A Full Data Augmentation Pipeline for Small Object Detection based on Generative Adversarial Networks

Brais Bosquet[a], Daniel Cores[a], Lorenzo Seidenari[b], Víctor M. Brea[a], Manuel Mucientes[a,*], Alberto Del Bimbo[b]

[a] *Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostela, Santiago de Compostela, Spain*
[b] *Media Integration and Communication Center (MICC), University of Florence, Italy*

Abstract

Object detection accuracy on small objects, i.e., objects under 32×32 pixels, lags behind that of large ones. To address this issue, innovative architectures have been designed and new datasets have been released. Still, the number of small objects in many datasets does not suffice for training. The advent of the generative adversarial networks (GANs) opens up a new data augmentation possibility for training architectures without the costly task of annotating huge datasets for small objects. In this paper, we propose a full pipeline for data augmentation for small object detection which combines a GAN-based object generator with techniques of object segmentation, image inpainting, and image blending to achieve high-quality synthetic data. The main component of our pipeline is DS-GAN, a novel GAN-based architecture that generates realistic small objects from larger ones. Experimental results show that our overall data augmentation method improves the performance of state-of-the-art models up to 11.9% $AP_s^{@.5}$ on UAVDT and by 4.7% $AP_s^{@.5}$ on iSAID, both for the small objects subset and for a scenario where the number of training instances is limited.

Keywords: small object detection, data augmentation, generative adversarial

*Corresponding author
Email addresses: braisbosquet@gmail.com (Brais Bosquet), daniel.cores@usc.es (Daniel Cores), lorenzo.seidenari@unifi.it (Lorenzo Seidenari), victor.brea@usc.es (Víctor M. Brea), manuel.mucientes@usc.es (Manuel Mucientes), alberto.delbimbo@unifi.it (Alberto Del Bimbo)

Figure 1: We describe a pipeline for small object data augmentation able to populate an original frame (left) with new generated small objects (right), highlighted in red.

network

1. Introduction[1]

The accuracy of object detectors has experienced a lot of progress year on year with the release of large training datasets and the continuous improvement of CNNs architectures [1, 2], which goes along with the ever increasing computing power of GPUs.

In this line, small object detection stands out as a field of its own with increasing interest [3, 4, 5]. This is mainly because many downstream tasks demand early detections of objects to act quickly: self-driving cars or applications like sense and avoid on UAVs need to detect as far an object as possible, or satellite image analysis, where almost all objects are just a few pixels in size. That is, all the previous applications require objects to be identified as soon as possible, i.e, when they are barely visible in the images. Recent CNN-based object detectors, like the work in [3], provide high accuracy over a wide range of scales, from less than 32×32 pixels up to the image size. Despite these improvements, existing solutions often underperform with small objects [6].

The problems of detecting such small objects are twofold: (i) in deep CNNs architectures commonly the deeper the feature map, the lower the resolution,

[1] Abbreviations list: unmanned aerial vehicle (UAV); convolutional neural network (CNN); generative adversarial network (GAN); downsampling GAN (DS-GAN); feature pyramid network (FPN); high resolution (HR); low-resolution (LR); synthetic low resolution (SLR); intersection over union (IoU); Frechet inception distance (FID).

which is counterproductive when the object is so small that it may be lost along the way, and (ii) the most popular datasets such as MS COCO [7] or ImageNet [8] focus their attention on larger objects. While to deal with the first problem new solutions are being proposed year by year [4, 3, 9, 10], the second is being tackled mostly with the tedious task of generating new datasets [4, 11, 12, 13].

We have noticed some reasons that call for a superior number of small objects in public datasets to train a small object detector. First, the relatively fewer images that contain small objects will potentially bias any detection model to focus more on medium and large objects. In addition, the scarce features in small objects hinder the model generalization, lacking a great deal of variability. Finally, the smaller the object the more places it can appear, increasing the object background diversity, demanding more context variability at training.

Moreover, pieces of evidence [14] have shown that good data augmentation can boost deep models to achieve state-of-the-art performance without changing the network architecture. Although data augmentation has shown to significantly improve image classification, its potential has not been thoroughly investigated for object detection. So, given the additional cost for annotating images for object detection, data augmentation may play an essential role in boosting performance of generic object detection.

The advent of the GANs [15] has led to a new approach in the field of data augmentation. This kind of models are trained in an adversarial manner, where one network (the generator) tries to cheat another network (the discriminator) by generating new images. The generator attempts to provide examples that are increasingly similar to those in the real world.

Data augmentation for object detection presents two major challenges: (i) the generation of new objects and (ii) the integration of those objects to adapt them to the new scenarios. The former is mostly tackled by reusing already existing objects in different positions [5] or by adjusting their scale by re-scaling functions [16]. However, it has been proven that common re-scaling functions cause artefacts that significantly distort the re-scaled object if compared to real-world objects [17, 18]. The latter could be approached by object segmentation

methods [2] to clear the original background and then insert the objects into plausible positions while tuning for color consistencies. In the case of small objects, there is the added issue that the performance of the segmentation methods decreases dramatically. In addition, many popular datasets [4, 11, 12] do not contain segmentation ground truth to train the segmentation models properly.

For all these reasons, in this paper we propose a full pipeline for small object data augmentation. Our pipeline takes a video dataset as input[2] and returns the same dataset but with new synthetic small objects (Fig. 1). The hypothesis is that, starting from the visual features of larger objects—which can be found in many datasets in a large number—high quality synthetic small objects can be generated and placed into an existing image. To do so, the pipeline has the following stages: (i) to generate small objects from large ones through a GAN; (ii) to seek a logical position within the image through optical flow; (iii) to integrate small object via inpainting and blending techniques. The main contributions described in this paper are:

- A full pipeline for small object data augmentation which is able to automatically generate small objects using larger ones and place them into an existing background in a congruent fashion.
- Downsampling GAN (DS-GAN), a generative adversarial network architecture that converts large size objects into high quality small objects.
- An extensive experimentation on the video dataset UAVDT [11] and the image dataset iSAID [19], where the base results of state-of-the-art approaches are improved.

2. Related Work

The small object data augmentation approach we present in this paper is based on several computer vision tasks. The execution flow starts with a GAN

[2]The input to the pipeline can also be an image dataset. That only requires minor modifications to the pipeline, as explained in Sec. 3.2.1.

that generates synthetic small objects from larger ones. This process can be seen as solving the opposite of image super-resolution. Then, a segmentation network obtains the pixels of the input object and this mask is adapted to the new generated small object. In parallel, the new position in the image is obtained exploiting optical flow. The synthetic object may or may not replace an existing small object in the image. If so, the real one is removed from the scenario via inpainting. Finally, the object is placed into the selected position and tuned by image blending to fit the new background.

Small object detection. Small object detection refers to improving the detection of those objects with small size and poor visual features, typically defined as the detection of objects with a size below 32×32 pixels [7]. The actual trend for common object detection is to go deeper to recognise more complex semantics [1], but small objects, that do not contain detailed visual features, may be lost in the deep network. More sophisticated architectures, such as the FPN [3] or the Region Context Network [4, 20], partially alleviate this problem.

Furthermore, another restriction is the fact that popular datasets have focused on larger objects, with small objects underrepresented [7, 8]. To some extent, this restriction has been reduced by the advent of video datasets like UAVDT [11], VisDrone2019-VID [12] and, especially, USC-GRAD-STDdb [4], which are video datasets with a large percentage of small objects.

Adversarial learning attempts to fool models through malicious input or adversarial attacking through two —or more— networks with contrasting objectives. So that, these samples could be added to the training set to improve weak spots in the learned decision boundary. The principles of adversarial training have led to the popular GANs. The model consists of two networks that are trained in an adversarial process where, iteratively, one network (the generator) generates fake images and the other network (the discriminator) discriminates between real and fake images. So that the adversarial loss forces the generated images to be, in principle, indistinguishable from real ones.

A way to increase small object detection accuracy is to improve object res-

olution, for example with a Perceptual GAN [21] or an SOD-MTGAN [22]. A similar technique based on GANs has been proposed to improve the detection of tiny faces [23] or small-scale pedestrians [24]. Our approach is different as it downsamples objects for data augmentation in the training set, and it has the advantage that the GAN only has to be executed during the training process. The previous proposals require the execution of the GAN generator also during the inference —detection of the small objects—, as their detector needs super-resolved images.

Data augmentation. Data augmentation strategies are widely used for training vision models to minimize the bias between the training and the testing subsets, i.e., leading to more generalized models. There are two main types of data augmentation: basic image manipulations and generative synthetic approaches. Basic manipulations are simple operations, so deep learning designs usually combine many of them. For object detection, image mirroring and object-centric cropping are the most widely used [25].

One straightforward solution to generate synthetic objects is to augment the number of small object instances by randomly copy-pasting them [5, 26]. The problem of this approach is twofold: (i) the features of the object remain the same, and (ii) the position and scale of the object may not fit the context —e.g., a car in the sky. The second issue is addressed in [16] by an adaptive augmentation strategy called AdaResampling that logically augments the instances. AdaResampling generates a prior context map using a segmentation CNN and then places the objects in accordance with the scale and position. Yet, [17, 18] show that the object features produced by conventional resizing functions are far from real-world object features.

In [27] they increase the number of person instances in a given dataset through two modules: shape-guided deformation and the environment adaptation. The former one produces data augmentation by changing the shape of a given entry person. The latter adapts the person to the background through blending. However, they keep the resolution of the objects, thus not addressing the resolution mismatch problem.

Another solution is to learn the space of possible augmentations with adversarial training. In [28], authors introduce PTGAN to transfer persons between datasets to tackle the classical domain gap issue. However, sizes of objects in the pair of datasets match, so that solutions like CycleGAN [29] with additional constraints can be used without downsampling. Moreover, in their setting, people are large enough objects to be segmented accurately, making it possible to feed PTGAN with such segmentations. Also, PTGAN does not deal with the object positioning, i.e, where to include the new person in the image of the dataset where the transfer is made.

DetectorGAN [30] is based on CycleGAN, which performs image-to-image translation, transforming object free images to images with objects and viceversa. Nevertheless, DetectorGAN does not place objects in a coherent location in the image, and it has not been tested with small objects.

Image super-resolution. Image super-resolution comprises the task of estimating an HR image from its LR counterpart. The techniques to achieve the final image can use a series of consecutive frames of a video or a single image. Multiple image-based (or classical) solutions are mostly reconstruction-based algorithms that try to address aliasing artefacts by simulating the image formation model. These models are highly dependent on the motion estimation between the LR images, so they are more unstable in real-world applications [31]. Henceforth, we will describe only single image super-resolution approaches.

Before the emergence of convolutional neural networks, single image super-resolution techniques ranged from simple prediction-based methods, which yield solutions with overly smooth textures, going through methods that attempt to address these shortcomings by exploiting different priors. With the remarkable CNN success, all efforts were turned in this direction. Within this scope, Dong *et al.* [32] used bicubic interpolation to upscale an input image and feed a three layer deep fully convolutional network to achieve state-of-the-art SR performance. The definition of a perceptual loss [33], instead of low-level pixel-wise error measures, represented a significant improvement. The perceptual loss function applies an L_2 loss over calculated feature maps using another pre-

trained CNN —such as VGG— to increase the perceptual similarity, which leads to recover visually more convincing HR images. More recently, GANs boosted even more the image super-resolution results. In [34], authors introduce a GAN trained with the perceptual loss in cooperation with the adversarial loss to infer photo-realistic natural images for 4× upscaling factors.

In spite of the progress obtained with GANs, to train these networks it is necessary to have pairs of LR and HR images. Most of the approaches use bilinear interpolation to obtain the LR images, which is shown in [17, 18], but they cannot produce good results for real-world low-resolution images. To address this, Bulat *et al.* [17] defined two consecutive GANs, where the first GAN learns how to degrade HR images to LR images, and the second GAN uses these LR images to learn the standard image super-resolution.

Image Inpainting. Image inpainting is a conservation process where damaged, deteriorated, or missing parts are filled in to present a complete image. In the same way as in image super-resolution, the establishment of the GANs has lead to better inpainting results, as the discriminator forces the generator to fill with coherent data within the dataset. Specifically, Pathak *et al.* [35] introduced a Context Encoder trained with both L2 pixel-wise reconstruction loss and generative adversarial loss as the objective function to complete large center regions of fixed size. More recently, Yu *et al.* [36] propose a novel contextual attention layer to borrow features from distant spatial locations during training to improve the final performance.

Image blending. The goal of image blending is to create a composite image from the superposition —partial or full— of one or more source images, optimizing the spatial and color consistencies in order to make the composite image look as natural as possible. A specific instance of image blending is when a foreground region from a source image is pasted into the target background at a specified location. The default way is to copy pixels from the source image and paste them onto the target image, but this would generate obvious artefacts because of the abrupt intensity change in the compositing boundaries.

Burt and Adelson [37] introduced Laplacian pyramid, a multiresolution rep-

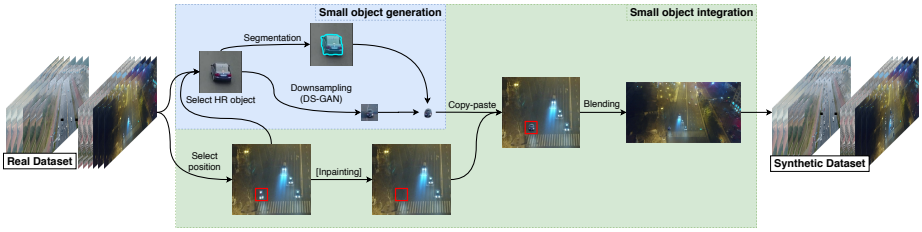

Figure 2: The proposed pipeline for data augmentation for small object detection. It takes a video dataset and produces the same frames but populated with synthetic small objects. The system comprises two main steps: small object generation and integration. This is repeated for each position/HR object pair.

resentation of the images of interest. The source images are decomposed into a set of band-pass filtered component images, then joined within each resolution band independently and, finally, adding up the different levels. So that, when coarse features occur near borders, these are blended gradually over a relatively large distance without blurring or otherwise degrading finer image details in the neighborhood of the border.

3. Small object data augmentation

Fig. 2 shows the architecture of the pipeline for data augmentation for small object detection. The purpose of this architecture is to increase the number of small objects in a video dataset. Our system consists of two procedures: the *small object generation*, which involves object downsampling and object segmentation, and the *small object integration* into the image, which involves position selection, object inpainting and object blending.

Through these components the system is able to generate SLR objects from real HR objects; these SLR objects will have similar features to real LR objects. Then, they are inserted in plausible positions within the image without enforcing any temporal consistency between frames. The following are the steps performed by the pipeline applied to an input video dataset (Fig. 2):

- The *small object generation* procedure produces SLR objects and their corresponding masks from HR objects.

1. The **object downsampling** generates an SLR object from an HR object with its context.
2. The **object segmentation** calculates the input HR object segmentation mask and transforms it to fit the SLR object.

- The *small object integration* procedure selects the optimal positions for the SLR object and inserts it into the image.
  1. The **position selector** selects the possible positions where some real LR objects exist —or existed in previous or successive frames— and optimizes the position and SLR object matching by comparing the direction and shape of both LR and HR objects through optical flow and overlap.
  2. The **object inpainting** deletes the objects that will be replaced.
  3. The **object blending** makes a copy-paste of each SLR object in the matched position and performs a blending operation to alleviate the abrupt boundary change and color intensity on the scene.

The final result provided by our system is a new dataset created with the same video images but populated with an increased number of SLR objects that replace the fixed number of LR objects.

3.1. Small object generation

3.1.1. Downsampling GAN (DS-GAN)

We have designed a Downsampling GAN (DS-GAN) to overcome the poor performance from well-known methods like bilinear interpolation or nearest neighbor to obtain SLR objects. DS-GAN is a generative adversarial network that learns to correctly degrade HR objects into SLR objects to increase the training set for object detection.

In this downsampling problem the aim is to estimate an SLR object from an input HR object with a downsampling factor r. The problem to solve is an unpaired problem where HR objects do not have a corresponding LR pair, but

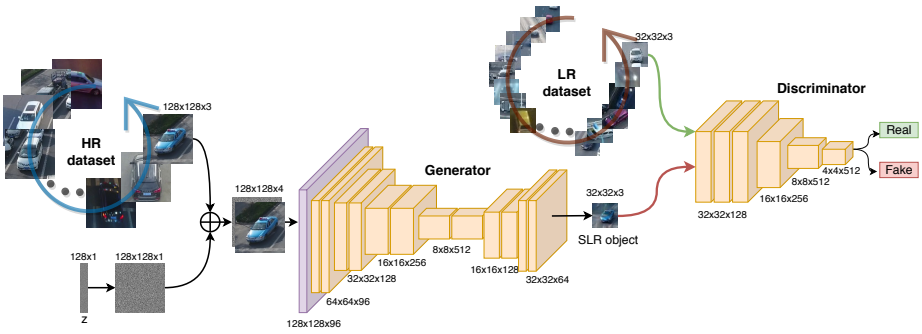

Figure 3: Downsampling Generative Adversarial Network (DS-GAN) architecture. The generator is trained with HR objects to synthesize small objects. A discriminator between real and fake small objects forces the generator to produce synthetic objects that are increasingly similar to real-world small objects.

the network would have to learn the distribution of the features of the whole LR subset while keeping similar visual appearance of the original HR object. For an image with C color channels, HR has size $W \times H \times C$ while both LR and SLR are described by $\frac{W}{r} \times \frac{H}{r} \times C$. So, for training the proposed GAN, two different image sets are required: (i) the *HR subset* composed of real large objects (HR objects) and (ii) the *LR subset* composed of real small objects (LR objects). Both the LR and HR subsets can be taken from the same dataset or from any additional one if more samples are needed.

Our DS-GAN architecture is shown in Fig. 3. The generator network (G) takes as input an HR image concatenated with a noise vector (z) and produces an SLR image 4× smaller than the input ($r = 4$). For example, a 128×128 object will lead to a 32×32 object. The noise vector is randomly sampled from a normal distribution and it is attached to the input image. This allows to produce numerous SLR objects from a single HR object, thus modeling the fact that the HR image will be affected by multiple types of LR noise. Following the methodology of [15] we further define a discriminator network (D) which we optimize in an alternating manner along with the generator (G).

The generator is an encoder-decoder network —see Fig. 3— composed of six groups of residual blocks [1]. Each group has two same-dimension residual blocks with pre-activation and batch normalization as defined in [38]. To achieve

a 4× downscaling, four 2× down-sample steps performed by pooling layers are placed at the end of each of the first four groups and two 2× up-sample steps performed by deconvolution layers at the end of each of the last two groups.

The discriminator —see Fig. 3— follows the same residual block structure (without batch normalization) followed by a fully connected layer and a sigmoid function. The discriminator comprises six residual blocks with two 2× down-sample steps. The details of the composition of both architectures are better shown in Fig. 3.

With this architecture, our goal is to train G to generate an SLR sample conditioned on an HR sample. To achieve this, the objective function chosen for the adversarial loss is the hinge loss [39]:

$$l^D_{adv} = \mathop{\mathbb{E}}_{s\sim\mathbb{P}_{LR}}[\min(0, 1 - D(s))] + \mathop{\mathbb{E}}_{\hat{s}\sim\mathbb{P}_G}[\min(0, 1 + D(\hat{s}))] \tag{1}$$

where $\mathbb{P}_{LR}$ is the LR subset distribution and $\mathbb{P}_G$ is the generator distribution to be learned through the alternative optimization. $\mathbb{P}_G$ is defined by $\hat{s} = G(b, z) \mid b \in \mathbb{P}_{HR}$, where $\mathbb{P}_{HR}$ is the HR subset. The general idea behind this formulation is that it allows to train G with the goal of fooling D, that is trained to distinguish SLR from LR images. With this approach our generator can learn to create SLR samples that are highly similar to real LR images, and thus difficult to classify by D.

Correspondingly, we train G by optimizing a loss function $\mathcal{L}$, defined as:

$$\mathcal{L} = l_{pixel} + \lambda l^G_{adv}, \tag{2}$$

where l^G_{adv} is the adversarial loss, l_{pixel} is the L_2 pixel loss, and λ is a parameter that balances the weight of both components.

The adversarial loss l^G_{adv} is defined based on the probabilities of the discriminator as:

$$l^G_{adv} = -\mathop{\mathbb{E}}_{b\sim\mathbb{P}_{HR}}[D(G(b, z))], \tag{3}$$

where $\mathbb{P}_{HR}$ is the HR subset and z is the noise vector. The adversarial loss is computed in an unpaired way, using the LR subset to make the SLR objects to

be contaminated with real-world artefacts.

The l_{pixel} minimizes the L_2 distance between the input HR and the output SLR:

$$l_{pixel} = \frac{r^2}{WH} \sum_{i=1}^{\frac{W}{r}} \sum_{j=1}^{\frac{H}{r}} (AvgP(b)_{i,j} - G(b,z)_{i,j}) \mid b \in \mathbb{P}_{HR}, \tag{4}$$

where W and H denote the input HR size, r is the downsampling factor and $AvgP$ is an average pooling function that maps the HR input to the output $G(b,z)$ resolution. The l_{pixel} is computed in a paired way between the SLR object and the HR object downsampled to the output SLR resolution using an average pooling layer. This component aims to keep the appearance of the synthetic objects similar to the original HR objects.

In addition, to solve the stabilization of the discriminator training we normalize its weights by the spectral normalization technique [39].

3.1.2. Object Segmentation

To integrate the SLR object in a new scenario, it is mandatory to extract the foreground object from its background. The approach chosen for object segmentation is to adapt the Mask R-CNN framework[3] [2] trained on the public dataset MS COCO to obtain the mask from HR objects (Fig. 4). As the segmentation results for small objects have a poor performance, we propose to get the mask from the large objects and fit it to the small objects. This is done just by resizing by factor r. This is possible because the pixel loss (Equation 4) forces the generator to keep the visual object appearance, i.e., pose, orientation, size, etc. Fig. 4 shows the masks adaptability from HR to SLR objects.

Adding this process solves three issues: (i) the pipeline does not limit its performance to the existence of objects with a mask ground truth, which is missing in many popular datasets [4, 11, 12] as the annotation is very costly; (ii) the small object segmentation is optimized, as the performance of segmentation

[3]Mask R-CNN extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branches of classification and bounding box regression.

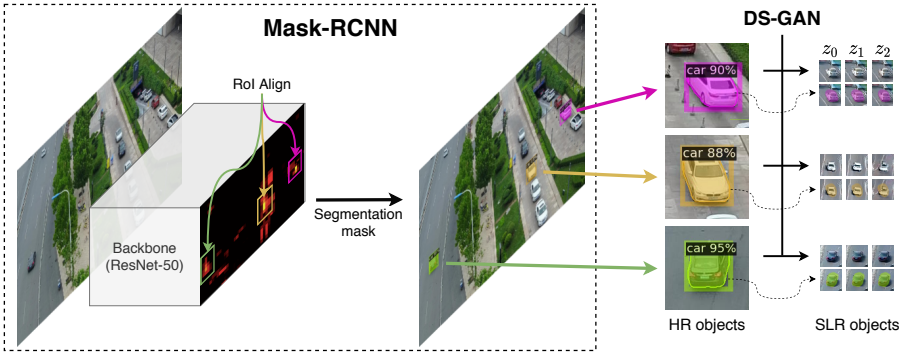

Figure 4: HR object segmentation using the Mask R-CNN framework [2] (right), and DS-GAN outputs for different noise vectors (z_i) with the masks fitted to the SLR objects (left).

methods declines dramatically for small objects; and (iii) there is no need to use the SLR objects to generate the segmentation mask —SLR objects do not contain enough context to get a proper mask (Fig. 4).

3.2. Small object integration

3.2.1. Position selector

The selection of a position within the image is a key issue when performing data augmentation for object detection. If this position is randomly selected, the new context surrounding the objects could be counterproductive, i.e., background mismatch may lead to more false positives. The reason is that the detector learns on not only the object features but also the context features, using the background prior knowledge to assist itself [16].

In order to sample a suitable position according to the image background, three premises must be fulfilled: (i) to have a plausible background —e.g., a car must be placed into the road—; (ii) the orientation has to fit the scene —e.g., a car's orientation has to match the direction of the road—; and (iii) the scale has to be according to the vanishing point of the frame —p.e. small objects cannot be placed in the foreground. As pointed out above, no temporal consistency for objects between frames is demanded; we only require objects to have a sensible spatial location within the frame. Using temporal consistency

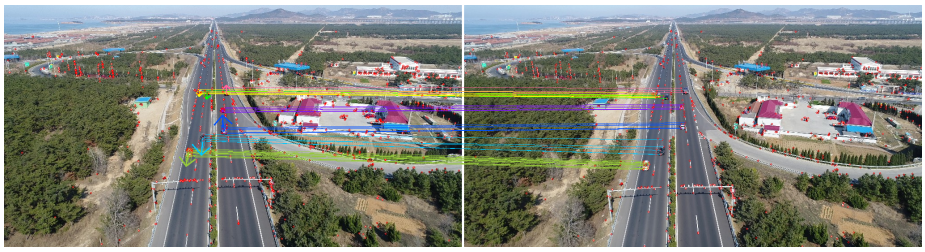

Figure 5: Angle of motion direction using optical flow for two frames f_t (right) and f_{t+1} (left). First, the feature points are computed using FAST (red dots). Second, f_{t+1} is stabilized with f_t by perspective transformation to remove camera motion. Then, the feature points are matched between frames (colored lines). Finally, the motion lines are summarized into a motion vector for each object (colored arrows).

would limit the number of object-background pairs, resulting in a less effective data augmentation system.

Therefore, to cover these requirements, our proposed position procedure is also based on three techniques: spatial memory of the objects to obtain a plausible background, optical flow to match orientations, and overlap to match scales. The spatial memory of the objects aims to collect plausible positions where to place an SLR object in the current frame. All locations of LR objects in the current frame are valid candidate positions. Also, LR object positions in previous and subsequent frames are candidates to place SLR objects as long as there is no overlap with objects in the current frame —this does not apply to image datasets. Optical flow and overlap aim to pair each candidate position with the SLR object that most closely resembles the orientation and size —for image datasets only overlap is taken into account. We exploit optical flow to compute the apparent motion of objects within two frames (Fig. 5): (i) we detect FAST keypoints [40]; (ii) stabilize camera motion by perspective transformation; (iii) link feature points between f_{t-1} and f_t within each bounding box by optical flow; (iv) compute the motion angle for each object in f_t by averaging all its points into a motion vector. The overlap between two objects is computed via IoU.

Given the angle of motion direction and the sizes associated to the HR and LR objects, each possible position gets this information from the LR object

Algorithm 1: Position selector

Input : $GT = \{GT^t = \{g_1^t, \ldots, g_{n_t}^t\}\ \forall\ t = 1, \ldots, T\}$
Input : $LR = \{LR^t = \{s_1^t, \ldots, s_{m_t}^t\}\ \forall\ t = 1, \ldots, T\} \mid LR \subset GT$
Input : $HR = \{b_1, \ldots, b_l\} \mid HR \subset GT$
Input : $SLR = \{\hat{s}_1, \ldots, \hat{s}_l\} \mid \hat{s}_i = G(b_i, z)\ \forall\ i = 1, \ldots, l$
Input : Search range τ
Output: $\mathbb{A} = \{\mathbb{A}^t = \{(e_i^t, \hat{s}_{k(i)}) \ldots (e_n^t, \hat{s}_{k(n)})\}\ \forall\ t = 1, \ldots, T,\ e_i^t \in E_t\}$

$\mathbb{A} \leftarrow \emptyset$
for $t = 1, \ldots, T$ **do**
  $E_t \leftarrow \emptyset$
  for $t' = \texttt{max}(0, t - \tau), \ldots, \texttt{min}(T, t + \tau)$ **do**
    if $t = t'$ **then**
      $E_t \leftarrow E_t \cup s_i^t$
    else
      for $i = 1, \ldots, m_{t'}$ **do**
        $valid\_spot = 1$
        for $j = 1, \ldots, n_t$ **do**
          if $\texttt{IoU}(s_i^{t'}, g_j^t) > 0$ **then**
            $valid\_spot = 0$
        for $j = 1, \ldots, \texttt{size}(E_t)$ **do**
          if $\texttt{IoU}(s_i^{t'}, e_j^t) > 0$ **then**
            $valid\_spot = 0$
        if $valid\_spot = 1$ **then**
          $E_t \leftarrow E_t \cup s_i^{t'}$
  for $i = 1, \ldots, \texttt{size}(E_t)$ **do**
    $max_v = max_j = 0$
    $\alpha_i^t = \texttt{OpticalFlow}(e_i^t)$
    for $j = 1, \ldots, l$ **do**
      $\alpha_j = \texttt{OpticalFlow}(b_j)$
      $\Delta_{i,j} = 1 - \texttt{normalize}(|\alpha_i^t - \alpha_j|)$
      $iou_{i,j} = \texttt{IoU}(e_i^t, \hat{s}_j)$
      if $\Delta_{i,j} + iou_{i,j} > max_v$ **then**
        $max_v = \Delta_{i,j} + iou_{i,j}$
        $max_j = j$
    $\mathbb{A}^t \leftarrow \mathbb{A}^t \cup (e_i^t, \hat{s}_{max_j})$
  $\mathbb{A} \leftarrow \mathbb{A} \cup \mathbb{A}^t$

from which it has given rise and each SLR object from its original HR object. Then, each position and SLR object pairing will be given by maximizing the overlap and angle of motion direction similarity between them. Alg. 1 shows the position selector method for each video:

- **Input**: The algorithm takes as input the total set of objects in the dataset (GT) within each frame f at time t (f_t) —which includes the LR and HR subsets—, the total set of SLR objects obtained by the DS-GAN generator G from HR objects and the search range τ.

- **Ouput**: The algorithm returns the association ($\mathbb{A}$) of an SLR object ($\hat{s}_i$) for each empty space (e_j) —$\hat{s}_i$ can be linked to more than one e_j.

- **Spatial memory** (lines 4-17): Given frame f at time t, the possible empty spots (E_t) to place an SLR object ($\hat{s}_i$) will be those where an LR object (s_j) existed in the frames from $f_{t-\tau}$ to $f_{t+\tau}$ (line 4) —s_i^t is always valid (line 6). For each frame $f_{t'}$ of the interval ($f_{t-\tau}$, $f_{t+\tau}$) the algorithm checks if the $LR^{t'}$ objects overlap with any of the objects of the current frame (GT^t) or with any space already selected (E^t) (lines 9-15). Otherwise, $s_i^{t'}$ is added as new empty spot to E^t (line 17). Thus, each possible empty spot e_j^t corresponds to a position of an LR object ($s_i^{t'}$).

  The value of τ will be influenced by the video dataset and, more specifically, by the camera motion. The more the camera moves, the less the value of τ will be to avoid background mismatch. If the camera motion is too quick, the positions of the objects in previous or subsequent frames may correspond to an erroneous position in the image –e.g., a car on a sidewalk.

- **Object association** (lines 18-28): The best $\hat{s}_i$ is calculated for each of the empty spots e_j^t by maximizing the motion direction and overlap.

  - **Optical flow**: For each ground truth LR and HR objects in the video dataset, an angle (α) associated with its motion vector is precalculated through optical flow (Fig. 5) —lines (20 and 22). As in the segmentation step, the $\hat{s}_i$ motion vector can be derived from its original HR object b_i (OpticalFlow($\hat{s}_i$) = OpticalFlow(b_i)). Considering the SLR and the LR subsets, motion similarity (Δ) associated with each pair $\hat{s}_i$, s_j is given by:

$$\Delta_{i,j} = 1 - \text{norm}(|\text{OpticalFlow}(b_i) - \text{OpticalFlow}(s_j)|) \quad (5)$$

– **Overlap**: Likewise, the $\hat{s}_i$ size can be derived from its original HR object b_i ($w(\hat{s}_i) = \frac{w(b_i)}{r}$; $h(\hat{s}_i) = \frac{h(b_i)}{r}$). Then, the overlap between $\hat{s}_i$ and s_j is computed using IoU.

Finally, the i-th SLR object selected to fill the position e_j^t will be given by:

$$i = \max(\Delta_{i,j} + \text{IoU}(\hat{s}_{k(j)}, e_j^t)) \ \forall \ t = 1, \ldots, T, \ e_j^t \in E_t, \quad (6)$$

3.2.2. Inpainting

The position selector procedure considers each s_j^t in f_t as an empty spot e_j^t for filling with $\hat{s}_i$. In these situations, it is mandatory to remove s_j^t associated to e_j^t via inpainting before inserting its pair. This ensures that the newly generated object is then blended with a uniform background which is the result of the inpainting. To this end we perform image inpainting using DeepFill [36]. DeepFill is a generative model-based approach which can synthesize novel image structures using surrounding image features.

Deepfill takes as input the frame f_t and a mask m_t and returns the same image f_t' but with the empty regions filled. For generating the mask m_t associated with the frame f_t, the bounding boxes of the selected LR objects $s_i^t \in E^t$ will be considered, so that those pixels contained in them will be flagged ($m_t = 1$).

The generator comprises two encoder-decoder networks for two different purposes. The first one —coarse network— aims to make an initial coarse prediction and, the second network —refinement network— takes the coarse prediction as inputs and predicts the final result f_t'. The reason for these two networks is intuitive: the refinement network sees a more complete scene than the original image with missing regions, so its encoder can better learn feature representation than the coarse network.

As LR^t may be surrounded by other objects, it is interesting to borrow

distant image features within the image without objects. This is addressed by DeepFill with two parallel refinement network encoders concatenated at the end into a single decoder. The standard encoder specifically focuses on refining local contents with layer-by-layer (dilated) convolution, while the attention encoder tries to capture background interest features.

3.2.3. Insertion and Blending

As a final stage, the pipeline blends the corresponding SLR object $\hat{s}_i$ obtained by Equation 6 over an f_t' inpainted image obtained in the previous step in each of the spots e_j^t to generate f_t^*. First, the segmented object $\hat{s}_i$ is placed in the selected position e_j^t. Then, the blending step is required to improve color consistencies and to soft the object edges in order to make the composite image look as natural as possible. We have adopted the Laplacian pyramid introduced by Burt and Adelson [37] to blend the SLR objects into the video frames.

This blending method takes as input an inpainted video frame f_t', the copy-pasted image f_t'' and the mask image m_t' that points out where to blend. In the inpainting stage, the flagged pixels in m_t are those inside the bounding box ground truth, but in m_t' the flagged pixels are those from the SLR segmented pixels. Alg. 2 details the procedure to obtain the final synthetic video frame:

1. Create the temporal image f_t'' by copy-pasting each $\hat{s}_{k(i)}^t$ object in e_i^t on f_t' (line 3). Generate the mask m_t' by flagging those pixels that belong to $\hat{s}_{k(i)}^t$ (line 4).

2. Compute p levels of Gaussian pyramids for f_t', f_t'' and m_t' (lines 5-9). Each Gaussian pyramid level is the result of blurring and downsampling the previous one.

3. From the Gaussian pyramids, calculate the Laplacian pyramid for f_t' and f_t'' (lines 10-13). Each Laplacian pyramid level is the result of subtracting each Gaussian pyramid level with the up-sampled and blurred previous one. The smaller level in the Laplacian pyramid is the same as the smaller in Gaussian pyramid.

Algorithm 2: Insertion and blending algorithm

Input : $\mathbb{A}^t = \{(e_i^t, \hat{s}_{k(i)}) \dots (e_n^t, \hat{s}_{k(n)})\}\ \forall\ t = 1, \dots, T,\ e_i^t \in E_t$
Input : Inpainted image: f_t'
Input : Pyramid levels: p
Output: Final synthetic image: f_t^*

1 $f_t'' \leftarrow f_t'$; $m_t' \leftarrow \emptyset$
2 **for** $i = 1, \dots, n$ **do**
3 $\quad f_t''[e_i^t] = \hat{s}_{k(i)}^t$
4 $\quad m_t'[\hat{s}_{k(i)}] = 1$
5 $F_t' \leftarrow \{f_t'\}$; $F_t'' \leftarrow \{f_t''\}$; $M_t'' \leftarrow \{m_t'\}$
6 **for** $i = 1, \dots, p$ **do**
7 $\quad F_t' \leftarrow F_t' \cup \texttt{PyramidDown}(F_t'[i])$
8 $\quad F_t'' \leftarrow F_t'' \cup \texttt{PyramidDown}(F_t'[i])$
9 $\quad M_t' \leftarrow M_t' \cup \texttt{PyramidDown}(F_t'[i])$
10 $L_t' \leftarrow \{F_t'[p]\}$; $L_t'' \leftarrow \{F_t''[p]\}$
11 **for** $i = p, \dots, 2$ **do**
12 $\quad L_t' \leftarrow L_t' \cup (F_t'[i-1] - \texttt{PyramidUp}(F_t'[i]))$
13 $\quad L_t'' \leftarrow L_t'' \cup (F_t''[i-1] - \texttt{PyramidUp}(F_t''[i]))$
14 $B_t \leftarrow \emptyset$; $M_t'' \leftarrow \texttt{Reverse}(M_t'')$
15 **for** $i = 1, \dots, p$ **do**
16 $\quad b = L_t'[i] \times M_t'[i] + L_t'' \times (1 - M_t'[i])$
17 $\quad B_t \leftarrow B_t \cup b$
18 $b \leftarrow B_t[1]$
19 **for** $i = 2, \dots, p$ **do**
20 $\quad b = \texttt{PyramidUp}(b) + B_t[i]$
21 $f_t^* \leftarrow b$

4. Next, each level of the Laplacian pyramid is blended according to m_t' of the corresponding Gaussian level (line 16). The set of masks (M_t') is previously reversed to match the dimensions (line 14).

5. Finally, from this blended pyramid, the output image (f_t^*) is reconstructed by up-sampling and blurring each level and adding it to the next one (line 18-21).

4. Experiments

In this section we address the datasets, evaluation metrics and implementation details to validate our approach.

4.1. DS-GAN

For this experimentation, the SLR objects generated by the DS-GAN are compared with the LR objects —aiming for the greatest similarity— as well as with the resizing functions: linear interpolation, bicubic interpolation, nearest neighbours and Lanczos [41]. For this purpose, two metrics will be used to validate the quality of the synthetic objects generated by DS-GAN: the Frechet Inception Distance (FID) [42] and object classification.

FID is a popular metric for comparing the feature vectors calculated for real and generated images. The FID score summarizes how similar the two groups are in terms of statistics on computer vision features of the raw images calculated using a pre-trained image classification model. The lower the scores the greater the similarity of the two groups, meaning that they have more similar statistics, which is the purpose of our DS-GAN.

To support the above metrics, we also train an LR object classifier which differentiates between background (negative) and LR object (positive). We resort to this metric since it is closer to the objective of the full pipeline, i.e., the improvement of small object detection. On the one hand, the classifier is trained with the LR training set as positive examples and a background set as negative examples. On the other hand, the SLR set is used for positive examples and keeping the same backgrounds as negative examples. We have generated different SLR sets, one for each of the resizing functions, and another one for the DS-GAN. All the learned models are evaluated with the LR testing subset and different backgrounds. The higher the accuracy, the better the quality of the objects synthetically generated.

The DS-GAN generator architecture has a final stride 4× smaller than the fixed size input image ($r = 4$). Most of the popular datasets —MS COCO [7], UAVDT [11], VisDrone [12]— consider as small objects those smaller than 32×32 pixels. Therefore, we will train the DS-GAN to learn how to reduce HR objects to that range.

We validate our data augmentation for small object detection approach with the car category on the UAVDT dataset [11]. This dataset was selected because

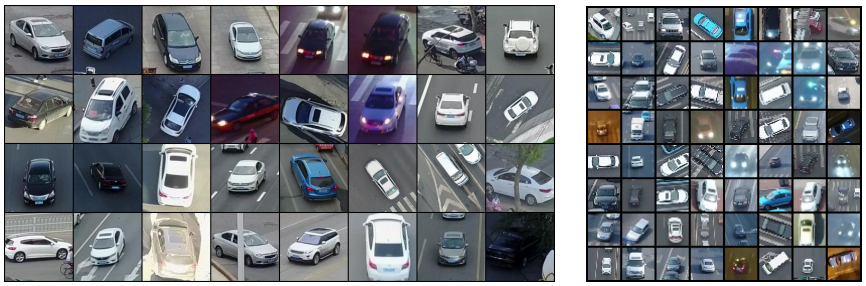

Figure 6: Real HR samples (left), and real LR samples (right).

the whole set of objects are vehicles, which allows us to isolate the results for a specific category, and also provides a large number of small instances in the testing set. Quantitatively, UAVDT comprises 23,829 frames of training data and 16,580 frames of test data, belonging to 30 and 20 videos of $\approx$ 1,024 $\times$ 540 resolution, respectively. The videos are recorded with an UAV platform over different urban areas. UAVDT includes a total of 394,633 car instances for training, where 107,091 are considered within the *small* subset (52.38%), and a total of 361,055 car instances for testing, where 274,438 are considered within the *small* subset (76.01%).

Considering that the camera motion in UAVDT slightly modifies the appearance of consecutive frames, in this section, only 10% of the video frames are selected for training to avoid overfitting. The details on the datasets for evaluating DS-GAN are given below:

- **Real HR subset**: To obtain the HR objects we select those objects from 48×48 to 128×128 pixels, and we add context to have an area of 128×128 pixels in objects with a smaller area. These conditions result in a total number of 517 HR objects in the UAVDT dataset. To have a larger number, we also select the cars in the VisDrone dataset with the same restrictions. VisDrone is a dataset with a very similar nature to that of UAVDT, i.e., high-resolution videos recorded with UAVs. The total number of HR objects is 5,731 after joining both datasets. Some HR examples are shown in Fig. 6(left).

- **Real LR training subset**: To obtain the LR objects we select those objects under 32×32 with sufficient context to cover an area of 32×32 pixels. This results in a total of 18,901 objects coming from the UAVDT training set —these objects are a part of the UAVDT *small* subset, where redundant instances have been discarded. However, in order to simulate a small object scarcity scenario, the LR subset will only consist of approximately 25% of the videos of the UAVDT dataset. The selected videos include a total of 5,226 LR objects. Some LR examples are shown in Fig. 6(right).

- **Real LR testing subset**: To evaluate the performance DS-GAN and the pipeline we use the 274,438 small objects coming from the UAVDT testing set with sufficient context to cover an area of 32×32 pixels.

For training the DS-GAN, we augment the training data by applying random image flipping to increase diversity. We provide a different noise vector (z) sampled from a normal distribution to each HR object in order to simulate a large variety of image degradation types. DS-GAN is trained during 1,000 epochs with an update ratio 1:1 between the discriminator and the generator, and it is optimized with Adam [43] with parameters $\beta_1 = 0$ and $\beta_2 = 0.9$. We set the base learning rate to 1e-4, decreasing it twice during the training phase by a factor of 10. We use $\lambda = 0.01$ in Eq. 2 to balance the relevance of the two components in the image generation process —l^G_{adv} is two orders of magnitude higher than l_{pixel}. Thus, the adversarial loss helps to learn to contaminate the HR input with noise and artefacts coming from the LR subset, and the pixel loss helps to preserve the visual features from the original input.

Fig. 7a and Fig. 7b show the experimental results to evaluate the quality of the synthetic objects generated by DS-GAN over the LR testing subset of UAVDT. Our approach is compared to the main re-scaling functions: linear and bicubic interpolation, nearest neighbors and Lanczos [41]. The reference values are obtained by the models trained on the LR training subset (blue bars).

The FID value in Fig. 7a is measured using the final average pooling features in Inception-v3 [44]. The reference value of the LR training objects compared

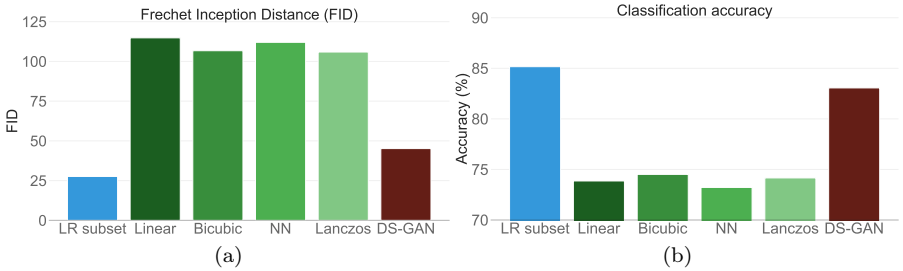

Figure 7: FID (a) and classification accuracy (b) for different subsampling methods on the LR testing subset of UAVDT.

with the LR testing subset is 27.62. The graph of Fig. 7a shows how the small objects obtained by any re-scaling function lead to values above 100, which is a poor performance relative to the reference value. The FID value of the SLR objects generated by DS-GAN for the LR test objects is 45.15. This FID value shows how the objects generated by the DS-GAN have better quality than those obtained by a simple re-scaling function, i.e., are more similar to the real ones.

To complement the FID distance, we have trained a classification network (ResNet-50 pre-trained on ImageNet [8]) with each of the defined subsets and tested them with the LR testing subset. Fig. 7b shows, again, how the SLR object generated by DS-GAN provides a considerably higher accuracy (83.06%) than the re-scaling functions (≈74%), and are very close to the reference accuracy obtained by the LR training subset (85.16%).

These results validate the conclusions reached in [17, 18], since re-scaling functions introduce artefacts that make the output object differ considerably from real-world objects. Even though these differences are not visually appreciable —as we will see in Fig. 9 (left) below—, they are identified by the layers within the CNNs (Inception-v3 and ResNet-50). DS-GAN significantly improves this issue by learning the different artefacts found in real-world objects.

4.2. Data augmentation pipeline

In order to evaluate our pipeline for data augmentation for small object detection, shown in Fig. 2, we use the UAVDT detection metrics that were

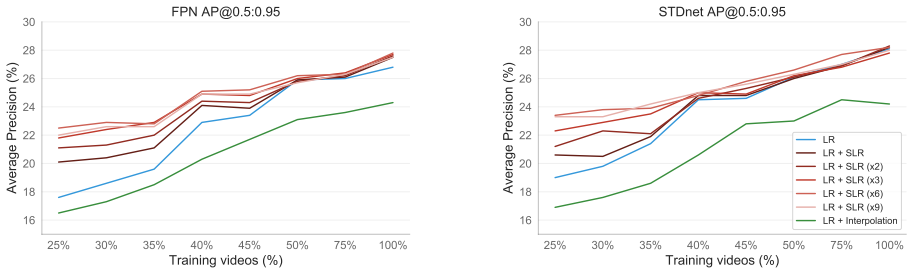

Figure 8: $AP_s^{@[.5,.95]}$ for small object detection in UAVDT for different percentage of training videos with the FPN and STDnet architectures.

originally defined by the MS COCO dataset, i.e., $AP^{@.5}$ and $AP^{@[.5,.95]}$. STDnet [4], FPN [3] and CenterNet [45] are adopted as the baseline detection networks.

The implementation details for DS-GAN are those defined in the previous section. The other component that requires training is DeepFill for image inpainting. In this case, the default parameters [36] are used to train the model on the UAVDT dataset. We have set $\tau = 40$ as the frame search range for the position selector. The rest of the components of the pipeline shown in Fig. 2 are also configured with their default values.

We detail the results obtained by STDnet [4], FPN [3] and CenterNet [45] on the UAVDT testing set for small objects. The training phase for all the models was conducted from the same 25% of the videos as in the DS-GAN training, in order to simulate a scenario with a low number of LR objects, up to the whole UAVDT training set. Here, the *LR* label means that no data augmentation has been applied for training, so the images come directly from the standard UAVDT training set. The *LR + Interp.* and *LR + SLR* labels mean the same images with real objects as in *LR*, and also duplicating those images replacing the real LR objects with synthetic objects ones generated with the pipeline using bilinear interpolation and DS-GAN, respectively. So that, in *LR + Interp.* and *LR + SLR*, the number of synthetic objects is equal to the number of LR objects. Notice that *LR + Interp.* is a more elaborated solution than [5], as it is the proposed pipeline, but replacing DS-GAN by bilinear interpolation. Finally, the *LR + SLR*×*n* labels mean that the number of SLR objects is *n* times higher

Data augmentation	FPN		STDnet		CenterNet	
	$AP_s^{@.5}$	$AP_s^{@[.5,.95]}$	$AP_s^{@.5}$	$AP_s^{@[.5,.95]}$	$AP_s^{@.5}$	$AP_s^{@[.5,.95]}$
LR	39.0	17.6	41.2	19.0	51.9	22.6
LR + Interp.	38.1	16.5	38.8	16.9	46.9	18.4
LR + SLR	46.3	20.1	48.1	20.6	60.6	26.1
LR + SLR×6	50.9	22.5	51.5	23.4	63.5	26.8

Table 1: Comparison of several data augmentation approaches for small object detection with FPN, STDnet and CenterNet networks on the small object testing subset of UAVDT. The training phase was conducted by simulating a low instance small object scenario —25% of the UAVDT training videos.

than the number of LR objects.

We do not provide an ablation study on the influence of the different components of the pipeline but DS-GAN, as the process would be incomplete —without segmentation—, it would create incoherent scenes —without position selecting—, or the generated synthetic small objects would have artifacts in the background surroundings —without inpainting or blending.

Table 1 studies the influence of different data augmentation methods for a scenario where the number of small objects for the training phase is reduced. So that, the first row refers only to the use of real objects contained in the 25% of the videos. The use of data augmentation with DS-GAN improves the performance of FPN by 4.9% $AP_s^{@[.5,.95]}$ and 11.9% $AP_s^{@.5}$, STDnet by 4.4% $AP_s^{@[.5,.95]}$ and 10.3% $AP_s^{@.5}$, and CenterNet by 4.2% $AP_s^{@[.5,.95]}$ and 11.6% $AP_s^{@.5}$ —Table 1, rows 1 and 4. It should be noted that the greatest influence is given by the nature of synthetic objects. If they did not contain useful information for learning the model, they would not improve the performance, or even worsen it, as seen with the bilinear interpolation method in Table 1. The improvement from data augmentation with objects re-scaled by bilinear interpolation to synthetic objects generated by DS-GAN is of 3.6% $AP_s^{@[.5,.95]}$ and 8.2% $AP_s^{@.5}$ in FPN, 3.7% $AP_s^{@[.5,.95]}$ and 9.3% $AP_s^{@.5}$ in STDnet, and 7,7% $AP_s^{@[.5,.95]}$ and 13.7% $AP_s^{@.5}$ in CenterNet —Table 1, rows 2 and 3.

Fig. 8 details the extended results for FPN and STDnet for the use of a different percentage of videos in the training phase and, also, shows how AP changes by increasing the number of SLR objects $\times n$ in the training phase.

These graphs are designed to show the improvement due to data augmentation for different percentages of training videos —with real LR objects. It is possible to appreciate a great improvement in AP for those solutions based on our data augmentation approach —the greater the number of SLR, the greater the improvement— especially when the percentage of training videos is low. As the percentage of training videos increases, the improvement is reduced, as there are more real objects in the training set. From the use of 50% of the videos onwards the AP shows a smaller improvement rate, so does the gain by adding SLR objects. That is, when adding more training images with real objects performance does not improve, and thus it is useless to try to use data augmentation techniques.

As expected, as training examples increase, so does AP. However, as mentioned above, the improvement from 50% of videos is considerably lower, moving from 25.9% $AP_s^{@[.5,.95]}$ for 50% of the videos to 26.8% $AP_s^{@[.5,.95]}$ for FPN and the whole UAVDT training set (blue line, left). Similarly, the performance of the trained model with data augmentation increases as objects are added, but the gain over the baseline is lower above 50% of training videos. The same conclusions can be drawn in the case of STDnet (blue line, right).

Moreover, the models are able to take advantage of the increasing number of SLR objects until reaching a point where the progression stops —9× with respect to LR objects. To synthesize new objects above SLR×3 requires to triplicate the images and exchange the synthetic objects, because there are not enough empty spots available where to insert SLR objects. This decreases the context variability, and thus the performance improvement.

Finally, we want to highlight how the generated synthetic objects constantly improve the performance even for the complete training set (100%), where they improve $AP_s^{@[.5,.95]}$. In contrast, the objects generated by bilinear interpolation do not provide information, and even they harm the learning of the models (green lines). This confirms the high quality of the synthetic dataset produced by our pipeline for data augmentation for small object detection.

As qualitative results, Fig. 9 compares the synthetic objects coming from

Figure 9: Synthetic objects obtained by bilinear interpolation (left); synthetic objects generated by the DS-GAN (middle); and real LR objects (right).

Figure 10: Data augmentation for small objects examples from UAVDT training set provided by our pipeline. From left to right and from top to bottom: standard real frame with LR objects; LR objects replaced by SLR objects; SLR×2 objects; and SLR×3 objects.

a simple re-scaling function with those generated by the DS-GAN and with the real LR objects. Objects obtained by simple re-scaling seem artificially defined with blurry artefacts. Objects from DS-GAN look more closely to real LR objects as they contain artefacts and are contaminated by low-resolution small objects features. Fig. 10 displays the outcome of the complete pipeline for different UAVDT scenarios.

	$AP_s^{@[.5,.95]}$	$AP_m^{@[.5,.95]}$	$AP_l^{@[.5,.95]}$	$AP_s^{@.5}$	ST	LV	P	S	SP
LR	24.8	45.0	31.3	43.2	38.2	14.6	62.3	50.0	50.7
LR + SLR	27.4	47.1	31.1	47.9	41.8	16.6	70.4	51.5	58.9

Table 2: Results of FPN on iSAID. The training phase was conducted by simulating a low instance small object scenario —real LR training subset. The results for the different categories —storage tank (ST), large vehicle (LV), plane (P), ship (S), and swimming pool (SP)— are for $AP_s^{@.5}$.

We have also tested the data augmentation pipeline on the iSAID dataset[19]. iSAID has 1,869 high-resolution aerial images with objects from 15 categories, 1,411 for training and 458 for testing. As the pipeline requires for training and testing several subsets —a real HR subset, and a real LR training and testing subsets—, we selected the categories with a sufficient number of objects in each subset. The number of objects selected for training is about 15% of the total number of annotated objects —6,628 objects for the LR subset and 1,405 for the HR subset.

Table 2 shows the results for the FPN detector for our augmentation method compared with the baseline on iSAID. The use of data augmentation with DS-GAN improves the performance of FPN by 2.6% $AP_s^{@[.5,.95]}$ and 4.7% $AP_s^{@.5}$. Depending on the category, the $AP_s^{@.5}$ increases between 1.5 (ship) to 8.2 (swimming pool). Moreover, DS-GAN does not harm the performance of large objects detection ($AP_l^{@[.5,.95]}$), and even it improves the medium objects detection ($AP_m^{@[.5,.95]}$) —as the larger small objects are close in size to the smaller medium objects.

Even though the performance of DS-GAN is good on iSAID, it is not as impressive as on UAVDT. We argue that there are two reasons for that. First, iSAID is an image dataset, so the position selector of DS-GAN has less places to insert synthetic objects in comparison with a video dataset like UAVDT. The second reason is that iSAID contains aerial images that are taken much further away and for objects much larger —the objects undergo a higher scale change— than those of the videos of UAVDT, so the texture of the small objects of UAVDT is much better than that of iSAID.

The runtimes for the generation of a dataset with synthetic objects ready to be used for training an object detector are quite fast. The small object generation, which includes the execution of the GAN and the object segmentation, creates 12.6 objects per second. The small object integration into the image, which involves position selection, object inpainting and object blending, inserts 10.1 objects per second.

5. Conclusions

We have designed a novel a pipeline for data augmentation for small object detection. The pipeline takes a dataset as input and returns the same dataset with the images populated with annotated small synthetic objects. The proposed pipeline requires both HR and LR objects to train the DS-GAN and, also, a trained object segmentation system for HR objects. The approaches based on super-resolution through GANs also need both HR and LR objects for training [21, 22, 23, 24]. However, our proposal has an advantage over super-resolution-based approaches, as the GAN only has to be executed during the training stage in order to generate the synthetic LR objects, so at inference time —small object detection— only the object detector has to be run. On the other hand, super-resolution-based pipelines require both the execution of the generator of the GAN and the object detector at the inference stage.

The quality of small objects generated by DS-GAN has been validated in an isolated way. Experiments show that the FID value for the SLR objects is very close to the FID value for real LR objects, as opposed to the simple downsampled objects, which have a very distant FID value. In addition, we reached the same conclusion by training a standard CNN classifier. So that, we confirm that small objects generated by DS-GAN boost small object classification. On the contrary, the small objects generated by direct large objects re-scaling are useless for data augmentation to recognise small objects, as the artefacts introduced by these functions differ greatly from real-world small objects.

The proposed pipeline for data augmentation method improves the performance of state-of-the-art models in the detection of small objects on both the UAVDT and iSAID datasets. The results on the UAVDT test set show an improvement of 10.3-11.9 $AP_s^{@.5}$, depending on the detector, in a scenario where the number of training small objects is limited –only 25% of the videos are considered. For iSAID, the improvement is of 4.7 $AP_s^{@.5}$ for a scenario in which only the 15% of objects were included in the LR subset. These results validate the initial hypothesis that, when a dataset contains few small objects, the proposed

data augmentation technique boosts the performance of the detector.

Acknowledgment

This research was partially funded by the Spanish Ministerio de Ciencia e Innovación [grant numbers PID2020-112623GB-I00, RTI2018-097088-B-C32], and the Galician Consellería de Cultura, Educación e Universidade [grant numbers ED431C 2018/29, ED431C 2021/048, ED431G 2019/04]. These grants are co-funded by the European Regional Development Fund (ERDF). This paper was supported by European Union´s Horizon 2020 research and innovation programme under grant number 951911 - AI4Media.

References

[1] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 770–778.

[2] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, in: IEEE Int. Conf. Comput. Vis. (ICCV), 2017, pp. 2961–2969.

[3] T.-Y. Lin, P. Dollár, R. Girshick, et al., Feature pyramid networks for object detection, in: IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2017, pp. 2117–2125.

[4] B. Bosquet, M. Mucientes, V. M. Brea, STDnet: Exploiting high resolution feature maps for small object detection, Eng. App. Artif. Intell. 91 (2020) 103615.

[5] M. Kisantal, Z. Wojna, J. Murawski, et al., Augmentation for small object detection, arXiv preprint arXiv:1902.07296.

[6] J. Wang, X. Tao, M. Xu, et al., Hierarchical objectness network for region proposal generation and object detection, Pattern Recognition 83 (2018) 260–272.

[7] T.-Y. Lin, M. Maire, S. Belongie, et al., Microsoft coco: Common objects in context, in: Eur. Conf. Comput. Vis. (ECCV), 2014, pp. 740–755.

[8] O. Russakovsky, J. Deng, H. Su, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.

[9] W. Ma, Y. Wu, F. Cen, G. Wang, MDFN: Multi-scale deep feature learning network for object detection, Pattern Recognition 100 (2020) 107–149.

[10] G. Tian, J. Liu, H. Zhao, W. Yang, Small object detection via dual inspection mechanism for UAV visual images, Applied Intelligence 52 (2022) 4244–4257.

[11] H. Yu, G. Li, W. Zhang, et al., The unmanned aerial vehicle benchmark: Object detection, tracking and baseline, International Journal of Computer Vision 128 (5) (2020) 1141–1159.

[12] P. Zhu, et al., VisDrone-VID2019: The vision meets drone object detection in video challenge results, in: IEEE Int. Conf. Comput. Vis. Workshops, 2019.

[13] J. Wang, W. Yang, H. Guo, et al., Tiny object detection in aerial images, in: Int. Conf. on Pattern Recognit. (ICPR), 2021.

[14] B. Zoph, E. D. Cubuk, G. Ghiasi, et al., Learning data augmentation strategies for object detection, in: Eur. Conf. Comput. Vis. (ECCV), 2020, pp. 556–583.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, et al., Generative adversarial nets, in: Adv. Neural Inf. Process. Syst. (NIPS), 2014, pp. 2672–2680.

[16] C. Chen, Y. Zhang, Q. Lv, et al., RRNet: A hybrid detector for object detection in drone-captured images, in: IEEE Int. Conf. Comput. Vis. Workshops, 2019.

[17] A. Bulat, J. Yang, G. Tzimiropoulos, To learn image super-resolution, use a gan to learn how to do image degradation first, in: Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 185–200.

[18] A. Shocher, N. Cohen, M. Irani, "zero-shot" super-resolution using deep internal learning, in: IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2018, pp. 3118–3126.

[19] S. Waqas Zamir, A. Arora, A. Gupta, et al., iSAID: A large-scale dataset for instance segmentation in aerial images, in: IEEE Conf. Comput. Vis. Pattern Recognit. Workshops, 2019, pp. 28–37.

[20] B. Bosquet, M. Mucientes, V. Brea, STDnet-ST: Spatio-temporal ConvNet for small object detection, Pattern Recognition 116 (2021) 107929.

[21] J. Li, X. Liang, Y. Wei, et al., Perceptual generative adversarial networks for small object detection, in: IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2017, pp. 1222–1230.

[22] Y. Bai, Y. Zhang, M. Ding, B. Ghanem, Finding tiny faces in the wild with generative adversarial network, in: IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2018, pp. 21–30.

[23] J. Wang, W. Yang, H. Guo, et al., SOD-MTGAN: Small object detection via multi-task generative adversarial network, in: Eur. Conf. Comput. Vis. (ECCV), 2018, p. 206–221.

[24] Y. Pang, J. Cao, J. Wang, J. Han, JCS-Net: Joint classification and super-resolution network for small-scale pedestrian detection in surveillance images, IEEE Trans. Inf. Forensics Security 14 (2019) 3322–3331.

[25] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Int. Conf. Learning Represen. (ICLR), 2015.

[26] D. Dwibedi, I. Misra, M. Hebert, Cut, paste and learn: Surprisingly easy synthesis for instance detection, in: IEEE Int. Conf. Comput. Vis. (ICCV), 2017, pp. 1301–1310.

[27] Z. Chen, W. Ouyang, T. Liu, D. Tao, A shape transformation-based dataset augmentation framework for pedestrian detection, Int. J. Comput. Vis. 129 (2021) 1121–1138.

[28] L. Wei, S. Zhang, W. Gao, Q. Tian, Person transfer GAN to bridge domain gap for person re-identification, in: IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2018, pp. 79–88.

[29] J. Zhu, T. Park, P. Isola, A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: IEEE Int. Conf. Comput. Vis. (ICCV), 2017, pp. 2223–2232.

[30] L. Liu, M. Muelly, J. Deng, et al., Generative modeling for small-data object detection, in: IEEE Int. Conf. Comput. Vis. (ICCV), 2019, pp. 6073–6081.

[31] K. Nasrollahi, T. B. Moeslund, Super-resolution: a comprehensive survey, Mach. Vis. App. 25 (6) (2014) 1423–1468.

[32] C. Dong, C. C. Loy, K. He, X. Tang, Image super-resolution using deep convolutional networks, IEEE Trans. Pattern Anal. Mach. Intell. 38 (2) (2015) 295–307.

[33] J. Johnson, A. Alahi, L. Fei-Fei, Perceptual losses for real-time style transfer and super-resolution, in: Eur. Conf. Comput. Vis. (ECCV), 2016, pp. 694–711.

[34] C. Ledig, L. Theis, F. Huszár, et al., Photo-realistic single image super-resolution using a generative adversarial network, in: IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2017, pp. 4681–4690.

[35] D. Pathak, P. Krahenbuhl, J. Donahue, et al., Context encoders: Feature learning by inpainting, in: IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 2536–2544.

[36] J. Yu, Z. Lin, J. Yang, et al., Generative image inpainting with contextual attention, in: IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2018, pp. 5505–5514.

[37] P. J. Burt, E. H. Adelson, A multiresolution spline with application to image mosaics, ACM Trans. Graphics 2 (4) (1983) 217–236.

[38] K. He, X. Zhang, S. Ren, J. Sun, Identity mappings in deep residual networks, in: Eur. Conf. Comput. Vis. (ECCV), 2016, pp. 630–645.

[39] T. Miyato, T. Kataoka, M. Koyama, Y. Yoshida, Spectral normalization for generative adversarial networks, in: Int. Conf. Learning Represen. (ICLR), 2018.

[40] E. Rosten, T. Drummond, Machine learning for high-speed corner detection, in: Eur. Conf. Comput. Vis. (ECCV), 2006, pp. 430–443.

[41] K. Turkowski, Filters for common resampling tasks, in: Graphics Gems, 1990, pp. 147–165.

[42] M. Heusel, H. Ramsauer, T. Unterthiner, et al., Gans trained by a two time-scale update rule converge to a local nash equilibrium, in: Adv. Neural Inf. Process. Syst. (NIPS), 2017, pp. 6626–6637.

[43] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[44] C. Szegedy, V. Vanhoucke, S. Ioffe, et al., Rethinking the inception architecture for computer vision, in: IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 2818–2826.

[45] K. Duan, S. Bai, L. Xie, et al., CenterNet: Keypoint triplets for object detection, in: IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2019, pp. 6569–6578.