# Tackling the Background Bias in Sparse Object Detection via Cropped Windows

Leon Amadeus Varga
Cognitive Systems Group
University of Tuebingen
Tuebingen, Germany
leon.varga@uni-tuebingen.de

Andreas Zell
Cognitive Systems Group
University of Tuebingen
Tuebingen, Germany
andreas.zell@uni-tuebingen.de

## Abstract

*Object detection on Unmanned Aerial Vehicles (UAVs) is still a challenging task. The recordings are mostly sparse and contain only small objects. In this work, we propose a simple tiling method that improves the detection capability in the remote sensing case. We identified one core component of many tiling approaches and extracted an easy to implement preprocessing step.*
*By reducing the background bias and enabling the usage of higher image resolutions during training, our method can improve the performance of models substantially. The procedure was validated on three different data sets and outperformed similar approaches in performance and speed.*

## 1. Introduction

In the recent years, object detection, which is a fundamental part of computer vision, improved significantly. One of the critical improvements was the usage of deep neural networks for object detectors [6, 18]. Nowadays, state-of-the-art models can accurately predict objects in generic object data sets like MS COCO [12] or Pascal VOC [5].

In addition, the development of camera technology has enormously increased the resolution of the recordings. For example, it is possible to equip small Unmanned Aerial Vehicles (UAVs) with high-resolution cameras.

Currently, there is a large gap between the available resolution of the recordings and the resolution, which is used by the object detectors. The main reason for this is the limitation of the graphics processing unit (GPU) memory and speed (see section 3.1).

The standard approach is the down-scaling of the input image. The advantage of the down-scaling is the reduced computation effort, allowing higher frames-per-seconds (FPS) for the detection [1]. However, down-scaling also has a significant disadvantage. The size of the small objects is reduced, and high level details are removed.

Small objects are defined by Lin *et al*. as the objects with an area (width $\times$ height) below $32^2$ pixels [12]. The detection of these objects is still a challenging task. For object detection on UAVs, planes, or satellites, detecting small objects is the common case. This application is called remote sensing. For this application, down-scaling is problematic.

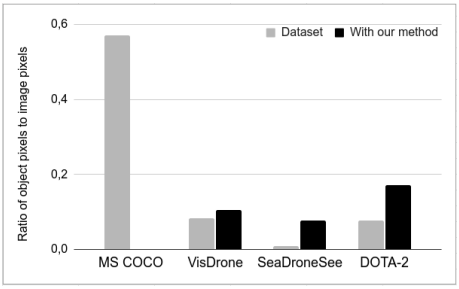



Figure 1: Ratio between the annotated pixels and the image pixels for different data sets. In contrast, MS COCO is much more balanced.

Further, the objects are often sparse in these recordings, which leads to an unbalanced ratio between objects and background. Therefore, there is a bias towards the background, which affects the prediction capability of the detectors. Our main contributions are the following:

- We give an in-depth analysis of one of the object detection problems on remote sensing images. We show that sparsity leads to a background bias.
- We utilize a straightforward cropping approach to solve this problem. Similar techniques are common practice for other applications. However, they are not yet widely used for remote sensing data. We aim to create awareness of background bias and provide a simple and reliable solution.
- We complete it with a comprehensive evaluation on three data sets with many one-stage detectors. We also analyze the influence of the parameters. In addition, we provide an official implementation.

## 2. Related work

In this section, we provide an overview of techniques that solve or reduce the impact of the stated problems and are related to our approach.

### 2.1. Object detection

Object detection is the task of localization and classification of objects. We focus on the two-dimensional case, which uses images as input data. The state-of-the-art approaches are based on deep neural networks [18, 1, 21].

There is a distinction between two-stage and one-stage detectors. The two-stage detectors utilize a region proposal network to predict regions of interest (ROIs). These ROIs describe the input of a second network. This network classifies and regresses the ROIs [18, 6]. In contrast, one-stage detectors combine the first stage and the second stage into a single step. They predict bounding boxes of the objects directly with the class [1, 21, 4]. The two-stage detectors have a better performance. In return, the one-stage detectors are faster. Due to the better performance-speed-ratio, our focus lies on the later ones.

Further, a distinction between anchor-based and anchor-free detectors is possible. Anchor-based detectors make use of precalculated anchors, which are distributed over the input images. These specify all possible bounding boxes and are the basis for further calculations [1, 21, 18]. Anchor-free approaches try to eliminate this dependency and use other ways to define the bounding boxes. Zhou *et al*. used, for example, a heat-map to predict the centers of the objects. Therefore, no anchor-boxes are necessary [31]. We have conducted experiments with anchor-based and anchor-free object detectors and showed that our method works in both cases.

Lin *et al*. introduced a loss function called Focal loss, which focuses on the hard examples. Therefore, it can perform well even with class imbalance [13] and is commonly used. We want to tackle the bias between foreground and background, so their loss is also beneficial in this application. Two of our models (CenterNet and EfficientDet) use the Focal loss by default. So Focal loss does not solve the imbalance in this case entirely.

### 2.2. Data augmentation

Our method uses crops of the images as training data. Data augmentation techniques such as Random Cropping, Random Image Cropping And Patching (RICAP) [19], or Cut Mix [29] commonly use a similar approach. Random Cropping cuts random crops out of the input image. These crops are used for training. Takahashi *et al*. introduced RICAP, which crops four images and combines them into a new training image [19]. Yun *et al*. proposed CutMix [29], which is a combination of Mixup and Cutout. Therefore, CutMix replaces the cutouts with crops of different images. In contrast to our method, these approaches focus on improving the object detector to recognize parts of an object, reduce the contextual impact, and simulate occlusion. These techniques are perfect for medium- or large-scale objects occurring in MS COCO or Pascal VOC. Here they can achieve significant improvements. These augmentation techniques are counterproductive for the detection on remote sensing recordings, which have mainly small and sparse objects. In 4.3, there is a deeper analysis of the mentioned augmentation techniques.

Our method is deterministic and defines only one representation of the training data. No random augmentation takes place. Thus, a combination of these augmentation techniques and our approach is still possible and shown in the experiments.

Besides these augmentation techniques, Hong *et al*. proposed a patch-level augmentation approach [9]. They address class imbalances as a problem of UAV data sets. After the training procedure, their proposed method generates hard samples with misclassified object instances. These hard samples are used for further training. Similar to their approach, we solve the class imbalance. We focus on the foreground to background imbalance. A direct comparison with their approach is not possible. They used additional data for their experiments. Further, the exact hyperparameters are not mentioned.

Kisantal et al. proposed a similar approach. Their method over-samples small objects by augmenting the images with additional instances of small objects [11]. For their augmentation, the segmentation masks of the objects are required, which is a costly requirement for the data-set. Many data sets do not provide the ground truth segmentation mask. This applies also to the used data-sets VisDrone [32], SeaDronesSee [25] and DOTA [3].

Xia et al. recommend the users of their data set DOTA [27] a cropping technique similar to our approach, but the reason and the effect are completely different. We use the new 2nd version of the DOTA data set for our experiments [3]. Therefore, an analysis of the differences can be found in section 4.1.

### 2.3. Architectures with higher resolution

Furthermore, there are methods, which can utilize high-resolution images. Najibi *et al*. proposed AutoFocus, a multi-scale inference approach [15]. On a coarse resolution, FocusPixels and large objects are predicted. FocusChips, which contain these FocusPixels, are processed in a further round with a finer resolution. In the end, the predictions of the different scales have to be merged. Their multi-stage approach seems very promising, but has some drawbacks. The multi-stage analysis of an image cannot be parallelized, so it is slower than a single forward pass.

Further, their method has to be adapted to other network

(a) MS COCO (b) VisDrone (c) SeaDronesSee (d) DOTA-2

Figure 2: Example images of different data sets with orange ground-truth bounding boxes. The orange-green pie-charts show the foreground-background-ratio in the training data set. (orange: foreground; green: background)

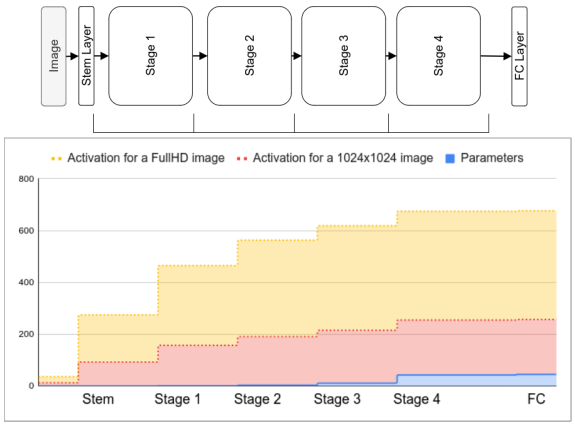



Figure 3: On the example ResNet18 classifier [8], the necessary space in MByte for the weights and the activation for two input images is shown.

architectures. They showed results for a Faster R-CNN and claimed compatibility with other models. But it is still necessary to adjust their process to other models. AutoFocus is helpful for data sets with an extensive range of object sizes. Here, the detection of large objects can already happen within a coarse resolution. In the remote sensing use case, there are primarily small objects. So the multi-stage approach does not reduce the inference time like for MS-COCO.

Unel *et al*. proposed a two-stage method. Similar to our approach, their method divides the input image into tiles [24]. They use these tiles and the full-frame for both, the training and the inference. For the inference, a merging step is necessary. Our approach is very similar to their method. The major difference lies in the inference. Our method utilizes only the full-frame. So we skip the merging of the predictions in the post-processing step and need only a single forward pass for the whole image. A more detailed comparison can be found in the experiment section 4.3.

Tang *et al*. proposed a combination of the two previous mentioned methods [22]. PENet could improve the accuracy for two data sets massively. But the technique has two drawbacks. First, the approach is slow. And second, the method needs additional annotations, which makes comparison impossible.

### 2.4. Technical solutions

To tackle the memory limitation during training, the usage of GPUs with larger memory is an option. In most cases, this is not possible. We kept the maximum resolution of the input images for all configurations fixed for the experiments in the following. By this, we show that our method can improve the detector performance by just solving the background bias.

Like a larger GPU memory, the representation of the floating-point numbers with half-precision (Float16) could diminish the memory limitation issue. For our experiments, we used single-precision (Float32) to be comparable to the related works. Our method is also compatible with half-precision.

We present a simple technique, which allows the usage of high-resolution images. Furthermore, it reduces the background bias in the training data. We further contribute experiments on different data sets and an in-depth analysis of the impact on the trained detector.

## 3. Proposed Method

In this section, we describe our method. Starting with the motivation, we continue with the method and some implementation details.

We offer an official implementation (https://git.io/JRQNI).

### 3.1. Motivation

The training process of a neural network is heavily data-driven. The benchmark, which is often used to measure state-of-the-art detectors, is still MS COCO [12]. In Fig. 2, four example images of different data sets are shown. The pie-charts indicate the ratio of the foreground to the background pixels in the different training sets. For MS COCO

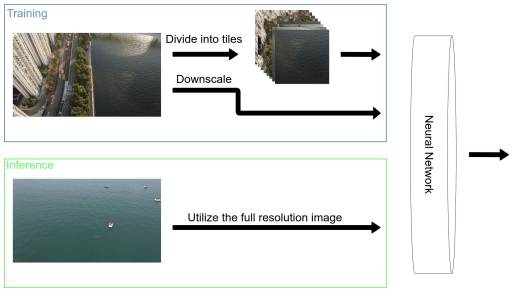



Figure 4: The method Cropping Window (CroW). The graphic shows the difference between the training and the inference procedure.

[12], this is almost balanced. This is not given for remote-sensing data sets (like VisDrone [32], SeaDronesSee-DET [25], and DOTA-2 [3]). So MS COCO is not a good representative for the sparse recordings, common for remote sensing. Even with Focal-Loss [13], which lightens the impact of unbalanced data sets, this is still a large source of errors.

Besides the bias-driven motivation, our goal was also to utilize high-resolution images in the training procedure. We already mentioned the limiting factor of GPU memory. Therefore, in many cases, the input images must be scaled down to fit into the memory. Especially for detecting small objects, this is counterproductive and can harm the performance of a model heavily. Nowadays, it is common to use a higher resolution for inference than for training [17]. Touvron *et al*. showed that this discrepancy of the train-test resolution is even beneficial for classification tasks [23]. With our method, it is possible to train on high resolutions and utilize even higher resolutions during inference. This improves the accuracy on small objects significantly.

The reason for the huge memory consumption during training can be found in the backpropagation formula. In Eq. 1 the weight adaption of the weight $w_{ij}$ for one sample $p$ of the backpropagation algorithm is given [30]. $\eta$ defines the learning rate, $o_{pi}$ is the activation of the previous layer $i$ and $\delta_{pj}$ the backpropagated error of the following layer $j$.

$$\Delta_p w_{ij} = \eta o_{pi} \delta_{pj} \tag{1}$$

The backpropagation algorithm starts at the end of the network. It propagates the error back through the network, so the activation of the previous layer must be kept all the time. Especially for convolution layers, the size of the intermediate results depends highly on the size of the input data. By sharing the weights through the kernel, the amount of weights is reduced drastically compared to a fully connected layer [7]. To calculate the adaption of the kernel weights, it is still necessary to consider all positions of the kernel on activation of the previous layer. In Fig. 3, the ratio between parameter space and activation space adds up over the layers. The necessary activation space is a multiple of the parameter space. Therefore, most memory for the weight adaptation step is required by the activation. This also applies to separable convolutions [14], which divide the spatial and the channel calculation to reduce the number of parameters. The memory usage becomes even worse with many layers because the amount of intermediate results increases.

The memory limitation causes a problem for large input data. The available memory becomes, therefore, the limitation for the maximal training image sizes. For the inference, it is not necessary to hold the activation of hidden layers.

With our method, we can use higher resolutions by splitting up the computations.

**Algorithm 1** Cropping Window (CroW)

**Input:** Training set $S$ with annotations,
Tile size $\alpha$ (pixels), Tile overlap $\beta$ (0 - 1), Down-scaling factor $\gamma$ (0 - 1)
**Output:** Training set $S_{\text{new}}$
*Initialization :*
$S_{\text{new}}$ is an empty list
*Tile generation :*
**for** Image with annotations $i$ in $S$ **do**
  Tiles $T$ = divide_image($i$, $\alpha$, $\beta$)
  **for** tile $t$ in $T$ **do**
    **if** ($t$ contains ground truth boxes) **then**
      Append $t$ to $S_{\text{new}}$
    **end if**
  **end for**
**end for**
*Downscale full frame :*
**for** Image with annotations $i$ in $S$ **do**
  Image $i_{\text{new}}$ = downscale_image($i$, $\gamma$)
  Append $i_{\text{new}}$ to $S_{\text{new}}$
**end for**
**return** $S_{\text{new}}$

## 3.2. Method

Fig. 4 shows a sketch of the approach. We focus fully on an adaptation of the training process, which is also the major difference to the method of Unel *et al*. [24] (see 4.3). We propose a deterministic way to change the representation of the training data that addresses the two problems mentioned above. The method is architecture-independent.

In Alg. 1 the procedure of our method is described. We split the image into tiles and discard the empty ones. The parameters $\alpha$ and $\beta$ define the tiles. $\alpha$ specifies the size of the tiles and $\beta$ the area of relative overlap. In our experiments, we kept the values fixed ($\alpha = 512$ pixels and $\beta = 0.25$). Further, we could show (see 5.2 and 5.3), that these parameters are independent of the data set. The third

| Data set | VisDrone | SeaDronesSee | Inference FPS (Desktop) | Inference FPS (embedded) | Number of Parameters |
|---|---|---|---|---|---|
| EfficientDet-$d0$ | 19.54 ± 0.64 | 18.85 ± 3.65 | 46 | 8 | 4.5M |
| With CroW (our) | **25.30 ± 1.93** | **31.21 ± 0.64** | 46 | | |
| EfficientDet-$d4$ | 19.66 ± 0.60 | 24.77 ± 1.24 | 21 | - | 17.7M |
| With CroW (our) | **27.61 ± 1.09** | **30.41 ± 0.44** | | | |
| YoloV4 | 17.74 ± 2.00 | 30.75 ± 1.64 | 20 | - | 63.9M |
| With CroW (our) | **28.65 ± 5.40** | **36.41 ± 1.40** | | | |
| CenterNet-ResNet18 | 24.20 ± 2.27 | 23.41 ± 1.52 | 78 | - | 14.4M |
| With CroW (our) | **26.88 ± 0.73** | **31.49 ± 1.54** | | | |
| CenterNet-ResNet50 | 29.73 ± 0.50 | 23.24 ± 2.99 | 33 | - | 30.7M |
| With CroW (our) | **35.25 ± 0.57** | **33.07 ± 0.37** | | | |
| CenterNet-ResNet101 | 26.14 ± 3.18 | 20.63 ± 1.21 | 22 | - | 49.7M |
| With CroW (our) | **33.63 ± 1.61** | **33.68 ± 2.59** | | | |
| CenterNet-Hourglass104 | 28.47 ± 0.25 | 26.4 ± 0.52 | 6 | - | 200M |
| With Random Cropping | 29.50 | 22.43 | | | |
| With CroW (our) | **31.63 ± 0.01** | **27.53 ± 0.08** | | | |
| Pelee_T5x3_I5x3 [24] | 15.34 | - | - | 6 | 5.4M |
| Pelee38_T5x3_I5x3 [24] | 16.61 | - | - | 5 | |
| EfficientDet-$d0$ With CroW (our) | **18.33** | - | 46 | **8** | 4.5M |
| CenterNet-Hourglass104 5th place in VisDrone-19[17] | 31.97 | - | 6 | - | 200M |
| With CroW (our) | **38.36** | - | | | |

Table 1: This table shows the mean Average Precision for the averaged IoU thresholds between 0.5 and 0.95 ($mAP^{0.5:0.95:0.05}$) for different configurations. Value per cell: mean ± standard deviation.

parameter $\gamma$ represents the down-scaling factor, which is used to down-scale the full-frame in the training set. The down-scaling is necessary if the processing of the full-frame does not fit into the GPU memory. So it would be possible to utilize higher resolution images for the tiles during the training.

To give a fair comparison with the baseline models, we down-scaled the data set images to the maximal resolution, for which even the full-frame fits into the memory. Therefore, the value $\gamma$ was set to 1 for the most experiments. Both functions ('divide_image' and 'downscale_image') process both, the image and the annotations.

The representation of the training set, which is generated this way, is used to train the neural network. With this technique, it is possible to use high-resolution images as tiles and reduce background, which decreases the background bias.

At inference, the full-frame in the maximal resolution is used. Because of a focus on fast inference, we did not consider test-time augmentation. Nevertheless, it should still be possible to use this combined with our method and could even benefit.

### 3.3. Cropping pattern

To improve the performance, a suitable cropping pattern for the tiles is required. An overlapping sliding window approach worked best in our experiments. The idea is that each object occurs at least once uncut in a tile. We achieved this by placing four fixed tiles in the corners of the image (see 1 in Fig. 5). Afterward, we filled the intermediate areas with equally distributed tiles (see 2 in Fig. 5). The number of tiles is calculated via the minimal overlap-parameter $\beta$.

## 4. Experiments

In this section, we describe our experiments. We give an overview of the training and test configuration. At the end of this section, we discuss the results (in Tab. 1 and 2).

For all experiments, we used PyTorch in version 1.8. For the batch experiments, we used computation nodes with four Geforce GTX 1080 Ti. We did single experiments on a Geforce RTX 2080 Ti and a computation node with eight RTX 3090. The inference time is measured on a Geforce RTX 2080 Ti (Desktop). To compare the speed with Unel *et al*. [24], we used a Jetson TX2 (embedded). For the embedded inference, half-point precision was used.

### 4.1. Data sets

We used three different data sets to evaluate our method. All three are based on aerial recordings. VisDrone [32] and SeaDronesSee [25] consist of Micro Air Vehicle (MAV) recordings. DOTA-2 contains mainly images recorded by

| Data set | VisDrone | SeaDronesSee | DOTA-2 |
|---|---|---|---|
| EfficientDet $d0$ | 19.54 ± 0.64 | 18.85 ± 3.65 | 19.35 ± 0.79 |
| with CroW | **25.30 ± 1.93** | **31.21 ± 0.64** | **33.97 ± 4.04** |
| EfficientDet $d4$ | 19.66 ± 0.60 | 24.77 ± 1.24 | 24.82 ± 0.83 |
| with CroW | **27.61 ± 1.09** | **30.41 ± 0.44** | **44.01 ± 2.43** |
| YoloV4 | 17.74 ± 2.00 | 30.75 ± 1.64 | 6.44 ± 0.53 |
| with CroW | **28.65 ± 5.40** | **36.41 ± 1.40** | **22.04 ± 0.83** |
| CenterNet ResNet18 | 24.20 ± 2.27 | 23.41 ± 1.52 | 15.18 ± 0.56 |
| with CroW | **26.88 ± 0.73** | **31.49 ± 1.54** | **19.67 ± 0.71** |
| CenterNet ResNet50 | 29.73 ± 0.50 | 23.24 ± 2.99 | 15.85 ± 2.31 |
| with CroW | **35.25 ± 0.57** | **33.07 ± 0.37** | **23.06 ± 7.53** |
| CenterNet ResNet101 | 26.14 ± 3.18 | 20.63 ± 1.21 | 14.19 ± 0.62 |
| with CroW | **33.63 ± 1.61** | **33.68 ± 2.59** | **27.05 ± 6.72** |
| CenterNet Hourglass104 | 45.77 ± 8.49 | 51.41 ± 1.06 | 43.26 ± 0.86 |
| with Random Cropping | 52.47 | 46.65 | 50.69 |
| with CroW | **55.51 ± 0.23** | **53.77 ± 0.22** | **52.40 ± 0.23** |

Table 2: This table shows the mean Average Precision with a IoU threshold of 0.5 ($mAP^{0.5}$) for different configurations. Value per cell: mean ± standard deviation.

satellites [3]. For all experiments, we use the common metric mean Average Precision and use the calculation metric of MS COCO [12]. We only adapted the maximal number of considered detections per image to fit for each data set.

**VisDrone-DET** Zhu *et al*. proposed one of the most prominent MAV recordings data set [32]. The application of this data set is traffic surveillance in urban areas. For our experiments, we used the detection task (VisDrone-DET). The training set contains 6,471 images with 343,205 annotations. The image resolution ranges from 960×540 pixels to 2000×1500 pixels. Unless otherwise mentioned, we reduced the maximum side length by down-scaling to 1024 pixels. So the whole image still fits into the GPU memory as a complete image.

As the test-set is not public for this data set, we used the validation set for the evaluation. For VisDrone-DET this is common practice [26, 22, 28]. The validation set contains 548 images.

**SeaDronesSee-DET** In contrast to VisDrone, SeaDronesSee aims at maritime environments [25]. For the search and rescue application, the detection of swimmers and boats is necessary. The training set includes 2,975 images with 21,272 annotations. The resolutions of the images range from 3840×2160 pixels to 5456×3632 pixels. We reduced the maximum side length of the images to 1024 pixels for this data set, too.

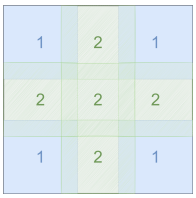



Figure 5: The overlapping cropping patter.

The reported accuracy is evaluated on the test set (with 1,796 images).

**DOTA-2** Ding *et al*. provide high-resolution satellite images with annotations [3]. The diversity of the image resolutions is much larger (475×547 pixels to 29,200×27,616 pixels). A script is provided to divide the recordings into tiles. This approach can be compared with our method. However, the goal of their method is to divide the image into smaller tiles. Our method uses an evaluated cropping pattern and reduces background bias. For the experiments on DOTA-2, we used their preprocessing technique. So our approach is compatible with their preprocessing step. Further, we used their annotations of task 2. For DOTA-2, the test-set is not public, further, the evaluation server was not reachable. Therefore, the validation set (with 2,619 image-tiles) was used. It is common for the DOTA data set to state the mean Average Precision with an IoU-threshold 0.5 ($mAP^{0.5}$)[3].

## 4.2. Models

We used three one-stage object detectors for our experiments. To prevent overfitting, we used early-stopping based on the validation loss for all models. If the validation loss did not decrease after the minimal epochs (50 epochs) within ten epochs, we stopped the training.

For each model, the hyper-parameters were optimized for the baseline experiment.

Besides the three mentioned models, we compared our results with approaches of Unel *et al*. [24], Pailla *et al*. [17] and similar augmentation techniques such as Random Cropping and Mosaic augmentation [1].

**EfficientDet** EfficientDet is optimized for efficiency and can perform well with small backbones [21]. Even there is an EfficentNetV2 announced, which should be more efficient as backbone [20], there is currently, to the best of our knowledge, no object detector using this backbone.

For our experiments, we used EfficientDet with two backbones. The $d0$-backbone is the smallest and fastest of

this family. And the $d4$-backbone represents a good compromise between size and performance.

We used three anchor scales (0.6, 0.9, 1.2), which are optimized to detect the small objects of the data sets. For the optimization, we used an Adam optimizer [10] with a learning rate of $1e^{-4}$. Further, we used a learning rate scheduler, which reduces the learning rate on plateaus with patience of 3 epochs.

**CenterNet** Duan *et al*. proposed CenterNet [4], an anchor-free object detector. The network uses a heat-map to predict the center-points of the objects. Based on these center-points, the bounding boxes are regressed.

Hourglass-104[16] is a representative for extensive backbones, while the ResNet-backbones [8] cover a variety of different backbone sizes.

The ResNet backbones were trained with Adam and a learning rate of $1e^{-4}$. Further, we also used the plateau learning scheduler. For the Hourglass104, we used the learning schedule proposed by Pailla *et al*. [17].

**YoloV4** Bochkovskiy *et al*. published YoloV4[1], which is the latest member of the Yolo-family providing a scientific publication. Besides a comprehensive architecture and parameter search, they did an in-depth analysis of augmentation techniques, called 'bag of freebies', and introduced the Mosaic data augmentation technique.

YoloV4 is a prominent representative of the object detectors because of impressive results on MS COCO. By default, YoloV4 scales all input images down to an image size of 608×608 pixels. For our experiments, we removed this preprocessing to improve the prediction of smaller objects. The down-scaling speeds up the inference time massively. That is the reason why YoloV4 achieves in our experiments only 20 FPS.

### 4.3. Results

In Tab. 1 and 2 the mean Average Precision ($mAP$) for the different configurations are given. For each configuration, we stated the mean and the standard deviation over three runs with different seeds.
All models improve for all data sets with 2 to 13 $mAP$ points compared to the baseline model. No adaption of the network architecture is necessary. Therefore, the number of parameters and the inference time are equal to the baseline model.

Further, we could improve the results of a participant of the VisDrone challenge 2019 by nearly 20%. Pailla *et al*. achieved 5th place in the challenge with a heavily optimized CenterNet-Hourglass104[17]. We achieved the improvement by just adding our method to their model. Both stated values in Tab. 1 are without test-time-augmentation, but with a larger validation image size (maximal image-side length of 2048 pixels). With our method, we could also utilize this higher resolution during training. Because of the memory limitation, they could not train on the same resolution. Therefore, the higher resolution was seen by their detector the first time at the inference time.

We also tested comparable data augmentation techniques. The Mosaic augmentation [1] is part of the YoloV4 pipeline. Hence, all experiments of YoloV4 used this augmentation technique. Since the YoloV4 experiments indicate the same improvement, it is shown that our method tackles another issue. Further, we showed for the CenterNet-Hourglass104 that Random Cropping could not achieve the same results as our method.

The technique proposed by Unel *et al*. [24] could perform well on the VisDrone data set. Their goal was to solve the small object detection problem with a tiling approach similar to ours. They missed an in-depth analysis for the cause of improvement and created an unnecessarily complex pipeline. The networks Pelee and Pelee38 were used for their experiments [26]. We could observe that the non-max-suppression (NMS), which they used to merge the prediction of the tiles into a full-frame prediction, is not helpful, especially for small objects, and even slows down the procedure. Their approach further needs multiple forward passes for a single frame. The superior method uses a network, which is not fixed to one input image size and utilizes only the full-frame while inference. We were able to outperform their results in performance and speed with a similar in size, EfficientDet-$d0$. The results must be viewed with caution. Unel *et al*. used only two superclasses of the VisDrone data set (pedestrian and vehicle). Hence, a comparison with the other results of our experiments is not possible.

Further, we compared two different network architectures. This comparison is not entirely fair, since EfficientDet benefits from more recent developments in architecture search. But it can still show the advantage of using the entire image and remove the merge process.

In summary, we showed the data set independence and could outperform similar methods, like data augmentation techniques and tiling approaches.

## 5. Ablation study

In this section, we analyze the impact of our method on the trained detectors. We analyze the influence of the parameters tile size $\alpha$ and tile overlap $\beta$.

### 5.1. Impact on detector

By eliminating other causes and comparing trained detectors, we found the reason for the improvement in the reduced background bias.

In Fig. 6 a TIDE [2] analysis of a CenterNet-Hourglass104 trained with and without our method is visible. Also, a model trained with Random Cropping is shown.

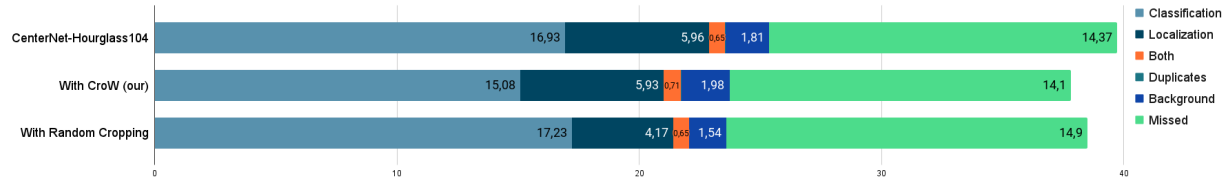
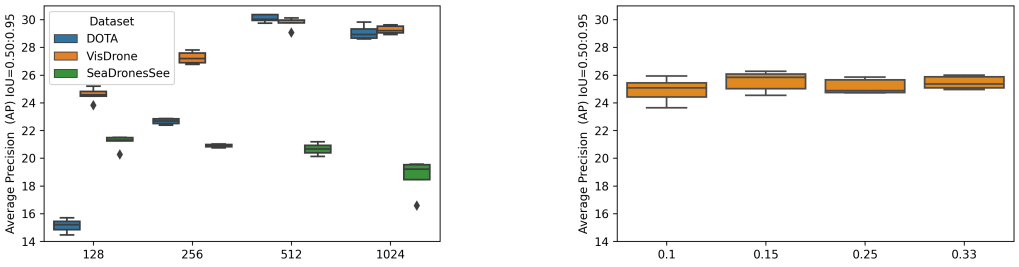



Figure 6: TIDE analysis of the trained CenterNet-Hourglass104 on VisDrone-DET. Larger values indicate a larger error.




(a) Tile size $\alpha$

(b) Tile overlap $\beta$

Figure 7: Impact of the tile size $\alpha$ and tile overlap $\beta$ on the test performance of a CenterNet-Hourglass104.

TIDE is based on the $mAP$ and can break down the cause for the missed accuracy. There are two essential differences visible between the baseline and our approach. First, our method reduced the *Missed* error and the *Classification* error. So the second detector was better in the distinction of classes and missed fewer objects. Further, the *Background* error is increasing minimally for our method and has no real impact on the overall prediction capability. So even after removing some background, there is enough background left to learn. In contrast, Random Cropping improves the localization only slightly and worsens classification. For the VisDrone data set, the ratio of the classification error is still enormous for all approaches.

With our technique, many tiles, which are full of background, are removed. This leads to a much better foreground-background-ratio, which is visible in Fig. 1.

### 5.2. Tile size $\alpha$

In Fig. 7a the test performance for different tile sizes $\alpha$ is displayed. Tiles with a size of 1024×1024 pixels correspond to a single tile for the whole image in our experiments. The influence of the tile size depends on several factors (sparsity, object distribution, object size). A good balance between these factors is provided by the tile size of 512×512 pixels, which was used for our further experiments.

### 5.3. Tile overlap $\beta$

The second hyper-parameter $\beta$ defines the minimal overlap of the tiles. In Fig. 7b, it is visible that the impact of this parameter is much smaller. A tile overlap of at least 0.15 between the tiles produces stable results. To get a better intuition of the parameter $\beta$ impact, we didn't add the full-frame to the training set of these experiments. This magnified the influence of the parameter $\beta$.

## 6. Conclusion

We introduced a simple technique to improve the capabilities of object detectors on sparse recordings by tackling background bias. Furthermore, the method allows the utilization of higher resolutions during training. This enabled us to improve the performance of a VisDrone challenge participant by nearly 20%.

Besides a validation on three different data sets, we could justify the improvement with an analysis of the improvement cause.

This method is easy to implement into an existing object detection pipeline and can improve the performance in addition to other state-of-the-art approaches. By showing this low-hanging improvement, we want to increase the awareness for the background bias in remote sensing recordings.

## Acknowledgment

We want to thank Martin Meßmer, Benjamin Kiefer and Nuri Benbarka for the helpful discussions. This work has been supported by the German Ministry for Economic Affairs and Energy, Project Avalon, FKZ: 03SX481B. Further the Training Center Machine Learning, Tübingen has been used for the evaluation of the models (01—S17054).

## References


[1] Alexey Bochkovskiy, Chien Yao Wang, and Hong Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection, 4 2020. 1, 2, 6, 7

[2] Daniel Bolya, Sean Foley, James Hays, and Judy Hoffman. TIDE: A General Toolbox for Identifying Object Detection Errors. 2020. 7

[3] Jian Ding, Nan Xue, Gui-Song Xia, Xiang Bai, Wen Yang, Micheal Ying Yang, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Object Detection in Aerial Images: A Large-Scale Benchmark and Challenges. 2021. 2, 4, 6

[4] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. CenterNet: Keypoint triplets for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-Octob, pages 6568–6577. Institute of Electrical and Electronics Engineers Inc., 10 2019. 2, 7

[5] Mark Everingham, Luc Van Gool, Christopher K.I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 1

[6] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2015 Inter, pages 1440–1448, 2015. 1, 2

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. 4

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 770–778. IEEE Computer Society, 12 2016. 3, 7

[9] Sungeun Hong, Sungil Kang, and Donghyeon Cho. Patch-level augmentation for object detection in aerial images. In *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, pages 127–134, 2019. 2

[10] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015. 7

[11] Mate Kisantal, Zbigniew Wojna, Jakub Murawski, Jacek Naruniec, and Kyunghyun Cho. Augmentation for small object detection. pages 119–133, 2019. 2

[12] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, page 10, 12 2014. 1, 3, 4, 6

[13] Tsung Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2017. 2, 4

[14] Franck Mamalet and Christophe Garcia. Simplifying ConvNets for fast learning. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7553 LNCS, pages 58–65, 2012. 4

[15] Mahyar Najibi, Bharat Singh, and Larry Davis. Autofocus: Efficient multi-scale inference. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-Octob, pages 9744–9754, 2019. 2

[16] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9912 LNCS, pages 483–499, 2016. 7

[17] Dheeraj Reddy Pailla, Varghese Kollerathu, and Sai Saketh Chennamsetty. Object detection on aerial imagery using CenterNet. 2019. 4, 5, 6, 7

[18] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 6 2017. 1, 2

[19] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. Data Augmentation Using Random Image Cropping and Patching for Deep CNNs. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(9):2917–2931, 2020. 2

[20] Mingxing Tan and Quoc V. Le. EfficientNetV2: Smaller Models and Faster Training. 4 2021. 6

[21] Mingxing Tan, Ruoming Pang, and Quoc V. Le. EfficientDet: Scalable and efficient object detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 10778–10787, 11 2020. 2, 6

[22] Ziyang Tang, Xiang Liu, Guangyu Shen, and Baijian Yang. PENet: Object Detection using Points Estimation in Aerial Images. 2020. 3, 6

[23] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy: FixEfficientNet, 2020. 4

[24] F. Ozge Unel, Burak O. Ozkalayci, and Cevahir Cigla. The power of tiling for small object detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, volume 2019-June, pages 582–591, 2019. 3, 4, 5, 6, 7

[25] Leon Amadeus Varga, Benjamin Kiefer, Martin Messmer, and Andreas Zell. SeaDronesSee: A Maritime Benchmark for Detecting Humans in Open Water. 2021. 2, 4, 5, 6

[26] Robert J. Wang, Xiang Li, and Charles X. Ling. Pelee: A real-time object detection system on mobile devices. In *Advances in Neural Information Processing Systems*, volume 2018-Decem, pages 1963–1972. Neural information processing systems foundation, 4 2018. 6, 7

[27] Gui Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. DOTA: A Large-Scale Dataset for Object Detection in Aerial Images. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3974–3983, 2018. 2

[28] Xuxi Yang, Marc Brittain, Peng Wei, and Michael Murphy. Computer vision for small uas onboard pedestrian detection. In *Aiaa Aviation 2020 Forum*, volume 1 PartF. American Institute of Aeronautics and Astronautics Inc, AIAA, 2020. 6

[29] Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Seong Joon Oh, Junsuk Choe, and Youngjoon Yoo. CutMix: Regulariza-

tion strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-Octob, pages 6022–6031, 2019. 2

[30] A Zell. *Simulation neuronaler Netze*. Oldenbourg, 1997. 4

[31] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as Points. 2019. 2

[32] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Qinghua Hu, and Haibin Ling. Vision meets drones: Past, present and future, 2020. 2, 4, 5, 6