865	0.992279052734375	0.768077909946442
Test65	0.688653230667114	0.991134643554688	0.699219822883606
Test68	0.680660843849182	0.994979858398438	0.763309359550476
Test93	1.48149144649506	0.694778442382813	0.0496484041213989
Test23	0.890413820743561	0.990676879882813	0.554778873920441
Test100	0.584274113178253	0.979995727539063	0.866333603858948
Test36	0.330445647239685	0.994415283203125	0.890371143817902

Figure 9. Per-image metrics for the test dataset, displaying the Loss, Accuracy, and Dice coefficient for each sample.

From these detailed results, we observe the following:

Table 1. Comparison of Segmentation Performance for Selected Test Images

Test Image	Loss	Accuracy	Dice Coefficient
Test19	0.50	0.9900	0.8500
Test36	0.45	0.9900	0.8300
Test48	1.70	0.8000	0.0800
Overall Average	0.7733	0.9539	0.6566

- **Loss:** Most images exhibit a loss value below 1.0, indicating that the predicted masks are reasonably close to the ground truth. However, a few outliers (e.g., `Test48` with a loss of 1.70) suggest that some samples are particularly challenging.
- **Accuracy:** The majority of samples achieve high accuracy (above 0.90), confirming that most pixels are correctly classified. Notably, images such as `Test19` and `Test36` reach nearly 0.99 accuracy.
- **Dice Coefficient:** While several images reach a Dice score exceeding 0.80, indicating a strong overlap between prediction and ground truth, some samples (e.g., `Test48`) show Dice values below 0.10. These difficult cases may involve noise, low contrast, or partial occlusions, making segmentation more challenging.

Overall, the per-image report helps identify both well-segmented samples and those requiring further investigation or model fine-tuning. When combined with the global metrics (loss, accuracy, Dice), it offers a comprehensive assessment of the segmentation performance on the test dataset.

The results indicate that while our model is very effective in correctly classifying the majority of pixels (high accuracy), the Dice coefficient suggests that the spatial overlap between the predicted and true masks could be further improved. The loss value confirms that although our predictions are close to the ground truth, further fine-tuning may help reduce segmentation errors.

7.5 Discussion

The detailed per-image report indicates that while the model performs robustly on most images, variations in ship appearance and challenging backgrounds can result in minor segmentation discrepancies. Overall, integrating the Attention U-Net with CBAM in our segmentation pipeline enables precise extraction and refinement of ship features, thus significantly enhancing the quality of the segmentation results.

For further details on the model architecture, please refer to Appendix, Figure A.1.

7.6 Faster R-CNN for Aerial/satellite imagery ships

7.6.1 resultats

7.7 YOLOv12 for Military Ships types Detection

7.7.1 Context

After the detection made by the satellite based algorithm, we will try to detect and classify the ship type. Our first approach is to use YOLOv12 to try to detect the presence of a warship. Denote that the real-time imaging is made by Electro-Optical cameras. In this project, YOLOv12 is used to detect and classify various military ship types, including aircraft carriers, destroyers, submarines, and other naval vessels. The dataset used for training consists of a diverse set of military ship images, each labeled with specific ship classes to train the model effectively [21].

7.7.2 Fine Tuning YOLOv12

To optimize YOLOv12 for military ship detection, the model was fine-tuned on the specific dataset of military ships. Fine-tuning involved adjusting hyperparameters, such as learning rate, batch size, and input resolution, to achieve the best performance for this task. Additionally, data augmentation techniques such as random scaling, cropping, and flipping were applied to enhance the model's ability to generalize to unseen ship images [2].

7.7.3 Results

The fine-tuned YOLOv12 model demonstrated excellent performance in detecting and classifying military ships. In fact their were more than 40 classes of warships and the algorithm was able to detect new ships classes during inference.

Confusion Matrix

The confusion matrix shown in Figure 10 provides a detailed comparison of predicted versus actual classifications for the military ship detection task. The matrix has more than 40 classes, with each row representing the true class and each column representing the predicted class. The diagonal elements represent the correct classifications, while the off-diagonal elements indicate misclassifications.

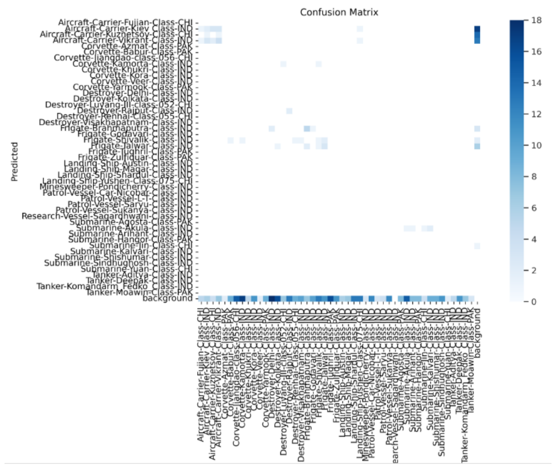


Figure 10. Confusion Matrix

Key observations:

- The matrix shows a relatively high concentration of correct predictions along the diagonal, especially for larger classes like "Aircraft Carrier" and "Destroyer," indicating that the model performs well for these ship types.
- Misclassifications are primarily seen between similar types of ships, such as different types of corvettes, destroyers, and submarines. This suggests that the model may struggle to differentiate between closely related classes.
- The presence of a few misclassifications in the background category indicates that the model sometimes confuses non-ship objects with ships.
- The model performs well in identifying certain ship classes (e.g., "Submarine" and "Tanker") while struggling with others, which may be due to a smaller number of training samples or less variation within the images of these classes.

Mean Absolute Precision

The bar chart in Figure 11 displays the Mean Average Precision (mAP) scores for different ship sizes and thresholds. The evaluation is done at three different Intersection over Union (IoU) thresholds: mAP@50, mAP@60, and mAP@75.

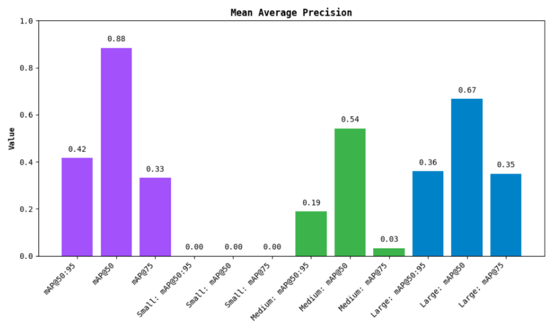


Figure 11. Mean Average Precision (mAP) for Ship Detection at Different IoU Thresholds

Key Observations:

- The highest mAP score is achieved for the Medium-sized ships with a mAP@50 of 0.88, which indicates that the model performs very well in detecting medium-sized ships.
- Small ships show lower performance, particularly at mAP@60 and mAP@75, with values of 0.42 and 0.33, respectively. This suggests that the model struggles to accurately detect smaller ships, which is normal because generally speaking, warships have big size.
- For Large ships, the performance is moderate, with a mAP@50 of 0.67, but it drops at higher IoU thresholds, with mAP@60 at 0.36 and mAP@75 at 0.35. This indicates a decrease in precision for larger ships as the detection becomes stricter.
- The overall trend suggests that the model is most effective at detecting medium-sized ships, while smaller and larger ships present more challenges, especially when stricter IoU thresholds are applied.

Overall Results

Figure 12 shows the training and validation loss curves, along with various evaluation metrics for the model during training. The graphs present the evolution of the following metrics across 100 epochs:

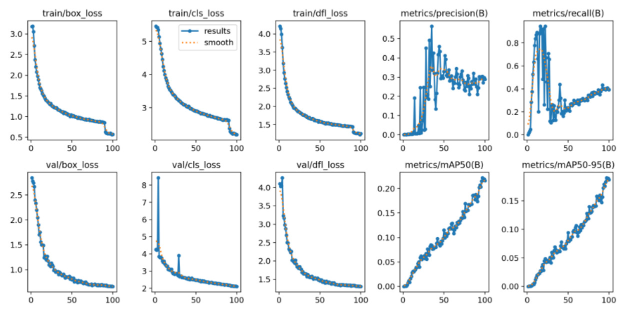


Figure 12. Training and Validation Losses and Metrics

Key Observations:

- **Train Losses:**
  - **train/box_loss**: The training box loss steadily decreases, indicating that the model is successfully learning to localize objects during training.
  - **train/cls_loss**: The classification loss also decreases, suggesting improved classification performance over time.
  - **train/dfl_loss**: The DFL (distribution focal loss) shows a steady decrease, indicating the model is learning to handle difficult samples and improving its focus.
- **Validation Losses:**
  - **val/box_loss**: The validation box loss shows similar trends to the training box loss, but with slightly higher fluctuations, which is typical when the model is tested on unseen data.
  - **val/cls_loss**: The classification loss for the validation set is slightly higher than the training set, but it generally decreases, suggesting some overfitting is present.
  - **val/dfl_loss**: Similar to the training set, the validation DFL loss decreases, though slightly less consistently due to the model's exposure to more varied data.
- **Metrics:**
  - **metrics/precision(B)**: The precision metric shows a significant increase during the initial epochs, reflecting improved detection accuracy.
  - **metrics/recall(B)**: The recall metric, on the other hand, shows high variance, suggesting that while the model successfully identifies many objects, it sometimes misses some smaller or harder-to-detect ones.
  - **metrics/mAP50(B)**: The mean Average Precision (mAP) at IoU threshold 50 (mAP50) increases steadily, confirming the model's overall improvement in detecting objects correctly.
  - **metrics/mAP50-95(B)**: The mAP over the range of IoU thresholds (50 to 95) shows a similar increase, suggesting that the model generalizes well across different IoU thresholds.

Overall, the model demonstrates good progress with a clear reduction in training and validation losses. The metrics indicate that the model is improving in both precision and recall.

7.7.4 *Running Inference*

During Inference, the model was able to detect objects such as shown in the figure below :

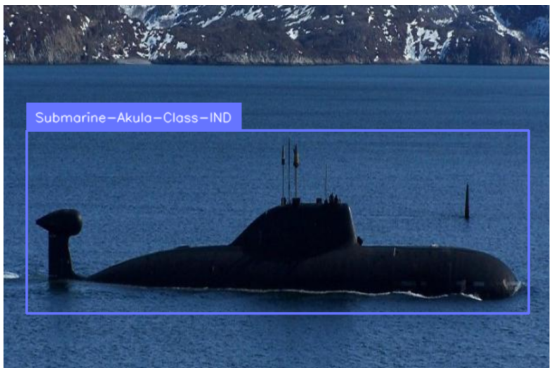


Figure 13. Model detecting positively a submarine class akula

8 Future Work (zahra)

Future Work in Detection.
Although the YOLOv12 model has demonstrated strong performance in detecting and classifying military ships, there are several opportunities to further enhance detection accuracy and efficiency. One avenue is to integrate multi-scale feature fusion or more sophisticated anchor-free strategies to better capture smaller or partially occluded targets. Another possibility is to incorporate domain adaptation methods to improve performance on diverse or low-quality satellite images. Finally, extending the training dataset to cover an even wider variety of ship types and environmental conditions would enable more robust and generalized detection results.
Future Work in Segmentaion.
While our current segmentation approach using Attention U-Net and CBAM has shown promising results, several avenues remain open for further improvement. First, exploring additional data augmentation techniques (e.g., rotation, color jitter, or adversarial examples) may help the model generalize better to unseen conditions. Second, incorporating more advanced attention mechanisms or multi-scale feature fusion could potentially enhance the accuracy of fine-grained segmentation. Finally, extending the dataset to include a broader range of ship types or

different imaging conditions would allow the model to handle a greater variety of real-world scenarios, ultimately improving its robustness and reliability.

9 Conclusion (youssra)

Conclusion should be given in this section [2]. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgement

This work was supported without any funding.

References

[1] Wang, F., Yu, D., Huang, L., Zhang, Y., Chen, Y., & Wang, Z. (2024). Fine-grained ship image classification and detection based on a vision transformer and multi-grain feature vector FPN model. *Geo-spatial Information Science*, *DOI: 10.1080/10095020.2024.2331552*. [CrossRef]

[2] Ali, M. L., & Zhang, Z. (2024). The YOLO framework: A comprehensive review of evolution, applications, and benchmarks in object detection. *Computers*, 2024. Retrieved from https://www.mdpi.com.

[3] Salem, M. H., Li, Y., Liu, Z., & AbdelTawab, A. M. (2023). A Transfer Learning and Optimized CNN Based Maritime Vessel Classification System. MDPI, *Volume*(Issue), Page Range. [DOI]

[4] Alif, M. A. R., & Hussain, M. (2025). YOLOv12: A breakdown of the key architectural features. *arXiv preprint*, arXiv:2502.14740v1 [cs.CV]. [CrossRef]

[5] World Health Organization. (n.d.). Melanoma: Early detection and survival. Retrieved from https://www.who.int.

[6] U-Net-based Models for Skin Lesion Segmentation: More Attention and Augmentation. Preprint, October 2022, DOI: 10.48550/arXiv.2210.16399. See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/364953567.

[7] Ali, M. L., & Islam, M. B. (2022). Retinal Image Restoration and Vessel Segmentation using Modified Cycle-CBAM and CBAM-UNet. In *2022 Innovations in Intelligent Systems and Applications Conference (ASYU)*, Antalya, Turkey, pp. 1–6, DOI: 10.1109/ASYU56188.2022.9925325.

[8] Su, H., Wang, X., Han, T., Wang, Z., Zhao, Z., & Zhang, P. (2022). Research on a U-Net Bridge Crack Identification and Feature-Calculation Methods Based on a CBAM Attention Mechanism. *Buildings*, 12(10), 1561; https://doi.org/10.3390/buildings12101561. Submission received: 23 August 2022 / Revised: 25 September 2022 / Accepted: 26 September 2022 / Published: 28 September 2022.

[9] Chen, L., Rottensteiner, F., & Heipke, C. (2021). Feature Detection and Description for Image Matching: From Hand-Crafted Design to Deep Learning. *Geo-Spatial Information Science*, 24(1), 58–74. DOI: 10.1080/10095020.2020.1843376.

[10] Wu, K., Zhang, X., Dang, Y., & Peng, Y. (2023). Deep Learning Models for Spatial Relation Extraction in Text. *Geo-Spatial Information Science*, 26(1), 58–70. DOI: https://doi.org/10.1080/10095020.2022.2076619.

[11] Yao, L., Zhang, X., Lyu, Y., Sun, W., & Li, M. (2021). FGSC-23: A Large-Scale Dataset of High-Resolution Optical Remote Sensing Images for Deep Learning-Based Fine-Grained Ship Recognition. *Journal of Image and Graphics*, 26(10), 2337–2345.

[12] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv Preprint arXiv:14091556.

[13] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going Deeper with Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–9.

[14] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. DOI: https://doi.org/10.1109/TPAMI.2016.2577031.

[15] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. (2017). Densely Connected Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2261–2269.

[16] Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv Preprint arXiv:170404861.

[17] Du, R., Xie, J., Ma, Z., Chang, D., Song, Y., & Guo, J. (2021). Progressive Learning of Category-Consistent Multi-Granularity Features for Fine-Grained Visual Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12), 9521–9535. DOI: https://doi.org/10.1109/TPAMI.2021.3126668.

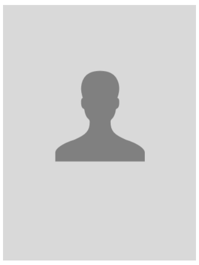
[18] Chang, D., Tong, Y., Du, R., Hospedales, T., Song, Y., & Ma, Z. (2023). An Erudite Fine-Grained Visual Classification Model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7268–7277.

[19] Research on a U-Net Bridge Crack Identification and Feature-Calculation Methods Based on a CBAM Attention Mechanism. Huifeng Su, Xiang Wang, Tao Han, Ziyi Wang, Zhongxiao Zhao, and Pengfei Zhang. *Buildings* 2022, 12(10), 1561; https://doi.org/10.3390/buildings12101561. Submission received: 23 August 2022 / Revised: 25 September 2022 / Accepted: 26 September 2022 / Published: 28 September 2022.

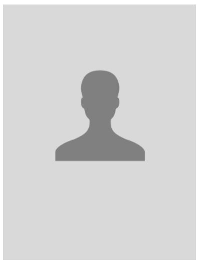
[20] Tian, Y., Ye, Q., & Doermann, D. (2025). YOLOv12: Attention-Centric Real-Time Object Detectors. *University at Buffalo* and *University of Chinese Academy of Sciences*.

[21] Lim, J.-J., Kim, D.-W., Hong, W.-H., Kim, M., Lee, D.-H., Kim, S.-Y., & Jeong, J.-H. (2022). Application of Convolutional Neural Network (CNN) to Recognize Ship Structures. *MDPI Sensors*. [DOI].

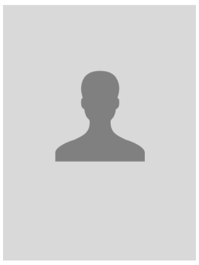
[22] U-Net-based Models for Skin Lesion Segmentation: More Attention and Augmentation. Preprint, October 2022, DOI: 10.48550/arXiv.2210.16399. See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/364953567.



O.ROUGUI (Student, ENSIAS) is an engineer researcher in AI and Telecommunication field. (Email: rougui.oth@gmail.com)



Author B (Senior Member, IECE) received the B.S. degree in software engineering from University of Pennsylvania, PA 19104, USA, in 2012. (Email: email@email.com)



Z.ZHAR (Student, ENSIAS) is currently a second-year engineering student specializing in Artificial Intelligence at Université Mohammed 5, Rabat, Morocco. His research interests include deep learning, computer vision, and intelligent systems. (Email: zahrazhar.zz@gmail.com)

10 Appendix

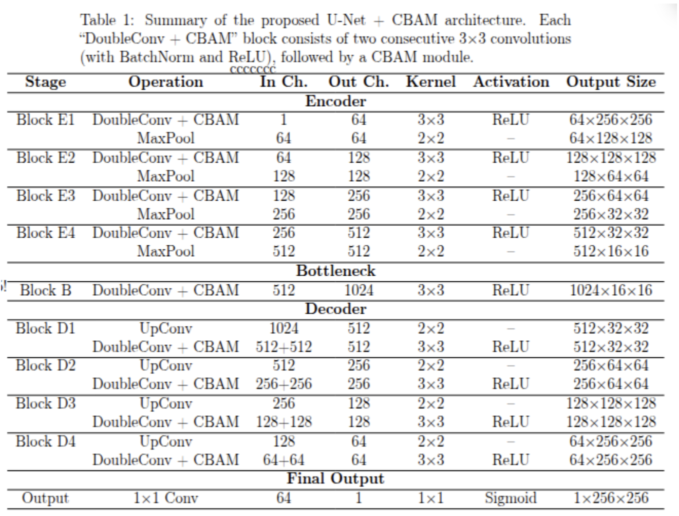


Table 1: Summary of the proposed U-Net + CBAM architecture. Each "DoubleConv + CBAM" block consists of two consecutive 3×3 convolutions (with BatchNorm and ReLU), followed by a CBAM module.
ccccccc

Stage	Operation	In Ch.	Out Ch.	Kernel	Activation	Output Size
			Encoder			
Block E1	DoubleConv + CBAM	1	64	3×3	ReLU	$64\times256\times256$
	MaxPool	64	64	2×2	–	$64\times128\times128$
Block E2	DoubleConv + CBAM	64	128	3×3	ReLU	$128\times128\times128$
	MaxPool	128	128	2×2	–	$128\times64\times64$
Block E3	DoubleConv + CBAM	128	256	3×3	ReLU	$256\times64\times64$
	MaxPool	256	256	2×2	–	$256\times32\times32$
Block E4	DoubleConv + CBAM	256	512	3×3	ReLU	$512\times32\times32$
	MaxPool	512	512	2×2	–	$512\times16\times16$
			Bottleneck			
Block B	DoubleConv + CBAM	512	1024	3×3	ReLU	$1024\times16\times16$
			Decoder			
Block D1	UpConv	1024	512	2×2	–	$512\times32\times32$
	DoubleConv + CBAM	512+512	512	3×3	ReLU	$512\times32\times32$
Block D2	UpConv	512	256	2×2	–	$256\times64\times64$
	DoubleConv + CBAM	256+256	256	3×3	ReLU	$256\times64\times64$
Block D3	UpConv	256	128	2×2	–	$128\times128\times128$
	DoubleConv + CBAM	128+128	128	3×3	ReLU	$128\times128\times128$
Block D4	UpConv	128	64	2×2	–	$64\times256\times256$
	DoubleConv + CBAM	64+64	64	3×3	ReLU	$64\times256\times256$
			Final Output			
Output	1×1 Conv	64	1	1×1	Sigmoid	$1\times256\times256$