An Automated Method for the Creation of Oriented Bounding Boxes in Remote Sensing Object Detection Datasets

Georgios Savathrakis

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Computer Science and Engineering

University of Crete
School of Sciences and Engineering
Computer Science Department
Voutes University Campus, 700 13 Heraklion, Crete, Greece

Thesis Advisor: Prof. *Antonios Argyros*

This work has been performed at the University of Crete, School of Sciences and Engineering, Computer Science Department.

The work has been supported by the Foundation for Research and Technology - Hellas (FORTH), Institute of Computer Science (ICS).

University of Crete
Computer Science Department

An Automated Method for the Creation of Oriented Bounding Boxes in Remote Sensing Object Detection Datasets

Thesis submitted by
Georgios Savathrakis
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: _____
Georgios Savathrakis

Committee approvals: _____
Antonios Argyros
Professor, Thesis Supervisor

_____
Panos Trahanias
Professor, Committee Member

_____
Xenophon Zabulis
Principal Researcher, Committee Member

Departmental approval: _____
Polyvios Pratikakis
Associate Professor, Director of Graduate Studies

Heraklion, March 2024

An Automated Method for the Creation of Oriented Bounding Boxes in Remote Sensing Object Detection Datasets

Abstract

The detection of objects in remote sensing images is an important application of computer vision. Given the increasing demands regarding the surveillance and monitoring of areas that may include objects of interest, the task of accurately detecting objects from remote sensing aerial images is of significant importance. Various object detection algorithms localize objects by identifying either their Horizontal Bounding Boxes (HBBs) or their Oriented Bounding Boxes (OBBs). OBBs provide a far more accurate/tighter localization of object regions as well as their orientation. Several object detection datasets provide annotations that include both HBBs and OBBs. However, many of them do not include OBB annotations. In this work, we propose a method which takes the objects' HBB annotations as input, and automatically calculates the corresponding OBBs. The proposed method consists of three main parts, (a) object segmentation that is built upon the segment-anything model (SAM) to calculate object masks based on the information provided by the HBBs, (b) morphological filtering which eliminates possible artifacts stemming from the segmentation process, and (c) contour detection applied to the post-processed masks that are used to compute the optimal OBBs of the target objects. By automating the process of OBB annotation, the proposed method permits the exploitation of existing HBB-annotated datasets to train object detectors of improved performance. Furthermore, we propose the development of two data augmentation methods that resolve the problem of the objects' orientation imbalance. We do this by either maintaining or increasing the number of objects in the dataset. Creating augmented datasets can lead to less biased datasets which when used for training can lead to more precise detections. We support this finding by reporting the results of several experiments that involve three standard remote sensing object detection datasets, as well as state of the art oriented object detectors.

Μία μέθοδος για την αυτοματοποιημένη δημιουργία περιστραμμένων πλαισίων αντικειμένων σε εικόνες δορυφορικής τηλεπισκόπησης

Περίληψη

Ο εντοπισμός αντικειμένων σε εικόνες δορυφορικής τηλεπισκόπησης αποτελεί μια σημαντική εφαρμογή της υπολογιστικής όρασης. Δεδομένης της ολοένα και αυξανόμενης ζήτησης σχετικά με την παρακολούθηση και εποπτεία περιοχών που ενδεχομένως περιέχουν αντικείμενα ενδιαφέροντος, ο ακριβής εντοπισμός αντικειμένων από εναέριες εικόνες δορυφορικής τηλεπισκόπησης είναι υψίστης σημασίας. Διάφοροι αλγόριθμοι που αποσκοπούν στον εντοπισμό αντικειμένων, καθορίζουν την περιοχή όπου βρίσκονται τα αντικείμενα, χρησιμοποιώντας είτε οριζόντια είτε περιστραμμένα ορθογώνια παραλληλόγραμμα/πλαίσια που περικλείουν τα αντικείμενα. Τα περιστραμμένα πλαίσια παρέχουν μία πολύ πιο ακριβή οριοθέτηση της θέσης των αντικειμένων, καθώς επίσης και την κατεύθυνσή τους. Τα περισσότερα σύνολα δεδομένων για εντοπισμό αντικειμένων, παρέχουν επισημάνσεις που συμπεριλαμβάνουν και οριζόντια και περιστραμμένα πλαίσια. Παρ΄ όλα αυτά υπάρχουν και ορισμένα τα οποία δεν παρέχουν επισημάνσεις με περιστραμμένα πλαίσια. Σε αυτήν την εργασία, προτείνουμε μία μέθοδο η οποία παίρνει σαν είσοδο τις επισημάνσεις με τα οριζόντια πλαίσια των αντικειμένων, και αυτόματα υπολογίζει τα αντίστοιχα περιστραμμένα πλαίσια. Η προτεινόμενη μέθοδος απαρτίζεται από τρία κύρια μέρη, (α) κατάτμηση αντικειμένων χρησιμοποιώντας το μοντέλο SAM, για να υπολογιστούν οι μάσκες των αντικειμένων, βάσει της πληροφορίας που παρέχεται από τα οριζόντια πλαίσια, (β) μορφολογικό φιλτράρισμα για εξάλειψη πιθανών τεχνητών σφαλμάτων που προκύπτουν από την διαδικασία κατάτμησης, και (γ) εντοπισμό περιγραμμάτων επί των φιλτραρισμένων μασκών, οι οποίες χρησιμοποιούνται για τον υπολογισμό των βέλτιστων περιστραμμένων πλαισίων των αντικειμένων. Αυτοματοποιώντας τη διαδικασία της επισήμανσης με περιστραμμένα πλαίσια, μας παρέχεται η δυνατότητα εκμετάλλευσης υφιστάμενων συνόλων δεδομένων που περιέχουν οριζόντια πλαίσια, ώστε να εκπαιδευτούν μοντέλα εντοπισμού αντικειμένων τα οποία θα οδηγήσουν σε βελτιωμένες επιδόσεις. Επιπροσθέτως, προτείνουμε την υλοποίηση δύο μεθόδων επαύξησης δεδομένων οι οποίες επιλύουν το πρόβλημα της ανομοιόμορφης κατανομής των κατευθύνσεων των αντικειμένων. Αυτό μπορεί να επιτευχθεί είτε διατηρώντας είτε αυξάνοντας το πλήθος των αντικειμένων. Η δημιουργία επαυξημένων συνόλων δεδομένων μπορεί να τα καταστήσει λιγότερο μεροληπτικά και να αυξήσει την ακρίβεια

εντοπισμού αντικειμένων. Στηρίζουμε αυτήν την υπόθεση, παραθέτοντας αποτελέσματα από έναν μεγάλο αριθμό πειραμάτων που περιλαμβάνουν τρία δημοφιλή σύνολα δεδομένων, καθώς επίσης και σύγχρονους αλγορίθμους εντοπισμού περιστραμμένων αντικειμένων.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω πρωτίστως τον επιβλέποντα καθηγητή μου κ. Αντώνη Αργυρό, για την ανεκτίμητη υποστήριξη και καθοδήγησή του, χωρίς την οποία η υλοποίηση αυτής της εργασίας θα ήταν αδύνατη.

Επίσης, θα ήθελα να ευχαριστήσω τα μέλη του εργαστηρίου CVRL, για τις πολύ χρήσιμες και εποικοδομητικές συζητήσεις που είχαμε οι οποίες οδήγησαν στην βελτίωση της ποιότητας της παρούσας εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω τα μέλη της τριμελούς επιτροπής για τον χρόνο που αφιέρωσαν για την ανάγνωση και αξιολόγηση της εργασίας.

Στους γονείς μου

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The detection of objects in remote sensing satellite images is a task that is both challenging and important for different types of applications. Tasks like environmental monitoring, marine tracking or land mapping among others, are increasingly reliant on methods that can automatically detect objects of interest in satellite images. During the last decade, most models that tackle the task of object detection, make use of deep learning architectures, for the extraction of important features from an image, as well as for the classification of an object and the definition of its bounding region. A bounding region can be determined in a variety of ways. Usually, in the task of object detection, the region is defined using a rectangular box which is either axis aligned, meaning that its sides are always parallel to the image axes, or object aligned, meaning that one axis is parallel to the orientation axis of the object and the other perpendicular to it. These bounding boxes are referred to as Horizontal and Oriented Bounding Boxes (HBBs, OBBs) respectively.

1.1 Object Detection

Object detection is the problem where given an image or a video sequence, as well as ground truth annotations, it is required to locate the regions in an image that contain an object, as well as classifying that object into one or more categories. The determination of the region where an object lies, is implemented by using bounding boxes that are rectangles in an image surrounding the existing objects. An example is shown in Figure 1.1.

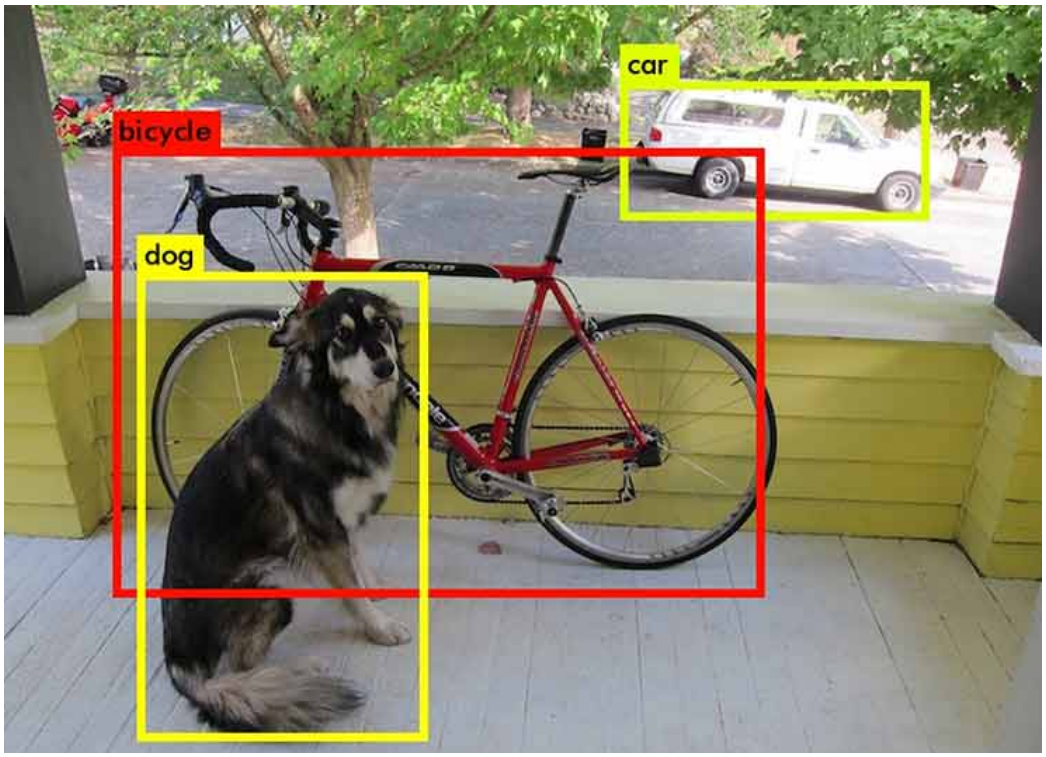


Figure 1.1: Example of object detection in an input image. Objects in the training domain are identified by a box surrounding them, as well as a label which determines the class of the object.https://miro.medium.com/v2/resize:fit:739/1*IrptRDRG8IL9o-55BKjbLA.png

1.1.1 Pre Deep Learning Era

Works that resolved the problem of object detection in the pre deep learning era (e.g., [34, 4]), mainly focused on the successful extraction of features from input images, such as edges, pixel value statistics within rectangular regions etc. With ground truth which includes information about the objects' location, these features can be used for training machine learning algorithms that can detect candidate object regions (e.g. [1, 10]). However, these methods generally rely on several hand-crafted features that are themselves dependent on the domain at which they are applied. This means that using the same models to tackle detection problems in different applications will not guarantee equivalent detection performances in general.

1.1.2 Deep Learning Era

Since the introduction of Convolutional Neural Networks (CNNs) in the field of image processing however, most of the modern object detectors use them

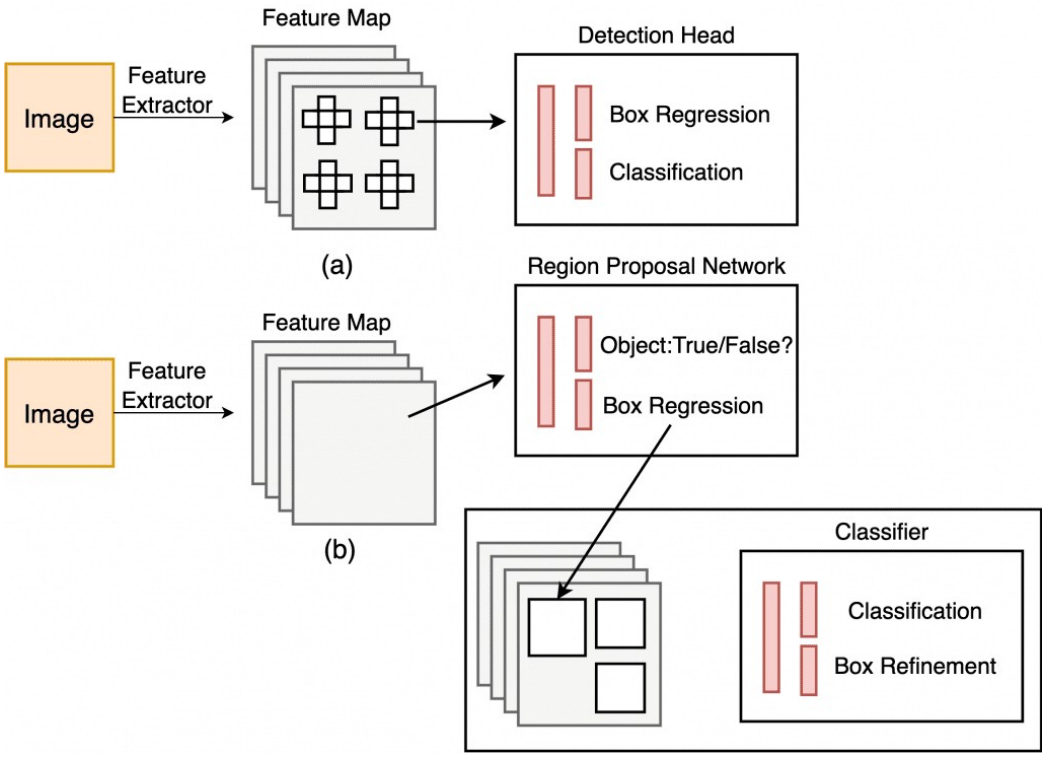


Figure 1.2: A visualization of the differences between one-stage and two-stage object detectors https://pyimagesearch.com/2022/04/11/understanding-a-real-time-object-detection-network-you-only-look-once-yolov1/

for the task of general feature extraction form input images. The advantage of using CNNs for this role, is that the feature extraction is considerably more adaptable to different scenarios. It can overcome issues like object occlusion or blurring, as well as variations in scale and illumination. That is because, CNNs use filters that are convolved with the input image using its original shape, thus maintaining the spatial information included in the image. Also, due to the depth of the CNN architectures, the model can learn more abstract features in the first layers and more detailed ones in the final layers. Added to the fact that feature extraction is done directly on the input data whenever the CNNs are getting trained, this makes CNNs more adaptable to different types of applications, thus reducing the need for many hand-crafted components.

CNNs however, only account for the feature extraction part. The next step is the processing of these features in order to detect and classify the objects in an image. This is where modern object detectors are split into two main categories. These are one-stage and two-stage detectors. One-stage object detectors use the generated feature maps, and then an objectness score is

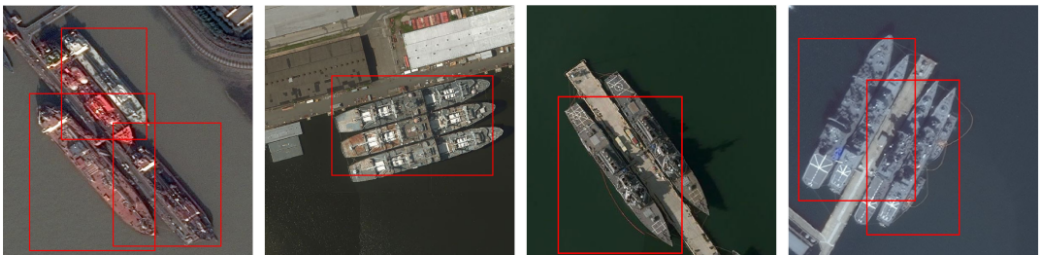


Figure 1.3: Examples of failed detections in remote sensing ship datasets generated from Faster-RCNN with HBBs.
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8438330

calculated within pre-determined rectangles, also called anchors, at several locations and with varying scales and sizes. This objectness is calculated using a Feed-Forward Neural Network added with a sigmoid activation which computes the probability that a region contains an object. Finally, a concurrent calculation determines the class of the detected object as well as its precise bounding box coordinates. Since all these steps happen in one pass, one-stage object detectors are considerably fast and have many uses in real life applications. Notable examples of one-stage object detectors are the, You Only Look Once (YOLO) algorithm [29], Single-Shot Detector (SSD) [24], and RetinaNet [22]. Two-stage object detectors on the other hand, use the generated feature maps from the CNN and generate possible object regions which are subsequently classified into object types, and refined in terms of the bounding box coordinates. The region proposals can be generated with several methods that can either depend on color and texture features (e.g. [11]), or by implementing fully convolutional neural networks that take the feature map as input, and generate rectangular proposals with objectness scores for each of the proposed regions (e.g. [30]). The benefit of two-stage object detectors is that they are in general more accurate compared to their one-stage counterparts. However, this comes at the cost of higher computational complexity as well as larger training and inference times. A visual example which depicts the discrepancies between one and two-stage detectors is provided in Figure 1.2.

1.1.3 Horizontal vs Oriented Object Detectors

Another differentiation between object detectors, is whether they detect objects using HBBs or OBBs, as mentioned previously. HBB object detectors are most widely used due to their regression efficiency since only 4 values have to be predicted, and they are aligned with the axes of the image. Also,

HBB detection is computationally cheaper because the predetermined anchor boxes only have to account for variations in aspect ratios and sizes. Orientation is another degree of freedom which will inevitably require more anchor boxes for effective detection. Finally, most tasks don't require the orientation of the objects since in the majority of scenarios, the objects occupy most of the HBB area (e.g. cars, household objects etc.) or their shapes render the use of OBBs irrational (e.g. circular objects). However, in the case of aerial images where the objects are mostly elongated and at arbitrary orientations, the use of OBBs can yield considerable improvements in the detection performances of object detectors. Works that have been done to detect ships in remote sensing satellite images, showed that in the cases where ships are moored close to each other, HBB object detectors fail to detect all of them as separate objects and instead there are cases where many objects are detected with only one bounding box [25, 39] as can also be seen from Figure 1.3. Apart from the reduction of the number of false negatives, OBBs are also useful in the cases of object tracking since the object orientation can serve as a prior for the expected location of the same object in subsequent frames.

1.2 Need for more training data

OBB detectors need to be trained on datasets where oriented bounding boxes ground truth is available. However, not all datasets provide OBB annotations (e.g., [20], [3]). To enrich the training potential of existing datasets and to come up with OBB ship detectors of improved performance, in this work, we propose a method which can automatically create OBBs for ship detection datasets, using only the provided HBB annotations as input. The proposed method is based on the Segment-Anything Model (SAM) [18], which is used to segment objects, given input prompts that can be obtained from the HBB ground truth. These segmented objects are in the form of masks that determine the image pixels where the objects are located. However, a large number of the generated masks exhibit artifacts such as object fragmentation in several connected components. This is resolved with morphological filtering, that connects possibly disconnected object regions, leading to more compact and accurate object masks [2]. In order to obtain OBBs that correspond to the segmented objects, we capitalize on the fact that the target OBB is the bounding box of minimum area. Therefore, we compute the boundaries of the post-processed object masks which are fed into a Minimum Area Rectangle calculation method which returns the target OBB.

The aforementioned process creates reliable OBBs for remote sensing

object detection datasets, making exclusive use of the objects' HBB annotations. This is validated through extensive experiments on three benchmark datasets for aerial object detection, DOTA [35], HRSC2016 [26] and ShipRSImageNet [40], where we make use of the HBBs of their training sets to calculate OBBs with the described method. Then, using these OBBs we train a wide range of OBB object detectors (i.e., [37, 14, 36, 6, 22, 13]), and compare their performance on the test set to that which would have been obtained if the ground truth OBBs had been used.

1.3 Need for removing dataset biases

In addition to creating more training data automatically, the proposed automated OBB annotation can lead to the reduction of spatial, scale or orientation imbalance in an existing dataset. Specifically, orientation or aspect ratio bias among the objects in the dataset, may lead to less accurate detections if they are used for training. Data augmentation methods have shown that they can lead to increased performance both in image classification [19, 31, 28] and object detection [43, 41] tasks. When it comes to object detection, geometric transformations (i.e. rotation, scaling) are the most effective ways to resolve these imbalances, since if they are applied correctly to both the images and the annotations, more orientations and sizes will be visible to an object detector during training. This can be done by rotating existing samples in order to cover the orientation space, and thus balance the number of samples per orientation. In our work, we demonstrate that data augmentation performed based on the proposed method leads to improved performance of existing OBB object detectors.

1.4 Contribution overview

In summary, the main contributions of this paper are the following:

- We provide an automated method which creates OBBs in remote sensing object detection datasets, built upon the SAM model and morphological filtering operations, using only the HBB ground truth as prior information.
- We develop a data augmentation technique, using the generated OBB annotations, to reduce orientation imbalance. We prove that by using augmentation we can increase the baseline performance of existing object detectors in remote sensing object detection datasets.

Chapter 2

Related Work

2.1 Pre-Deep Learning Detectors

One of the most predominant detectors used during the era prior to the establishment of deep learning approaches, was the one proposed by Viola and Jones [34] named after the authors, and was used to solve the problem of face detection. It consisted of three main parts. The first implemented feature extraction using the pixel difference of two-rectangle regions, computed using the integral image method. Despite the coarsity of these features, they are quite capable of detecting edges or other plain structures. The second part was responsible for the correct classification of image sub-regions into foreground or background areas. This was implemented using the AdaBoost algorithm [10], which given a set of classifiers that make use of different features during the training, it can separate them into strong and weak classifiers, by calculating the difference of the classifier output of a region and the ground truth which states whether it is foreground (1), or background (0). The error is calculated by multiplying that difference with the respective region's normalized weight, and the selected classifier is the one with the smallest error. Subsequently, the region weight is updated for the next iteration in a way that it gets penalized if it belongs to the background and rewarded otherwise. The third and final part is essentially a method to create stronger classifiers and reduce training time. It implements a cascade of classifiers, that hierarchically eliminate negative sub-windows with little computations, and as the stages progress, the classifiers become boosted and thus more complex, and the rejected regions get reduced. That way the model converges to the most important regions faster, without the need to spend lots of computational time on unimportant regions. This model proved very effective for the task of face detection, since the features used were highly

sensitive to the shape of upright faces. This led to very accurate detections at faster times than any of the previous methods.
Another such very efficient detector was the one proposed by Dalal and Triggs [4], called Histogram of Oriented Gradients Detector (HOG). In their work, they made the assumption that the local gradient information about its magnitude and orientation in different regions, can constitute very informative features that can subsequently be used to detect object regions. The methodology starts with a normalization of the color and contrast within different image blocks where the method is to be applied. Each of these blocks is attributed an orientation histogram, whose bins are determined by the selected angle granularity. Then in each block, the region is separated into cells upon which the gradient magnitudes and orientations are computed. This is usually done by convolving the regions with the X and Y Sobel filters. The gradient magnitudes and orientations are calculated as:

$$G_{mag}[i,j] = \sqrt{G_x^2[i,j] + G_y^2[i,j]} \tag{2.1}$$

$$G_{or}[i,j] = arctan(\frac{G_y[i,j]}{G_x[i,j]}) \tag{2.2}$$

At each cell, the resulting gradient magnitude and orientation is the average of the pixels within it. The gradient histogram of each block is calculated in the following way. Each bin has an angle width $\mathbf{w}$. The value each orientation bin has, is calculated in the following way. Assuming that a pixel has gradient orientation θ, gradient magnitude $\mathbf{m}$, and lies between bin values θ_1 and θ_2. These two bins are the ones whose value will increase according to a weighted vote originating from the cell in question. Bin θ_1 will be increased by $\theta_1 = \theta_1 + (\theta_2 - \theta)\mathbf{m}$, and θ_2 by $\theta_2 = \theta_2 + (\theta - \theta_1)\mathbf{m}$. By creating a concatenated vector consisting of the gradient histograms in different blocks, and using ground truth vectors that specify if a region contains an object or not, an SVM module is trained which classifies regions to class object or no object. This method was validated on pedestrian datasets and showed remarkable detection performances.

2.2 One-stage Object Detectors

Since the introduction of deep learning in the field of object detection several algorithms have been proposed that tackle this problem with significantly improved efficiency. Redmon et al. 2016 [29] proposed the YOLO algorithm which is a one-stage object detector that simultaneouly predicts all bounding

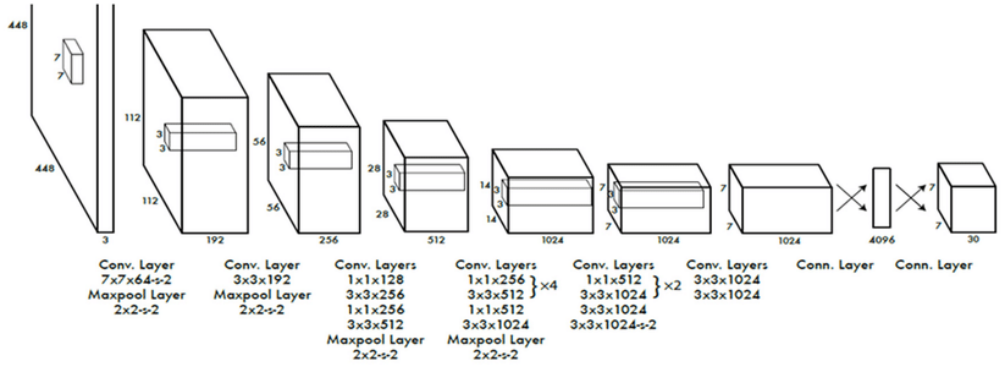


Figure 2.1: Visualization of the YOLO algorithm architecture [29]

boxes for all classes in one image. Its architecture consists of 24 convolutional layers followed by 2 fully connected layers as can be seen in Figure 2.1. In the CNN part, several max pooling layers are used, which reduce the spatial dimensions of the features while extending their channel dimensions. The final result of the CNN part is a 7×7×30 feature map, which passes through the 2 FFNs which respectively regress the bounding box coordinates and the probability scores. The input image is separated into multiple cells, each of whom have several bounding box predictions. However, the bounding box with the highest IoU is the one selected as the prediction for a specific object, which eliminates several other predictions from different cells. The implemented loss function penalizes classification and bounding box coordinates errors only if the specific predictor belongs to the cell which contains an object. For more efficient training, a learning rate schedule is used that increases the initial learning rate gradually during the first few epochs, and then decreases it in a step-wise manner twice during subsequent epochs of training. Additionally, dropout layers and augmentation are implemented for overfitting avoidance. Another issue that often appears, is multiple detections due to an object's location near the borders of multiple cells. This is resolved using non-max suppression which leads to a slight increase in detection performance. This model was trained on the Pascal VOC 2007 dataset [7] and it turned out that it achieved significant improvements in detection performance compared to other contemporary state-of-the-art detectors like R-CNN [12]. Apart from that, YOLO showed far more significant improvements with regard to the training speed compared to other detectors, which made it a very useful tool for real-time applications.

Liu et al. 2016 [24], proposed another one-stage detector called SSD, which provided an improved detection method compared to its predecessor YOLO,

both in terms of detection precision and training speed. This model consists of a base CNN which outputs a feature map with dimensions 38×38×512. The remaining of the network is based on 3 main additions. The first is the introduction of more feature layers that reduce the spatial dimensions of the main network's output at an ever increasing extent. That way the model is able to learn the location of objects that are of varying scales. For instance, smaller objects will be able to get detected from feature maps of higher resolution, which is not necessary for larger objects. After the creation of these feature maps, each of them is convolved using a 3×3×p kernel which produces a confidence score and bounding box offsets at each grid cell for each of the existing object categories. The third and final part is the use of fixed size boxes with different aspect ratios at each cell of each of the feature maps. The output from the kernel convolution with each of the feature maps consists of bounding box offsets, as mentioned previously. These offsets are calculated with respect to each of the fixed bounding boxes at the corresponding feature map and at the corresponding point. Therefore, if there are x×y points, m fixed boxes and c classes, the output of the convolution has dimensions (c+4)xym. However, the most important part is to match the detections with the ground truth, which is done by examining all the fixed boxes at every feature map and calculating the overlap with the ground truth object's bounding box at hand. The selected fixed box is the one with the highest overlap with the ground truth. Then, for easing the training, each fixed box is matched to all ground truth boxes with overlap exceeding 50%. Regarding the remaining of the implementation details, a weighted loss function is used, which combines localization and classification loss, SGD optimizer with learning rate decay policy and in addition data augmentation is used to improve the training performance. This augmentation is achieved by either sampling an image, a random patch of an image, or a patch of an image conditioned over a required overlap threshold. SSD was trained on the Pascal VOC 2007 and 2012 datasets [7, 8] and the result was that compared to some of the most contemporary two-stage detectors like Fast and Faster R-CNN [11, 30], SSD achieved improved detection performances. In addition SSD managed to surpass plain YOLO [29] in terms of both accuracy and inference speed.

Lin et al. 2017 [22], in an attempt to solve the problem of the general lag-behind in terms of accuracy of one-stage compared to two-stage object detectors, focused on resolving the issue of class imbalance between foreground and background which leads to less predictive capabilities of dense detectors. To that end, they proposed an improvement of the cross-entropy loss function, which decreases the loss attributed to correctly detected objects.

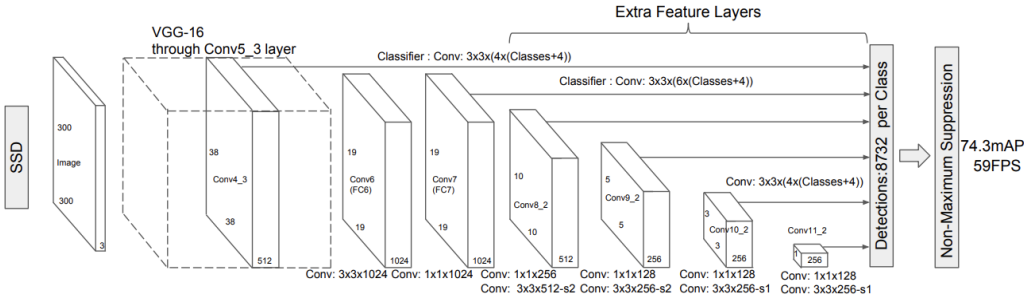


Figure 2.2: Visualization of the SSD algorithm architecture [24]

The Focal Loss function which aims to achieve this goal, multiplies the cross-entropy function by a factor of $(1\text{-}p_t)^\gamma$. That way, correctly detected objects (i.e. with $p_t > 0.5$), will have smaller losses for a value of $\gamma > 0$. In addition to that loss variant, a Feature Pyramid Network (FPN) [21] is used as part of the proposed network, which creates multi-scale feature maps with lateral connections to the convolutional backbone, which assist the network to detect objects at different scales. Finally, a classification and a box regression subnet are used, in order to calculate object class probabilities and bounding box offsets, with respect to each of the spatial positions at each feature map respectively. This is implemented using small Fully Connected Convolutional Networks (FCNs) that consist of 4 layers with $3\times3\times$C dimensionality, apart from the last one whose channels are the number of anchors times the length of the output. This length is the number of classes for the classification subnet, and 4 for the box regression subnet. The experimental validation of this method was done using the benchmark COCO object detection dataset [23]. Compared with the most contemporary object detectors, either one or two-stage, the result was that RetinaNet achieved better detection accuracies even with the addition of FPNs on two-stage detectors like Faster R-CNN. This indicated that the implemented loss function did indeed contribute a significant improvement in terms of predictive performance. With regards to speed performances, RetinaNet has smaller inference speeds compared to other one-stage detectors like YOLO or SSD, but higher ones compared to detectors like Faster R-CNN.

2.3 Two-stage Object Detectors

Girshick et al. 2014 [12] introduced the concept of using both region proposals and images as input for a CNN backbone. Initially, a set of object proposals are extracted using selective search [33]. These proposals are in

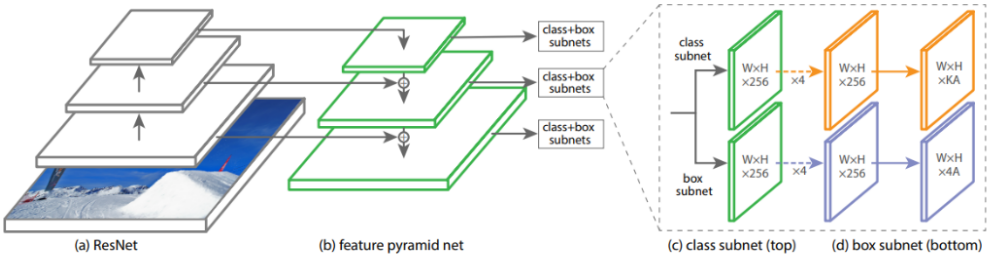


Figure 2.3: Visualization of the RetinaNet architecture [22]

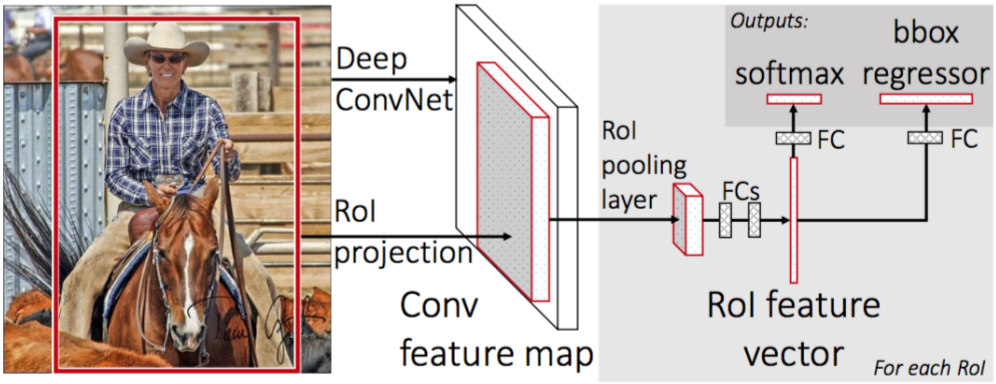


Figure 2.4: Visualization of the Fast R-CNN architecture [11]

the form of an image patch that is considered to contain objects. After their generation, they pass through a CNN to extract a feature map for each of the generated proposals. The model is called R-CNN from Region-CNN. The final component of the model is a linear SVM which takes the output of the CNNs, specifically the output from the final layers which are fully connected layers, and does a classification for the object category. Also, a bounding box regression method is applied by implementing linear regression on the region outputs from the final convolutional layer of the main CNN. The final model was validated on the Pascal VOC 2007 and 2012 datasets [7, 8], and it achieved satisfying performances compared to other predecessors (e.g. DPM [9]).

An improvement of this method was implemented by Girshick 2015 [11] who proposed the Fast R-CNN model, whose architecture is shown in Figure 2.4. The input images pass through a convolutional backbone which outputs a feature map for the entire image. The object proposals, pass through a Region of Interest (RoI) pooling layer, which essentially takes the corresponding

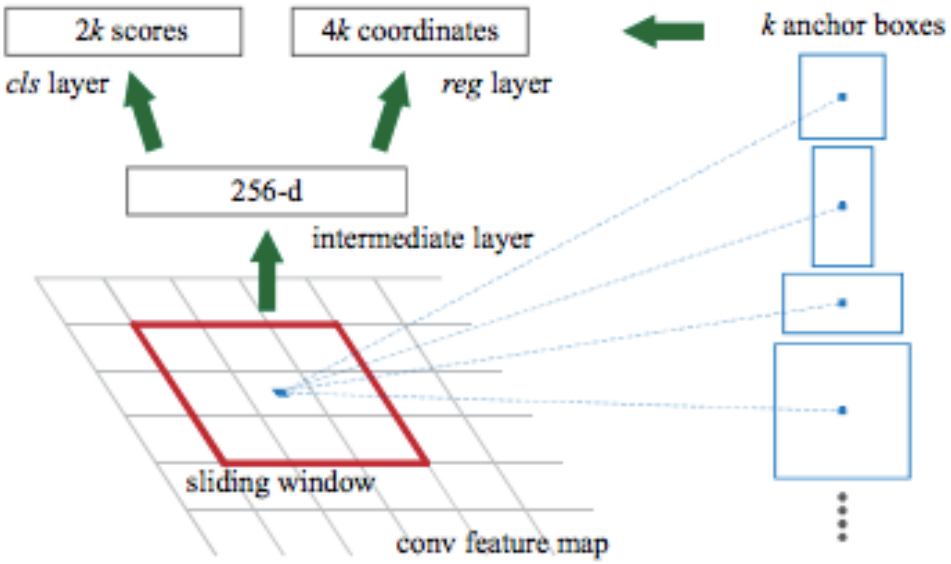


Figure 2.5: Visualization of the RPN architecture [30]

feature map region coming out of the CNN, and uses max-pooling to map it into new spatial dimensions which remain fixed and are the same for all proposals. Each of these regional feature maps coming out of the pooling operation, pass through two Fully Connected Layers (FCs), which transform these maps into feature vectors. Then, they pass through two separate FCs, one responsible for the classification of the region, and the other for bounding box refinement. In comparison with the immediate predecessor R-CNN, Fast R-CNN managed to surpass it on the Pascal VOC 2007 and 2012 datasets by 4 and 6% increases in mAP respectively.

Ren et al. 2015 [30], managed to further improve the existing two-stage object detectors by proposing the Faster R-CNN model. Its most significant contribution was the introduction of the Region Proposal Networks (RPNs), an illustration of whom can be seen in Figure 2.5. These RPNs eliminate the need for a predetermined object proposal generation using other algorithms like Selective Search. Instead, the images that pass through a convolutional backbone, yield feature maps that are then used by the RPN to generate proposals using only the information from the feature maps. It is essentially a deep learning framework that can learn itself which are the most probable candidate regions. When it comes to their architecture, RPNs are essentially FCNs which use kernels of fixed size that are convolved with the input, and map it into a lower dimensional vector. The output of that intermediate layer

is subsequently passed through two different FCs, one for classification and the other for bounding box regression. Regarding the detection, during the convolution of the kernel with the feature map, at each location of the kernel there are as many predictions as a pre-determined number of anchor boxes of varying aspect ratios. Therefore, the classification and box regression layers, yield 2k and 4k outputs respectively, where k is the number of anchor boxes. In order to allow the utilization of the generated proposals by the detection network, a scheme that enables the sharing of convolutional layers between the RPN and the main network is implemented. This is done by starting the training of the RPN and then using its output as the proposals required by the main detector. Subsequently, the detector initializes the new RPN training and keeping the common convolutional layers fixed, the only fine-tuning occurs to the RPNs FC layers. Other implementation details include the use of SGD for optimization and non-maximum suppression for the reduction of proposals that have significant overlap with each other. This model was also trained on the Pascal VOC 2007 and 2012 datasets, and the result was that by using RPNs instead of other proposal mechanisms like Selective Search, the precision on the test sets of both datasets was increased and the inference speed was significantly reduced.

2.4 Oriented Object Detectors

Due to the demand for the more accurate localization of objects in an image, as well as for the determination of other object characteristics, such as their orientation, extensive work has been done to develop detectors that generate oriented bounding boxes. Jiang et al. 2017 [16] proposed the Rotated Region CNN (R^2CNN) shown in Figure 2.6. It mostly follows the same approach as Faster R-CNN [30]. An RPN is used to extract Regions of Interest from the feature maps that come out of the convolutional backbone. The proposed regions are axis-aligned and therefore not taking into account arbitrary orientations yet. Then, they are passed through RoI pooling layers that create three different pooled feature maps in order to account for varying object shapes. Finally, two Fully Connected layers are used with three objectives. The first is to classify the proposals, the second is to predict the HBBs and the third to predict the OBBs. After their generation, an inclined non-maximum suppression is applied to remove multiple detections in the scene. Experiments done on the ICDAR2015 dataset [17], showed that by implementing this model using all three pooling shapes and non-max suppression on the inclined bounding boxes, there is an increase in Precision, Recall and F-measure by 20, 31 and 26% respectively, compared to the Faster

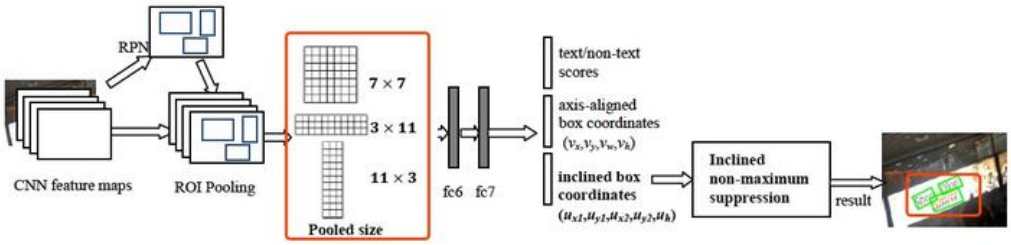


Figure 2.6: Visualization of the R^{2}CNN architecture [16]

R-CNN model.
Notably, the RPN in the previous work, created horizontal region proposals despite the fact that the final predictions included OBBs. Ma et al. 2018 [27] proposed the Rotated Region Proposal Networks (RRPNs) which generate oriented regions that are essentially rotated rectangles. Like in Faster R-CNN [30], RRPNs consist of convolutional layers which use kernels to create lower dimensional representations of the input data. However, instead of using anchors that vary only in terms of scale and aspect ratio, the parameter of the orientation is also included to account for the different directions an object may point at. A proposal is generated if there is a significant overlap between ground truth and generated oriented proposal. This overlap is calculated by the IoU of the skewed rectangles and if the IoU is over 0.7 and the deviation of the respective orientations is less that $\pi/12$, it is considered a valid generation to be used for training. These outputs pass through two separate FCs one used to classify the proposal, and the other to regress the bounding box coordinates. The proposals that come out of the RRPN pass through a Rotated Region of Interest Pooling (RRoI) layer that implements max pooling on oriented feature regions. This is done in the following fashion. A proposal is generated with corresponding parameters (cx, cy, w, h, θ), where w, h are the width and height of the oriented proposal and θ its orientation. The proposal is multiplied by the rotation matrix with angle θ, which brings it to an upright direction, and re-scaled to match square shape. Then, the max pooling is done to the transformed region which reduces the feature dimensions. Finally, the output from the RRoI layers passes through Fully Connected layers responsible for classifying these regions into foreground or background. The loss is of a multi-task form which accounts for errors in classification and deviations in the bounding box regressions, with a weighted trade-off between these two terms. Extensive experiments done using the ICDAR2015 dataset [17], showed that by using RRPNs, there are significant improvements in all three main detection metrics, compared to several other contemporary oriented detectors.

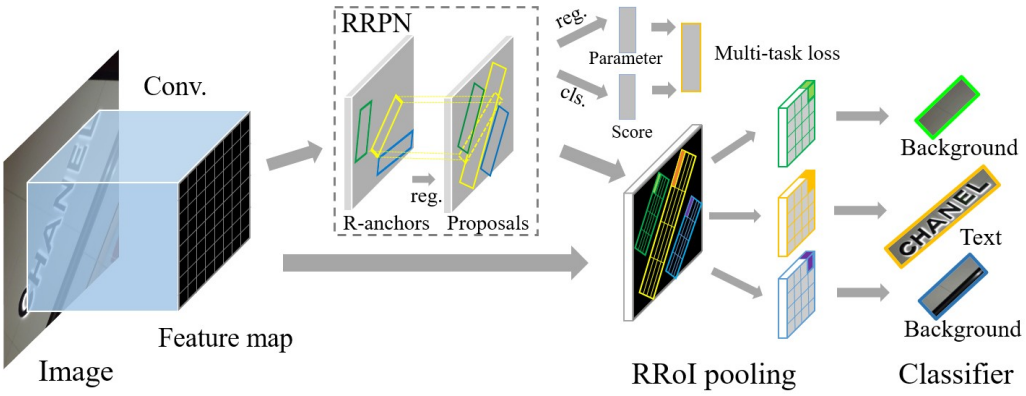


Figure 2.7: Visualization of the RRPN architecture [27]

Ding et al. 2019 [6] proposed the RoI Transformer, seen in Figure 2.8, which also aimed to bridge the gap between horizontal and oriented proposals. It consists of two main components. The first is the Rotated Region of Interest (RRoI) Learner. Its main objective is to learn the RRoIs using HRoIs coming out of the convolutional backbone. This is done by taking all the proposed HRoIs and their corresponding feature maps, passing them through fully connected layers, whose output is a vector of relative offsets that can be translated into predicted OBB coordinates using translation between global to local coordinates system. For training efficiency, before the OBB predictions are done, each HRoI is matched to a unique ground truth HBB, and then the ground truth parameters of the corresponding OBB are used for the regression of the predicted OBB parameters. The second component is the RRoI Warping module which translates the learned RRoIs into rotation invariant feature maps that can then be used to pool rotation invariant features. This warping is done by dividing the RRoI into squared bins, and then doing a rotation matrix operation on the input feature map at each bin and each channel. The output of the RRoI Warping is combined with the feature map and passes through two sibling fully connected layers, one for the classification of the object and the other for the further regression of the final bounding boxes. This work mostly focused on datasets with aerial images, which contain far more complex backgrounds and far more intense variations when it comes to objects' scales, aspect ratios and orientations. This work was validated using the DOTA dataset [35] which is a large scale aerial image dataset used for object detection, and the HRSC2016 dataset [26] which focuses on ships from aerial images. In both datasets RoI transformer showed that compared to methods like R^2CNN [16] or RRPN [27],

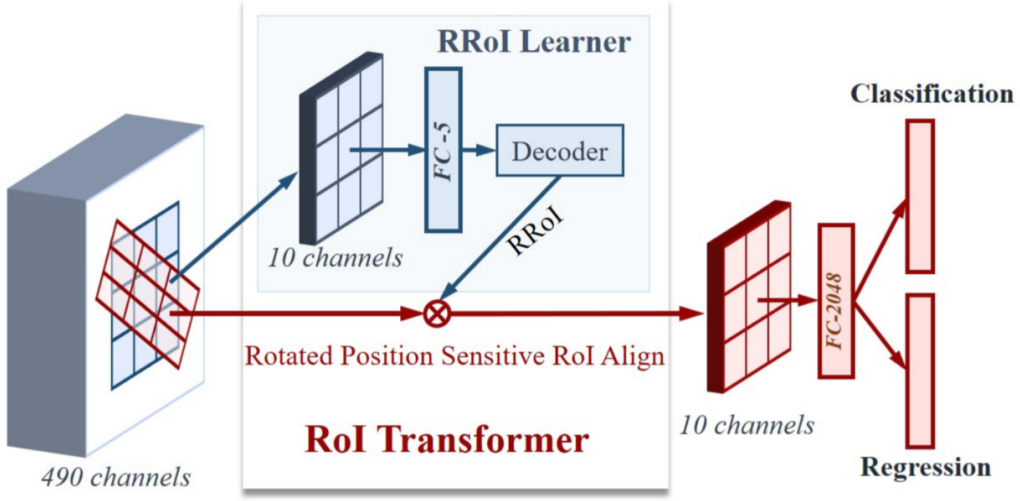


Figure 2.8: Visualization of the RoI transformer architecture [6]

it achieves significant improvements with regard to mAP by about 9% in DOTA and 7% in HRSC2016. With respect to other aspects of the efficiency of the method, regarding the train and test times it is slightly slower. Additionally, the RRoIs are slightly enlarged in order to contain background information to enhance the model's predictive performance.

An improvement made to incorporate oriented object detection capabilities using one-stage detectors, was done by Yang et al. 2021 [37] who proposed R^3Det. Using the RetinaNet architecture as a baseline, they took advantage of the network's speed and then introduced feature refinement modules (FRM) to further improve the detection accuracy. In the initial phase, the output of the classification and regression sub-networks of RetinaNet (see also Figure 2.3) are passed through the FRM, which initially implements bilinear feature interpolation to get accurate location information about each feature. Then, the initially predicted box is translated, to account for mapping transformations. A large kernel convolution done on the feature map is then added with the refined feature maps to yield the final result. Results obtained by training the model on the DOTA and HRSC2016 datasets, showcase its increased predictive capabilities, even compared to two-stage counterparts (i.e. RoI transformer), as well as its increased speed which is more than twice that of most two-stage detectors.

Another important issue that was resolved by Han et al. 2021 [14], was that the features coming out of convolutional backbones themselves, are not rotation invariant. Therefore, extracting orientation equivariant features would lead to better detection performances in oriented objects. This module was

called ReDet, and it consists of two main components. The first is a rotation-equivariant CNN, which extracts features that under image rotation, are also rotated in the same manner. These features then pass through RPN and RoI transformer in order to generate RRoIs. The second component is a rotation invariant RoI alignment module, whose aim is to generate rotation invariant features given rotation equivariant features from the convolutional backbone. This is done by initially applying warping on the RRoIs, which resolves the spatial alignment task. After that the features of the warped region are switched along their orientation channel dimension in a way that the orientation at which they point is the first they encounter. Also, in order for that to be made feasible, interpolation is applied since the channel orientations have a pre-determined granularity. This model provided a new benchmark for aerial object detection since on the same datasets that R^3Det and RoI transformer were trained, it yielded 3 and 10% improvement respectively, with regard to the mAP at the test set of the DOTA dataset, and 1 and 4 % for the HRSC2016 dataset.
Xie et al. 2021 [36] proposed the Oriented R-CNN which improves the Region Proposal Networks and generates reliable oriented proposals at considerably faster speeds. Its architecture consists of a feature proposal network backbone from which feature maps are extracted at different scales, in order to obtain meaningful proposals for objects of varying sizes. Each feature map passes through a convolutional layer, and during the kernel convolution, at each point 3 horizontal anchor shapes are examined which create the first stage proposals. These proposals are convolved with a 1×1 kernel which corresponds to the regression branch that is used to decode the oriented proposal parameters. At the second stage, these oriented proposals pass through a RoI alignment module which extracts a fixed-size feature vector to be used for classification and coordinates refinement. Its performance on aerial image datasets, exhibit a considerable improvement compared to its predecessors since both on DOTA and HRSC2016 datasets, the mAP is increased by at least 5% and 1% respectively.
An overview of the most widely used object detectors, along with their stage and box prediction type, is shown in Table 2.1.

2.5 HBB to OBB Transformation

All the detectors mentioned up to this point, are based on the fact that either HBBs or OBBs will be their input annotations. While it is not generally feasible to use HBB annotations to predict OBBs in any object detector, it is possible to use HBB annotations and transform them to OBBs in order

Method	Number of stages	Box type
YOLO[29]	one-stage	HBB
SSD[24]	one-stage	HBB
RetinaNet[22]	one-stage	HBB
R-CNN[12]	two-stage	HBB
Fast R-CNN[11]	two-stage	HBB
Faster R-CNN[30]	two-stage	HBB
R^3Det[37]	one-stage	OBB
Rotated-RetinaNet[22]	one-stage	OBB
ReDet[14]	two-stage	OBB
Oriented-RCNN[36]	two-stage	OBB
RoI-Transformer[6]	two-stage	OBB
S^2A-Net[13]	two-stage	OBB

Table 2.1: List of popular object detectors, with their number of stages and bounding box type.

to enrich the datasets available for oriented object detectors. There are generally two ways to achieve that. One is to use HBBs to learn object masks, and then use these masks to calculate the corresponding OBBs. One of the first methods which implemented weakly-supervised learning for object masks using only HBBs as input, was proposed by Tian et al. 2021 [32], called BoxInst, who used an adaptive backbone, which dynamically changes the weights of the mask heads, in order to enable the successful discrimination between foreground and background when predicting each instance. They also implement two loss terms, one for minimizing the distance between the projected HBB of the predicted mask and the ground truth HBB, and the other to ensure the pairwise label consistency between neighboring pixels, based on their color information. The other way is to go directly from HBBs to OBBs using weakly and self supervised learning. Yang et al. 2023 [38] developed the H2RBox model, which uses two branches. The first is a weakly-supervised branch which can be any rotated object detector that gives OBBs as output and then it computes its corresponding HBB that gets compared with the GT HBB to compute the loss. The second is a self-supervised branch which uses different angle and scale views of the input images, and then after doing the inverse scale and rotation transformations, calculates the loss of the respective HBB with its GT counterpart.

2.6 Data augmentation

It has been shown that the strategy of tweaking the data in given datasets in order to widen the range of the current data properties (e.g. illumination, scale, rotation), can lead to performance improvements in several computer vision tasks. Krizhevsky et al. [19] and Simonyan et al. [31] show that performing data augmentation using translation and intensity changes leads to reduced image classification errors. Zoph et al. [43], show that in object detection problems, a combination of color and geometric transformations which, in turn, also affects the Bounding Boxes, leads to improvements in the detection performance of several object detectors in different benchmark datasets. A different approach for data augmentation is proposed by Zhong et al. [41], who implement random removal of patches within either the entire image or within object regions. The result of this strategy was an increase of accuracy in both image classification and object detection tasks.

Chapter 3

Methods

3.1 Overview

Our work focuses on improving the automated transformation of Horizontal to Oriented Bounding Boxes (H2OBB) by implementing an HBB → Mask → OBB approach. While this approach is similar to BoxInst [32], it has significant differences with regards to the separate components of the model. The proposed method (see also Figure 3.1) consists of three main components. The first part is responsible for the instance segmentation of objects within an HBB region of an image. This is done using the segment anything model (SAM), which takes as input the images of the dataset as well as the objects' HBB annotations. The output of this component is in the form of object masks, which are binary arrays signifying the object regions. The second component is responsible for the correction of the resulting masks and the removal of artifacts. This is done by implementing a morphological closing operation on each of the resulting masks. Finally, a contour detection method is applied on the closed masks in order to compute the border regions of the object. Using this information, one can calculate the minimum area rectangle which surrounds the computed contour, and in turn obtain the width and height of the OBB. Also, since the main direction is computed, one can obtain the orientation of the mask as well, which leads to the 5-elements tuple (cx, cy, w, h, θ) that unilaterally define an OBB. The final masks are then passed through the ground truth overlap calculator, which calculates the IoU between the HBBs of the predicted masks and those of the respective ground truth objects. We manually define a threshold for the IoU, above which we consider the detection as valid. In this case, the OBBs of the accepted objects are then represented as the 2D coordinates of their four corners and constitute the final output of the model.

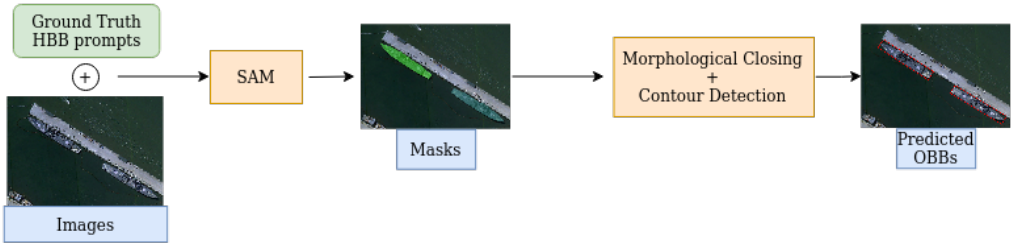


Figure 3.1: Overview of our proposed method. The images, together with their HBB annotations, are fed as prompts to the SAM model, and the calculated object masks pass through a morphological closing operation and a contour detection function that finally provide the corresponding OBBs.

Given the generated OBB annotations, we also examine how improved the detection performance of existing oriented object detectors would be, if the training datasets were pre-processed in a way that resolved spatial imbalances and more specifically, imbalances with respect to the objects' orientation. Most works do this by using data augmentation techniques, which however do random rotations and rescalings that do not necessarily lead to uniform orientations in the new dataset. In this work, we resolve this problem as well, by implementing an iterative algorithm which rotates the images in a way that constantly tries to minimize the variance in the orientation histogram of the objects at hand. We can either do this by maintaining the same number of images, or by increasing them. By the time the maximum number of objects is reached, it is guaranteed that the orientation histogram of the resulting dataset will be as uniform as possible and thus the orientation imbalance will be effectively eliminated.

3.2 SAM

The Segment-Anything Model (SAM) [18], is a novel image segmentation method which enables a zero-shot generalization, namely it can segment objects even in images that are in a significantly different domain than the one on which the model was trained. The implementation is based on an encoder-decoder architecture. Two different encoders are implemented, each taking a different type of input. The first takes the image as input and maps it into an embedding space. The second takes prompts as input, which can be dense (mask) or sparse (points, boxes). The purpose of these prompts is to determine the areas where the segmentation has to focus on. The outputs from both encoders are then passed through a mask decoder which returns different sets of segmentation masks that define the segmented regions of an

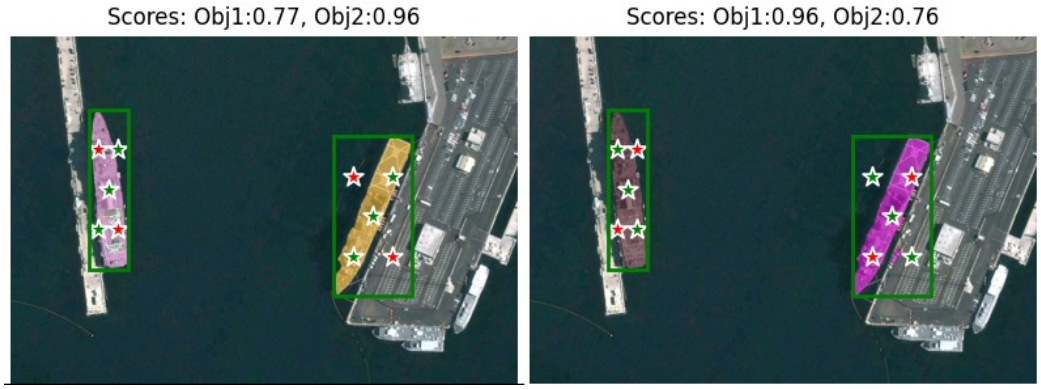


Figure 3.2: Segmentation results for an image with two objects. Green and red stars correspond to foreground and background points respectively. The green boxes that surround the objects, are their HBBs. In the left figure, foreground points lie on the diagonal directed from bottom left to top right, and vice versa in the right figure. Each setting yields a different score for the segmentation of each object.

image, each with a confidence score.

The most important factor that determines the quality of the detection, is the accuracy with which we determine the region corresponding to an object. To achieve this, we make use of the SAM predictor and we apply it to all images and all objects. SAM can segment objects in a specified region given a set of m 2D points which are specified to belong to the image foreground or background.

The main assumption upon which our implementation is based, is that the objects at hand are elongated, meaning that their length is considerably larger than their width. Since the objects we are interested in are obtained from aerial images, we expect that this is a valid assumption. An important feature that elongated objects present however, is that their orientation will lie upon one of the two diagonals of the surrounding HBB. Due to the fact that we don't have any prior knowledge regarding which diagonal that is, we adopt the following strategy. We take the center of the object's HBB as a foreground point because we are sure that it belongs to the object. The remaining $m-1$ points are equally distributed across the two diagonals so that each diagonal has $(m-1)/2$ points, symmetrically placed with accordance to the center point. Then, we run the segmentation for two cases as shown in Figure 3.2. In the first case, the input points of one diagonal are treated as foreground points and the rest as background, and in the second case the opposite. Both segmentations yield a score that quantifies SAM's confidence about the accuracy of the segmentation and we assume that the one with

the highest score has correctly segmented the object at hand. The output of the segmentation for one image, is a set of binary masks, each corresponding to one object. Each mask has the same size as the input image, with ones in the pixels that belong to the object and zeros otherwise.

3.3 Morphological filtering

The segmentation masks specify which pixels belong to a certain object. However, a problem that arises is that the pixels belonging to the mask do not necessarily form a single connected component. In several cases there are gaps in the interior and the border regions of the mask. This is resolved through a morphological closing operation:

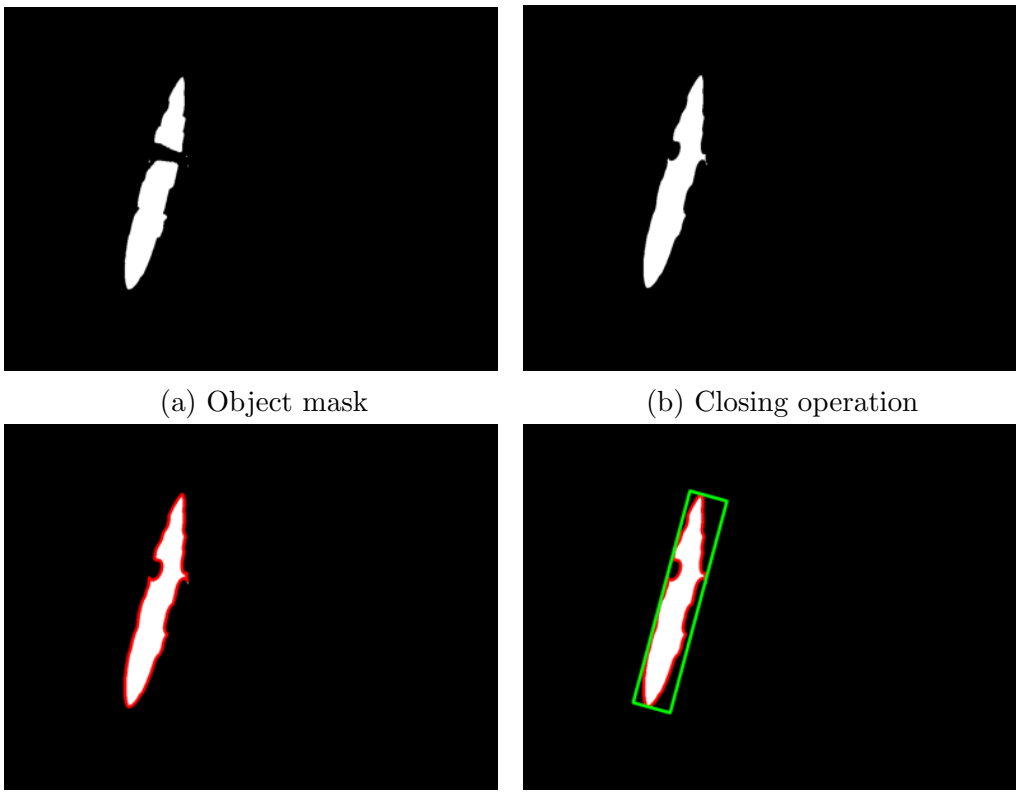
$$A \bullet B = (A \oplus B) \ominus B. \tag{3.1}$$

In Equation (3.1), A is the binary mask resulting from SAM and B is the structuring element used for morphological filtering. Symbols $\oplus$ and $\ominus$ denote dilation and erosion operations, respectively.

Essentially, the closing operation firstly expands the mask according to the structuring element, which leads to the filling of any gaps within the mask, and then reduces it by the same element, which removes the expanded regions at the edges of the object area. In general a structuring element can have any shape and size. Therefore depending on the task at hand, we have to carefully select a setting that will be suitable for the shape of the objects in the datasets. Popular choices for element shapes, are squared and circular ones. However, in aerial images, most objects (e.g. trucks, bridges, ships) have an elongated shape as mentioned before, and this allows for the use of an elongated element as well. We therefore make use of either a circular or an ellipse-shaped structuring element B, whose size is adaptable to the length of the object at hand. The ellipse however, due to its eccentricity, has to be directed in the same angle as the object. To achieve that we find the diagonal of the HBB upon which the object lies, via the SAM segmentation score, and then we calculate the inverse tangent of the angle formed between the object's direction and the horizontal x-axis. This gives the orientation the ellipse should have, and we also specify the minor and major axes.

3.4 Contour detection

After the closing operation, the resulting masks are used for the calculation of the object region boundaries. This is done by implementing a contour



(a) Object mask (b) Closing operation

(c) Contour calculation (d) OBB calculation

Figure 3.3: Process of obtaining an object's OBB. (a) The initial object's mask. The gap between its two components is apparent. (b) Shows the mask after the closing operation which unites the two components. (c) Calculation of the contour surrounding the mask (red). (d) Calculation of the OBB (green).

detection method which initially detects changes in the intensity of the binary image, and then after implementing a connected component analysis, it returns one or more sets of points, that enclose the regions containing the object pixels. The reason why there may be more than one contours, is that even after the closing operation, it is possible that some isolated blobs, signified as object regions, are not included within the main object region. To resolve this, we assume that the contour corresponding to the object region, is the one encompassing the largest area in terms of pixel amount.

After computing the contour surrounding the object mask, we obtain the optimal OBB surrounding it, by calculating the minimum bounding rectangle that encompasses the contour. This is done via a convex hull computation

which is done using the points belonging to the object's contour. The resulting hull is then used to fit a minimum area rectangle, via iterative rotations of a rotated rectangle around the hull, until the minimum area rectangle is obtained, which will also yield the width, height and orientation of the OBB. The center of the OBB is obtained from the centroid of the contour points. The center, width, height and orientation of the object are all the parameters needed to define an OBB. An example of the mentioned steps is shown in Figure 3.3.

A way to prove that the segmentation has been done correctly, is to compare the ground truth HBB of each object with the HBB that would be computed, given the segmentation mask of the respective object. The predicted HBB of an object is computed in the following way. The edge points of an object's segmentation mask are considered to be the same as the ones that would define the HBB surrounding it. We then calculate the IoU between the ground truth and predicted HBB of each object, and define a hyperparameter that corresponds to the IoU threshold above which we consider the segmentation as correct. The masks of the correctly segmented objects, will subsequently be used for the calculation of their respective OBBs.

3.5 Dataset Augmentation

Aerial image datasets contain images of objects in various orientations. However, there might be significant imbalance in the number of object samples per orientation. Given the proposed method, we can correct such imbalances. Specifically, by having access to the OBBs of the objects, we can augment the given imbalanced dataset so as to cover the space of ship orientations more uniformly. This can be done in several ways; in this work we explore two such techniques.

Same Size Object-wise (SSO) augmentation: We create an augmented dataset, with the same number of samples as the original dataset but with images rotated in such a way so that objects are as equally distributed among orientations as possible. To do that, we firstly create an empty histogram, hereby referred to as SSO-Histogram, with as many bins as the number of quantized ship orientations. Then, we sort the images according to the number of included objects, in descending order. Starting with a random image from those with the highest number of objects, we calculate their orientation histogram and add it to the SSO-Histogram. The remainder of the process is the following. We iteratively select random images, calculate the orientation histogram of the included objects, and then rotate the image in such a way so that the newly updated SSO-Histogram has the minimum variance. Each

time an image is selected, it gets removed from the selection pool, so that the random selection begins with images with the most objects and gradually with images with fewer ones. The iterations terminate when the total number of objects in the SSO-Histogram reaches the total number of objects in the dataset. Refer to Algorithm 1 for more details.

Increased Size Object-wise (ISO) augmentation: The second way for performing augmentation is to increase the number of objects in underrepresented orientations so that each orientation has twice as many samples as the most prevalent orientation in the pre-augmented dataset. To do this we use the dataset's orientation distribution with the same quantization as before, and we assign it to an ISO-Histogram. We select the orientation with the most objects and define the upper bound of all the orientation bins as the double of that number of objects. If for example, the dataset has 80 objects pointing at 70 degrees, the upper bound for all bins is set to 160. The next step is to constantly select random images, and rotate them in such a way that at least one of the objects is brought to the most prevalent orientation, while simultaneously adding the orientation distribution of the rotated image, to the ISO-Histogram. This is done until the number of objects in that orientation, reaches the upper limit. Then, we iteratively select random images from the dataset, we calculate their objects' orientation distribution, and rotate the image by the angle at which the updated ISO-Histogram will have the minimum variance. This is done while constantly checking if at any orientation the number of objects exceeds the upper bound, and if it does we select another image. The iterations terminate when the total number of objects in the ISO-Histogram reaches the upper bound per orientation times the number of orientation bins, or if there are no more images left to fill the underrepresented orientations. A pseudocode representation of this process is shown in Algorithm 2.

The calculation of the rotated OBB coordinates is done by applying the rotation matrix on the OBB center, and because the image orientation increases the image size, the center points are adjusted to the new image size, as shown in Equations 3.2, 3.3,

$$c'_x = cos\theta \cdot \left(c_x - \frac{w}{2}\right) - sin\theta \cdot \left(\frac{h}{2} - c_y\right) + \frac{w'}{2} \tag{3.2}$$

$$c'_y = -\left(sin\theta \cdot \left(c_x - \frac{w}{2}\right) + cos\theta \cdot \left(\frac{h}{2} - c_y\right) - \frac{h'}{2}\right), \tag{3.3}$$

where θ is the difference between the object's orientation and the orientation to which we want to bring it, w and h are the width and height of the original

image, and w' and h' are the width and height of the rotated image, respectively. The height and width of the OBB remain the same since the number of pixels corresponding to each of the object's sides, is rotation invariant and also invariant with respect to the image size.

Algorithm 1 SSO Algorithm

N: total number of objects
CH: current orientation histogram
UB: upper object bound per orientation bin
n: current number of objects in histogram
w: angle bin width
$n \leftarrow 0$
$UB \leftarrow N * w/180$
CH $\leftarrow \emptyset$
while $n < N$ **do**
  $f\_max \leftarrow files\ with\ maximum\ number\ of\ objects$
  $f\_curr \leftarrow random(f\_max)$
  $FH \leftarrow orientation\ histogram\ of\ objects\ in\ f\_curr$
  if $n = 0$ **then**
    $CH \leftarrow CH + FH$
    Rotate image and annotations by 0 degrees
  else
    Rotation angle $\leftarrow$ 0
    min variance $\leftarrow$ Var(CH,FH)
    for $0 \leq i \leq 180/w - 1$ **do**
      Shift FH elements by index i
      variance $\leftarrow$ Var(CH,FH')
      if variance<min variance **then**
        Rotation angle $\leftarrow$ i*w
        min variance $\leftarrow$ variance
      end if
    end for
    CH $\leftarrow$ CH + FH'
    Rotate image and annotations by "Rotation angle" degrees
  end if
  n $\leftarrow$ sum(CH)
  remove f_curr
end while

Algorithm 2 ISO Algorithm

```
CH: current orientation histogram
UB1: number of objects in orientation with max objects
n: current number of objects in histogram
w: angle bin width
UB1 ← max(Data Hist)
ind max ← index(max(Data Hist))
n ← UB1
CH ← Data Hist
while CH[ind max]<2*UB1 do
    f_curr ← random(files)
    FH ← orientation histogram of objects in f_curr
    rand ind ← random index with non zero objects in FH
    Rotate image and annotations by (ind max - rand ind)*w
    FH' ← shifted FH by (ind max - rand ind)
    CH ← CH + FH'
    n ← sum(CH)
end while
UB2 ← n
while n<max(CH)*180/w do
    f_curr ← random(files)
    FH ← orientation histogram of objects in f_curr
    Rotation angle ← 0
    min variance ← Var(CH,FH)
    for 0 ≤ i ≤ 180/w − 1 do
        Shift FH elements by index i
        variance ← Var(CH,FH')
        if variance<min variance then
            Rotation angle ← i*w
            min variance ← variance
        end if
    end for
    CH ← CH + FH'
    Rotate image and annotations by "Rotation angle" degrees
    n ← sum(CH)
end while
```

Chapter 4

Experiments

4.1 Datasets

In order to validate the effectiveness of the developed method, we run experiments on three benchmark datasets for aerial object detection, DOTA [35], HRSC2016[26] and ShipRSImageNet[40]. All three, are remote sensing object detection datasets, that consist of high resolution images containing objects of several types. They provide extensive annotation files that include both HBBs and OBBs for the image objects. HRSC2016 and ShipRSImageNet are datasets that focus specifically on ships from satellite images. DOTA is a more general aerial object detection dataset that consists of 16 classes that are entirely different from each other.

DOTA v1.5 is a benchmark aerial object detection dataset, consisting of images with significantly large sizes, that are mostly around 4000×4000 pixels. There are in total 2806 images which include 188282 objects. Their annotations come in the form of both HBBs and OBBs. The OBBs are 4-point quadrilaterals, namely for each object, its OBB is in the form of (x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4). The 1.5 version of the dataset which is used in this work, consists of 16 object categories. These are: plane, ship, storage tank, baseball diamond, tennis court, basketball court, ground track field, harbor, bridge, large vehicle, small vehicle, helicopter, roundabout, soccer ball field, swimming pool and container crane. The split into training, validation and test sets, attributes 1411, 458 and 937 images to the sets respectively, that are randomly selected in order to avoid biases. With regard to the distribution of objects in an image, there are cases where there is no object in an image and cases with as many as 10000 instances. In general however, the distribution follows a logarithmically decreasing trend of the number of images with respect to the number of instances, namely there are many images

with few instances, and less with ever increasing ones. The aspect ratios of both HBBs and OBBs, have wide distributions, with most OBBs having ARs of about 3:1, and HBBs ranging mostly from 1:1 to 3:1. The images themselves are taken from a wide variety of scenarios. There are almost as many images taken at open fields and city environments. This makes the DOTA dataset a very challenging and useful dataset, since it can generalize to most aerial detection scenarios.

HRSC2016 is the most extensively used dataset for optical remote sensing ship detection. Its image sizes are smaller than those of DOTA, which range between 300×300 and 1500×900. There are 1070 images in total, of whom 436 belong to the training set, 181 to the validation, and 453 to the test set. The object annotations include both HBBs and OBBs, which are in the form of (x_{min}, y_{min}, x_{max}, y_{max}) and (c_x, c_y, w, h, θ) respectively. The object orientation θ ranges between $-\pi/2$ to $\pi/2$. This dataset includes images with varying conditions. For example, there are images with ships in the outer sea, moored in harbors, packed close to each other, or even under partial occlusions. Additionally, the classification of the ships follows a 3-level hierarchical classification scheme regarding the object type. The first level simply describes whether an object is a ship or not, and the remaining levels classify the ships in ever increasing category detail (e.g. cargo, oil tanker, aircraft carrier etc.). The deeper the level, the more detailed the classification is. For example, in level 2 the categorization is at the level of separating a merchant ship from a destroyer, and in level 3, the classification specifies even the type of the ship (e.g. Destroyer→Arleigh Burke, Merchant→Container). In this work we are only interested in the level-1 classification, namely simply knowing the existence of a ship.

ShipRSImageNet is a dataset that extends the existing ship detection datasets by including a combination of images from these datasets, as well as images collected from satellites. In total, it consists of 3435 images, 2198 of whom are in the training set, 550 in the validation and 687 in the test set, while the total number of instances is 17573. The majority of the collected images have a size of approximately 930×930 pixels and there is an extensive variety of image conditions, namely with respect to the weather conditions, the image quality, spatial resolution, light conditions, as well as the surrounding environment, since there are many cases of images taken of ships in the outer sea and moored in harbors. The HBBs are again in the form of (x_{min}, y_{min}, x_{max}, y_{max}), but the OBBs are provided in two forms. The first is (c_x, c_y, w, h, θ) where θ is the angle between the object's orientation and the horizontal axis. The second is (x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4), which contains the coordinates of the bounding box edge points, and is the

form we use for the experiments with this dataset. The classification of the objects follows a 4-level hierarchy of detail, but in this case the first level object classification includes the "ship" and "dock" classes.

4.2 Implementation details and experimental setup

As was explained previously, we want to automate the transformation of HBBs to OBBs. Therefore, despite the fact that OBB ground truths are provided in the datasets, we will not use them at all during the creation of our own OBBs. Instead, only the HBBs will be used as input to the model. The expected form of the HBBs is $(x_{min}, y_{min}, x_{max}, y_{max})$.

For the first part of the model, which is the segmentation process using SAM (see section 3.2), we have to specify the number of input prompt points, their discrimination between foreground and background, as well as the coordinates of the HBB within which the segmentation shall take place. We decide to select 5 input points for the HRSC2016 and ShipRSImageNet datasets and 1 point for the DOTA dataset. If one point is used, it is in the center of the HBB and is considered foreground point by default. In the case of the other two datasets, where more than one points are used, one of them is the center of the HBB, and the others are placed halfway between the center of the box and the respective box edge. For instance, the points belonging to the 1st diagonal are the center, the one in the middle between the top left edge and the center, and the one in the middle between the bottom right edge and the center. Similarly, for the 2nd diagonal, the points are the center, the middle between the top right edge and the center, and the middle between the bottom left edge and the center. In general, if the usage of more than one input points is required, their number should yield a modulo of 1 if divided by 4 (i.e. 5, 9, 13, 17 etc.). This is because there should be an equal number of points on each half-diagonal in order to ensure symmetry. Additionally, the respective objects' HBBs are also used as input prompts to SAM.

For the closing operation, we use an ellipse shaped structuring element with minor axis equal to $\sqrt{2}$ times 3% that of the HBB diagonal, for the HRSC2016 dataset. For the DOTA and ShipRSImageNet datasets we use a circular structuring element with the same radius. Regarding the true and predicted HBB overlap calculation, we set the IoU threshold to be 0.6 for the DOTA dataset, and 0.7 for the other two datasets, in order to consider it as a valid segmentation. Objects with invalid segmentation are not considered

parts of the newly generated dataset. After the calculation of the OBBs, we create new annotation files that include the HBBs and OBBs of the objects whose segmentation from SAM was deemed valid. For the augmentation part, we set the discretization of the orientations to be equal to an angle range of 10 degrees for both methods.

The most decisive factor which determines the performance of the proposed method, is its ability to generate OBB annotations that are accurately aligned with the actual objects in the images. To validate this, we make use of several oriented object detectors by training them using the generated annotations from our method, and testing their performance on the test sets that were left intact during the entire process. However, a fair evaluation should take into consideration that any object detector would itself have a certain performance on the original datasets. Therefore, in order to estimate the relative effectiveness of our proposed method, we also need to train the object detectors that are to be used, on the training sets of the original datasets, and then compare their respective performances on the test sets. The oriented object detectors that we use, are obtained from the OpenMMLab project [42], and they are all implemented with ResNet-50 [15] backbone, pretrained on ImageNet [5]. The optimizer used is SGD with momentum, where the momentum is 0.9 and the weight decay is 1e-4. These object detectors are either single-stage (e.g. R^3Det [37]), or two-stage (e.g. ReDet [14], Oriented-RCNN [36]). The learning rate for the ReDet detector is 0.01, and 0.0025 for all the other detectors used. We used one NVIDIA GeForce GTX 1080 Ti GPU with 11GB RAM.

4.3 Evaluation metrics

The evaluation metric used for the detection performance is the mean average precision (mAP) as it is used in PASCAL 2007. The training lasts for 36 epochs using the HRSC2016 and ShipRSImageNet datasets, and 12 epochs for the DOTA dataset, since it has been found that for most object detection tasks, it is a reasonable limit before convergence is reached. In the case of the HRSC2016 dataset, we make use of the training set only and not the validation set. Therefore, the results obtained from the training of selected object detectors will not be the same as the ones presented in the original papers (i.e. [37, 14, 36, 6, 22, 13]), but since the purpose of this work is to provide a comparative assessment between original and generated OBBs, our main focus is not on the performance metric values of the detection, but on the relative differences between the detection performances yielded from training on the different sets.

Type	Size	IoU threshold			
		90%	80%	70%	60%
circle	$\sqrt{2}$*3%	76.22	96.52	98.67	99.50
	$\sqrt{2}$*6%	75.97	96.93	99.25	99.83
	$\sqrt{2}$*9%	74.81	96.77	99.34	**99.92**
	$\sqrt{2}$*12%	73.82	96.35	99.34	99.83
	$\sqrt{2}$*15%	73.65	96.27	99.34	99.83
ellipse	$\sqrt{2}$*3%	**78.62**	**97.27**	99.17	99.67
	$\sqrt{2}$*6%	77.30	97.02	**99.42**	99.75
	$\sqrt{2}$*9%	75.14	96.52	99.25	99.59
	$\sqrt{2}$*12%	72.49	95.44	99.01	99.67
	$\sqrt{2}$*15%	69.76	94.12	98.34	99.42

Table 4.1: Different structuring element settings and percentage of objects from the HRSC2016 dataset that exceed certain IoU thresholds. The types of the element are either circular or elliptical and the sizes correspond to the diameter and minor axis respectively. Their values are in the form of object length percentage.

4.4 Parameter setting and ablation studies

For the implementation of the proposed method, it was important to carefully tune three main parameters. These were, the number of points given as prompts to SAM, the shape and size of the structuring element used for the closing operation, and the IoU threshold between predicted and ground truth HBBs. The prompt points were set to 1 for the DOTA dataset. The reason for that is that in the DOTA dataset, in most cases there are a lot of objects, very closely packed next to each other, which makes the examination of the segmentation along diagonals more harmful that beneficial. As for the HRSC2016 and ShipRSImageNet datasets, the number of points was set to 5, which was the minimum possible from the available selections above 1, because qualitative assessments of the created segmentation masks, indicated that more points lead to segmentation of the background, which is more uniform, and therefore yielding higher segmentation scores than the object. For the same reason, it is very important to also include the HBB annotations as prompt inputs, since failure to do so, results in even more background segmentations, especially in cases of ships in open seas.

For identifying the best configuration of the structuring element, we investigated which setting provided the maximum percentage of objects from the original datasets, for each IoU threshold between predicted and ground truth

Type	Size	IoU threshold			
		90%	80%	70%	60%
circle	$\sqrt{2}$*3%	**9.40**	44.24	**78.98**	**92.60**
	$\sqrt{2}$*6%	9.38	44.06	78.62	92.58
	$\sqrt{2}$*9%	9.09	43.88	78.07	92.46
	$\sqrt{2}$*12%	9.05	43.20	77.61	92.25
	$\sqrt{2}$*15%	8.70	43.29	77.30	91.96
ellipse	$\sqrt{2}$*3%	9.24	44.34	77.65	91.62
	$\sqrt{2}$*6%	9.35	44.88	78.07	91.70
	$\sqrt{2}$*9%	9.39	**45.04**	78.74	92.46
	$\sqrt{2}$*12%	9.09	44.61	78.51	92.48
	$\sqrt{2}$*15%	8.59	43.87	77.87	92.14

Table 4.2: Different structuring element settings and percentage of objects from the ShipRSImageNet dataset that exceed certain IoU thresholds. The types of the element are either circular or elliptical and the sizes correspond to the diameter and minor axis respectively. Their values are in the form of object length percentage.

HBBs. For the HRSC2016 dataset, we present the results in Table 4.1 from which we obtain that an elliptic structuring element with minor axis equal to $\sqrt{2}$ times 3% the length of the object, meets this requirement. The major axis is set equal to twice the minor axis. For the ShipRSImageNet however, the results are different. From Table 4.2 we see that a circular structuring element with diameter equal to $\sqrt{2}$ times 3% the length of the object, maximizes the percentage of objects above 0.9, 0.7 and 0.6 IoU thresholds. Therefore, we will use this setting for this dataset.

Finally, we set the IoU threshold to 70% for the HRSC2016 and ShipRSImageNet datasets, and to 60% for the DOTA dataset. The reason for this comes from information obtained from Figures 4.1, 4.5 and 4.8, where we determine the threshold according to a point above which we can visibly realize that the vast majority of objects lie.

4.5 Segmentation Results

In this part we want to examine the degree at which the model properly creates reliable OBBs based only on the HBB inputs. To that end, we examine both the segmentation masks that were calculated using SAM and morphological closing, as well as the resulting OBBs that are computed by

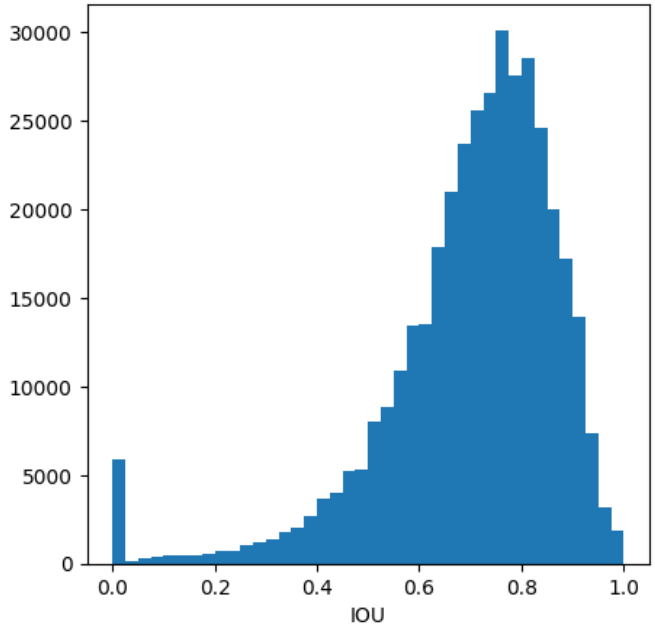
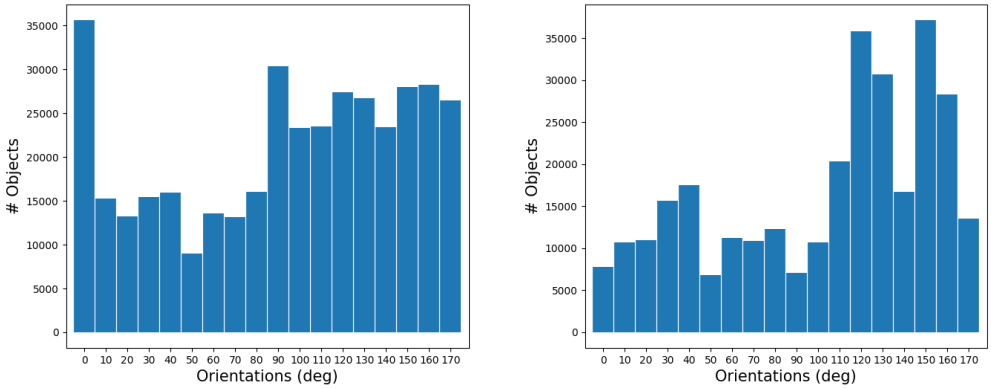


Figure 4.1: Histogram of objects' IoU, between predicted and ground truth HBBs, in the DOTA dataset[35]

the contour detection and the minimum area rectangle operation.

4.5.1 Results on the DOTA dataset

The performance of our model with respect to its ability to generate new OBBs from HBBs using the DOTA dataset, was validated through the examination of several qualitative metrics that provide valuable information regarding the quality of the segmentation and OBB creation process. One such metric is the IoU overlap between the ground truth HBB and the HBB that would have surrounded the object, if its OBB was the one predicted by the model. We show the results in Figure 4.1 from which we observe that the vast majority of objects, have an IoU overlap that exceeds 60%. Specifically, 80% of the total number of objects in the training set exceed this threshold. This is an indication that the model can segment the objects in an image to a satisfying extent. However, simply the overlap between the HBBs is only one part of the picture, which indicates that the model is successful, but it is not guaranteed. This is because there are many combinations of object masks that may lead to very similar HBBs as that of the ground truth. Therefore,



(a) ground truth data (b) generated data

Figure 4.2: Orientation distribution of the objects' OBBs, for the DOTA dataset in the: (a) ground truth data, (b) generated annotations

in order to make a comparison that will focus on the created OBBs, we take the orientation distribution of the objects as computed from the ground truth OBBs and compare it to the orientation distribution that is obtained using the OBBs calculated from our method. The results are shown in Figure 4.2 and we can observe that there is almost no correlation between the two distributions. This means that the subset of the original dataset that is maintained after the segmentation and OBB calculation, differs significantly with respect to the included objects' orientation. While this is not an encouraging indicator, thanks to the enormous number of objects in the dataset, it may not necessarily be considerably detrimental as far as the quality of the generated OBBs is concerned. In order however to obtain a more comprehensive insight regarding the quality of the segmentation masks and the generated OBBs, we evaluate the output of the masks obtained from SAM after the closing operation, as well as the resulting OBBs that arise after the contour detection and the minimum area rectangle fitting. For the most part, the segmentation and the OBB creation is successful, and we present some of the results in Figure 4.3. We can observe that for a wide variety of objects appearing in aerial images (e.g. cars, swimming pools, trucks etc.) the generated segmentation masks are of high quality and indeed accurately encompass the object regions. It can also be observed that this applies for objects of varying scales and orientations, since the presented examples include images with different zoom scales. In the presented examples we can notice that the method works equally well in segmenting either large objects (e.g. tennis courts - case 2) or smaller ones (e.g. cars on highways - case 3). Due to the appearance and

object arrangement trends that are frequent in this data domain, namely the closely packed objects in several cases, we can be confident that the selection of only one prompt point during the segmentation process was a correct call, since a diagonal mask score comparison would include object location in any case and would disregard the object borders. One such example of the ones presented, is the case of the trucks that are parked diagonally next to each other (case 5) and the model manages to successfully segment each one of them. We can also observe that the method works even in cases where an object region is located within another object region. For instance, we present a case of boats that are docked in a marina (case 4), and we see that both the marina and the docked boats are segmented successfully. It is also apparent that the OBBs that are created using the segmentation masks, capture accurately the minimum rectangle of the objects' masks and have the same orientation as the objects' direction. We can see this in several object cases varying in both size and aspect ratio. For instance, in cases of small cars driving on highways (case 3), the OBBs have to be very small and also have to capture the cars' direction, which we can see is indeed the case. However, the pipeline has some flaws, particularly when segmenting objects that do not have a sufficiently elongated shape. In such cases, the model may successfully segment the object, but fail at generating a correct OBB. We show such examples in Figure 4.4. We can see that there are objects like airplanes (cases 2 and 3) or baseball diamonds (case 1), which due to the fact that their shape does not exhibit a predominant direction, it is hard for the model to determine the OBB. This happens because is not guaranteed that the minimum area rectangle will have its small side perpendicular to the front of the object. For example, in the cases with the airplanes, a "correct" rectangle would need to have a side parallel to the wings, and one parallel to the airframe. However, such a rectangle does not satisfy the minimum area requirement, and if we calculate the OBB using the Minimum area rectangle algorithm on the segmented masks, the resulting OBB will have the shape that appears in the examples, namely a rectangle with one edge at the wing tips and the other at the cockpit. More generally however, objects that do not have clearly elongated shapes are challenging for the model, because it is increasingly difficult to have a one to one optimal match between an object mask and an OBB. This means that there can be possibly more than one OBBs that can surround an object. For instance, square shaped objects can have 2 different OBBs, circles may have infinite OBBs in principle. Therefore, it is important to notice that this methodology works best when applied on objects with nearly rectangular shapes. When it comes to the cases where both the segmentation and the OBB generation fail, they are relatively few

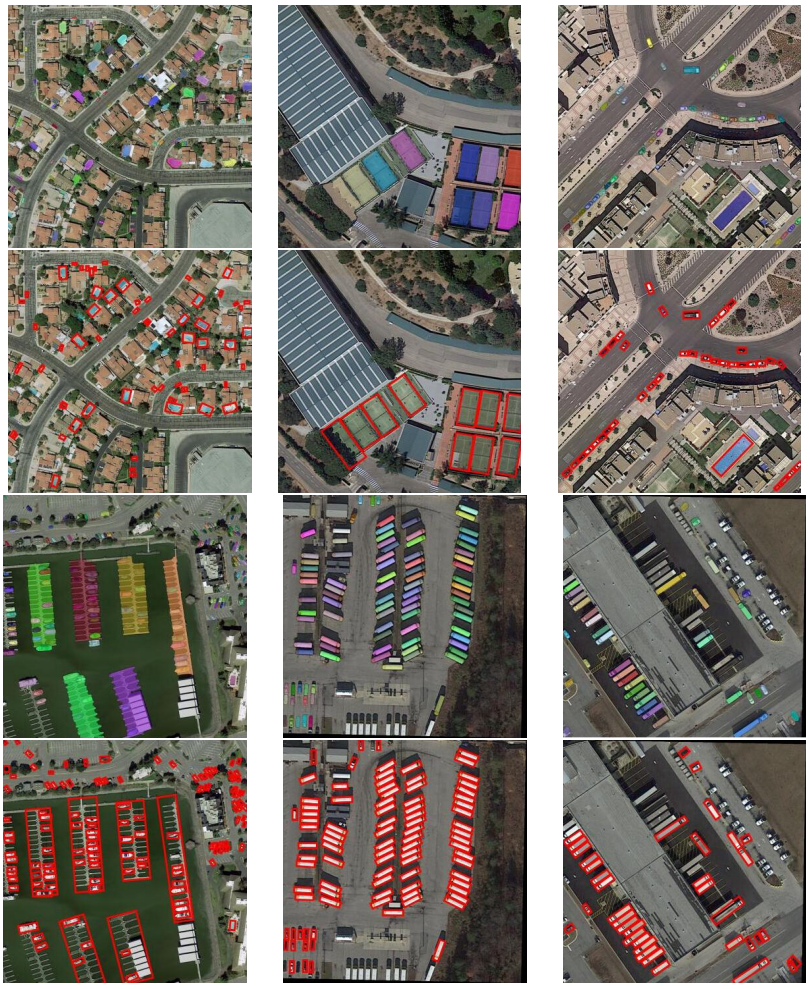


Figure 4.3: Successful examples of segmentation masks obtained from SAM and of resulting OBBs for selected images in the DOTA dataset

in number, and can mostly be attributed to texture similarity with the background (e.g. ground track fields), and also to illumination factors, such as the existence of shadows that are very close to the object from which they originate. These cases are however a very small minority, and for the most part the model is successful in both the segmentation and OBB generation process.

4.5.2 Results on the HRSC2016 dataset

After running the OBB generation model on the HRSC2016 dataset, we calculated the distribution of the IoU overlap between the ground truth HBB

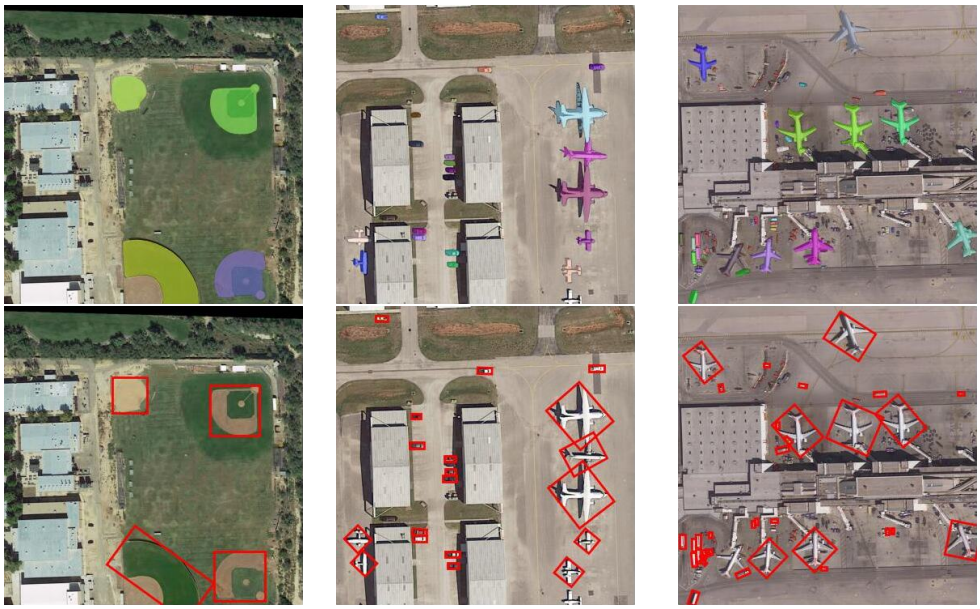


Figure 4.4: Examples of correct segmentations but failed OBB calculations in images of the DOTA dataset

and the predicted HBB. We show the results in Figure 4.5, where we see that the vast majority of objects, have a predicted HBB that exceeds in terms of IoU a 70% overlap with the respective ground truth HBB. In this specific case, we also observe that the IoU distribution is very tight and around 90%. This means that in most cases, the segmentation masks from SAM, combined with the closing operation, lead to predicted HBBs that would most likely be overlapping by 90% with those from the ground truth. This is a far better result compared to that obtained from the DOTA dataset, but we still have to verify the similarity between the orientation distributions of the objects' ground truth OBBs and the predicted OBBs. The results are shown in Figure 4.6, from which we can easily observe that the two orientation distributions are almost identical, both in terms of shape but also in terms of number of objects per orientation bin. This essentially means that the dataset created from the generated OBBs can be considered as of almost the same quality as that of the original dataset. Subsequently, we examine the outputs of the segmentation part of our model in order to see whether they indeed highlight the objects that they were meant to. In Figure 4.7 we present several examples of segmentation masks created for objects in some images. It can be observed that the masks, encompass almost exactly the regions of the ships that they are meant to segment, in a variety of scenarios. In the provided examples, we see that the segmentation is successful in cases

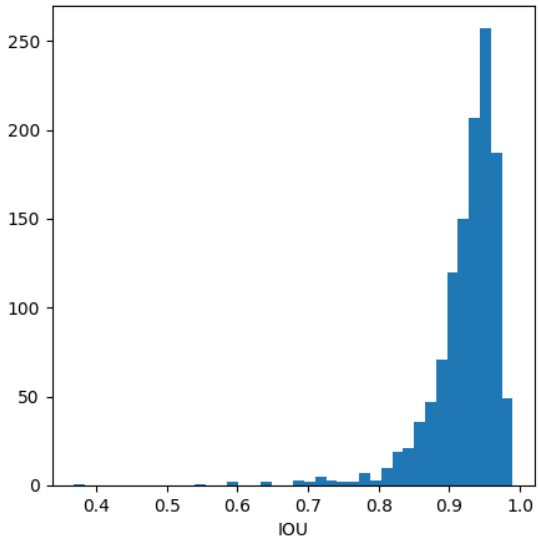
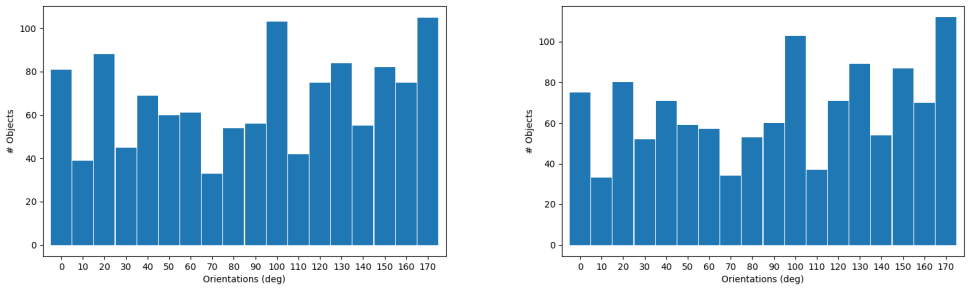


Figure 4.5: Histogram of objects' IoU, between predicted and ground truth HBBs, in the HRSC2016 dataset[26]

where the ships are in the open sea, docked in harbors, both in distance and closely moored to each other, even in cases where there are partial occlusions like cranes that pass on top of the ships. This robustness can be attributed to several factors. The most significant one is the usage of input points in alternate diagonals. In nearly every scenario, the ship regions, upon which one of the diagonals lies, are more uniform compared to the surrounding areas. For example, in the case of a ship moored in a dock, the other diagonal, namely the one that does not contain the ship in question, is very likely to include sea surface, harbor environments, or ships of different styles, among the center of the ship itself. Also, another factor is the proper location of the prompt points on the diagonals. The ships in the dataset have varying widths and lengths, which means that if the ship is very wide and the prompt points are placed close to the center, then it is likely that in both diagonals all points will belong to the object. This is why the points are selected to lie at the middle of each half-diagonal. Given the visual quality of the generated masks, we can be confident that the model is successful in its first part which is the correct determination of the objects' regions. Additionally, in the same figure we provide the respective results with regard to the calculation of the objects' OBBs. We can easily observe that all OBBs do indeed tightly enclose the ships in the images, and perhaps in some cases in an even more



(a) ground truth data (b) generated data

Figure 4.6: Orientation distribution of the objects' OBBs, for the HRSC2016 in the: (a) ground truth data, (b) generated annotations

accurate way than a human annotator would manage. However, there are a few cases where due to the probable inclusion of areas near the ships' edges as object points, the resulting OBBs have slight deviations, with regard to their orientation, compared with the ships' bow. Despite these slight issues, we can realise that the generated OBBs from our method are more than reliable for usage in object detection tasks.

4.5.3 Results on the ShipRSImageNet dataset

The final dataset on which the model is to be evaluated, belongs to the same domain as HRSC2016 and focuses primarily on ship detection. ShipRSImageNet is far more extensive, both quantitatively and in terms of variety of conditions. Therefore, a qualitative assessment of our model on this dataset will determine its robustness if the images domain gets more challenging. In Figure 4.8 we present the histogram of the IoUs between the ground truth HBBs from the training set objects, and the predicted HBBs obtained after the segmentation, closing and OBB prediction steps. The majority of the objects have an IoU overlap that exceeds 60%, which is a good indication about the quality of the predicted masks. We can also observe that there is a peak of the histogram, centered at around 80% IoU overlap, at which point approximately 1000 to 1200 objects lie. However, the distribution around that peak is wider and the peak itself is located at an IoU smaller by 10% compared to the HRSC2016 dataset. While this suggests that the segmentation is of lesser quality than that in the previous dataset, it is worth mentioning that there are several factors which may handicap this metric. One such factor is that the HBB annotations in the dataset's ground truth are obtained by the extreme points of the respective ground truth OBBs. This is not how we

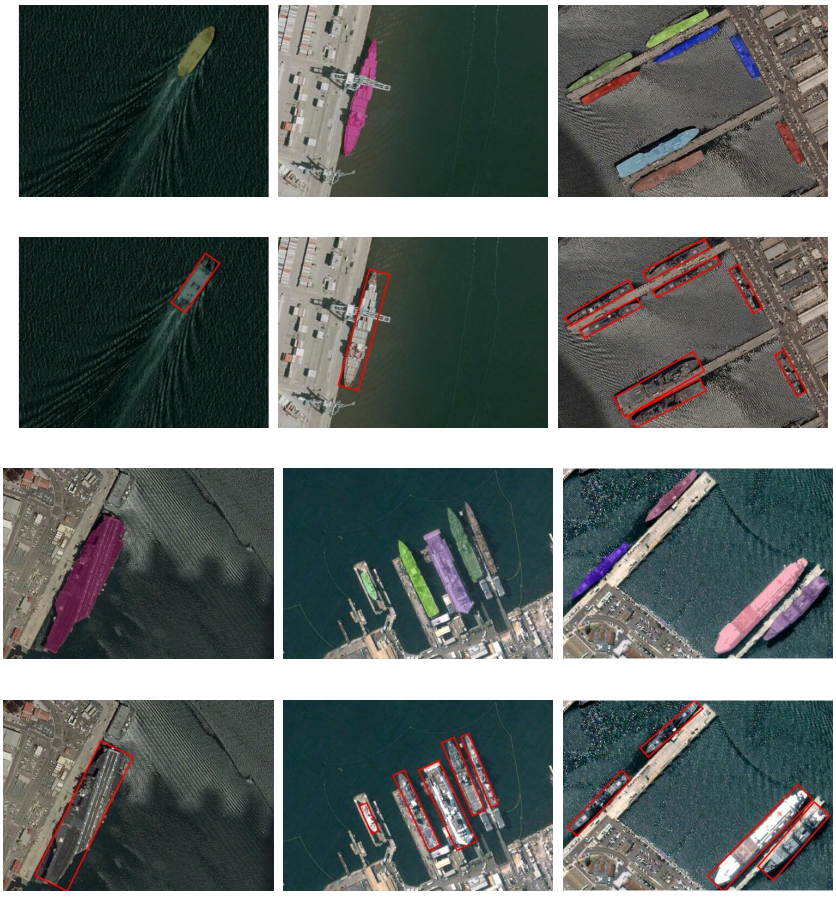


Figure 4.7: Examples of segmentation masks obtained from SAM and of resulting OBBs for selected images in the HRSC2016 dataset

calculate the respective HBBs because the HBBs edge points are determined by the masks' extremes and not by the OBB ones. That is because we want the HBBs to be as tight as possible around the objects in question. Another factor is the existence of classes that usually have significant context similarities with regions outside their HBBs. A predominant such example is the "dock" class, which in most images appears as a long strip of land passing next to one, or between two or more ships. However, its texture is very similar to the part of the harbor which lies behind it and as a result, part of the harbor outside the HBB also gets segmented. The next aspect of the segmentation we examine, is the orientation distribution between ground truth and segmentation generated OBBs. The results are shown in Figure 4.9, from which we can see that the distributions are not identical to the extent that

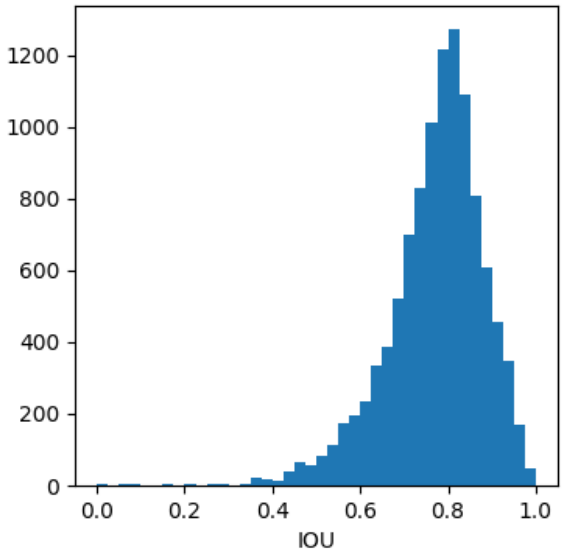
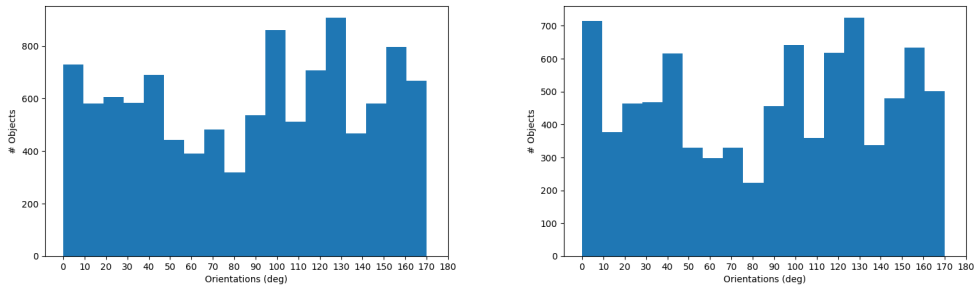


Figure 4.8: Histogram of objects' IoU, between predicted and ground truth HBBs, in the ShipRSImageNet dataset[40]

they were for the HRSC2016 dataset, but still we can observe that the shape of the distributions is very similar, particularly at the orientations at which most objects are directed. For example, we can easily notice the similarity of the peaks at 100, 120 and 130 degrees. As before, we also examine the masks' quality as obtained from SAM with morphological closing, in order to get a better understanding of the model's efficiency on this dataset. In this dataset, there were several good cases, as well as several problematic ones. In Figure 4.10 we present examples of successful segmentations and OBB creations, in a variety of scenarios. As we can see, the model was able to successfully create segmentation masks and predict reliable OBBs in several cases. One such case is when the objects are almost in total darkness (i.e. cases 2,3), which is remarkable since there are very slight variations in terms of pixel intensities, and yet the segmentation part manages to successfully discern them. Another case is when the objects are partially occluded from environmental factors such as clouds (i.e. case 5), which is also indicative of the model's robustness in varying conditions. More importantly however, we notice that the model is also successful in cases where there are simultaneously lots of objects, very close to each other and surrounded by texture rich environments. One such example is shown in case 4, where we see that in an image with a large amount of objects, of varying sizes, and often surrounded



(a) ground truth data (b) generated data

Figure 4.9: Orientation distribution of the objects' OBBs, for the ShipRSImageNet dataset in the: (a) ground truth data, (b) generated annotations

by non uniform areas, for instance ships surrounded by harbor areas, the model not only manages to successfully determine their masks, but also the resulting OBBs are calculated for the majority of the existing objects and are significantly accurate regardless of the respective objects' sizes and orientations. There are however several other examples, where the model was not successful either partially or entirely in its objective to obtain object masks and OBBs. In Figure 4.11 we present examples of cases where the model partially fails, namely it succeeds in obtaining the object masks, but fails to determine OBBs for the respective objects. This may be due to a number of reasons. The most probable reason is that the HBBs of the ground truth are not tight enough around the object, and therefore the predicted HBB which surrounds the object more tightly since its edges touch the mask edges, have a smaller area which may not be enough to exceed the predetermined IoU threshold. Another reason may be the existence of artifacts in the masks, that extend the mask region, and as a result lead to larger HBBs than the ground truth ones. For example, in the first case of Figure 4.11, we notice that the right side dock mask includes a region in its bottom left which does not seem to be a part of the dock. This will lead to the prediction of an HBB that will also incorporate that extra region, but the effect of that will be an HBB of larger area which may considerably decrease the IoU. Cases where both the segmentation and the OBB prediction fail, occur more rarely but still manage to cause damage to the model's performance. Such failed cases for the most part, can be attributed to scenarios where objects are packed next to each other, they belong to the same class and therefore have similar textures, and the image is zoomed in on them. In that case, the segmentation along alternate diagonals will result in masks that cover regions belonging to one or more objects, and will therefore be full of artifacts and

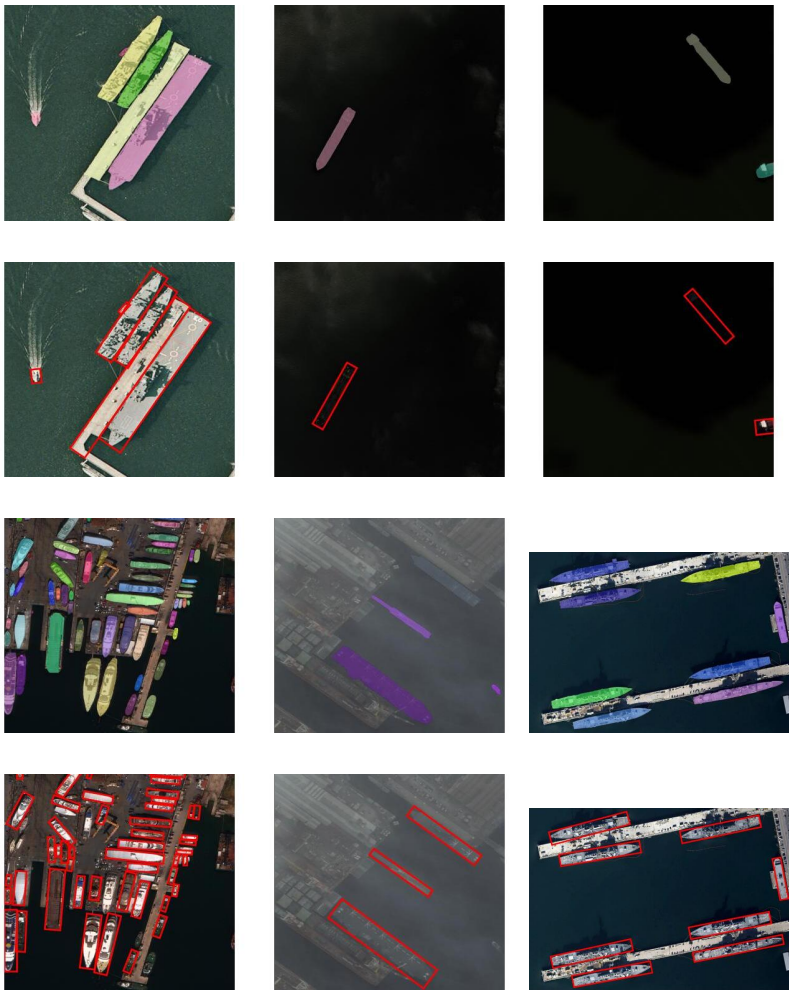


Figure 4.10: Successful examples of segmentation masks obtained from SAM and of resulting OBBs for selected images in the ShipRSImageNet dataset

disconnected regions. In these cases, the most probable scenario is that the predicted HBBs will have very low IoUs and the respective masks will not be used to predict OBBs, but even if that is not the case, the resulting OBBs will be of very low accuracy.

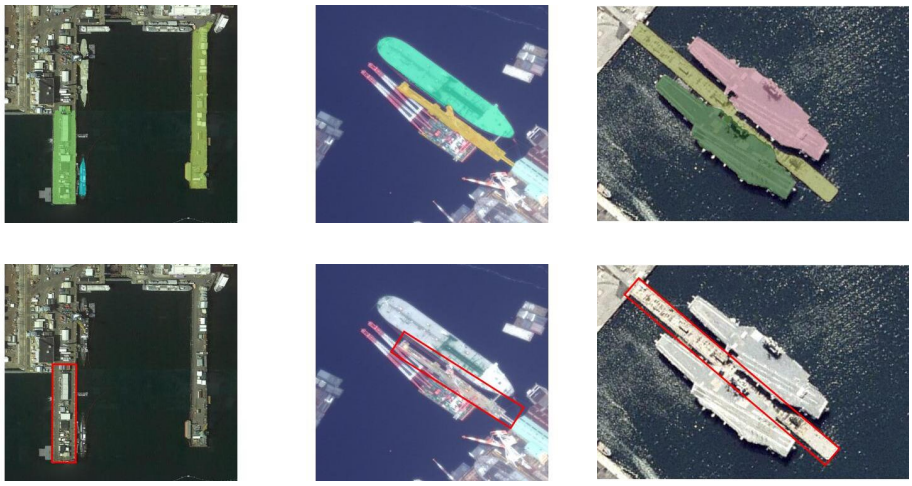


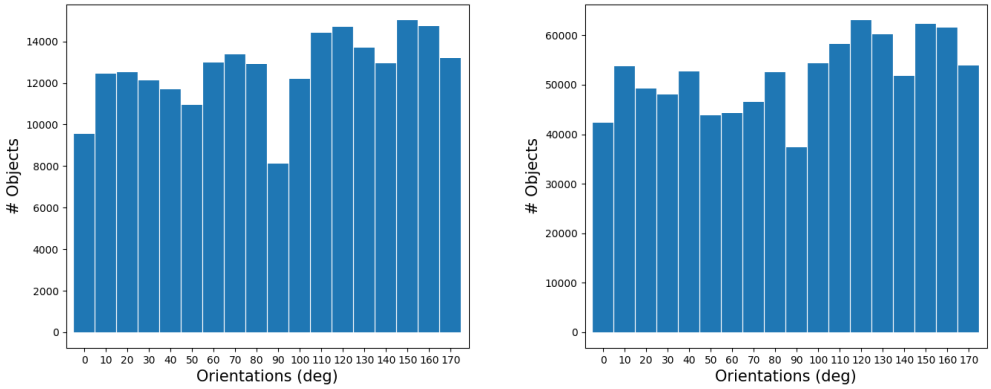
Figure 4.11: Examples of correct segmentations but failed OBB calculations in images of the ShipRSImageNet

4.6 Augmentation and Detection Results

At this point, having already generated new OBBs for all three of the mentioned datasets, we will exploit the information obtained, to create augmented datasets which eliminate the orientation variance of the existing objects. After doing this, we will train several oriented object detectors using the ground truth OBBs, then using the generated OBBs, all from the training sets, and validate the performance on the test set which is the same for both the ground truth and generated data. Subsequently we will train the same detectors using the augmented versions of the datasets as training inputs, and validate them on the test sets in order to examine whether the detection performance of the detectors can be improved.

4.6.1 Results on the DOTA dataset

Firstly, we examine the orientation distributions of the augmented datasets obtained by implementing the SSO and ISO methods. The results are shown in Figure 4.12, from which we can see that the new distributions are far more uniform compared to the ground truth and generated data distributions that are shown in Figure 4.2. In addition, we observe the significant differences between the ISO and SSO datasets, with regard to the number of existing objects in each bin. This means that the ISO method indeed generated more images, and furthermore, the average number of objects in each bin



(a) SSO data (b) ISO data

Figure 4.12: Orientation distribution of the objects' OBBs, for DOTA in the: (a) SSO augmentation dataset and, (b) ISO augmentation dataset.

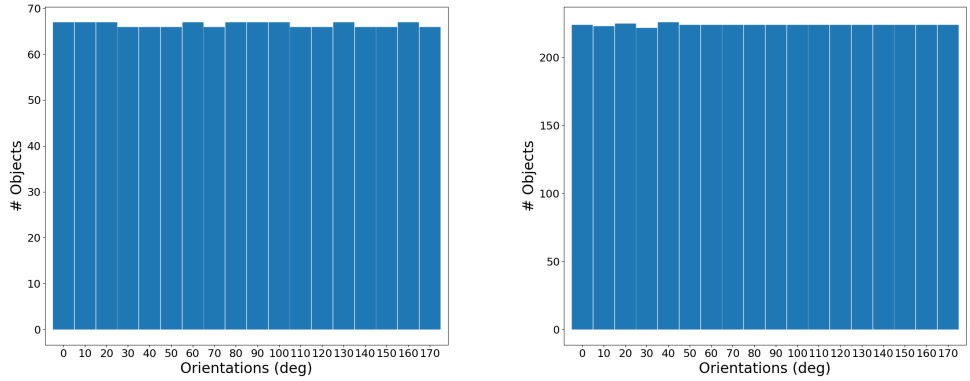
approximates the ISO bound that is twice the number of objects in the most prevalent orientation in the pre-augmented dataset.

However, the distributions are not completely flat, and the reason for this is that almost all images have a significantly large number of objects, in many cases more than a hundred, and they are pointing at arbitrary orientations. This means that when the augmentation starts, and the randomly selected image is to be rotated in a way which minimizes the variance of the new orientation histogram, there may occur orientations at that setting, whose number of objects will exceed the upper bound. If that happens, the iterations will continue until there are no more images that satisfy the upper bound limitation for all orientations, and in that case the algorithm terminates, giving as output the dataset and the orientation histogram which is computed by that point. Nevertheless, the ground truth dataset, the datasets with the generated OBBs, the SSO and ISO datasets are given as input to the rotated object detectors, and their precision performances are presented in Table 4.3. In this table, we present the average precision for all classes in the DOTA dataset and the mean of that, yielded by six oriented object detectors, trained on all four of the training set versions of the dataset. Comparing the results obtained when training the models with the ground truth OBBs with those obtained using the generated OBBs, it can be seen that the generated annotations yield slightly lower precisions and in some cases they even exceed the ground truth in terms of detection performance. This is indicative of the model's generation quality since in this dataset there are many difficult cases like non-elongated objects and objects in significantly

classes	PL	BD	BR	GTF	SV	LV	SH	TC	BC	ST	SBF	RA	HA	SP	HC	CC	mAP
Data/Method	R^3Det [37]																
Ground truth	**87.25**	**68.91**	**32.03**	28.83	55.00	70.43	78.47	**90.00**	39.60	78.49	18.78	**67.53**	53.84	**68.47**	16.52	0.00	53.38
Generated	75.42	48.67	21.18	32.39	49.27	65.34	77.83	89.14	30.21	77.93	21.19	55.63	51.14	60.13	7.58	0.00	47.69
SSO	66.28	30.23	13.65	4.39	51.64	62.11	75.02	83.33	16.90	76.18	17.79	33.33	44.38	59.19	0.00	0.00	39.65
ISO	75.59	55.01	28.67	**55.69**	**55.60**	**77.50**	**78.65**	89.59	**51.16**	**80.98**	**52.52**	53.07	**61.53**	65.54	**18.57**	0.00	**56.23**
Data/Method	ReDet [14]																
Ground truth	**90.04**	**77.37**	**53.26**	**61.36**	**63.43**	81.48	**88.24**	**90.61**	**69.10**	**87.14**	**61.15**	**77.45**	**74.74**	**74.89**	**52.77**	0.00	**68.94**
Generated	86.30	70.43	38.80	60.04	59.75	76.70	80.14	89.72	62.28	86.12	64.09	73.41	65.73	72.36	33.42	**0.53**	63.74
SSO	84.08	61.11	27.71	39.90	57.57	75.95	79.40	90.03	54.85	82.21	48.35	61.52	61.30	68.74	14.88	0.00	56.72
ISO	70.61	54.57	38.79	60.04	61.02	**82.99**	79.72	90.19	64.49	85.30	55.66	67.02	74.03	68.50	25.20	0.00	61.13
Data/Method	Oriented R-CNN [36]																
Ground truth	**88.98**	**76.24**	**49.68**	60.50	**63.63**	**83.01**	**88.77**	**90.66**	**64.89**	**85.56**	51.23	**73.67**	67.13	**74.26**	**54.95**	0.32	**67.09**
Generated	85.27	72.27	43.06	**61.79**	56.80	80.87	80.04	90.27	60.69	78.84	51.51	70.80	63.52	72.46	38.28	**0.67**	62.95
SSO	75.45	62.87	29.10	47.36	58.05	80.45	79.14	89.11	41.53	75.06	34.73	63.24	55.67	66.68	11.69	0.00	54.38
ISO	77.88	60.14	40.98	59.96	62.42	82.53	86.42	90.41	58.05	83.57	**52.69**	64.82	**72.72**	67.84	36.98	0.00	62.34
Data/Method	RoI Transformer [6]																
Ground truth	**89.77**	**75.59**	**47.95**	**66.60**	**63.05**	77.34	**80.57**	**90.69**	**62.56**	**85.54**	**55.29**	**73.60**	73.22	**76.09**	**54.03**	**2.95**	**67.18**
Generated	78.52	67.41	38.23	64.01	56.89	76.72	79.83	90.24	54.09	85.16	53.82	70.20	62.95	71.88	41.13	0.76	61.99
SSO	75.50	62.59	24.62	51.77	57.46	75.07	79.10	89.61	47.96	75.24	50.44	60.56	58.54	71.13	13.35	0.00	55.81
ISO	78.76	53.63	36.22	61.20	60.48	**77.78**	79.70	90.46	55.46	77.97	54.70	63.60	**73.82**	68.71	23.03	0.00	59.72
Data/Method	Rotated RetinaNet [22]																
Ground truth	**87.16**	**73.26**	**22.99**	31.17	50.47	55.94	72.80	**89.91**	46.93	80.05	45.10	**63.98**	52.60	**66.92**	**15.98**	0.00	53.45
Generated	76.28	62.55	17.93	27.85	45.96	51.51	68.93	88.53	39.33	76.76	41.78	57.32	47.84	58.82	1.05	0.00	47.65
SSO	74.56	41.96	2.00	6.06	47.17	41.20	67.06	87.87	21.94	74.11	28.81	32.71	41.03	54.94	0.00	0.00	38.84
ISO	79.59	62.01	21.63	**50.85**	**54.40**	**69.89**	**78.06**	89.79	**51.13**	**80.38**	**53.16**	61.49	**59.00**	63.24	9.87	0.00	**55.28**
Data/Method	S^2A-Net [13]																
Ground truth	**88.80**	**72.55**	**37.23**	40.70	**60.99**	75.74	**80.29**	**90.30**	48.59	**84.73**	43.25	**69.56**	62.18	**69.53**	**43.34**	0.00	**60.49**
Generated	79.43	63.33	29.41	43.68	53.36	74.62	79.03	89.21	45.17	81.72	38.52	68.41	54.09	62.13	33.70	0.00	55.99
SSO	73.92	33.71	13.36	10.93	54.29	72.28	78.60	88.75	22.62	79.37	21.92	39.67	48.61	59.24	0.16	0.00	43.59
ISO	82.49	63.61	31.74	**62.95**	57.73	**80.52**	79.11	89.71	**49.49**	83.56	**51.89**	58.24	**68.21**	64.99	30.21	**0.03**	59.65

Table 4.3: Object detectors AP scores for classes in the DOTA dataset, using ground truth annotations, generated annotations, and augmented data with the proposed methods (SSO, ISO). The classes are, PL: Plane, BD: Baseball Diamond, BR: Bridge, GTF: Ground Track Field, SV: Small Vehicle, LV: Large Vehicle, SH: Ship, TC: Tennis Court, BC: Basketball Court, ST: Storage Tank, SBF: Soccer Ball Field, RA: Roundabout, HA: Harbor, SP: Swimming Pool, HC: Helicopter, and CC: Container Crane

texture-rich environments. However, we also observe that only for two out of the total six detectors used, namely the R^3Det and the Rotated RetinaNet, the mean average precision is maximized by the ISO augmented dataset. For the remaining detectors the maximum precision is obtained by training with the original ground truth OBB training set. However, even in detectors that yield better mAP using the ground truth, there are some classes whose AP is larger when trained using the ISO augmented set. These classes are usually those that have significantly elongated shapes (e.g. Large Vehicle, Soccer Ball Field, Harbor etc.). The reason for this is the one explained previously, namely the objects that do not have clearly elongated shapes are segmented correctly, thus are used for OBB generation, but the created OBBs are not pointing in the direction of these objects and are therefore not accurate. This is not the case for elongated objects like vehicles, sport courts etc. since their shapes lead to OBBs that tightly match the segmentation masks. Despite these flaws however, we can see that the method is effective with respect to the quality of the automated OBB generation, taking into account the



(a) SSO data (b) ISO data

Figure 4.13: Orientation distribution of the objects' OBBs, for HRSC2016 in the: (a) SSO augmentation dataset and, (b) ISO augmentation dataset.

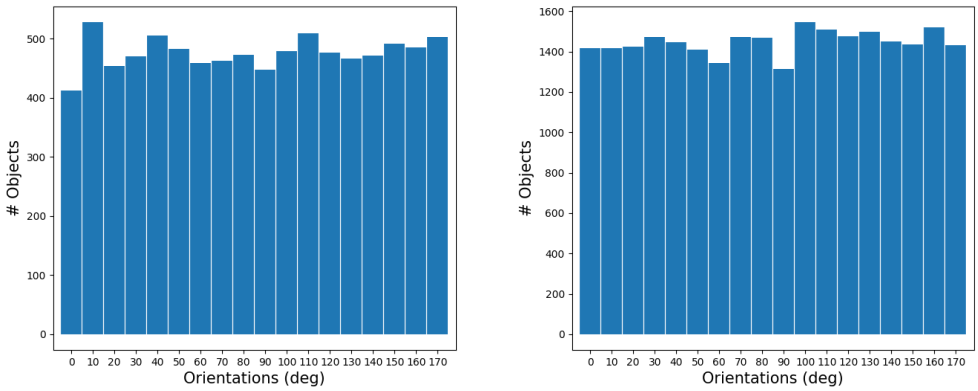
Method	GT	Gen	SSO	ISO
	mAP	mAP	mAP	mAP
R^3Det[37]	83.06	79.49	79.18	**88.33**
ReDet[14]	79.73	78.96	77.78	**87.71**
Oriented-RCNN[36]	**89.93**	81.13	88.91	89.77
RoI-Transformer[6]	87.07	86.92	76.38	**87.86**
Rotated-RetinaNet[22]	62.26	63.01	60.42	**78.43**
S^2A-Net[13]	**89.61**	80.59	86.68	89.23

Table 4.4: mAP scores for object detectors, trained on HRSC2016, using ground truth annotations (GT), generated annotations (Gen), and augmented data with the proposed methods (SSO, ISO).

challenges posed by the complexity of the DOTA dataset.

4.6.2 Results on the HRSC2016 dataset

In Table 4.4 we present the performance of our annotation generation and augmentation methods, for the same six object detectors, using ground truth and generated data for the HRSC2016 dataset. We can see that the object detectors' performance on the test set, when trained using the OBBs generated from our method, is for the most part slightly lower but close to that obtained by training them using the ground truth annotations. This means that the generated OBBs are, at least in their vast majority, of similar quality



(a) SSO data (b) ISO data

Figure 4.14: Orientation distribution of the objects' OBBs, for ShipRSImageNet in the: (a) SSO augmentation dataset and, (b) ISO augmentation dataset.

to the ones from the ground truth. As it was mentioned in paragraph 4.5.2, and referring to Figure 4.6, due to the intense similarity between the ground truth and generated orientation histograms, we expected that the detection performances of object detectors trained on these two datasets, would yield similar results. Considering the promising results of the annotation method, we use the generated annotations for the creation of augmented datasets. Figure 4.13 shows the distribution of the objects' orientations in the SSO and in the ISO augmented dataset. For both augmentation methods used (SSO, ISO), the resulting orientation histograms are completely uniform within the specified orientation quantization, which means that the resulting datasets have nearly zero orientation bias, as can be verified from figures 4.13a and 4.13b. The performance of the same object detectors using the augmented datasets, indicates that the most effective augmentation method is the ISO. SSO yields results that are similar to the ones obtained from ground truth training, but it does not exceed them for any of the object detectors used. Apart from the Oriented-RCNN[36] and S^2A-Net[13] object detectors, ISO yields improved detection performances ranging from 5 to 16% increases in the mAP metric.

4.6.3 Results of the ShipRSImageNet dataset

The detection performances obtained for the ShipRSImageNet dataset, are shown in Table 4.5. Contrary to the HRSC2016 dataset, here the detection

Method	GT	Gen	SSO	ISO
	mAP	mAP	mAP	mAP
R^3Det[37]	42.61	36.88	33.99	**45.73**
ReDet[14]	**59.31**	46.69	41.09	51.78
Oriented-RCNN[36]	57.43	48.22	49.71	**58.93**
RoI-Transformer[6]	**44.91**	35.85	34.18	43.01
Rotated-RetinaNet[22]	37.61	29.56	27.42	**38.69**
S^2A-Net[13]	**55.61**	42.16	36.95	52.60

Table 4.5: mAP scores for object detectors, trained on ShipRSImageNet, using ground truth annotations (GT), generated annotations (Gen) and augmented data with the proposed methods (SSO, ISO).

performances obtained by training with the generated annotations are significantly lower, sometimes by more than 10% mAP. This is a significant decrease compared to the ground truth performances, which can mostly be attributed to the fact that there are many false negatives due to the more extended ground truth HBBs, and due to the less accurate OBB predictions for difficult classes like the docks. Nonetheless, we examine the performances obtained by using the SSO and ISO augmented datasets, whose orientation distributions are shown in Figure 4.14. We notice that the distributions are considerably flat, which means that the bias imposed due to the objects' orientation imbalance is reduced to almost zero. After generating the respective annotations we train the same detectors using the augmented versions of the dataset. We can see that in this dataset, three out of the six detectors used, yielded better performance when trained with the ISO augmented data compared to the one obtained using ground truth data. Specifically, the improvements range from $\sim$0.2% to $\sim$2%, which are not as significant as those in the HRSC2016 dataset, but still exhibit that even in more complex datasets, our method can lead to improved detection performances.

Chapter 5

Conclusions

We proposed a new method to automatically generate OBBs in aerial object detection datasets, and demonstrated its effectiveness by showing that the performance of object detectors when trained on the data with the generated annotations is similar to that obtained when using the ground truth annotations for training. This was validated through the usage of three benchmark aerial object detection datasets, with varying complexities, sizes and object types. The main observation was that the proposed method works significantly better when used on objects with significantly elongated shapes, due to the proper matching between the minimum area rectangle and the generated segmentation mask. An indication of this is that the closest performances between ground truth and generated annotations are achieved using the HRSC2016 dataset which is a ship detection dataset, and for the classes of DOTA that are strongly rectangular (e.g. harbors, vehicles). We also proposed two data augmentation techniques aiming to make the dataset more balanced in terms of orientations which are, object-wise with same size (SSO) and object-wise with increased size (ISO). With respect to the degree at which the uniformity of orientations is achieved, we showed that for the ship detection datasets, the distributions are almost completely flat, and for the DOTA dataset they are slightly less so but still significantly more uniform than the original distributions. Using the SSO and ISO augmented dataset versions, we trained the same oriented object detectors in order to quantitatively evaluate the performance of the augmented versions of the datasets. It was observed that the augmentation using the same number of objects, did not lead to any higher detection performance despite the elimination of the orientation bias. The ISO method however, managed to improve the performance achieved by benchmark object detectors in the test set, with varying margins depending on the dataset complexities and the detectors themselves, a fact that proves the importance of the proposed methods in aerial object

detection tasks.

Bibliography

[1] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[2] Shao-Yi Chien, Shyh-Yih Ma, and Liang-Gee Chen. Efficient moving object segmentation algorithm using background registration technique. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(7):577–586, 2002.

[3] Aaron Walter Avila Cordova, William Condori Quispe, Remy Jorge Cuba Inca, Wilder Nina Choquehuayta, and Eveling Castro Gutierrez. New approaches and tools for ship detection in optical satellite imagery. *Journal of Physics: Conference Series*, 1642(1):012003, sep 2020.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[6] Jian Ding, Nan Xue, Yang Long, Gui-Song Xia, and Qikai Lu. Learning roi transformer for oriented object detection in aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2849–2858, 2019.

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes

Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[9] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[10] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[11] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[12] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[13] Jiaming Han, Jian Ding, Jie Li, and Gui-Song Xia. Align deep features for oriented object detection. *IEEE Transactions on Geoscience and Remote Sensing*, 2021.

[14] Jiaming Han, Jian Ding, Nan Xue, and Gui-Song Xia. Redet: A rotation-equivariant detector for aerial object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2786–2795, 2021.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[16] Yingying Jiang, Xiangyu Zhu, Xiaobing Wang, Shuli Yang, Wei Li, Hua Wang, Pei Fu, and Zhenbo Luo. R2cnn: Rotational region cnn for orientation robust scene text detection. *arXiv preprint arXiv:1706.09579*, 2017.

[17] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, Faisal Shafait, Seiichi Uchida, and Ernest Valveny. Icdar 2015 competition on robust reading. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1156–1160, 2015.

[18] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4015–4026, October 2023.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[20] Darius Lam, Richard Kuzma, Kevin McGee, Samuel Dooley, Michael Laielli, Matthew Klaric, Yaroslav Bulatov, and Brendan McCord. xview: Objects in context in overhead imagery. *arXiv preprint arXiv:1802.07856*, 2018.

[21] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[22] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2017.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[25] Zikun Liu, Hongzhen Wang, Lubin Weng, and Yiping Yang. Ship rotated bounding box space for ship extraction from high-resolution optical satellite images with complex backgrounds. *IEEE Geoscience and Remote Sensing Letters*, 13(8):1074–1078, 2016.

[26] Zikun Liu, Liu Yuan, Lubin Weng, and Yiping Yang. A high resolution optical satellite image dataset for ship recognition and some new baselines. In *International conference on pattern recognition applications and methods*, volume 2, pages 324–331. SciTePress, 2017.

[27] Jianqi Ma, Weiyuan Shao, Hao Ye, Li Wang, Hong Wang, Yingbin Zheng, and Xiangyang Xue. Arbitrary-oriented scene text detection via rotation proposals. *IEEE Transactions on Multimedia*, 20(11):3111–3122, 2018.

[28] Agnieszka Mikołajczyk and Michał Grochowski. Data augmentation for improving deep learning in image classification problem. In *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, pages 117–122, 2018.

[29] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society.

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[32] Zhi Tian, Chunhua Shen, Xinlong Wang, and Hao Chen. Boxinst: High-performance instance segmentation with box annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5443–5452, 2021.

[33] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 2013.

[34] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001.

[35] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Dota: A large-scale dataset for object detection in aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[36] Xingxing Xie, Gong Cheng, Jiabao Wang, Xiwen Yao, and Junwei Han. Oriented r-cnn for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3520–3529, October 2021.

[37] Xue Yang, Junchi Yan, Ziming Feng, and Tao He. R3det: Refined single-stage detector with feature refinement for rotating object. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3163–3171, 2021.

[38] Xue Yang, Gefan Zhang, Wentong Li, Yue Zhou, Xuehui Wang, and Junchi Yan. H2RBox: Horizontal box annotation is all you need for oriented object detection. In *The Eleventh International Conference on Learning Representations*, 2023.

[39] Zenghui Zhang, Weiwei Guo, Shengnan Zhu, and Wenxian Yu. Toward arbitrary-oriented ship detection with rotated region proposal and discrimination networks. *IEEE Geoscience and Remote Sensing Letters*, 15(11):1745–1749, 2018.

[40] Zhengning Zhang, Lin Zhang, Yue Wang, Pengming Feng, and Ran He. Shiprsimagenet: A large-scale fine-grained dataset for ship detection in high-resolution optical remote sensing images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:8458–8472, 2021.

[41] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020.

[42] Yue Zhou, Xue Yang, Gefan Zhang, Jiabao Wang, Yanyi Liu, Liping Hou, Xue Jiang, Xingzhao Liu, Junchi Yan, Chengqi Lyu, Wenwei Zhang, and Kai Chen. Mmrotate: A rotated object detection benchmark using pytorch. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 7331–7334, New York, NY, USA, 2022. Association for Computing Machinery.

[43] Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for object detection. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*, pages 566–583. Springer, 2020.