DPNet: Dynamic Pooling Network for Accurate and Efficient Size-Aware Tiny Object Detection

Luqi Gong, *Student Member,IEEE*, Haotian Chen, Yikun Chen, *Member,IEEE*, Tianliang Yao, *Student Member,IEEE*, Chao Li, *Member,IEEE*, Shuai Zhao, *Member,IEEE*, Guangjie Han, *Fellow,IEEE*

***Abstract*—In unmanned aerial systems, especially in complex environments, accurately detecting tiny objects is crucial. Resizing images is a common strategy to improve detection accuracy, particularly for small objects. However, simply enlarging images significantly increases computational costs and the number of negative samples, severely degrading detection performance and limiting its applicability. This paper proposes a *Dynamic Pooling Network (DPNet)* for tiny object detection to mitigate these issues. *DPNet* employs a flexible down-sampling strategy by introducing a factor (*df*) to relax the fixed down-sampling process of the feature map to an adjustable one. Furthermore, we design a lightweight predictor to predict *df* for each input image, which will be used to decrease the resolution of feature map in backbone. Thus, we achieve input-aware down-sampling. We design an *Adaptive Normalization Module (ANM)* to make a unified detector well compatible with different *dfs*. At the same time, we also design a guidance loss to supervise the predictor's training. DPNet realizes the dynamic allocation of computing resources to trade off detection accuracy and efficiency through this. Experiments on the TinyCOCO and TinyPerson datasets show that our DPNet can save over 35% and 25% GFLOPs, respectively, while maintaining comparable detection performance.The code will be made publicly available.**

***Index Terms*—Dynamic neural network, small object detection,unmanned aerial systems**

I. Introduction

TINY object detection (TOD) [1], [2], [3], [4], [5], [6], [7], [8], [9], [10] has long been a challenging research direction in object detection. TOD aims to accurately detect objects with small sizes and a low signal-to-noise ratio. Due to a large number of tiny objects in the real scene, tiny object detection has a wide range of application prospects. It plays an indispensable role in various fields such as automatic driving [11], [12], smart medical treatment [13], defect detection [14], and aerial image analysis [15], [16], [17], [18], [19],particularly in drone system image recognition and analysis. Deep learning has injected fresh blood into tiny object detection research in recent years. With the rapid development of deep convolutional neural networks (CNNs), research in object detectors [20], [21], [22], [23] has made significant strides. For extremely small sizes compared with objects in MS-COCO [24], one common approach is to resize pre-training objects to tiny object scales, as demonstrated in methods like Scale Match (SM) [1] and its enhanced version SM+ [3]. However, these methods suffer information loss when downsizing images. Another common way is to enlarge the images' size to make tiny objects larger [6], [2]. Taking Fig. 1 as an example, when we downsizing the input image, the detector will be able to detect small objects better. Therefore, appropriately resizing up the input image or reducing the down-sampling operation can improve the detector's performance on tiny objects. However, resizing up the images also introduces some disadvantages: 1) increasing computational costs; 2) more negative samples; and 3) redundancy, as shown in Fig. 2 (for some objects, it is unnecessary to resize the image to a larger size).

In order to obtain a trade-off between speed and perfor-

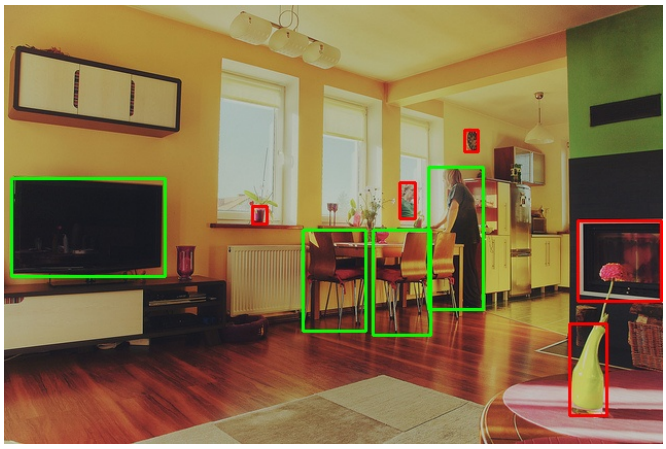

Fig. 1. Small objects can be better detected through resizing up the images. Objects in green boxes are correctly detected by both the detector trained on the original images, and the detector trained on enlarged images. Objects in red boxes are only correctly detected by the detector trained with enlarged images.

Manuscript received XXXXXX. (Corresponding author: Shuai Zhao; Chao Li).

Luqi Gong is with the School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China, and also with the Research Center for Space Computing System, Zhejiang Lab, Hangzhou, China(e-mail:luqi@bupt.edu.cn, luqi@zhejianglab.org).

Haotian Chen is with the SWJTU-LEEDS Joint School, Southwest Jiaotong University, Chengdu, China(email: cht@my.swjtu.edu.cn).

Yikun Chen is with the Guangdong Zhiyun City construction Technology Co., LTD, Zhuhai, China(email: kenter1643@163.com).

Tianliang Yao is with the Department of Control Science and Engineering, College of Electronic and Information Engineering, Tongji University, Shanghai, China(email: yaotianliang@tongji.edu.cn).

Chao Li is with the Research Center for Space Computing System, Zhejiang Lab, Hangzhou, China(email: lichao@zhejianglab.com).

Shuai Zhao is with the School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China(email: zhaoshuaiby@bupt.edu.cn).

Guangjie Han is with the Key Laboratory of Maritime Intelligent Network Information Technology, Ministry of Education, Hohai University, China (e-mail: hanguangjie@gmail.com).

Luqi Gong and Haotian Chen contribute equally to the article.

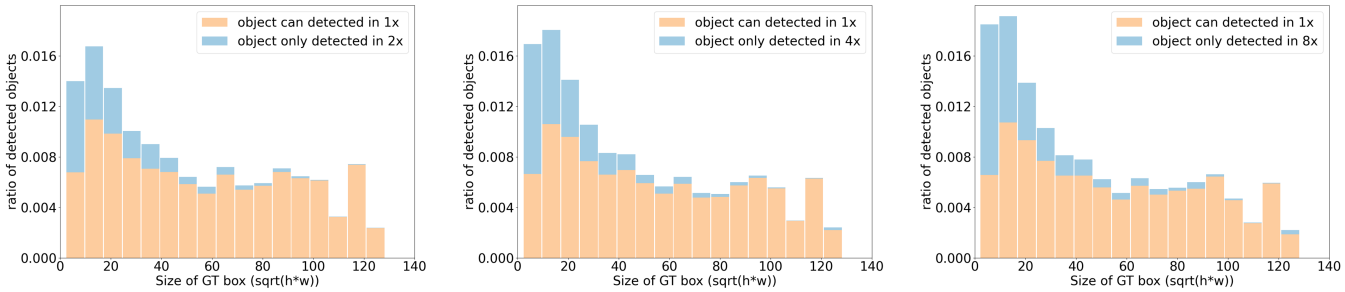

Fig. 2. The performance gain over different objects' scales by enlarging (resizing up). We train and inference on origin images (1x) and enlarged images (2x, 4x, 8x) respectively. Then we statistic the objects that can be detected by both 1x input and enlarged input (orange), and the objects that can only be detected by enlarging the input (blue). This figure indicates two points: 1) The performance gain from enlarged images mainly comes from tiny/small objects. 2) For medium and large objects, blindly enlarging input images does not improve performance, i.e. there is computational redundancy in enlarging. However, for tiny objects, the higher the enlarging level, the greater the performance gain. Therefore, we need a strategy that can dynamically adjust the resolution for different input.

mance, and eliminate the redundancy caused by enlarging all images, we design a dynamic neural network with variable dfs (denoted by df), called Dynamic Pooling Network (DPNet). Different from the traditional (static) neural network [25], [26], [27], [28] that uses the same parameters of down-sampling for all input images for inference during testing, our DPNet applies adaptive inference in the term of df. In order to get an appropriate df of the input, we introduce a down-sampling factor predictor (DFP), which is embedded into the backbone of the detector. In practical terms, several different dfs are set as candidates. The predictor is fed with an input feature map, produces a probability distribution over candidate dfs as the output, and chooses the best df. Then in the next stage of the backbone, the feature map will be scaled by the selected df. DFP is lightweight enough to be ignored to calculate the computation cost. For the detection task, the intrinsic difficulty level of the sample for recognition and the size and number of instances in the image may affect the amount of information required for correct detection. Therefore, when designing the DFP, we also introduce a branch to predict the statistical value of instances in the input image, which can help improve the rationality and effectiveness of DFP. By exploiting the proposed DPNet inference approach, we can excavate the spatial redundancy of each feature map from its size.

In summary, our core contributions are as follows:

- We propose Dynamic Pooling Network (DPNet), which to our best knowledge, is the first attempt to introduce the dynamic neural network to object detection tasks to achieve the trade-off of detection performance and computation.
- We design adaptive normalization module (ANM) to solve the aggravated scale variation issue brought by the mixed scale training (MST) scheme. A down-sampling factor predictor (DFP) is adopted to choose df adaptively during inference, and we design a guidance loss to supervise its training.
- Our DPNet can save over 35% computing cost while maintaining the comparable detection performance on the TinyCOCO dataset. Experiments on different backbone networks validate the effectiveness of our method.

II. Related Work

Tiny object detection is a widely concerned problem in object detection, which attracts increasing research. The object scale influences the detection performance. Thus many works focus on the scale problems in object detection. Dynamic neural networks can adaptively adjust models or input, which can help achieve the trade-off of computation and performance. In addition, to decrease computation, making the model lightweight is a traditional thought. Our method is a new paradigm that can work with the lightweight model method.

A. Tiny Object Detection

Extensive research has been carried out on tiny object detection from various aspects. Low resolution, weak signals, and high noise increase the difficulty of tiny object detection, limiting its research and application. To get reliable tiny object feature representation, Yu *et al.* [1] and Jiang *et al.* [3] use methods that align the scale distribution of the pre-training dataset and the target dataset. Gong *et al.* [5] designed effective fusion factors of adjacent layers in FPN [20]. Several methods recover information on low-resolution objects by way of Super-Resolution (SR). Employing large-scale SR features, EFPN [29] extends the original FPN with a high-resolution level specialized for small-sized object detection. To get super-resolution features, Noh *et al.* [30] uses high-resolution object features as supervision signals that match the relevant receptive fields of input and object features. [31] proposes a Gaussian Receptive Field based Label Assignment (RFLA) strategy for tiny object detection.

B. Scale Based Detection

The object scale has an important influence on the accuracy of the detection task. And it must also be considered in the design process of our method. Therefore, the object detection method that focuses on the scale problem needs to be discussed in the following. Large-scale variation across object instances is a challenging problem in object detection. Handling this problem can improve the ability of the detector to deal with the object detection task on various scales, especially for

objects of extreme size, *e.g.*tiny objects. A common strategy to improve the detection methods is the multi-scale image pyramid [32], [33]. Based on this scheme, SNIP [34] and SNIPER [35] apply a scale regularization method to multi-scale training, which guarantees the sizes of objects fall into a fixed scale range for various resolution images. Another normalization method, TridentNet [36], constructs parallel multi branches with different receptive fields to generate scale-specific feature maps. These feature maps have a uniform representational power. Instead of using multiple resolution input images and receptive field convolution kernels, SSD [37] and MS-CNN [38] utilize multiple spatial resolution feature maps to detect objects with different scales. They assign small targets to the bottom layers with high resolution, and large targets to the up layers with low resolution. To enhance the semantic representation of low-level features at the bottom layers, TDM [39] and FPN [20] further introduce a top-down pathway and lateral connections that merge deep layers and shallow layers. On the basis of FPN, PANet [33] adds a bottom-up path and proposes adaptive feature pooling to enhance the representational power of feature layers.

C. Dynamic Neural Networks

As an emerging topic in deep learning, dynamic neural networks have the advantages of high efficiency, great adaptability and strong representative ability, etc.To meet varying computing resource demands, [40], [41], [42], [43] perform inference with dynamic network depths, which allows "easy" samples to be output at shallow exit without exceuting deeper layers. [44], [45] implement halting scheme on Transformers [46] to achieve dynamic network depth on NLP tasks. Yu *et al.* [47], [48] present switchable batch normalization and in-place distillation to train a neural network executable at different widths. According to on-device benchmarks and computing resource constraints, the trained network, named slimmable neural network, can dynamically adjust its width. DS-Net [49] is also a network with variable width, which achieves dynamic routing for different samples by learning a slimmable supernet and a dynamic gating mechanism. RANet [50] implements resolution adaptive learning in deep CNNs for performing classification tasks efficiently. DRNet [51] dynamically adjust the resolution of input images for classification to achieve the trade-off of efficiency and classification accuracy. Our method first adopts these dynamic neural network ideas into detection tasks for wider application.

D. Light Weight Models

The efficient lightweight CNN models, such as MobileNet [52], [53] and ResNeXt [54] are widely utilized in the object classification task, which can be used as the backbone of object detectors. As we know, the backbone for feature extraction occupies most computation in detectors, so a lightweight backbone is the main research hotspot in the efficient network. MobileNet-v1 [52] adopts depthwise separable convolution to reduce parameter and computation without losing detection performance. MobileNet-v2 [53] introduces inverted residual block and linear bottleneck to improve the performance further. ResNeXt [54] introduces aggregated transformations, which replaces the residual structure of ResNet [55] by stacking blocks with the same topology in parallel. These models save computation by simplifying the network, however, our methods reduce that by dynamic controlling the size of the input feature map, which can work together with these lightweight models.

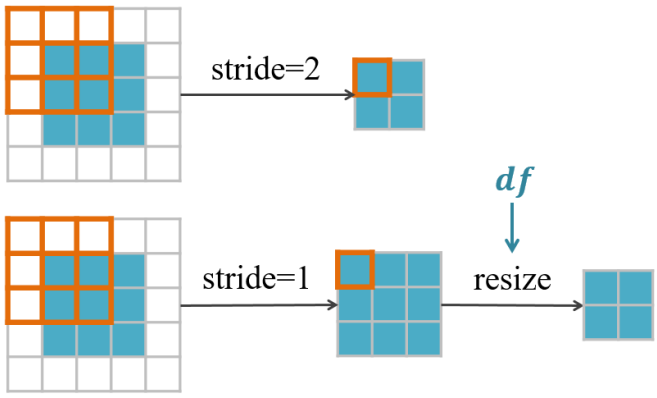

Fig. 3. The implement of the dynamic down-sampling. The down-sampling of feature maps is often implemented by setting the convolution kernel's strides, thus making the resolution of feature maps in the backbone fixed (top of the figure). We replace this process by resizing the features with the factor *df* through bilinear interpolation (bottom of the figure). while $df<0.5$, the required GFLOPs of subsequent network modules will reduce.

III. Dynamic Pooling Network

In this section, we first present the overall architecture of our approach. Next, we elaborate on the training procedure, the Adaptive Normalization Module (ANM) and the down-sampling factor predictor (DFP) individually.

Enlarging the image can significantly improve the overall performance of tiny object detection as showing in Table VIII (a) (in Section IV-E) and describing in [2], [6], but at the same time it will bring a large computational cost. Therefore in this paper, We choose the enlarged images as the detector's input to take advantage of the information gained from enlarging. And then an adaptive down-sampling process is carried out on the feature map of the detector's backbone, reducing redundancy computation while retaining necessary information. As shown in Fig. 3, the down-sampling factor of the traditional static network structure is fixed(df=0.5), which is implemented by a convolution or pooling layer with strides=2. To enable down-sampling at any *df*, We perform bilinear interpolation on the feature map with the given down-sampling factor *df*. Since the computational complexity of subsequent network modules is highly related to the resolution of the input feature map, using different *df* can control the computation cost in the network. In addition, the difficulty and the required minimum resolution of detecting objects in different images are different. For some images (e.g. size of objects inside is tiny), good results can only be obtained on high-resolution feature map, so we use a larger *df* (*e.g.*0.5). Conversely, for some images (e.g. size of objects inside is large), good performance can also be achieved on lower-resolution feature map, which means there is some computational redundancy, so we use a smaller *df* (e.g.

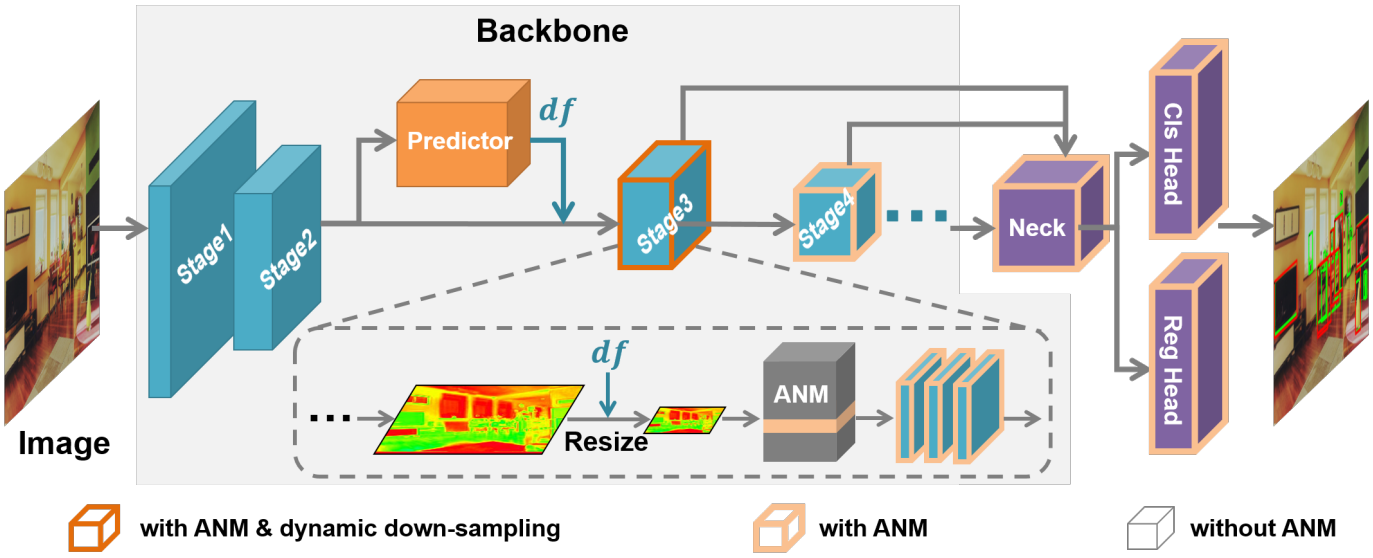

Fig. 4. Framework of DPNet. DFP is inserted into the backbone as a plug-in and guides a stage's the *df* selection. After the feature is rescaled by *df*, all normalization layers in the detector are replaced with ANM. The ANM will switch to the corresponding normalization layer for the feature map. The blue cubes are the stages of the backbone, the orange one is the proposed DFP, and the purple ones are the neck and head parts of the detector. (Best viewed in color)

0.25). Therefore, by dynamically selecting the down-sampling factor *df* based on the input image, we can achieve a trade-off between performance and computational cost.

The whole work can be divided into two tasks: 1) Train a detector that can well handle multiple *df*s, which means it can achieve high detection performance for all different *df*s. To this end, we adopt a mixed down-sampling factor training policy (detail in Section III-A) and design an Adaptive Normalization Module (ANM, detail in III-B) to make a unified detector compatible with different *df*s. 2) Choose the smallest *df* that still allows the detector to achieve strong detection performance on given input image under limited computational resources. For this purpose, We build a lightweight predictor (detain in III-C) embedded into detector to adaptively select the appropriate *df* according to the input image to decrease the feature map resolution.

In this section, we will introduce the training procedure. As shown in Fig. 4, our DPNet mainly consists of two components, a CNNs-based detector (such as RepPoints [56]) and a plugin-in DFP. To achieve a better performance, We optimized the basic detector network and *df* predictor one by one for better performance, as illustrated in Algorithm 1.

Mixed down-Sampling factor Training (MST).

Note that there can be an arbitrary value for *df* candidates from 0 to 1, making it difficult, also meaningless, As a simplification strategy under practical requirement, we choose m discrete *df*s $\{d_1, d_2, ..., d_m\}$ to shrink the exploration range, where d_m means the default down-sampling in a common detector. To achieve high detection performance over all different *df*s, one naive solution is training a separate network for each *df*, without any weight sharing between networks (the line of SF in Table I). However, maintaining multiple networks in this way is neither elegant nor practical for deployment. Therefore, a more common way is to train a single network under multiple *df*s (mixed downsampling factor training) to enable it to handle various *df*s (the line of MF in Table I).

A. DPNet training

Algorithm 1 Training of DPNet M

Require: Training set D_{train}, DPNet M

// Train the detector D
Initialize shared convolutions and fully-connected layers of D.
Initialize independent normalization layer of each d_j in df_list: $[d_1, d_2, ..., d_m]$
for $i = 1, \ldots, n_{iters\_pretrain}$ **do**
  Image x and label y of a mini-batch.
  for d_j in df_list **do**
    Switch to the normalization parameters of d_j.
    Resize the feature map of x with d_j.
    Compute $\mathcal{L}_{MST_j} = L_{cls_j} + \alpha L_{reg_j}$ with d_j, Eq. 1.
  end for
  Compute loss over all *df* $\mathcal{L}_{MST} = \sum_j \mathcal{L}_{MST_j}$, Eq. 2.
  Backward with $\mathcal{L}_{MST}$ and update parameters of detector D.
end for
// Train the DFP predictor P
Initialize P
Freeze the parameters of D
for $i = 1, \ldots, n_{iters\_pretrain}$ **do**
  Compute $\mathcal{L}_P$, Eq. 7, backward with it and update parameters of P.
end for
return M

As Fig. 5 showing, when training the DPNet detector, the same image is input to the network with m different *df*s

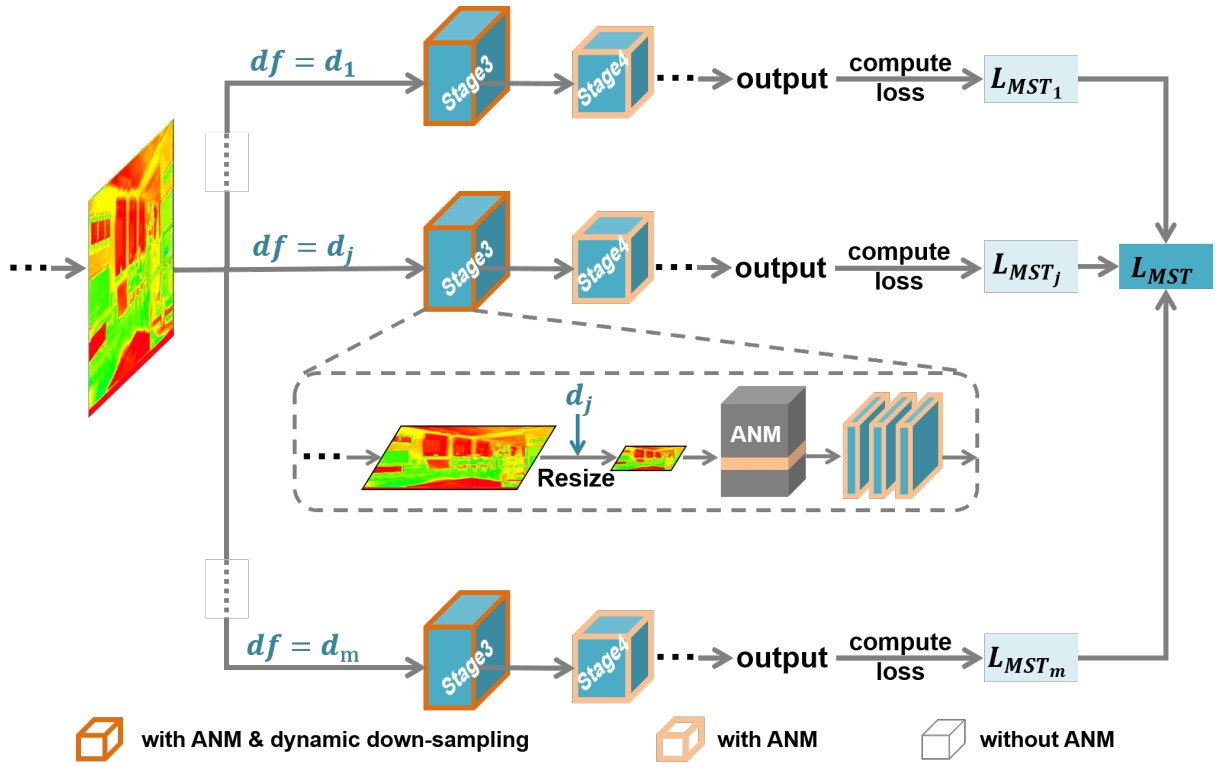

Fig. 5. The pipeline of Mixed Scale Training. To achieve high detection performance over all different *df*s, the same image is input to the network with m different *df*s $\{d_1, d_2, ..., d_m\}$ conducted forward m times parallel and the loss are summed as Eq. 2

$\{d_1, d_2, ..., d_m\}$ conducted forward m times, where the loss function is calculated as follows:

$$\mathcal{L}_{MST_j} = L_{cls_j} + \alpha L_{reg_j}, \tag{1}$$

where $\mathcal{L}_{MST_j}$ corresponds to the loss function of the down-sampling factor d_j, weighted and summed by the classification loss and the regression loss. Each iteration will be updated by the loss summation of the reverse conduction computed by the network under m *df*s for the forward conduction:

$$\mathcal{L}_{MST} = \sum_{j=1}^{m} \mathcal{L}_{MST_j}, \tag{2}$$

However, simply using MST policy causes significantly worse performance in most *df* (Table I) relative to the SF policy that adopt a single *df* in both training and inference. This indicates that the network has not fully grasped the ability to handle multiple *df*s for inference. We conjecture that this is because feature maps produced by different *df*s differ substantially. In a perspective, the feature maps produced by different *df*s can be regarded as belonging to different domains. Therefore, inspired by [57], [58], we design an Adaptive Normalization Module (ANM) to handle the domain issue, making a unified detector compatible with different *df*s.

DFP Training. During the DFP training, the parameters of the detector network are fixed. The DFP aims to automatically select the smallest *df* that can achieve the best performance under limited computational resources for the input image. Since there are m choices for *df*, we model this problem as an m-class classification problem. The DFP takes the feature map M as input and passes through a lightweight structure to predict the selecting probability $p_s \in \mathbb{R}^m$ over m *df*s:

$$DFP(M) = p_s = [p_{d_1}, p_{d_2}, \ldots, p_{d_m}], \tag{3}$$

To obtain *df* predicted by DFP, the factor d_j corresponding to the highest probability p_{d_j} is selected as the *df* fed to the next stage of the backbone:

$$df = argmax_{d_j}\ p_{d_j} \tag{4}$$

Here is an extremely important issue for training DFP: how to obtain supervision for training DFP classifier. This issue will be addressed in Section III-C. Here we assume that we already have the supervision y_{d_j}. Then a traditional multi-label classification loss $\mathcal{L}_{df}$ is defined as follows:

$$\mathcal{L}_{df} = \sum_{j=1}^{m} -y_{d_j} \log\left(p_{d_j}\right) - \left(1 - y_{d_j}\right) \log\left(1 - p_{d_j}\right). \tag{5}$$

Since the prediction of *df* is closely related to the objects' size in the image, we added a statistical branch to assist the training of DFP. This statistical branch attempts to predict the statistical information $\hat{v}$ (e.g. mean, min, max) about the object's size in the image. This allows DFP to learn the scale information of the objects in the image, so as to better predict *df*. The Smooth L1 loss [59] is performed between predicted $\hat{v}$ and target label v as (β_{sta} is set as $1/9$, following [59]):

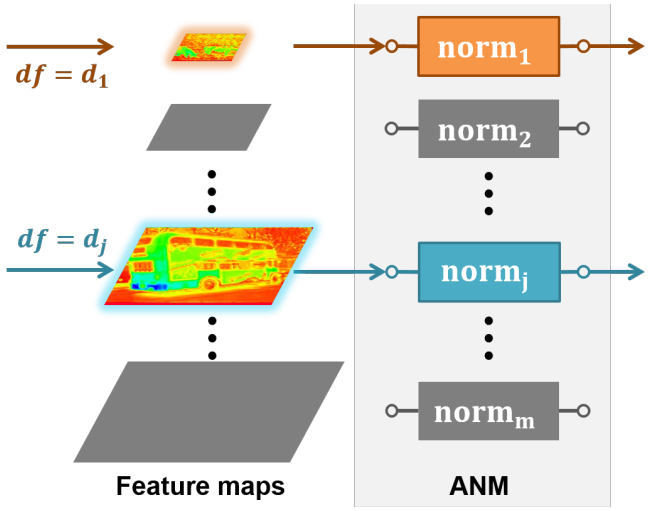

Fig. 6. The mechanism of the ANM. Different features scaled by different *df*s will be fed into ANM, and ANM will switch to the corresponding normalization layer for the *df*.

$$\mathcal{L}_{sta} = SmoothL1(\hat{v}, v) = \begin{cases} 0.5(\hat{v}-v)^2/\beta_{sta}, \ if \ |\hat{v}-v| \leq \beta_{sta}, \\ |\hat{v}-v| - 0.5 * \beta_{sta}, \ otherwise. \end{cases} \quad (6)$$

The objective function of DFP is a weighted summation of two losses:

$$\mathcal{L}_P = \mathcal{L}_{sta} + \lambda \mathcal{L}_{df}, \quad (7)$$

where $\lambda = 1.0$ (by default in this paper).

B. Adaptive Normalization Module

Considering the limitation of computational resources, we train a shared detector with the MST strategy. It can inference under different *df*s, enabling it to adapt to different computational loads for inference on various devices. To handle the distributions' differences in feature map introduced by different *df*s, inspired by [57], [58], we set different normalization layers for different *df*s and auto switch these layers with given *df*, which named Adaptive Normalization Module (ANM).

Normalization plays an important role in solving the internal covariate shift problem in deep neural network learning [60]. It can encode condition information to feature representations [61], [62]. It also can accelerate convergence and improve stability [63] through normalizing the input feature by re-centering and re-scaling:

$$y' = \gamma \frac{y-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta, \quad (8)$$

where y is the input to be normalized and y' is the output. γ, β are learnable scale and bias, μ, $\sigma^2$are the mean and variance of input.

To train a network with variable *df*s, *ANM* privatizes all normalization layers for each switch in a union network. It solves the problem of feature aggregation inconsistency between different switches by independently normalizing the feature mean and variance during testing. The scale and bias in *ANM* may be able to encode conditional information of *df* configuration of the current switch. Moreover, in contrast to incremental training, with *ANM* we can jointly train all switches at different *df*s; therefore all weights are jointly updated to achieve better performance.

As shown in Fig. 6, *ANM* decouples the normalization for each *df*, $d_j \in \{d_1, d_2, ..., d_m\}$ and chooses the corresponding normalization layer to normalize the features:

$$y' = \gamma_j \frac{y-\mu_j}{\sqrt{\sigma_j^2+\epsilon}} + \beta_j, j \in \{1, 2, \ldots, m\}, \quad (9)$$

where ϵ is a small number for numerical stability, μ_j and σ_j^2 are private averaged mean and variance from the activation statistics under separate *df*s; β_j and γ_j are private learnable scale weights and bias.

C. Down-sampling Factor Predictor

In order to choose the *df*s of a feature map, a *df*s predictor (DFP) network is designed. The goal of DFP is to predict an appropriate *df* that reduces the calculation consumption as much as possible while maintains a high detection performance, as described in Section III-A. In this section, we will introduce how to obtain supervision y_{d_j} and v for training DFP.

Algorithm 2 Guidance loss of DFP *P*

Require: Training set D_{train}, DPNet *M*

for I_i in D_{train} **do**
  for d_j in $dflist$ **do**
    Get the predicted result box $bboxes_i$ for $D(I_i)$.
    Get pos_i, neg_i samples by assigning $bboxes_i$.
    $neg_i = sorted(NMS(neg_i))$.
    $pos = []$
    for gt in I_i **do**
      $pos_* = argmax_{pos_* \in pos_i} IoU(pos_i, gt)$
      $pos = pos \cup pos_*$
    end for
    $k = length(pos)$
    $neg = top_k(neg_i)$
    $\mathcal{L}_i^{(j)}$ = loss of pos and neg.
  end for
end for

return $\mathcal{L}_i^{(j)}$

Guidance loss. How to obtain the supervision y_{d_j} for *df* classification? The category label of the DFP is related to the detector, that is to say, when the images are input to the detector with different *df*s, the *df* with high detection performance should be labeled as 1 (positive). Therefore, each image can be input to the detector to obtain the label of the DFP. If the performance is high, the *df* is labeled as 1 for the predictor, and if the performance is too poor compared to other

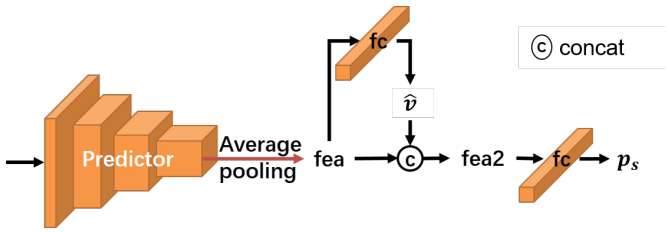

Fig. 7. The structure of DFP. DFP refers to and simplifies the structure of ResNet in its design. Additionally, a statistic branch is added to improve the effectiveness of prediction.

Train Method	Test df	mAP	AP_{50}	AP_s	AP_m	AP_l
SF	0.5	29.4	47.7	22.5	44.2	53.6
	0.33	25.3	41.5	17.4	42.9	52.9
	0.25	19.2	32.5	11.4	38.3	48.4
MF	0.5	24.1	40.4	17.9	37.1	45.9
	0.33	23.1	39.2	15.7	38.8	48.7
	0.25	21.3	36.2	13.6	37.4	50.3
MF + ANM	0.5	29.5	47.9	22.6	43.8	52.2
	0.33	28.9	47.6	21.0	46.8	57.1
	0.25	27.4	45.0	18.5	47.2	59.5

TABLE I
INFLUENCE OF ANM. SF MEANS TRAINING ONLY WITH SINGLE $df = 0.5$, MF MEANS TRAINING DETECTOR WITH $df = 0.5, 0.33, 0.25$.

Method	Flops	mAP	AP_{50}
Random-1	107.80 GFLOPs	28.90	47.50
Random-2	107.80 GFLOPs	29.00	47.70
Random-3	107.80 GFLOPs	28.90	47.60
Random(mean)	107.80 GFLOPs	28.93	47.60
DPNet(ours)	107.80 GFLOPs	29.70	48.50

TABLE II
DPNET VS. RANDOM FACTORS

dfs, it is labeled as 0 (negative). To measure the performance of a detector, one of the most useful metric is AP (average precision [64]). If a df gets the best AP performance among all dfs, then that df is the corresponding category label of this image in the DFP. However, it is impractical to use the AP metric to label each image due to AP of a single image is not representative. Another method is to calculate the loss function directly. The design of the loss function needs to be as relevant and close to the AP as possible. So much so that after entering the detector, we can directly determine the detection performance of the input image at a certain df based on this loss value. Therefore, in this paper, we designs a loss called the **guidance loss** noted as $L^{(j)}$, which is more closely related to the AP compared to the original loss of the detector. The guidance loss is calculated in Algorithm 2. The purpose of the guidance loss is not for optimizing the detector, but for obtaining supervised information about the DFP, which is simply designed as a function that is negatively correlated with the AP. The label y_{d_j} in Eq 5 is defined in the following way:

$$y_{d_j} = \begin{cases} 1, \ if \ \dfrac{L^{(j)}}{min(L^{(j)})} \leq T, \\ 0, \ otherwise. \end{cases} \quad j = 1, 2, ...m \qquad (10)$$

where T is is set to 1.1 by default, $L^{(j)}$ is the loss value calculated when $df = d_j$ by feeding the input image into the detector trained in detector training step (see Algorithm 1), $min(L^{(j)})$ is the minimum of all $L^{(j)}$s ($j = 1, 2, ...m$).

Statistic Branch. The rescaled feature map should better adapt to the detector. To this end, a branch to predict the characteristics of instances in the input image is added to the DFP, called *Statistic Branch*. Statistic Branch predicts a vector $\hat{v}$ containing the number of instances in the input image and the statistic values (*e.g.*the mean, maximum and minimum values) of their objects' size. **These values of supervision v can be easily obtained from the box annotation of the image.** As shown in Fig. 7, the structure of the predictor consists of 4 basic blocks. Each basic block consists of two convolution layers with batch normalization and ReLU activations.

IV. EXPERIMENTS

A. Experimental Setting

Dataset. TinyCOCO dataset (MS-COCO 2017) [24] is a widely-used benchmark to evaluate the detection performance of the tiny object detection tasks, which consists of 118k training images, 5k validation images and 20k testing images in 80 categories. All the studies are conducted on the validation set. TinyCOCO is the COCO100 [1], which resizes the shorter edge of each image to 100 and keeps the height-width ratio unchanged. The mean of objects' size in TinyCOCO is the same as the definition of the tiny object in [1]. **TinyPerson** is a commonly used dataset for tiny object detection, specifically designed for drone aerial photography scenarios. The dataset is collected from high-quality drone videos and online images. It contains 72,651 annotated human objects with low visual resolution, and a total of 25,945 images cropped to a size of 640x512, aimed at validating the model's applicability in drone-based object recognition tasks.

Evaluation Metric. According to Tinybenchmark[1], we mainly use Average Precision(AP) [64] for the tiny-scale performance evaluation, which is the most popular metric in various object detection tasks. It reflects both the precision and the recall ratio of objects' detection results. The threshold value of IoU is set to 0.25, 0.5, 0.75. Under the scenarios of the tiny-scale task, we pay far more attention to whether the target can be found than its location accuracy. Therefore, $IoU = 0.5$ naturally becomes the main evaluation criterion. Instead of the small, medium and large rules in the MS-COCO evaluation metric, we use tiny, small and reasonable as our basis for splitting scales. $\{mAP, AP_{50}, AP_{75}, AP_s, AP_m, AP_l\}$ is used as the reported evaluation metric. To evaluate the efficiency, we calculate the average GFLOPs. We use the average size of the validation set as the input size when calculating the FLOPs of the model under all different dfs. Then, the proportion of each df selected by the predictor is used as the weight to weighted sum each corresponding FLOPs. For TinyPerson, we follow [1] that divides tiny [2, 20] into 3 sub-intervals: tiny1[2, 8], tiny2[8, 12], tiny3[12, 20] and choose IoU= 0.5 as the threshold for evaluation.

Implement Details. The implements of DPNet are based on MMDetection [67]. Same to the default setting of the object detection on MS-COCO. And the stochastic gradient descent

Model	Params	Flops	↓ Flops	mAP	AP_{50}	AP_s	AP_m	AP_l
ResNet-50								
Faster R-CNN-ResNet-50	41.48 M	56.72	-	18.7	34.2	10.7	34.1	53.8
Faster R-CNN-ResNet-50 †	41.48 M	183.96	-	15.9	31.1	8.6	30.2	45.9
Cascade R-CNN-ResNet-50	69.17 M	84.36	-	18.0	32.3	9.7	34.4	53.7
Cascade R-CNN-ResNet-50 †	69.17 M	211.61	-	30.7	46.5	23.0	47.8	54.1
RetinaNet-ResNet-50	37.74 M	52.76	-	11.1	22.8	5.1	20.2	39.9
RetinaNet-ResNet-50 †	37.74 M	211.61	-	26.3	41.7	19.7	41.2	47.8
RepPoints-ResNet-50	36.62 M	41.99	-	18.7	34.2	10.7	34.1	53.8
RepPoints-ResNet-50 †	36.62 M	168.25	-	29.4	47.7	22.5	44.2	53.6
DPNet-ResNet-50 †(ours)	39.64 M	107.80	↓35.93%	29.7	48.5	22.0	46.5	56.6
ResNet-101								
RepPoints-ResNet-101	55.62 M	-	-	18.4	32.7	9.6	36.5	52.3
RepPoints-ResNet-101†	55.62 M	217.54	-	31.7	50.7	24.6	47.3	56.1
DPNet-ResNet-101†(ours)	58.64 M	157.78	↓27.47%	31.6	50.8	23.8	48.8	59.8
ResNeXt-50								
RepPoints-ResNeXt-50	35.44 M	-	-	19.5	34.5	10.7	37.9	59.0
RepPoints-ResNeXt-50†	35.44 M	168.25	-	29.9	48.3	23.0	44.5	52.8
DPNet-ResNeXt-50†(ours)	38.46 M	108.44	↓35.55%	29.8	48.2	22.9	44.7	53.1
MobileNet-v2								
RepPoints-MobileNet-v2	8.52 M	-	-	14.9	26.7	8.3	27.3	43.8
RepPoints-MobileNet-v2†	8.52 M	95.71	-	25.2	41.7	18.7	38.6	50.4
DPNet-MobileNet-v2†(ours)	11.06 M	56.08	↓41.40%	25.1	41.9	17.5	40.0	52.3

TABLE III
COMPARISON WITH OTHER MODEL COMPRESSION METHODS ON TINYCOCO DATASET.† MEANS THAT WE ENLARGE THE INPUT IMAGES TO 8 FOLDS. WE USE REPPOINTS AS OUR BASIC DETECTOR AND PERFORM ON DIFFERENT BACKBONES FOR COMPARISON. AP IS REPORTED FOR DETECTION PERFORMANCE AND GFLOPS IS REPORTED FOR COMPUTATION.

Model	Params	Flops	↓ Flops	AP_{tiny}	AP_{tiny1}	AP_{tiny2}	AP_{tiny3}
FreeAnchor [65]	37.81	222.23	-	44.26	25.99	49.37	55.34
Adap RetinaNet [5]	36.12 M	177.52	-	46.56	27.08	52.63	57.88
Grid RCNN [66]	64.32	189.15	-	47.14	30.65	52.21	57.21
RetinaNet-S-α [5]	36.12 M	177.52	-	48.34	28.61	54.59	59.38
RetinaNet-SM [1]	36.12 M	177.52	-	48.48	29.01	54.28	59.95
RepPoints	36.62 M	158.92	-	44.51	27.94	51.22	54.85
RepPoints-SM	36.62 M	158.92	-	48.88	33.79	55.10	58.48
RepPoints-SM †	36.62 M	635.66	-	52.17	39.37	56.59	61.22
DPNet †(ours)	39.64 M	471.74	↓25.79%	52.33	39.69	57.58	60.50

TABLE IV
COMPARISON WITH OTHER METHODS ON TINYPERSON DATASET.† MEANS THAT WE ENLARGE THE INPUT IMAGES TO 1.85 FOLDS. ALL METHODS APPLIES RESNET-50 AS BACKONE.

Method	Backbone	Sched.	Size	mAP
SSD	ResNet-50-FPN	1x	1024	29.86
R-FCN	ResNet-50-FPN	1x	1024	52.58
Faster-RCNN	ResNet-50-FPN	1x	1024	60.46
RepPoints	ResNet-50-FPN	1x	1024	59.44
DAPNet(ours)	ResNet-50-FPN	1x	1024	**63.55**

TABLE V
DPNET VS. OTHER METHODS ON DOTA DATASET

Model	**Input Size**	$AP_{50:95}$	**Params(M)**	**Flops (G)**
Faster-RCNN	640x640	17.0	82.3	65.3
YOLOV10m	640x640	27.08	15.4	59.1
YOLOV11m	640x640	28.06	20.1	68.0
YOLOV12m	640x640	29.02	20.2	67.5
DPNet-MobileNet-v2	640×640	**28.74**	**11.06**	**56.08**

TABLE VI
PERFORMANCE COMPARISON ON THE VISDRONE DATASET (HIGHER AP IS BETTER; LOWER PARAMS/FLOPS ARE BETTER)

(SGD [68]) algorithm is used to optimize in 1× training schedule over 8 GPUs with a total of 16 images per minibatch (2 images per GPU). The learning rate is set to 0.02 and decays by 0.1 at the 8-th and 11-th epochs, respectively. For candidate dfs of the large detector, we choose dfs of [0.5, 0.33, 0.25]

Guidance loss	mAP	AP_{50}	AP_s	AP_m	AP_l
	29.8	47.8	21.6	48.2	60.4
✓	30.9	50.1	23.0	47.8	59.4

TABLE VII
PERFORMANCE UPPER BOUND INFLUNCE OF GUIDANCE LOSS.

(which means m=3).

B. Ablation Study

Adaptive Normalization Module. As shown in Table I, we first train under the single 0.5 df and evaluate it on $\{0.5, 0.33, 0.25\}$ dfs. The performance is 29.4, 25.3 and 19.2 mAP, respectively. With mixed factors, we down-sample the image with 0.5, 0.33 and 0.25 respectively and optimize the shared detector head with respective losses. However, we find the performance decreases a lot (over 5 points) which shows that same normalization layer is not suitable for different dfs. So we use an adaptive normalization module, in which different dfs have different normalizations. The performance increases to 29.5, 28.9 and 27.4 mAP, which makes the detector obtain good performance under all down-samplings.

(a)

Size	mAP	AP_{50}	AP_s	AP_m	AP_l
(100, 167)	18.7	34.2	10.7	34.1	53.8
(150, 250)	23.4	40.2	14.6	41.4	57.7
(200, 333)	24.8	43.1	16.5	41.7	54.8

(b)

Depth	mAP	AP_{50}	AP_s	AP_m	AP_l
34	17.1	30.6	08.9	32.5	53.7
50	18.7	34.2	10.7	34.1	53.8
101	18.4	32.7	09.6	36.5	52.3
152	19.6	34.0	10.2	39.0	56.3

(c)

Width	mAP	AP_{50}	AP_s	AP_m	AP_l
0.5	16.4	29.4	08.4	31.9	50.6
1	18.7	34.2	10.7	34.1	53.8
1.5	19.2	34.1	10.3	37.3	54.5

TABLE VIII
PERFORMANCE COMPARISON: REPPOINTS DETECTOR USING DIFFERENT INPUT SIZES OR WITH BACKBONE USING DIFFERENT DEPTHS OR SIZES. THIS SHOWS THAT FOR TINY OBJECT DETECTION TASKS, INCREASING THE RESOLUTION TO LEADS TO GREATER PERFORMANCE GAINS THAN INCREASING THE DEPTH OR WIDTH OF THE NETWORK.

Down-Sampling Factor Predictor. In order to verify the effectiveness of our dfs predictor, we use a random strategy for comparison (s shown in Table II). We random sample the df based on the same GFLOPs to evaluate the performance. The operation is conducted three times and the mean performance is 28.93 mAP $\pm$ 0.05 and 47.60 AP_{50} $\pm$ 0.05. The performance with our designed predictor is 1 point higher than those, which shows that the result does not come from randomness and our predictor really works.

Guidance Loss. In order to verify the effectiveness of the guidance loss, the experiment first uses the optimization with and without the guidance loss as a df directly to the detector, whose performance indicates the upper limit of the performance under the supervision. The higher the upper bound, the better the supervised information. Table VII shows that the upper bound performance mmAP with the guidance loss is higher than that without the guidance loss by 1.1, which shows that the guidance loss is correlated with the detection performance, i.e., the lower the supervision loss, the higher the detection performance of the picture in the detector.

C. Performance Comparison

We compare the previous network with our DPNet on TinyCOCO dataset. As shown in Table III, different backbones for our detector are used for comparison, especially some lightweight networks like ResNeXt and MobileNet. These can verify that our method is not a succedaneum of lightweight models, but is combinative with those to save computation. Lightweight models save computation mainly through reducing the complexity of the model structure. Our method reduces computation from another aspect: adaptively adjusting the size of input. The baseline means the performance is based on TinyCOCO (COCO100) and the performance of enlarging is conduct on the input resized to 800, which brings the performance gain. This shows that enlarging images can increase detection performance, especially for tiny objects. The performance in the third line represents our method, which the image are enlarged (resized up) in to 800 and dynamic down-sampling is used for detection. Our basic backbone is ResNet-50, and the performance of our method is mAP 29.7. We use the larger ResNet-101 to demonstrate the consistent validity of our method, which shows that our DPNet can effectively reduce the amount of calculation.

Meanwhile, we further benchmark DPNet with the state-of-the-art approaches on the TinyPerson dataset, the popular dataset for tiny object detection tasks. We have added another three representative methods in tiny object detection task, namely RetinaNet-S-α, RetinaNet-SM. As shown in Table IV, DPNet is almost superior to all baselines. In AP_{tiny}, DPNet achieves 52.33, while RetinaNet-S-α, RetinaNet-SM are 48.34, 48.48 respectively. This indicates that our method can also achieve comparable performance with the enlarging policy under less computation cost on the TinyPerson dataset.

Additionally, our DPNet against state-of-the-art approaches on the DOTA dataset for Horizontal box object detection in TableV.DPNet achieves an AP of 63.55, compared to 60.46 and 59.44 for faster-rcnn and repPoints, resprectively. DPNet also achieves a SOTA efficiency-accuracy balance on the VisDrone dataset in TableVI, providing a superior solution for real-time small object detection in drone applications. Future work will focus on refining the dynamic strategy to further push the accuracy boundary.

D. Visualization

The prediction results of the DFP are visualized in Fig. 8. The top seven images with big sizes or obvious foregrounds which occupy most part of the whole images are predicted with 0.25 df. The middle seven images in which objects are smaller or lightly blurred are predicted with 0.33 df. The bottom seven images have really small objects, and their hidden foregrounds nearly blend with the background; thus the largest df is selected. Although the "easy" and "hard" examples may be different for humans and machines, these results are compatible with the human perception system. As shown in Fig. 9, we also compared the visualization results of DPNet, the small object domain method Scale Match, and Scale Match* using resized (up) input on the TinyPerson dataset, a real-world scenario. The results showed that DPNet had fewer missing detections at the same precision.

We present DPNet's visualization results for small object detection in low-computing scenarios, demonstrating its advantages over the baseline YOLOv11m model. All images are sourced from the VisDrone dataset test set. In scenarios with fewer objects, particularly for simple and common small targets, we compare the original image, YOLOv11m detection results, and DPNet-MobileNet-v2 detection results sequentially, as shown in Figure 10.

On the left, YOLOv11m fails to detect a truck in the bottom-right corner, while DPNet-MobileNet-v2 successfully identifies it. In the middle, a very small target missed by YOLOv11m is detected by our method. On the right, two tiny car targets near the upper-middle region are missed by YOLOv11m but accurately detected by DPNet-MobileNet-v2, capturing both category and position.

Figure 11 compares the performance of YOLOv11m and DPNet-MobileNet-v2 in dense detection scenarios. On the

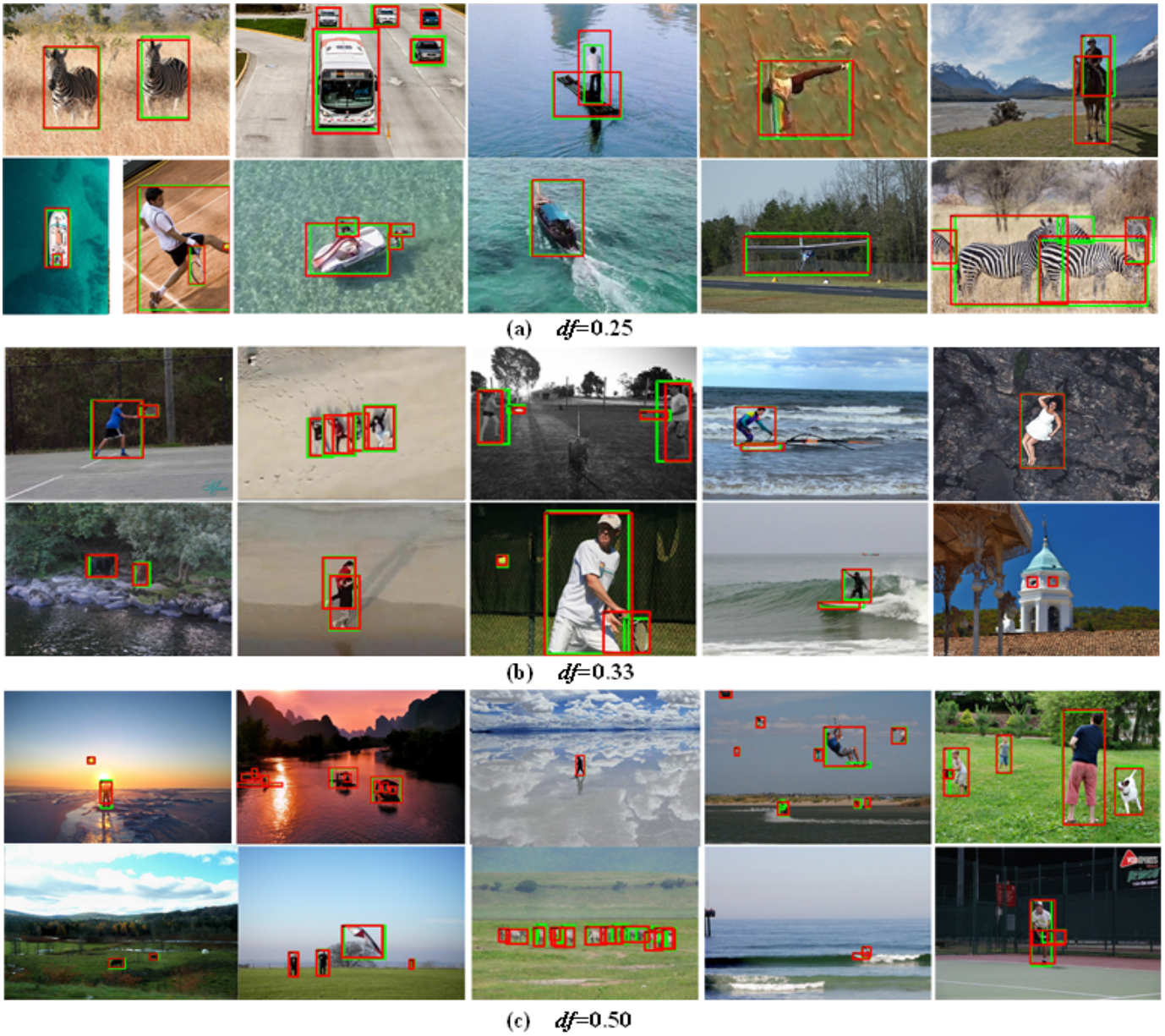

Fig. 8. Image visualization of DPNet on validation. Green boxes denote the ground truth and red boxes denote the predicted boxes. We conduct experiments on TinyCOCO, however visualize the results on MS-COCO for a better view.

left, YOLOv11m misses small targets at the farthest end, while DPNet mitigates this issue, though some farthest targets remain undetected. At the bottom, DPNet detects a pedestrian missed by YOLOv11m. On the right, where objects are highly concentrated, DPNet shows significantly higher detection density than YOLOv11m.

The visualization clearly demonstrates DPNet's superior performance. In comparison with the baseline method, DPNet shows greater accuracy with fewer false positives in general object detection. For planes, DPNet offers tighter bounding boxes. It also excels in detecting ships and buildings by providing more precise and correctly placed bounding boxes. Overall, DPNet delivers enhanced accuracy and precision across various object types, highlighting its effectiveness over the baseline method.

E. Analysis

Why Down-Sampling Factor? There are many ways to design a dynamic neural network, such as dynamic network depth, width, parameter, or feature map scale. We implement a series of experiments to simply analyze which factor is more important for the detection task. Table VIII shows that when the depth and width of the network are changed, the performance of the detector fluctuates within a narrow range, while resizing up the input image, the performance will dramatically raises. This is also why we pay attention to dynamic down-sampling.

V. Conclusion

This paper gives a visual analysis of the advantage of enlarging input images to detect tiny objects. Furthermore,

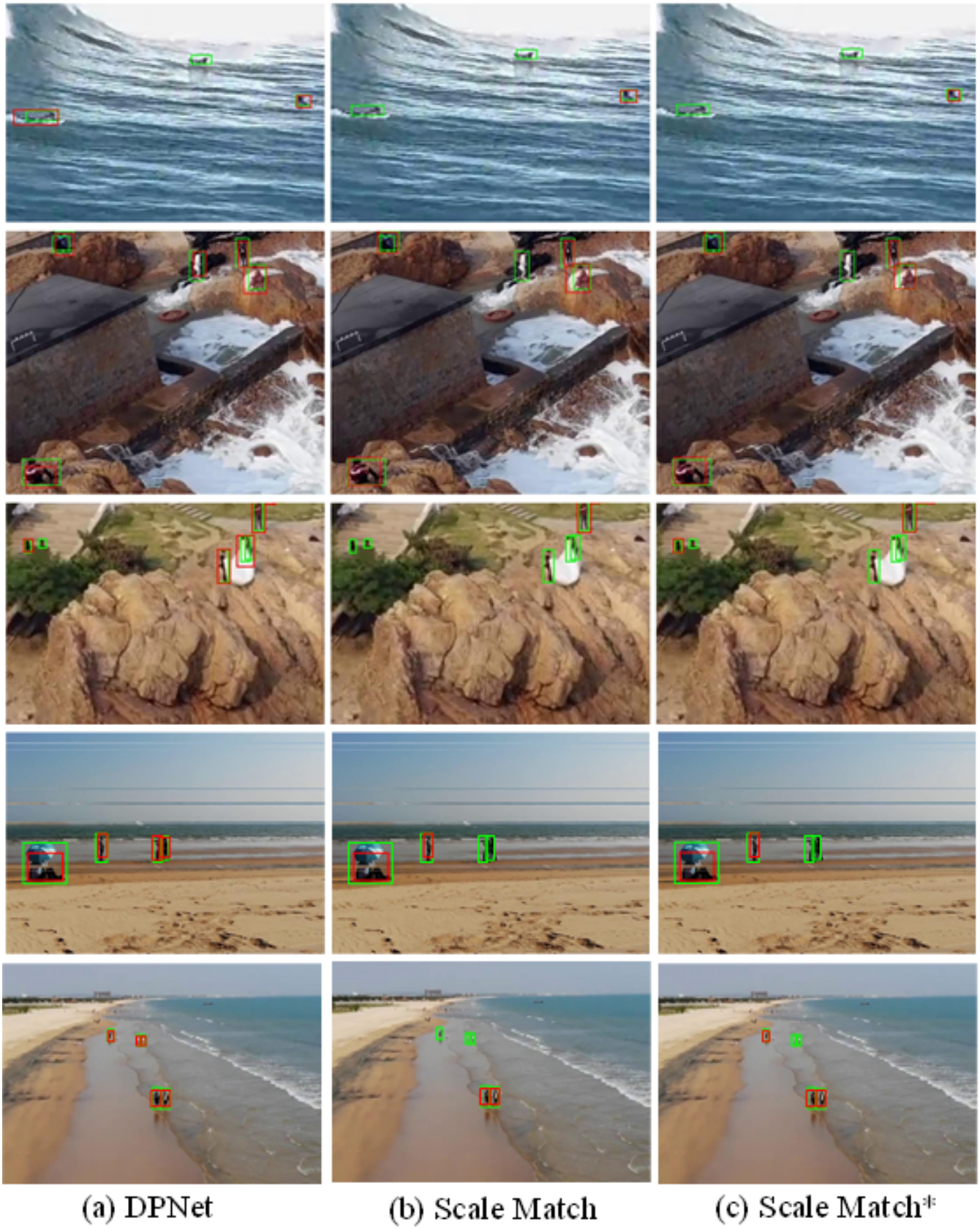

Fig. 9. Image visualization of DPNet and other methid on TinyPerson. Scale Match and Scale Match* are RepPoints-SM and RepPoints-SM† respectively.Green boxes denote the ground truth and red boxes denote the predicted boxes. It shows our DPNNet have less missing under same precision.

Fig. 10. Comparison between YOLOV11m (middle) and DPNet-MobileNet-v2 (bottom) in a simple scenario.

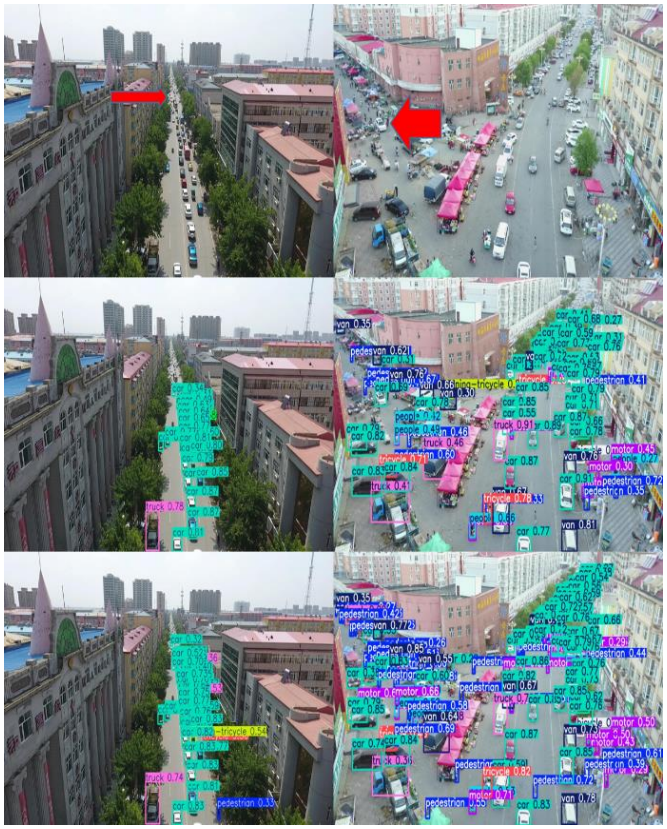

Fig. 11. Comparison between YOLOV11m (middle) and DPNet-MobileNet-v2 (bottom) in a dense detection scenario.

we propose a novel DPNet to adaptively select the most proper *df* of a feature map. ANM is designed to solve the problem of feature aggregation inconsistency between different *df* switches by privatizing all normalization layers for each switch in a union network. The guidance loss is designed to better optimize the detector's performance under each *df* in backbone through the way of re-weighting the loss of each instance in an image according to its size. The DFP in DPNet predicts the probability distribution of *df*s, helping to choose the performance-sufficient and cost-efficient *df* for each image. Thus, it can significantly reduce computational costs and enhance the application of recognition in unmanned aerial systems. As the first dynamic neural network suitable for detection tasks, DPNet can provide inspiration for more researchers in this field.

ACKNOWLEDGMENT

This work The authors would like to thank the reviewers and associate editor for their valuable feedback on the paper. The authors extend their appreciation to Researcher Supporting Project National Natural Science Foundation of China (Grant No.62472043, U21A20468), Beijing Natural Science Foundation (Grant No.4232050).

REFERENCES

[1] X. Yu, Y. Gong, and N. J. *et al.*, "Scale match for tiny person detection," in *IEEE Winter Conference on Applications of Computer Vision(WACV)*, 2020, pp. 1246–1254.

[2] F. Ozge Unel, B. O. Ozkalayci, and C. Cigla, "The power of tiling for small object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[3] N. Jiang, X. Yu, and X. P. *et al.*, "SM+: refined scale match for tiny person detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing,(ICASSP)*, 2021, pp. 1815–1819.

[4] B. Bosquet, M. Mucientes, and V. M. Brea, "Stdnet-st: Spatio-temporal convnet for small object detection," *Pattern Recognition*, vol. 116, p. 107929, 2021.

[5] Y. Gong, X. Yu, and Y. D. *et al.*, "Effective fusion factor in FPN for tiny object detection," in *IEEE Winter Conference on Applications of Computer Vision(WACV)*, 2021, pp. 1159–1167.

[6] X. Yu, Z. Han, Y. Gong, N. Jan, J. Zhao, Q. Ye, J. Chen, Y. Feng, B. Zhang, X. Wang, *et al.*, "The 1st tiny object detection challenge: Methods and results," in *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*. Springer, 2020, pp. 315–323.

[7] G. Cheng, X. Yuan, X. Yao, K. Yan, Q. Zeng, and J. Han, "Towards large-scale small object detection: Survey and benchmarks," *arXiv preprint arXiv:2207.14096*, 2022.

[8] X. Liang, J. Zhang, L. Zhuo, Y. Li, and Q. Tian, "Small object detection in unmanned aerial vehicle images using feature fusion and scaling-based single shot detector with spatial context analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 6, pp. 1758–1770, 2019.

[9] K. Duan, D. Du, H. Qi, and Q. Huang, "Detecting small objects using a channel-aware deconvolutional network," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 6, pp. 1639–1652, 2019.

[10] F. Fang, W. Liang, Y. Cheng, Q. Xu, and J.-H. Lim, "Enhancing representation learning with spatial transformation and early convolution for reinforcement learning-based small object detection," *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.

[11] X. Zheng, C. Lu, P. Zhu, and G. Yang, "Visual multitask real-time model in an automatic driving scene," *Electronics*, vol. 12, no. 9, p. 2097, 2023.

[12] Z. Yang, J. Li, and H. Li, "Real-time pedestrian and vehicle detection for autonomous driving," in *2018 IEEE intelligent vehicles Symposium (IV)*. IEEE, 2018, pp. 179–184.

[13] K. Rezaei, R. Hosseini, M. Mazinani, *et al.*, "A fuzzy inference system for assessment of the severity of the peptic ulcers," *Computer Science & Information Technology*, pp. 263–271, 2014.

[14] Y. Wu, H. Cao, G. Yang, T. Lu, and S. Wan, "Digital twin of intelligent small surface defect detection with cyber-manufacturing systems," *ACM Transactions on Internet Technology*, 2022.

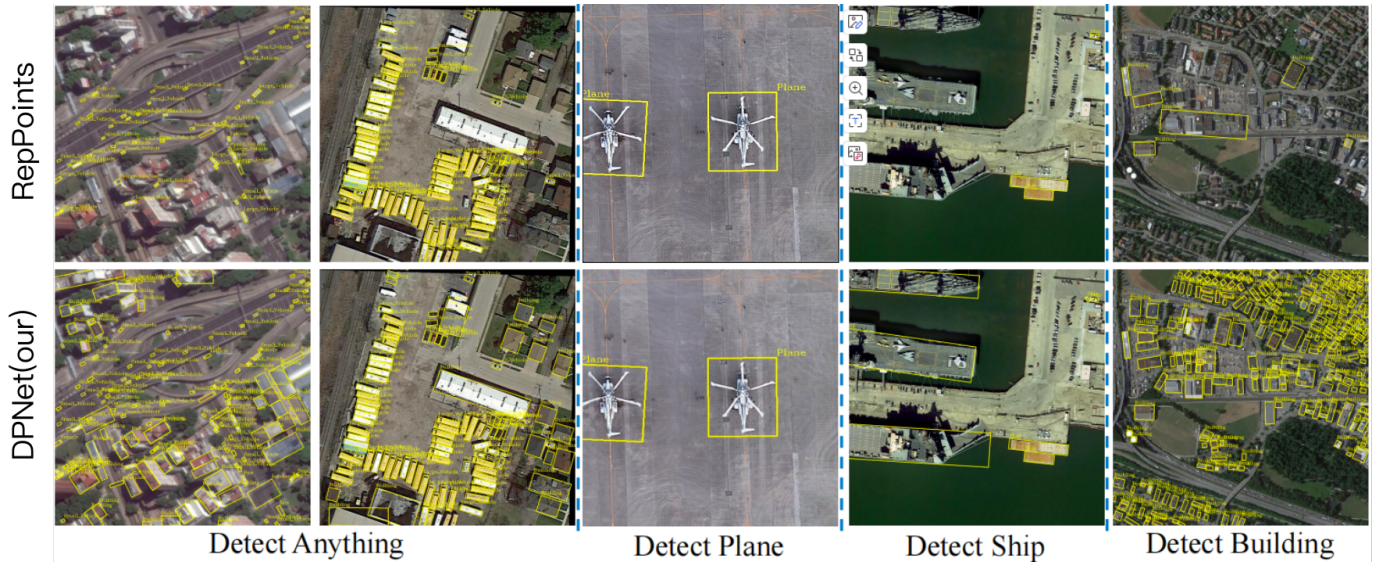

Fig. 12. Visualization results of RepPionts on the DOTA validation set before (the top row) and our DPNet (the bottom row), demonstrating four prompts: detecting any objects, detecting plane, detecting ship, and detecting buildings.

[15] K. Segl and H. Kaufmann, "Detection of small objects from high-resolution panchromatic satellite imagery based on supervised image segmentation," *IEEE Transactions on geoscience and remote sensing*, vol. 39, no. 9, pp. 2080–2083, 2001.

[16] J. Wang, G. Zhang, K. Zhang, Y. Zhao, Q. Wang, and X. Li, "Detection of small aerial object using random projection feature with region clustering," *IEEE Transactions on Cybernetics*, vol. 52, no. 5, pp. 3957–3970, 2020.

[17] M. Liu, X. Wang, A. Zhou, X. Fu, Y. Ma, and C. Piao, "Uav-yolo: Small object detection on unmanned aerial vehicle perspective," *Sensors*, vol. 20, no. 8, p. 2238, 2020.

[18] Z. Chen, H. Ji, Y. Zhang, Z. Zhu, and Y. Li, "High-resolution feature pyramid network for small object detection on drone view," *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.

[19] K. Cheng, H. Cui, H. A. Ghafoor, H. Wan, Q. Mao, and Y. Zhan, "Tiny object detection via regional cross self-attention network," *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

[20] T. Lin, P. Dollár, and R. B. G. *et al.*, "Feature pyramid networks for object detection," in *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2017, pp. 936–944.

[21] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, pp. 318–327, 2020.

[22] Z. Tian, C. Shen, H. Chen, and T. H. *et al.*, "Fcos: Fully convolutional one-stage object detection," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9626–9635.

[23] P. Sun, R. Zhang, Y. Jiang, T. Kong, C. Xu, W. Zhan, M. Tomizuka, L. Li, Z. Yuan, C. Wang, and P. Luo, "Sparse R-CNN: end-to-end object detection with learnable proposals," in *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2021, pp. 14 454–14 463.

[24] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," in *European Conference on Computer Vision (ECCV)*, 2014, pp. 740–755.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[26] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[28] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[29] C. Deng, M. Wang, L. Liu, Y. Liu, and Y. Jiang, "Extended feature pyramid network for small object detection," *IEEE Trans. Multim.*, pp. 1968–1979, 2022.

[30] J. Noh, W. Bae, and W. L. *et al.*, "Better to follow, follow to be better: Towards precise supervision of feature super-resolution for small object detection," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9724–9733.

[31] C. Xu, J. Wang, W. Yang, H. Yu, L. Yu, and G. Xia, "RFLA: gaussian receptive field based label assignment for tiny object detection," in *European Conference on Computer Vision (ECCV)*, 2022, pp. 526–543.

[32] J. Huang, V. Rathod, and C. S. *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2017, pp. 3296–3297.

[33] S. Liu, L. Qi, and H. Q. *et al.*, "Path aggregation network for instance segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2018, pp. 8759–8768.

[34] B. Singh and L. S. Davis, "An analysis of scale invariance in object detection SNIP," in *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2018, pp. 3578–3587.

[35] B. Singh, M. Najibi, and L. S. Davis, "SNIPER: efficient multi-scale training," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 9333–9343.

[36] Y. Li, Y. Chen, and N. W. *et al.*, "Scale-aware trident networks for object detection," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6053–6062.

[37] W. Liu, D. Anguelov, and D. E. *et al.*, "SSD: single shot multibox detector," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 21–37.

[38] Z. Cai, Q. Fan, and R. S. F. *et al.*, "A unified multi-scale deep convolutional neural network for fast object detection," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 354–370.

[39] A. Shrivastava, R. Sukthankar, and J. M. *et al.*, "Beyond skip connections: Top-down modulation for object detection," *CoRR*, vol. abs/1612.06851, 2016.

[40] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, "Multi-scale dense networks for resource efficient image classification," in *International Conference on Learning Representations(ICLR)*, 2018.

[41] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 2464–2469.

[42] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama, "Adaptive neural networks for efficient inference," in *International Conference on Machine Learning (ICML)*, 2017, pp. 527–536.

[43] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, pp. 7436–7456, 2022.

[44] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser,

"Universal transformers," in *International Conference on Learning Representations (ICLR)*, 2019.
[45] M. Elbayad, J. Gu, E. Grave, and M. Auli, "Depth-adaptive transformer," in *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2020.
[46] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.
[47] J. Yu, L. Yang, N. Xu, J. Yang, and T. S. Huang, "Slimmable neural networks," in *International Conference on Learning Representations (ICLR)*, 2019.
[48] J. Yu and T. S. Huang, "Universally slimmable networks and improved training techniques," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1803–1811.
[49] C. Li, G. Wang, and B. W. *et al.*, "Dynamic slimmable network," in *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*. Computer Vision Foundation / IEEE, 2021, pp. 8607–8617.
[50] L. Yang, Y. Han, and X. C. *et al.*, "Resolution adaptive networks for efficient inference," in *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2020, pp. 2366–2375.
[51] M. Zhu, K. Han, E. Wu, Q. Zhang, Y. Nie, Z. Lan, and Y. Wang, "Dynamic resolution network," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 27 319–27 330.
[52] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.
[53] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2018, pp. 4510–4520.
[54] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2017, pp. 5987–5995.
[55] K. He, X. Zhang, and S. Ren, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2016, pp. 770–778.
[56] Z. Yang, S. Liu, and H. H. *et al.*, "Reppoints: Point set representation for object detection," in *IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2019.
[57] W.-G. Chang, T. You, S. Seo, S. Kwak, and B. Han, "Domain-specific batch normalization for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2019, pp. 7354–7362.
[58] Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu, "Adaptive batch normalization for practical domain adaptation," *Pattern Recognition*, vol. 80, pp. 109–117, 2018.
[59] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
[60] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning (ICML)*, 2015, pp. 448–456.
[61] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville, "Film: Visual reasoning with a general conditioning layer," in *Association for the Advance of Artificial Intelligence (AAAI)*, 2018, pp. 3942–3951.
[62] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou, "Revisiting batch normalization for practical domain adaptation," in *International Conference on Learning Representations (ICLR)*, 2017.
[63] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *International Conference on Learning Representations (ICLR)*, 2016.
[64] M. Zhu, "Recall, precision and average precision," *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, vol. 2, no. 30, p. 6, 2004.
[65] X. Zhang, F. Wan, C. Liu, R. Ji, and Q. Ye, "Freeanchor: Learning to match anchors for visual object detection," *Advances in neural information processing systems*, vol. 32, 2019.
[66] X. Lu, B. Li, Y. Yue, Q. Li, and J. Yan, "Grid r-cnn," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7363–7372.
[67] K. Chen, J. Wang, and J. P. *et al.*, "Mmdetection: Open mmlab detection toolbox and benchmark," *CoRR*, vol. abs/1906.07155, 2019.
[68] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade - Second Edition*. Springer, 2012, pp. 421–436.

Luqi Gong (Student member, IEEE) received his B.Eng. and M.Eng. degree from Shandong University of Science Technology, Qingdao, China in 2017 and Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China in 2020, respectively. He is currently purchasing the Ph.D. degree in Beijing University Of Posts and Telecommunications. His current research interests include satellite computing, acceleration of deep neural network and embedded intelligent computing system.

Haotian Chen is currently pursuing a Bachelor's degree in Computer Science and Technology at Southwest Jiaotong University, Chengdu, Sichuan, China. With a strong academic focus on computer vision, his research interests include semantic segmentation and action recognition. In addition to his academic achievements, Haotian has demonstrated exceptional leadership and technical expertise as a project leader and technical director. His innovative contributions have earned him several prestigious accolades, including the National First Prize in the National Creative Crane Competition for Mechanical Innovation for College Students and the National Third Prize in the Challenge Cup National College Students' Innovation and Entrepreneurship Competition.

Yikun Chen (Member, IEEE) received his B.Eng. and M.Eng. degree from South China Normal University, Guangzhou, China in 2011 and North Borneo University, Malaysia in 2021, respectively. His research interests encompass the application of computer vision technology in the intelligent transformation of construction sites and smart parks. He was honored with the Guangdong Province Zhuhai City Software Innovation Talent Award in 2014.

Tianliang Yao (Student Member, IEEE) is currently pursuing a B.Eng. degree in automation at Tongji University, Shanghai, China. His current research interests include computer vision, medical image analysis and embodied intelligence in medical robotics. He was a recipient of the Best Conference Paper Finalist Award at the IEEE International Conference on Advanced Robotics and Mechatronics (ARM) in 2023 and IEEE ICRA RAS Travel Grant Award in 2025.

Chao Li (Member, IEEE) is a principal investigator at Zhejiang Lab, Hangzhou, China, and an associate professor at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. His research interests include interpretable learning, machine learning, data mining, and edge intelligence systems. He has published more than 20 papers in the prestigious refereed journals and conference proceedings, such as AAAI, ICCV, ICML etc.

Shuai Zhao (Member, IEEE) received the Ph.D. degree in computer science and technology from the Beijing University of Posts and Telecommunications in 2014. He is currently a Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. His current research interests include the Internet of Things Technology, distributed learning, and intelligent systems.

Guangjie Han (Fellow, IEEE) is currently a Professor with the Department of Internet of Things Engineering, Hohai University, Changzhou, China. He received his Ph.D. degree from Northeastern University, Shenyang, China, in 2004. In February 2008, he finished his work as a Postdoctoral Researcher with the Department of Computer Science, Chonnam National University, Gwangju, Korea. From October 2010 to October 2011, he was a Visiting Research Scholar with Osaka University, Suita, Japan. From January 2017 to February 2017, he was a Visiting Professor with City University of Hong Kong, China. From July 2017 to July 2020, he was a Distinguished Professor with Dalian University of Technology, China. His current research interests include Internet of Things, Industrial Internet, Machine Learning and Artificial Intelligence, Mobile Computing, Security and Privacy. Dr. Han has over 500 peer-reviewed journal and conference papers, in addition to 160 granted and pending patents. Currently, his H-index is 75 and i10-index is 341 in Google Citation (Google Scholar). The total citation count of his papers raises above 20000 times.

Dr. Han is a Fellow of the UK Institution of Engineering and Technology (FIET). He has served on the Editorial Boards of up to 10 international journals, including the IEEE TII, IEEE TCCN, IEEE Systems, etc. He has guest-edited several special issues in IEEE Journals and Magazines, including the IEEE JSAC, IEEE Communications, IEEE Wireless Communications, Computer Networks, etc. Dr. Han has also served as chair of organizing and technical committees in many international conferences. He has been awarded 2020 IEEE Systems Journal Annual Best Paper Award and the 2017-2019 IEEE ACCESS Outstanding Associate Editor Award. He is a Fellow of IEEE.