STDnet: Exploiting high resolution feature maps for small object detection

Brais Bosquet, Manuel Mucientes and Víctor Brea

Version: accepted article

Available online at www.sciencedirect.com

ScienceDirect

Engineering Applications of Artificial Intelligence (2020) 1–16

Journal Logo

STDnet: Exploiting high resolution feature maps for small object detection

Brais Bosquet*, Manuel Mucientes, Víctor M. Brea

Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS)
Universidade de Santiago de Compostela, Santiago de Compostela, Spain

Abstract

The accuracy of small object detection with convolutional neural networks (ConvNets) lags behind that of larger objects. This can be observed in popular contests like MS COCO. This is in part caused by the lack of specific architectures and datasets with a sufficiently large number of small objects. Our work aims at these two issues. First, this paper introduces STDnet, a convolutional neural network focused on the detection of small objects that we defined as those under 16×16 pixels. The high performance of STDnet is built on a novel early visual attention mechanism, called Region Context Network (RCN), to choose the most promising regions, while discarding the rest of the input image. Processing only specific areas allows STDnet to keep high resolution feature maps in deeper layers providing low memory overhead and higher frame rates. High resolution feature maps were proved to be key to increasing localization accuracy in such small objects. Second, we also present USC-GRAD-STDdb, a video dataset with more than 56,000 annotated small objects in challenging scenarios. Experimental results over USC-GRAD-STDdb show that STDnet improves the $AP_{@.5}$ of the best state-of-the-art object detectors for small target detection from 50.8% to 57.4%. Performance has also been tested in MS COCO for objects under 16×16 pixels. In addition, a spatio-temporal baseline network, STDnet-bST, has been proposed to make use of the information of successive frames, increasing the $AP_{@.5}$ of STDnet in 2.3%. Finally, optimizations have been carried out to be fit on embedded devices such as Jetson TX2.

Keywords: small object detection, convolution neural networks (ConvNets), deep learning

1. Introduction

In the last years, solutions to visual object detection have experienced a fast evolution. This evolution goes from primary approaches based on machine learning (Papageorgiou and Poggio, 2000; Felzenszwalb et al., 2010; Viola and Jones, 2001) to deep learning techniques (Ren et al., 2015; Liu et al., 2016; He et al., 2016; Redmon and Farhadi, 2017), boosted by publicly available datasets with millions of annotated images (Russakovsky et al., 2015; Lin et al., 2014; Everingham et al., 2010).

Generic image and/or video datasets have become valuable benchmarks to assess the quality and advance of object detectors. Nevertheless, datasets for more specific scenarios or applications are also a need. This situation might lead to new state-of-the-art object detectors, sometimes cutting across different topics. As an example, a timely trend is to apply ideas from the visual object tracking field (Fernández-Sanjurjo et al., 2019; Pang et al., 2017) to exploit temporal features in video through spatio-temporal networks (Carreira and Zisserman, 2017; Peng and Schmid, 2016).

In this line, 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 convolutional neural networks (ConvNets) object detectors, like the work in Lin et al. (2017a), provide high accuracy over a wide range of scales, from less than 32×32 pixels up to the image size. Qualitatively, we refer to small as those objects without definitive visual cues to assign them to a category. Quantitatively, small refers to sizes under 16×16 pixels. Figure 1a shows examples of this kind of objects where it can be seen that had it not been by the context around the foreground object, it would not be possible for a person to sort them out in a given category. Despite the existence of solutions focused on small objects, the most remarkable

*Corresponding author at: Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostela, 15782 Santiago de Compostela, Spain.

Email addresses: brais.bosquet@usc.es (Brais Bosquet), manuel.mucientes@usc.es (Manuel Mucientes), victor.brea@usc.es (Víctor M. Brea)

(a) Small objects as defined in this paper, i.e., under 16 × 16 pixels.

(b) Smallest objects available in public datasets.

Figure 1: Examples of 48 × 48 image patches from different databases with small objects.

ones have been validated in face detection datasets (Zeng et al., 2018; Zhang et al., 2017; Bai et al., 2018; Hu and Ramanan, 2017), but none of them have been tested in generic datasets such as MS COCO small objects subset (<32×32 area) (Lin et al., 2014).

When it refers to the scope of small targets, as those below 16 × 16 pixels as defined in this paper, the available set of datasets presents several drawbacks as shown in Figure 1b. The top row shows small objects extracted from the FlickrLogos dataset presented in Kalantidis et al. (2011). As apparent, this is a very specific image dataset. Also, their size exceeds that of 16 × 16 pixels, and it is possible to assign them to a given category. The middle row displays small objects from the image dataset introduced in Yang et al. (2016c). Again, this dataset is very specific and the small objects like faces are actually part of a whole person —which although in the case of faces are of interest by themselves, this is not the general case. Finally, the last row shows samples from a video dataset addressed in Rozantsev et al. (2017). These are monochrome and low quality videos, which, although relevant, are not the trend with the advent of ever better quality color image sensors. Another dataset with very specific small objects is Kestur et al. (2019).

As Liu et al. (2018) point out, detecting very small objects stands out as one of the key challenges in object detection. The main obstacle presented by state-of-the-art ConvNets is found at the region proposal level, which is inefficient when proposing valid regions for such small objects. This is given by the fact that the proposal generation of regions is applied on a very low resolution feature map, where the characteristics of small objects get too small to be detectable. One straightforward solution could be to modify a state-of-the-art ConvNet, disabling some backbone's downsampling to keep larger feature map resolutions. This, however, leads to large memory requirements, easily beyond the capacity of current high performance GPUs, and possibly to slower solutions. Another approach is to apply a region proposal generator directly in early stages of the network. However, this leads to the problem that the semantic information does not suffice to locate and classify the small objects appropriately.

This paper introduces STDnet, a novel ConvNet architecture for small object detection, along with a public video dataset with small targets in real-life scenarios, overcoming the drawbacks of current datasets aforementioned above. STDnet takes advantage of the following hypothesis: being such small objects almost visually unrecognizable, they have simple characteristics that can be learned by the neural network at early stages, where the semantic information is low but enough to select zones of interest containing small objects. To implement this hypothesis, an early high resolution feature map feeds a novel promising regions extractor called Region Context Network (RCN), which does not have to delimit the objects perfectly, but larger regions —promising areas— which contain them. This set of disordered regions is encapsulated in a unique and reduced feature map by RoI Collection Layer (RCL). Thanks to these two techniques, the algorithm is able to remove overhead by not processing a large part of the image from that point onwards, allowing to work with high resolution feature maps. Finally, a common region proposal method takes the feature map of disordered regions as input to obtain the final bounding boxes. The STDnet approach allows to consume less memory than its counterparts for higher resolution of the last feature map.

The main contributions of our proposal are:

1. STDnet, a new ConvNet for small object detection able to work with high resolution feature maps in deepest layers. STDnet relies on two novel components, RCN and RCL, which work together to select the most promising areas of the image, generating a new single filtered feature map with them. Therefore, the filtered feature maps can keep a high resolution but with a lower memory overhead and a higher frame rate.
2. As STDnet has a final region proposal method that works with anchor boxes, we propose to automatically select the number and sizes of these anchors

through a novel algorithm based on k-means. Our proposal differs from Redmon and Farhadi (2017) in that our algorithm selects not only the sizes of the anchors, but also their number, making the anchors selection fully automatic.

3. A new video database, USC-GRAD-STDdb, for small object detection to cover the indicated dataset gap. USC-GRAD-STDdb presents more than 56,000 annotated objects of sizes between 4×4 and 16×16 (e.g., Figure 2). USC-GRAD-STDdb comprises 115 video segments (>25,000 frames) over the three principal landscapes: air, sea and land.
4. A spatio-temporal evolution/version of STDnet, called STDnet-bST, to exploit video information of USC-GRAD-STDdb.
5. To bring STDnet closer to real on-board systems we have mapped both STDnet and STDnet-bST onto the embedded GPU Jetson TX2 through a series of computational optimizations.

2. Related Work

Modern object detectors are based on ConvNets (Huang et al., 2017b; Gu et al., 2018). One-shot and two-stage solutions are the two main configurations of ConvNets adopted today. The former has two principal baselines, namely, SSD (Liu et al., 2016) and YOLO (Redmon and Farhadi, 2017), which feature an excellent performance in computational cost and accuracy trade-off. As a drawback, both of them are outperformed by the two-stage approach when it refers to small objects (Liu et al., 2018; Huang et al., 2017b). There are three main reasons for this: (1) the generation of the bounding boxes takes place in deep layers with low spatial resolution, which misses the opportunity of locating small objects —e.g., SSD and YOLO apply a downsampling of $8\times$ and $32\times$, respectively, to the original image to locate the smallest objects; (2) the use of a fixed sampling grid that works worse when objects are too close to each other or are too small and (3) the lower accuracy, at all scales, mostly produced by the great class imbalance between foreground objects and background proposals as these detectors evaluate $\approx 10^4$ candidate regions per image (Lin et al., 2017b). It is worth noting that the last drawback can be partially addressed by solutions such as hard negative mining (Liu et al., 2016) or max-out background label (Zhang et al., 2017), and that RetinaNet (Lin et al., 2017b) outperforms these solutions with a novel cost function. All the above and our own experiments have made our research focus on the two-stage solution.

The two-stage approach was popularized by R-CNN (Girshick et al., 2014). Its extension Faster-R-CNN (Ren et al., 2015) has become a milestone with the introduction of the Region Proposal Network (RPN). This approach, also known as region based object detectors, uses the RPN to generate a set of candidate object locations based on anchor boxes, which are predefined regions of different sizes and aspect ratios to cope with multiple scales. At a second stage, the backbone's upper layers are applied to classify the candidate locations into object of interest or background, besides refining the bounding box.

Sharing the same problem as one-shot detectors, the off-the-shelf Faster-R-CNN is not adequate for small object detection due to the fact that the global effective stride (GES) —downscaling of the input image with respect to the feature map that is the input to the region proposal method— is 16, which means that a 16×16 object is represented by just one pixel in that feature map. In addition, the anchor boxes are predefined manually and they were not conceived to handle such small objects. To tackle small objects, a finer GES is required. This leads to a very large memory overhead, making the implementation impossible for current GPUs[1].

In Dai et al. (2016), authors propose an additional functionality to Faster-RCNN called Region-based Fully Convolutional Network (R-FCN). R-FCN exploits the different parts of a given object using position-sensitive maps. R-FCN generates $k\times k\times(C+1)$ feature maps —$k\times k$ object parts for C object categories plus background— instead of only one in the second stage detection. Additionally, R-FCN is fully convolutional, so it avoids the fully connected layers overhead. Still, this interesting improvement does not fix the GES problem and cannot be used to detect small objects, as their objects parts are very small, and as such indistinguishable in the input images.

The capability of dealing with objects of different sizes, and specially small ones, in Faster-R-CNN and R-FCN is limited due to low resolution feature maps and the few scales produced with the anchors as we outlined above. Hence, more recent ConvNets for object detection tackle scale invariance and small object detection with more elaborated solutions.

Yang et al. (2016b) propose a scale-dependent pooling along with layer-wise cascade rejection classifiers in several branches for the different object sizes. Then, if an object proposal has a height lower than 64 pixels, the regions are processed by a scale-dependent pooling with more resolution than if they were larger objects. Still, this proposal does not meet our needs for objects under 16×16 pixels.

In Li et al. (2017), authors focus on learning to transfer the information of small objects to similar large objects, what they call super-resolved objects. For this purpose they introduce a Perceptual Generative Adversarial Network for small object detection. The performance of this approach has been tested on the Tsinghua-Tencent 100k dataset (Zhu et al., 2016), considering small objects those smaller than 32×32, and on the Caltech benchmark (Dollár et al., 2012), with pedestrians over 50 pixels tall, so it does not suffice for objects under 16×16 pixels.

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

Figure 2: USC-GRAD-STDdb examples. Ground truth objects are enclosed in red boxes (best seen in color).

Eggert et al. (2017) propose a Faster-R-CNN based approach for small objects. Authors validate the proposal in the FlickrLogos dataset (Romberg et al., 2011) similar to the examples presented in Figure 1b (first row). The solution presents three levels of RPN which make use of feature maps with different resolution. To match different levels, high-level feature maps are upscaled through bilinear interpolation and then summed with the lower-level maps. Finally, classification and bounding box regression receive as inputs the combination of them. Similarly, in Cai et al. (2016) several RPNs are proposed with the shallowest RPN working with the smallest objects. In the experimental evaluation the smallest object size ranges from 25 to 50 pixels of height, which does not suffice our needs.

An effective solution to detect objects in different scales, approached by Bell et al. (2016), Kong et al. (2016) or Hariharan et al. (2017), is to combine layers with different resolutions throughout the ConvNet by creating a single convolutional block that gathers the information from upper layers into the bottom ones. A similar solution is adopted in Zeng et al. (2018) for face detection. This way, highly semantic information in wide receptive fields provided by the upper layers is combined with the low semantic information in narrow receptive fields. This combination is usually generated by skip connections, which discard some layers to connect more distanced ones. A single region proposal method takes this convolutional block as input to detect different scale objects.

Regarding the specific field of face detection, many advances have been made with outstanding results. In this line, Zhang et al. (2017) propose a modified SSD architecture which anticipates the object proposal to GES = 4, among other improvements. Bai et al. (2018) employ a generative adversarial network (GAN) to improve the resolution of blurry small faces. Finally, Hu and Ramanan (2017) study the crucial influence of context in detecting small objects while proposing a model that uses specific scale templates to detect faces of different sizes —including very small ones.

Joining information from several convolutional layers and using several RPNs at different scales, Lin et al. (2017a) introduce Feature Pyramid Network (FPN). FPN builds a neural network that joins several levels of convolutional blocks through lateral connections. In each level junction, an RPN adapted naturally to a different object scale is applied. The FPN architecture features several RPNs at different GESs. The shallower convolution block —which is fed by the shallower combined feature map, with a GES = 4— is the RPN that detects the small objects, obtaining outstanding results. Since its release, the proposal has become a milestone, being currently adopted as baseline for the researches that lead the top entries of the MS COCO challenge (MS COCO Leaderboard, 2019).

Applying the improvements provided by FPN, RetinaNet (Lin et al., 2017b) sheds light on how to improve performance metrics in the single-shot approach, including small objects. The work in Lin et al. (2017b) implements a simple FPN-based architecture that removes the RPNs and adds two subnetworks —class subnet and bbox subnet— to detect objects in one-stage. The main improvement is obtained through a novel loss function (Focal Loss) to address the class imbalance problem in single-shot detectors, leading to very promising results. The accuracy that reaches the proposal is similar to those obtained by the two-stage FPN, even in MS COCO small object scale (<32×32 area).

This paper focuses on small targets as defined in this paper, i.e., under 16×16 pixels. Such small object targets make us different from the previous approaches: our sizes are significantly smaller than those of the above solutions and the categories we are dealing with are large objects –car, person, boat, etc.— but they are located at such a great distance that most of them do not feature definitive visual cues to classify them into a category (Figure 1a), making the object detection more difficult. Based on the knowledge from the aforementioned research, our STDnet architecture generates candidate object locations proceeding over the deepest layers, which exploits their semantic information, but keeping high feature map resolution — GES = 4— which allows to deal with small targets successfully with a reasonable memory overhead.

3. STDnet architecture

STDnet is an unified neural network approach proposed to detect small objects under 16×16 pixels. To address this, STDnet invests time in selecting promising zones at shallower layers to discard the rest of the input image and, thus, improving the overall computing performance. This allows to keep high resolution feature maps

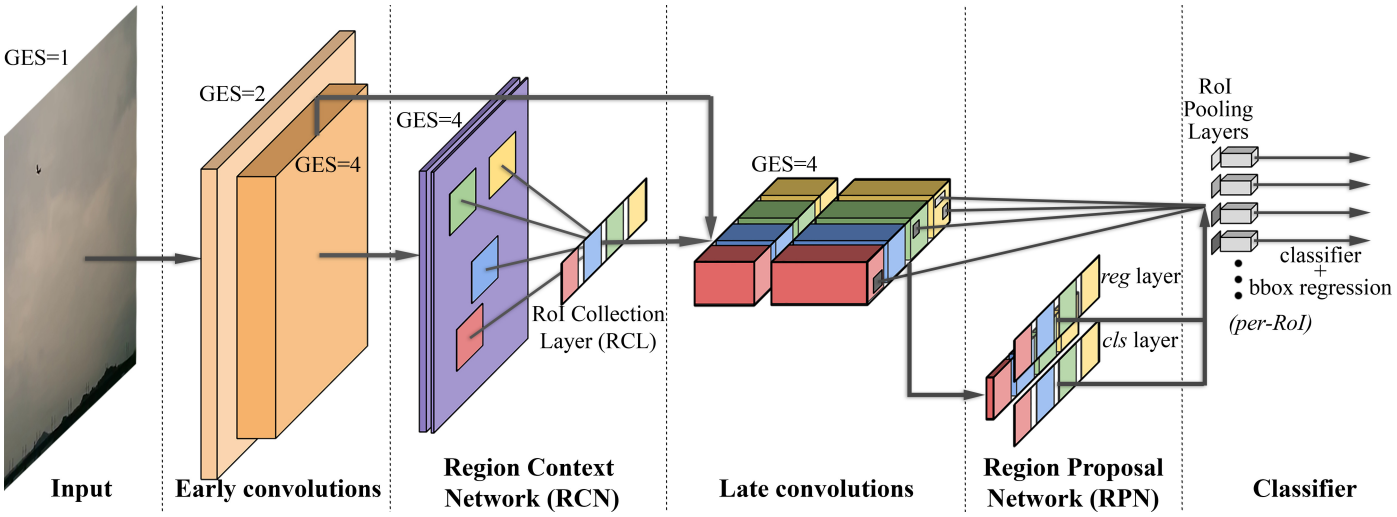

Figure 3: STDnet architecture. The RCN is placed after early convolutions to select only promising areas from the input image. Those zones are concatenated by the RCL so that the late convolutions act on the new feature map in a conventional way.

in these layers, especially favorable to detect small objects.

Figure 3 shows an overview of STDnet. It comprises five stages: early convolutions, Region Context Network (RCN), late convolutions, Region Proposal Network (RPN), and the classifier. Thereby, STDnet is presented as a Faster-R-CNN-based approach which employs ResNet-50 as a backbone —a good trade-off between accuracy, speed and GPU memory (He et al., 2016). ResNet is represented as early convolutions that comprise the shallowest layers and late covolutions that encompass the deepest ones. The backbone can be any of the most widely state-of-the-art solutions found in the literature —e.g., ResNet (He et al., 2016), DenseNet (Huang et al., 2017a), VGG (Simonyan and Zisserman, 2015), etc.

Like most state-of-the-art ConvNets methods, STDnet begins to learn simple features from the objects of interest in shallower convolutional layers, named here as early convolutions. Unlike other methods, just after the shallower convolutions, STDnet applies a novel detector of promising areas over the shallower feature map, RCN, to select regions that most likely contain small objects. Then, top scored regions are gathered in a single feature map by RCL for the deeper convolutional layers —late convolutions— to act as usual on the disjoint areas. There is a caveat, so that the convolutions between the different areas do not affect each other, RCL introduces a null 1px size padding —since ResNet applies convolution filters no greater than 3×3. This padding must be reset to 0-padding after each convolution higher than 1×1. In the late convolutions stage, the memory saved by ruling out not promising areas can be used to keep the feature map in high resolution at the same time that the semantic information is increasing. STDnet applies a single RPN that takes as input the fourth convolutional block (C_4), which contains the most promising areas provided by the RCN but with richer semantic information. The RPN proposes as output the locations of the objects more precisely, performing bounding box regression and classification as object and background inside those RCN promising areas. These regions are further refined in a final classifier. This architecture differs from cascaded region proposal approaches (Zhong et al., 2019; Yang et al., 2016a) in that the entire image information is not used during the whole computation, rather a new synthetic feature map is constructed only with the areas of the image that the RCN considers to be the most important.

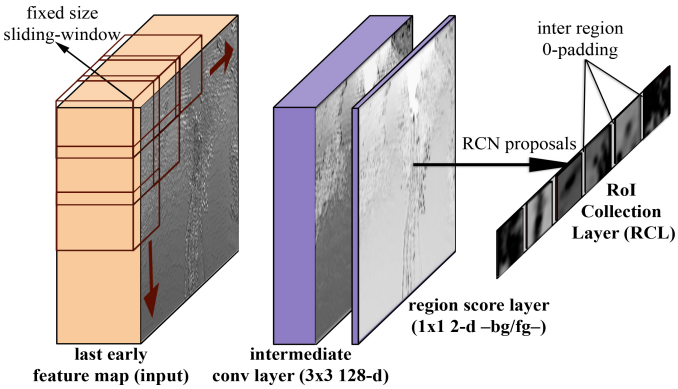

Figure 4: Region Context Network (RCN) architecture.

As a summary, the key methods of the designed architecture, RCN and RCL, allow to focus all efforts on promising areas, not having to pay attention to areas without relevant information. This allows to increase the resolution of the deeper layers and, even so, reduce the use of memory and increase the frame rate. STDnet does not increase the GES to more than 4 during the whole network, while the semantic information grows. Both, high resolution and high semantic information are crucial to detect small objects.

3.1. Region Context Network (RCN)

The Region Context Network (RCN) is a fully convolutional network that scans a shallow convolution map in order to detect fixed-size areas where there is most likely an object. The RCN architecture is shown in Figure 4. RCN selects the most likely candidate regions with one or more small objects together with their context. As at this stage 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.

RCN consists of a first 3×3 convolutional layer over the input layer to map it into an intermediate 128-d layer with ReLU (Nair and Hinton, 2010) following. This layer feeds a 1×1 convolutional 2-d layer to classify regions as foreground (fg), i.e., region with small objects inside, or background (bg), i.e., region without objects. RCN processes the region as a sliding-window on the feature map. The output are the scores associated with the different zones of the input image.

During the training phase, the bounding box ground truths must be grown proportionally in all directions until they reach the defined size to verify easily that the region under study represents a positive or a negative candidate. Then, as ground truths and regions have the same size, the overlap (measured by the intersection-over-union (IoU) ratio) between them is representative and can be calculated to assign each region a positive label, if IoU>0.8, or negative, if IoU<0.3. The objectness score of the candidate regions in RCN is minimized through:

$$L_{RCN}(\{p_i\}) = \underbrace{\frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*)}_{\text{fg/bg classifier}}, \quad (1)$$

where p_i is the predicted probability of the i-th region being foreground in an 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 cross-entropy loss over regions with or without objects (fg/bg).

3.1.1. RoI Collection Layer (RCL)

The promising regions generated by RCN cannot be processed separately due to the overhead that it entails and the need to modify the remaining backbone stages –late convolutions. Instead, a novel layer is implemented where RCN ends up, the so-called RoI Collection Layer (RCL) (Figure 4). 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. These regions will be concatenated in the new feature map in a disorderly way. 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 4.

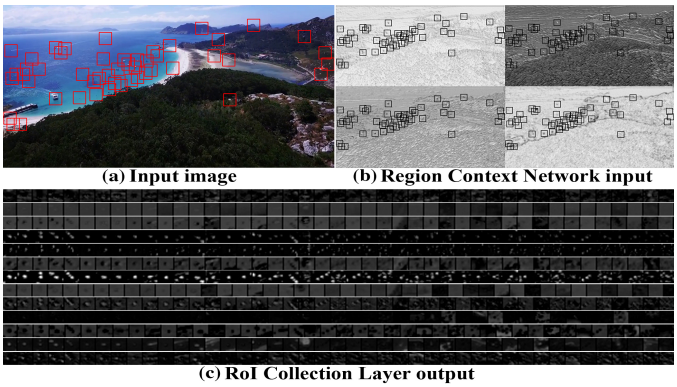

Figure 5: An example of the feature maps involved in the RCN.

With this configuration, the dimensions of the feature map output are obtained as follows:

$$RCL_{output\ size} = \underbrace{(r_w n + p_d(n-1))}_{\text{width}} \times \underbrace{r_h}_{\text{height}}, \quad (2)$$

where n is the number of regions from RCN, r_w and r_h are the dimensions of the regions in the RCL input feature map and p_d is the size of the 0-padding between regions. For example, a 1280×720 input image has an RCL input feature map of 320×180, and the output RCL generates a 649×12 feature map: 50 regions of size 48×48 at the input image —12×12 at the RCL input feature map for GES = 4— with $1px$ 0-padding in the example; i.e., a reduction of 7.4× of GPU memory usage —86.5% saved memory.

Figure 5 shows some examples of input and output feature maps of the RCN: Figure 5(a) original input image and the most promising regions proposed by the RCN; Figure 5(b) four of the input feature maps to the RCN and the most promising regions; and Figure 5(c) some feature maps composed by the RCL with the most promising regions, where each row represents a different channel. Each of the feature maps in Figure 5(b) generates a row in Figure 5(c). Each row in Figure 5(c) represents a feature map with the promising regions separated by 0-padding.

3.2. Region Proposal Network (RPN)

The Region Proposal Network (RPN) used in this paper is a modification from the one presented in Faster R-CNN (Ren et al., 2015) to deal with the feature map composed by RCL, i.e., the RPN input contains unsorted regions. In the original RPN, the anchors were processed linearly since the coordinates of its input feature map correspond with those of the input image but scaled. With the unpromising areas removed, this correspondence no longer exists and the correlation is not straightforward. RPN must take as input, besides the last feature map, the top scored promising regions information from the RCN 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 (Everingham et al., 2010) or MS COCO (Lin et al., 2014), to check if a detection is positive or negative with respect to a ground truth. This approach can be adopted by any other object detection network with anchors, e.g., Faster-R-CNN, regardless the target size of the objects.

A similar contribution was defined in Redmon and Farhadi (2017) where a k-means algorithm selects the anchors' size according to the dataset, but where the selection of the number of anchors is done manually, visualizing the best trade-off between the number of anchors and the average intersection of these with the dataset objects. Our approach makes the anchors selection completely automatic.

3.3. Implementation Details

In this paper, we adopt the approximate joint training (Ren et al., 2015) to train STDnet. To implement this end-to-end training, the ResNet-50 layers are shared with the two modules, RCN and RPN, so that all learnable layers can be trained by backpropagation and stochastic gradient descent (LeCun et al., 1989).

To train the RCN module, a mini-batch is obtained from a single input image by randomly selecting foreground and background regions. 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. In order to eliminate overlapping regions from those proposed by the RCN, 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. The RCN promising regions' fixed-size was obtained estimating the effective receptive field (ERF) which, in practice, follows a Gaussian distribution (Luo et al., 2016), so half of the theoretical receptive field of the convolutions between RCN and RPN was selected as ERF. The fact that RCN and RCL modules do not alter the global batch size, makes the rest of the training identical to other two-stage networks like Faster-R-CNN. The initialization of anchors by k-means does not affect training either, since it is performed once for each new dataset and previously to STDnet training.

RCN and RCL can be theoretically integrated in any object detection framework based on ConvNets, either one-stage or two-stage approach. The main modification in addition to the new modules is to adapt 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 C_4 and C_5 as it is done in He et al. (2016) for Faster-R-CNN. Finally, at test, we apply a box-voting scheme after non-maximum suppression (Gidaris and Komodakis, 2015). Our implementation uses the framework Caffe (Jia et al., 2014).

4. STDnet with spatio-temporal features (STDnet-bST)

STDnet detects objects using only the information coming from the current frame. Nevertheless, exploiting the information of a set of consecutive frames might help to improve detection, specially in those cases where the confidence of a detection is low. ConvNets that make detections based on a set of frames are called spatio-temporal networks (Carreira and Zisserman, 2017), in contrast to conventional ConvNets —also referred to as spatial ConvNets.

STDnet with spatio-temporal features for small target detection, namely, STDnet-bST, consists of two modules: (i) STDnet as a spatial detector, which provides a set of detections for the current frame; and (ii) the temporal module, which combines the detections of a set of frames and generates the final set of detections for the current frame.

The spatial module of the STDnet-bST, i.e., STDnet, feeds the temporal module with the set of detections of the current frame. The temporal module generates the data association among detections across the last T frames through the Viterbi algorithm (Gkioxari and Malik, 2015) based on the scores for all the spatio-temporal detections at the current frame, and building up what is known as *tubelet*. Figure 6 shows the composition of a single *tubelet* over T different frames given a set of detections in each one. The final spatio-temporal score in STDnet-bST is estimated with an approach similar to Peng and Schmid (2016), but computing the Viterbi algorithm at a *tubelet-level* object detection instead of at a video-level action detection.

The temporary score s_{tp} or temporal confidence between two detections d_t^i and d_{t-1}^j in two consecutive frames t and $t-1$ is given by:

$$s_{tp}(d_t^i, d_{t-1}^j) = s(d_t^i) + s(d_{t-1}^j) + \mathrm{IoU}(d_t^i, d_{t-1}^j), \qquad (3)$$

where $s(d)$ is the confidence returned by STDnet from detection d and IoU is the overlap, measured as the intersection over union, between both detections.

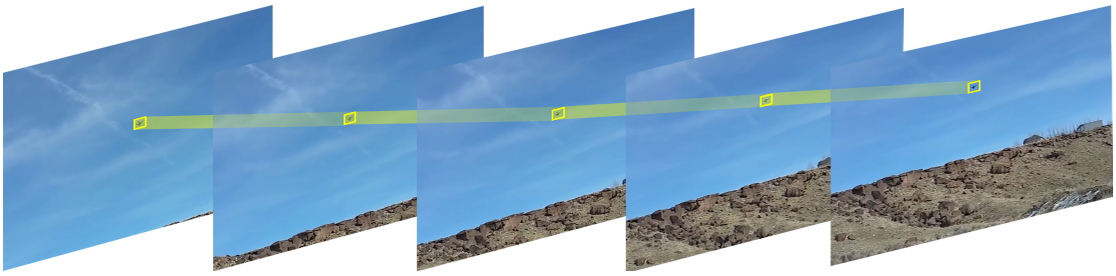

Figure 6: *Tubelet*, in yellow, generated by STDnet-bST through the Viterbi algorithm applied to the spatial detections from STDnet.

After calculating the score s_{tp} for all the combinations of detections throughout the T frames, the Viterbi algorithm is applied to obtain the most probable sequences of detections, i.e., *tubelets*, with size T. The Viterbi algorithm maximizes the conditional probability of the *tubelets* —each one represents an object seen at different time instants— given a set of detections over time. This process is executed recursively adding at each iteration the new set of all detections belonging to $t+1$ (d_{t+1}) and estimating the new set of *tubelets* (X_{t+1}) through the following equation:

$$\max_{x_1 \dots x_t} \mathbf{P}(\mathbf{x}_1, \dots, \mathbf{x}_t, \mathbf{X}_{t+1} | \mathbf{d}_{1:t+1}) = \alpha \mathbf{P}(\mathbf{d}_{t+1} | \mathbf{X}_{t+1}) \max_{x_t} \left(\mathbf{P}(\mathbf{X}_{t+1} | \mathbf{x}_t) \max_{x_1 \dots x_{t-1}} \mathbf{P}(\mathbf{x}_1, \dots, \mathbf{x}_{t-1}, \mathbf{x}_t | \mathbf{d}_{1:t}) \right) \tag{4}$$

where α is a normalization factor.

The final confidence (final score, s_f) for a detection $d_t^{i_t}$ will be given by the set of detections that formed the sequence $d_{Vit.} = [d_{t-T+1}^{i_{t-T+1}}, d_{t-T+2}^{i_{t-T+2}}, \dots, d_t^{i_t}]$, if $d_t^{i_t}$ belongs to a *tubelet*, or $s(d_t^{i_t})$, otherwise. We have experimented with several options as the mean, maximum or median of the detections' score to compute s_f, being the mean the most accurate one:

$$s_f(d_t^{i_t}) = \frac{1}{T} \sum_{j=t-T+1}^{t} s(d_j^{i_j}) \tag{5}$$

5. Experiments

In this section, we release our Small Target Detection database (USC-GRAD-STDdb), and we conduct extensive experiments for our approach and previous state-of-the-art works. We also assess STDnet on the 80 category Microsoft COCO 2017 detection dataset (Lin et al., 2014). Finally, a series of computational optimizations are made over STDnet and STDnet-bST to map them into an embedded GPU.

5.1. The 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 1a and Figure 2 show some samples 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 7 shows a histogram of the number of objects in each category and their pixel area (see Table 1 for more details). 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, so that the output of the STDnet is either object or background. As there are many potential small object candidates, we restrict to those targets that can potentially move, even though they can be still in a given frame or set of frames. The videos in USC-GRAD-STDdb comprise the three main landscapes with 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.

5.2. Evaluation Metrics

USC-GRAD-STDdb has been evaluated with four different metrics for all networks under study:

- 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, averaged over categories (Everingham et al., 2010).
- $AP_{@[.5,.95]}$, which is the average AP when the percentage of IoU goes from 50% to 95% in 5% steps, as reported in MS COCO (Lin et al., 2014).
- The *average number of false positives per image* (FPPI) when recall = 0.5, and the Recall for FPPI = 1 (Rozantsev et al., 2017). The Recall measures the

[2] USC-GRAD-STDdb is publicly available under request.

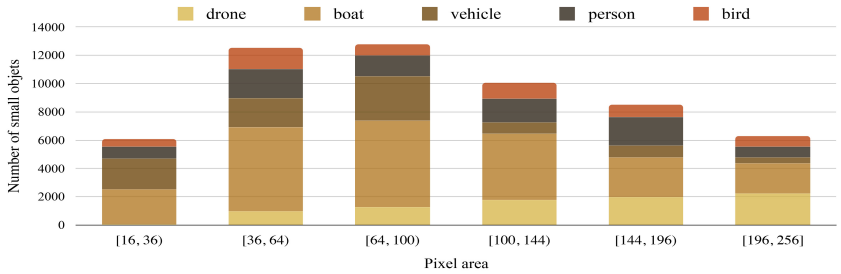

Figure 7: Statistics by object category and size in the USC-GRAD-STDdb database.

Area	[16,36]	(36,64]	(64,100]	(100,144]	(144,196]	(196,256]	[16,256]
# objs	6,074	12,513	12,759	10,056	8,497	6,303	56,202
% db	10,8%	22,3%	22,7%	17,9%	15,1%	11,2%	100%

Table 1: Statistics of the number of objects in the USC-GRAD-STDdb database according to their size.

ratio of true object detections to the total number of objects in the dataset for a given confidence threshold. In this case, for the confidence threshold that obtains exactly FPPI = 1.

Additionally, in the case of MS COCO, we report the COCO-style metrics, i.e., $AP_{@.5}$, $AP_{@[.5,.95]}$, $AR_{@.5}$ and $AR_{@[.5,.95]}$.

All the above metrics are calculated on the basis of precision (P) and recall (R), whose definitions are:

$$P = \frac{TP}{TP + FP}$$
$$R = \frac{TP}{TP + FN}, \tag{6}$$

where TP stands for true positives, FP for false positives and FN for false negatives for a given IoU threshold. Then, with the output detections ranked by confidence, each one is assigned to a label (TP, FP or FN), generating a set of precision-recall points —they can be represented in a precision-recall curve as Figure 8(left).The Average Precision (AP) is given by finding the area under the precision-recall curve for each category and averaged over all categories. The Average Recall (AR) is the maximum recall value obtained for each category and averaged over all categories.

5.3. Results on USC-GRAD-STDdb

Table 2 through Table 5 show the experimental results on USC-GRAD-STDdb with the spatial network, i.e., STDnet. Our approach is compared to the state-of-the-art Faster-R-CNN (Ren et al., 2015), FPN (Lin et al., 2017a) and RetinaNet (Lin et al., 2017b) networks. FPN is the base of the top 3 entries of 2018 COCO object detection challenge (MS COCO Leaderboard, 2019). 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 shallower RPN —C_4 for Faster-R-CNN and C_2 for FPN. Regarding RetinaNet, the original paper indicates that they do not use the high resolution feature map C_2 to locate objects for computational reasons, but the experiments on the USC-GRAD-STDdb report that starting at C_3, where GES = 8, the performance is very poor since the objects are too small. Therefore, a configuration more similar to that of FPN to locate objects has been selected.

Table 2 compares the performance of Faster-R-CNN with its original GES (16) for different values of the anchors of the RPN. The first row corresponds with the configuration for the Pascal VOC (Ren et al., 2015); the anchors of the second and third rows have been adapted manually to the new database, USC-GRAD-STDdb; 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 3 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 C_2 to C_4 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 C_3 and C_4. Thus, regions of 48×48 —$\approx 12 \times 12$ with GES 4— and 32×32 —$\approx 8 \times 8$ with GES 4— will be used, respectively. Results support this hypothesis. Larger regions pass more true objects, increasing the recall of RCN ($Recall_{RCN}$), but with a lower AP because also more background 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

Method	Anchors			$AP_{@.5}$	$AP_{@[.5,.95]}$
	Scales	Aspect ratios	# anchors		
Faster-R-CNN(Ren et al., 2015)	128^2, 256^2, 512^2	1:1, 2:1, 1:2	9	19.3	5.2
Faster-R-CNN(Ren et al., 2015)	8^2, 16^2, 32^2	1:1, 2:1, 1:2	9	21.7	5.4
Faster-R-CNN(Ren et al., 2015)	4^2, 8^2, 16^2	1:1, 2:1, 1:2	9	20.3	5.4
Faster-R-CNN(Ren et al., 2015)+k	8×7, 14×10, 10×16, 21×9		4	**25.5**	**6.4**

Table 2: Performance of different RPN anchor scales compared to k-means on USC-GRAD-STDdb.

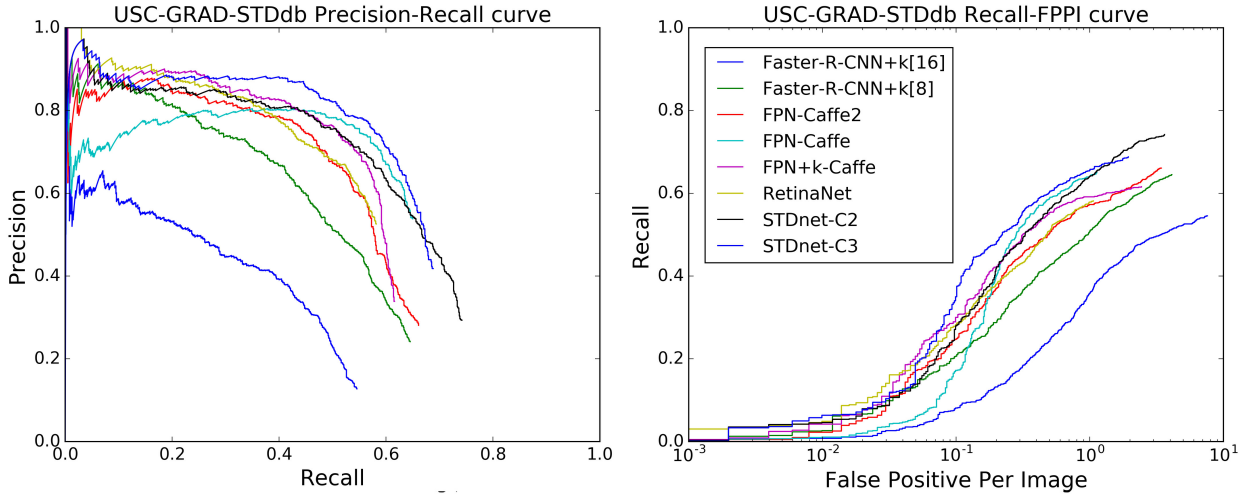

Figure 8: Precision-recall (left) and recall-FPPI (right) curves. The numbers inside the brackets indicate the global effective stride (GES).

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

Table 3: STDnet performance obtained by varying the size of the RCN output regions.

objects defined as small targets, and exploit the ERF of late convolutions between RCN and RPN.

Table 4 and Figure 8 compare the performance of Faster-R-CNN, FPN, RetinaNet and STDnet on 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, respectively. 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. GES = 4 in the baseline Faster-R-CNN exceeds the size of our GPU memory at training. 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× and the speed rate 1.4×.

When it comes to the FPN, the performance of two different implementations have been reported on USC-GRAD-STDdb. FPN (Lin et al., 2017a) runs on Caffe2, in the same repository as RetinaNet[3], and they take advantage of its speed performance and memory optimization improvements. FPN (Caffe) is programmed in Caffe framework, starting from the Faster-R-CNN official code[4] —just like STDnet. FPN in Caffe and Caffe2 provide similar results in AP, as seen in Table 4.

The architecture of FPN presents RPNs at different scales —with the first one at the C_2 level with GES = 4— , which leads to 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, which reaches 57.4%$_{@.5}$ and 20.0%$_{@[.5,.95]}$. Moreover, the FPPI is 1.3× better for STDnet, which is also 1.2× faster. During the experimentation, we tested two configurations for the combination of FPN and k-means: (i) the same anchors at each RPN; and (ii) the set of anchors distributed among the different RPNs. The best results for FPN+k (shown in Table 4) 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.

RetinaNet works with GES = 4 as FPN, and it obtains

[3]https://github.com/facebookresearch/Detectron
[4]https://github.com/rbgirshick/py-faster-rcnn

Method	GES	$AP_{@.5}$	$AP_{@[.5,.95]}$	Recall	FPPI	fps	Mem.(GB)
Faster-R-CNN(Ren et al., 2015)+k	16	25.5	6.4	35.78	3.35	2.9	7.9
Faster-R-CNN(Ren et al., 2015)+k	8	44.0	14.4	50.73	0.95	2.6	10.8
Faster-R-CNN(Ren et al., 2015)+k	4	—	—	—	—	—	$>24.0_{train}$
FPN(Lin et al., 2017a)	4	49.2	16.6	57.28	0.48	7.6*	2.8*
FPN	4	50.8	16.3	63.02	0.29	3.0	**6.9**
FPN+k	4	50.7	16.8	59.14	0.31	3.5	**6.9**
RetinaNet(Lin et al., 2017b)	4	47.6	16.2	57.87	0.47	6.5*	3.1*
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 4: Evaluation metrics of Faster-R-CNN, FPN, RetinaNet and STDnet on USC-GRAD-STDdb. +k indicates that the anchors were defined by the k-means algorithm. The computational entries denoted by "*" run on Caffe2 framework, so the comparison with Caffe implementations is not fair in terms of fps and memory consumption.

competitive results using a one-stage approach. Nevertheless, the need to place the shallowest set of anchors in C_2 lowers both the accuracy and the computational performance (Lin et al., 2017b).

Finally, Table 5 shows the results for different object sizes of small targets as defined in Section 5.1. As expected, the smaller the size of the objects, the lower the performance. STDnet outperforms FPN in 5 out of the 6 object sizes for both $AP_{@.5}$ and $AP_{@[.5,.95]}$ metrics. We highlight that STDnet is over 20% in $AP_{@[.5,.95]}$ for most of the object segments. $AP_{@[.5,.95]}$ is a very meaningful metric as it encompasses AP as IoU reaches perfection.

As addressed in Section 4, STDnet-bST includes temporal features through *tubelets* built up with the Viterbi algorithm. To determine the STDnet-bST hyperparameters, we used the USC-GRAD-STDdb training set. The impact of the number of frames (T) is analyzed in Figure 9 which shows a comparison between the time needed to process the temporal stage for each frame and the AP for different number of frames. $T = 4$ presents a good trade-off between computation time and accuracy.

To sum up the achieved performance, Table 6 shows a comparison between STDnet and STDnet-bST. As seen, STDnet-bST outperforms STDnet in all metrics. The processing time is practically identical, reaching 3.7 fps, since the overhead added by the temporal module to the STDnet is negligible.

5.4. Results on MS COCO 2017

MS COCO (Lin et al., 2014) is a popular image dataset for object detection with 108,556 small objects defined as those objects with an area of less than 32 × 32 pixels. We have defined a new scale subset —*extrasmall* (AP^{xs})— within the category small objects of COCO to include small targets as defined in this paper, i.e., those enclosed in bounding boxes with less or equal than 256 pixels of area —not the segmentation area as in the original annotations. As we have defined our own subset, we cannot evaluate the results with the official COCO test-dev 2017. Instead, we train with COCO train 2017 and evaluate with the popularly extended COCO val 2017 (5k) (Bell et al., 2016). Considering this, the total amount data used is: 62,658 of 236,574 objects from COCO train 2017 and 2,562 of 10,000 objects from COCO val 2017.

There are a few minor changes that should be made in STDnet for this dataset. The images are re-scaled such that their shortest side is 600 as in the baseline Faster R-CNN (Ren et al., 2015). Also, the RCN output regions have a size of 64×64 due to both the re-scaling and the extremely elongated nature of some categories —such as book, or skateboard. To conclude, we work with a mini-batch size of 128 regions to train RCN.

Table 7 shows the performance comparison between our approach, its baseline Faster-R-CNN, FPN and RetinaNet. Here, we report the MS COCO evaluation metrics AP^{xs} and AR^{xs} for the *extrasmall* subset. In the same way as in the USC-GRAD-STDdb, the STDnet obtains much better results than its baseline Faster-R-CNN, improving $AP_{@.5}$ by more than 2× and $AP_{@[.5,.95]}$ by 3×, while surpassing by 14.0% and 9.0% in $AR_{@.5}$ and $AR_{@[.5,.95]}$, respectively. Comparing STDnet with the state-of-the-art FPN, it is observed how the detections provided by STDnet suit better to the ground truth, yielding 0.7% ($AP_{@[.5,.95]}$) and 1.4% ($AR_{@[.5,.95]}$) higher than FPN detections. This metric is considered the most important —primary challenge metric— by MS COCO (Lin et al., 2014) because it encompasses AP adding information on how it behaves as the IoU reaches perfection. Finally, as expected, RetinaNet obtains lower performance than FPN and STDnet due to the same reasons mentioned for the dataset USC-GRAD-STDdb. In addition, the MS COCO presents objects very close to each other, which causes a disadvantage for the one-stage approaches.

It should be noted that the object detection MS COCO dataset, despite being the most complete and used repository in the field, features some issues when we refer to very small objects, which affects performance metrics.

The first issue is the lack of annotations when a large number of objects of the same class are grouped. Some of these occurrences are solved with the *iscrowd* label in the annotation, but in some others this label does not exist or, if it does, it is incorrect. Some examples are displayed on Figure 10a, where COCO annotations are shown.

The second issue is the existence of parts of large ob-

AP_{IoU}	Method	[16,36]	(36,64]	(64,100]	(100,144]	(144,196]	(196,256]
@.5	FPN	22,74	48,70	**65,48**	71,24	61,22	65,12
	STDnet-C3	**27,14**	**53,50**	61,33	**76,88**	**69,83**	**76,06**
@[.5,.95]	FPN	6,48	11,89	**21,69**	23,75	22,19	20,60
	STDnet-C3	**7,79**	**14,17**	20,93	**26,37**	**29,41**	**31,06**

Table 5: STDnet and FPN performances for different object sizes (area in pixels) of the USC-GRAD-STDdb database.

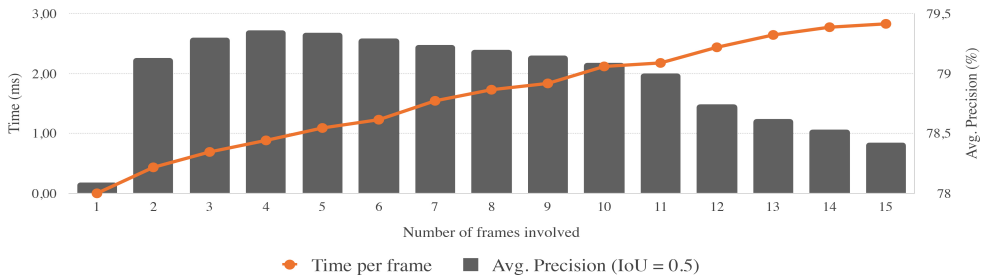

Figure 9: Viterbi algorithm runtimes and the average precision obtained for different values of T in the USC-GRAD-STDdb training set.

Method	$AP_{@.5}$	$AP_{@[.5,.95]}$	Recall	FPPI
STDnet	57.4	20.0	65.49	0.22
STDnet-bST	**59.7**	**20.6**	**66.81**	**0.20**

Table 6: Performance of STDnet-bST compared to STDnet over USC-GRAD-STDdb database.

Method	$AP^{xs}_{@.5}$	$AP^{xs}_{@[.5,.95]}$	$AR^{xs}_{@.5}$	$AR^{xs}_{@[.5,.95]}$
Faster-R-CNN	5.0	1.5	22.0	7.6
FPN	**11.8**	4.8	**36.7**	15.9
RetinaNet	9.1	4.5	33.0	16.2
STDnet-C3	11.4	**5.5**	36.0	**17.3**

Table 7: Evaluation metrics of Faster-R-CNN, FPN, RetinaNet and STDnet on the *extrasmall* objects of MS COCO val 2017 subset, i.e., objects under 256 pixels of area.

jects labeled as small objects with an *extrasmall* size for being largely occluded. Some of these examples are shown in Figure 10b. This poses a challenge for any detection algorithm. Nevertheless, our approach suffers more from this issue than FPN since STDnet features a receptive field considerably smaller than that of the FPN, as the size of the feature maps do not change in STDnet after passing through the RCN.

5.5. Execution on Embedded GPUs

Embedded GPUs are oriented to on-board platforms and, as such, they feature a limited computing capacity when compared to their GPU desktop counterpart. Therefore, it is necessary to reduce memory consumption and to improve computation time to migrate STDnet and STDnet-bST to embedded GPUs. The optimization carried out in this work is based on the unification of the contiguous blocks of convolution and batch normalization methods at test stage (Hong et al., 2016).

The description of this merger method uses the Caffe's notation (Jia et al., 2014). The batch normalization step from Ioffe and Szegedy (2015) also included a per-channel learned bias and scaling factor so, in Caffe's implementation, it is splitted into two layers named batch normalization and scale layers with the following parameters.

- Convolution layer: convolutional weights (c_w) and convolutional bias (c_b).
- Batch Normalization layer: global mean (bn_{mean}), global variance (bn_{var}) and moving average factor (bn_{norm}).
- Scale layer: scaling factor (s_w) and per-channel bias (s_b).

Considering the above notation, the three layers — convolution, batch normalization and scale— can be unified at test stage without altering the final result as follows:

1. A β vector is computed as a multiplier factor for the convolutional data:
$$\beta = \frac{s_w}{\sqrt{\frac{bn_{var}}{bn_{norm}+\epsilon}}} \tag{7}$$
where ϵ is a small value added to the variance estimate to avoid division by zero.
2. The convolutional weights c_w and bias c_b trained can be updated as:
$$\begin{aligned} c_w(i) &= \beta(i)c_w(i) \\ c_b &= \beta c_b + \left(s_b - \beta\frac{bn_{mean}}{bn_{norm}}\right) \end{aligned} \tag{8}$$
3. The batch normalization and scale layers can be bypassed by resetting their parameters as default, which is the same as removing those layers.

(a) Images with non-labeled objects or incorrect labels where only the category of interest is marked.

(b) Images with objects parts with extrasmall size where only the category of interest with extrasmall size is marked.

Figure 10: Some examples of controversial small labels on MS COCO val 2017. Normal objects are colored green and *iscrowd* objects are colored red (best seen in color).

Method	STDnet-C2		STDnet-C3	
	mem. (GB)	fps	mem. (GB)	fps
STDnet	7.22	4.80	10.61	3.73
STDnet opt.	4.29	5.75	5.95	4.59

Table 8: Comparison of STDnet and the optimized version of STDnet in memory consumption and computation time (fps).

Method	VGA		HD 720p	
	mem. (GB)	fps	mem. (GB)	fps
STDnet-C2	4.76	1.41	5.19	0.77
STDnet-C3	4.84	1.23	5.98	0.59

Table 9: Performance in memory consumption and computation time, shown in frames per second (fps), over the Jetson TX2 architecture for VGA and HD 720p images.

Table 8 shows the performance of the optimizations on HD 720p images for the two versions of STDnet implemented, with the RCN layer after the two and three convolution blocks of the network (STDnet-C2 and STDnet-C3, respectively). These metrics have been measured on a high-performance cluster GPU. The memory is reduced by 61.4% for the STDnet-C2 version and by 65.5% for STDnet-C3, in addition to improving the computation time by 52.2% for STDnet-C2 and 40.9% for STDnet-C3.

The NVIDIA Jetson TX2 (Franklin, 2017) has been selected as an embedded and portable device to perform the tests. This architecture has a middle-low graphic card (NVIDIA Pascal with 256 cores) and a limited memory of 8GB shared between the CPU and the GPU. The performance on this device has been evaluated both for HD 720p images —the originals of the USC-GRAD-STDdb database— and VGA images. In the case of VGA images, to perform the tests, segments of size 640×480 were selected from videos in the database that contained some small objects. The computational performance results on Jetson TX2 are shown in Table 9. As seen, the memory used differs slightly from that of a cluster GPU due to their different memory management procedures.

6. 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. 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.

In addition, we have released a new video dataset, USC-GRAD-STDdb, with more than 56,000 annotated small objects in complex backgrounds with clutter. STDnet obtains the best results on USC-GRAD-STDdb with a 57.4% $AP_{@.5}$ and 20.0% $AP_{@[.5,.95]}$, clearly outperforming its counterparts. STDnet-bST even improves these results without adding overhead, from 57.4% $AP_{@.5}$ to 59.7% $AP_{@.5}$. Furthermore, we have tested our approach with the *extrasmall* objects that exist in MS COCO, where the overall performance of STDnet is very competitive.

Finally, we have deployed STDnet and STDnet-bST on an embedded GPU, the Jetson TX2. For that, we have implemented a number of optimizations that allow to run both ConvNets on Jetson TX2 by reducing the consumed memory by 59% and increasing the fps by 20%.

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 RTI2018-097088-B-C32 (MICINN), 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

Bai, Y., Zhang, Y., Ding, M., Ghanem, B., 2018. Finding tiny faces in the wild with generative adversarial network. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 21–30.

Bell, S., Lawrence Zitnick, C., Bala, K., Girshick, R., 2016. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2874–2883.

Cai, Z., Fan, Q., Feris, R., Vasconcelos, N., 2016. A unified multi-scale deep convolutional neural network for fast object detection. In: European Conference on Computer Vision (ECCV). pp. 354–370.

Carreira, J., Zisserman, A., July 2017. Quo vadis, action recognition? A new model and the kinetics dataset. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4724–4733.

Dai, J., Li, Y., He, K., Sun, J., 2016. R-FCN: Object detection via region-based fully convolutional networks. In: Advances in Neural Information Processing Systems (NIPS). pp. 379–387.

Dollár, P., Wojek, C., Schiele, B., Perona, P., 2012. Pedestrian detection: An evaluation of the state of the art. IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (4), 743–761.

Eggert, C., Zecha, D., Brehm, S., Lienhart, R., 2017. Improving small object proposals for company logo detection. In: ACM International Conference on Multimedia Retrieval (ICMR). pp. 167–174.

Everingham, M., Van Gool, L., Williams, C., Winn, J., Zisserman, A., 2010. The PASCAL Visual Object Classes (VOC) Challenge. International Journal of Computer Vision 88 (2), 303–338.

Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D., 2010. Object detection with discriminatively trained part-based models. IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (9), 1627–1645.

Fernández-Sanjurjo, M., Bosquet, B., Mucientes, M., Brea, V. M., 2019. Real-time visual detection and tracking system for traffic monitoring. Engineering Applications of Artificial Intelligence 85, 410–420.

Franklin, D., 2017. NVIDIA Jetson TX2 delivers twice the intelligence to the edge. NVIDIA Accelerated Computing Parallel Forall, (2017).

Gidaris, S., Komodakis, N., 2015. Object detection via a multi-region and semantic segmentation-aware cnn model. In: IEEE International Conference on Computer Vision (ICCV). pp. 1134–1142.

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 580–587.

Gkioxari, G., Malik, J., 2015. Finding action tubes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 759–768.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., et al., 2018. Recent advances in convolutional neural networks. Pattern Recognition 77, 354–377.

Hariharan, B., Arbelaez, P., Girshick, R., Malik, J., 2017. Object instance segmentation and fine-grained localization using hypercolumns. IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (4), 627–639.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778.

Hong, S., Roh, B., Kim, K.-H., Cheon, Y., Park, M., 2016. PVANET: Deep but lightweight neural networks for real-time object detection. arXiv preprint arXiv:1611.08588, (2016).

Hu, P., Ramanan, D., 2017. Finding tiny faces. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 951–959.

Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K., 2017a. Densely connected convolutional networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4700–4708.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al., 2017b. Speed/accuracy trade-offs for modern convolutional object detectors. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, (2015).

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T., 2014. Caffe: Convolutional architecture for fast feature embedding. In: ACM International Conference on Multimedia. pp. 675–678.

Kalantidis, Y., Pueyo, L., Trevisiol, M., van Zwol, R., Avrithis, Y., 2011. Scalable triangulation-based logo recognition. In: ACM International Conference on Multimedia Retrieval (ICMR). p. 20.

Kestur, R., Meduri, A., Narasipura, O., 2019. Mangonet: A deep semantic segmentation architecture for a method to detect and count mangoes in an open orchard. Engineering Applications of Artificial Intelligence 77, 59–69.

Kong, T., Yao, A., Chen, Y., Sun, F., 2016. Hypernet: Towards accurate region proposal generation and joint object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 845–853.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L., 1989. Backpropagation applied to handwritten zip code recognition. Neural Computation 1 (4), 541–551.

Li, J., Liang, X., Wei, Y., Xu, T., Feng, J., Yan, S., 2017. Perceptual generative adversarial networks for small object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S., 2017a. Feature pyramid networks for object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Vol. 1. p. 4.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2017b. Focal loss for dense object detection. In: IEEE International Conference on Computer Vision (ICCV). pp. 2980–2988.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C. L., 2014. Microsoft coco: Common objects in context. In: European Conference on Computer Vision (ECCV). pp. 740–755.

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., Pietikäinen, M., 2018. Deep learning for generic object detection: A survey. arXiv preprint arXiv:1809.02165 (2018).

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A., 2016. SSD: Single shot multibox detector. In: European Conference on Computer Vision (ECCV).

Luo, W., Li, Y., Urtasun, R., Zemel, R., 2016. Understanding the effective receptive field in deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NIPS). pp. 4898–4906.

MS COCO Leaderboard, 2019. Microsoft COCO detection leaderboard. http://cocodataset.org/#detection-leaderboard, accessed: 2019-02-08.

Nair, V., Hinton, G., 2010. Rectified linear units improve restricted boltzmann machines. In: International Conference on Machine Learning (ICML). pp. 807–814.

Pang, S., del Coz, J. J., Yu, Z., Luaces, O., Díez, J., 2017. Deep learning to frame objects for visual target tracking. Engineering Applications of Artificial Intelligence 65, 406–420.

Papageorgiou, C., Poggio, T., 2000. A trainable system for object detection. International Journal of Computer Vision 38 (1), 15–33.

Peng, X., Schmid, C., 2016. Multi-region two-stream r-cnn for action detection. In: European Conference on Computer Vision (ECCV). pp. 744–759.

Redmon, J., Farhadi, A., 2017. YOLO9000: Better, faster, stronger. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6517–6525.

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems (NIPS). pp. 91–99.

Romberg, S., Pueyo, L., Lienhart, R., van Zwol, R., 2011. Scalable logo recognition in real-world images. In: ACM International Conference on Multimedia Retrieval (ICMR). pp. 25:1–25:8.

Rozantsev, A., Lepetit, V., Fua, P., 2017. Detecting flying objects using a single moving camera. IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (5), 879–892.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al., 2015. Imagenet large scale visual recognition challenge. International Journal of Computer Vision 115 (3), 211–252.

Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (ICLR).

Viola, P., Jones, M., 2001. Rapid object detection using a boosted cascade of simple features. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Yang, B., Yan, J., Lei, Z., Li, S. Z., 2016a. Craft objects from images. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6043–6051.

Yang, F., Choi, W., Lin, Y., 2016b. 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). pp. 2129–2137.

Yang, S., Luo, P., Loy, C., Tang, X., 2016c. Wider face: A face detection benchmark. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Zeng, D., Zhao, F., Ge, S., Shen, W., 2018. Fast cascade face detection with pyramid network. Pattern Recognition Letters 119, 180–186.

Zhang, S., Zhu, X., Lei, Z., Shi, H., Wang, X., Li, S. Z., 2017. S3fd: Single shot scale-invariant face detector. In: IEEE International Conference on Computer Vision (ICCV). pp. 192–201.

Zhong, Q., Li, C., Zhang, Y., Xie, D., Yang, S., Pu, S., 2019. Cascade region proposal and global context for deep object detection. Neurocomputing.

Zhu, Z., Liang, D., Zhang, S., Huang, X., Li, B., Hu, S., 2016. Traffic-

sign detection and classification in the wild. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2110–2118.