'ALEXANDRU IOAN CUZA UNIVERSITY' IAŞI

FACULTY OF COMPUTER SCIENCE

MASTER THESIS

Applying the Expectation - Maximization algorithm for aerial view ship detection

proposed by

Radu-George Rusu

Session: *July, 2019*

Scientific coordinator

Associate Professor, Dr. Liviu Ciortuz

ALEXANDRU IOAN CUZA UNIVERSITY IAȘI

FACULTY OF COMPUTER SCIENCE

Applying the Expectation - Maximization algorithm for aerial view ship detection

Radu-George Rusu

Session: *July, 2019*

Scientific coordinator

Associate Professor, Dr. Liviu Ciortuz

Contents

Introduction

Machine Learning (ML) is the field that addresses the problem of how to use historical data to produce accurate models or predictions for new data. A Machine Learning Algorithm, based on data-points from the historical data or train data, produces a model that can be later used for giving the best approximation for new inputs or test data. With the increase in computing resources in the recent years, a lot of ML algorithms have became usable in practice in various situations.

The process of computing the model is called training or learning. Starting from this, the ML learning processes can be split into two main categories: *supervised learning* and *unsupervised learning*. In *supervised* learning the training data set (historical data), has labels, the classification (number of classes and the model of each class) is known at the training phase. After the training, the computed model must classify the new inputs (test data) with the best class approximation of the input. When the unsupervised learning procedure is used, the classes of train data are not known from the beginning (data has no labels), but a certain model is enforced on them (number of classes, a distribution function that generated the data etc.), and the training phase produces the best parameters for the enforced model.

A linked domain with ML is computer vision. Recently, with the incentive for autonomous driving, automatic image/video moderation and others, the computer vision algorithms have become more used in practical environments. Three main problems can be highlighted from this field: *object classification*, *object detection* and *image segmentation*. Object classification is the process of assign a label to a certain image, linked to the object that is inside that picture (i.e. labeling and image with a cat with the label "Cat"). Object detection in images try to find the exact position of an object in an image (usually by giving bounding boxes for every object of interest in the picture). In most practical cases detection is used

together with classification. The image segmentation problem refers to grouping pixels that are "semantically similar".

This thesis take into consideration both supervised learning and unsupervised learning, and tries to solve a segmentation problem, in conjunction with object classification.

This thesis is structured to discuss the problem statement that is proposed for solving, the theoretical background needed, and two proposed solutions with the corresponding results and comparison between them.

The Airbus Ship Detection Challenge section 1 describes the origin of the data set used in this thesis, an analysis over how the data looks and how is distributed and a metric that will be used for later evaluation of the models. The theoretical background section 2 presents the models and the basic theory behind them, that will be used in the solutions.

The experiment setup section 3 is showing the thought process of the solution and the two solutions with differences in structure and example runs. The result section 4 is summarizing the results by the two models, and other base models, and compares them to one another. The last chapter 5 draws the conclusions of this thesis and sets the future improvements/experiments that can be conducted on the same problem.

Chapter 1

Airbus Ship Detection Challenge

This chapter presents the data set used for the experiment in this paper, the Kaggle competition that it was used in, and an analysis over how the data looks and can impact the experiments.

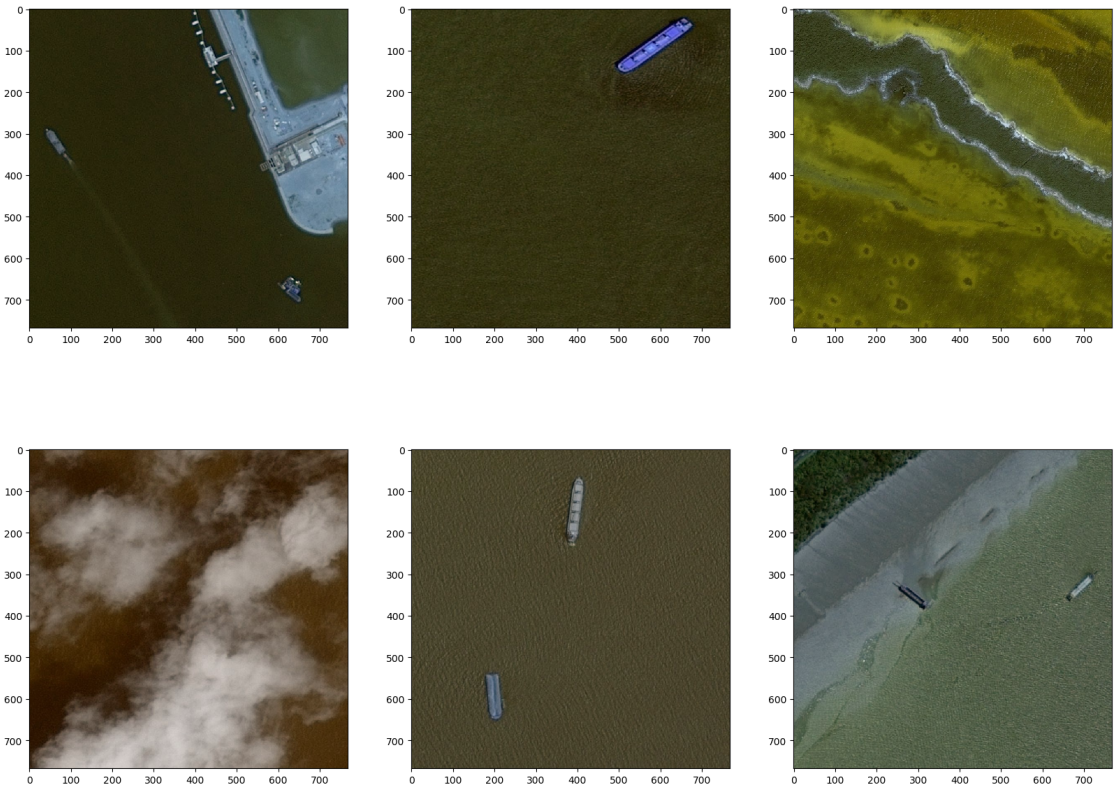

FIGURE 1.1: Training set examples

1.1 Airbus Ship Dataset

The Airbus Ship Dataset is an image dataset that was used in the Airbus Ship Detection Challenge [1] on www.kaggle.com. The aim of this competition is to detect ships inside an image taken from an aerial view. As the competition description says, this detection can help multiple organizations such as environmental organization and insurance companies to have knowledge of illegal activities done at sea. A few examples of the training set can be found in Figure 1.1.

1.2 Analysis

The competition provides both testing and training datasets in a large amount. The train set is composed of **231723** pictures, of size **768x768** pixels, in JPEG format, each of them having a set of pixels assigned as being ship pixels. To avoid specifying each pixel separately, for size reasons, the run length encoding is used. Every pixel is numbered from 1 to maximum size, top down, left right order. Pixel (1, 1) will have index 1, pixel (2, 1) index 2 etc. Run length encoding is formed of 2k numbers, the numbers in odd position (1 - indexed) specifies the starting index of a pixel ship sequence, and the ones in even position specifies the length of the sequence. For example, the sequence: *5 3 10 2*, encodes the ship pixels {*5, 6, 7, 10, 11*}. Those encodings are provided via an **.csv** file, with two columns, **ImageId** and **EncodedPixels**. Also, in this file, every ship is given as a separate entry, so a picture can appear multiple times in this file and at least once. (See table 1.1). The figure 1.2 shows the example images with their training masks.

ImageId	EncodedPixels
00003e153.jpg	NaN
0001124c7.jpg	NaN
000155de5.jpg	264661 17 265429 33 266197 33 266965 33 267733...
000194a2d.jpg	360486 1 361252 4 362019 5 362785 8 363552 10 ...
000194a2d.jpg	51834 9 52602 9 53370 9 54138 9 54906 9 55674 ...
000194a2d.jpg	198320 10 199088 10 199856 10 200624 10 201392...
000194a2d.jpg	55683 1 56451 1 57219 1 57987 1 58755 1 59523 ...
000194a2d.jpg	254389 9 255157 17 255925 17 256693 17 257461 ...
0001b1832.jpg	NaN
00021ddc3.jpg	108287 1 109054 3 109821 4 110588 5 111356 5 1...

TABLE 1.1: Train data set sample

The test set provided by the competition contains **15606** pictures of the same size as the ones in the training set, and the result that is submitted must also use the run length encoding described above.

On a first inspection of this dataset, it can be observed that there are more pictures with no ship in them (0 encoded pixels), then the number of pictures with ships (as shown in table 1.2). It can be seen that for every picture that has at least one ship in it, there are two pictures with no ship in it.

Total	No Ship Pictures	Ship Pictures
231723	150000	81723

TABLE 1.2: Ship/no ship count training data

Figure 1.3 has a histogram of number of images grouped by number of ships inside an image. The number of pictures that have only 1 ship stands out as being around 35% from the total number of images with ship, and there are also pictures that contains up to 15 ships.

The last metric that should be analyzed in this dataset is the ship size in pixels, to measure the magnitude of the task. As the table 1.3 presents, the average size is around 1500 pixels, with half of the ships in the training set being under 408 pixels in size. Based on this, we can draw the conclusion that one ship that should be detected in the test set forms less than 0.0006% of the entire image.

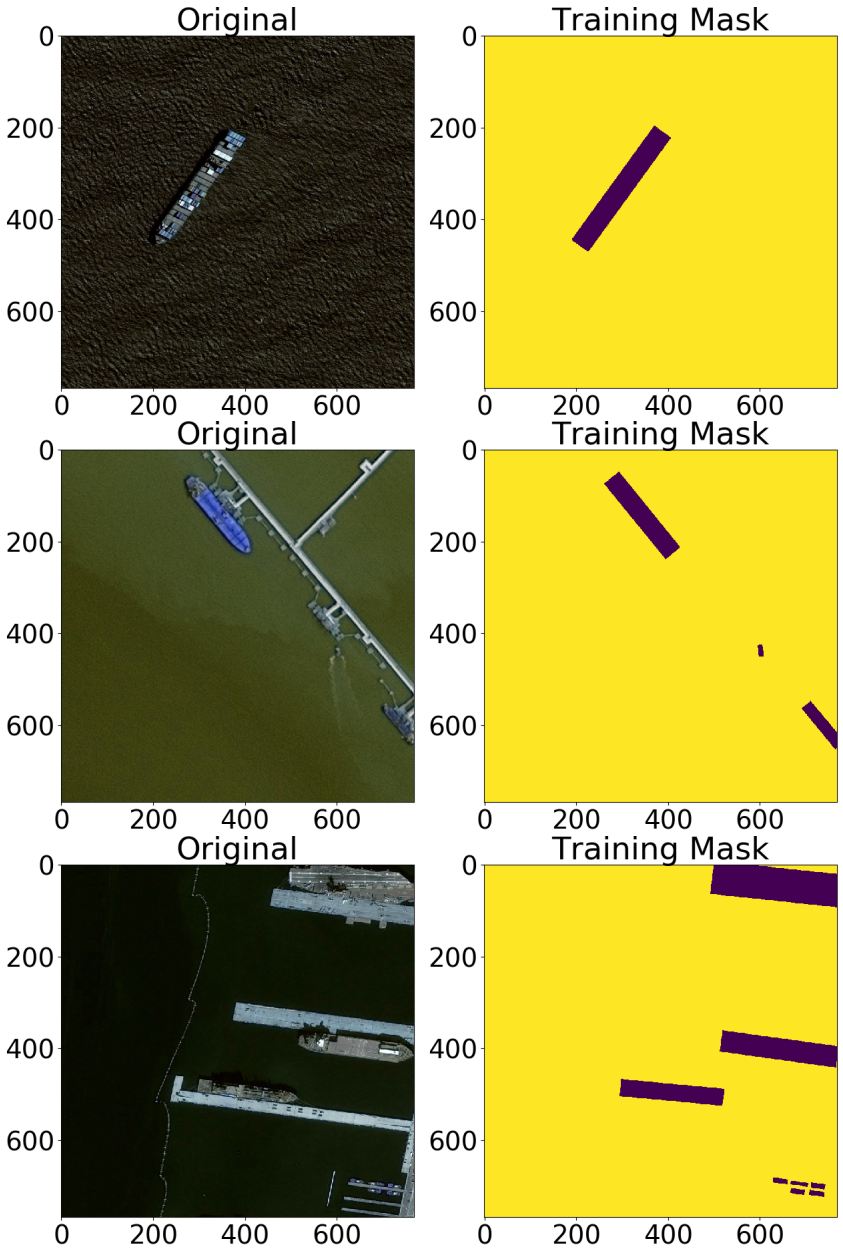

FIGURE 1.2: Original Images with Training Mask

Average	Min	Q1	Median	Q3	Max
1,567.40	2.00	111	408.00	1550	25,904.00

TABLE 1.3: Ship size in pixels

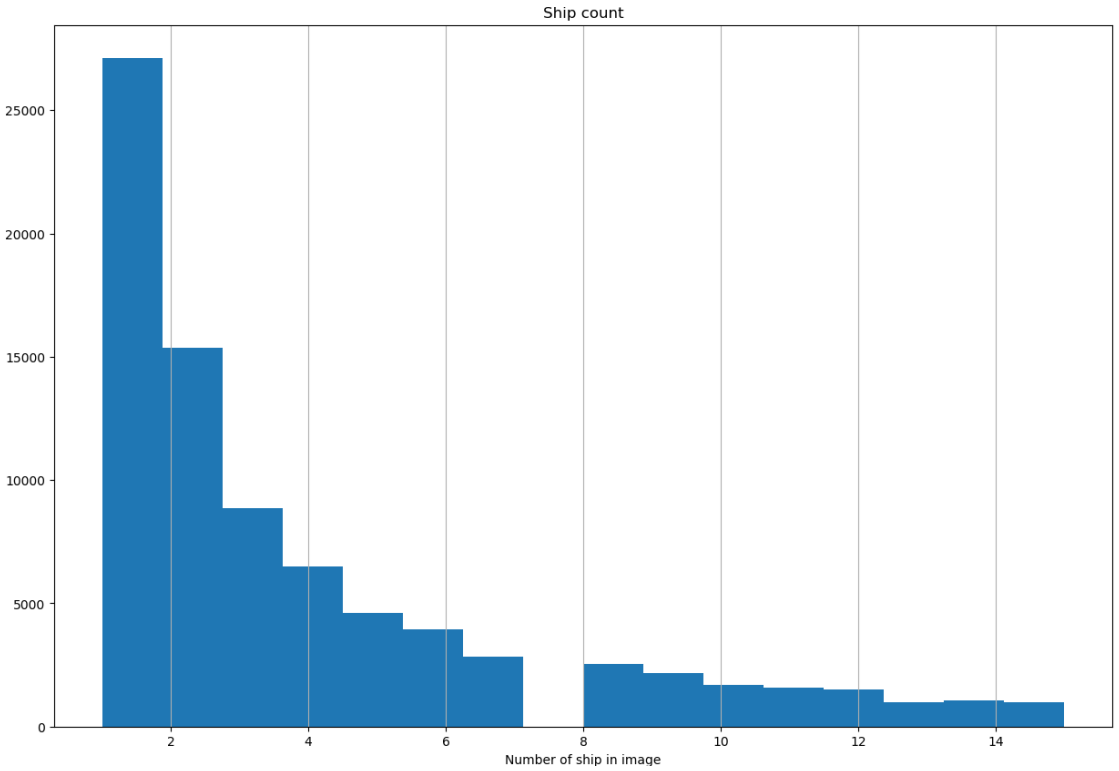

FIGURE 1.3: Number of ships/image

1.3 Evaluation Metric

To evaluate possible solutions for this competition, the F_2 score was used. This is defined as follows:

- We define a set of thresholds $T = \{0.5, 0.55, 0.6, \ldots 0.95\}$.
- For every threshold $t \in T$:
- the IoU between predicted pixels (pp) and true pixels (tp) of a ship is computed as $IoU = \frac{pp \cap pt}{pp \cup pt}$
- if this value is higher than t, then this item is consider a true positive (a hit)
- if this value is lower than t, then this is consider either a false negative (a predicted ship that has no true ship) or a false positive (there is no predicted ship for a true ship)
- the $F_2(t) = \frac{5TP(t)}{5TP(t)+4FN(t)+FP(t)}$ is computed
- the final score for an image is computed as $FS(image) = \frac{\sum_{t \in T} F_2(t)}{\left|T\right|}$

It is worth mentioning that this metric, in this form, penalizes more false negatives than false positives which means that it encourages rather not to predict a ship if the algorithm is not certain of the ship presence. This will have later implications in this paper. For this reason, the "no-machine algorithm" behaves really well, since this will give a percentage of images with no ships over a dataset.

Chapter 2

Theoretical background

This chapter will briefly describe the theoretical models used in the experiments presented in this paper.

2.1 Expectation-Maximization

The Expectation-Maximization (EM) is an algorithmic template for finding the Maximum Likelihood Estimation (MLE) parameters in statistical models which are based on missing or latent data.
The Maximum Likelihood Estimation problem can be defined as follows:

MLE Problem
Input
Y - set of observed data
p - probability distribution that we assume generated the Y data set
Output
$h^{MLE} = \underset{h}{argmax}\ L(Y|h) = \underset{h}{argmax}\ p(Y|h)$, where h is the set of distribution parameters

When the entire set Y is formed out of observable data, the above problem can be easily solved. A common approach for this instance of the problem, is to take the derivative of the log-likelihood function and solve it for h. However, real life data is not always fully observed, and inside of the given data we have latent variables. In this particular case, the above-mentioned method doesn't work anymore, and an EM approach can be used.

In the case of latent data we can consider the Y set as being a reunion between $X \cup Z = Y$, where X is the set of visible data, and Z is the set of hidden/latent data. The initial method doesn't work since it is impossible to compute $p(Z|h)$.

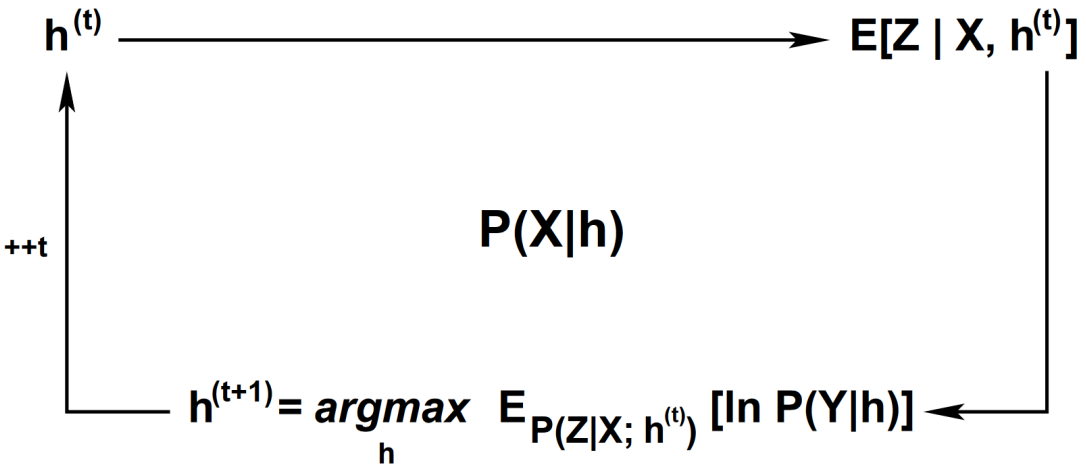

FIGURE 2.1: EM Algorithm [2]

The EM algorithm (as show in 2.1) applies an iterative method, that for each iteration computes two steps:

- E-Step: expectation of the latent variables, based on the distribution parameters from the previous iteration and the visible data ($\mathbf{E}[\mathbf{Z}|\mathbf{X}, \mathbf{h}^{(t)}]$).
- M-Step: based on the results from the E-Step, the latent variables, can be considered as observed and the h parameters for this step can be solved like in the classic MLE problem.

In the most basic form of this algorithm, the initial values for $h^{(0)}$ are randomly selected, and as a stop condition a maximum number of iterations is used, but these two conditions can vary from problem to problem.

2.2 Neural Networks for object detection

In computer vision and object detection/recognition in images, the classical neural networks involves a high number of weights that have to be learned, which in turn, makes the networks work slow on images that are bigger in size and sometimes

even impractical. The main advantage of the **Convolutional Neural Networks (CNN)** [3] is the idea of using the convolution operator (2.2), which drastically reduces the number of weights to be learned during training phase and makes them usable in practice.

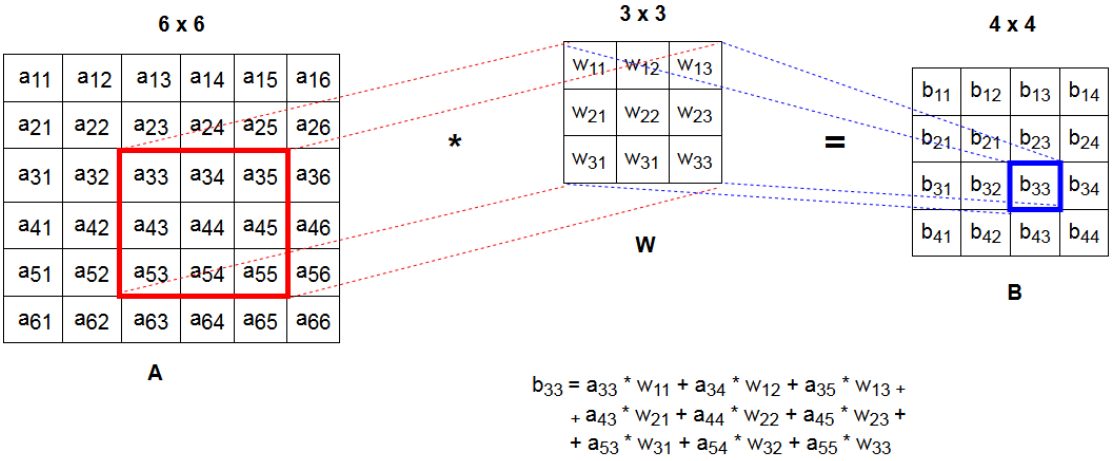

FIGURE 2.2: Convolution operator [3]

Other two popular operators are the AveragePooling and MaxPooling, which as the name suggests, just extract either the average or the maximum value of the values in the corresponding square. Although these operators doesn't have as much expression power as the linear combination 2.2, they don't have any weights to be learned during the training phase.

The convolution operator result size depends on the following variables:

- n - size of the input image (considering the input image is squared)
- f - size of the filter (also squared)
- p - padding of the initial image
- s - stride of the convolution operation

The convolution operator result size is:

$$(n, n) * (f, f) = \left(\left\lfloor \frac{n + 2 \times p - f}{s} + 1 \right\rfloor, \left\lfloor \frac{n + 2 \times p - f}{s} + 1 \right\rfloor \right)$$

To observe the advantages of CNN over the classical NN we can take an example of a network formed only of two layers. A first one of size 1024×1024 which can

be viewed as the input image, that is given in grayscale, and a second layer of size 512×512, which is a first feature extraction layer.

In a classical neural network every input from the first layer must be connected with every output on the second layer, and every connection has it's own weight, which, in the above case, would result in computing $1024 \times 1024 \times 512 \times 512 = 274,877,906,944$ weights. Also, in computer vision problem there is a need of more than one intermediate layer to extract the features inside an image.

In the case of CNNs, the only weights that have to be computed for a layer are the values inside the kernel, which in the above example can be considered a kernel of size $512 \times 512 = 262,144$ with a stride of 1 and no padding. The number of weights that will be learned it's way less than in the case of classical neural networks, which allows more layers, and makes the CNN more practical for this problem. Also, in practical cases, the convolution operator can be an average/maximum over the target values, instead of a linear combination, and there will be no weights to learn on that level.

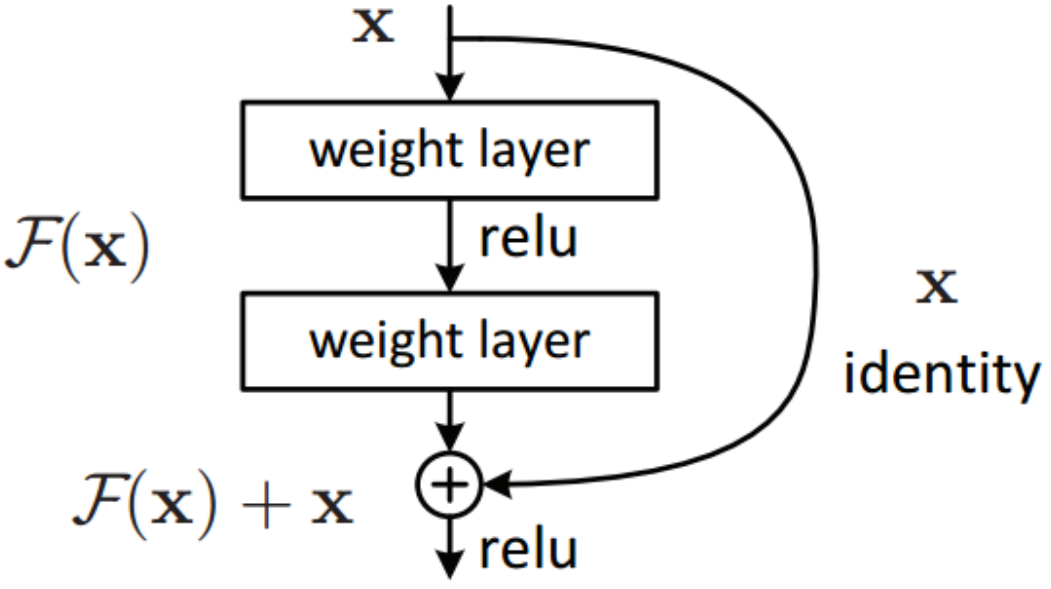

FIGURE 2.3: Residual Block [4]

Another issue that occurs in practical object detection in image cases, is the depth of the network. In order to learn complex features inside an image, a neural network needs a high number of layers, to have space for learning every feature, which, in practice, leads to the vanishing gradient problem [4]. This problem

makes the training of these networks hard or even impossible, depending on the task. To solve this **Residual Networks** [4] were developed.

The fundamental idea of these networks is the addition of the residual blocks (2.3). In this blocks, the inputs are once again added to the output value of a later layer, before the activation function, avoiding a "squishing" derivative, which in turn will result in a higher overall derivative of the entire block.

With these methods, neural networks structures with 34, 50 or 152 layers were developed and used as show in 2.4.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

FIGURE 2.4: Residual Networks Structure [4]

The neural networks described above behave well on object recognition and object detection problems. Another common case in image processing is the segmentation problem, where similar pixels inside a picture must be grouped together. In this particular case, the CNN might not work well in the form described above, because every pixel of the image can have it's own classification, and the convolution operator is specialized in contracting image features, hence reducing the number of outputs of the network. To solve this problem, **U-Nets** networks [5] have been developed. The name U-Net comes from the visual representation of the architecture (2.5) which resembles the U letter. There are two parts of this architecture, the contracting path, and the expansive path. The contracting path behaves as an usual convolutional neural network extracting features from the image. The expansive path uses the Transposed Convolution Operator [6], which helps the networks reconstruct the image at the initial resolution, from the feature vector that was computed by the contracting path.

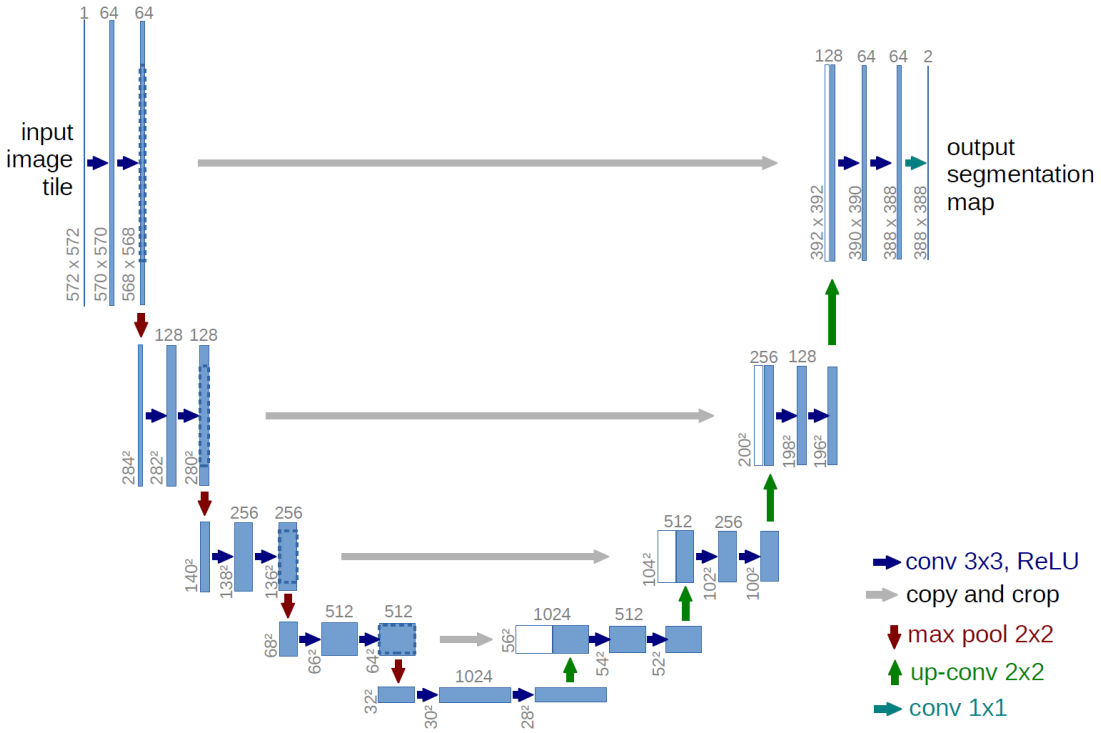

FIGURE 2.5: U-Net Architecture [5]

Chapter 3

Experiment setup

This chapter will present the thought process and the setup that was used to run the experiments on the dataset presented in Chapter 1.

3.1 Initial idea

Taking a high level look through the training data set, a general case can be seen: pictures are taken at sea (no port in site) and as such, they are conceived from water and ships only (which can divide pixels in two group as shown in image 3.1). If the images are moved into grayscale, and their histogram is computed (3.2), then the pixels of the image can be split into two groups, both generated by a Gaussian distribution, which will also give the ship/no-ship classification. This problem is a classical problem solved by using the EM algorithm.

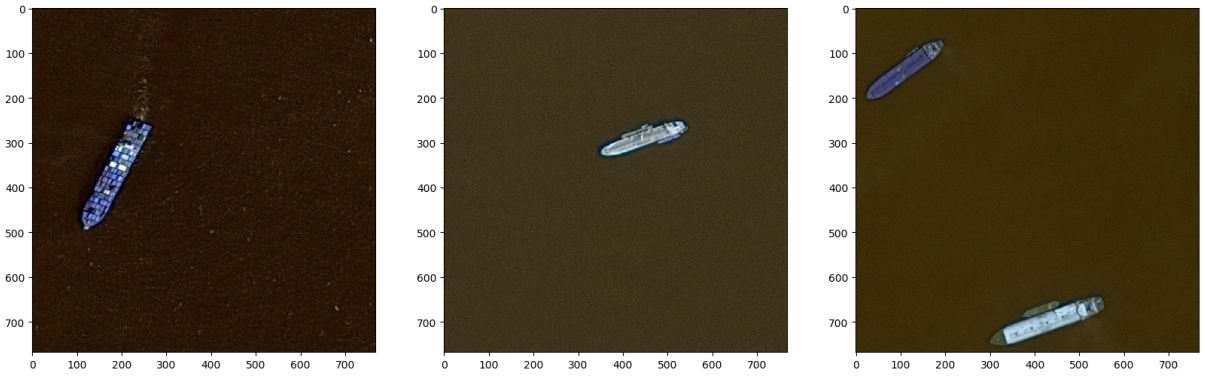

FIGURE 3.1: Ships at sea

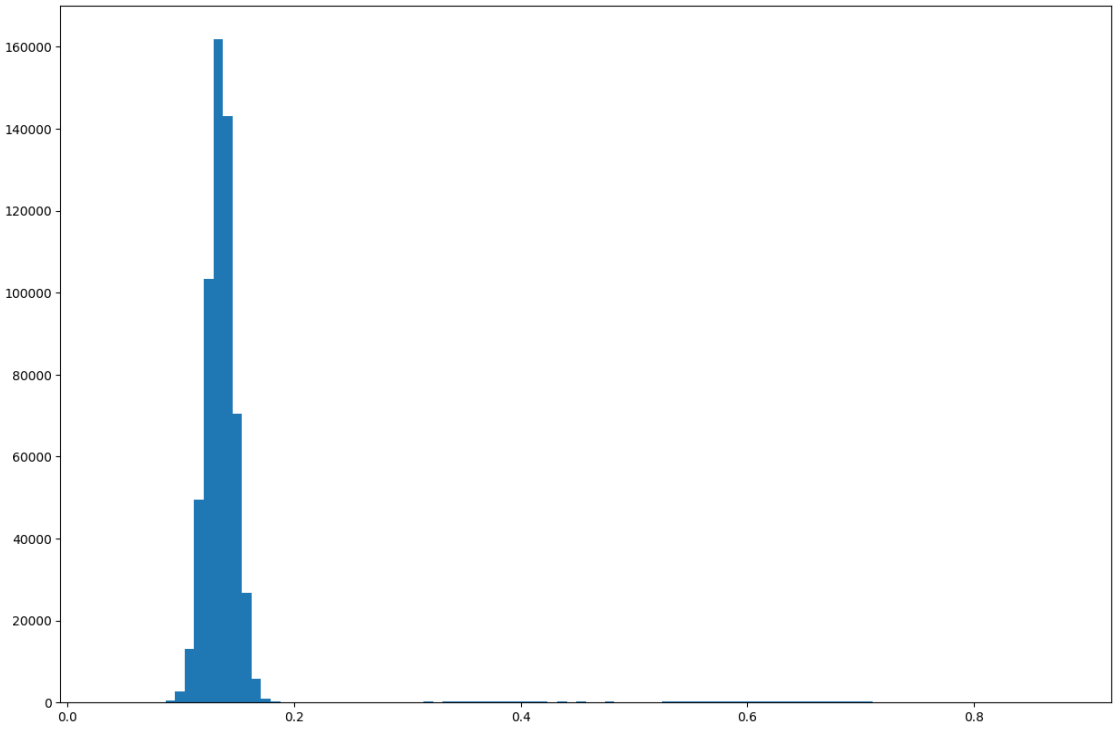

FIGURE 3.2: Histogram of grayscale at sea image

3.2 Expectation-Maximization

For the rest of this section the following notations will be used:

$\mathbf{p(x_i)}$	probability of the pixel i
$\mathcal{N}(\mathbf{x_i}\|\mu_\mathbf{s}, \sigma_\mathbf{s})$	PDF value of pixel i generated by the ship Gaussian distribution
$\mathcal{N}(\mathbf{x_i}\|\mu_\mathbf{b}, \sigma_\mathbf{b})$	PDF value of pixel i generated by the background Gaussian distribution
$\mathbf{P(G_s)}$	probability of ship pixels in image
$\mathbf{P(G_b)}$	probability of background pixels in images
$\mathbf{n}$	number of pixels in the image

TABLE 3.1: Notations

In order to apply EM to our problem, some adaptations must be done. First, we will be considering two random variables G_s and G_b, both with probabilities in $[0, 1]$, which will represent the presence of ship, respectively background in the image. Also the following relations stands for the two probabilities: $P(G_s)+P(G_b) = 1$. The hidden variables in this problem will be some random variables that assign every pixel either to the ship group or background group. Also, considering the

observation from the beginning of this section, the value of every pixel will be a probability generated by a univariate mixture of two Gaussians as follows:

$$p(x_i) = \mathcal{N}(x_i|\mu_s, \sigma_s)P(G_s) + \mathcal{N}(x_i|\mu_b, \sigma_b)P(G_b)$$

3.2.1 E-Step

The likelihood of the data will be computed as follows:

$$L = \prod_{i=1}^{n} p(x_i) = \prod_{i=1}^{n} \mathcal{N}(x_i|\mu_s, \sigma_s)P(G_s) + \mathcal{N}(x_i|\mu_b, \sigma_b)P(G_b)$$

Next we compute the mean of the likelihood for this iteration:

$$Q(h|h^{(t)}) = E_{P(G|x,h^{(t)})}\left[\prod_{i=1}^{n} \mathcal{N}(x_i|\mu_s, \sigma_s)P(G_s) + \mathcal{N}(x_i|\mu_b, \sigma_b)P(G_b)\right]$$

$$= \prod_{i=1}^{n} E_{P(G|x,h^{(t)})}\left[\mathcal{N}(x_i|\mu_s, \sigma_s)P(G_s) + \mathcal{N}(x_i|\mu_b, \sigma_b)P(G_b)\right]$$

$$= \prod_{i=1}^{n} E_{P(G|x,h^{(t)})}\left[\mathcal{N}(x_i|\mu_s, \sigma_s)P(G_s)\right] + E_{P(G|x,h^{(t)})}\left[\mathcal{N}(x_i|\mu_b, \sigma_b)P(G_b)\right]$$

$$= \prod_{i=1}^{n} \mathcal{N}(x_i|\mu_s, \sigma_s)E_{P(G|x,h^{(t)})}P(G_s) + \mathcal{N}(x_i|\mu_b, \sigma_b)E_{P(G|x,h^{(t)})}P(G_b)$$

The value for the $P(G_s)$ and $P(G_b)$ must be computed considering that:

$$P(G_s) = \frac{1}{n}\sum_{i=1}^{n} P(G_s|x_i)$$

$$P(G_b) = \frac{1}{n}\sum_{i=1}^{n} P(G_b|x_i)$$

which leads to:

$$E_{P(G|x,h^{(t)})}P(G_s) = \sum_{i=1}^{n} E_{P(G|x,h^{(t)})}P(G_s|x_i)$$

$$E_{P(G|x,h^{(t)})}P(G_b) = \sum_{i=1}^{n} E_{P(G|x,h^{(t)})}P(G_b|x_i)$$

The ML estimation of the above elements is as follows [7]:

$$E_{P(G|x,h^{(t)})}P(G_s|x_i) = \frac{\mathcal{N}(x_i|\mu_s^{(t)}, \sigma_s^{(t)})P(G_s)^{(t)}}{\mathcal{N}(x_i|\mu_s^{(t)}, \sigma_s^{(t)})P(G_s)^{(t)} + \mathcal{N}(x_i|\mu_b^{(t)}, \sigma_b^{(t)})P(G_b)^{(t)}}$$

$$E_{P(G|x,h^{(t)})}P(G_b|x_i) = \frac{\mathcal{N}(x_i|\mu_b^{(t)}, \sigma_b^{(t)})P(G_b)^{(t)}}{\mathcal{N}(x_i|\mu_s^{(t)}, \sigma_s^{(t)})P(G_s)^{(t)} + \mathcal{N}(x_i|\mu_b^{(t)}, \sigma_b^{(t)})P(G_b)^{(t)}}$$

3.2.2 M-Step

At this point, considering the previous section, we have a form for the expectation of the likelihood function, so the Maximization step can be applied. By taken the above formulas and computing the derivatives based on the parameters we obtain the following actualization rules [7]:

$$\mu_s^{(t+1)} = \frac{\sum_{i=1}^{n} P(G_s|x_i) * x_i}{\sum_{i=1}^{n} P(G_s|x_i)} \qquad \mu_b^{(t+1)} = \frac{\sum_{i=1}^{n} P(G_b|x_i) * x_i}{\sum_{i=1}^{n} P(G_b|x_i)}$$

$$\sigma_s^{(t+1)} = \frac{\sum_{i=1}^{n} P(G_s|x_i)(x_i - \mu_s^{(t)})^2}{\sum_{i=1}^{n} P(G_s|x_i)} \qquad \sigma_b^{(t+1)} = \frac{\sum_{i=1}^{n} P(G_b|x_i)(x_i - \mu_b^{(t)})^2}{\sum_{i=1}^{n} P(G_b|x_i)}$$

$$P(G_s)^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} P(G_s|x_i) \qquad P(G_b)^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} P(G_b|x_i)$$

The initialization values for the EM parameters were given as a random value between 0 and 1, for μ and σ. The $P(G_b)$ had an initial value of 0.9 and $P(G_s)$ a value of 0.1, since we expect to have more background than ships in an image. In order to stop the algorithm, a maximum iteration value of 12 was used, without any other restrictions. Parameter learning was executed on a resized (256×256) for performance reasons, and the classification phase was ran on the full-size image. After the parameters were computed, the following rule was used to classify pixels as either ship or background:

- if $P(G_s|x_i) \geq P(G_b|x_i)$ then classify x_i as **ship**

- if $P(G_s|x_i) < P(G_b|x_i)$ then classify x_i as **background**

Before applying the EM algorithm, all images were converted to grayscale, and normalized to $[0, 1]$, so that every pixel has only one characteristic value and not 3, like in the RGB format. Another preprocessing technique used was a Gaussian filter with the purpose of smoothing the edges in the picture.

3.2.3 EM results

After applying the EM algorithm over a subset of images we obtain results as shown in 3.3.

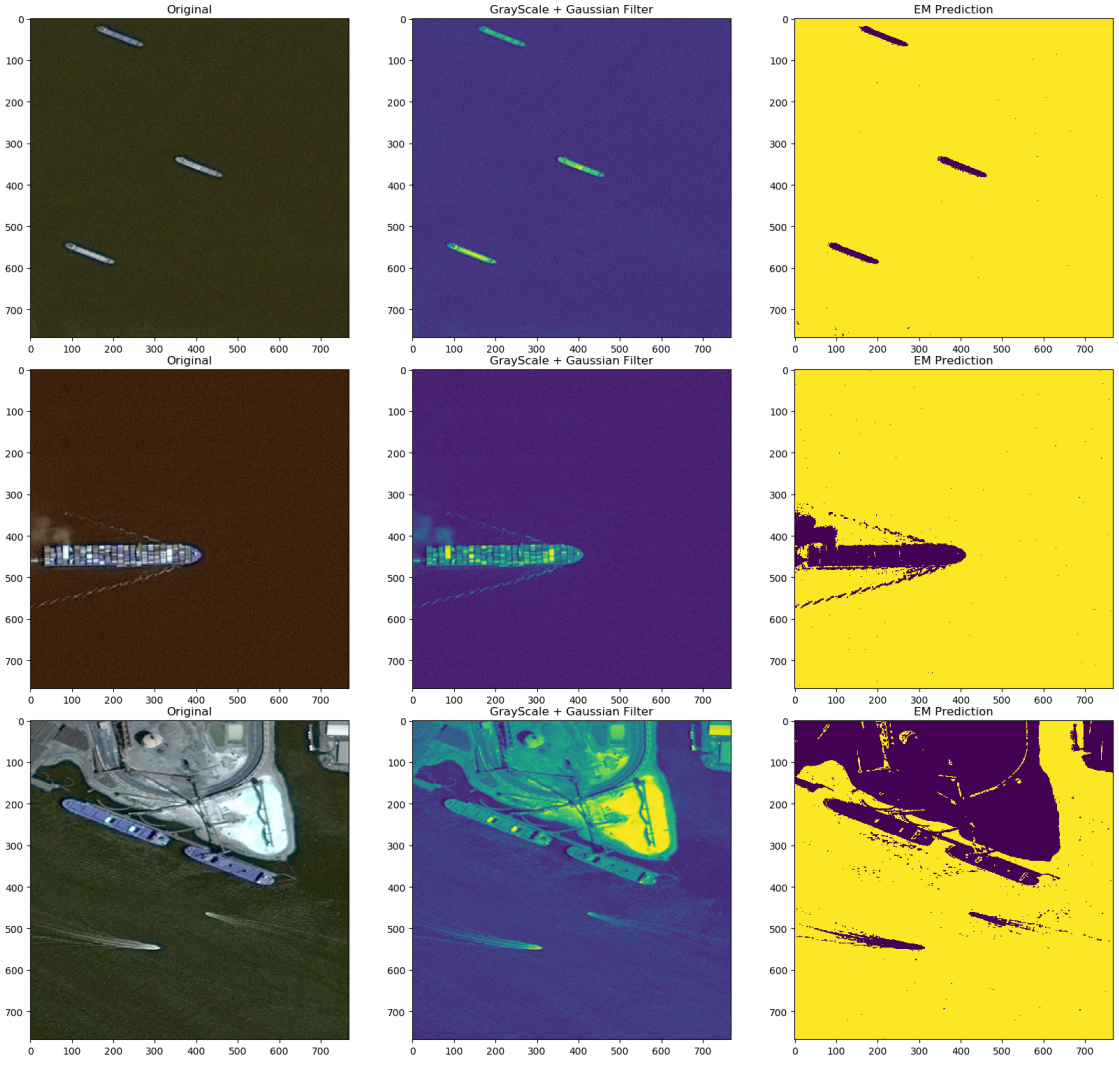

FIGURE 3.3: Initial Results

It can be observed that in "at sea" pictures, the algorithm behaves quite good (although it still predicts some pockets of pixels that are not ships), but in the

port pictures the algorithm detects the port as being a ship as well. Also, because the algorithm enforce that every pixel is generated by the formula specified earlier, in case of no-ship picture this algorithm will still detect ships.

After this steps, 3 main caveats of the EM algorithm have been identified:

1. the algorithm is classifying pixels in images that have no ship (the algorithm is not able to distinguish this particular case) **(P1)**
2. the algorithm is classifying small pockets of pixels as ship (usually waves that visually distinguish themselves from the rest of the water) **(P2)**
3. in the images that have ports, the algorithm identifies the port pixels as being ships (mainly because of the color similarity between them) **(P3)**

3.3 Region Proposal Solution

Since the EM algorithm by itself had the caveats mentioned in 3.2.3, a pipeline 3.4 around the algorithm was developed. To solve **(P1)** a Residual Network (with a ResNet34 architecture [4]) was used for classifying the images, in images with ships and with no ships. Using this, the EM algorithm could be ran, only on the images that are certain to have ships. The network was train on **10000** images, plus rotation preprocessing with -90 and 90 degrees, randomly selected from the train set of the initial problem.

After this phase, on the images classified as ship images, the EM algorithm was ran. In order to solve **(P2)**, a de-noising technique was applied on the image result. The fast de-noising Nl means from cv2 library was chosen, which helps removing small pockets of isolated pixels, and also smoothen the edges of the ships. The method is applied for each pixel, by taken a surrounding area of the target pixel and averaging it with similar areas in the image.

Finally, in the third phase, another Residual Network was used in order to remove the ship labels from port pixels. This is a more difficult issue to solve, since ports have a color scheme really similar to ships.

- First we split the EM predicted image in 25 crops, of size 224×224.

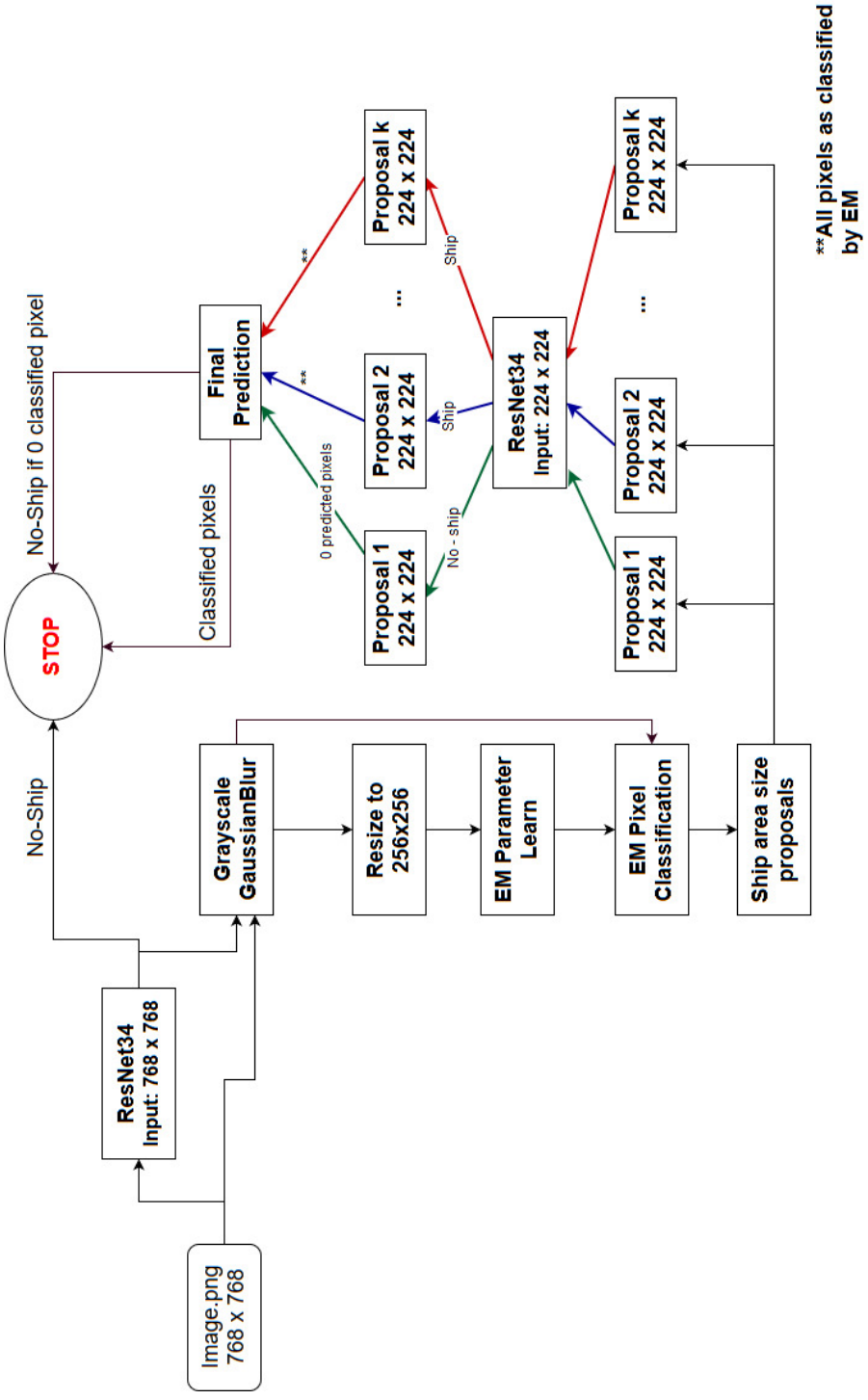

FIGURE 3.4: Region Proposal

- Only the windows that have at least 35 (1.3) pixels predicted as ship will be considered as a possible valid prediction.

- This Residual Network also used the ResNet34 architecture [4] that is capable of classifying the idea of ship/no-ship, and was trained on 224×224 random

crops ($\sim$ **50000**, taken the same way as the selection for prediction).

3.4 Region Proposal Example Run

The following figures shows a picture in different stages of processing:

1. 3.5 - the input image of the algorithm (left), the image after the Gaussian filter was apply and the encoding was moved to grayscale (right)
2. 3.6 - the classification result after the em algorithm where the yellow pixels are background, and the purple one are ships (left), and the em result after the de-noising technique has been applied (right)
3. 3.7 - the 9 proposed regions for second classification
4. 3.8 - the final result compared to the initial picture

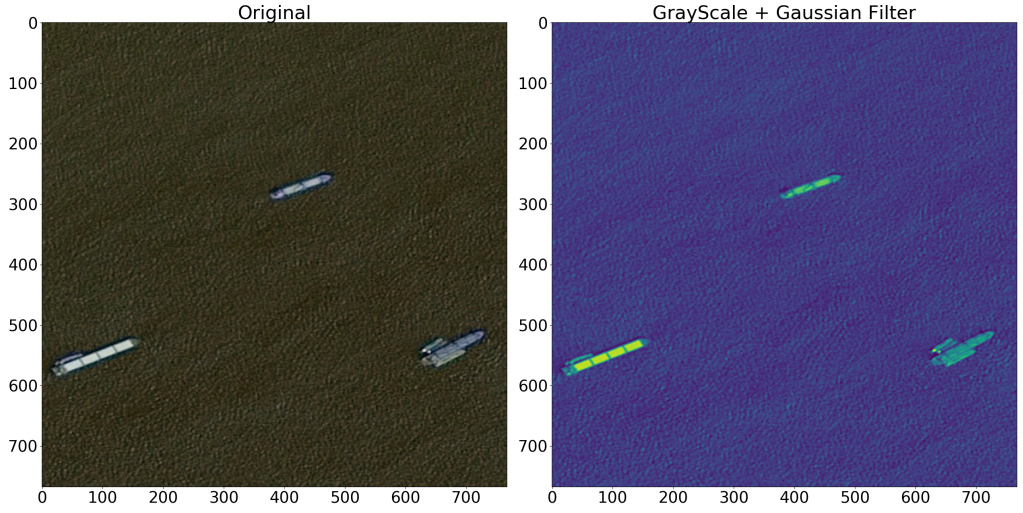

FIGURE 3.5: Input/Original Picture, Grayscale and Gaussian Filtering

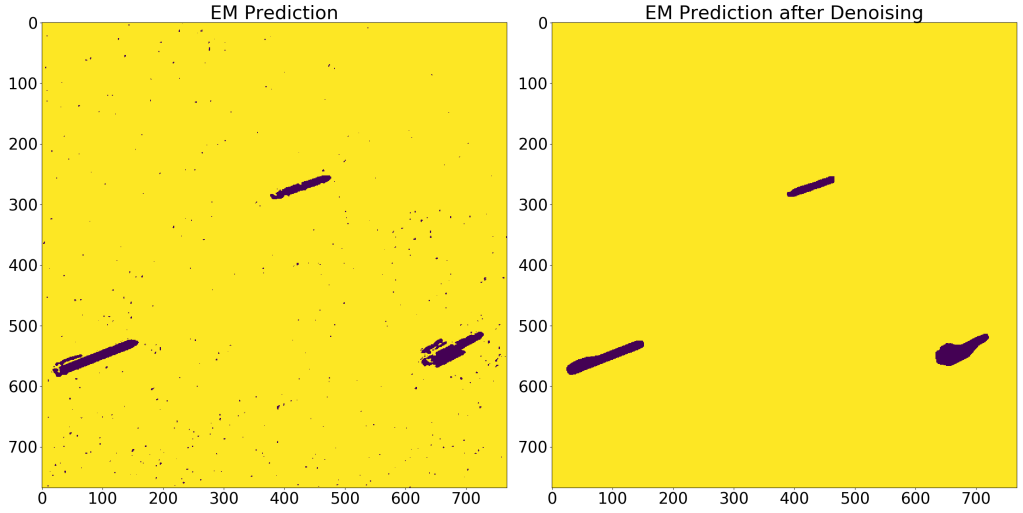

FIGURE 3.6: EM Result before and after denoising

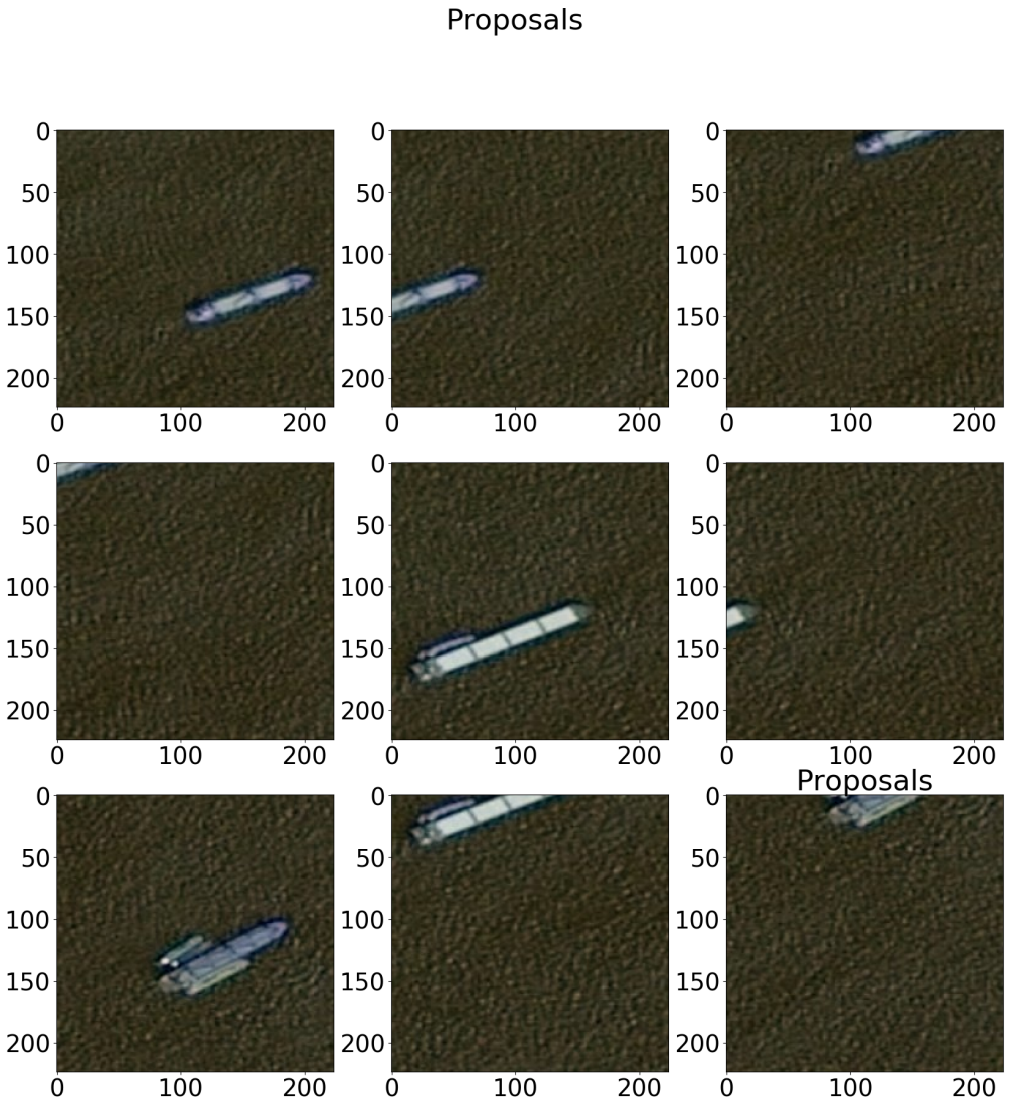

FIGURE 3.7: 9/25 proposed regions

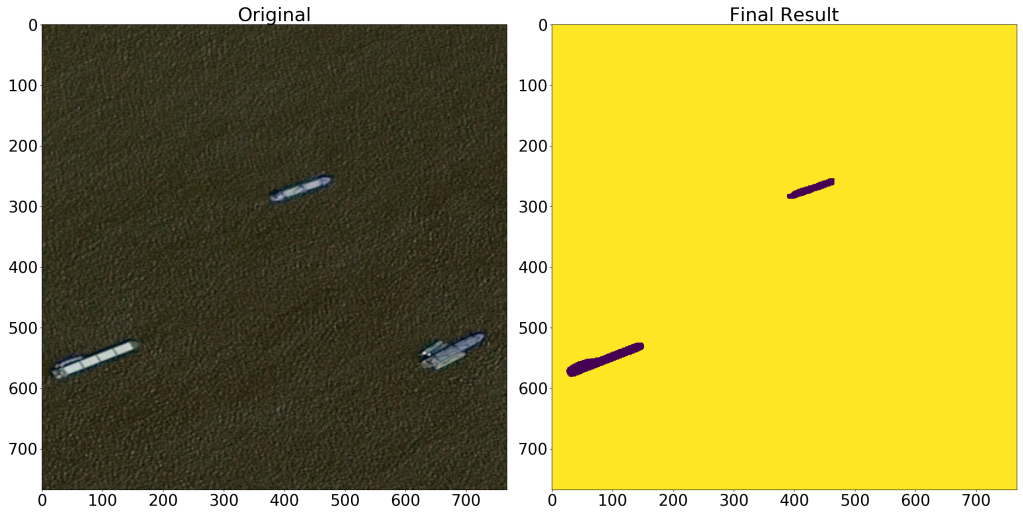

FIGURE 3.8: Final Result

3.5 U-Net Solution

Another solution for solving the **(P3)**, was the usage of U-Nets [5]. As show in 3.9 the initial part of the classification stays the same as in the Region Proposal Solution. This solutions changes after the EM classification is finished. Instead of extracting proposed region for another classification, the pixels that were classified as background are turned black (Pixel Exclusion phase) in the original image, and afterwards the image is sent through a U-Net and a Gaussian smoothing for the final prediction.

For the training of the U-net, a modified version of the F_2 score was used. Since inside a model training, the loss function must be minimized, and in the case of the F_2 score, a higher score means a better prediction, a loss function equal to $-F_2$ was used. The model was trained until the loss function reached the value of -0.2. On every training phase, the model pass thorough 10 epochs, and for each epoch 9 training steps were executed on the model. For every epoch 48 images were randomly selected from the training set.

3.6 U-Net Example Run

The following figures shows a picture in different stages of processing:

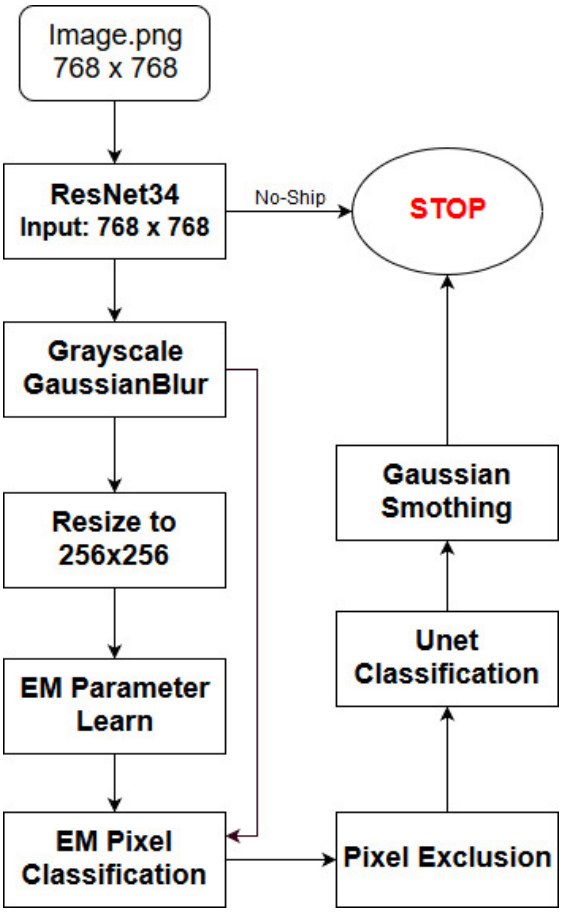

FIGURE 3.9: U-Net

1. 3.10 - as in the Region Proposal Solution the initial steps don't change. The picture is moved to grayscale, and the Gaussian filter is applied.

2. 3.11 - the classification result after the em algorithm where the yellow pixels are background, and the purple one are ships (left), and new image with the pixels classified as background are colored as black in the original image (right).

3. 3.12 - the final result compared to the initial picture. The final result is obtained by inserting the exclusion pixel image in the U-Net network.

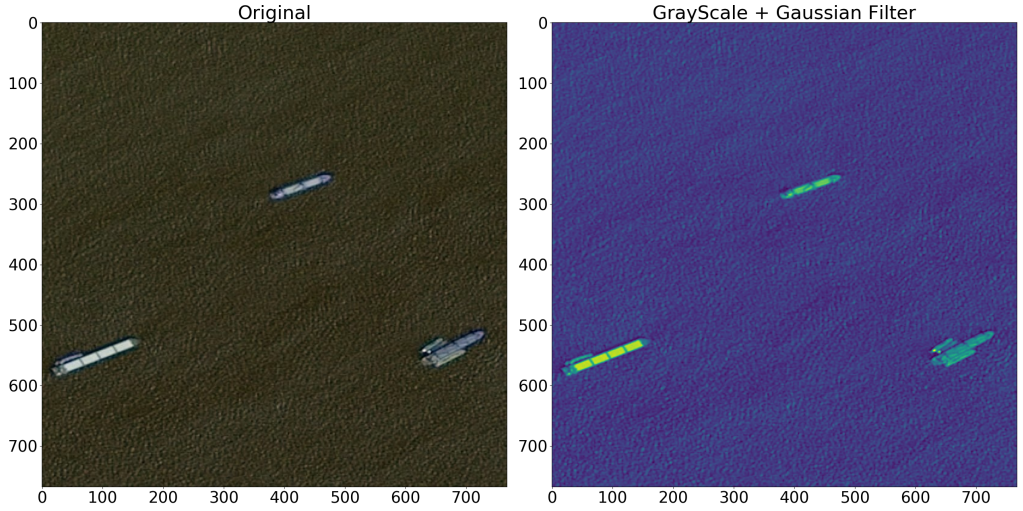

FIGURE 3.10: Input/Original Picture, Grayscale and Gaussian Filtering

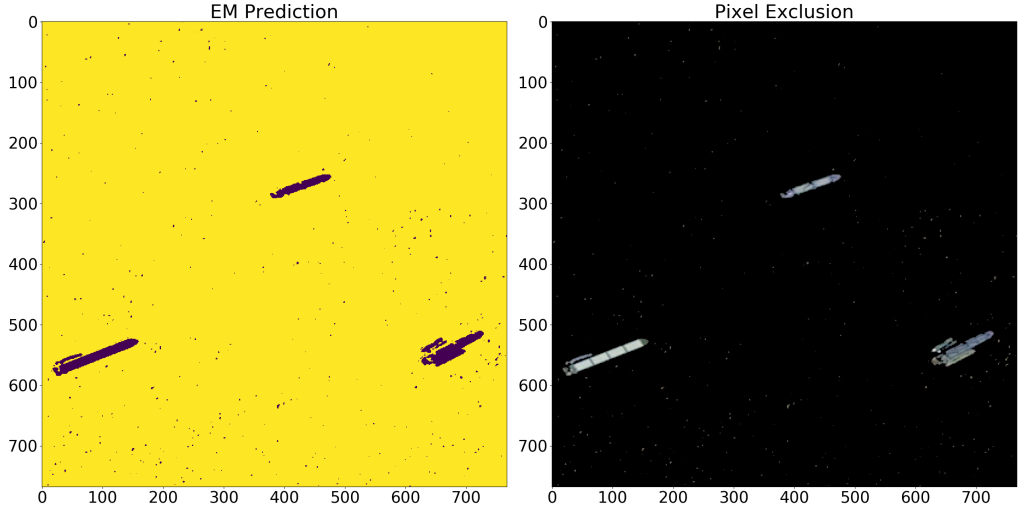

FIGURE 3.11: EM Result and Pixel Exclusion

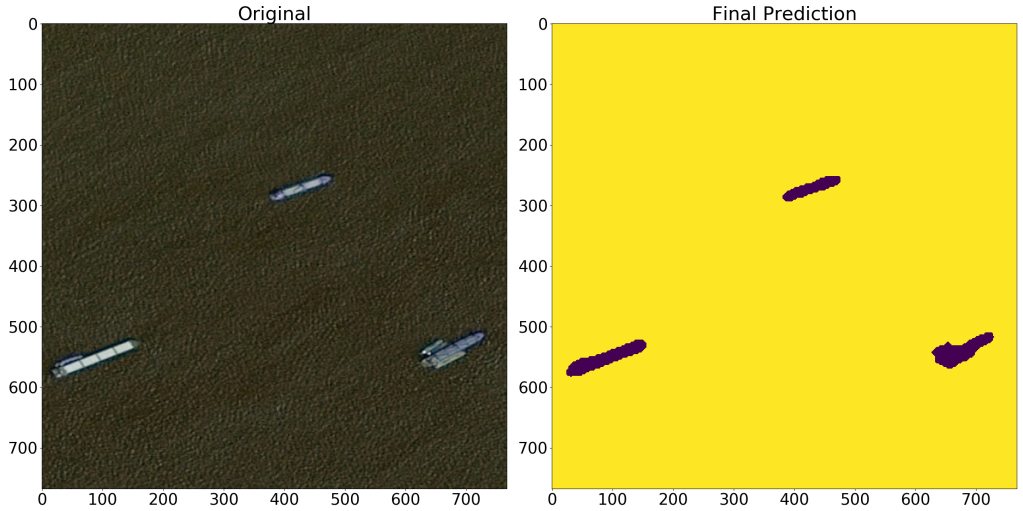

FIGURE 3.12: Final Result

Chapter 4

Results

The Airbus Ship Dataset Challenge [1] offers an automatic method of evaluation that, for every solution, computes the F_2 score. In order to avoid leader-board probing (competitors trying to over-fit the model on the test data), the test data set is split in two datasets. A public test dataset that contains 12% of the original test images, and a private test dataset that contains the other, 88% of the original test images.

To have a base model for comparison against the presented solutions, a No-Machine algorithm has been chosen. This algorithm evaluates every pixel as being a background pixel (no detection involved). The F_2 score for this algorithm is actually a percentage of how many images doesn't have ships in the image.

In the image recognition area is known that neural networks behave well, so, another model was chosen to have a top score value for comparison. This model is a combination between a residual network (ResNet34) which classifies the image in ship/no-ship images, and for images that have ships a U-Net, trained in the same fashion as the U-Net solution presented in this thesis, for the final classification phase. For the rest of the section we will refer to this solution as Hybrid.

Method	Private Test Dataset	Public Test Dataset
No-Machine	0.76566	0.52090
Region Proposal	0.78052	0.58810
U-Net	0.78424	0.61663
Hybrid	0.79396	0.62259

Table 4.1: Results on the competition score

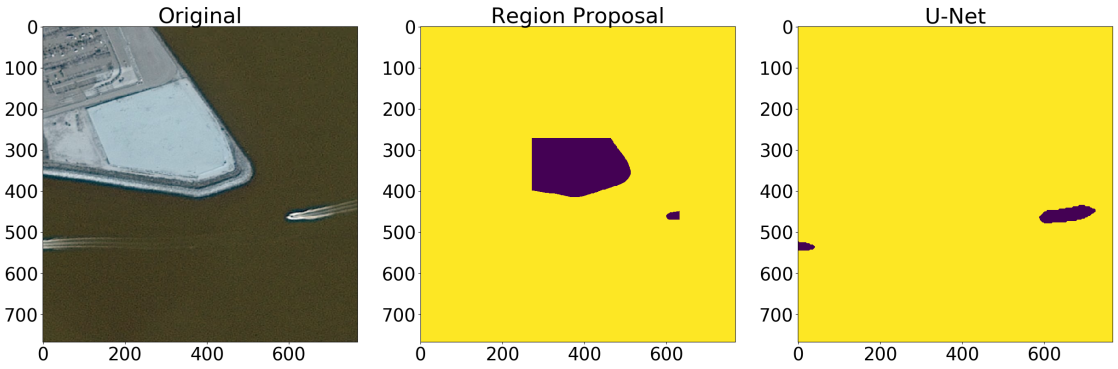

FIGURE 4.1: False Negative

The table 4.1 presents the results as scored by the Kaggle automatic evaluation. Both of the proposed solution have a better score than the No-Machine algorithm, which means that the two proposed solutions correctly identify images that have ships and the ones that does not. The results of the two proposed solutions are similar on both datasets, with small improvements when using the U-Net solution. This improvement is happening because of the granularity that the U-Net is using over the region proposal solution, which uses hard-coded dimensions and positions for the crops that can became region proposal for final prediction. It is worth mentioning that the F_2 score, computed as described in 1, penalizes false negatives over no prediction, which explains the small difference between No-Machine and the proposed solutions. As seen in 4.1, the two solution have the tendency to classify small pockets of pixels as ships, which will drastically decrease the F_2 score for that respective image. Both of the proposed solutions behave worse than the Hybrid solution. The Region Proposal solution has the same weakness of using bulk windows for predictions while the Hybrid solution, as the U-Net proposal, has the advantage of granularity because of the end network used. In the case of U-Net Proposal, due to the pixel exclusion phase, it loses some information from the picture when the pixels are turned black, which leads to a weaker final prediction.

Analyzing 4.3, the two algorithms, behaves in a similar fashion. The better behavior of the U-Net solution can be spotted since the false negatives produced by this, are smaller in area than the ones from the Region Proposal solution 4.1. Again, the rigid character of the Region Proposal solution is seen in the 4.1, where a big part of the port wasn't declassified because it was part of the same region with a ship inside, which the second Residual Network classified it as ship. Although from a visual perspective the U-Net architecture gives a better result, the F_2 score

will be similar (not equal since the two predictions have different area of the ship mask).

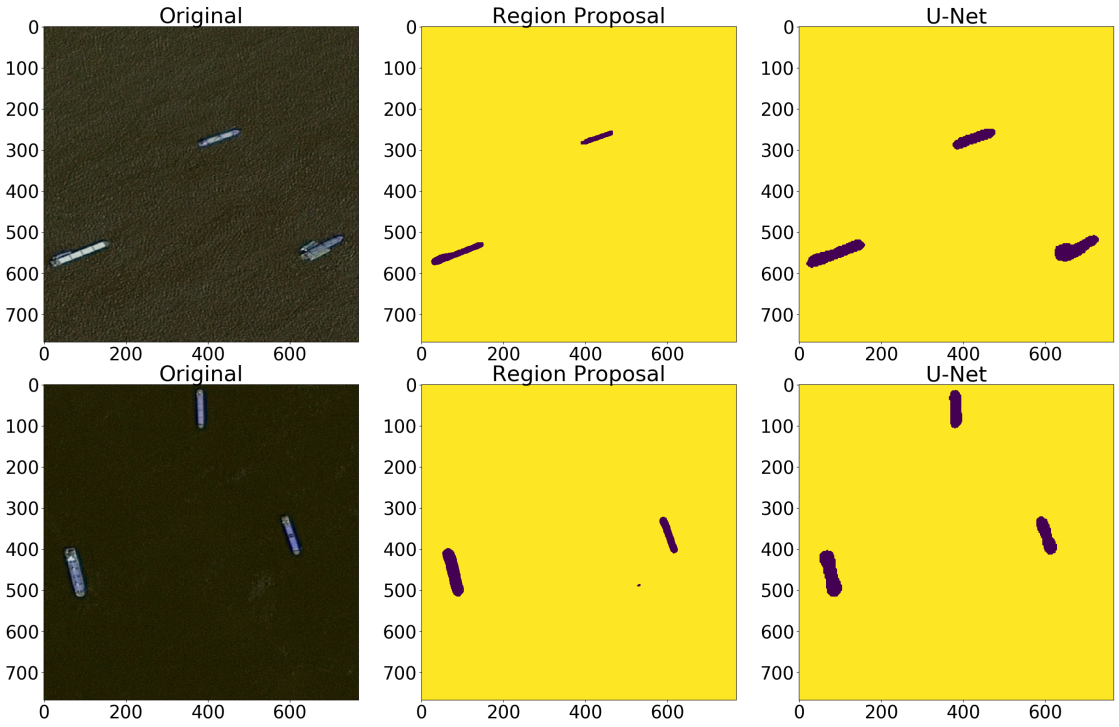

FIGURE 4.2: False Positive

Overall the two solutions are comparable in terms of F_2 score, but on a visual analysis of the results, the U-Net architecture produces less false positives 4.2 and false negatives. The figure 4.3 presents a few more uses cases that confirms the above conclusions.

In the figure 4.4, it is shown that pictures with ports are still a challenge for both solutions. The weak flexibility of the region proposal determined by using bulk windows, can be spotted when a ship is presented in the same window with a piece/or multiple pieces of the port, which in turn also leads to a lot of false negatives. The U-Net solution also struggles to differentiate the port from the ships, the results detection having a lot of pockets that will be evaluate as false negatives and in turn a smaller F_2 score.

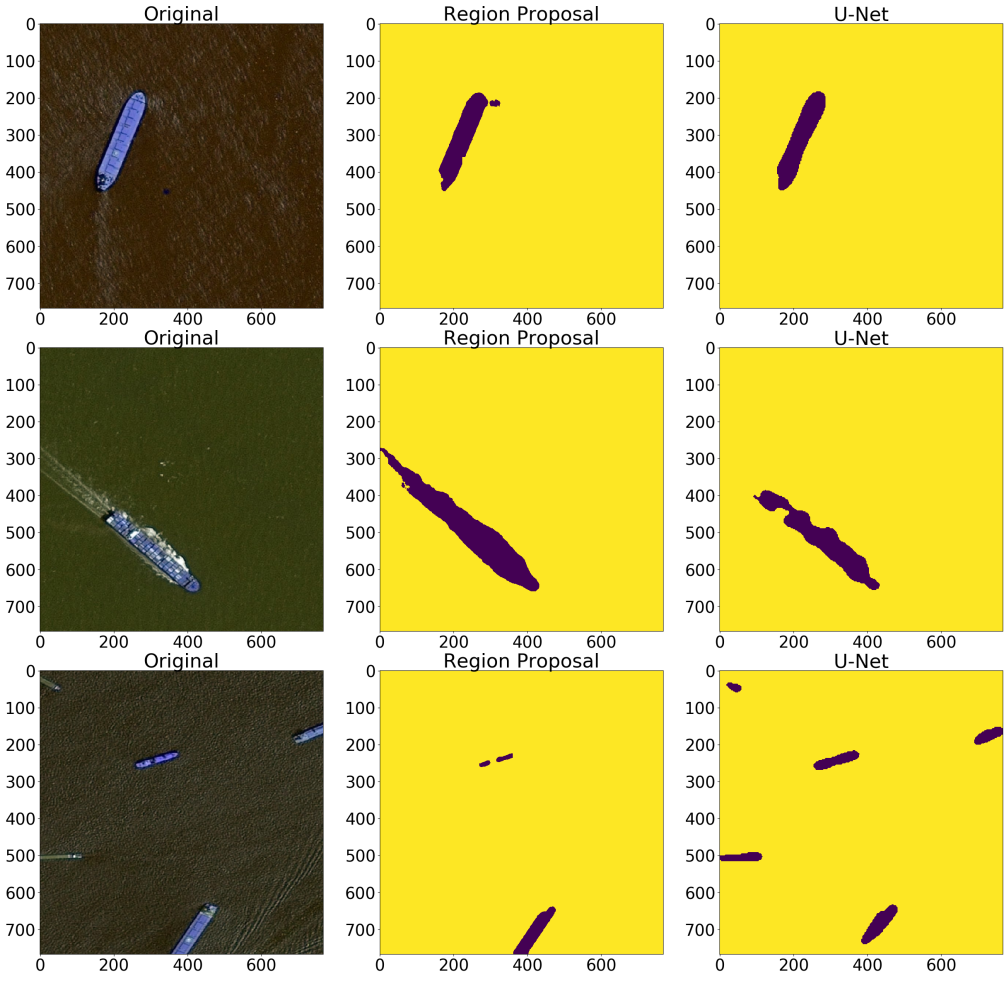

FIGURE 4.3: Final Result comparison

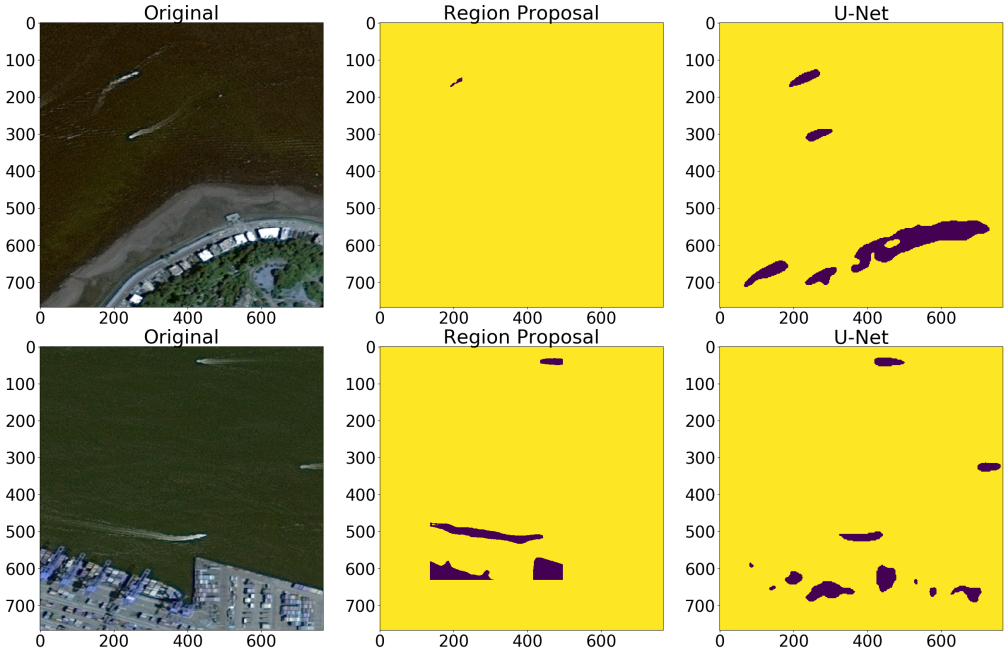

FIGURE 4.4: Final Result Port Comparison

Chapter 5

Conclusions

This thesis presented a good starting point for using an unsupervised learning procedure, with the help of supervised learning, to detect ships from an aerial point of view. The EM algorithm works comparably well with both Residuals Networks and U-Net Network Architecture as shown in the Results chapter. Although, from the F_2 metric perspective the two solutions proposed didn't produce a large improvement over the base model, the expectation-maximization algorithm by itself, behaved well from a visual perspective, and can be used, in a practical environment, in conjunction with external help. The algorithm can be improved or even parallelized to reduce the execution time and to not give up accuracy. Since all the operation involved in the EM parameter learn and classification phase are associative, a multi-threading run can be implemented for usage of the algorithm in live instances. In this case the EM has the advantage of not needing a prior training before being put to use and is more suited for live predictions (e.g. in videos).

The EM underlying model used is relatively simple, hence it can be improved with additional work. A first idea could be the usage of multi-level EM as shown in [7], or by using a different probability function (e.g. more than two Gaussians that generated the data). The biggest challenge that these solutions faced was the color similarity between ships and ports which gives us false negatives and a lower F_2 score. On the other hand, when the picture were taken "at sea", the algorithm produces good results. With this in mind, the algorithm in the actual form (or eventually improved forms) can be used on datasets that match the property of having two main colors scheme inside them.

Bibliography

[1] https://www.kaggle.com/c/airbus-ship-detection.

[2] Liviu Ciortuz. Bayesian learning, 2019. https://profs.info.uaic.ro/~ciortuz/SLIDES/ml6.pdf.

[3] Yann Lecun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, 2015. URL http://arxiv.org/abs/1512.03385.

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4.

[6] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv e-prints*, art. arXiv:1603.07285, Mar 2016.

[7] Dong-Su Lee, Seokwon Yeom, Jung-Young Son, and Shin-Hwan Kim. Automatic image segmentation for concealed object detection using the expectation-maximization algorithm. *Optics express*, 18:10659–67, 05 2010. doi: 10.1364/OE.18.010659.