Implementation and comparison of U-Net and YOLO for ship detection

Margherita Bencini (s241773), Simone Carletti (s242168), Tomaz Gonçalves-Silva (s243023)

Introduction

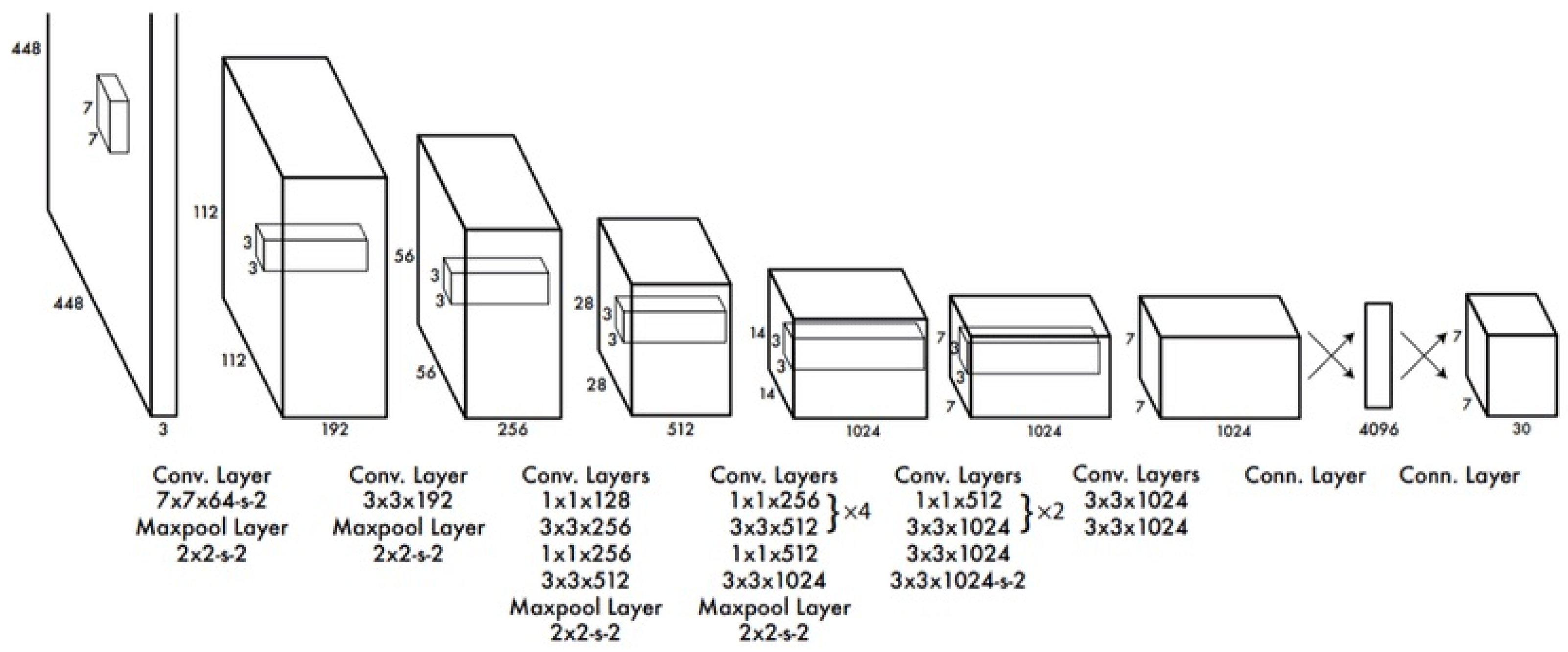
Shipping traffic is growing fast, leading to higher chances of infractions or incidents. This has compelled many organizations to have a closer watch over the open seas: it's here that image segmentation and detection come to the rescue. In this analysis, we chose to implement from scratch two milestone models, U-Net and YOLO, based solely off of their original papers. We tried many different configurations to reach the best implementation possible of both architectures, and compared their performance.

Data

- Publicly available as part of one of kaggle's competitions [1] and consists of 192556 images
- Very unbalanced, only 22% of images contain at least one ship
- As ships are small, only a tiny fraction of pixels can be actually used for segmentation

YOLOv1

We implemented YOLOv1 [2] from scratch. The architecture consists of 24 convolutional layers followed by 2 FC layers. We added dropout in the FC layers.



- The input image is divided into an SxS grid, each grid cell is responsible for detecting objects whose center fall within it
- Each cell predicts B bounding boxes, with each of them being defined by five components $(class, C, x_c, y_c, w, h)$.
- Each grid cell predicts the probability of each class given that an object is present in the cell

YOLO Parameters

$$
\begin{aligned}
L = \lambda_{coord} \sum_{i=1}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \Big[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \Big] \\
+ \sum_{i=1}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=1}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 \\
+ \sum_{i=1}^{S^2} \mathbb{1}_{i}^{obj} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$

$\boldsymbol{\lambda_{coord}}$ penalizes the errors in the predicted bounding box coordinates. To emphasize bounding box prediction quality, we experiment with bigger values.
$\boldsymbol{\lambda_{noobj}}$ reduces the weight for non-object confidence loss. As satellite images have sparse ships in large, empty backgrounds, we want to reduce false positives in the background, we experiment with smaller values.

Results for YOLOv1

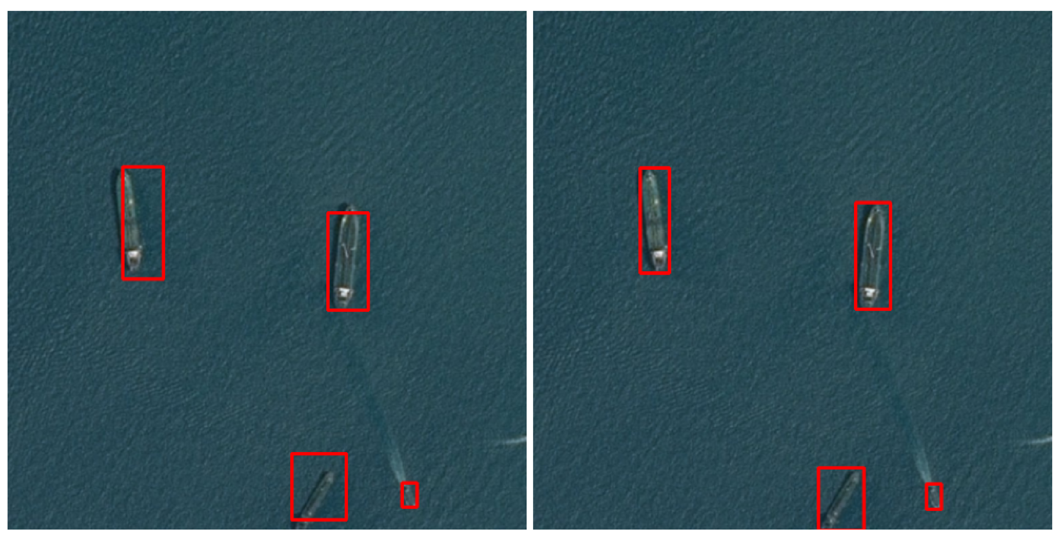


Figure 1: YOLO outputs vs. Ground Truth

After several tryouts, we achieved the following optimal metrics:

mAP = 0.98516	**DICE** = 0.99708

The above have been obtained by training 20 epochs with batch size of 16, learning rate = 0.00002, Adam optimizer with weight decay = 0.0005, dropout = 0.2 and $\lambda_{coord} = 5$, $\lambda_{noobj} = 0.3$

U-Net

We implemented a U-Net [3] from scratch, consisting of an encoder followed by a decoder, as shown in the figure below.

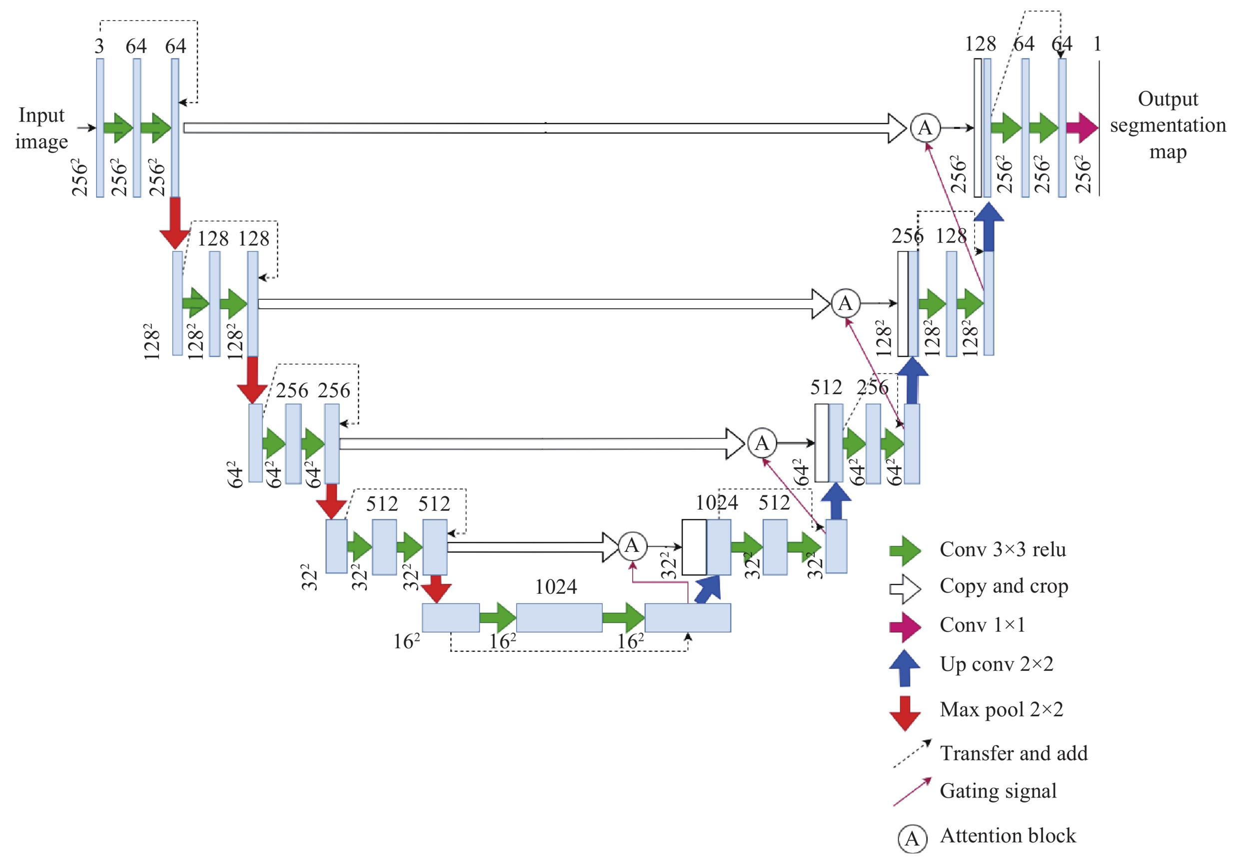


Figure 2: U-Net architecture diagram.

- **Encoder.** Applies two 3x3 convolutions with ReLU, followed by 2x2 max pooling for downsampling. Feature channels double at each step
- **Decoder.** Upsamples the feature map, applies 2x2 up-convolution to halve feature channels, and concatenates with skip connections. Two 3x3 convolutions with ReLU are applied, reducing feature channels

In addition to the original paper, we added batch-norm and *same* padding.

Results for U-Net

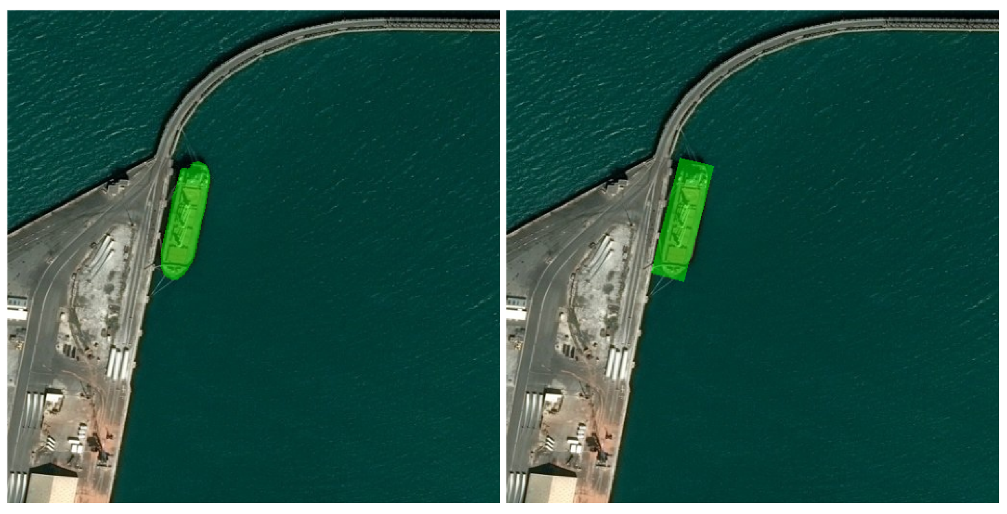


Figure 3: U-Net output mask vs. Ground Truth

For U-Net we experimented by changing the learning rate and batch size. We achieved the following optimal metrics:

Jaccard = 0.67321	**DICE** = 0.72788

The values above were obtained training for 15 epochs with a training batch size of 16 (validation batch size of 4), a learning rate of 0.0001 and Adam o ptimizer with no weight decay.

Conclusion and Comments

In this study, we implemented and compared U-Net and YOLO models for ship detection in satellite imagery. Both architectures showed robust performance, with YOLO emerging as the clear frontrunner.
While YOLO excels in generating precise bounding boxes, U-Net offers the advantage of providing exact segmentation of ships.
For this reason, one may argue that YOLO numerically wins, but it is *playing a different game*.

References

[1] Airbus. Airbus ship detection challenge, 2018. URL https://www.kaggle.com/competitions/airbus-ship-detection/data.
[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection, 2016. URL https://arxiv.org/abs/1506.02640.
[3] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.