Ship Detection using Deep Learning and Image Segmentation

Sai Vineeth Kaza,[1] Rajesh Karumanchi,[2] Aishwarya Kurnutala[3]
Northeastern University[1,2,3]
kaza.s@northeastern.edu,[1] karumanchi.r@northeastern.edu,[2] kurnutala.a@northeastern.edu[3]

Abstract

Increased maritime traffic has amplified the risk of ship-related accidents. Traditional tracking systems such as Automatic Identification Systems (AIS) often fail to detect vessels that do not comply with standard transponder protocols, underscoring the need for more sophisticated ship detection techniques. In response to this challenge [4], we have leveraged satellite imagery from an Airbus Kaggle competition to implement advanced image segmentation using deep learning algorithms. Our approach employs Exploratory Data Analysis (EDA) to understand data, Data Pre-processing to decode the masks from RLE format, the use of DL algorithms like Mask R-CNN[12], a pre-trained Mask R-CNN model with MSCOCO weights [1] for instance segmentation, and U-Net [10] for semantic segmentation. These models are designed to identify ships and generate corresponding boundary boxes in satellite images. Through hyperparameter tuning, we enhanced the performance of the pre-trained Mask R-CNN model, leading to improved evaluation metrics. We also conducted a comparative analysis of instance and semantic segmentation results using metrics such as Intersection Over Union (IOU) score, Mean Average Precision (MAP), and F2-score. The pre-trained Mask R-CNN model emerged as the top performer, achieving an F2 score of 0.78 and a MAP score of 0.63. Interestingly, the U-Net model, without any pre-trained weights, also yielded a notable F2 score of 0.669. Given that our focus was on two classes - ships and the background - the semantic segmentation approach produced significant results even without pre-trained weights.

Introduction

Maritime traffic plays a crucial role in global trade and transportation, but it also presents significant risks and challenges. Traditional ship tracking systems, such as Automatic Identification Systems (AIS), have limitations in detecting non-compliant or transponder-disabled vessels. Due to the limited operational efficacy of the current system, reliance on satellite imagery is essential for creating advanced ship detection and segmentation algorithms to boost maritime safety. Leveraging the power of deep learning, this project aims to develop innovative approaches to accurately detect and segment ships from satellite images.

The main problem addressed in this project is accurate image segmentation and localization of ships within satellite images. This task poses several challenges, including blurred images, small ship sizes, instances where ships are too close to provide individual boundary boxes, and potential confusion with small houses. To overcome these challenges, the project focuses on utilizing the Mask R-CNN (Mask Region-based Convolutional Neural Network) framework, which performs pixel-level segmentation. By leveraging deep learning techniques, Mask R-CNN not only identifies the presence of ships but also provides precise masks that delineate the boundaries of each ship.

The dataset for this project is provided by Airbus's satellite Data Division and obtained from the Airbus Ship Detection Challenge [4] on Kaggle. It consists of approximately 208,000 satellite images, each with dimensions of 768 by 768 pixels. The dataset includes ground truth segmentation masks for ships in the training set, provided in run-length encoded format.

Although alternative object detection algorithms like YOLO have become popular, they were not suitable for this project because they lack the capability to directly provide pixel-level segmentation masks. As a result, the Mask R-CNN is considered as the baseline model, which effectively integrates object detection with instance segmentation. By utilizing deep convolutional neural networks and region proposal techniques, Mask R-CNN can accurately detect ships and generate segmentation masks, enabling precise localization and identification of ships in satellite imagery.

In addition to the Mask R-CNN framework, the project also explores the U-Net architecture for semantic segmentation tasks. U-Net is known for its effectiveness in segmenting objects in medical imaging, and its capabilities extend to ship segmentation in this project. By fine-tuning hyperparameters, comparing different models, and evaluating the output using various metrics, valuable insights into the effectiveness of Mask R-CNN and U-Net models for ship detection and segmentation tasks are obtained.

The project aims to achieve precise ship detection and segmentation in satellite images, ensuring that the generated segmentation masks accurately align with the actual ship locations and shapes. To assess the performance, our team intends to compare the outcomes with a pre-trained R-CNN model utilizing MS COCO weights. However, a potential concern lies in the training time and performance speed, which may be impacted by the extensive size of the dataset.

The project responsibilities are distributed among the team members, with each member contributing to different sub-tasks such as exploratory data analysis (EDA), satellite image processing for super-resolution, implementation of the Mask R-CNN base model, training the pre-trained R-CNN with MS COCO weights, hyperparameter tuning and implementation of the U-Net model.

Background

Segmentation is a computer vision task that involves partitioning an image into distinct regions or segments based on certain criteria.

The goal is to group pixels or regions that share similar characteristics or belong to the same object or class.

Semantic segmentation aims to classify each pixel or region in an image into pre-defined semantic categories or classes. The goal is to assign a single label to each pixel, representing the class or category to which it belongs. The output of semantic segmentation is a pixel-level mask that provides a high-level understanding of the scene by labeling each pixel with its corresponding class. The segmentation is typically performed on a per-pixel basis, without distinguishing between individual instances of the same class. (Considers all dogs as one class). Instance segmentation not only classifies pixels into semantic categories but also differentiates individual instances of objects within the same class. The objective is to assign a unique label or identity to each instance of an object in the image. The output of instance segmentation is a pixel-level mask that precisely outlines the boundaries of each object instance. (Considers each dog as a different class). Differences b/w instance vs semantic segmentation can be seen in Fig 1.

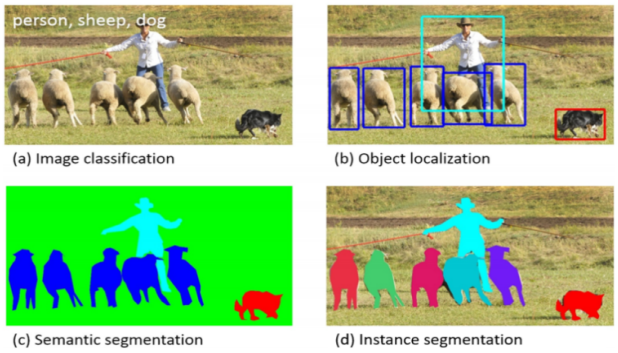

Fig 1. Instance vs Semantic Segmentation

Image Segmentation vs Object Detection:

The main goal of image segmentation is to identify and group pixels or regions that belong to the same object or have similar characteristics. Segmentation techniques assign a label to each pixel or region, indicating its membership to a particular object or class. The output of segmentation is a pixel-level mask or a set of segmented regions (0 & 1's in 768x768 array). Whereas object detection is the task of identifying and localizing objects within an image, typically by drawing bounding boxes around them. The main objective of object detection is to detect the presence of objects in an image and provide their approximate locations. The output of object detection consists of the class labels of the detected objects and the corresponding bounding box coordinates (y1, x1, y2, x2). Some of the key differences include granularity where segmentation provides pixel-level or region-level labelling, whereas object detection provides bounding box-level localization. Another key difference is segmentation can provide detailed information about the boundaries and shapes of objects (rather than simple rectangles), while object detection focuses on identifying objects without detailed per-pixel information.

Related Work

After exploring Kaggle competition discussions and open notebooks, we came across various approaches for the Mask RCNN (instance-based) model, as well as a few YoLo (You Only Look Once) approaches. The notes from the competition winner explained that they achieved success by creating an ensemble of deeper and shallower networks, combining InceptionResNetV2, ResNet34, ResNet152, and Mask R-CNN, which followed instance-based segmentation.

Based on this information, we decided to pursue a comparative analysis between instance and semantic segmentation approaches. Our baseline approach was the Mask R-CNN (instance) model, and we compared it with the U-Net (semantic) model. Additionally, we used the best-trained weights of Mask R-CNN on the Coco dataset. While Yolo is considered a state-of-the-art model, we discovered through discussions that it might not be the right approach for our dataset.

Exploratory Data Analysis

During the exploratory data analysis, we conducted a comprehensive examination of the dataset to gain a deeper understanding of ship presence characteristics. A crucial feature we investigated was the presence of encoded pixels, where images devoid of ships had "NA" value in Encoded Pixels column. This allowed us to accurately identify and differentiate between images with and without ships. Additionally, we calculated the Ship Area Percentage for each image, providing valuable insights into the relative size of ships compared to the overall image dimensions. Furthermore, we observed that the image height and width remained consistent across all samples, ensuring a standardized framework for subsequent model development and evaluation. A sneak peak of the dataset can be seen in Fig 2.

Data Exploration

```
In [11]: print(f'Number of rows in the data - {train_segmentation.shape[0]}')
         Number of rows in the data - 231722

In [12]: #preview of the data in train_segmentation
         train_segmentation.head(20)
```

Out[12]:

	ImageId	EncodedPixels	Img_Height	Img_Width	percentage_ship_area
0	00003e153.jpg	<NA>	768	768	0.000000
1	0001124c7.jpg	<NA>	768	768	0.000000
2	000155de5.jpg	264661 17 265429 33 266197 33 266965 33 267733...	768	768	0.574409
3	000194a2d.jpg	360486 1 361252 4 362019 5 362785 8 363552 10 ...	768	768	0.030009
4	000194a2d.jpg	51834 9 52602 9 53370 9 54138 9 54906 9 55674 ...	768	768	0.025092
5	000194a2d.jpg	198320 10 199088 10 199856 10 200624 10 201392...	768	768	0.081380
6	000194a2d.jpg	55683 1 56451 1 57219 1 57987 1 58755 1 59523 ...	768	768	0.001187
7	000194a2d.jpg	254389 9 255157 17 255925 17 256693 17 257461 ...	768	768	0.109863
8	0001b1832.jpg	<NA>	768	768	0.000000
9	00021ddc3.jpg	108287 1 109054 3 109821 4 110588 5 111356 5 1...	768	768	0.067817

Fig 2. Illustration of Ship Segmentation Dataset

```
ship_counts = count_ships.value_counts()
print(ship_counts)
```

```
NumberOfShips
0                149999
1                 27104
2                  7674
3                  2954
4                  1622
5                   925
6                   657
7                   406
8                   318
9                   243
10                  168
11                  144
12                  124
14                   76
13                   75
15                   66
dtype: int64
```

Fig 3. Number of ships in Images

In the dataset, there are approximately 149,999 images without ships and 27,104 images with one ship. Fig 3 visualizes the count of ships in images, highlighting the distribution of ships in images.

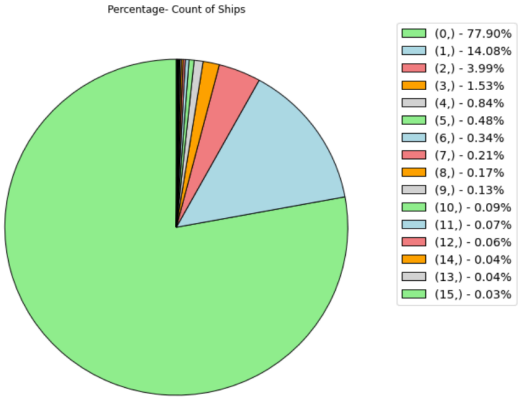

Fig 4. Percentage of Images with ships

Fig 4 depicts a pie chart showing the percentage of images in the dataset having ships. We observed around 78% of the images did not contain any ships. Approximately 14% of the images had a single ship, while the remaining 8% contained multiple ships, ranging from 2 to 15. These findings from our analysis provide a solid foundation for effectively addressing ship detection and segmentation challenges in satellite imagery.

To further analyze the ship sizes, we generated box plot and histogram to illustrate the Ship Area Percentage. Fig 5, Fig 6 shows the box plot and histogram of the Ship Area Percentage respectively. These visual representations revealed that the majority of ships occupy a remarkably small area within the image, often less than 1%. Notably, a significant proportion of ships were found to occupy as little as 0.1% of the image. These findings highlight the prevalence of small-sized ships in the dataset and emphasize the need for accurate detection and segmentation algorithms capable of effectively identifying and delineating these minute ship instances.

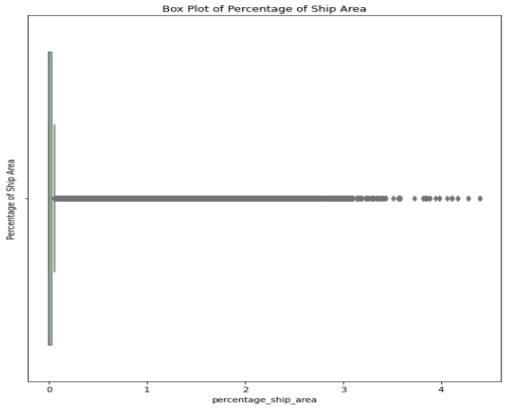

Fig 5. Box plot of Ship Area Percentage

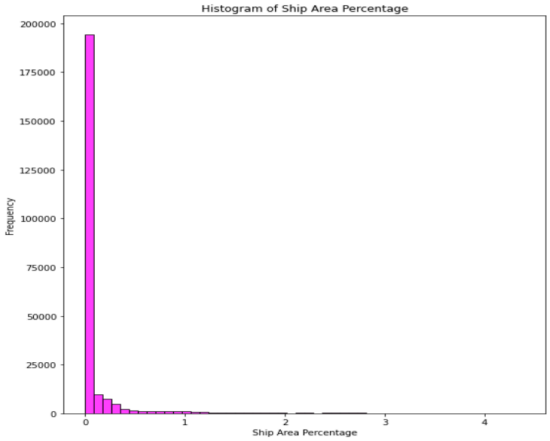

Fig 6. Distribution of Ship Area Percentage

Why Not YOLO? & Super Resolution Failure

While exploring different Kaggle works, we came across various approaches for decoding the RLE format input to generate masks. Thanks to their visualizations, we discovered that the masks did not conform to rectangular or square shapes as shown in Fig 7. This observation indicates that using a simple rectangular bounding box (y1, x1, y2, x2) as output may not be the most suitable approach.

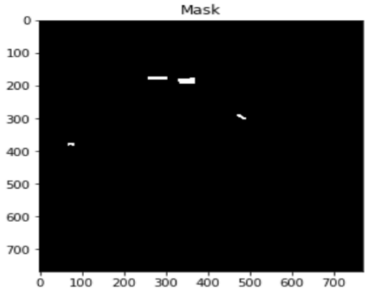

Fig 7. Mask problems with YOLO

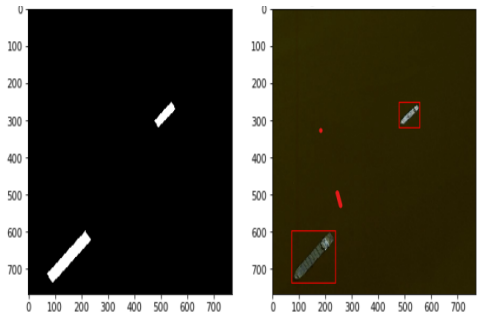

Fig 8. Mask problems with YOLOv3

We also encountered a YOLOv3 approach where the model correctly identified ships as shown in Fig 8, but the Intersection over Union (IoU) values were very low, averaging only 13%. The primary reason for this was the small size of the ship in the image (approximately 1% of the whole image). Even a small discrepancy in the bounding box's placement had a significant impact on the IoU calculation. In the example figure displayed here, a ship is depicted in a slant position. The overall box represents the YOLO detection, while the smaller border around the ship represents our desired mask. The additional space between the mask and the border caused the IoU values to rise. Hence, a pixel-level approach (segmentation) is more appropriate than generating bounding boxes. Therefore, we opted for segmentation-based algorithms such as Mask R-CNN and U-Net, which offer better IoU performance.

Super Resolution Failure:

Initially, we believed that enhancing images using vision transformers (specifically SWINR) would improve object detection. We thought that image transformers could increase boundary clarity and edge preservation, thereby making objects more well-defined and preserving important edges, which would aid object detection models in localizing objects more effectively.

However, our attempt to apply this approach to a sample dataset yielded no significant results. Jacob Shermeyer and Adam Van Etten conducted experiments on super-resolution in satellite imagery [1] and found that if the ground sample distance (GSD) falls within the range of 15 to 30 cm, super-resolution techniques can improve accuracy by 13-36% in terms of mean Average Precision (MAP). However, for satellite images with a GSD greater than 30 cm, the super-resolution approach fails to deliver notable improvements. In our dataset, the GSD is 1 meter, which is why the super-resolution technique is not suitable and won't be effective for our purposes.

Project Description

The final workflow of our project is to perform comparative analysis on Masked RCNN (baseline), Mask RCNN with pre-trained MS COCO weights (Instance segmentation) and U-Net (semantic segmentation).

Mask R-CNN

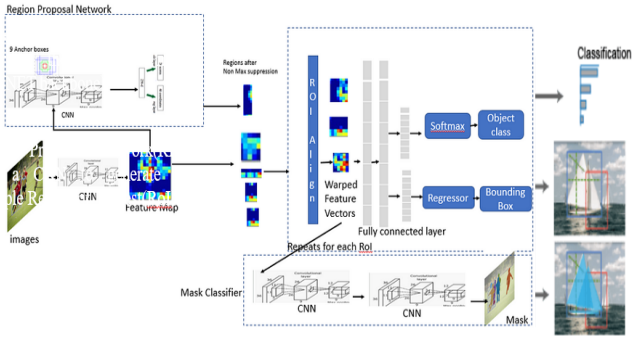

Fig 9. Mask R-CNN Architecture

Mask R-CNN is a powerful and versatile algorithm introduced by Kaiming He et al. in 2017. It extends the earlier Faster R-CNN object detection model, adding a branch for predicting segmentation masks on each Region of Interest (RoI), thereby enabling instance segmentation. The algorithm operates in two main parts: the first part involves the identification of RoIs, which are areas of the input image that likely contain an object. This process is handled by a Region Proposal Network (RPN) that generates potential bounding boxes or "proposals" that might contain an object. Mask R-CNN architecture is shown in Fig 9.

In the endeavour to develop an effective image segmentation model, we employed a sequence of systematic procedures. Firstly, our model's input was prepared by taking images and corresponding masks in Run-length encoding (RLE) format. The masks were decoded, and images were resized to an appropriate input size for the model. This process ensures the effective and accurate representation of the original image data in a format that our model can process.

The next step involved passing the input through a Convolutional Neural Network (CNN) backbone network, which was responsible for extracting features from the input image. This backbone network was pre-trained on a large dataset like ImageNet to avail a comprehensive feature extraction mechanism. We employed the ResNet 50 architecture as our backbone network, a popular choice for its balance of efficiency and depth. Following this, we utilized a Region Proposal Network (RPN), a separate network that generates a set of region proposals using feature maps from the backbone network. By placing anchor boxes of varying sizes at

different locations in the feature maps, the RPN identifies potential object bounding boxes.

Subsequently, these region proposals were processed using Region of Interest (RoI) Pooling. This operation re-extracts fixed-sized feature maps for each region by dividing each proposed region into a grid of sub-regions and applying max pooling within each sub-region. This results in robust and scale-invariant features for each region proposal. The feature maps generated by RoI pooling were then subjected to classification and bounding box regression, through a series of fully connected layers. The classification branch of the network predicts the probability of each proposed region belonging to the target object class. At the same time, the regression branch refines the coordinates of the bounding boxes for accurate localization of the objects.

Finally, a mask prediction branch is utilized to generate a binary mask for each proposed region, thus providing pixel-wise segmentation of the object within the region. This branch uses a set of convolutional layers applied to the features extracted from the RoI pooling layer, allowing for precise demarcation of the object. The network's learning was facilitated using a multi-task loss function, composed of classification loss, bounding box regression loss, and mask prediction loss. These losses were calculated based on the network's predictions and the ground truth annotations provided in the training data. Then, the gradients of this loss function were computed, and backpropagation was performed to update the network's parameters, resulting in a continually improving model. Thus, this sequence of operations allowed us to segment the images effectively and accurately.

The inference or prediction phase of our image segmentation process involves the trained Mask R-CNN model processing an input image. Firstly, the image is passed through the backbone network to extract relevant features. Utilizing these features, the Region Proposal Network (RPN) then generates potential regions of interest within the image. The subsequent application of Region of Interest (RoI) Pooling standardizes these proposed regions into fixed-size feature maps. Once the regions have been processed, the model proceeds to make several key predictions. It predicts the object classes, delineates the bounding boxes, and creates masks for the proposed regions. These masks are typically used in the computation of Intersection over Union (IoU) calculations, a common metric for evaluating the accuracy of object detection models.

However, the Mask R-CNN framework provides built-in functions that incorporate all three prediction parameters - object classes, bounding boxes, and masks - in the computation of overlaps. These overlaps, in turn, are used to calculate IoU and mean Average Precision (MAP), another crucial metric for assessing the performance of detection models. This comprehensive approach to prediction allows for a holistic view of object detection, ensuring that our model's performance is evaluated in the most robust and thorough manner possible.

Mask R-CNN with pre-trained MS COCO weights

The Mask R-CNN model was pretrained on the MS COCO dataset, which enabled it to learn meaningful representations and develop object detection capabilities. The pretrained weights for this model are publicly available on a GitHub repository [2]. We leveraged these weights and further trained the model on our dataset, which resulted in improved performance. Initially, we trained only the last fully connected layer, but we observed better loss curves when training all the weights of the model.

In the code comments, we have provided explanations for all the hyperparameters used. Some of the important hyperparameters include:

BACKBONE: 'resnet50' - identifies the feature map backbone architecture.
RPN_ANCHOR_SCALES: [8, 16, 32, 64] - defines anchor boxes of different sizes used at the Region Proposal Network (RPN) step.
TRAIN_ROIS_PER_IMAGE: 64 - specifies that the model can consider up to 64 regions of interests in an image during training.
DETECTION_MIN_CONFIDENCE: 0.95 - sets the minimum confidence threshold required for classifying an object as a ship.

We trained the two models with around 40k images with ships and around 20k images without ships for the models to identify the images with ships and without ships more precisely.The train and validation losses for Masked RCNN is shown in Fig 10 and Masked RCNN with MS COCO weights is shown in Fig 11

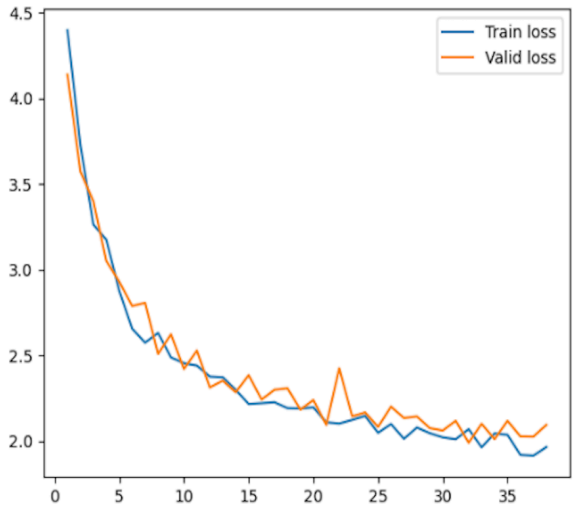

Fig 10. Masked R-CNN Train vs Val Loss Graph

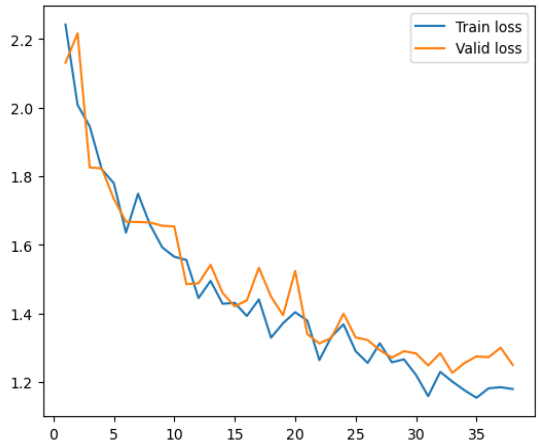

Fig 11. Pre-trained Mask R-CNN Train vs Val Loss Graph

The mask prediction(bottom image) for a random input image (top image) can be shown in the Fig 12.

Fig 12. Original Image (Top), Predicted Mask (Bottom)

U-NET

U-Net is a convolutional neural network designed primarily for biomedical image segmentation. Introduced by Ronneberger et al. in 2015 [10], U-Net was originally designed to segment neuronal structures in electron microscopic stacks. The U-Net architecture has since been widely adopted for various tasks due to its efficiency and accuracy. U-Net is a popular model used for semantic segmentation tasks. In this context, 'semantic segmentation' means that the model classifies each pixel of an image into a category, such as "ship" or "background", but does not distinguish between different instances of the same category (e.g., different individual ships in a single image). In our project, U-Net takes an input satellite image with ship and produces a binary mask, where 1 indicates a ship and 0 indicates the background. This is a semantic segmentation task, where all pixels belonging to the same class (in this case, ship) are labeled the same.

The architecture of U-Net shown in Fig 13 is built upon the principles of a convolutional neural network (CNN). The unique U-shaped design of the U-Net architecture includes a contracting path and an expansive path. The contracting path, or the encoder, captures the context in the image, while the expansive path, or the decoder, enables precise localization using transposed convolutions. This combination of context capture and localization allows U-Net to produce high-quality segmentation maps from input images.

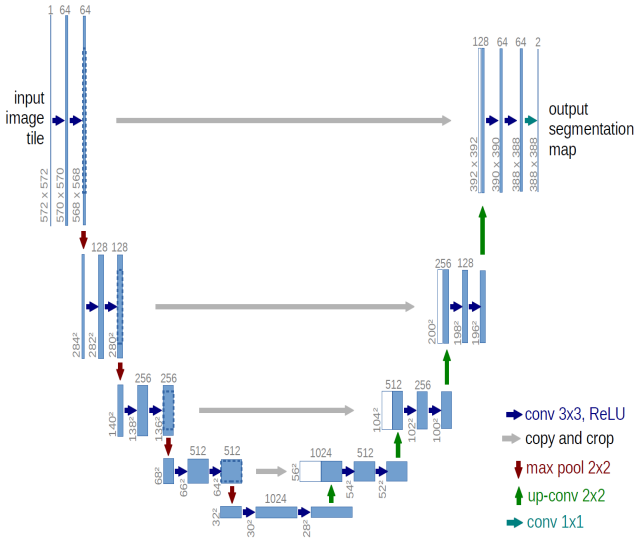

Fig 13. U-Net Architecture

It consists of 23 convolutional layers and incorporates skip connections to combine low-level and high-level features, enabling accurate localization. The contracting path follows the typical architecture of a convolutional network. It consists of the repeated

application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for down sampling. At each down sampling step, the number of feature channels is doubled. The expansive path is where the U-Net starts to differ from the traditional FCN. Every step in the expansive path involves an up sampling of the feature map, followed by a 2x2 convolution (works as "up-convolution"), a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer, a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. For example, in a binary classification problem like ship segmentation, the 1x1 convolution would map the 64-component vector to 2 classes: ship and no-ship (background in our case). The U-Net architecture also introduces skip connections between layers at the same level in the contracting and expansive paths. These skip connections forward the feature maps from the contracting path to the corresponding layers in the expansive path, essentially allowing the network to use the features at multiple levels of abstraction. This feature is crucial for the U-Net model's success as it enables the network to consider both the global and local features when generating the segmentation map.

We trained U-Net with around 40k images with ships and around 20k images without ships for the model to identify the images with ships and without ships more precisely. Trained the U-Net model with the parameters mentioned in Fig 14 and achieved train vs validation loss graph as mentioned in Fig 15.

```
import tensorflow_addons as tfa
EPOCHS = 12
STEPS_PER_EPOCH = TRAIN_LENGTH // BATCH_SIZE

optimizer = tfa.optimizers.RectifiedAdam(
    learning_rate=0.005,
    total_steps=EPOCHS * STEPS_PER_EPOCH,
    warmup_proportion=0.3,
    min_lr=0.0001,
)
optimizer = tfa.optimizers.Lookahead(optimizer)

loss = tf.keras.losses.CategoricalCrossentropy()
mIoU = IoU(num_classes=2, target_class_ids=[0, 1], sparse_y_true=False,
           sparse_y_pred=False, name='mean-IoU')
f2score = F2Score(name = 'f2_score')
MAP_metric = MAP_metric(num_classes = 2,  name='mean_average_precision',)

model = unet()
model.compile(optimizer=optimizer,
              loss=loss,
              metrics=[mIoU,f2score,MAP_metric],)
```

Fig 14. U-Net Parameters

Our U-Net model is trained using a categorical cross-entropy loss function, which is typically used for binary classification tasks, which measures the dissimilarity between the predicted segmentation and the ground truth. However, U-Net introduces a weighting scheme into the loss function, which allows the model to pay more attention to certain pixels over others. We used Rectified Adam (RAdam) Optimizer which is a variant of the popular Adam

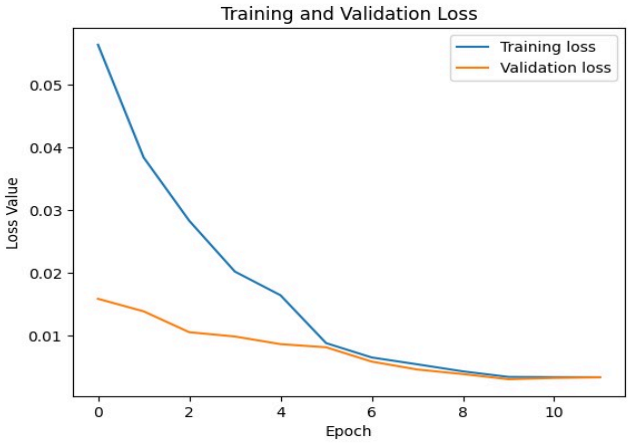

Fig 15. U-Net Training vs Validation Loss Graph

(Adaptive Moment Estimation) optimization algorithm. RAdam addresses the “warm-up” issue in Adam by introducing a term that rectifies the variance of the adaptive learning rate. It calculates the length of the approximated variance and dynamically incorporates it into the learning rate which makes RAdam more robust to the initialization of the network and the choice of the learning rate. During our presentation, due to an error in the code, only 10k images went into model training which resulted in the model converging at the 1st epoch itself which is identified and rectified in the report.

U-Net predicted output with the boundary boxes for the ships are shown in the Fig 16. We can clearly see that the model is performing better when the images are clear, as the image gets blurred or covered with fog the model is unable to predict the mask accurately.

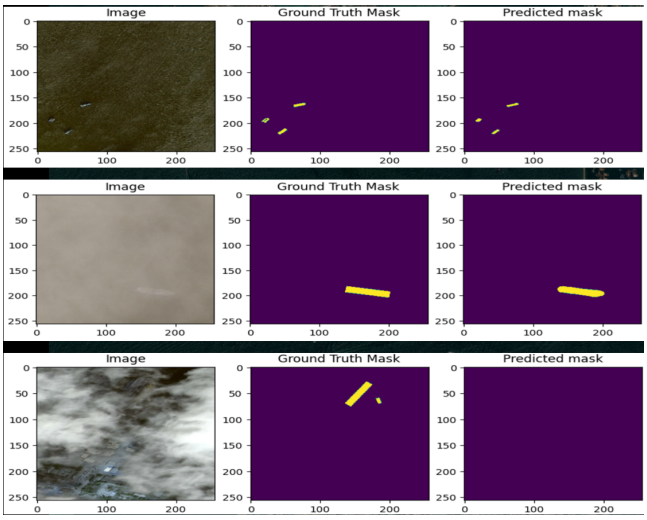

Fig 16. U-NET Original Image vs Predicted Mask

Comparative Analysis & Conclusion

Our final comparative analysis between the models is given in the Table 1. We considered evaluation metrics like Intersection Over Union(IoU), Mean Average Precision(MAP), F2 Score & run times between each model to perform comparative analysis.

	MAP	IoU	F-2	Runtime
Masked R-CNN (Baseline)	0.334	0.298	0.653	4:08:22
Masked R-CNN with MS COCO Weights	0.592	0.405	0.736	4:48:03
Masked R-CNN with MS COCO Weights (Hyperparameter tuned)	0.632	0.413	0.787	4:33:18
U-NET	0.347	0.329	0.669	7:15:45

Table 1: Model Comparative Analysis

All the Masked R-CNN models are trained for 40 epochs and ran faster compared to the U-Net model which was trained for 12 epochs but took around 7 hrs to complete the model training. Also, we can observe that since there are only 2 classes in our project i.e., ship & background, semantic segmentation (U-Net) worked slightly better than instance segmentation (Mask R-CNN baseline). Our best model turned out to be Mask R-CNN with pre-trained MS COCO weights with a F2 score of 0.787 as the MS COCO dataset has a huge collection of images that help the model to perform better. Finally, to conclude semantic segmentation works better but we need to train the model for longer durations, whereas to find a trade-off between time and accuracy, instance segmentation is the best possible task for our current dataset & project.

Future work

In our future work, we will focus on enhancing ship detection accuracy by exploring ensemble models and combinations of detection algorithms. By training multiple models with different architectures or variations of Mask R-CNN and U-Net and combining their predictions, we aim to improve robustness and overall performance. Additionally, we will investigate the SEEM framework from Hugging Face, specifically designed for semantic segmentation evaluation, to gain deeper insights into model performance and identify areas for improvement. Furthermore, we plan to implement the Detectron2 library for Mask R-CNN, which offers advanced features and optimizations, to further optimize our model for ship detection and segmentation. Lastly, we will explore the adaptation of the SAM AI model by Meta AI research to ship detection and segmentation tasks, leveraging its ability to segment a wide range of objects and scenes. These future directions aim to enhance accuracy, performance, and evaluation of our ship detection and segmentation models, contributing to the development of more effective solutions in maritime safety and surveillance.

GitHub Link & Instructions

GitHub Repository : DS5500 Code Base
Instructions to run the code:
1. Download dataset from kaggle link
2. Save all the contents in your local or google drive with name "AirbusShipDetection".
3. Check the data location path in 2nd block of Data Wrangling part. Once these paths are updated everything else runs normally
4. Since dataset is large. Downloading and uploading might take a while. Consider using drive desktop for data upload!

REFERENCES

[1] Pretrained-COCO weights: https://github.com/matterport/Mask_RCNN/releases/download/v2.0/mask_rcnn_coco.h5
[2] The Effects of Super-Resolution on Object Detection Performance in Satellite Imagery: https://arxiv.org/abs/1812.04098
[3] https://glassboxmedicine.com/2020/01/21/segmentation-u-net-mask-r-cnn-and-medical-applications/
[4] Airbus Ship Detection Challenge | Kaggle. (n.d.). https://www.kaggle.com/competitions/airbus-ship-detection
[5] Gandhi, R. (2018, December 3). R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms. Medium. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e
[6] Rich feature hierarchies for accurate object detection and semantic segmentation: https://a Rrxiv.org/pdf/1311.2524.pdf
[7] Fast RCNN by Ross Girshick, R. (2015, April 30). Fast R-CNN. arXiv.org.:https://arxiv.org/abs/1504.08083
[8] Super resolution using SWINIR transformer: htttps://github.com/JingyunLiang/SwinIR
[9] UNet — Line by Line Explanation: https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5
[10] U-Net: Convolutional Networks for Biomedical Image Segmentation by Olaf Ronneberger, Philipp Fischer, Thomas Brox: https://arxiv.org/abs/1505.04597
[11] Mask RCNN version 2 model repo: https://github.com/maxw1489/Mask_RCNN.git
[12] Mask RCNN by Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick: https://arxiv.org/abs/1703.06870
[13] Object detection using YOLO: challenges, architectural successors, datasets, and applications: https://link.springer.com/article/10.1007/s11042-022-13644-y