UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Computer Science

End-to-end Trainable Ship Detection in SAR Images with Single Level Features

Markus Tiller

INF-3981 Master's Thesis in Computer Science

Abstract

Kongsberg Satellite Services (KSAT) use machine learning and manual analysis done by synthetic aperture radar (SAR) specialists on SAR images in real time to provide a ship detection service.

Since heuristic post processing in object detection limit the models ability to distinguish ships close to each other, we investigate challenges related to employing an end-to-end trainable ship detection model. Since access to ground truth annotations in SAR images is limited, size and rotation labels are not available for all ships, and rotation labels are inaccurate. Since KSATs internal datasets are collected as part of a time critical operational service, position labels are not exact. Since existing evaluation metrics for object detection are too strict, they do not reflect user needs for this service.

To tolerate missing size and rotation annotations, we base loss label assignment on the distance between objects instead of their IoU, and replace DIoU bounding box loss with a novel size regression loss named *Size IoU (SIoU)* combined with smooth_{L_1} position loss. To tolerate inaccurate rotation labels, we propose angular direction vector (ADV) regression. To tolerate inaccurate position labels, the loss label assignment makes all predictions responsible for large overlapping regions instead of small disjoint regions. To compare models performance according to user needs, we propose an evaluation metric named Distance-AP (dAP), which is based on mAP, but replaces the IoU overlap threshold with an object center point distance threshold. To reduce duplicate ship predictions, we propose multi layer attention.

Using the LS-SSDD SAR ship dataset, we find that replacing IoU based label assignment with position based label assignment increases dAP from 79% to 86%, and that replacing DIoU with SIoU decreases dAP by only 1%. Using a rotation regression benchmark where datasets have different amounts of rotation label noise, we find that ADV outperforms CSL in terms of mean predicted inaccuracy at all noise levels, and median predicted inaccuracy at high noise levels. Using an object detection benchmark where the datasets have varying amount of position label inaccuracy, we find that the proposed loss label assignment tolerates large amounts of noise without reduced performance. Using KSATs dataset of Sentinel 1 images, we measure 83% dAP.

The proposed mechanisms allow effective training of a ship detection model, despite the missing size and rotation annotations, inaccurate position annotations, and inaccurate rotation annotations. We believe this is useful for KSATs ship detection service, as it can better distinguish nearby ships. However, more work is required to compare its performance with their existing solution. Source code is available at https://github.com/matill/Ship-detection

D

Acknowledgments

I would like to thank my supervisors Professor Lars Ailo Bongo, Professor Robert Jenssen, and Torgeir Brenn for their guidance in this project.

Markus Olav Tiller,
Tromsø, May 2022.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Problem definition

Synthetic aperture radar (SAR) satellite images provide real time information about the position, direction, and size of ships at sea regardless of weather conditions, which is useful for maritime surveillance. Although, ships are required by law to be equipped with Automatic Identification System (AIS) trackers, they will sometimes deactivate AIS trackers to hide illegal or military activity. These ships can still be detected in SAR images. Ship recognition service providers, such as Kongsberg Satellite Services (KSAT), use live streams of SAR images from many satellites in combination with machine learning and manual image analysis to provide real time ship detection.

Figure 1.1 shows a SAR image. Compared to optical images, SAR images have the advantage of not being obscured by clouds, and therefore work in all weather conditions. However, SAR images are more difficult to interpret even for SAR specialists. When inferring the physical size and orientation of ships, the average error is high. It is also difficult to distinguish ships from other radar reflectors at sea.

KSATs current ship detection model is based on semantic segmentation. This works well, but we investigate how we can employ a newer object detection model to overcome two main limitations. First, their current model is a composition of several processing steps and machine learning models that need to be trained in different stages, and the model is therefore difficult to optimize

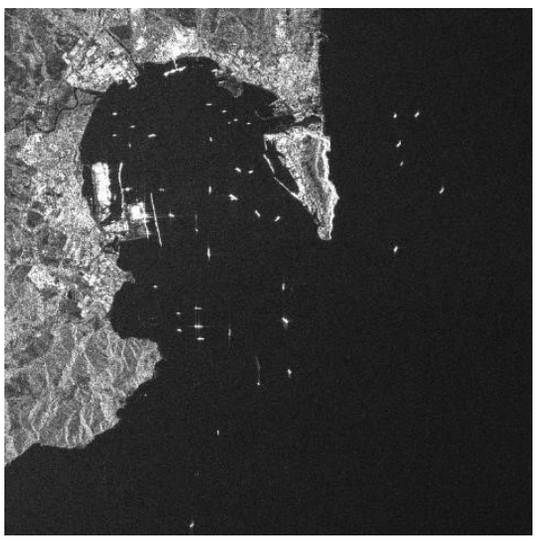

Figure 1.1: SAR image illustrating vessels observed between Gibraltar and Algesiras on September 2017, Copernicus Sentinel Data. From [1]

and deploy. Second, limitations in their model make it difficult to distinguish nearby ships, although the ships may be easily distinguishable in the images. However, to employ an object detection model for ship detection in SAR images, we first need to address four sub problems. (1) The physical size and rotation of ships in SAR images is not available for all ships in any available datasets, as it is difficult for SAR specialists to infer accurate estimates, and size and rotation annotations therefore rely on external data sources. Existing oriented object detection models require size and orientation of all objects in the training data to be known, and can therefore not be applied. (2) If available, estimates of ship direction that are found in SAR ship recognition data sets can be highly inaccurate, especially for low resolution imaging modes. State of the art rotation regression models are sensitive to inaccurate rotation labels. (3) Exact locations of detected ships are not important to end-users, so training data collected from such real time manual SAR ship recognition services, may contain inaccurate ship positions. Many object detection models are sensitive to this inaccuracy. (4) Existing evaluation metrics for object detection systems reflect strict bounding box location requirements. However, end users of ship detection systems have more relaxed location accuracy needs. By using the wrong performance metric, system properties such as exact bounding box overlap which is unimportant to end users may receive large development efforts, while more important properties such as the actual detection rate suffer.

Summarized, we need (1) an oriented object detection model that allows some training samples to have unknown size and orientation, (2) a rotation regression model that tolerates inaccurate rotation labels in training data, (3)

an object detection model that is not sensitive to inaccurate position labels in training data, and (4) a new performance metric reflecting the relaxed location requirements of ship detection systems.

1.2 Problems with previous approaches

Existing object detection systems require all objects in the training data to have known size. Two main mechanisms cause this limitation. First, to compute loss, they use a label assignment method that compares the position, size, and orientation of true bounding boxes of objects against either predicted bounding boxes [24, 3] or anchor boxes [2, 16, 17, 19, 12, 4]. If the true size of an object is not known, the label assignment method cannot operate, and they cannot assign ground truth labels to the different predictions. Second, some detectors apply Distance IoU (DIoU) loss [30] for joint optimization of bounding box size and position. If the size labels for a given ship are missing while position labels are available, we still want to optimize the predicted position of that ship. However, when using DIoU loss, a lack of size labels makes it impossible to optimize predicted positions although position labels are available.

With inaccurate orientation labels, we need a rotation regression model that is robust to inaccurate labels. Rotation regression is nontrivial, because of the border discontinuity problem. Minimum and maximum rotations (eg. 0° and 360°) are logically adjacent, and should therefore be represented with similar network outputs. However, direct regression is insufficient for modelling rotations, since it represents minimum and maximum rotations with dissimilar network outputs. Circular smooth labels (CSL) and grey coded labels (GCL) [26] overcome the border discontinuity problem, and are the current state of the art in rotation regression. However, they are sensitive to inaccurate rotation labels.

The object detection model must tolerate inaccuracies in position labels. Most existing object detectors enforce a strict policy where different predictions are responsible for different (disjoint) regions in images [16, 17, 2, 12, 4], and are therefore sensitive to inaccurately labeled positions. For example, if the position label for a given object is inaccurate, as in Figure 1.2, then the image region that contains the given object according to the position label is random. This causes randomness in which predictions are assigned negative or positive classification labels, which harms the models ability to distinguish objects from background. So, inaccurately labeled positions does not only harm the models ability to accurately determine the exact *position* of objects, but more importantly harms the models ability to distinguish objects from background. DETR [3] employs a loss label assignment method that is less dependent on

changes in position labels for large objects, and may enforce consistent label assignments in face of position label inaccuracies. However, DETR suffers from long training time. DeFCN [24] employs a similar label assignment method to DETR with shorter training time. DeFCN may be robust to inaccurate position labels for large objects, but as some parts of the DeFCN loss label assignment are not described in [24] we do not know.

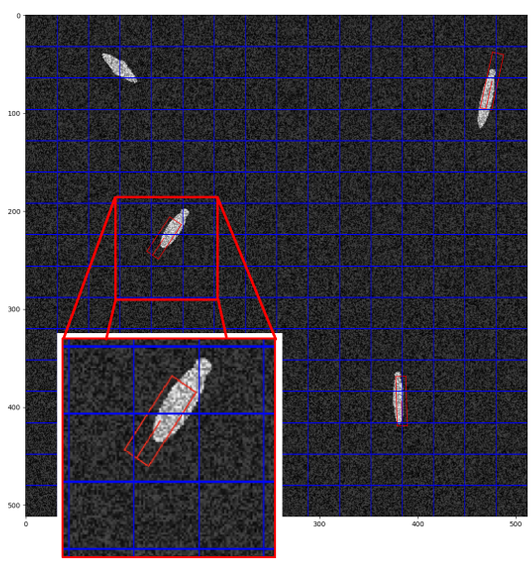

Figure 1.2: A randomly generated image imitating ships in SAR images. Labeled center positions are inaccurate. The image is divided in 32×32 pixel regions, to resemble the stride of the output grid. The labeled center point of the highlighted ship is contained in a different region than the true center point. Some label assignment methods are strict about which such regions contains the center point of a given object, and assign labels inconsistently depending on how accurate position labels are.

Existing evaluation metrics for object detection are based more strict than necessary. Currently, the most used performance metrics for object detection research are based on average precision (AP) [13]. AP requires a predicted bounding box to have high IoU overlap with a true bounding box to count as a true positive. However, this requirement is not consistent with end user needs for ship detection services.

1.3 Proposed solution

Since the training data contains some ships where size and orientation is unknown, we propose two new mechanisms for oriented object detection. First, we propose a loss label assignment method based entirely on the predicted

and true positions of ships, since the true size and rotation of ships size is not always known. This way, we can perform loss label assignment even if size or rotation is unknown. This is based on a modified version of the DeFCN loss label assignment [24]. Second, we use a loss function for position and size of predictions that is simply a weighted sum of a position loss and size loss. This way, if a size label is missing, we ignore size loss and only use position loss, which is not possible with state of the art DIoU loss [30]. For this purpose, we propose a loss function for object length and width called Size IoU (SIoU) loss. SIoU is scale invariant, similarly to DIoU, meaning it does not induce disproportional high loss for large objects compared to small objects.

We propose angular direction vector (ADV) regression as a new rotation regression method. It is robust against inaccurately labeled rotations, which is important for low resolution SAR images where ship rotation labels can be very inaccurate. Instead of predicting rotation angles directly, ADV predicts direction vectors corresponding to said angles. Minimum and maximum angles (0° and 360°) are logically adjacent, and ADV can represent them using the adjacent direction vector.

We modify the loss label assignment in the DeFCN object detection model [24] to ensure that the different predictions are responsible for large overlapping image regions instead of small disjoint regions. This reduces label assignment inconsistency when position labels are inaccurate. So, the DeFCN label assignment is modified for two reasons: To tolerate inaccurate position labels, and to tolerate missing size annotations (the latter was mentioned two paragraphs above).

We propose a new evaluation metric named distance average precision (dAP). dAP is defined similarly to average precision (mAP), with the main difference being the requirement for counting a predicted object as a true positive instead of a false positive. dAP counts a predicted object as a true positive if its center point is less than a fixed distance from the true object, whereas AP requires predicted objects to overlap true objects with IoU higher than a predefined threshold. Furthermore, whereas AP is averaged over ten different IoU thresholds $(0.5, 0.55, ..., 0.95)$, dAP uses only a single distance threshold. This way, dAP more accurately reflects end user requirements. As default, we use a 50 pixel threshold for dAP, corresponding to 500 meters in the Sentinel 1 SAR images we used.

In addition to the solutions to the four main sub problems, we propose an attention module with multiple layers. This module increases an object detection models ability to avoid making duplicate predictions.

1.4 Summary of results

We use SIoU loss and position based loss label assignment to enable training an oriented object detection model with a data set where size and rotation are unknown for some samples. We compare these to state of the art methods, using the Large scale - SAR ship detection dataset (LS-SSDD) [28]. (1) We compare position based label assignment against IoU based label assignment and position plus SIoU based label assignment, and find that position based label assignment outperforms the other configurations. (2) We compare SIoU against DIoU loss for bounding box regression, and find that SIoU loss results in only 1% lower dAP than DIoU loss. We also find that SIoU results in significantly lower AP50 due to less accurate positioning of detections, which is not important for ship detection.

We compare ADV rotation regression against logistic regression and state of the art CSL and GCL [26] models by training a neural network that processes an image and predicts the rotation of the image. We created a dataset using natural images where we rotate the center portion by a random amount, and the label is the amount of rotation. We inject noise in the training data labels, to imitate random noise in SAR ship rotation labels. The different models are evaluated using different distributions of random noise. The performance metrics used are the mean and median deviation between model predictions and true rotations in the test set (in degrees). We found that ADV was superior in mean deviation across all label inaccuracy levels, and in terms of median deviation at high label inaccuracy levels, while CSL was superior in terms of median deviation at low label inaccuracy levels.

We confirm the effect of inaccurate position labels on models with overlapping and non-overlapping label assignment methods by generating a data set of images that imitate SAR images with rotated ships, where there is random noise in position labels, and some ships have unknown rotation and or size. Figure 1.2 shows a sample image. Each model is trained 14 times at 14 different (uniform) noise distributions. By plotting the models dAP as a function of the amount of label noise, we show that label assignment with overlapping regions is barely affected by label noise and is overall superior.

We use KSATs large dataset of Sentinel 1 images, where position annotations have some inaccuracy, and perform an ablation experiment to compare the effect of two mechanisms using a 2D grid search in terms of dAP. (1) We compare the effect of (position based) label assignment with overlapping against non overlapping regions, and see an increase in dAP, consistent with the results in the previous experiment. (2) We compare training a model that only detects ships against a model that both detects *and* predicts the physical size of ships in one network pass, and find that a multi-task model decreases dAP slightly.

The best performing model achieved 83% dAP.

The multi layer attention module is used in several experiments, and achieves good results. However, we have not performed ablation experiments for the multi layer attention module that confirm its effectiveness, compared to no attention module, or alternative attention module architectures.

Our main contributions are as follows. (1) We propose a rotation regression method named ADV that is robust against noise in rotation labels. (2) We propose SIoU loss and position based loss label assignment, allowing training data with missing size annotations. (3) We show that a (position based) loss label assignment method with large overlapping regions of responsibility increases detection rate when using training set with inaccurate position labels. (4) We propose a new ship detection performance metric name dAP, which more accurately represent user needs than AP in ship detection applications, and show that the model architecture can be made less complex by using appropriate performance metrics.

The proposed model can be trained with a dataset where size labels are missing, and position labels and rotation labels are inaccurate. Combined, these mechanisms enable KSAT to train a ship detection detection model that requires zero post processing, and handles both ship detection and ship description tasks using a single neural network. This can have an improved ability to distinguish nearby ships.

2 Background

In this section provides background information relevant to this thesis. We use SAR images, and although comprehensive knowledge about SAR data is not necessary, we describe some fundamental concepts. Throughout the thesis we discuss architectures, loss functions and other terms that are widely used in object detection, and in this section we describe a few object detection models that represent different types of designs. Then, we describe state of the art methods in rotation regression, as we contribute to this research and compare our method to state of the art. Finally, we discuss related work on ship detection in SAR images, and how this focus on different challenges than us.

We assume the reader is familiar with core concepts in deep learning as listed. (1) Basic concepts such as loss functions, backpropagation, and stochastic gradient descent and its variants such as Adam and Adagrad [9, 5]. (2) Convolutional neural networks (CNN) and deep CNNs such as VGG and ResNet [21, 7], and familiarity with the terms such as *fully convolutional network*. (3) More advanced models such as transformer models [23] which were originally proposed for natural language processing, are useful prerequisites as they are used by some object detection methods, but not necessary.

2.1 Synthetic aperture radar (SAR)

In this project we use synthetic aperture radar (SAR) images. SAR is an imaging radar widely used in satellite based remote sensing. Recall the SAR image in Figure 1.1. Conventional imaging radars transmit a pulse of radar waves, and record the echo to construct an image. Increasing the antenna aperture (size) results in higher resolution images. Using c-band ($5cm$ wavelength), which is used in the Sentinel 1 images in this project, to achieve a spatial resolution of $10m$, a $4250m$ antennas is required [15]. However, instead of using large antennas, SAR satellites achieve high resolution by recording the echo of several radio wave pulses from different positions, which synthetically increases the spatial resolution of images. For more information about SAR data, we refer to [15].

Polarization of electromagnetic waves affects absorption and reflection in contact with different materials. Depending on how the waves are reflected, the received polarization relative to the transmitted polarization may change. In SAR satellites, transmitters ensure either a horizontal or vertical polarization. Receivers typically filter the input to either horizontal or vertical polarization. Some satellites can receive with horizontal and vertical filters in parallel, resulting in SAR images with two channels. Both the sending and receiving polarization affect a SAR image, and there are four possible combinations of sending and receiving polarization. Image polarization is denoted using either vv, vh, hv or hh, where the first and second letters correspond to the transmitted and received polarization respectively. SAR images may have multiple channels, where each channel corresponds to one polarization. Eg. a single image can have one vv channel and one vh channel, which we denote vv+vh images.

Reflected radar waves may change polarization depending on the material and structure of different objects. Therefore, different image polarizations are effective for detecting different types of objects. For ship detection, vv polarization is superior, while for oil spill detection, a vh polarization is superior. In this project, we use Sentinel 1 images with vv+vh polarization.

In SAR imagery, there is a trade off between the spatial resolution of images and the amount of sea area covered. Larger sea area is often prioritized over high resolution as it generally provides more information. However, the low resolution makes the analysis more difficult, as there is less information available about each entity in the image. Still, for certain tasks, there may be visual patterns in low resolution images that humans are unable to see, meaning machine learning algorithms can potentially surpass human level performance given enough data. Ship classification, size regression, and rotation regression on ships may be examples of this.

2.2 Horizontal object detection

This section aims to provide an understanding of central concepts and terms used in object detection with horizontal bounding boxes, which represents most object detection research. We describe some specific detectors that represent different design choices and components used in many object detection models. We use concepts from each of the models described.

Some fundamental concepts to pay attention to are: (1) How is the output of an object detector interpreted? We use an interpretation to YOLOF. (2) What is loss label assignment, and which loss label assignment methods do different detector models use? We propose a new loss label assignment method. (3) What is the difference between end-to-end trainable detectors, and those that are not? We predict that object detection research will tend towards end-to-end trainability in the near future due to certain benefits, and we emphasise this in our work. (4) What are feature pyramids? This is used in virtually all modern object detection models, but we omit feature pyramids due to the relaxed dAP performance metric. (5) What are the state of the art bounding box regression loss functions. Furthermore, what are their advantages, and why are they incompatible with our problem.

2.2.1 You only look one-level feature (YOLOF)

Our CNN architecture is based on the YOLOF model. YOLOF [4] is a simple object detection model, that is designed for detection in natural images such as traffic images. In this description, we assume a YOLOF model with 9 anchors and 3 classes to simplify visualization, although a larger number of anchors and classes is normal. The architecture is described by Figures 2.1, 2.2 and 2.3. YOLOF is a fully convolutional neural network, using a ResNet backbone [7], and a *dilated encoder* [4] to extract high level features. The dilated encoder a specific structure of convolutional operations, and understanding the exact details is not important for this project. The high level feature map is processed by a final convolution. This architecture is fully convolutional with an output stride of 32, meaning an input image with size $h \times w$ results in an output grid of size $h/32 \times w/32$, and each cell in the output grid corresponds to a unique 32×32 pixel region in the input image.

The interpretation of the YOLOF network's output is described in Figure 2.2. The output grid consists of 9 subgrids, where each subgrid has 8 (5+3) channels, where 5 channels are for (p, x, y, h, w), and 3 channels for softmax classification into 3 classes. Each spatial dimension in each subgrid corresponds to a single prediction, and with a 512×512 image, there are $32 \times 32 \times 9 = 9216$ predictions.

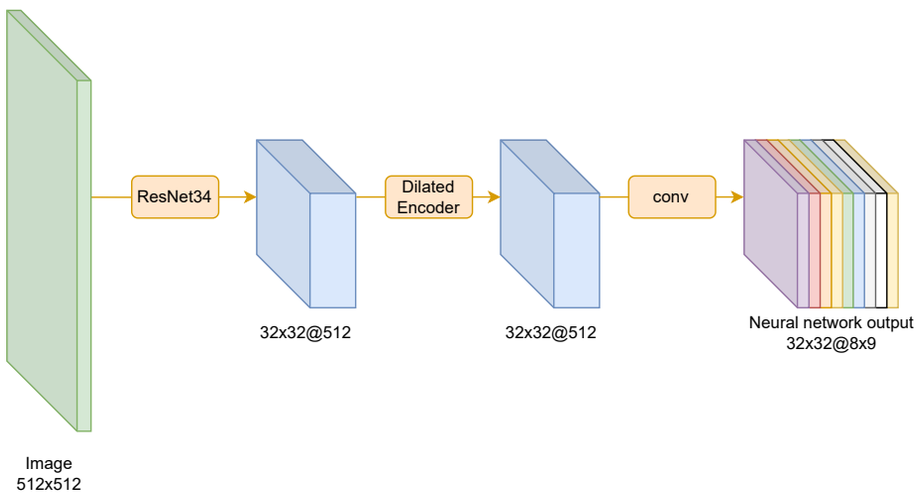

Figure 2.1: The YOLOF CNN architecture. YOLOF is fully convolution, and the CNN in itself is a simple feed forward network. The output has a stride / down sample factor of 32. Interpretation of the output is illustrated in Figure 2.2

.

A single prediction, among the 9216 predictions, is interpreted as follows. $p \in (0, 1)$ has sigmoid activation, and classifies the prediction as either positive or negative (background / nothing). The $(x, y) \in (0, 1)^2$ have sigmoid activation, and represent the (potential) object's center position as an offset from the top-left corner of the unique 32×32 image region corresponding to the prediction. To be clear, $(x, y) = (0, 0)$ means the object's *center* position is at the top-left corner of the cell's corresponding image region, and $(x, y) = (1, 1)$ means the object's center position is at the bottom-right corner of the image region. The (c_1, c_2, c_3) tuple represent a simple softmax classification of the (potential) object. The (h, w) tuple represents size, and is explained below.

As mentioned, the neural network output is a stack of 9 grids. Hence, for each spatial position in the output grid, there are 9 (potential) object predictions. The 9 predictions are responsible for the exact same 32×32 pixel region in the image, and are also responsible for objects with different height and width combinations (h_a, w_a). This height and width combination is typically referred to as the *anchor box*. Typically, 3 aspect ratios (h_a/w_a) and 3 sizes $(h_a \times w_a)$ are chosen, giving $3 \times 3 = 9$ combinations of aspect ratio and size. The CNN therefore computes a stack of 9 grids. To interpret the (h, w) value of a single detection, the final predicted size of the object in pixels (h', w') is computed as a *multiplicative offset* from the anchor box size. Specifically, $h' = h_a \cdot h$ and $w' = w_a \cdot w$, where w and h have an exponential activation.

Figure 2.3 displays how the YOLOF output grid is decoded during infer-

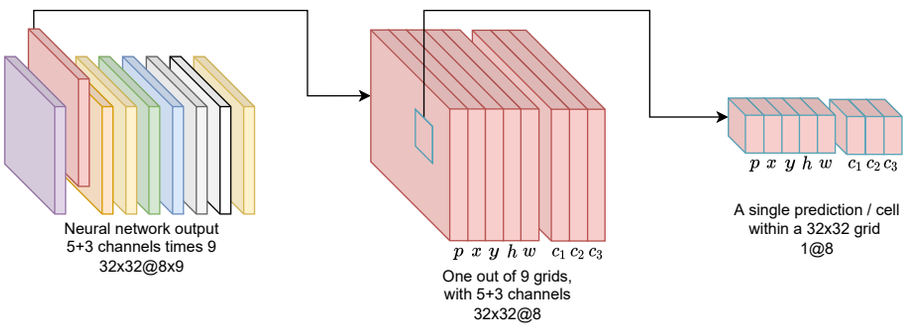

Figure 2.2: Structure of YOLOF CNN output. The output grid consists of 9 sub grids. Each subgrid has $5 + 3 = 8$ channels (in the special case of 3 classes). Each spatial position in a (sub) grid is a vector that describes a (potential) detection, determined by the positivity classification score p.

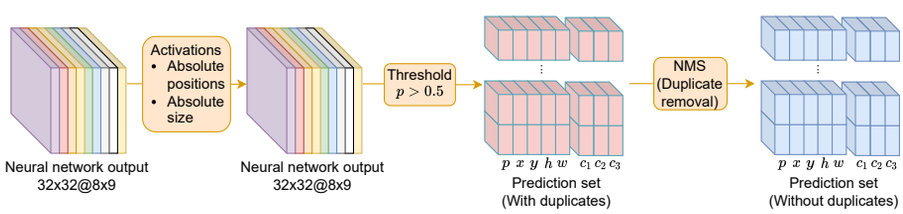

Figure 2.3: Interpretation of the YOLOF CNN output. First, relative position offsets are converted to absolute positions, and (w, h) predictions are multiplied with anchor box sizes to get $(h', w') = (h \cdot h_a, w \cdot w_a)$. Second, detection vectors with positivity score p above a certain threshold (eg. $p > 0.5$) are extracted from the grid, into an $n \times (5+3)$ matrix. Finally, the detection set is post processed with non-max suppression (NMS), to remove duplicate detections with high IoU overlap.

ence.

A 32×32 output grid with 9 (potential) detections in each spatial position has in total $32 \times 32 \times 9 = 9216$ (potential) detection vectors, and an image may contain n objects (eg. $n = 10$). During training, we need to assign labels to each of the 9216 (potential) prediction vectors. Some are assigned positive classification labels and regression target labels, while others are assigned negative classification labels and no regression target labels. This is referred to as loss label assignment, and is a fundamental concept in object detection.

After a loss label assignment has been determined, each (potential) prediction is assigned to either be positive or negative. A prediction that is positively assigned receives a positive classification label ($p = 1$), and receives labels for position, size and class to match the object that it is assigned to. A prediction

that is negatively assigned (background) receives a negative classification label ($p = 0$), and the predicted position, size and class induces no loss since the prediction is negative.

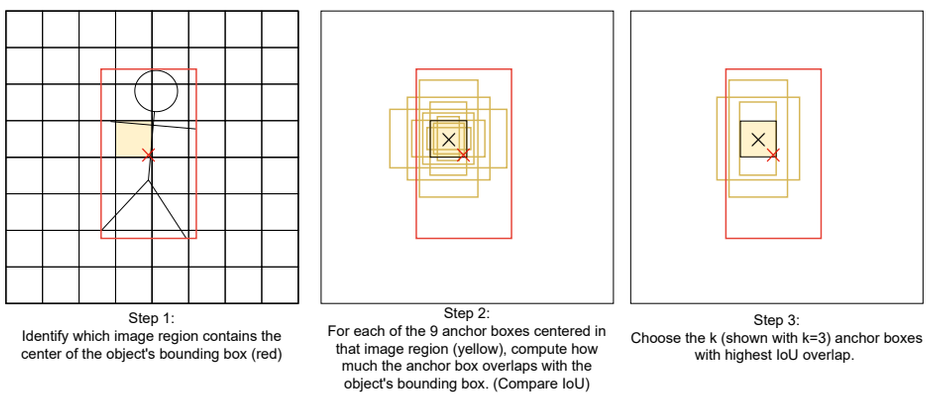

Figure 2.4: YOLOF's loss label assignment method. This procedure is repeated for each object in the image.

YOLOF uses the following loss label assignment method. The procedure illustrated in Figure 2.4 is repeated for each object in an image. First, they find which 32×32 pixel region in the image that contains the center position of the object's bounding box. Then, for each of the 9 anchor boxes centered in that region, they compute IoU overlap of the anchor box and the true object box. Finally, choose the k anchor boxes with highest overlap. The k prediction vectors corresponding to the k anchor boxes are assigned to predict the given object.

There are some key things to note about YOLOF's loss label assignment method. (1) It assigns $k > 1$ prediction vectors to each object in the image. As a consequence, YOLOF predicts several duplicate predictions for each object, making duplicate removal step (NMS) an essential post-processing step during inference. (2) One prediction may sometimes be assigned to two or more objects. (3) The label assignment does not depend on the neural network output, but is entirely based on the shapes of the predefined anchor boxes and the ground truth bounding boxes.

2.2.2 Detection Transformer (DETR)

The Detection Transformer (DETR) [3] represents an important paradigm shift in object detection. DETR emphasises *end-to-end training*, in the sense that the output of the neural network is not post-processed (eg. no duplicate removal with non-max suppression). The following two paragraphs motivate this choice, and the rest of this section describes the DETR model. Our loss label

assignment is inspired by the one proposed by DETR, as it enables end-to-end training.

To be clear, there are some conflicting definitions of *end-to-end training* in deep learning. Some will consider a model end-to-end trainable if all model parameters can be optimized simultaneously (with one loss function), while DETR introduced the additional requirement that the loss function should operate on the *final* output of the model, which implies no post-processing of the network output during inference.

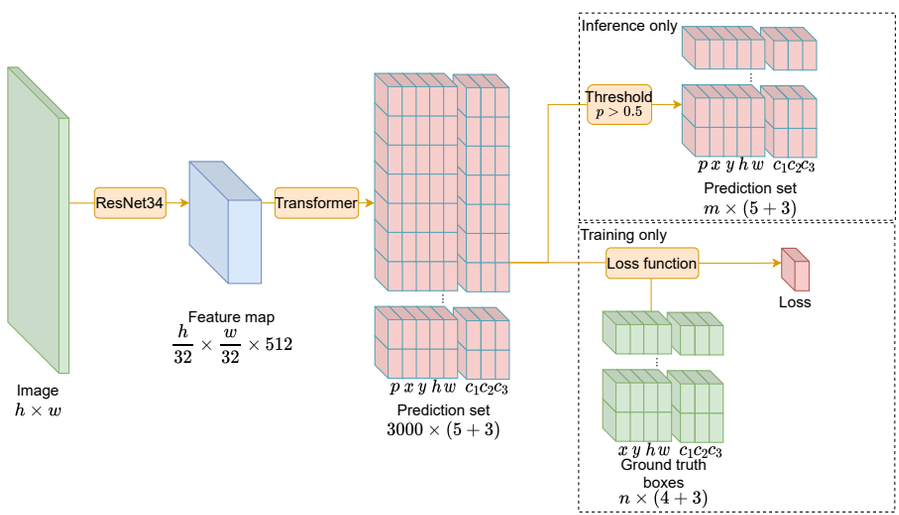

Figure 2.5: The DETR architecture. A CNN backbone (ResNet) extracts features from an image with any shape. Then, a transformer computes a 3000×8 matrix (output shape does not depend on input shape). During inference, the only additional processing is a trivial threshold operation (so no duplicate removal with NMS). During training, the loss function operates on the final output.

Post processing algorithms will in general limit the performance of detectors. Non-max suppression (NMS) removes overlapping predicted bounding boxes if their IoU is above a certain fixed threshold (eg. $IoU > 0.6$). The only input NMS sees is a set of bounding boxes, which is not always enough information to decide if two detections are duplicates or two different objects. In particular, in applications with crowded scenes such as pedestrian detection, bounding box overlap is not a good indicator of duplicate predictions, and NMS brings a severe negative effect on model accuracy.

Figure 2.5 illustrates the DETR architecture [3]. A ResNet backbone [7] (or any other CNN backbone) extracts a feature map from an image. Then, given a feature map of any shape, a transformer [23] computes a fixed size matrix containing 3000 (potential) detection vectors. The number of predictions

(3000) is assumed to be a lot larger than the typical number of objects in an image. The network predicts absolute positions, differently from YOLOF which predicts positions as an offset within small 32×32 regions. DETR predicts absolute sizes, instead of multiplicative factors that are multiplied by predefined anchor box sizes.

In DETR, none of the 3000 output predictions are dedicated to any image regions, aspect ratios, object sizes, or classes. Instead, the loss label assignment is based entirely on which prediction vectors best fit which ground truth (target) boxes. The loss label assignment is done as follows. First, do a forward pass. Then, compute a $3000 \times n$ assignment cost matrix, which quantifies how well each prediction fits each of the n targets, based on IoU overlap and the distance between the two boxes. Finally, using the $3000 \times n$ matrix, they determine a loss label assignment such that exactly one prediction is assigned to each target, and each prediction is assigned to either one or zero targets. The next paragraph elaborates this.

Again, assume the number of objects in an image is always less than the number of output predictions ($n < 3000$). The assignment cost matrix is denoted $\mathcal{L}_{match} \in \mathbb{R}^{3000 \times n}$, where large entries imply bad matchings (little similarity) and small entries imply good matchings (high similarity). $\mathcal{L}_{match}$ corresponds to the colored lines in Figure 2.6. The set $\mathcal{A}$ (defined in Equation 2.1) represents all possible label assignments. $\underline{a}' \in \mathcal{A}$ is a label assignment (index) vector, such that the target with index i is assigned to the prediction with index a_i. $\underline{a}'$ is a non-repeating index vector, meaning a single prediction can only be assigned to one target. The final loss label assignment $\underline{a}$ optimizes the linear sum assignment in Equation 2.2. The number of allowed permutations is large ($|\mathcal{A}| = \frac{3000!}{(3000-n)!}$), but the optimal permutation ($\underline{a} \in \mathcal{A}$) can be determined efficiently using the Hungarian algorithm [10].

$$\mathcal{A} = \{\underline{a}' \in \{1, 2, ..., 3000\}^n : i \neq j \implies a'_i \neq a'_j\} \tag{2.1}$$

$$\underline{a} = \arg\min_{\underline{a}' \in \mathcal{A}} \sum_{i=1}^{n} \mathcal{L}_{match}(a'_i, i) \tag{2.2}$$

2.2.3 DeFCN

Want et al. designed a model for *End-to-end object detection with fully convolutional network*, named DeFCN. They found how the loss label assignment in DETR could be modified to fit into a fully convolutional framework, which is similar to ours. The DeFCN architecture is illustrated in Figure 2.7. First, a CNN

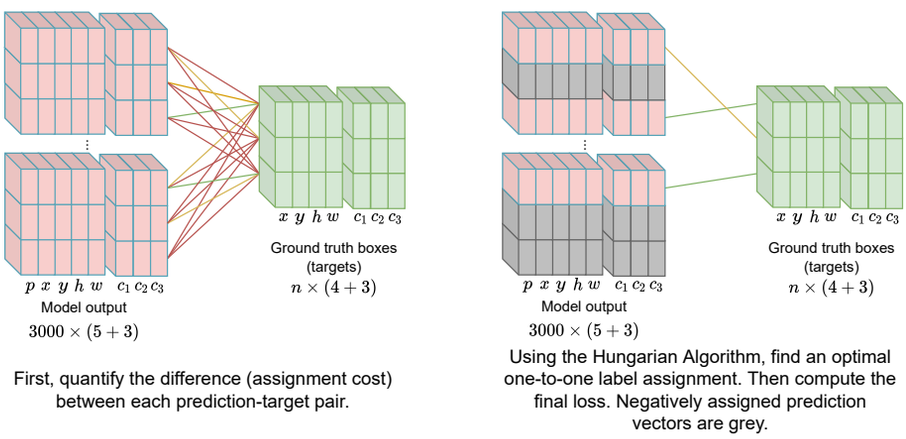

Figure 2.6: The DETR loss label assignment. The color of lines connecting prediction vectors and target vectors represents the label assignment cost for that combination.

backbone such as ResNet [7], paired with a feature pyramid network (FPN) [11] extracts semantically strong feature maps at different scales (strides of 2, 4, 8, ...). Each feature map is further processed by convolutional operations and a 3D max filtering operation [24], to produce YOLOF-like detection grids. During inference, a simple thresholding operation (eg. $p > 0.5$) is applied to each detection grid, and results are concatenated. No post-processing such as NMS is required.

DeFCN uses a loss label assignment method inspired by DETR. Each ground truth box is assigned to exactly one unique prediction vector, using a linear sum assignment. The assignment cost function $\mathcal{L}_{match}$ used in DeFCN includes a prior multiplier that favors prediction vectors corresponding to image regions close to the target.

The feature pyramid network backbone (FPN) [11] in DeFCN improves its accuracy for detecting small objects. FPN is therefore used in most leading object detectors. However, as pointed out by the YOLOF authors, an FPN backbone will approximately double the computational costs of a detector [4].

DeFCN uses 3D max filtering (3DMF) [24] to predict fewer duplicate detections. 3DMF combines features of all feature levels, and increases the detector's ability to decide which prediction vectors are responsible for detecting which objects across feature levels. Exact mathematical details are omitted.

In addition to the detection grids learned by DeFCN shown in Figure 2.7, DeFCN has an auxiliary detection branch at each feature level. This is only used during

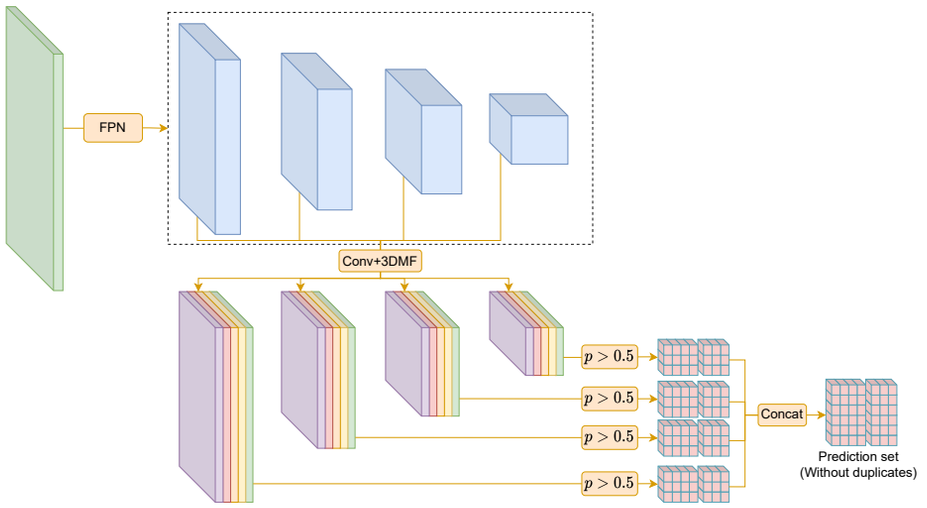

Figure 2.7: Rough illustration of the DeFCN architecture. First, a backbone CNN with FPN extracts feature maps at different scales. Then, a 3D max filter operation and sequence of convolutions produce YOLOF-like detection vector grids at each feature level. During inference, a thresholding operation is applied to each detection grid, and detections are concatenated. Duplicate removal is not required.

training, and is used for an auxiliary loss function. The auxiliary branches use a many-to-one loss label assignment (similar to the one in YOLOF [4]), resulting in more regression labels being assigned to output heads, which strengthens the feature maps. This benefits the main detection branches that are used during inference, without resulting in duplicate predictions.

2.2.4 Bounding box losses

This section describes some influential loss functions that have been used throughout the object detection research used to optimize the predicted position and size of bounding boxes. Current state of the art DIoU and CIoU [30] are incompatible with data sets where size annotations are missing from certain objects, and we propose a new loss function based on a combination of concepts from different loss functions used by different object detectors.

Originally, object detection models used separate loss functions for position optimization and size optimization. Fast R-CNN [6] proposed smooth_{L_1} defined in Equation 2.3, and used this function for height, width, x, and y regression of bounding boxes as defined in Equation 2.4. smooth_{L_1} is used in many object detectors. However, smooth_{L_1} has an imbalance problem as large objects tend to induce orders of magnitude more loss than small objects, causing a bias

towards large bounding boxes.

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{, } |x| > 1 \\ |x| - 0.5 & \text{, otherwise} \end{cases} \tag{2.3}$$

$$\mathcal{L}_{fast-rcnn} = \sum_{v \in \{x,y,w,h\}} \text{smooth}_{L_1}(v - \hat{v}) \tag{2.4}$$

You only look once (YOLO) [18] tackled the bias towards large objects by using a loss function that computes the squared difference between the square root of the height and width, as shown in Equation 2.5. This improved the loss imbalance problem slightly, but still suffered from some imbalance.

$$\mathcal{L}_{yolo}(h, w, \hat{h}, \hat{w}) = (\sqrt{h} - \sqrt{\hat{h}})^2 + (\sqrt{w} - \sqrt{\hat{w}})^2 \tag{2.5}$$

Intersection over union (IoU) has two key benefits when used for joint optimization of position and size of objects. First, it induces the same amount of loss for both small and large objects, and is therefore a simple solution to the balance problem. Furthermore, it implicitly scales down loss induced by small errors in position predictions for large objects. Therefore, small localization errors in small objects receive more attention than small localization errors in large objects.

However, IoU has a major drawback as a loss function. IoU is zero when bounding boxes do not overlap, meaning gradients are also zero. IoU loss therefore takes a lot of time to optimize, since most update steps are noneffective. Generalized IoU loss (GIoU) [20] addressed this problem and results in faster convergence. This inspired Distance IoU (DIoU) and Complete IoU (CIoU) [30], where CIoU is considered state of the art for object localization loss. DIoU is defined in Equation 2.6, where c is the length of the diagonal of the smallest bounding box containing both bounding boxes. DIoU has nonzero gradients everywhere. CIoU is defined in Equation 2.7, and adds some additional loss if the predicted aspect ratio is erroneous.

$$
\begin{aligned}
\mathcal{L}_{DIoU} &= 1 - IoU + \frac{(x-\hat{x})^2 + (y-\hat{y})^2}{c^2} \\
c_y &= \max(y + \frac{h}{2}, \hat{y} + \frac{\hat{h}}{2}) - \min(y - \frac{h}{2}, \hat{y} - \frac{\hat{h}}{2}) \\
c_x &= \max(x + \frac{w}{2}, \hat{x} + \frac{\hat{w}}{2}) - \min(x - \frac{w}{2}, \hat{x} - \frac{\hat{w}}{2}) \\
c^2 &= c_y^2 + c_x^2
\end{aligned}
\tag{2.6}
$$

$$
\begin{aligned}
\mathcal{L}_{CIoU} &= \mathcal{L}_{DIoU} + \alpha v \\
v &= \frac{4}{\pi^2} \left(\arctan \frac{w}{h} - \arctan \frac{\hat{w}}{\hat{h}} \right)^2 \\
\alpha &= \frac{v}{(1 - IoU) + v}
\end{aligned}
\tag{2.7}
$$

2.2.5 Summary of horizontal bounding box detectors

We have now described YOLOF, DETR, and DeFCN [4, 3, 24], with focus on fundamental concepts that are used across many object detection systems. Our model's architecture is similar to YOLOF (Figure 2.1), but uses a different loss function and discards NMS. We adapt a loss label assignment method and auxiliary loss inspired by DeFCN, but discard feature pyramids and 3DMF. We use CIoU and DIoU loss as inspiration and propose SIoU loss, which is compatible with datasets where size annotations are not available for all obejcts.

2.3 Oriented object detection

Object detection with rotated bounding boxes, referred to as oriented object detection, is widely used in aerial images captured by satellites and air crafts. Oriented object detection is based on horizontal (non-rotated) object detection models such as those described above, with small modifications. This section describes the two main paradigms in oriented object detection. These paradigms differ in how responsibility is distributed across the different prediction vectors.

The most important concept to understand in this section is as follows. What is the border discontinuity problem, and how do state of the art rotation regression methods overcome this problem. We propose a new rotation regression method that is simpler and on average more accurate than current state of the art.

The rest of this section is organizes as follows. The first two subsections describe the main paradigms in oriented object detection. The third subsection describes the state of the art in rotation regression which solves the border discontinuity problem, which is only a sub problem in oriented object detection.

2.3.1 Oriented object detection with rotated anchor boxes

First, detectors with rotated anchor boxes use a fixed responsibility, where the different predictions are responsible for detecting differently rotated objects. This was proposed by DRBox [14], and was a generalization of the anchor box concept which was later adapted by YOLOF [4]. In YOLOF, the different predictions are responsible for objects at different sizes and aspect ratios. In DRBox, the different predictions are responsible for objects with different sizes, aspect ratios, *and rotations*. Additionally, prediction vectors in DRBox contain a scalar for rotation regression, which is an offset within the rotation interval the given prediction is responsible for. For instance, one prediction may be responsible for objects with rotation in the small $[20°, 30°)$ range, and a rotation regression output corresponding to that prediction is simply a small offset within the given range. However, rotated (and horizontal) anchor boxes are incompatible with datasets where the size and rotation of some ground truth bounding boxes is unknown, since the different predictions have different mathematical restrictions on which bounding boxes they can represent. During training, given a ground truth box, it is crucial that a prediction that is able to represent that ground truth box is assigned to the task, but this is impossible if the true rotation of the box is unknown. We need a model with more flexibility for this reason.

2.3.2 Oriented object detection with rotation regression

The second paradigm in oriented object detection, which we use in our work, does not use a fixed distribution of responsibility. So, the different predictions are not responsible for unique small rotation intervals as with rotated anchor boxes where one prediction may be responsible for only objects with rotations in the $[20°, 30°)$ range. Instead, any prediction vector must be able to predict objects with rotation in the entire $[0°, 360°)$ range. Our work is based on this approach, because it is compatible with datasets where rotation labels may be unknown.

2.3.3 Rotation regression methods

However, when rotation regression represents the entire range $[0°, 360°)$, the *border discontinuity problem* is encountered; A *minimum rotation* (0°) and a *maximum rotation* (360°) are logically adjacent. Therefore, neural network outputs corresponding to minimum and maximum rotations should be similar. When rotation regression is represented with direct angle regression, a minimum angle of 0° and a maximum angle of 360° will be represented with different neural network outputs. Hence, the neural network is trying to learn a discontinuous mapping. However, since neural networks are continuous functions, it is difficult for them to approximate discontinuous functions. Y. Zhou et. al [31] provides a mathematical definition of the border discontinuity problem and continuous rotation representations.

Full rotation regression is a challenging problem. Yang et. al [26] proposed circular smooth label (CSL) and grey coded labels (GCL). CSL is state of the art in rotation regression accuracy, but GCL requires fewer parameters and is a more lightweight approach. Both models represent rotation regression as a classification problem to overcome the border discontinuity problem, and are described below.

Circular smooth labels (CSL) predicts an n bit classification into n ranges of either $360°/n$ width. Each output bit has sigmoid activation. During training, each bit receives a label between 0 and 1, based on the difference between the true rotation and the rotation corresponding to that class. Figure 2.8 illustrates how the n classification bits are assigned labels in CSL. Multiple classes that are close to the true rotation will receive a positive label. During inference, the class with highest activation is chosen. CSL overcomes the border discontinuity problem, since minimum and maximum rotations are represented by similar network outputs. However, the number of bits and the width of the window function (label assignment rule) need to be decided and tuning of these hyper parameters affects performance.

Grey Coded Labels (GCL) predicts n classification bits, and uses an encoding scheme that represents 2^n classes using n bits. The encoding scheme is based on grey codes. Grey codes is similar to binary encoding, but uses an alternative ordering instead of $(000, 001, 010, 011..., 111)$ which is used by binary codes. A valid three bit grey code could have the following ordering: $(000, 001, 011, 010, 110, 111, 101, 100)$. Notice how two adjacent codes only differ by only a single bit, and how the first and last codes also satisfy this property. Across the entire rotation range, similar rotations have similar output encoding, including the wrapping border. Therefore, GCL overcomes the border discontinuity problem.

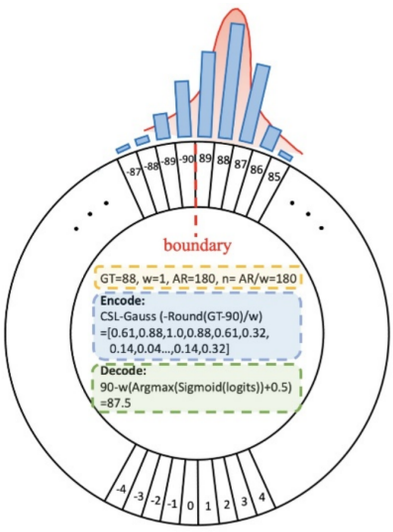

Figure 2.8: Circular Smooth Label (CSL) classification for angular regression. (Yang et. al [26])

In addition to the discrete classification outputs, CSL and GCL predict a real valued regression output, which represents a small offset from the predicted rotation classification.

2.4 SAR ship detection research

Open datasets such as Large scale SAR ship detection dataset (LS-SSDD) [28] and High resolution SAR images dataset (HRSID) [25] are commonly used for researching ship detection in SAR images. LS-SSDD consists of 800×800 pixel images with $10m$ pixel spacing from Sentinel 1, with horizontal bounding box annotations. HRSID also contains 800×800 pixel images from various satellites and $1m$ to $5m$ pixel spacing. HRSID has horizontal bounding box annotations and instance segmentation annotations. Neither data sets have labels for physical ship size, but since they have bounding box annotations, we use them for comparing SIoU against DIoU and to compare position based loss label assignment against IoU based loss label assignment.

Some SAR ship detection research focus on feature extraction in SAR images. They have found that CNNs have a limited ability to learn SAR feature representations end-to-end, due to specific properties in SAR data such as speckle noise, and that certain mechanisms can improve feature representation in SAR images. In [29], they found that jointly training an object detector with auxiliary tasks improved detection accuracy. They propose auxiliary tasks that

require low level semantic features which easily learned compared to high level object detection task, and therefore results in better feature representations. In [27], they propose a CNN backbone that incorporates HOG features, improves the models accuracy. Hence, this research addresses feature representation challenges in SAR data, while we address orthogonal challenges related to limited access to ground truth in SAR data.

BiFA-YOLO [22] is an object detector for rotated ships in high resolution SAR images, and uses CSL [26] for rotation regression. However, it uses datasets where ship size annotations are available for all ships, which is possible because annotations correspond to their appearance in images and not their physical size. BiFA-YOLO is not compatible with datasets where certain size annotations are missing.

3

Proposed solutions

3.1 Loss label assignment with overlapping responsibility

Some object detection datasets have inaccurately labeled object positions. KSATs ship detection datasets are examples of this, where the data collection is done as part of a real time service, where exact localization of ships is a low priority compared to fast detection, and SAR specialists do not waste time on annotating accurate ship detections. Some loss label assignment methods are highly sensitive to noise in position labels. We propose a loss label assignment method where different predictions have large overlapping regions of responsibility, which is significantly more robust to inaccurately labeled positions. We define the method as follows.

Given an image with size $h \times w$, we have an output grid containing $h/32 \times w/32 \times n$ vectors. The output grid has a stride of 32 pixels relative to the input.

We let $(i, j) \in \mathcal{I} \times \mathcal{J} = \{0, 1, ..., h/32 - 1\} \times \{0, 1, ..., w/32 - 1\}$ be spatial coordinates in the output grid. In grid cell (i, j), we let $k \in \mathcal{K} = \{0, 1, ..., n-1\}$ denote a single prediction's index, where n is the number of predictions per cell. For convenience, we let $\hat{i} = (i, j, k) \in \hat{\mathcal{I}} = \mathcal{I} \times \mathcal{J} \times \mathcal{K}$ denote the index of a single prediction vector.

The unit of coordinate vectors is relative to the stride (downsample rate) of the neural network output, which is 32 pixels, with positive direction towards the bottom right of images. So, a coordinate vector $\underline{x} = (0, 0)$ is the upper left corner, and $\underline{x} = (1, 1)$ is 32 pixels from the upper left corner in the x and y directions.

$p^{\hat{i}}$ denotes a single prediction. $p_p^{\hat{i}} \in (0, 1)$ denotes the positivity classification score of the prediction, and $\underline{p_x^{\hat{i}}} \in (-\gamma, \gamma)^2$ denotes the predicted center position, as an offset from the center of the corresponding 32×32 image region. We use $\gamma = 2.3$. $\underline{p_{xa}^{\hat{i}}}$ denotes the absolute position of the prediction, and is computed using $\underline{p_{xa}^{\hat{i}}} = \underline{p_x^{\hat{i}}} + \underline{c_{ij}}$, where $\underline{c_{ij}} = (i + 0.5, j + 0.5)$ is the center of the image region corresponding to grid cell (i, j).

We let $l \in \mathcal{T} = \{0, ..., m-1\}$ denote the index of ground-truth object t^l, where m is the true number of objects in the image. We let $\underline{t_x^l}$ denote the position of the objects center point.

We use the following loss label assignment. Let $\underline{\underline{A}} = (\underline{a}(0), ..., \underline{a}(l), ..., \underline{a}(m-1))$ where $\underline{a}(l) \equiv \hat{i} = (i, j, k)$ denote the index of the anchor assigned to target l. Following DETR and DeFCN [3, 24], we choose $\underline{\underline{A}}$ from the set of bipartite matchings ($\mathcal{A}$). That is, we constrain $\underline{\underline{A}}$ to be a one-to-one label assignment, meaning one prediction vector can be assigned to at most one target, and each target is assigned to exactly one prediction. We choose $\underline{\underline{A}} \in \mathcal{A}$ to maximize the linear sum assignment in Equation 3.1, where $Q_{\hat{i},l}$ is a quality metric for assigning $p^{\hat{i}}$ to t^l.

$$\underline{\underline{A}} = \arg\max_{\underline{\hat{A}} \in \mathcal{A}} \sum_{l \in \mathcal{T}} Q_{\underline{\hat{a}}(l),l} \tag{3.1}$$

We define $Q_{\hat{i},l}$ in Equation 3.2. The spatial prior $\sigma_{ijl} \in [0, 1]$ is defined such that if the cell-center to object-center distance ($\|\underline{c_{ij}} - \underline{t_x^l}\|$) is longer that a hyper-parameter β, we have $\sigma_{ijl} = 0$, and otherwise we have $0 < \sigma_{ijl} \leq 1$. Q'_{ijkl} is the match quality before applying the spatial prior, and is a weighted sum of the positivity of the prediction (to favorize positive predictions over negatives) and a regression similarity metric Q_{ijkl}^{Reg} (to favorize predictions that are similar to the target).

$$
\begin{aligned}
Q_{\underline{\hat{a}}(l),l} = Q_{ijkl} &= \sigma_{ijl}\, Q^{'}_{ijkl} \in [0,1] \\
\sigma_{ijl} &= \max(\sigma^{'}_{ijl}, 0) \in [0,1] \\
\sigma^{'}_{ijl} &= 1 - \frac{\|\underline{c_{ij}} - \underline{t^l_x}\|^2}{\beta^2} \in (-\infty, 1] \\
Q^{'}_{ijkl} &= \lambda_p\, p^{ijk}_p + \lambda_x\, Q^{Reg}_{ijkl} \\
\lambda_p + \lambda_x &= 1
\end{aligned}
\tag{3.2}
$$

We use Q^{Reg}_{ijkl} as defined in Equation 3.3. Given $\sigma_{ijl} > 0$, the center-to-center distance has an upper bound $\|\underline{p^{ijk}_{xa}} - \underline{t^l_x}\| < \beta + \sqrt{2}\gamma$. Hence, $Q^{Reg}_{ijkl} \in [0,1]$ given $\sigma_{ijl} > 0$. If $\sigma_{ijl} = 0$, the value of Q^{Reg}_{ijkl} may be negative, but will in turn be multiplied with zero. Therefore, we effectively have $Q^{Reg}_{ijkl} \in [0,1]$.

$$
Q^{Reg}_{ijkl} = 1 - \frac{\|\underline{p^{ijk}_{xa}} - \underline{t^l_x}\|^2}{(\beta + \sqrt{2}\gamma)^2}
\tag{3.3}
$$

Both DeFCN and DETR, which are the models most similar to ours in terms of loss label assignment, use $Q^{Reg}_{ijkl} = IoU(p^{ijk}, t^l)$. However, our definition of Q^{Reg}_{ijkl} does not require that the true size is known, which is beneficial when size (and rotation) labels are unavailable for some samples in the data set.

3.2 Non-overlapping position based loss label assignment

Position based loss label assignment with non-overlapping responsibility was proposed in an pre-project prior to writing this thesis. This is similar to the overlapping position based loss label assignment described in Section 3.1. This allows training a detector when the dataset is missing size annotations, but is still sensitive inaccurately labeled positions. In this section, we describe simple modifications that can be made to the method described above, to create a non-overlapping loss label assignment method. By simply redefining the spatial prior σ_{ijl}, the regression similarity metric R^{Reg}_{ijkl}, and the hyper parameter γ, we get a non-overlapping label assignment.

We describe this label assignment method to illustrate a small part of the

combined project that has already been proposed. To be clear, in the pre-project, we never compared the performance of position based label assignment to other types of label assignment. We only defined the method and did an experiment that showed decent performance, but not compared to any baseline.

We redefine the spatial prior σ_{ijl} in Equation 3.4, so it is either 0 or 1 instead of anything in between, and is only based on which (disjoint) image regions contains the given object center position $\underline{t_x^l}$.

$$\sigma_{ijl} = \begin{cases} 1 & , \ \underline{t_x^l} \in [i, i+1) \times [j, j+1) \\ 0 & , \ \text{otherwise} \end{cases} \tag{3.4}$$

We redefine the regression similarity metric R_{ijkl}^{Reg} in Equation 3.5. This is still only based on the squared distance between the prediction and the target as the one described previously, but since the regions are smaller we re-scale the distance term.

$$Q_{ijkl}^{Reg} = 1 - \frac{\|\underline{t_x^l} - \underline{p_{xa}^{ijk}}\|^2}{(\sqrt{2} + \sqrt{2}\gamma)^2} \tag{3.5}$$

We use a different configuration of the hyper parameter γ. Since the predictions are responsible for small 32×32 regions, the predicted position offsets do not need the capacity to be very large, and we do not need as large γ. We use $\gamma = 0.65$ (where 0.5 would be the minimum valid configuration), which allows predicted positions slightly outside the dedicated regions of responsibility, to avoid tanh saturation problems near the edges of the image regions, following YOLOv4 [2]. The β hyper parameter is obsolete, as the size of regions of responsibility are no longer tunable.

These three modifications to overlapping loss label assignment result in a non-overlapping loss label assignment which is equivalent to the one proposed in the project prior to this thesis. However, due to the non-overlapping regions of responsibility, this loss label assignment makes detectors sensitive to inaccurately labeled object positions.

3.3 Multi layer attention module

An important aspect of end-to-end training in object detection is to eliminate all post-processing of the models output. This has two main benefits. First, post-processing steps such as duplicate removal with non-max suppression (NMS) limit the performance of detectors, as they do not utilize enough information to make correct decisions about removing duplicates. Second, post-processing steps may require careful hyper parameter tuning to achieve good performance, such as the tuning the IoU overlap threshold in NMS, which is inconvenient. For these reasons, we aim to design an end-to-end trainable object detection model.

However, as pointed out by the authors of DeFCN [24], a one-to-one loss label assignment method, such as the one we propose, is not enough to fully eliminate post-processing. It is important that the neural network architecture is able to learn how to decide which prediction vector is responsible for a given object when processing an image. Convolutions with ReLU activation have a limited representational capacity for this problem, causing issues with duplicate predictions, and for this reason, DeFCN includes the 3D max filter operation which suppresses a significant amount of duplicate predictions [24].

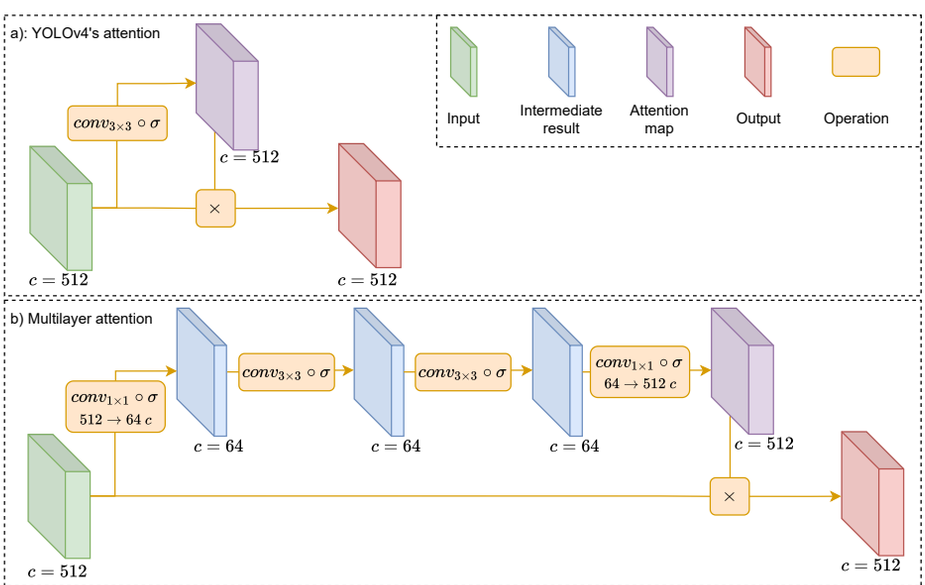

Figure 3.1: Our multi layer attention module compared to YOLOv4's spatial attention module. c denotes the number of channels in each feature map.

We propose an attention module that achieves the same as 3D max filtering. Put simply, attention modules learn to recognize when certain features in a feature map should be switched off by being multiplied with either zero or

one. This means that an attention module has the power to decide which predictions become positive or negative, as it can enable or disable the features that trigger the classification output of each prediction vector. So, if an attention module has the capacity to approximate complex logic, it should be able to efficiently synchronize the positivity classification scores of nearby detection vectors.

Attention modules used in object detection are usually a single convolutional layer with sigmoid actication, such as the spatial attention module (SAM) in YOLOv4 [2]. However, a single neural network layer with sigmoid activation has very limited capabilities in terms of which logic functions they can approximate, and can for instance not learn the xor function. However, multiple such layers are able to learn complex logic functions, providing a simple way to increase the representational capacity of an attention module.

For there reasons, we propose the multi layer attention module shown in Figure 3.1 b). This attention module can learn more complex logic functions that single layer attention modules, which can be an effective way to suppress duplicate predictions, as it can improve the architectures ability to synchronize the positivity classification scores of the different prediction vectors.

The multi layer attention module first applies a 1×1 convolution to reduce the number of features, and then applies a series of 3×3 convolution, before it increases the number of features with another 1×1 convolution, producing the final attention mask. To compare, YOLOv4's attention module (Figure 3.1 a)) is a single 1×1 convolution, but 3×3 convolutions are important for this purpose. Without 3×3 convolutions, the attention module can only learn logic that operates on each spatial position in the feature map in isolation. So, the YOLOv4 SAM cannot learn logic that synchronizes prediction vectors at different spatial positions in the output, which is important to suppress duplicates. However, the SAM module was not designed for this purpose, as YOLOv4 uses NMS.

Our multi layer attention module actually requires fewer parameters that the YOLOv4 SAM [2]. We use bottleneck layers to reduce the number of parameters in the 3×3 convolutions. We decrease the number of parameters in each 3×3 convolution from $3 \times 3 \times 512 \times 512 = 2M$ to $3 \times 3 \times 64 \times 64 = 36k$. The down sample and up sample layers have $1 \times 1 \times 64 \times 512 = 32k$ parameters. In total, the attention module in Figure 3.1 a) has $32k + 63k \cdot 3 + 32k = 172k$ parameters, which is around a tenth of what is required for YOLOv4's SAM [2]. Still, the number of layers and number of features in each layers are easily tunable hyper parameters.

3.4 Angular direction vector (ADV) regression

In low resolution SAR images, access to ground truth rotation labels is limited, since it depends on external data sources such as AIS, resulting in inaccurate rotation labels. Circular smooth labels (CSL) has achieved state of the art in rotation regression by overcoming the border discontinuity problem, but is sensitive to inaccurate rotation labels. We propose ADV, an alternative rotation regression method that overcomes the border discontinuity problem, but is more robust to inaccurate rotation labels than CSL. Instead of direct rotation regression, ADV predicts a direction vector corresponding to the rotation angle.

3.4.1 ADV definition

We let θ_y and $\theta_x \in [0°, 360°)$ denote the true and predicted angles respectively. We define $d(\cdot)$ to map a rotation angle to a unit length direction vector and define $d^{-1}(\cdot)$ as its inverse, as in Equations 3.6 and 3.7.

$$d(\theta) = (\sin\theta, \cos\theta) \tag{3.6}$$

$$d^{-1}(x_1, x_2) = \begin{cases} \arcsin x_1 \bmod 360° & , \ x_2 > 0 \\ 180° - \arcsin x_1 & , \ \text{otherwise} \end{cases} \tag{3.7}$$

We let $\underline{y} = d(\theta_y)$ be a direction vector representing θ_y, which is used as ground-truth when computing ADV loss.

$$\underline{y} = (y_1, y_2) = d(\theta_y) \tag{3.8}$$

During inference, ADV predicts an (arbitrary length) direction vector $\underline{x'} \in \mathbb{R}^2$, which is unit normalized by computing $\underline{x}$ as in Equation 3.10. Finally, ADV computes $\theta_x = d^{-1}(\underline{x})$ to get the rotation angle which is the final output during inference.

$$\underline{x'} = (x'_1, x'_2) \in \mathbb{R}^2 \tag{3.9}$$

$$\underline{x} = \frac{\underline{x'}}{\|\underline{x'}\|} \tag{3.10}$$

$$\theta_x = d^{-1}(\underline{x}) \tag{3.11}$$

As mentioned, during training, we use $\underline{y} = d(\theta_y)$ as ground truth, as shown in Equation 3.8. We propose three different loss functions for ADV, and show that

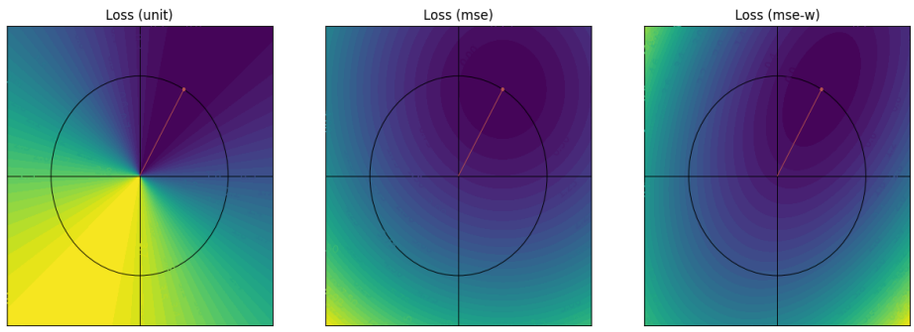

Figure 3.2: Contour map of the three alternatives to the ADV loss function. The red vector is the ground truth vector $\underline{y}$, and the color at each position corresponds to loss at that position. $\mathcal{L}_{unit}$ (left) is unaffected by the length of the vector, resulting in a strangely shapes loss surface. $\mathcal{L}_{mse}$ (middle) has a circular hyperbolic loss surface. $\mathcal{L}_{mse-w}$ (right) is shown with $(\lambda_1, \lambda_2) = (0.5, 1.5)$, and has an elliptic hyperbolic loss surface, which induces less loss if errors in the prediction do not contribute to an erroneous predicted direction.

the latter brings superior performance. $\mathcal{L}_{unit}$, $\mathcal{L}_{mse}$, $\mathcal{L}_{mse-w}$ are defined in Equations 3.12, 3.13 and 3.14. The first two are simply the squared difference between the predicted and true vectors, but differ in that the first uses the unit-normalized $\underline{x}$ while the second uses the un-normalized $\underline{x'}$. The third option $\mathcal{L}_{mse-w}$ is actually a generalization of the second $\mathcal{L}_{mse}$ with two weight hyper parameters, where $\lambda_1 = \lambda_2 = 1$ implies $\mathcal{L}_{mse-w} = \mathcal{L}_{mse}$. We use $(\lambda_1, \lambda_2) = (0.5, 1.5)$ for this loss function as it provides the best performance overall. Figure 3.2 shows a contour map of the three different loss functions.

$$\mathcal{L}_{unit} = \|\underline{y} - \underline{x}\|^2 \tag{3.12}$$

$$\mathcal{L}_{mse} = \|\underline{y} - \underline{x'}\|^2 \tag{3.13}$$

$$\begin{aligned} \mathcal{L}_{mse-w} &= \lambda_1 \, (x_1^p - 1)^2 + \lambda_2 \, (x_2^p - 0)^2 \\ x_1^p &= \underline{x'} \cdot (y_1, y_2) \\ x_2^p &= \underline{x'} \cdot (y_2, -y_1) \\ (\lambda_1, \lambda_2) &= (0.5, 1.5) \end{aligned} \tag{3.14}$$

The first option $\mathcal{L}_{unit}$ may seem as a good alternative as it is not affected by the length of the predicted vector and instead focuses entirely on the direction of the vector which is more important. However, we show analytically and empirically that this loss function is difficult to optimize, and results in poor performance.

The second option $\mathcal{L}_{mse}$ is easier to optimize. However, it induces the same amount of loss for errors in all directions. Errors that affect the direction of the vector are more important than errors that affect the length of the vector. This motivates the third option $\mathcal{L}_{mse-w}$, which generalizes $\mathcal{L}_{mse}$ from a circular hyperbolic loss surface to an elliptic hyperbolic loss surface. We do this by first transforming the predicted vector $\underline{x'}$ into a new vector room. The first component x_1^p is the projection of $\underline{x'}$ in the direction of $\underline{y}$, and the second component is the projection of $\underline{x'}$ in the direction perpendicular to $\underline{y}$. We want $\underline{x^p} = (1, 0)$ as it is equivalent to $\underline{x'} = \underline{y}$, but $x_2^p = 0$ is more important than $x_1^p = 1$ as it affects the direction of the vector and not the length of the vector.

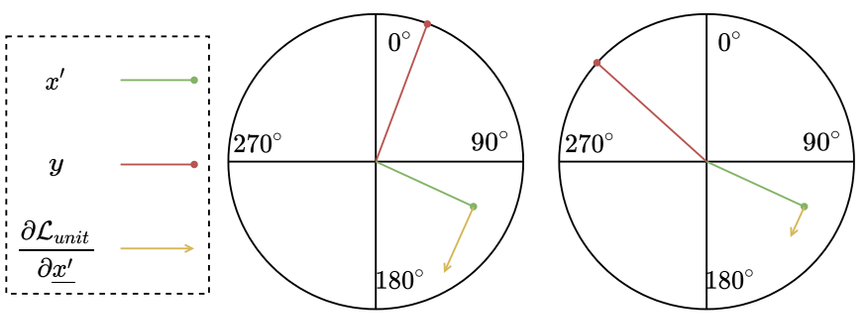

Figure 3.3: The gradient's direction when using $\mathcal{L}_{unit}$ The gradient is always perpendicular to the predicted direction vector, regardless of the target direction. In many cases (eg. right) this gradient direction results in highly inefficient weight updates.

We inspect the gradient of $\mathcal{L}_{unit}$ with respect to $\underline{x'}$ to demonstrate as important weakness. The partial derivatives the the loss with respect to x_1' and x_2' are shown in Equations 3.15 and 3.16.

$$\frac{\partial \mathcal{L}_{unit}}{\partial x_1'} = 2\frac{x_2' \ (x_1' y_2 - x_2' y_1 \ )}{(\ {x_1'}^2 + {x_2'}^2 \ )^{\frac{3}{2}}} \tag{3.15}$$

$$\frac{\partial \mathcal{L}_{unit}}{\partial x_2'} = 2\frac{x_1' \ (x_2' y_1 - x_1' y_2 \ )}{(\ {x_1'}^2 + {x_2'}^2 \ )^{\frac{3}{2}}} \tag{3.16}$$

We identify the common scaling factor in both partial derivatives, α, shown in Equation 3.17. We substitute α to simplify the partial derivatives, and get the gradient in Equation 3.18.

$$\alpha = 2\frac{x_1' y_2 - x_2' y_1}{(\ {x_1'}^2 + {x_2'}^2 \ )^{\frac{3}{2}}} \tag{3.17}$$

$$\frac{\partial \mathcal{L}_{unit}}{\partial \underline{x'}} = \alpha \ (x_2', -x_1' \ ) \tag{3.18}$$

$$\frac{\partial \mathcal{L}}{\partial \underline{x'}} \cdot \underline{x'} = 0 \tag{3.19}$$

Evidently, the predicted direction vectors gradient is always perpendicular to the predicted direction vector, according to Equation 3.19. This is visualized in Figure 3.3, and is also evident in Figure 3.2 (left) since the gradient points in the direction of steepest ascent, which *is* perpendicular to the predicted vector everywhere in the contour map. If the initial direction vector is far off from the target, the gradient will point in a direction that barely improves the accuracy, meaning it will need many update steps to converge. Instead, we want the gradient to point more directly in the direction of the target, which ADV achieves using either $\mathcal{L}_{mse}$ or $\mathcal{L}_{mse-w}$.

3.4.2 Generalized ADV

In some rotation regression problems, such as ship rotation in SAR images, it is not always possible to distinguish the front and back of objects. If a network is attempting to predict the orientation of an object where the front and back are indistinguishable, it will produce an angle that is a combination of the right angle and the inverse angle (+180°), which will likely be a very bad estimate. The typical solution in these applications is to instead model the rotation modulo 180°, embracing the fact that the front and back are indistinguishable, and that the predicted rotation is either the right direction or the inverse direction.

However, with ADV regression (and also CSL and GCL), this is only a partial solution, and results in a border discontinuity problem as illustrated in Figure 3.4 b). Rotations of 3° and 177° which close to the minimum and maximum rotation, are logically adjacent, and may even be indistinguishable. Still, their direction vector representations are at opposite sides of the representation space. To adapt ADV in these applications, we instead use the representation in Figure 3.4 a), where an angle $\phi \in [0°, 180)$ is represented by a direction vector corresponding to the angle $2\phi \in [0°, 360°)$. This overcomes the border discontinuity problem.

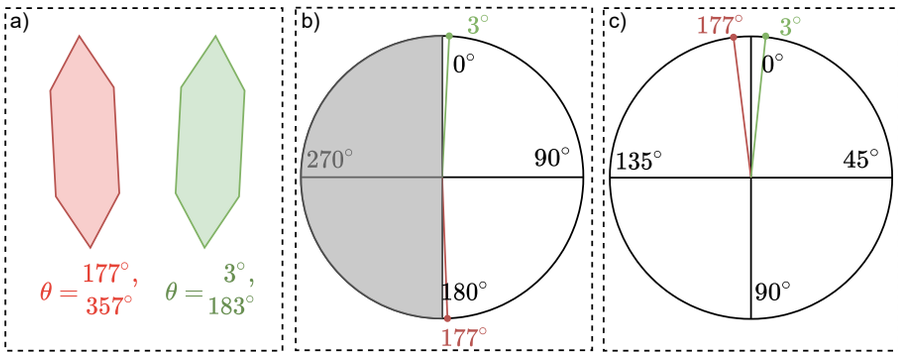

Figure 3.4: Generalized ADV regression for applications where the front and back of objects are indistinguishable. a) Two differently oriented objects with indistinguishable front and back sides due to their symmetric shape. b) Naive solution. ADV learns $\theta \bmod 180° \in [0°, 180°)$ covering half of the representation space. However, $\theta = 3°$ and $\theta = 177°$ are represented with very different direction vectors while they should have similar representations. This results in a border discontinuity problem. c) Better solution. ADV learns 2 $(\theta \bmod 180°) \in [0°, 360°)$, covering the entire representation space. $\theta = 3°$ and $\theta = 177°$ are represented with similar direction vectors, solving the border discontinuity problem. Note the rotation axis labels ($0°, 45°, 90°, 135°$ in black text) corresponding to the rotation of the objects $\theta \bmod 180°$ and not the angle of the ADV vector 2 $(\theta \bmod 180°)$.

3.5 Size IoU loss (SIoU)

In SAR images there is limited access to ground truth size annotations of ships, since estimating the physical size of ships from SAR images can be extremely difficult and inaccurate. Instead we rely on external data sources such as AIS to collect ground truth size annotations of ships, but as these are less reliable, the resulting data set contains some ships with missing size annotations.

3.5.1 Independent size and position losses

When the true size of an object is unknown, we cannot optimize the predicted size, but we still want to optimize the predicted position. However, state of the art loss functions for optimizing bounding box localization such as DIoU and CIoU loss [30] are incompatible with this requirement, since they operate on predicted position and size jointly. As a solution, we phrase position and size regression as two independent regression problems with independent loss functions, as in Equation 3.21, where $\mathcal{L}^{Pos}$ and $\mathcal{L}^{Size}$ are the position and size losses. If the true size of an object is unknown, we simply ignore size loss for

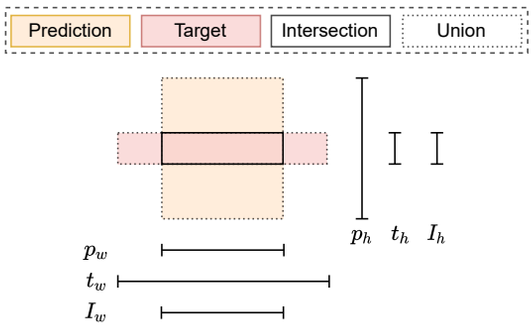

Figure 3.5: Size IoU (SIoU) loss. Predicted and true bounding boxes are assumed to have the same center, which in general is not true. SIoU is simply the IoU, given this assumption. p_h, p_w is the predicted height and width, t_h, t_w is the true height and width, and I_h, I_w is the intersections height and width.

that object, and can still optimize the predicted position.

$$\mathcal{L}^{Box}_{DIoU}(p,t) = DIoU(p,t) \tag{3.20}$$

$$\mathcal{L}^{Box}_{Sum}(p,t) = \mathcal{L}^{Pos}(p,t) + 1[size]\mathcal{L}^{Size}(p,t) \tag{3.21}$$

$$1[size] = \begin{cases} 1 & , \text{ Size of ground truth known} \\ 0 & , \text{ Otherwise} \end{cases} \tag{3.22}$$

We need to choose which $\mathcal{L}^{Pos}$ and $\mathcal{L}^{Size}$ functions to use. Prior to DIoU loss, when independent loss functions for size and position loss were state of the art, the loss functions induced disproportional loss for large and small objects, which is one of the main problems DIoU solved. In ship detection, any choice of position loss will work, since exact positioning is not a priority, so we simply adapt smooth_{L_1} for position loss, as in Equation 3.23. However, size loss is more important, since end users are concerned with ship sizes, so we design SIoU loss for size regression, which induces similar loss for small and large objects just like DIoU.

$$\mathcal{L}^{Pos}(p,t) = \text{smooth}_{L_1}(\underline{p_x} - \underline{t_x}) \tag{3.23}$$

3.5.2 SIoU definition

Size IoU (SIoU) is defined in Equation 3.24 and illustrated in Figure 3.5. Intuitively, SIoU is IoU of the prediction and the target, under the assumption that both boxes have the exact same center position, but different height and width. t_h, t_w, p_h and p_w denote the true and predicted height and width. I_h and

I_w is the height and width of the intersection, and I_a is the area of intersection. U_a is the area of the union.

$$
\begin{aligned}
SIoU &= \frac{I_a}{U_a} \in [0, 1] \\
t_a &= t_h \cdot t_w \\
p_a &= p_h \cdot p_w \\
I_a &= \min(t_h, p_h) \cdot \min(t_w, p_w) \\
U_a &= t_a + p_a - I_a
\end{aligned}
\tag{3.24}
$$

3.6 Detector design

This section summarizes the overall composition of the detector model. Figure 3.6 displays the architecture. An image is first processed by a ResNet34 backbone [7] and then a dilated encoder [4]. The dilated encoder output is further processed by two independent branches. The upper branch does the actual detection, while the lower branch is only used to compute an auxiliary loss with a many-to-one loss label assignment, which reinforces feature representations used by the main branch [24]. Loss from the main and auxiliary branches are added through a weighted sum to compute the final model loss.

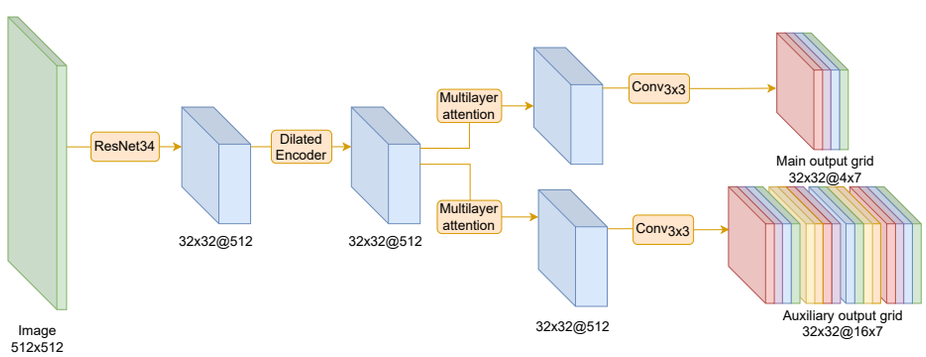

Figure 3.6: High level view of the detector architecture.

As shown in Figure 3.6, the model has a single feature level. This may reduce its accuracy on exact localization. However, exact localization is not needed in ship detection, and by using only a single feature level, energy consumption during inference is approximately halved compared to if feature pyramids are used [4].

To compute main loss, we use loss label assignment with overlapping responsibility. For the auxiliary head we use uniform matching [4], with four predictions per target. For both heads we use focal loss for positivity classification [12],

smooth L1 loss for positioning [6], SIoU loss for length and width, and ADV with $\mathcal{L}_{mse-w}$ for orientation.

The auxiliary heads loss function is equivalent to the main head, but instead of a one-to-one loss label assignment with overlapping responsibility, it a modified version of the many-to-one loss label assigned from YOLOF [4] based on positions instead of IoU. The loss of the two detection heads are summed, where the auxiliary heads output is multiplied with a small constant (0.1) to not receive too much attention.

3.7 Distance-AP (dAP)

Existing evaluation metrics for object detectors are far more strict than the user needs in the ship detection service. Therefore, we propose Distance-AP (dAP), a relaxed performance metric for ship detection. dAP is is based on mAP, but instead of an IoU threshold for counting predictions as true positives, dAP uses a threshold based on the distance between two objects center points.

$$Precision = \frac{TP}{TP + FP} \tag{3.25}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.26}$$

To compute dAP, we first need a definition of precision and recall. Given TP, FP, and FN counters for a test set, precision and recall are trivially computed according to Equation 3.25 and Equation 3.26. However, counting TP, FP, and FN is more complicated in object detection than in classification applications. Formally, for a single image and a single positivity classification threshold π, TP, FP, and FN are counted according to Algorithm 1. Put simply, a detection is counted as a true positive if it is within 50 pixels from a true object, and a true object that is not close to any detections is counted as a false negative. However, Algorithm 1 avoids some additional edge cases. For instance, each ground truth object can only be used for one true positive.

So, by using Algorithm 1 we can compute precision and recall at a given positivity classification threshold. However, Distance Average Precision (dAP) is not the precision at a single positivity classification threshold. Instead, dAP is defined as the area under the (interpolated) precision recall curve, as in Equation 3.27. $p(r)$ is the precision achieved at the same positivity threshold as when recall is r. $p'(r)$ is the interpolated precision, defined such that if a

Algorithm 1 Formal definition of how TP, FP and FN is counted in a single image, for a single positivity threshold π.

input $\mathcal{P}_\pi \leftarrow \{p^1, p^2, ...\}$ ▷ The set of predictions with $p_p > \pi$, limited to at most 100 elements
input $\mathcal{T} \leftarrow \{t^1, t^2, ...\}$ ▷ All true objects (targets) in the image
$\mathcal{T}_U \leftarrow \mathcal{T}$ ▷ Targets that have not been yet been assigned to a prediction
$TP \leftarrow 0$
$FP \leftarrow 0$
for $p \in \mathcal{P}_\pi$ **do** ▷ With descending positivity p_p
    $\mathcal{T}_M \leftarrow \{t \in \mathcal{T}_U : \|\underline{t_x} - \underline{p_x}\| < 50\}$ ▷ Subset of unassigned targets that are within a 50 pixels of the prediction p
    if $\mathcal{T}_m = \emptyset$ **then** ▷ No targets within 50 pixels. Increment FP
        $FP \leftarrow FP + 1$
    else ▷ Find the closest target, remove it from $\mathcal{T}_U$, and increment TP
        $TP \leftarrow TP + 1$
        $t \leftarrow \arg\min_{t' \in \mathcal{T}_M} \|\underline{t'_x} - \underline{p_x}\|$
        $\mathcal{T}_U \leftarrow \mathcal{T}_U - \{t\}$
    end if
end for
$FN \leftarrow |\mathcal{T}_U|$ ▷ FN: The number of missed / unassigned targets
return TP, FP, FN

higher recall $r^* \geq r$ results in higher precision, $p'(r)$ instead uses the value of the higher precision. Thus, $p'(r) \geq p(r)$ is monotonically decreasing.

$$\begin{aligned} dAP &= \int_0^1 p'(r)\, dr \\ p'(r) &= \max\{p(r^*) : r^* \in [r, 1]\} \end{aligned} \tag{3.27}$$

In addition to dAP (dAP), we also use F2. Using the same definition of precision, recall, TP, FP, and FN as dAP, we choose the positivity classification threshold π that maximizes F2. F2 is defined in Equation 3.28. Intuitively, F2 acts as a mid value between precision and recall, with more emphasis on recall.

$$F2 = \frac{5 \cdot p \cdot r}{4 \cdot p + r} \in [\min(p, r), \max(p, r)] \tag{3.28}$$

4

Evaluation and discussion

We first evaluate rotation regression, and answer the following questions. (1) Can a simple logistic regression method learn a very simple rotation regression task, despite the border discontinuity problem? (2) How does ADV compare to logistic regression, CSL and GCL, and how robust are they against different amounts of rotation label noise.

We then do three experiments with object detection tasks, to answer the following questions. (3) Can SIoU loss replace DIoU loss, and can position based label assignment replace IoU based label assignment, and how does this affect performance? This enables use of datasets where the true size of objects is not known. (4) How does inaccurate position label affect dAP in object detectors with different loss label assignment methods? We compare overlapping and non-overlapping label assignment. (5) How is dAP and F_2 on KSATs dataset affected by overlapping assignment compared to non-overlapping assignment, and how are the metrics affected by training the detector for joint detection *and* size regression?

4.1 Methodology

This section describes experimental setup that is common for the different experiments. Some model configurations vary for different experiments, but those differences are specified per experiment.

4.1.1 Rotation regression

For rotation regression experiments, we train a dedicated rotation regression network, that does not perform any other tasks such as object detection in combination with rotation regression. The network has a ResNet18 backbone with pre trained weights from torchvision followed by two fully connected layers, where the first has 500 neurons and the number of neurons in the output layer depends on which rotation regression method is used. For an isolated regression task, a (relatively shallow) ResNet18 backbone should be sufficient as it requires to extract less information that a full object detection architecture.

We train the models using the Adam [9] algorithm, based on stochastic gradient descent. We decrease the learning rate geometrically by 5% each epoch, starting at 10^{-3}.

4.1.2 Object detection

In the object detection experiments, we use a ResNet34 backbone with pre trained weights from torchvision, a default dilated encoder [4], multi layer attention, and auxiliary loss [24]. The different models vary in terms of loss label assignment method and bounding box localization loss, which is specified per experiment.

We train the models using the Adam [9] algorithm. We first do ten warm up epochs where the learning rate is increased linearly, and the remaining epochs we decrease the learning rate geometrically by 5% each epoch.

4.2 The rotation border discontinuity problem

This experiment demonstrates the border discontinuity problem in logistic regression for rotation modeling. We show that a very simple rotation regression problem can be easily learned by ADV and GCL, while logistic regression struggles. We use GCL without angle fine tuning regression and we omit CSL from this experiment, since the goal of this experiment is to demonstrate that logistic regression is insufficient for rotation regression, and not compare our ADV against GCL and CSL, since that is done in the next experiment.

The models are trained with randomly generated 512×512 images as shown in Figure 4.1. They contain a single line-segment, that has constant length, constant width, is always centered in the image center, and has a random

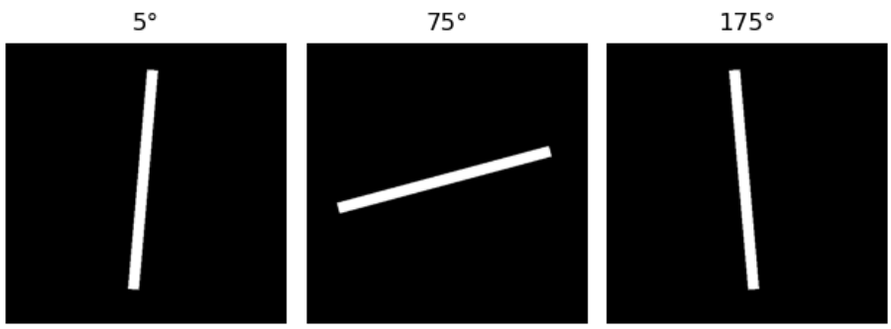

Figure 4.1: Example pictures of a randomly generated line segment. These images are used for test and training data in an experiment that demonstrates the border discontinuity problem. Logistic regression is unable to learn this regression task.

orientation in the $[0°, 180°)$ range since the front and back of the line segments are indistinguishable. The images vary only in terms of the lines orientation, and do not contain noise. Rotation labels are always accurate. The test set consists of the 180 images with $\theta \in \{0°, 1°, ..., 179°\}$, and the training set consists of $8k$ images with rotation $\theta \sim [0°, 180°)$. We train the models in 40 epochs, and sample 200 training images per epoch. Since training set and test set contains very similar images, the models do not need powerful generalization properties, and a large dataset is not required, and the task should be easy to learn. The models are trained as specified in Subsection 4.1.1.

The network is evaluated in terms of absolute difference between the inferred and true orientation. The absolute difference $d(a_1, a_2)$ between two angles is defined in Equation 4.1, and chooses the minimum of the direct distance and the wrapped distance. We compare the average distance over images in the whole range, and the average and maximum distance within smaller sub-intervals of 30°.

$$\begin{aligned} a_{min} &= \min(a_1, a_2) \\ a_{max} &= \max(a_1, a_2) \\ d_{direct} &= a_{max} - a_{min} \\ d_{wrapped} &= 180° - a_{max} + a_{min} \\ d(a_1, a_2) &= \min(d_{direct}, d_{wrapped}) \end{aligned} \tag{4.1}$$

Table 4.1 shows results from this experiment. Logistic regression struggles at border cases close to 0° and 180°, due to the border discontinuity problem, but performance is generally good in the rest of the images. The errors of GCL

Metric		Logistic Regression	GCL ($n = 3$)	ADV
0° – 180° (all)	Mean	5.3°	5.6°	1.04°
	Max	**75.4** °	11.25°	2.67°
0° – 30°	Mean	**11.9** °	6.2°	1.87°
	Max	**75.4** °	11.25°	2.67°
30° – 60°	Mean	1.8°	4.6°	1.19°
	Max	5.2°	11.25°	2.35°
60° – 90°	Mean	2.3°	5.9°	0.27°
	Max	3.9°	10.75°	0.63°
90° – 120°	Mean	5.2°	6.2°	1.39°
	Max	7.9°	11.25°	2.39°
120°–150°	Mean	2.5°	4.6°	1.06°
	Max	6.5°	11.25°	1.80°
150°–180°	Mean	**8.3** °	6.0°	0.46°
	Max	**13.2** °	10.75°	1.00°

Table 4.1: These results demonstrate the border discontinuity problem in logistic regression. It shows the average and maximum absolute difference between predicted and true rotation in the rotated line experiment. The results are grouped in intervals of 30° based on the angle of the true rotation. Border case values from logistic regression are highlighted as the inaccuracy is particularly high.

and ADV is not higher along the border cases, as they do not have a border discontinuity problem.

This is not a fair comparison of GCL and ADV, and does not show that ADV is superior to GCL. GCL was configured with few output bits resulting in course output granularity and was implemented without angle fine tuning. Changing this configuration would improve GCL performance significantly. Furthermore, the task in this experiment is very simple and does not represent their performance at more difficult tasks.

4.3 Rotation regression

This experiment aims to find which rotation regression methods are superior at different levels of training set label inaccuracy. Eg. which methods are superior when labels are accurate, and which are superior when labels are inaccurate. We compare ADV rotation regression against logistic regression and the state of the art CSL and GCL methods [26]. We use natural images instead of SAR images, since rotation regression is a general problem that is not restricted to ship rotation estimation in SAR imagery.

We use training sets where rotation labels contain different amounts of noise. First, we use these datasets to evaluate different hyper parameter configurations of the different models. Then, we compare the different models against each other, using the best performing hyper parameter configurations.

4.3.1 Experiment setup

We create a benchmark by augmenting the Stanford Dogs Dataset [8] (some examples in Figure 4.2), which has $22k$ images. We generate two rotation labels, referred to as the true and the known rotation labels, respectively r^i_{true} and r^i_{label}. The true label r^i_{true} is used for augmenting (rotating) images but is never presented to the models directly. The known label r^i_{label} is the noisy label used during training.

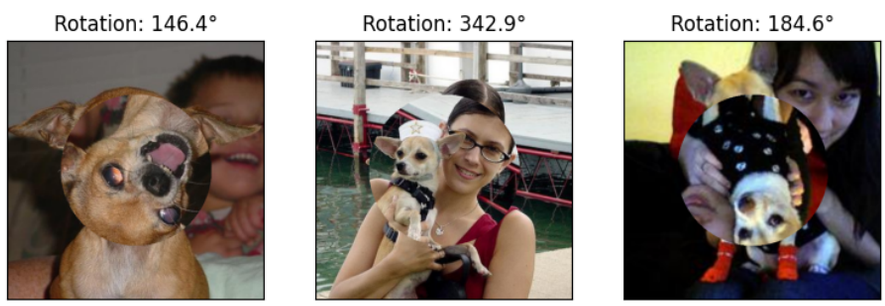

Figure 4.2: Example pictures from our rotation regression benchmark. Images are adapted from Stanford Dogs Dataset [8], but with rotated centers

The labels are generated according to Equations 4.2, 4.3 and 4.4. The Δ in Equation 4.3 is always $\Delta = 0°$ for test data, but for the training set we use $\Delta \in \{0°, 10°, 50°, 100°\}$. The modulo operation in Equation 4.4 makes negative rotations become corresponding positive rotations (eg. $-30°$ mod $360° = 330°$), since $r^i_{true} + \epsilon$ can become negative in some cases.

$$r^i_{true} \sim Unif(0°, 360°) \tag{4.2}$$

$$\epsilon \sim Unif(-\frac{\Delta}{2}, \frac{\Delta}{2}) \tag{4.3}$$

$$r^i_{label} = (r^i_{true} + \epsilon) \bmod 360° \tag{4.4}$$

The image rotation for a given image is done in two steps. (1) We ensure that the image is 512×512 pixels. For images whose height or width is smaller

than 512 pixels, we resize the image such that the smallest axis is 512 pixels. Then, we center-crop the image to 512×512 pixels. (2) With a 512×512 pixel image and a rotation label r^i_{true}, we perform the rotation as follows. We mask out a circle in the center of the image with a diameter of 300 pixels, and then rotate the circle counter clockwise according to r^i_{true}.

Each model is trained with different label noise $\Delta \in \{0°, 10°, 50°, 100°\}$. The models are trained as specified in Subsection 4.1.1.

4.3.2 ADV loss functions

First, we compare four different loss functions for ADV. Namely, we compare the $\mathcal{L}_{unit}$, $\mathcal{L}_{mse}$, $\mathcal{L}_{mse-w}$, and for $\mathcal{L}_{mse-w}$ we use two different configurations: $(\lambda_1, \lambda_2) = (0.5, 1.5)$ and $(\lambda_1, \lambda_2) = (0.25, 1.75)$. As seen in Figure 4.3, the $\mathcal{L}_{unit}$ loss function performs significantly worse than the rest. The $\mathcal{L}_{mse-w}$ loss function with $(\lambda_1, \lambda_2) = (0.5, 1.5)$ configuration is adapted as the default as it outperforms the rest at all metrics, with the lowest mean and median predicted inaccuracy, across all noise levels.

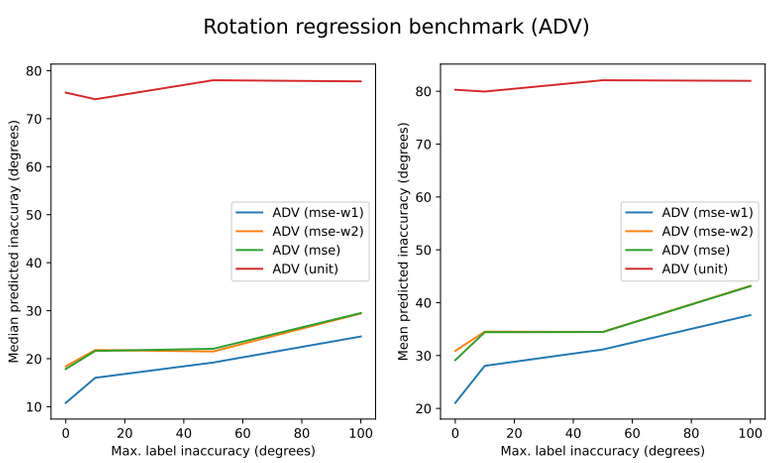

Figure 4.3: Performance of different ADV loss functions. Note the yellow and green overlapping plots. X axis is the amount of rotation label noise in the training set. Y axis is median (left) and mean (right) inaccuracy in predicted rotations achieved by the given model.

4.3.3 CSL hyper parameters

We compare CSL hyper parameter configurations. We use the triangle window function [26] with two different window radius configurations $\in \{0.1 \cdot 360°, 0.3 \cdot 360°\}$. We use three different output bit configurations $\in \{32, 128, 256\}$ cor-

responding to class interval lengths $\in \{11.25°, 2.81°, 1.41°\}$. The results in Figure 4.4 show no clearly superior configurations. For the final summary where we compare ADV against CSL and GCL, we include only the highest performing CSL configurations. Namely, we include both 32 bit configurations and the 128 bit with $0.3 \cdot 360°$ radius configuration, as they are superior at different noise levels.

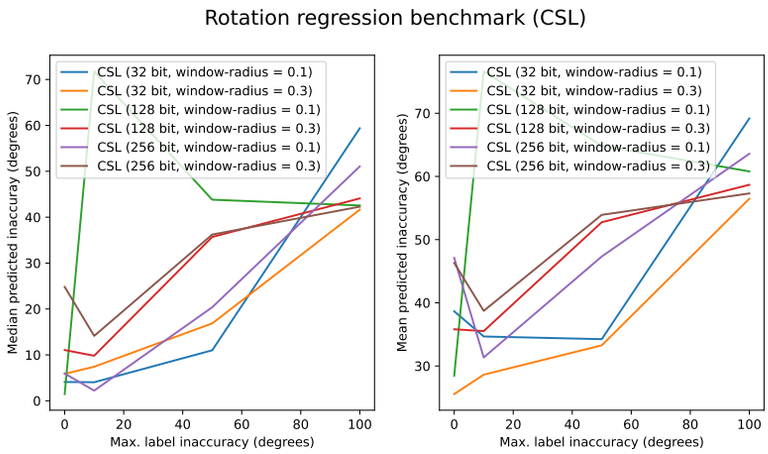

Figure 4.4: Performance of different CSL configurations. X axis is the amount of rotation label noise in the training set. Y axis is median (left) and mean (right) inaccuracy in predicted rotations achieved by the given model.

4.3.4 GCL hyper parameters

In Figure 4.5, we compare different GCL hyper parameter configurations: 3, 5 or 7 bits, corresponding to 8, 32 and 128 classes and rotation intervals of $45°, 11.25°$ and $2.81°$. Surprisingly, the 7 bit configuration performs better with little ($\Delta = 10°$) noise than no noise at all. None of the configurations have any clear benefits, so for the final comparisons of all models we just use the 3 bit configuration.

4.3.5 Summary

Figure 4.6 compares ADV, CSL, GCL, and logistic regression. GCL performs worse than CSL, consistent with results from Yang et al. [26]. ADV is overall superior in terms of mean inaccuracy across all noise levels. In terms of median inaccuracy, there is some variability in which model is superior depending on the noise level, where ADV is superior in high noise levels.

We found three key properties of data in rotation regression that affect models

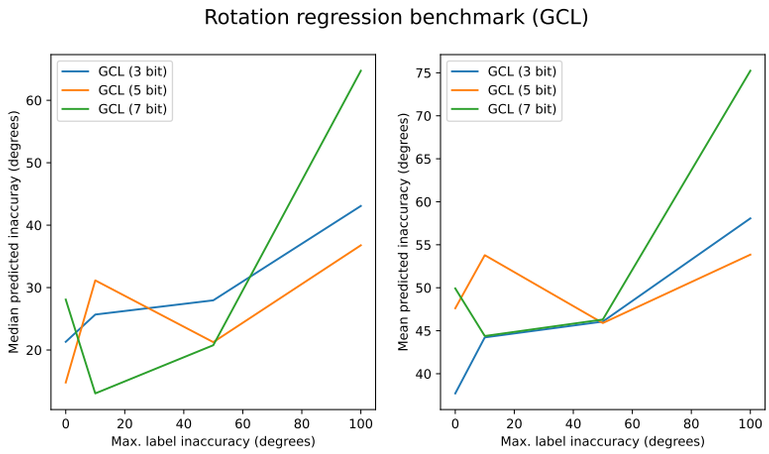

Figure 4.5: Performance of different GCL configurations. X axis is the amount of rotation label noise in the training set. Y axis is median (left) and mean (right) inaccuracy in predicted rotations achieved by the given model.

performance. (1) Rotation labels inaccuracy. (2) *Difficult data*, where it is difficult to determine rotation from the images. Difficult data indirectly causes rotation label inaccuracy, since it makes it difficult to produce accurate ground truth data, which holds for SAR images. However, rotation label inaccuracy can have more causes than this. For instance, the annotation work may have not focused on exact labeling. In this experiment we have compared various levels of rotation label inaccuracy, but we not considered various levels of *data difficulties*. Furthermore, we have only compared uniform label noise distributions, but have not considered normal distributed noise which may be more representable for some applications. Still, this experiment indicates that ADV regression is superior in applications with high label noise such as ship rotation in SAR images, but there is a possibility that CSL and GCL are superior for images where the rotation of objects is not easily visible.

We have not performed an exhaustive hyper parameter search for CSL. CSL can be configured in terms of the number of classes and the choice of window function. We have only used the triangle window function with different widths, whereas other window functions such as the Gaussian kernel have not been considered.

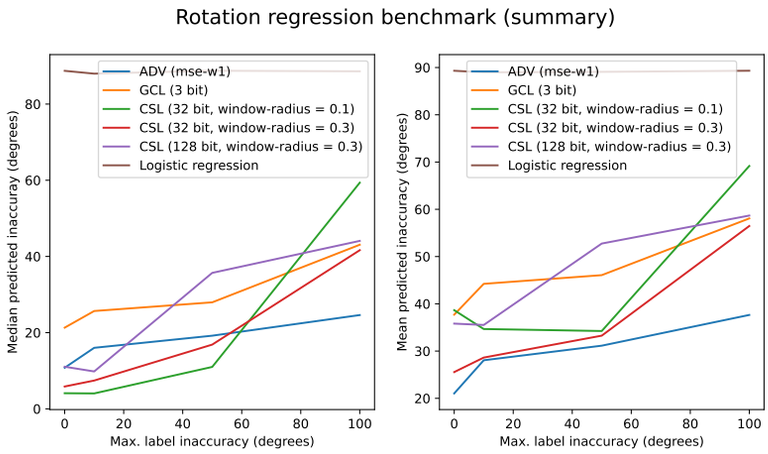

Figure 4.6: Performance of ADV, CSL, GCL and logistic regression for rotation regression. Note the brown curve for logistic regression at the top of the figure. Poor performing configurations of ADV, CSL, and GCL are omitted. X axis is the amount of rotation label noise in the training set. Y axis is median (left) and mean (right) inaccuracy in predicted rotations achieved by the given model.

4.4 Box regression loss and label assignment quality metric

In this experiment we answer two questions. First, can SIoU loss for size optimization plus smooth L1 loss for position optimization replace state of the art DIoU loss? Second, can loss label assignment be based entirely on objects center points, instead of IoU which is used in other object detectors? Position based label assignment and SIoU loss allow training a detector using a dataset with missing size annotations for some objects.

For all models, we use overlapping loss label assignment, but we use three different variations of the regression similarity metric Q^{Reg}. The different variations are defined in Equations 4.5, 4.6, and 4.7. The first is the default position based loss label assignment. The second one is IoU, which is used by DETR and DeFCN [3, 24]. The third one is a weighted sum of the default position based metric and SIoU, which has the benefit of including size estimates in the label assignment, as an alternative to IoU.

$$Q^{Reg}_{L_2}(p,t) = 1 - \frac{\|\underline{p_{xa}} - \underline{t_x}\|^2}{(\beta + \sqrt{2}\gamma)^2} \tag{4.5}$$

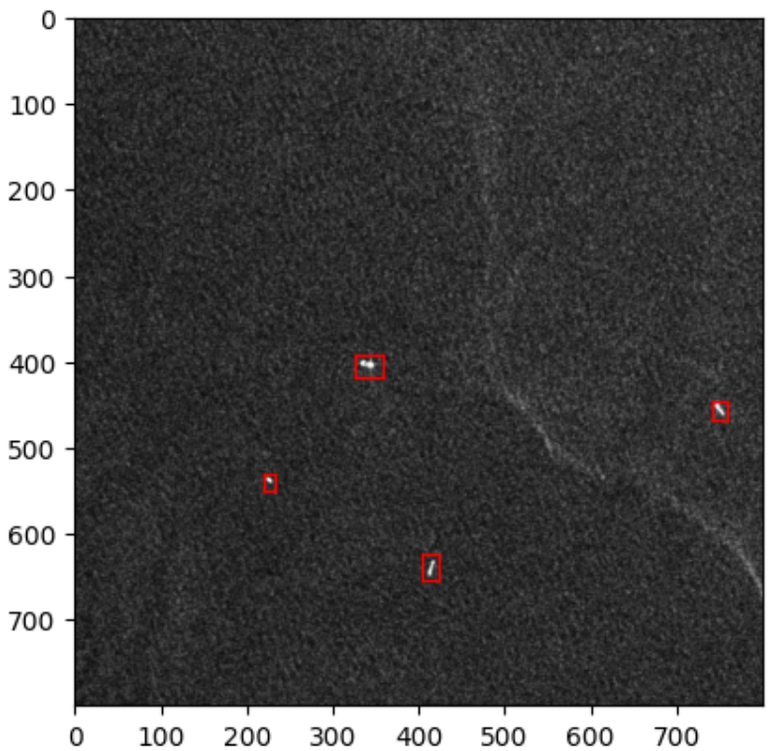

Figure 4.7: A sample image from the LS-SSDD dataset, with bounding box annotations. Objects is the LS-SSDD dataset are very small compared to other object detection datasets.

$$Q_{IoU}^{Reg}(p,t) = IoU(p,t) \tag{4.6}$$

$$Q_{L_2+SIoU}^{Reg}(p,t) = \frac{1}{2}\, Q_{L_2}^{Reg} + \frac{1}{2}\, SIoU(p,t) \tag{4.7}$$

The main performance metrics used in this experiment are dAP and F2. Precision and recall correspond to the classification threshold resulting in the highest F2. For each prediction plus target pair determined while computing dAP we compute SIoU, and we present the average SIoU across all matches. We include AP and AP50 for reference, but they are not important metrics in this experiment.

We train the models on the Large Scale SAR Ship Detection Dataset (LS-SSDD) [28], which has $6k$ training images and $3k$ test images. It has horizontal bounding box annotations for the size of ships as they appear in the images, but not the physical size of ships, and no annotations for rotations. Figure 4.7

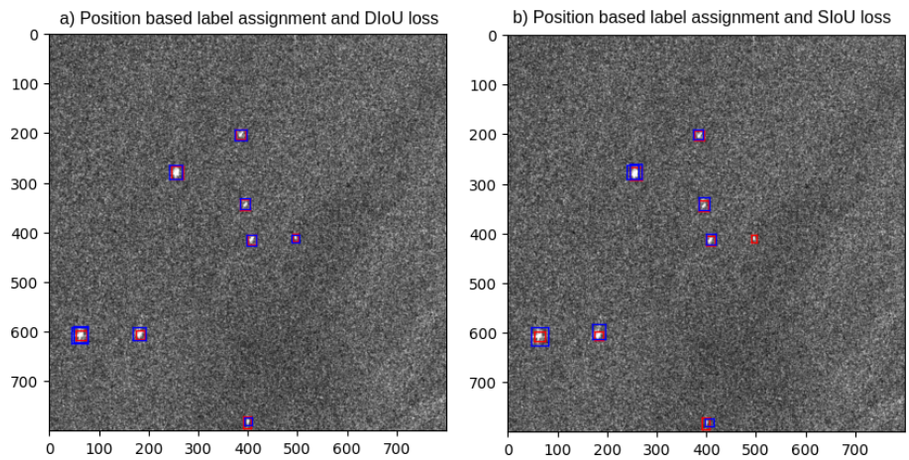

Figure 4.8: Test images from the LS-SSDD dataset. Both models use position based label assignment, but a) uses DIoU loss while b) uses SIoU loss as they were the best performing models. Red bounding boxes are ground truth. Blue bounding boxes are model predictions. Note the duplicate prediction at the bottom left ship in a), and at the top left ship in b).

shows a sample image from LS-SSDD.

This experiment is based on the methodology in Subsection 4.1.2. We use Mosaic data augmentation, random cropping, and self-adversarial training (SAT) [2]. The detectors are trained for 100 epochs, with 4 images per batch.

Configuration		**Performance**						
Assignment	Loss	dAP	AP	AP50	F2	p	r	SIoU
L_2	$SIoU + L_1$	0.85	0.13	0.46	**0.80**	0.65	0.85	**0.67**
L_2	$DIoU$	**0.86**	**0.20**	**0.60**	**0.80**	0.72	0.83	0.65
$L_2 + SIoU$	$SIoU + L_1$	0.83	0.15	0.50	0.78	0.71	0.80	0.66
$L_2 + SIoU$	$DIoU$	0.85	**0.20**	0.59	**0.80**	0.69	0.83	0.62
IoU	$SIoU + L_1$	0.79	0.09	0.37	0.75	0.62	0.79	0.62
IoU	$DIoU$	0.65	0.17	0.47	0.65	0.47	0.73	0.64

Table 4.2: Results from training models with different loss label assignment methods and different box localization losses on using the Large Scale SAR Ship Detection Dataset (LS-SSDD) [28]. The best measured performance on each performance metric is highlighted.

Table 4.2 shows the results of the experiment. In terms of the assignment quality metrics Q^{Reg} effect on dAP, R^{Reg}_{IoU} is clearly inferior. $Q^{Reg}_{L_2+SIoU}$ and $Q^{Reg}_{L_2}$ are similar, but $Q^{Reg}_{L_2}$ is slightly better.

$DIoU$ outperforms $SIoU + L_1$ loss significantly in terms of AP and AP50, while their dAP is similar. This implies that both loss functions result in similar detection performance, but DIoU loss is superior in terms of bounding box overlap. Both models have similar SIoU on the test set, implying similar size regression performance. Therefore, the inferior bounding box overlap must be a result of inferior position regression. So, the superior AP and AP50 shows that DIoU is a better center positioning loss, as it prioritizes positioning of small objects higher than large objects. Still, position regression accuracy is not a priority in ship detection, and the dAP and SIoU achieved on the test set are similar. So, SIoU loss is comparable to DIoU in the metrics that are important to ship detection users, but slightly worse in terms of the less important AP and AP50.

Output of models with the two different loss functions are shown in Figure 4.8. The model with DIoU loss has better overlap.

So, position based label assignment outperforms the other label assignment methods, and SIoU performs similar to DIoU. This means we can train a detector with a dataset where size annotations are missing from certain objects, without considerable decrease in performance. A $Q^{Reg}_{L_2}$ assignment (position based) and $SIoU + L_1$ is only 1% dAP below the best configuration. $DIoU$ loss results in better position estimates than $SIoU + L_1$, and although there may be room for improving $SIoU + L_1$ loss to better match DIoU, exact bounding box overlap is not a priority, so we omit this work.

These results complement some findings in YOLOF [4]. The large performance drop when using IoU as assignment metric is surprising since IoU is used for this purpose in (almost?) all object detectors. The combination of two properties of this experiment may likely have caused this. First, ships in LS-SSDD images are smaller than objects in most object detection datasets, with average height and width being 18 and 16 pixels respectively, as seen in Figure 4.7. Second, this detector architecture does not use feature pyramids, and feature pyramids improve mAP on small objects. IoU is zero when bounding boxes do not overlap, regardless of their distance and difference in size. Since we do not use feature pyramids and objects are small, the probability that any predictions overlap the true bounding boxes is relatively small. Hence, during training, all Q^{Reg}_{ijkl} scores will frequently be equal (zero) for a given true bounding box, although some predictions are better fit, resulting in bad label assignments. This problem with IoU when combined with single level detectors was pointed out in YOLOF [4], but instead of using an alternative match quality metric than IoU, they used random shifting augmentation of images during training to increase probability that some anchors overlap the ground truth objects. However, using a different match quality metric is simpler and may even speed up training as good label

assignments are determined at every iteration. So, our results indicate that the use of IoU instead of other similarity metrics in loss label assignment in most (if not all?) object detectors may be part of the reason for feature pyramids success, and not because feature pyramids provide more information in their fine grained feature maps.

4.5 The effect of inaccurate position labeling on different label assignments

This experiment demonstrates the effect inaccurate position labels have on object detectors with different loss label assignment methods. We compare three different position based loss label assignments. Namely, non-overlapping assignment, overlapping assignment with small regions (64px radius), and overlapping assignment with large regions (128px radius). We train these models using datasets with varying amounts of inaccuracy, and evaluate the models in terms of dAP.

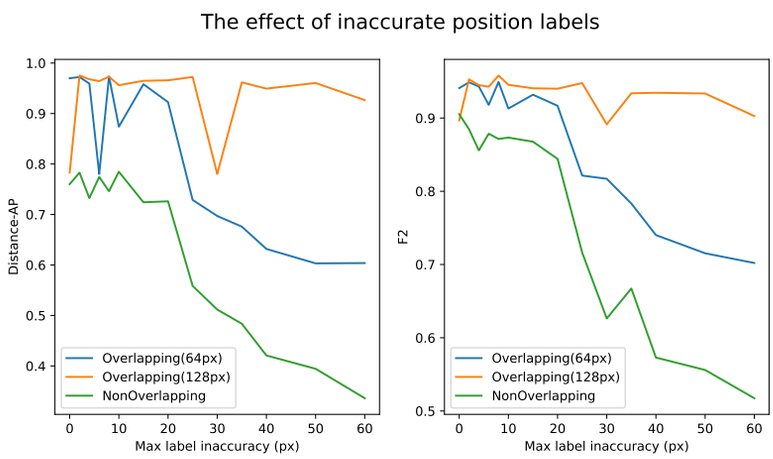

Figure 4.9: Evaluation results with synthetically generated datasets. We test two models with different loss label assignment methods (either overlapping or non-overlapping). The x-axis corresponds to the amount of noise in position labels (the position label is within a $\lambda = x$ pixel radius from the true center position). The y-axes correspond to highest achieved Distance-AP or F_2 for the given configuration. Distance-AP and F_2 scores may come from different training epochs.

For this experiment we randomly generate a datasets with properties that imitate SAR images containing ships. Figure 1.2 shows a sample image from this dataset. By generating data randomly, we can control the amount of

inaccuracy in position labels. The test set consists of 400 images with zero position label inaccuracy. The training set consists of $160k$ images.

To generate an image with max label inaccuracy of λ pixels, we first decide a random number of objects $n \sim Unif(\{0, 1, 2, 3\})$. For each object, we sample a random *label position* with $x, y \sim Unif(\lambda, 512 - \lambda)$ (anywhere in the 512×512 pixel image, except for a λ pixel padding along the edges), a random direction (angle) $\theta \sim Unif(0°, 360°)$, and a random distance $d \sim Unif(0, \lambda)$. We compute the *true center position* using an offset from the *label position* with direction θ and distance d. This ensures that both the true and labeled positions are contained in the image. However, a side effect is that no objects in the training set are labeled to be along the edges of the images, while the objects may be close to the edges.

We base the experiment on the methodology in Subsection 4.1.2, with an initial learning rate of $5 \cdot 10^{-6}$. We train the models in 100 epochs with 1600 unique images per epoch, 8 images per batch, and use self-adversarial training (SAT) [2] as the only data augmentation.

The results are in Figure 4.9 displays. As expected, the overlapping label assignment is more robust to label inaccuracy than non-overlapping assignment, and larger area of overlap increases robustness. However, we expected all models to have similar performance at $\lambda = 0$, whereas the plot shows that non-overlapping label assignment is considerably worse at all levels of inaccuracy. Still, the results demonstrate that position label noise negatively affects detection performance in certain models.

4.6 Ablation: The effect of overlapping label assignment and multi-task loss on KSATs data

In this experiment we aim to measure how dAP and F2 on KSATs data is affected by two mechanisms. First, how much does overlapping label assignment improve performance? Since KSATs data has noise in position labels, we expect this to improve dAP and F2. Second, is dAP and F2 affected by simultaneously training the network for the additional size regression subtask? We refer to as using *multi-task loss*. This may strengthen the shared feature maps as they are trained with more data, but it may also interfere in a negative way.

4.6.1 Experiment setup

We use initial learning rate of $1.5 \cdot 10^{-6}$. We use self-adversarial training (SAT) [2] as the only data augmentation. We sample $12.8k$ tiles per epoch where 66% are positive, with 16 tiles per batch, and stop training after 42 epochs as all models dAP has stopped progressing at this point.

4.6.2 Data collection

The dataset used in this experiment is collected as part of a real time service. First, the satellite records raw SAR data while positioned over a given region. Second, the SAR data is downloaded at an antenna ground station, preferably at a geographical location that is visible for the antenna shortly after the image was recorded to minimize waiting until the image is further processed. Third, the raw SAR data is used to construct the SAR image. Fourth, the complete SAR image is divided into smaller tiles of 512×512 pixels with 128 pixel overlap. Fifth, a machine learning model processes each tile in isolation to detect ships. Sixth, SAR specialists manually analyse the images, while verifying and correcting the automatic output produced by the machine learning model. This output is also correlated with external data sources such as AIS (descried below). Finally, the human verified detections and images are stored for re-training the machine learning models in the future.

The Automatic Identification System (AIS) is used to track maritime traffic. KSAT uses it as an external data source to verify detections as described above. Most ships are legally required to be carry AIS trackers that periodically transmit messages containing their position, size, and other descriptions. However, some ships such as small boats and military ships are not required to carry AIS trackers, and some ships may disable their AIS trackers to hide illegal activity. AIS data can virtually confirm the correctness of certain ship detections, and provides exact descriptions of ships' physical size, and may in addition provide descriptions of ships' direction and speed. However, there are some caveats. (1) The absence of an AIS correlation for a given detection may be a result of either a false detection or that the ship has disabled their AIS tracker. Hence, we do not have ground truth for hard negatives (entities at sea that look like ships but are not actually ships). (2) AIS messages from ships in a given SAR image may not have been sent at the exact moment the SAR image was recorded. Since direction and speed is variable, these estimates may be highly inaccurate in certain cases. (3) The length and width information in AIS messages is exact in most cases, but is not available for all ships. (4) With multiple ships in the same area, a correct correlation is difficult, and the size annotations of two ships may be swapped in the training set.

4.6.3 Data description and statistics

We use an internal KSAT dataset of Sentinel 1 images. The images have 10 meter pixel spacing. The entire data set contains 147k tiles of 512×512 pixels (5×5 kilometers). There are 13k positive tiles that contain ships. 77% of the positive tiles contain one ship, 12% contain two ships, 4% contain three ships, and the remaining 7% contain up to between 4 and 50 ships. 99% of ships in this dataset have length and width labels, while none of the ships have known orientation. The amount of inaccuracy in the position labels is unknown. All tiles used are offshore images. The dataset is split into one training set and one test set.

The test set initially contains 14k tiles where 8.8% (1.2k) are positive, but we discard 95% of the negative tiles to speed up evaluation, since most negative offshore images are fairly similar. The resulting test set contains 2.0k tiles where 66% (1.2k) are positive.

The training set contains $132k$ tiles where 8.7% ($11.8k$) are positive. However, when we train the models, during each epoch, we sample 7920 positive and 3960 negative tiles, resulting in $12.8k$ tiles in total where 66% are positive. This way, the model is trained with a more efficient data balance of 66% positives instead of 8.7%. Alternatively, we could sub sample the training set the same way as the test set, by simply discarding 95% of the negative tiles, which would result in the same data balance. However, this would reduce the variety in negatives.

4.6.4 Results

Configuration				
Overlapping label assignment	✓	✓		
Multi-task loss	✓		✓	
Performance				
dAP (%)	81	**83**	79	77
F_2 (%)	78	**79**	77	76
Precision (%)	66	64	57	57
Recall (%)	82	83	84	82

Table 4.3: Results from ablation experiments on KSATs dataset. Best achieved dAP and F_2 is highlighted. The shown precision and recall correspond to the highest achieved F_2 score.

The results are shown in Table 4.3. Overlapping label assignment consistently improves dAP and F_2. Multi-task loss decreases performance when paired with overlapping label assignment, but increases performance when paired

with non-overlapping label assignment. The precision to recall trade off that maximized F_2 in each model is shown. All models achieve similar recall, while precision varies significantly, mainly depending on the use of overlapping or non-overlapping label assignment.

The fact that multi-task loss is either positive or negative for dAP and F_2 depending on what loss label assignment is used, is surprising. It is difficult to explain this strange behaviour, but evidently, the regularizing effect multi-task loss has on the detection task is negative when paired with overlapping label assignment. Increasing the capacity of the backbone (eg. by replacing ResNet34 with ResNet50) may improve performance for the overlapping label assignment + multi-task loss configuration. Still, the performance drop when introducing multi-task loss is only 2% dAP which is not severe, and the practical benefits of having a completely end-to-end trainable model for both detection and additional descriptions out weighs this slight performance drop.

5

Conclusion

5.1 Summary

We have designed a ship detection model that solves four key sub problems. First, we introduced SIoU for object size loss and a loss label assignment based entirely on predicted positions. These mechanisms allow training an object detector where size and rotation annotations are missing from certain objects, which is important for ship detection in SAR imagery, where access to ground truth is limited. We show that these solutions have good performance, as SIoU results in similar dAP to DIoU, and position based assignment results in higher dAP than IoU based assignment, on LS-SSDD [28]. However, we did find that SIoU loss results in worse detection positions than DIoU, which is not a priority in ship detection and does not reduce dAP.

Second, we show that the loss function used in angular direction vector (ADV) regression influences performance significantly. Our experiments indicate that ADV with our improved loss function may be able to outperform state of the art CSL and GCL [26], and is especially more robust against noise in ground truth rotation labels. This is important for ship direction regression in low resolution SAR images, where training data labeling is based on external data sources such as AIS that only provide a rough estimate of the ship direction.

Third, we demonstrate that inaccurately labeled object positions negatively affect performance of object detectors with certain loss label assignment methods. Furthermore, we propose a loss label assignment method where the predictions

have large overlapping regions of responsibility, which tolerates large amounts of inaccuracy in position labels. This is important for organizations that perform data annotation in a time critical manner where exact object localization is not a priority.

Fourth, we propose Distance-AP (dAP), an alternative performance metric for object detection applications, where exact localization is not a high priority, such as maritime surveillance in aerial images. This provides three key benefits. First, this allows us to choose detector architectures that create the most value to end users. Second, this relaxed metric may allow designing a simpler and cheaper architecture in future work, as it is less strict in terms of exact localization. This may significantly reduce power consumption. Third, a relaxed metric may allow rapid engineering as it is not distracted by small localization errors such as AP^{50}. These three benefits hold for most machine learning applications, and for this reason we want to encourage machine learning researchers to be more aware of how their research will be used in the future, when choosing evaluation metrics.

Combined, these mechanisms enable end-to-end trainable ship detection with zero post-processing, with datasets where some size and rotation labels are missing, and rotation labels and position labels are inaccurate. Compared to KSATs previous solution, this has two main benefits. First, the proposed solution should be better at discriminating nearby ships. Second as it is only a single model it should be easier to to train, and in general easier to manage in a production environment. Ship detection service providers such as KSAT can therefore adapt the solutions described in this thesis, but should first validate its performance against their current solution. Source code is available at https://github.com/matill/Ship-detection

5.2 Limitations

Our experiments indicate that ADV with our improved loss function can achieve state of the art performance in rotation regression. However, additional evaluation is needed. We can (1) compare with more configurations of CSL (eg. different window functions), (2) compare the models on rotation regression tasks with varying difficulty, such as ship direction estimation in low resolution images, (3) compare the models at different types of label noise distribution such as Gaussian noise. Nevertheless, ADV is much easier to use than CSL, as it is easy to implement, requires no hyper parameter tuning (since the best ADV configuration was superior to the other ADV configurations at all noise levels), and is more computationally efficient and requires fewer model parameters. These benefits alone may outweigh potential performance benefits with a fine

tuned CSL model.

Position based loss label assignment has shown promising results, demonstrating significantly higher dAP than the alternatives such as IoU based assignment. However, we have only compared these methods on LS-SSDD [28] where bounding boxes are very small compared to most other datasets. IoU is zero for all non-overlapping bounding boxes, regardless of how close they are, and how similar their shapes are, and for this reason, IoU based label assignment struggles with small ground truth boxes. For datasets with larger bounding boxes such as HRSID [25], IoU based label assignment will likely achieve more similar dAP compared to position based. Still, distance based assignment was not proposed to outperform IoU based assignment, but to enable the use of datasets with unknown sizes. On a side note, given the observation that IoU based loss label assignment is inferior for small objects, we suggest that the performance of single feature level (no feature pyramid) object detectors such as DETR and YOLOF [3, 4] can increase performance significantly by employing a different regression similarity metric than IoU (eg. DIoU).

We have demonstrated similar dAP sores for both SIoU and DIoU loss. DIoU has demonstrated more accurate object positioning, and therefore AP and AP50, which is not a priority in ship detection.

Overlapping label assignment has shown to be consistently superior to non-overlapping assignment, and is robust against position label noise.

Some design choices have not been verified. First, we have not confirmed the effect of the multi-layer attention module. Second, we have not confirmed the effect of discarding feature pyramids in our work. We assume that since the application has relaxed positioning requirements, that feature pyramids will not provide considerable improvements, but this has not been verified.

5.3 Future work

The following topics can be further researched. (1) More extensive experiments can conclude if ADV with our improved loss function achieves state of the art. (2) Investigate the impact of the multi layer attention module. How does it compare to 3D max filtering [24] and other attention modules for enabling end-to-end duplicate suppression. (3) Investigate if discarding feature pyramids from the architecture reduced dAP. We could employ an end-to-end trainable architecture with feature pyramids such as DeFCN [24] and see if dAP is increased, or if only AP and AP50 are increased. (4) Investigate how well position based loss label assignment compares to IoU based loss label assignment for

other ship detection datasets and other object detection applications such as pedestrian detection. (5) We observed great performance increase with position based loss label assignment. We suggest that similar performance increase can be seen in other single level object detection architectures such as DETR and YOLOF [3, 4] when working with small objects, by simply replacing IoU in loss label assignment with something different, such as DIoU. This may be an important step towards discarding expensive feature pyramids from object detection models.

References

[1] European Space Agency. *Maritime Monitoring?* `https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-1-sar/applications/maritime-monitoring`. Accessed 2022-05-20. 2017.

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection." In: *arXiv preprint arXiv:2004.10934* (2020).

[3] Nicolas Carion et al. "End-to-end object detection with transformers." In: *European Conference on Computer Vision*. Springer. 2020, pp. 213–229.

[4] Qiang Chen et al. "You only look one-level feature." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 13039–13048.

[5] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of machine learning research* 12.7 (2011).

[6] Ross Girshick. "Fast R-CNN." In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[7] Kaiming He et al. "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[8] Aditya Khosla et al. "Novel dataset for fine-grained image categorization: Stanford dogs." In: *Proc. CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*. Vol. 2. 1. Citeseer. 2011.

[9] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980* (2014).

[10] Harold W Kuhn. "The Hungarian method for the assignment problem." In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.

[11] Tsung-Yi Lin et al. "Feature pyramid networks for object detection." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[12] Tsung-Yi Lin et al. "Focal loss for dense object detection." In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[13] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context." In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[14] Lei Liu, Zongxu Pan, and Bin Lei. "Learning a rotation invariant detector with rotatable bounding box." In: *arXiv preprint arXiv:1711.09405* (2017).

[15] NASA. *What is Synthetic Aperture Radar?* `https://earthdata.nasa.gov/learn/backgrounders/what-is-sar`. Accessed 2022-05-06. 2020.

[16] Joseph Redmon and Ali Farhadi. "YOLO9000: better, faster, stronger." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

[17] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement." In: *arXiv preprint arXiv:1804.02767* (2018).

[18] Joseph Redmon et al. "You only look once: Unified, real-time object detection." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[19] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." In: *Advances in neural information processing systems* 28 (2015), pp. 91–99.

[20] Hamid Rezatofighi et al. "Generalized intersection over union: A metric and a loss for bounding box regression." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 658–666.

[21] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." In: *arXiv preprint arXiv:1409.1556* (2014).

[22] Zhongzhen Sun et al. "BiFA-YOLO: A Novel YOLO-Based Method for Arbitrary-Oriented Ship Detection in High-Resolution SAR Images." In: *Remote Sensing* 13.21 (2021), p. 4209.

[23] Ashish Vaswani et al. "Attention is all you need." In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[24] Jianfeng Wang et al. "End-to-end object detection with fully convolutional network." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15849–15858.

[25] Shunjun Wei et al. "HRSID: A high-resolution SAR images dataset for ship detection and instance segmentation." In: *Ieee Access* 8 (2020), pp. 120234–120254.

[26] Xue Yang, Junchi Yan, and Tao He. "On the arbitrary-oriented object detection: Classification based approaches revisited." In: *arXiv preprint arXiv:2003.05597* (2020).

[27] Tianwen Zhang et al. "HOG-ShipCLSNet: A novel deep learning network with hog feature fusion for SAR ship classification." In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2021), pp. 1–22.

[28] Tianwen Zhang et al. "LS-SSDD-v1. 0: A deep learning dataset dedicated to small ship detection from large-scale Sentinel-1 SAR images." In: *Remote Sensing* 12.18 (2020), p. 2997.

[29] Xin Zhang et al. “Multitask Learning for Ship Detection from Synthetic Aperture Radar Images.” In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 14 (2021), pp. 8048–8062.

[30] Zhaohui Zheng et al. “Distance-IoU loss: Faster and better learning for bounding box regression.” In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 12993–13000.

[31] Yi Zhou et al. “On the Continuity of Rotation Representations in Neural Networks.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.