Optimizing ship detection efficiency in SAR images

Arthur Van Meerbeeck[*]
arthurvanmeerbeeck@gmail.com

Jordy Van Landeghem
jordy.vanlandeghem@kuleuven.be

Ruben Cartuyvels
ruben.cartuyvels@kuleuven.be

Marie-Francine Moens
sien.moens@kuleuven.be

Department of Computer Science
KU Leuven, 3001 Leuven (Belgium)

Abstract

The detection and prevention of illegal fishing is critical to maintaining a healthy and functional ecosystem. Recent research on ship detection in satellite imagery has focused exclusively on performance improvements, disregarding detection efficiency. However, the speed and compute cost of vessel detection are essential for a timely intervention to prevent illegal fishing. Therefore, we investigated optimization methods that lower detection time and cost with minimal performance loss. We trained an object detection model based on a convolutional neural network (CNN) using a dataset of satellite images. Then, we designed two efficiency optimizations that can be applied to the base CNN or any other base model. The optimizations consist of a fast, cheap classification model and a statistical algorithm. The integration of the optimizations with the object detection model leads to a trade-off between speed and performance. We studied the trade-off using metrics that give different weight to execution time and performance. We show that by using a classification model the average precision of the detection model can be approximated to 99.5% in ±44% of the time or to 92.7% in ±25% of the time.

1   Introduction

Biodiversity conservation and climate change are significant and current global issues. A major contemporary problem in biodiversity conservation is overfishing, which often results from Illegal, Unreported and Unregulated (IUU) fishing. It is estimated that IUU fishing is responsible for 30% of all fishing in the world, with an estimated economic loss of up to $23 billion per year [1]. Not only do illegal activities threaten marine ecosystems, but the resistance of many marine species to climate change is compromised by overfishing, which increases the vulnerability of marine fisheries production to ocean warming. Furthermore, ongoing warming will impede efforts to recover overfished populations [2]. Satellite imagery can be used to detect vessels on sea and to determine if these vessels are conducting illegal activities.

Synthetic Aperture Radar (SAR) images are created using satellites that emit radio waves towards the earth and capture the waves reflected back by objects [3]. When a ship is detected on these images, it can be determined whether a corresponding Automatic Identification Signal (AIS), which all ships should transmit every few seconds, can be linked to that ship. If this is not the case, the ship could possibly be conducting illegal activities [4]. Some satellites can image about 7 million square miles per day and thus various machine learning algorithms are used nowadays to detect ships in these SAR

---

[*]Corresponding author.

images [5]. However, current models are not yet able to process the SAR data both efficiently and with high accuracy, which results in high resource costs and which could impede their use in real-time systems. Therefore, there is a need for models that can quickly and accurately detect ships in SAR images o allow the Coast Guard to take timely and efficient action when the models detect illegal fishing [6]. The speed and performance of machine learning models are often negatively correlated and, therefore, it is necessary to make an appropriate trade-off for this.

We investigated optimizations for machine learning models, agnostic as to how the base model functions, that detect ships in SAR images to make the best possible trade-off. These optimizations consist of a fast classification model and a statistical algorithm. Their goal is to efficiently determine as a first step whether at least one ship is present in an SAR image before applying the expensive base detection model. Figure 1 gives an overview.
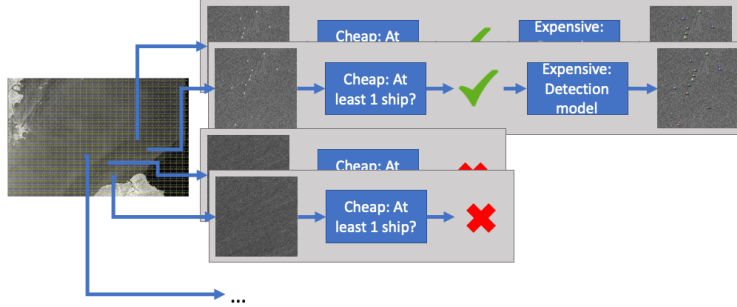

Figure 1: Overview of the optimizations and the detection model. If in a first step, a fast algorithm predicts that no ship is present, the expensive detection model is skipped.

2 Related Work

The launch of the first SAR satellite in the United States in 1978 led to the emergence of several SAR ship detection methods [7]. These initial methods are based on traditional detection methods that manually design features [8, 9, 10]. These traditional methods proved too slow, required complex manual work and were inaccurate [11]. The rise of neural networks led to the use of advanced object detection techniques due to their higher accuracy and lesser need for human intervention. This was also the case for the SAR ship detection community: the proliferation of SAR data [12] in recent years has contributed to the increase in the use of deep learning (DL) in SAR ship detection [13, 14, 15]. In these models, the traditional features (such as gray level, HOG, etc.) are replaced by features produced by CNNs. The SAR dataset used in this study is the Large-Scale SAR Ship Detection Dataset-v1.0 (LSSSDD) [16]. The dataset contains 15 SAR images from the Sentinel-1 satellite [2], with a size of 16000×24000 pixels, divided into 900 sub-images of size 800×800.

3 Efficiency optimizations

A large percentage of the SAR sub-images do not contain ships and so time is lost by performing object detection on them. The goal of our optimizations is to perform detection on as few images as possible with minimal loss of performance. To achieve this, we use two models that efficiently determine for each sub-image whether a ship is present or not, before applying the more expensive ship detection model that locates each ship. If the models conclude that no ship is present on an image, the image is not fed to the detection model. A trade-off has to be made between inference time and performance, by choosing the positive classification threshold of the efficient first-stage classification.

Fast classification with small CNN This optimization implements a binary classification model with a CNN. The CNN produces a score s between -1 and 1, with a score lower than the chosen threshold t^{clf} indicating that at least one ship is present, a higher score that no ship is present. Since the goal of optimization is saving time, the small and efficient MobileNetV2 [17] architecture was

---

[2]`https://scihub.copernicus.eu`

used. I is a SAR sub-image of size 800×800.

$$f_{\text{clf}}(I, t^{\text{clf}}) = \text{CNN}(I) = s^{\text{clf}} \begin{cases} I \text{ contains a ship} & \text{if } s^{\text{clf}} \leq t^{\text{clf}} \\ I \text{ does not contain a ship} & \text{if } s^{\text{clf}} > t^{\text{clf}} \end{cases} \quad (1)$$

The MobileNetV2 model minimizes inference time with minimal loss of performance by using a new type of layer module. Initially, weights trained on the ImageNet dataset [18] are loaded in. Then, the final layers of the model are retrained on the LSSSDD until the validation loss converges.

Correlation in ship presence between neighboring images   Another way to avoid running object detection is to predict ship presence in a given sub-image based on the presence of ships in neighboring sub-images. In this optimization, object detection is first conducted on a subset of the sub-images. The images on which object detection is initially performed are chosen via two patterns shown in Figure 5. The optimization with the checkers and α pattern are respectively noted as $f_{\text{cor-checkers}}$ and $f_{\text{cor-}\alpha}$. The patterns check respectively 50% and 25% of all the sub-images in the initial step. Afterward, using that information, the correlation algorithm predicts for each of the remaining sub-images whether a ship is present. This is done by taking the weighted average of a ship presence indicator of neighboring images on which ship detection was conducted. If s^{cor} is greater than a chosen threshold t^{cor}, the image is fed to the object detection model. The time to calculate s_{cor} is negligible in comparison to the time necessary to perform object detection.

$$f_{\text{cor}}(I, K, w, t^{\text{cor}}) = \sum_{j \leq K} \sum_{i \in N_j} w_j \mathbb{1}_i = s^{\text{cor}} \begin{cases} I \text{ contains a ship} & \text{if } s^{\text{cor}} \geq t^{\text{cor}} \\ I \text{ does not contain a ship} & \text{if } s^{\text{cor}} < t^{\text{cor}} \end{cases} \quad (2)$$

In this equation, N_j is the set of neighbours that are j tiles away, w_j is the weight given to all neighboring tiles that are j tiles away, and $\mathbb{1}_i$ indicates whether neighbor i contains at least one ship or not. Neighbours that are 1 tile away get the largest weight w_1, neighbours that are 2 tiles away a smaller weight, and so on.

4   Experiments

An overview of the split of the data set is given in Table 2. We used a Faster R-CNN model with a ResNet-50 backbone to detect ships in the SAR images [19, 20]. We start training from pre-trained weights for object detection on MSCOCO [21], and retrain on the LSSSDD dataset with use of the Detectron2 library[3] until the validation loss converges.

4.1   Results

Model	Total Time (s)	AP
Baseline	810.84	0.711
$f_{\text{clf}}(I, t_1^{\text{clf}})$	283.93	0.693
$f_{\text{clf}}(I, t_2^{\text{clf}})$	362.70	0.706
$f_{\text{cor-}\alpha}(I, K_1, w_1, t_1^{\text{cor}})$	405.41	0.616
$f_{\text{cor-}\alpha}(I, K_2, w_2, t_2^{\text{cor}}) + f_{\text{clf}}(I, t_3^{\text{clf}})$	202.71	0.638

Table 1: Total time and average precision for five SAR images.

Table 1 shows that the baseline Faster R-CNN model achieves an Average Precision (AP) of 0.711 in a total of 811s (including load and detection time). All optimizations drastically reduce the total time, while losing limited AP. When applying the optimizations, time savings can be weighed against performance retention. For this purpose, the F_β score is calculated between the AP of the model and the time savings compared to the baseline model. The time gain is equal to 1 - the relative time (RT) compared to the baseline model (RT = $\frac{\text{Total time with optimization}}{\text{Total time of baseline}}$). The smaller β, the more weight the AP gets in the F_β score.

$$F_\beta = (1 + \beta^2) \cdot \frac{AP \cdot (1 - RT)}{\beta^2 \cdot AP + (1 - RT)} \quad (3)$$

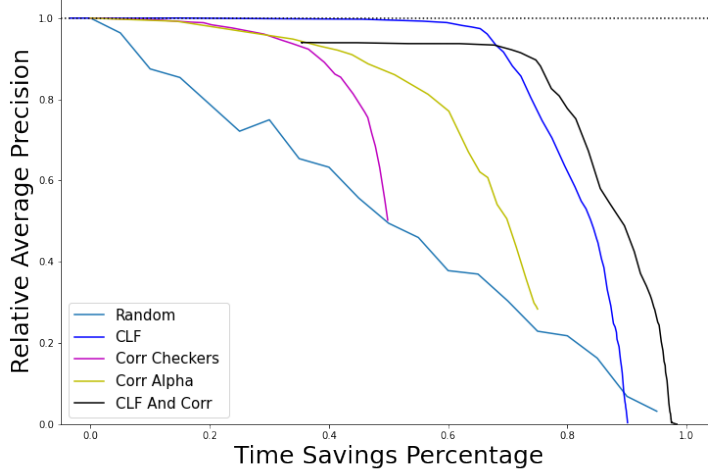---

[3]https://github.com/facebookresearch/detectron2

Figure 2: Relative AP and percentage of time saving for the optimizations and an algorithm that deletes random sub-images from the testset.

Classification model A fast CNN first predicts which images contain a ship. The classification time of all sub-images of one SAR image is approximately 6s. The best results are given in Table 1. With $t_1^{\text{clf}} = 0$ the model achieves an AP of 0.693 in 35% of the time of the baseline model, resulting in a F_1 score of 0.673. This optimization produces the highest overall $F_{0.25}$ score (AP has higher weight) with $t_2^{\text{clf}} = 0.2$, with $F_{0.25} = 0.695$. This model achieves an AP of 0.706 in 44% of the time.

Correlation algorithm This optimization determines whether to run detection on a sub-image based on the presence of ships in neighboring images, following eq. (2). The calculation of the correlation score is negligible compared to the object detection time of the sub-images. The highest F_1 score achieved is equal to 0.553 with $K = 2, w_1 = 1, w_2 = 0.1$ and $t_1^{\text{cor}} = 0.4375$ as seen in 1. The highest $F_{0.25}$ score is equal to 0.637. This optimization, therefore, does not surpass the F_β of the classification optimization.

Combination Both optimizations are combined by first dividing the SAR sub-images according to the chosen correlation pattern and, before performing detection, first predicting using the fast CNN whether the sub-image contains a ship. As a result, the detection model only performs detection on the sub-images that are classified as containing a ship by both optimizations. This combination produces a F_1 score (time savings and AP have equal weight) of 0.688, higher than the individual classification optimization. It achieves an AP of 0.638 in only 25% of the time of the baseline model. This is shown in Table 1 were $K = 2, w_1 = 1, w_2 = 0.5, t_2^{\text{cor}} = 0.25$ and $t_3^{\text{clf}} = 0$.

Figure 2 shows the time savings vs. the AP as a curve per optimization, with the points on the curve corresponding to different thresholds t. As expected from the F_β scores, the classification optimization retains performance best (curve with highest y-coordinates). Both optimizations perform better than an algorithm that randomly removes images from the test set. It is also visible that the combination of both optimizations outperforms individual optimizations in terms of time savings (curve with highest x-coordinates).

Conclusion In this paper, we studied vessel detection in SAR images to counter illegal fishing. We introduced and tested two optimization techniques as a first-stage filter to make the detection as efficient as possible. From the experiments, we concluded that a classification model based on MobileNetV2 yields the best results when more weight is given to performance retention: approximating the baseline AP to 99.5% in 44% of the time. If the reader gives equal weight to time savings and performance, the combination of the correlation- and classification-optimization is best, approximating the baseline AP to 92.7%, in only 25% of the time. We hope this study makes clear the importance of detection efficiency, and paves the way for more efficiency improvements.

4

References

[1] David J. Agnew, John Pearce, Ganapathiraju Pramod, Tom Peatman, Reg Watson, John R. Beddington, and Tony J. Pitcher. Estimating the worldwide extent of illegal fishing. *PLOS ONE*, 4:1–8, 02 2009.

[2] Christopher M. Free, James T. Thorson, Malin L. Pinsky, Kiva L. Oken, John Wiedenmann, and Olaf P. Jensen. Impacts of historical warming on marine fisheries production. *Science*, 363(6430):979–983, 2019.

[3] Alberto Moreira, Pau Prats-Iraola, Marwan Younis, Gerhard Krieger, Irena Hajnsek, and Konstantinos P. Papathanassiou. A tutorial on synthetic aperture radar. *IEEE Geoscience and Remote Sensing Magazine*, 1(1):6–43, 2013.

[4] Fábio Manzoni Vieira, François Vincent, Jean-Yves Tourneret, David Bonacci, Marc Spigai, Marie Ansart, and Jacques Richard. Ship detection using sar and ais raw data for maritime surveillance. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pages 2081–2085, 2016.

[5] Jordan Steward. This AI is set to help stop illegal fishing, here's how. *Skylight global news: xview3*, January 2022.

[6] Tianwen Zhang and Xiaoling Zhang. High-speed ship detection in sar images based on a grid convolutional neural network. *Remote Sensing*, 11(10), 2019.

[7] Jerzy Stefanowicz, Irfan Ali, and Simon Andersson. Current trends in ship detection in single polarization synthetic aperture radar imagery. In Ryszard S. Romaniuk and Maciej Linczuk, editors, *Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments 2020*, volume 11581, pages 66 – 77. International Society for Optics and Photonics, SPIE, 2020.

[8] A. Banerjee, P. Burlina, and R. Chellappa. Adaptive target detection in foliage-penetrating SAR images using alpha-stable models. *IEEE Transactions on Image Processing*, 8(12):1823–1831, 1999.

[9] Ming-Dian Li, Xing-Chao Cui, and Si-Wei Chen. Adaptive superpixel-level cfar detector for sar inshore dense ship detection. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2022.

[10] Pasquale Iervolino and Raffaella Guida. A novel ship detector based on the generalized-likelihood ratio test for sar imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(8):3616–3630, 2017.

[11] Tianwen Zhang, Xiaoling Zhang, Jianwei Li, Xiaowo Xu, Baoyou Wang, Xu Zhan, Yanqin Xu, Xiao Ke, Tianjiao Zeng, Hao Su, Israr Ahmad, Dece Pan, Chang Liu, Yue Zhou, SHI JUN, and Shunjun Wei. Sar ship detection dataset (ssdd): Official release and comprehensive data analysis. *Remote Sensing*, 13:3690, 09 2021.

[12] Fernando Paolo, Tsu-ting Tim Lin, Ritwik Gupta, Bryce Goodman, Nirav Patel, Daniel Kuster, David Kroodsma, and Jared Dunnmon. xview3-sar: Detecting dark fishing activity using synthetic aperture imagery, 2022.

[13] Jiao Jiao, Yue Zhang, Hao Sun, Xue Yang, Xun Gao, Wen Hong, Kun Fu, and Xian Sun. A densely connected end-to-end neural network for multiscale and multiscene SAR ship detection. *IEEE Access*, 6:20881–20892, 2018.

[14] Quanzhi An, Zongxu Pan, Lei Liu, and Hongjian You. Drbox-v2: An improved detector with rotatable boxes for target detection in sar images. *IEEE Transactions on Geoscience and Remote Sensing*, 57(11):8333–8349, 2019.

[15] Wei Bao, Meiyu Huang, Yaqin Zhang, Yao Xu, Xuejiao Liu, and Xueshuang Xiang. Boosting ship detection in SAR images with complementary pretraining techniques. *IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens.*, 14:8941–8954, 2021.

[16] Tianwen Zhang, Xiaoling Zhang, Xiao Ke, Xu Zhan, Jun Shi, Shunjun Wei, Dece Pan, Jianwei Li, Hao Su, Yue Zhou, and Durga Kumar. Ls-ssdd-v1.0: A deep learning dataset dedicated to small ship detection from large-scale sentinel-1 sar images. *Remote Sensing*, 12(18), 2020.

[17] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4510–4520. Computer Vision Foundation / IEEE Computer Society, 2018.

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

[19] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 91–99, 2015.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[21] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014.
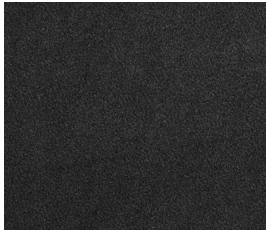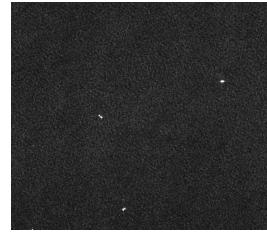
A Appendix

A.1 LSSSDD

In Figure 3 an example of a SAR image is given. Figure 4 depicts a sub-image containing a ship and one without a ship. Table 2 information about the split of the LSSSDD for training and testing is given.

Figure 3: Example of a SAR image [16].

(a) Sub-image without a ship.

(b) Sub-image with a ship.

Figure 4: Example of a SAR sub-image without and with a ship.

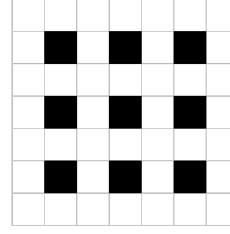| Dataset	Ids	# Subimgs	# Vessels	% Subimg with a vessel	# Vessels / subimg with a vessel
train	1-10	6000	3637	18.7%	3.24
test	11-15	3000	2378	24.5%	3.23

Table 2: Overview of the split into train and test set of the LSSSDD [16].

A.2 Patterns for the correlation algorithm

Figure 5 shows the two used patterns for the correlation algorithm.
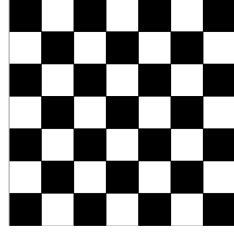
(a) Checkers pattern.

(b) α pattern.

Figure 5: Patterns used in the correlation-optimization.

A.3 Results and F_β scores for the optimizations

Method	Precision	Recall	AP	RT		F_1	$F_{0.5}$	$F_{0.25}$
				GPU	CPU			
$f_{\text{clf}}(I, 0)$	0.788	0.724	0.693	35%	32%	**0.673**	**0.684**	0.690
$f_{\text{clf}}(I, 0.1)$	0.776	0.731	0.698	38%	35%	0.659	0.682	0.693
$f_{\text{clf}}(I, 0.2)$	0.768	0.741	0.706	44%	41%	0.622	0.670	**0.695**
$f_{\text{clf}}(I, 0.3)$	0.758	0.746	0.710	70%	68%	0.421	0.557	0.657
$f_{\text{clf}}(I, 0.38)$	0.754	0.748	0.711	95%	93%	0.098	0.203	0.410

Table 3: Results Faster R-CNN model with f_{clf}

Method	Precision	Recall	AP	RT		F_1	$F_{0.5}$	$F_{0.25}$
				GPU	CPU			
$f_{\text{cor-checkers}}(I, 3, [1, 0.1, 0.1], 0.45)$	0.769	0.689	0.657	63%	63%	0.470	**0.566**	0.627
$f_{\text{cor-checkers}}(I, 2, [1, 0.33], 0.5)$	0.768	0.672	0.640	62%	62%	**0.479**	0.564	0.616
$f_{\text{cor-checkers}}(I, 2, [1, 0.1], 0.5)$	0.768	0.672	0.640	62%	62%	0.478	0.564	0.616
$f_{\text{cor-checkers}}(I, 1, [1], 0.5)$	0.766	0.508	0.701	67%	67%	0.444	0.556	0.630
$f_{\text{cor-checkers}}(I, 2, [1, 0.1], 0.375)$	0.766	0.701	0.667	67%	67%	0.442	0.554	0.629
$f_{\text{cor-checkers}}(I, 3, [1, 0.1, 0.1], 0.35)$	0.764	0.706	0.672	68%	68%	0.436	0.552	**0.632**
$f_{\text{cor-checkers}}(I, 3, [1, 0.1, 0.1], 0.2)$	0.760	0.735	0.699	80%	80%	0.315	0.470	0.611
$f_{\text{cor-checkers}}(I, 3, [1, 0.33, 0.1], 0.2)$	0.761	0.737	0.701	79%	79%	0.319	0.474	0.614
$f_{\text{cor-checkers}}(I, 3, [1, 0.5, 0.25], 0.2)$	0.761	0.738	0.703	81%	81%	0.300	0.457	0.607

Table 4: Results Faster R-CNN model with $f_{\text{cor-checkers}}$.

Method	Precision	Recall	AP	RT		F_1	$F_{0.5}$	$F_{0.25}$
				GPU	CPU			
$f_{\text{cor-}\alpha}(I, 3, [1, 0.1, 0.1], 0.35)$	0.778	0.660	0.631	53%	53%	0.536	0.589	0.618
$f_{\text{cor-}\alpha}(I, 2, [1, 0.1], 0.4375)$	0.788	0.644	0.616	50%	50%	**0.553**	0.589	0.608
$f_{\text{cor-}\alpha}(I, 3, [1, 0.5, 0.1], 0.35)$	0.775	0.651	0.623	52%	52%	0.545	**0.589**	0.612
$f_{\text{cor-}\alpha}(I, 3, [1, 0.1, 0.1], 0.2)$	0.764	0.694	0.662	62%	62%	0.486	0.578	0.635
$f_{\text{cor-}\alpha}(I, 2, [1, 0.1], 0.1875)$	0.770	0.690	0.658	60%	60%	0.496	0.582	0.634
$f_{\text{cor-}\alpha}(I, 2, [1, 0.1], 0.125)$	0.767	0.696	0.664	62%	62%	0.487	0.580	**0.637**
$f_{\text{cor-}\alpha}(I, 3, [1, 0.5, 0.1], 0.1)$	0.757	0.743	0.708	79%	79%	0.327	0.479	0.614
$f_{\text{cor-}\alpha}(I, 3, [1, 0.75, 0.1], 0.1)$	0.757	0.743	0.708	81%	81%	0.304	0.460	0.606
$f_{\text{cor-}\alpha}(I, 3, [1, 1, 0.1], 0.1)$	0.757	0.743	0.708	81%	81%	0.304	0.459	0.606

Table 5: Results Faster R-CNN model with $f_{\text{cor-}\alpha}$.

Method	AP	RT		F_1	$F_{0.5}$	$F_{0.25}$
		GPU	CPU			
$f_{\text{cor-}\alpha}(I, 2, [1, 0.5], 0.25)$ en $f_{\text{clf}}(I, 0)$	0.638	25%	24%	**0.688**	0.657	0.643
$f_{\text{cor-}\alpha}(I, 2, [1, 0.5], 0.25)$ en $f_{\text{clf}}(I, 0.2)$	0.664	32%	30%	0.671	**0.666**	**0.665**

Table 6: Results Faster R-CNN model with f_{cor} and f_{clf}

A.4 Hyperparameters for the used DL models

Hyperparameter	Value
Batch size	256
Base learning rate	0.001
Momentum beta	0.9
Weight decay	0.0001
Anchor sizes	10, 16, 32, 40, 64
Anchor aspect ratio's	0.5, 1, 2
NMS threshold	0.5

Table 7: Hyperparameters for the Faster R-CNN model

Loss function	Function
RPN classification loss function	softmax binary CEL
RPN localisation loss function	L1 loss
Bounding box localisation loss function	smooth L1 loss
Bounding box classification loss function	softmax CEL

Table 8: Loss functions for the Faster R-CNN model

Hyperparameter	Value
Batch size	32
Base learning rate	0.0001

Table 9: Hyperparameters for the MobileNetV2 model