STDnet: A ConvNet for Small Target Detection

Brais Bosquet
brais.bosquet@usc.es
Manuel Mucientes
manuel.mucientes@usc.es
Víctor M. Brea
victor.brea@usc.es

Centro Singular de Investigación en
Tecnoloxías da Información (CiTIUS)
University of Santiago de Compostela
Santiago de Compostela, Spain

Abstract

This paper introduces STDnet, Small Target Detection network, a fully convolutional network (ConvNet) focused on small targets. STDnet includes an early visual attention mechanism, called Region Context Network (RCN), to choose the most promising regions with small objects and their context. RCN allows to work with high resolution feature maps but with a reduced memory usage. The filtered feature maps, which only contain the most likely regions with small objects, are forwarded across the network up to an ending Region Proposal Network (RPN) which feeds a final classification stage. RCN is key to increase localization accuracy through finer spatial resolution due to finer global effective strides, low memory overhead and higher frame rates. We present a new video database, USC-GRAD-STDdb, with more than 56,000 annotated small targets, with sizes under 16×16 px, in challenging scenarios with clutter as a waving sea or air scenes below the skyline. Experimental results over USC-GRAD-STDdb show that STDnet improves the $\text{AP}_{@.5}$ of the best state-of-the-art approach for small target detection from 50.8% to 57.4%.

1 Introduction

Object detection has undergone a great progress through deep convolutional neural networks (ConvNets). Current solutions integrate feature extraction, region proposal, and bounding box and object category in the ConvNet. Applications like sense and avoid on board of unmanned aerial vehicles (UAVs) or video surveillance over wide areas demand early detections of objects to act quickly. This means to detect as far —and therefore small— an object as possible. Recent CNN object detectors provide high accuracy over a wide range of scales, from less than 32×32 pixels up to the image size. Nevertheless, there are no specific networks focused on small targets. Qualitatively, we refer to small as those objects without definitive visual cues to assign them to a category or subcategory. Quantitatively, small refers to sizes under 16×16 pixels.

Most of the state-of-the-art ConvNets for object detection are unsuitable for the detection of such small objects, because both the region proposal, the bounding box regression and the final classification take the feature maps generated in the last convolutional layers as inputs.

© 2018. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

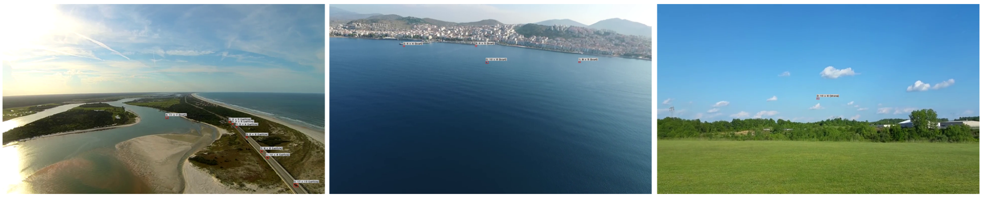

Figure 1: USC-GRAD-STDdb examples. Ground truth objects are enclosed in red boxes.

These feature maps have much lower resolution than the input image —in most cases the reductions in resolution are up to 16×. Thus, many small objects are represented in the last feature maps by only one pixel, which makes classification and bounding box regression very hard, if not impossible.

A straightforward solution for small object detection would be to modify a state-of-the-art ConvNet keeping the resolution of the initial image in all the feature maps. Of course, this approach is non-viable because, due to the size of the network, it would not fit on a GPU and, also, the forwarding pass would be very slow.

This paper introduces a new ConvNet architecture for small object detection called STDnet. STDnet has a size that is significantly lower than its counterparts for the same resolution of the last feature map. Our hypothesis is that, after a few convolutional layers, the feature map contains enough information to decide which regions of the image contain candidate objects, but there are not enough data to classify the region or to perform bounding box regression. Given an intermediate feature map, STDnet applies a novel component, called Region Context Network (RCN), which is a filter that allows to select the most promising regions —all of them with the same size— of the feature map, avoiding the processing of the remaining areas of the image. The RCN ends up with the RoI Collection Layer (RCL), that builds a new, reduced and filtered feature map by arranging all the regions selected by the RCN. Therefore, the memory overhead of the feature maps following the RCN is much lower while keeping a dense spatial resolution. Finally, STDnet applies a Region Proposal Network (RPN) to the last filtered feature map, classifies the regions and performs bounding box regression inside them.

The main contributions of our proposal are:

- STDnet, a new ConvNet for small object detection able to work with high resolution feature maps in the deeper layers with an acceptable memory overhead. STDnet relies on a novel component, RCN, that selects the most promising regions of the image and generates a new filtered feature map with these areas. Therefore, the filtered feature maps can keep the same resolution but with a lower memory overhead and a higher frame rate.

- As STDnet has an RPN that works with anchors, we propose to automatically select the number and sizes of the anchors through a novel algorithm based on k-means. Our proposal differs from [17] in that our algorithm selects not only the sizes of the anchors, but also their number, making the anchors selection fully automatic.

- A new dataset, USC-GRAD-STDdb, for small object detection with more than 56,000 annotated objects of sizes between 4×4 and 16×16 (*e.g.* Figure 1). USC-GRAD-STDdb comprises 115 video segments (>25,000 frames) over the three principal landscapes: air, sea and land.

2 Related Work

Modern object detectors are based on ConvNets [9]. Faster-R-CNN [18] has become a milestone in ConvNets for object detection with the introduction of the Region Proposal Network (RPN). The RPN is based on anchors, which are predefined regions of different sizes and aspect ratios to cope with multiple scales. The RPN yields the coordinates of the bounding boxes and their corresponding classes, namely, object and background. Finally, given the output of the RPN and the last feature map of the feature extractor network, the bounding box and class of the object are determined through a fully-connected classification network.

The off-the-shelf Faster-R-CNN is not adequate for small object detection due to two reasons. First, the sizes of the predefined anchors are very large for small objects. Second, and more important, the global effective stride —downscaling of the input image with respect to the feature map that is the input to the RPN— is 16, which means that a 16×16 object is represented by just one pixel in that feature map. The detection of small objects requires a finer global effective stride in Faster-R-CNN. This leads to a very large memory overhead, making the implementation impossible for current GPUs [1].

In [2], a fully convolutional approach to object detection, called Region-based Fully Convolutional Network (R-FCN), is presented. R-FCN generates $k \times k \times (C+1)$ feature maps in the last convolutional layer, instead of only one. These maps are position-sensitive, i.e., each of the $k \times k \times (C+1)$ maps corresponds with a part of an object of one of the C object categories (+1 for background).

The capability of dealing with objects of different sizes in Faster-R-CNN and R-FCN is limited to a few scales produced with the anchors. Hence, more recent ConvNets for object detection tackle scale invariance and small object detection with more elaborated solutions.

Li *et al.* [12] introduce a Perceptual Generative Adversarial Network for small object detection. This proposal has been tested with two datasets: (i) traffic signs from the Tsinghua-Tencent 100k dataset [23], where they consider as small objects those with an area under 32×32 pixels; (ii) pedestrians over 50 pixels tall from the Caltech benchmark [3].

In [4] a Faster-R-CNN-based approach for company logo detection validated in the FlickrLogos dataset is presented. Authors present an architecture with three RPNs to detect objects of different sizes. Both the RPNs and the final classification and bounding box regression receive as inputs the combination of the feature maps of the last three convolutions: high-level feature maps are upscaled through bilinear interpolation and then summed with the lower-level maps.

Also, in [1] an architecture with several RPNs is proposed. Shallower RPNs are adequate for small objects, while deeper ones are appropriate for larger targets. In order to have a more informative RoI pooling —mainly for small objects—, the ConvNet applies upsampling to the last feature map of each of the branches of the net. In the experimental evaluation the smallest objects range from 25 to 50 pixels of height.

Yang *et al.* [21] separate the detection of objects of different sizes in different branches. Their proposal relies on scale-dependent pooling and, also on layer-wise cascades rejection classifiers in several branches for the different object sizes. This approach considers objects of less than 64 pixels of height as small targets.

In [14] authors propose a ConvNet, called Feature Pyramid Network (FPN), in which the deeper feature maps are upsampled and combined with shallower feature maps. Object detection relies on these combined feature maps: the shallower ones for small objects and the

[1]For reference, we use the NVIDIA Tesla P40 which has 24GB of memory.

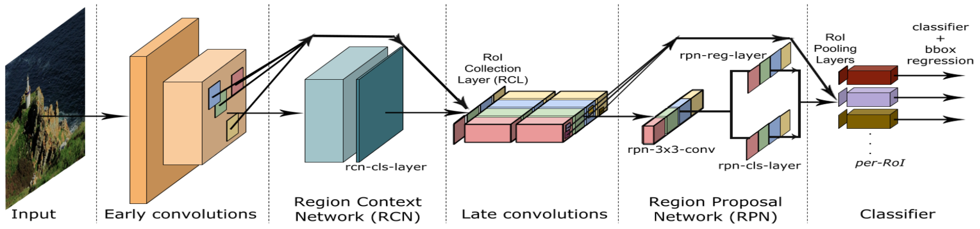

Figure 2: STDnet architecture. The RCN ends up with the RCL, which provides a feature map of the most likely candidate regions with objects surrounded by context. Late convolutions act on such regions independently thanks to inter-region 0-padding.

deeper ones for larger objects. FPN has 5 RPNs, and the shallower one —which is feeded by the shallower combined feature map, with stride 4— is the RPN that detects the small objects, obtaining outstanding results.

Our approach is focused on small targets as defined in this paper, i.e. under 16×16 pixels. This makes a big difference with the above works, as, firstly, many of the objects of interest do not feature definitive visual cues to classify them into a category and, secondly, our sizes are significantly smaller than those of the above solutions, making the object detection more difficult. In order to detect such small objects, the global effective stride must be low, which requires a new architecture to keep a reasonable memory overhead. Finally, as all the above solutions, our approach is an image object detector and, as such, it does not feature temporal information as the video object detectors reported in [19] and [6].

3 STDnet Architecture

STDnet is a fully convolutional network to detect small objects able to work with high resolution feature maps while keeping a low memory overhead. This is done by deleting the regions of the feature maps with the least likely objects and building filtered feature maps with the same resolution but lower memory requirements.

STDnet is a unified network that shares weights among all components as a progressive object abstraction representation. Figure 2 shows an overview of STDnet. It comprises five stages: early convolutions, Region Context Network (RCN), late convolutions, Region Proposal Network (RPN), and the classifier. The underlying feature extractor of STDnet —for early and late convolutions— can be any of the most widely state-of-the-art solutions found in the literature —e.g. ResNet [8], VGG [20], ZF [22], etc. In this work, ResNet-50 will be used, as it provides a good trade-off between accuracy, speed and GPU memory [8].

STDnet starts with a set of convolutional layers —early convolutions—, followed by the RCN, which looks for the most promising regions in the corresponding feature map. Regions are defined as areas of the image that might contain small objects together with their context. RCN assigns to each region a score, and the top scored regions are passed on to the final stage of the RCN, the RoI Collection Layer (RCL). RCL collects the input features to RCN and maps them onto such candidate regions, making up filtered feature maps. Late convolutions of STDnet act on the forwarded regions from RCN independently thanks to inter-region 0-padding —displayed as gaps between the different candidate regions in Figure 2 and Figure 3. The final RPN performs an initial bounding box regression and classification as object

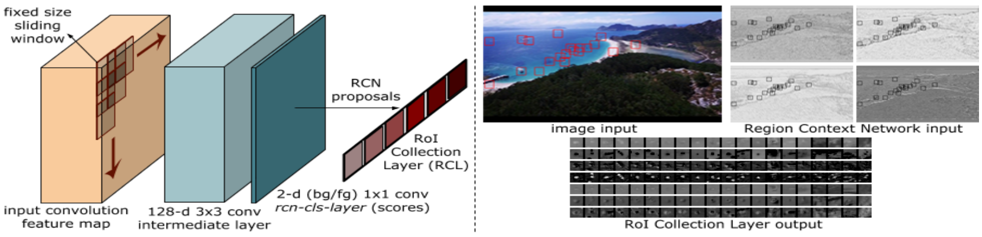
fixed size sliding window
input convolution feature map
128-d 3x3 conv intermediate layer
RCN proposals
RoI Collection Layer (RCL)
2-d (bg/fg) 1x1 conv rcn-cls-layer (scores)
image input
Region Context Network input
RoI Collection Layer output

Figure 3: RCN architecture (left) and some examples of its feature maps (right).

(fg) and background (bg) [18], which are finally refined in the classifier stage.

As seen, RCN avoids forwarding the regions of the input image with the least likely objects to the deepest convolutional layers, saving memory and increasing frame rate. Memory saving is key to increase spatial resolution through finer global effective strides across convolutional layers, mandatory in order not to miss the spatial localization of small objects.

3.1 Region Context Network (RCN)

The structure of RCN with its components can be seen in Figure 3. RCN selects the most likely candidate regions with one or more small objects together with their context, and returns them as a set of disjoint regions. As at this stage of the net the goal is not to get accurate object localization, the output regions' size will be the same for all of them and neither a box regression approach, nor a set of anchors with different scales and aspect ratios are needed. A single anchor of a given size suffices to return the most likely candidate regions with small objects. The first step in RCN applies a 3×3 convolutional filter to each window of the input feature map generating an intermediate 128-d layer with ReLU [16] following. This structure feeds a box-classification layer (*rcn-cls-layer*) represented by a 1×1 convolutional 2-d layer (fg, i.e. object, and bg, i.e. no object) which scores regions obtained with sliding-windows over the last early convolution.

To verify that the anchor is a positive or a negative candidate in each sliding window region during the RCN's training phase, the ground truths of the objects are grown proportionally in all directions until equaling the anchor's defined size. Then, those anchors that have a considerable overlap with the modified ground truth (greater than 0.8 by default) are assigned as positive labels, leaving for negative those regions that barely have an overlap (lower than 0.3 by default). As usual, we measure the overlap by the intersection-over-union (IoU) ratio. The objectness score of the candidate regions in RCN is minimized through:

$$L_{RCN}(\{p_i\}) = \underbrace{\frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*),}_{\text{object/not object classifier}} \tag{1}$$

where p_i is the predicted probability of the i-th anchor being an object in a RCN mini-batch, and p_i^* is the adapted ground-truth label. The term $\frac{1}{N_{cls}}$ normalizes the equation and it refers to the size of the RCN's mini-batch. L_{cls} is a softmax loss over object and not object categories.

3.1.1 RoI Collection Layer (RCL)

RCN ends up with the so-called RoI Collection Layer (RCL) (Figure 3). RCL layer takes as input the feature map generated by the last early convolution and the top scored proposals from RCN to return a single filtered feature map with the same information as that of the input feature map, but only for the set of selected regions. Successive convolutions with filters greater than 1×1 will affect the neighboring regions' outputs. To solve this problem RCL adds an inter region 0-padding —shown by gaps between regions in Figure 3.

3.2 Region Proposal Network (RPN)

The RPN module is based on the RPN presented in [18], but we include a set of modifications in order to deal with the fact that the coordinates of its input feature map do not correspond with those of the input image, i.e., the RPN input contains unsorted regions. To map the regions on the input image to the RPN's training function, which is based on the IoU between anchors and ground truth, RCN passes the 4 coordinates of every region as a parameter to the RPN to generate the anchors relative to those regions. Finally, the output of the bounding box regression is transformed to the input image coordinates.

3.2.1 Automatic anchors initialization by k-means

The approaches that rely on RPNs define the number of anchors and their sizes heuristically. In our proposal, both the number and the size of the anchors are learned through k-means. The k-means anchor learning procedure is implemented as a preprocessing stage of STDnet. k-means is applied to the training set of ground truth boxes' height and width. In order to obtain the number of kernels, which will be the number of anchors, we perform an iterative k-means with an increasing number of kernels until the maximum inter-kernels IoU exceeds a certain threshold. We have set this threshold to 0.5, which is the value used in well-known repositories, as PASCAL VOC [5] or MS COCO [13], to check if a detection is positive or negative with respect to a ground truth.

3.3 Implementation Details

As all STDnet learnable layers are convolutional and shared, both the network that acts as a backbone (ResNet-50) and the two modules of the network (RCN and RPN) can be trained end-to-end by backpropagation and stochastic gradient descent (SGD) [11]. In this paper, we have selected the approximate joint training [18].

The RCN module is trained in a similar way to RPN, except for bounding box regression, which does not exist here. The fact that RCL keeps the same number of output images per mini-batch as that of input images, makes the rest of the training identical to other RPN networks like Faster-R-CNN. The initialization of anchors by k-means does not affect training either, since it is performed previously to STDnet training. In the same way as the RPN, our RCN obtains its mini-batch from a single image by selecting positive and negative anchors. The mini-batch used within the RCN is 64 examples trying to maintain whenever possible a ratio of 1:1 of positive and negative labels. The anchor's size was obtained estimating the effective receptive field (ERF) which, in practice, follows a Gaussian distribution [15], so half of the theoretical receptive field of the convolutions between RCN and RPN was selected as ERF. In order to eliminate overlapping regions from those proposed by the RCN,

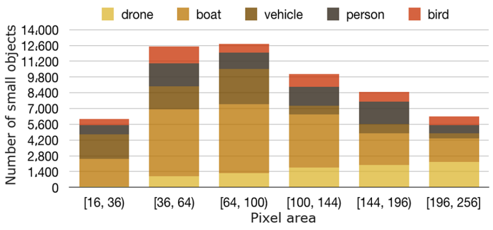

Figure 4: Statistics by object category and size in our USC-GRAD-STDdb database.

we apply an aggressive non-maximum suppression with a low threshold (0.3) over the 2,000 best proposals before the RCL, resulting in a low number of scattered regions —around 200 on average. At test, we let pass through the RCN those regions with confidence higher than 0.3, up to a maximum of 50 regions.

RCN and RCL can be integrated in any object detection convolutional framework just adapting the corresponding region proposal method to work with unsorted regions. In this paper we have implemented STDnet over Faster-R-CNN. The hyper-parameters for training and testing STDnet are the same as those used in Faster-R-CNN. The RPN module in STDnet is placed between convolutional layers *conv4* and *conv5* as it is done in [8] for Faster-R-CNN. Finally, at test, we apply a box-voting scheme after non-maximum suppression [7]. Our implementation uses the framework Caffe [10].

4 Experiments

4.1 Small Target Detection database (USC-GRAD-STDdb)

The Small Target Detection database (USC-GRAD-STDdb) [2] is a set of annotated video segments retrieved from YouTube. USC-GRAD-STDdb comprises 115 video segments with more than 25,000 annotated frames of HD 720p resolution ($\approx 1280 \times 720$) with small objects of interest from 16 ($\approx 4 \times 4$) to 256 ($\approx 16 \times 16$) as pixel area. Figure 1 shows a sample of USC-GRAD-STDdb. The length of the videos changes from 150 up to 500 frames. The total number of labeled small objects is over 56,000.

Figure 4 shows a histogram of the number of objects in each category and the pixel area of them in the dataset. Although USC-GRAD-STDdb has been generated by identifying the different categories of objects through human intervention, for the experiments carried out below, a single category of object will be used. As there are many potential small object candidates, we restrict to those targets that can potentially move, even they can be still in a given frame or set of frames. This makes room for future further improvement of our network through the inclusion of temporal features. Finally, the majority of the videos in USC-GRAD-STDdb were recorded by drones or with bird eye view over three main landscapes and five object categories, namely: air (drone, bird), 57 videos with 12,139 frames; sea (boat), 28 videos with 7,099 frames; and land (vehicle, person), 30 videos with 6,619 frames.

In the following experiments, 80% of the videos of USC-GRAD-STDdb were used for training (92 videos), while the remaining 20% were used for testing (23 videos), keeping as much a similar ratio as possible for the three different landscapes, object sizes and categories.

[2]https://citius.usc.es/t/usc-grad-stddb

Method	Anchors			$AP_{@.5}$	$AP_{@[.5,.95]}$
	Scales	Aspect ratios	#		
Faster-R-CNN[18]	$128^2, 256^2, 512^2$	1:1, 2:1, 1:2	9	19.3	5.2
Faster-R-CNN[18]	$8^2, 16^2, 32^2$	1:1, 2:1, 1:2	9	21.7	5.4
Faster-R-CNN[18]	$4^2, 8^2, 16^2$	1:1, 2:1, 1:2	9	20.3	5.4
Faster-R-CNN[18]+k	8×7, 14×10, 10×16, 21×9		4	**25.5**	**6.4**

Table 1: Performance of different RPN anchor scales compared to k-means on USC-GRAD-STDdb.

$RCN_{region\ size}$	Method	$Recall_{RCN}$	$AP_{@.5}$	Method	$Recall_{RCN}$	$AP_{@.5}$
32×32	STDnet-C2	91.8	54.5	STDnet-C3	93.9	**57.4**
48×48	STDnet-C2	95.2	**56.5**	STDnet-C3	96.6	57.0
64×64	STDnet-C2	95.4	55.8	STDnet-C3	96.6	56.2

Table 2: STDnet performance obtained by varying the size of the RCN output regions. $Recall_{RCN}$ stands for the percentage of objects belonging to regions correctly classified as positive by the RCN. C2 and C3 indicate the convolution block before the RCN.

4.2 Results

In order to compare the performance of the different networks, we have used four different metrics. The Average Precision ($AP_{@.5}$) gives the percentage of objects correctly detected, i.e., the objects for which there is at least 50% of IoU between the detected and the ground-truth bounding boxes [5]. Additionally, we measure $AP_{@[.5,.95]}$, which is the average AP when the percentage of IoU goes from 50% to 95% in 5% steps. We also report the *average number of false positives per image* (FPPI) when recall = 0.5 and the recall for FPPI = 1 [19].

Table 1 through Table 3 show the experimental results on USC-GRAD-STDdb. Our approach is compared to the state-of-the-art Faster-R-CNN [18] and FPN [14] as baseline networks. We report all metrics described above as well as the GPU memory and frame rate during testing. The global effective stride (GES) refers to the downscaling of the input image with respect to the feature map in the convolutional layer before the first RPN —*conv4* for Faster-R-CNN and *conv2* for FPN.

Table 1 compares the performance of Faster-R-CNN with its original global effective stride (16) for different values of the anchors of the RPN. The first row corresponds with the configuration for the Pascal VOC [18]; the anchors of the second and third rows have been adapted manually to the new database; finally, the last row uses the anchors selected by our proposal based on k-means. The k-means learning for USC-GRAD-STDdb results in just 4 defined anchors. All the evaluation metrics with heuristic anchors are below those met with our proposal based on k-means. From these results, in the experiments that follow, the baseline Faster-R-CNN will take as anchors those defined through our k-means proposal.

Table 2 studies the sizes of the regions in the RCN to assess that the estimation of the ERF is half of the theoretical receptive field. For STDnet-C2, as from *conv2* to *conv4* there are ten 3×3 convolutions with nonlinear activations in addition to the 3×3 RPN convolution, the theoretical receptive field is 23×23 between these blocks of convolutions. For STDnet-C3, the theoretical receptive field is 15×15 between *conv3* and *conv4*. Thus, regions of 48×48 —$\approx 12\times12$ with GES 4— and 32×32 —$\approx 8\times8$ with GES 4— will be used respectively. Results support this hypothesis. Larger regions pass more true objects, increasing the recall ($Recall_{RCN}$), but with less performance in AP because also more back-

Method	GES	AP$_{@.5}$	AP$_{@[.5,.95]}$	Recall	FPPI	fps	Mem.(GB)
Faster-R-CNN[18]+k	16	25.5	6.4	35.78	3.35	2.9	7.9
Faster-R-CNN[18]+k	8	44.0	14.4	50.73	0.95	2.6	10.8
Faster-R-CNN[18]+k	4	—	—	—	—	—	$>23.0_{train}$
FPN[14]	4	50.8	16.3	63.02	0.29	3.0	**6.9**
FPN[14]+k	4	50.7	16.8	59.14	0.31	3.5	**6.9**
STDnet-C2	4	56.5	17.9	64.03	0.32	**4.8**	7.2
STDnet-C3	4	**57.4**	**20.0**	**65.49**	**0.22**	3.7	10.6

Table 3: Evaluation metrics of Faster-R-CNN, FPN and STDnet on USC-GRAD-STDdb. +k indicates that the anchors were defined by the k-means algorithm.

ground is passed through. The key idea is that these regions should be as small as possible to preserve memory but, also, they have to contain the largest objects and exploit the ERF of deep convolutions between RCN and RPN.

Table 3 compares the performance of Faster-R-CNN, FPN and STDnet in our USC-GRAD-STDdb. It also shows the effect of placing the RCN in STDnet after the second or third convolutional blocks, namely STDnet-C2 and STDnet-C3. The deeper the RCN, the better the evaluation metrics, but at the cost of more memory usage and less frame rate.

For Faster-R-CNN, as expected, finer effective strides lead to better metrics. A GES of 4 in the baseline Faster-R-CNN exceeds the size of our GPU memory at training [3]. The STDnet allows to work with lower GES, outperforming Faster-R-CNN in AP$_{@.5}$ —57.4% vs. 44.0%— and AP$_{@[.5,.95]}$ —20.0% vs. 14.4%. STDnet also improves the FPPI 4.3$\times$ and the speed 1.4$\times$.

With respect to FPN, RPNs at different scales —with the first one at a feature map with GES = 4— lead to a higher performance than Faster-R-CNN, reaching 50.8%$_{@.5}$ and 16.3%$_{@[.5,.95]}$. Nonetheless, the upsampling performed from the deepest convolutions causes a lower performance compared to STDnet that reaches 57.4%$_{@.5}$ and 20.0%$_{@[.5,.95]}$. Moreover, the FPPI is 1.3$\times$ better for STDnet, which is also 1.2$\times$ faster. During the experimentation, we tested two configurations for the combination of FPN and k-means: a) the same anchors at each RPN; and b) the set of anchors distributed among the different RPNs. The best results for FPN+k (shown in Table 3) were obtained with the second option. Comparing FPN and FPN+k, the performance of FPN+k for AP$_{@.5}$ is slightly worse than FPN, because the first RPN in FPN already has a set of anchors suitable for small objects. Nevertheless, for AP$_{@[.5,.95]}$ FPN+k improves FPN by a 0.5%, which indicates that the k-means algorithm helps to generate better bounding boxes.

5 Conclusions

We have introduced STDnet, a region-proposal-based ConvNet to detect small targets under 16×16 pixels. The key of STDnet is an additional visual attention mechanism that we call RCN that chooses the most likely candidate regions with one or more small objects and their context. Such regions are forwarded across the network with filtered feature maps. RCN allows for finer effective strides that lead to greater precision while saving memory usage and increasing frame rate. We have also included an automatic definition of the anchors with k-means that improves the classical heuristic approach. STDnet has been tested on a

[3]The experiments have been run on an NVIDIA Tesla P40 GPU with 24GB.

new video data set, USC-GRAD-STDdb, with small objects in complex backgrounds with clutter. STDnet obtains the best results on USC-GRAD-STDdb with a 57.4% $\text{AP}_{@.5}$ and 20.0% $\text{AP}_{@[.5,.95]}$. Moreover, STDnet clearly outperforms its counterparts: Faster-R-CNN by $13.4\%_{@.5}$ and $5.6\%_{@[.5,.95]}$, and FPN by $6.6\%_{@.5}$ and $3.7\%_{@[.5,.95]}$.

Acknowledgments

This research was funded by Gradiant, and also partially funded by the Spanish Ministry of Economy and Competitiveness under grants TIN2017-84796-C2-1-R and TEC2015-66878-C3-3-R, and the Galician Ministry of Education, Culture and Universities under grant ED431G/08. Brais Bosquet is supported by the Galician Ministry of Education, Culture and Universities. These grants are co-funded by the European Regional Development Fund (ERDF/FEDER program). We thank Raquel Dosil Lago, David de la Iglesia Castro and Daniel González Jiménez from Gradiant for their collaboration. We thank NVIDIA for donating GPUs.

References

[1] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision (ECCV)*, pages 354–370. Springer, 2016.

[2] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 379–387, 2016.

[3] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761, 2012.

[4] Christian Eggert, Dan Zecha, Stephan Brehm, and Rainer Lienhart. Improving small object proposals for company logo detection. In *ACM on International Conference on Multimedia Retrieval*, pages 167–174. ACM, 2017.

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

[6] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3038–3046, 2017.

[7] Spyros Gidaris and Nikos Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1134–1142, 2015.

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

[9] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017.

[10] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *22nd ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.

[11] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[12] Jianan Li, Xiaodan Liang, Yunchao Wei, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. Perceptual generative adversarial networks for small object detection. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2017.

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014.

[14] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 4, 2017.

[15] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4898–4906, 2016.

[16] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *27th International Conference on Machine Learning (ICML)*, pages 807–814, 2010.

[17] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[18] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

[19] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. Detecting flying objects using a single moving camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(5):879–892, 2017.

[20] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[21] Fan Yang, Wongun Choi, and Yuanqing Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2129–2137, 2016.

[22] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833. Springer, 2014.

[23] Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, and Shimin Hu. Traffic-sign detection and classification in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2110–2118, 2016.