Effective Fusion Factor in FPN for Tiny Object Detection

Yuqi Gong[§†] Xuehui Yu[§†] Yao Ding[†] Xiaoke Peng[†] Jian Zhao[‡] Zhenjun Han[†*]
[†]University of Chinese Academy of Sciences, Beijing, China
[‡]Institute of North Electronic Equipment, Beijing, China
{gongyuqi18, yuxuehui17, dingyao16, pengxiaoke19}@mails.ucas.ac.cn
zhaojian@u.nus.edu, hanzhj@ucas.ac.cn

Abstract

FPN-based detectors have made significant progress in general object detection, e.g.*, MS COCO and PASCAL VOC. However, these detectors fail in certain application scenarios,* e.g.*, tiny object detection. In this paper, we argue that the top-down connections between adjacent layers in FPN bring two-side influences for tiny object detection, not only positive. We propose a novel concept, fusion factor, to control information that deep layers deliver to shallow layers, for adapting FPN to tiny object detection. After series of experiments and analysis, we explore how to estimate an effective value of fusion factor for a particular dataset by a statistical method. The estimation is dependent on the number of objects distributed in each layer. Comprehensive experiments are conducted on tiny object detection datasets,* e.g.*, TinyPerson and Tiny CityPersons. Our results show that when configuring FPN with a proper fusion factor, the network is able to achieve significant performance gains over the baseline on tiny object detection datasets. Codes and models will be released.*

1. Introduction

Tiny object detection is an essential topic in the computer vision community, with broad applications including surveillance, driving assistance, and quick maritime rescue.

FPN-based detectors, fusing multi-scale features by top-down and lateral connection, have achieved great success on commonly used object detection datasets, *e.g.*, MS COCO [17], PASCAL VOC [9] and CityPersons [34]. However, these detectors perform poorly on tiny object detection, *e.g.*, TinyPerson [33] and Tiny CityPersons [33][1]. An intuitive question arises: why current FPN-based detectors unfit tiny object detection and how to adapt them to tiny object detection.

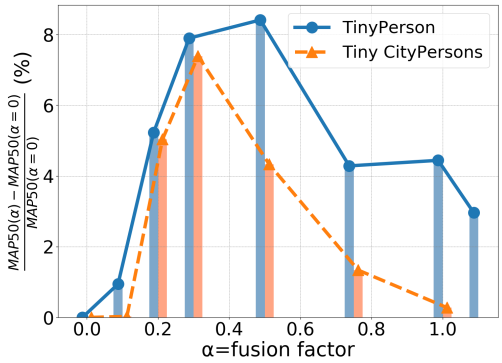

Figure 1. The performance based on different fusion factors on TinyPerson and Tiny CityPersons. The y-axis shows the performance improvement of AP_{50}^{tiny} when given a fusion factor. Fusion factor denotes the coefficient weighted on the deeper layer when fusing features of two adjacent layers in FPN.(Best viewed in color)

The motivation to answer the problem origins from an interesting phenomenon when analyzing experimental results of tiny object detection using FPN. As shown in Fig. 1, the phenomenon is that the performance first increases and then decreases along with the increasing of information that deep layers delivering to shallow layers. We define fusion factor as the coefficient weighted on the deeper layer when fusing feature of two adjacent layers in FPN.

We further explore why the phenomenon occurs by analyzing the working principle of FPN. We find that FPN indeed is multi-task learning due to the fusion operation of adjacent layers. To be more specific, if omitting top-down connection in FPN, each layer only needs to focus on detecting objects with the highly relevant scale, for example, shallow layers learn small objects and deep layers learn large objects. However, in FPN, supervised by losses from other layers indirectly, each layer nearly needs to learn all size objects, even the deep layers need to learn small objects. For tiny object detection, there exist two facts need to be considered. The first one is that small objects domi-

*corresponding author §indicates equal contribution.
[1]Tiny CityPersons is obtained by four times of down-sampling of CityPersons.

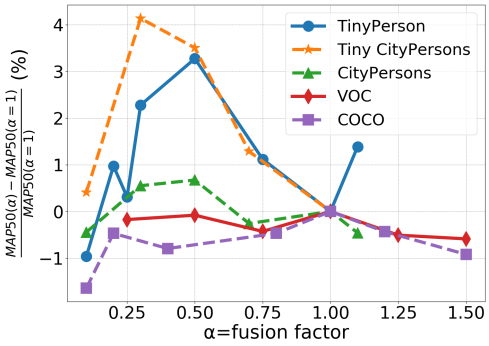

Figure 2. The performance based on different fusion factors under AP_{50}^{all} on different datasets. The y-axis shows the performance improvement when given a fusion factor. The performance of TinyPerson and Tiny CityPersons fluctuates with different fusion factors, while the performance of CityPersons, PASCAL VOC and MS COCO are relatively stable. (Best viewed in color)

nate the dataset and the second one is that the dataset is not large. Therefore, each layer not only needs to focus on its corresponding scale objects, but also needs to get help from other layers for more training samples. The fusion factor controls the priorities of these two requirements and then get a balance of them. The conventional FPN corresponds to that fusion factor is 1, improper for tiny object detection. In light of this, firstly, we explore how to explicitly learn effective fusion factor in FPN from several aspects, for improving the performance of FPN for tiny object detection. An effective value of fusion factor for a particular dataset is estimated by a statistical method, which is dependent on the number of objects distributed to each layer. Secondly, we further analyze whether fusion factor can be learned implicitly from two aspects. Finally, we explain the rationality of designing α for tiny object detection in the respective of gradient backpropagation. Extensive experimental results indicate that FPN fusion factor provides a significant boost to the performance of commonly used FPN for tiny object detection. The main contributions of our work include:

1. We propose a new concept, fusion factor, to describe the couple degree of adjacent layers in FPN.
2. We analyze how the fusion factor affects the performance of tiny object detection and further investigate how to design an effective fusion factor for improving performance. Moreover, we provide mathematical explanation in details.
3. We show that significant performance improvements over the baseline on tiny object detection can be achieved by setting an proper fusion factor to FPN.

2. Related Work

2.1. Dataset for Detection

To deal with various challenges in object detection, many datasets have been reported. MS COCO [17], PASCAL VOC [9] and ImageNet [1] are for general object detection. IVIS [12] is also for general object detection; however, it has a long tail of categories in images. There are some datasets applied to specific detection tasks. [4, 8, 31, 7, 6, 10, 34] are scene-rich and well-annotated datasets used for pedestrian detection task. WiderFace [32] mainly focuses on face detection, TinyNet [24] involves remote sensing object detection in a long-distance and TinyPerson [33] is for tiny person detection, whose average absolute size is nearly 18 pixels. In this paper, we focus on the tiny person detection, and the TinyPerson and Tiny CityPersons are used for experimental comparisons.

2.2. Small Object Detection

Extensive research has also been carried out on the detection of small objects. [33] proposes scale matching that aligns the object scales from the pretrain dataset to targets dataset for reliable tiny-object feature representation. SNIP [25] and SNIPER [26] use a scale regularization strategy to guarantee the size of the object is in a fixed range for different resolution images. SNIPER uses the method of region sampling to further improve training efficiency. Super Resolution(SR) is used to recover the information of low-resolution objects; therefore, it is introduced to small object detection. EFPN [5] constructs a feature layer with more geometric details, which is designed for small objects via SR. Noh et al. [22] propose a feature level super-resolution approach using high-resolution object features as supervision signals and matching the relevant receptive fields of input and object features. Chen et al. [3] propose a feedback-driven data provider to balance the loss for small object detection. TridentNet [14] constructs parallel multi branches of different receptive fields and generates more discriminative small objects' features to improve the performance. These methods mentioned above improve the performance of small object detection to some extent.

2.3. Feature Fusion for Object Detection

In the deep network, the shallow layers are generally lack of abstract semantic information and rich in geometric details. In contrast, the deep layer is just the opposite of the shallow layer. FPN [15] merges deep layer and shallow layer features in a top-down way to build a feature pyramid. PANet [19] proposes a bottom-top way to help deep layer object recognition with shallow layer detailed features. Kong [13] proposed the method of global attention and local reconfiguration, which combined the high-level semantic features with the low-level representation to reconstruct the feature pyramid. MHN [2] is a multi-branch and high-level semantic network proposed to solve the semantic gap problem of merging different feature maps. In the process of addressing semantic inconsistence problem, it significantly improves performance on

detecting small-scale objects.. Nie [21] introduces a feature enrichment scheme to generate multi-scale contextual features, HRNet [27] performs multi-scale fusion through repeated cross parallel convolution to enhance feature expression, and Libra-RCNN [23] uses the fusion results of all feature layers to reduce the unbalance between feature maps. ASFF [18] predicts the weight of features from different stages via a self-adaptive mechanism when fused again. SEPC [30] proposes pyramid convolution to improve the efficiency of feature fusion of adjacent feature layers. Nas-FPN [11] explores the optimal combination way for feature fusion of each layer using AutoML. Tan [28] proposes the learnable weight of feature fusion in BiFPN. These approaches further improved the effect of feature fusion from different aspects. However, they all ignore that feature fusion is affected by dataset scale distribution.

3. Effective fusion factor

Two main elements affect the performance of FPN for tiny person detection, including the downsampling factor and the fusion proportion between adjacent feature layers. Previous studies have explored the former element, and conclude that the lower downsampling factor is, the better the performance will be, despite the increased computational complexity. However, the latter element has been ignored.

FPN aggregates adjacent feature layers in the following manner:

$$P_i = f_{layer_i}(f_{inner_i}(C_i) + \alpha_i^{i+1} * f_{upsample}(P_{i+1}^{'})), \quad (1)$$

where f_{inner} is a 1×1 convolution operation for channels matching, $f_{upsample}$ denotes the $2\times$ upsampling operation for resolution matching, f_{layer} is usually a convolution operation for feature processing, and α denotes fusion factor. The conventional detectors set α to 1. The black dashed box on the right of Fig. 4 shows this process. In practice, if FPN fuses features from level P_2, P_3, P_4, P_5, P_6, there are three different α, including α_2^3, α_3^4 and α_4^5, which represent the fusion factors between two adjacent layers, respectively. (Since P_6 is generated by directly downsampling the P_5, there is no fusion factor between P_5 and P_6). The proportion of features from different layers, when they are fused, are adjusted by setting different α. In the following, the fusion factor will be deeply investigated and analyzed.

3.1. What affect the effectiveness of fusion factor?

To explore how to obtain the effective α, we first investigate what can affect the effectiveness of fusion factor. We hypothesize that four attributes of dataset affect α: 1. The absolute size of objects; 2. The relative size of objects; 3. The data volume of the dataset; 4. The distribution of objects in each layer in FPN.

Firstly, we conduct experiments to evaluate the fusion factors effect on different datasets. The experimental results are given in Fig. 2. Different datasets exhibit different trends, *e.g.*, the curve peak value, under different fusion factors. The cross-scale datasets, CityPersons, VOC, and COCO, are not sensitive to the variation of α, except when $\alpha = 0$, which corresponds to no feature fusion. However, on TinyPerson and Tiny CityPersons, the performance increases first and then decreases with the increase of α, which means that the fusion factor is a crucial element for performance, and there exists an optimal value range. In this paper, the fusion factor greater than 1.1 is not conducted due to the difficulty of converging on TinyPerson, Tiny CityPersons, and CityPersons.

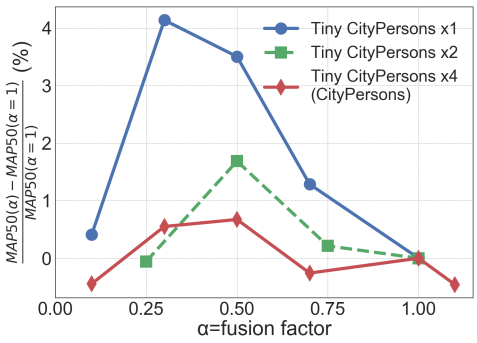

Figure 3. The performance based on different fusion factor under AP_{50}^{all} on different datasets: Tiny CityPersons upsampled $\times 1$, $\times 2$ and Cityperson, respectively. (Best viewed in color)

The common characteristic of TinyPerson and Tiny CityPersons is that the average absolute size of instances is less than 20 pixels, which brings a great challenge to network learning. Therefore, we resize images in CityPersons and COCO to obtain different datasets (Images in CityPersons are zoomed out by 2 times and 4 times, and in COCO by 4 times and 8 times, respectively). As shown in Fig. 3, when the absolute size of the objects is reduced, the trends of performance with the change of α become similar to that of TinyPerson. For Tiny CityPersons and CityPersons, the amount of data and the relative size of the objects are precisely the same; however, the performance changes differently when the fusion factors increase.

The distribution of objects in each layer in the FPN will determine whether the training samples are sufficient or not, which directly affects the feature representation in each layer. CityPersons shares similar stratification of FPN with TinyPerson and Tiny CityPersons. Although Tiny CityPersons are obtained by 4 times of downsampling of CityPersons, the stratification of CityPersons in FPN is still similar to that of Tiny CityPersons since the anchor of Tiny CityPersons is also reduced by four times. To be specific, a large number of tiny objects are concentrated in P_2, and P_3, which brings about those objects in deep layers of FPN are insufficient. However, the trend of performance to fusion factor on CityPersons differs from that of TinyPerson and Tiny CityPersons.

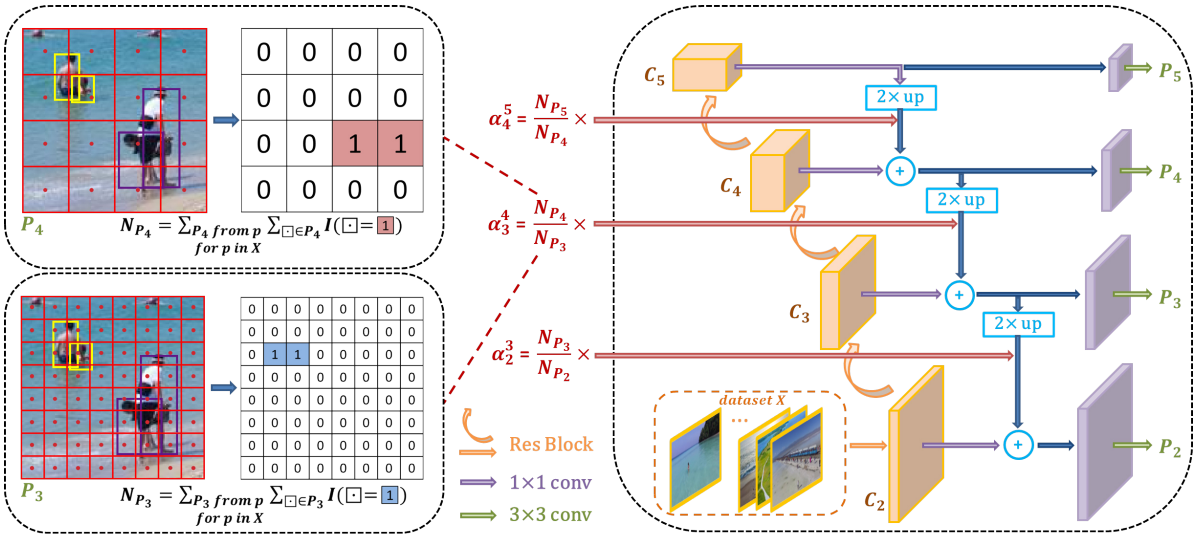

Figure 4. The framework of our method. The dotted boxes on the left show the calculation of N_P, where 1 and 0 are positives and negatives, respectively. The image is from TinyPerson. Red boxes and red points represent anchor boxes and anchor points. For simplification, only one anchor is displayed with a anchor point. Yellow box and blue box are ground-truth on P_3 and P_4 layer, respectively. The dotted box on the right is the framework of original FPN. We can obtain the effective fusion factor α by statistic-based method.

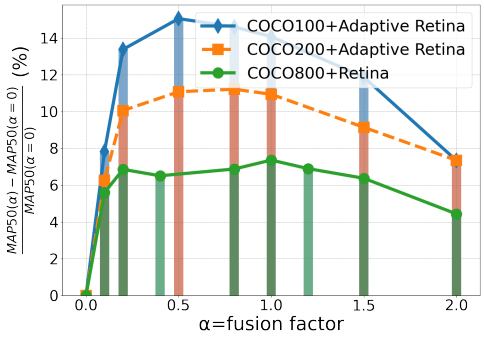

Figure 5. The performance based on different fusion factor under AP_{50}^{all} of different input sizes of MS COCO, showing the influence of the absolute size of objects. And the Adaptive RetinaNet builds FPN using P_2, P_3, P_4, P_5, P_6. (Best viewed in color)

Therefore, we conclude that the absolute size of objects rather than the other three factors exactly affects the effectiveness of the fusion factor. Accordingly, why and how fusion factor works are given as follows. α determines how degree deep layers in FPN participate in the learning of shallow layers by reweighting loss in gradient back propagation. The object in the dataset is tiny-size, which brings many difficulties for the learning of each layer in FPN. Hence, each layer's learning capability is not enough, and the deep layers have no extra ability to help the shallow layers. In other words, the supply-demand relationship between deep layers and shallow layers in FPN has changed when the learning difficulty of each layer increases and α has to be reduced, indicating that each layer should be more focused on the learning of this layer.

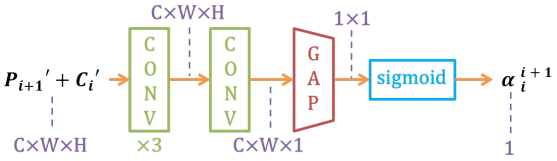

Figure 6. The network structure of self-attention.

3.2. How to obtain an effective fusion factor?

To further explore how to get an effective fusion factor, we design four kinds of α and conduct experiments on TinyPerson: 1. A brute force solution, which enumerates α according to the Fig. 1. 2. A learnable manner, where α is set as a learnable parameter that is optimized by the loss function. 3. An attention-based method, where α is generated by the self-attention module, illustrated in the Fig. 6. 4. A statistic-based solution, which utilizes the statistical information of datasets to compute α, calculated as follows:

$$\alpha_i^{i+1} = \frac{N_{P_{i+1}}}{N_{P_i}}. \tag{2}$$

where $N_{P_{i+1}}$ and N_{P_i} represent the number of objects on the layer P_{i+1} and P_i in FPN, respectively. The quantitative experiments of the four methods are given in the Tab. 1. Accordingly, we summarize several conclusions.

Firstly, the brute force search explores the best α. Nevertheless, it contains redundant computations, which limits its large-scale applications. Secondly, all non-fixed α settings are superior to the baseline, where α is set as 1, the attention-based method increases the amount of calculation that cannot be negligible. Thirdly, only the statistic-based

Method	AP_{50}^{tiny}	MR_{50}^{tiny}
baseline	46.56	88.31
one-α	46.86	88.31
three-α	47.66	87.98
atten-α	47.88	87.80
bf-α	48.33	87.94
S-α	48.34	87.73

Table 1. The performance of AP_{50}^{tiny} on TinyPerson based on different calculation strategies of fusion factor. α in baseline is set to 1 by default. one-α and three-α represent using one and three learnable parameters, respectively. atten-α is obtained via attention mechanism. α-bf represents the optimum via brute force solution. The performance of S-α is obtained via RetinaNet with S-α. Lower MR(Miss Rate) means better performance.

approach achieves comparable performance with that obtained by the brute force search.

The statistic-based method, named as S-α, sets α according to the proportion of the object number between adjacent layers in the FPN, as shown Eq. 2. The object number is counted from the whole dataset. We design the formulation based on the fact that for tiny object detection, it is hard for each layer to capture representative features for detection tasks, which intensifies competition between layers. More concretely, all layers in different heads want parameters they share to learn proper features for their corresponding detection tasks. Unfortunately, some layers may have much less training samples than others, leading to that when updating shared parameters, the gradient of these layers is at a disadvantage compared with other layers. Therefore, when $N_{P_{i+1}}$ is small, or N_{P_i} is large, the method sets a small α to reduce the gradient generated by the detection task in P_i layer, vice versa, which prompts the network to learn detection tasks in each layer equally. Thus, the learning efficiency of tiny objects is improved.

The statistic procedure of N_P and calculation of α are as follows: 1) Taking IoU as a principle, we choose the anchors with the largest IoU with ground-truth as positive in an image. 2) Based on the positive anchors and the predefined number of anchors in each layer, the number of ground-truth in each layer is calculated. 3) Repeat steps 1 and 2 on each image in the dataset to get a statistical result, and then we calculate the α according to Eq. 2, as shown in the left dashed box of Fig. 4. The calculation procedure does not involve the front propagation of networks since the anchor is predefined, and ground-truth is provided by the dataset. Details are given in Alg. 1.

3.3. Can fusion factor be learned implicitly?

In Section 3.2, the effective α is explored by explicit learning. We further discuss whether the fusion factor could be learned implicitly in this section.

Algorithm 1 Fusion Factor Statistics

INPUT: M (M denotes IoU matrixes of all images, M_i is the IoU matrix of i^{th} image.)
INPUT: A (A denotes the set of all predefined anchors in the FPN, A_i denotes the set of preset anchors of i^{th} layer.)
INPUT: $List_{tn}$ (A list which records the number of ground-truth allocated to each layer, tn represents total number of [N_{P_2},N_{P_3},N_{P_4},N_{P_5},N_{P_6}].)
OUTPUT: $List_\alpha$ (A list of α_3^2,α_4^3,α_5^4)
NOTE: 1. According to the IoU matrix, $MatchGT()$ selects the anchor with max IoU with a specified ground-truth as a positive example; 2. $CalNumAnchor()$ counts the number of anchors matching to the ground-truth of the corresponding layer; 3. R represents the intermediate result outputed by $MatchGT()$, R_i is the result of i^{th} image.

$list_{tn} \leftarrow [0,0,0,0,0]$
$R \leftarrow \emptyset$
for M_i *in* M **do**
    $R \leftarrow MatchGT(M_i)$
end for
for R_i *in* R **do**
    for A_i *in* A **do**
        N_{P_i} += $CalNumAnchor(A_i, R_i)$
    end for
end for
for N_{P_i} *in* $List_{tn}$ **do**
    if $i < 4$ **then**
        $\alpha_i^{i+1} = \frac{N_{P_{i+1}}}{N_{P_i}}$
    else
        $\alpha_i^{i+1} = \frac{N_{P_{i+1}}+N_{P_{i+2}}}{N_{P_i}}$
    end if
    $List_\alpha \leftarrow \alpha_i^{i+1}$
end for
return $List_\alpha$

method	AP_{50}^{tiny}	MR_{50}^{tiny}
baseline	46.56	88.31
0.5-power + α=1	46.94	87.98
0.5-power + α=0.5	48.17	87.17

Table 2. Detection results of σ-power initialization on TinyPerson

Firstly, we analyze the structure of the FPN and find an equivalent realization of the fusion factor. In the conventional FPN (α = 1), multiplying the parameters of f_{inner_i} by σ^{i-2} and dividing the parameters of f_{layer_i} by σ^{i-2} is equivalent to keeping f_{inner_i}, f_{layer_i} fixed and setting α = σ. Thus, the conventional FPN has the latent ability to learn the effective α implicitly. We further investigate how to activate the ability by adjusting initialization of parameters of f_{inner_i}. We carry out experiments with different initialization of f_{inner_i} and f_{layer_i} by multiplying their cor-

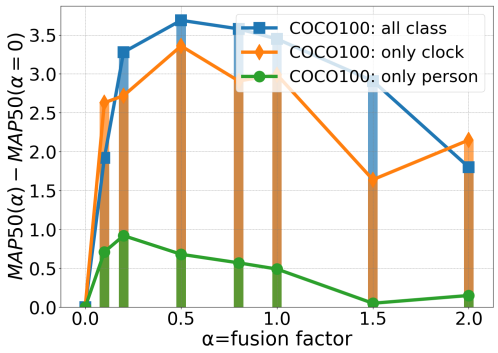

Figure 7. The performance of clock, person and all classes in COCO100, which have 6587 clock instances, 268030 person instances and 886284 instances totally. (Best viewed in color)

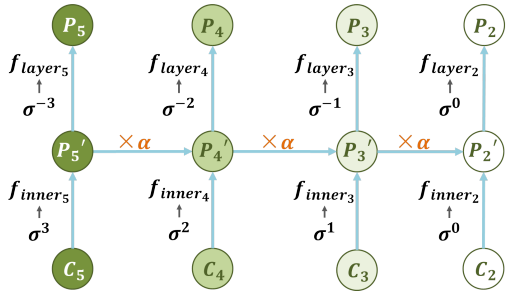

Figure 8. The structure of FPN.

responding coefficients, which indicate different powers of σ ($\alpha = 1$) [2], illustrated in Fig. 8. As shown in Tab. 2, the setting fails to promote performance over baseline. We further conduct an experiment: set α as σ and keep the above initial configuration of f_{inner_i} and f_{layer_i}, the performance is similar to that without defining initialization of f_{inner_i} and f_{layer_i}, shown in Tab. 2, which shows the failure of this strategy.

Secondly, there is the fact that the learning of a neural network is data-driven and a phenomenon that Tiny CityPersons and TinyPerson are sensitive to different α. They have similar data volumes, which is not large enough. Motivated by the antagonistic mechanism, we analyze whether large datasets could propel FPN to implicitly learn fusion factors. Specifically, set different fusion factors and explore when the impact of the fusion factor can be offset. We carry out confirmatory experiments on COCO100. In Fig. 5, the peak phenomenon caused by α is evident. However, COCO is a long-tail dataset (samples of different categories are unbalanced). For example, persons exceed a quarter of COCO, and other categories are relatively much less. Therefore, we further explore the impact of α on different categories which have different data volumes. As shown in the Fig. 7, the peak phenomenon caused by α is greatly weakened when the category is the person. We argue that the network possesses the ability to learn fusion factor latently when the training dataset is large enough. Even in COCO, most of categories are not satisfies the requirement, resulting in that the final performance are sensitive to fusion factor.

[2]We set σ=0.5 in experiments.

3.4. Mathematical explanation for fusion factor

We discuss the mathematical fundamentals of α from the perspective of gradient propagation. Without loss generality, we use α_3^4 and C_4 as an example to analyze how the fusion factor in FPN affects parameter optimization of the backbone. The gradient of C_4 layer can be represented as Eq. 3.

$$\Delta C_4 = \underbrace{-\eta * [\sum_{i=1}^{N_{P_4}} \frac{\partial(loss_{P_4})}{\partial C_4} + \sum_{i=1}^{N_{P_5}} \frac{\partial loss_{P_5}}{\partial C_4}}_{\Delta C_4^{deep}} + \underbrace{\alpha_3^4 * (\sum_{i=1}^{N_{P_2}} \frac{\partial loss_{P_2}}{\partial P_3'} + \sum_{i=1}^{N_{P_3}} \frac{\partial loss_{P_3}}{\partial P_3'}) * \frac{\partial P_3'}{\partial C_4}]}_{\Delta C_4^{shallow}}, \tag{3}$$

where $loss_{P_i}$ denotes the classification and regression loss corresponding to the i^{th} layer.

Eq. 3 means there are two kinds of tasks that affect C_4: object detection in deep layers(P_4, P_5) and object detection in shallow layers(P_2, P_3). While applying bigger α_3^4, C_4 will acquire more information used for detection task of shallow layers, and lost more information used for detection task of deep layers, vice versa. In addition, the deep and shallow is relative. P_4 is deep layer to P_3 and a shallow layer to P_5.

For detection in larger object dataset, such as COCO800, the information of object is rich, and even detection head learns a lot of highly relevant information. If we give up part of information for detection of deep layers(apply small α_3^4), the final performance almost does not decline and if we keep them(apply big α_3^4), the performance will neither not be greatly improved. As a result, the setting of α_3^4 is less sensitive on such dataset.The larger the dataset objects, the less sensitive it is to α setting. In other words, the performance of α_3^4 setting in a larger range is almost the same. The analysis is consistent with the Fig. 5.

For detection in tiny object dataset, the amount of information is not enough, which means that the amount of information that can be learned at each layer is less. Therefore, detection tasks in deep layers and shallow layers both prefer that C_4 can retain more information that is beneficial to themselves, that is, they prefer to obtain the greater ratio of gradient of C_4. Detection task in P_2 and P_3 incline to a greater α_3^4. However, P_4, P_5 incline to a smaller the α_3^4. Finally, the optimal performance lies in a compromise value. The more deviated from this value, the worse the

Detector	MR_{50}^{tiny}	MR_{50}^{tiny1}	MR_{50}^{tiny2}	MR_{50}^{tiny3}	MR_{50}^{small}	MR_{25}^{tiny}	MR_{75}^{tiny}
FCOS [29]	96.28	99.23	96.56	91.67	84.16	90.34	99.56
RetinaNet [16]	92.66	94.52	88.24	86.52	82.84	81.95	99.13
FreeAnchor‡ [35]	89.66	90.79	83.39	82.34	73.88	79.61	98.78
Libra RCNN [23]	89.22	90.93	84.64	81.62	74.86	82.44	98.39
RetinaNet†	88.31	89.65	81.03	81.08	74.05	76.33	98.76
Grid RCNN [20]	87.96	88.31	82.79	79.55	73.16	78.27	98.21
Faster RCNN-FPN [15]	87.57	87.86	82.02	78.78	72.56	76.59	98.39
RetinaNet-SM [33]	88.87	89.83	81.19	80.89	71.82	77.88	98.57
RetinaNet-MSM [33]	88.39	87.8	79.23	79.77	72.18	76.25	98.57
Faster RCNN-FPN-SM [33]	86.22	87.14	79.60	76.14	68.59	74.16	98.28
Faster RCNN-FPN-MSM [33]	85.86	86.54	79.2	76.86	68.76	74.33	98.23
RetinaNet with S-α(ours)	87.73	89.51	81.11	79.49	72.82	74.85	98.57
Faster RCNN-FPN with S-α(ours)	87.29	87.69	81.76	78.57	70.75	76.58	98.42
RetinaNet+SM with S-α	87.00	87.62	79.47	77.39	69.25	74.72	98.41
RetinaNet+MSM with S-α	87.07	88.34	79.76	77.76	70.35	75.38	98.41
Faster RCNN-FPN+SM with S-α	85.96	86.57	79.14	77.22	69.35	73.92	98.30
Faster RCNN-FPN+MSM with S-α	86.18	86.51	79.05	77.08	69.28	73.9	98.24

Table 3. Comparisons of MRs on TinyPerson.

Detector	AP_{50}^{tiny}	AP_{50}^{tiny1}	AP_{50}^{tiny2}	AP_{50}^{tiny3}	AP_{50}^{small}	AP_{25}^{tiny}	AP_{75}^{tiny}
FCOS [29]	17.90	2.88	12.95	31.15	40.54	41.95	1.50
RetinaNet [16]	33.53	12.24	38.79	47.38	48.26	61.51	2.28
FreeAnchor‡ [35]	44.26	25.99	49.37	55.34	60.28	67.06	4.35
Libra RCNN [23]	44.68	27.08	49.27	55.21	62.65	64.77	6.26
RetinaNet†	46.56	27.08	52.63	57.88	59.97	69.6	4.49
Grid RCNN [20]	47.14	30.65	52.21	57.21	62.48	68.89	6.38
Faster RCNN-FPN [15]	47.35	30.25	51.58	58.95	63.18	68.43	5.83
RetinaNet-SM [33]	48.48	29.01	54.28	59.95	63.01	69.41	5.83
RetinaNet-MSM [33]	49.59	31.63	56.01	60.78	63.38	71.24	6.16
Faster RCNN-FPN-SM [33]	51.33	33.91	55.16	62.58	66.96	71.55	6.46
Faster RCNN-FPN-MSM [33]	50.89	33.79	55.55	61.29	65.76	71.28	6.66
RetinaNet with S-α(ours)	48.34	28.61	54.59	59.38	61.73	71.18	5.34
Faster RCNN-FPN with S-α(ours)	48.39	31.68	52.20	60.01	65.15	69.32	5.78
RetinaNet+SM with S-α	52.56	33.90	58.00	63.72	65.69	73.09	6.64
RetinaNet+MSM with S-α	51.60	33.21	56.88	62.86	64.39	72.60	6.43
Faster RCNN-FPN+SM with S-α	51.76	34.58	55.93	62.31	66.81	72.19	6.81
Faster RCNN-FPN+MSM with S-α	51.41	34.64	55.73	61.95	65.97	72.25	6.69

Table 4. Comparisons of APs on TinyPerson.

performance will be. Because if it prefers the deep layers or shallow layers too much, the important information from the other layers will be ignored.

4. Experiment

Experimental setting: The codes are based on Tinybenmark [33]. If there is no special statement, we choose the pre-trained ResNet-50 on ImageNet as the backbone, and RetinaNet is chosen as a detector. There are 12 epochs totally and the initial learning rate is set to 0.005, which then decreased by 10 and 100 times at the 6^{th} epoch and the 10^{th} epoch, respectively. Anchor size is set to (8, 16, 32, 64, 128), and aspect ratio is set to (0.5, 1.0, 2). Due to the dense objects (more than 200) in some images in TinyPerson, we select images with less than 200 objects for training and testing. In terms of data augmentation, only flip horizontal is adopted in our experiments.And we use the original image/sub-image size.

Evaluation standard: According to Tinybenmark [33], we mainly use Average Precision(AP) and Miss Rate(MR) for evaluation. AP is a widely used metric in various object detection tasks, reflecting both the precision and recall ratio of detection results. Due to TinyPerson is a pedestrian dataset, MR is also used as the evaluation standard. And the threshold value of IoU is set to 0.25, 0.5, and 0.75. Tinybenmark [33] further divides tiny[2, 20] into 3 sub-intervals: tiny1[2, 8], tiny2[8, 12], tiny3[12, 20]. In this paper, we pay more attention to whether the object can be found out rather than the location accuracy; therefore, we choose IoU = 0.5 as the main threshold for evaluation.

4.1. Experiment on TinyPerson

The average absolute size of persons in TinyPerson is 18 pixels. And the aspect ratio of persons in TinyPerson varies greatly. Moreover, the diversity of persons is more complicated, making the detection more difficult. TinyPerson contains 794 and 816 images for training and inference, respectively. Most images in TinyPerson are large, resulting in insufficient GPU memory. Therefore, the original images are cut into sub-images with overlapping during the training and inference.

4.1.1 Comparisons with other SOTA detectors

We compare the performance of detectors with the proposed S-α with state of the art methods on TinyPerson [33]. Due to the extremely small(tiny) size, the performance of SOTA detectors significantly decreases, as shown in Tab. 3 and Tab. 4. FreeAnchor‡ and RetinaNet† are improved versions with building FPN using the P_2, P_3, P_4, P_5, P_6 and adjusting anchor size to [8, 16, 32, 64, 128]. The improved versions of RetinaNet is used in subsequent experiments. The imbalance of positive and negative examples is severe on TinyPerson. The performance of two-stage detectors is better than one-stage detectors. Faster RCNN with S-α improves the performance by 1.04% and 0.28% of AP_{50}^{tiny} and MR_{50}^{tiny}, respectively, without adding more parameters of the network. It shows that the modification based on FPN is beneficial to the two-stage detectors. The performance of RetinaNet with S-α is better than other detectors except SM/MSM [33]. SM/MSM needs to perform pretraining on COCO via scale matching between COCO and TinyPerson, and then finetune on TinyPerson, while RetinaNet with S-α is only based on the pre-trained model on ImageNet. RetinaNet with S-α achieves comparable performance without adding a new network parameter. SM/MSM (the SOTA method) and S-α are complementary, shown in Tab. 4 and Tab. 3. RetinaNet+SM with S-α achieves a new SOTA and improves the AP_{50}^{tiny} and MR_{50}^{tiny} over RetinaNet+SM by 4.08% and 1.87%.

4.1.2 Comparisons with Different Backbones

detector	backbone	AP_{50}^{tiny}	MR_{50}^{tiny}
RetinaNet	ResNet-50	46.56	88.31
	ResNet-101	46.99	88.16
RetinaNet with S-α	ResNet-50	48.34	87.73
	ResNet-101	47.99	87.81

Table 5. Object detection results with different backbones on the TinyPerson.

The performance of with RetinaNet S-α, shown in Tab. 5, has been improved 1.78% and 1% of AP_{50}^{tiny} with ResNet-50 and ResNet-101, respectively. Compared with ResNet-50, ResNet-101 has no better performance in tiny person detection, which may be caused by the tiny absolute size. In the case of the fixed size of images, tiny objects are mainly distributed in P_2 and P_3 of FPN, and there are fewer training samples in deep layers. The extra 51 blocks of ResNet-101 compared with ResNet-50 are located in stage-4 of ResNet, which is too deep to help tiny object recognition, but increases the calculation burden.

4.2. Experiment on other tiny datasets

detector	AP_{50}^{tiny}	MR_{50}^{tiny}
RetinaNet	36.36	78.03
RetinaNet with bf-α	38.94	75.91
RetinaNet with S-α	38.60	76.45

Table 6. Object detection results on the Tiny CityPersons.

detector	AP	AP_{all}^{50}
RetinaNet	14.60	27.96
RetinaNet with bf-α	14.68	28.09
RetinaNet with S-α	14.86	28.27

Table 7. Object detection results on the COCO100.

RetinaNet with S-α has also made improvement with Resnet-50 as the backbone on Tiny CityPersons and COCO100, given in Tab. 6 and Tab. 7, the bf represents the optimum via brute force solution.The result show that RetinaNet with S-α is still valid on other tiny datasets as good as the best result of brute force algorithm.

5. Conclusion

In this paper, inspired by the phenomenon that fusion factor affects the performance of tiny object detection, we analyze why the fusion factor affects the performance and explore how to estimate an effective fusion factor to provide best positive influence for tiny object detection. We futher provide the mathematical explanation for the above statement from the perspective of gradient propagation in FPN. We conclude that adjusting the fusion factor of adjacent layers of FPN can adaptively propell shallow layers to focus on learning tiny objects, which leads to improvement for tiny object detection. Moreover, extensive experiments demonstrate the effectiveness of our method by configuring different experimental conditions, including different detectors, different backbones and different datasets. In the future, we will extend our method to other scale dataset and other difficult object detection tasks, such as occluded or truncated.

Acknowledgement: The work was partially supported by the National Science Foundation of China 62006244.

References

[1] Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[2] Jiale Cao, Yanwei Pang, Shengjie Zhao, and Xuelong Li. High-level semantic networks for multi-scale object detection. *TCSVT*, pages 1–1, 2019.

[3] Yukang Chen, Peizhen Zhang, Zeming Li, Yanwei Li, Xiangyu Zhang, Gaofeng Meng, Shiming Xiang, Jian Sun, and Jiaya Jia. Stitcher: Feedback-driven data provider for object detection. *CVPR*, 2020.

[4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005.

[5] Chunfang Deng, Mengmeng Wang, Liang Liu, and Yong Liu. Extended feature pyramid network for small object detection. *CVPR*, 2020.

[6] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *T-PAMI*, 34(4):743–761, 2011.

[7] Markus Enzweiler and Dariu M Gavrila. Monocular pedestrian detection: Survey and experiments. *T-PAMI*, 31(12):2179–2195, 2008.

[8] Andreas Ess, Bastian Leibe, Konrad Schindler, and Luc Van Gool. A mobile vision system for robust multi-person tracking. In *CVPR*, pages 1–8, 2008.

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

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, pages 3354–3361, 2012.

[11] Golnaz Ghiasi, Tsungyi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. *CVPR*, pages 7036–7045, 2019.

[12] Agrim Gupta, Piotr Dollr, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. *CVPR*, 2019.

[13] Tao Kong, Fuchun Sun, Wenbing Huang, and Huaping Liu. Deep feature pyramid reconfiguration for object detection. In *ECCV*, pages 172–188, 2018.

[14] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. *CVPR*, 2019.

[15] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, July 2017.

[16] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017.

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755, 2014.

[18] Songtao Liu, Di Huang, and Yunhong Wang. Learning spatial fusion for single-shot object detection. 2019.

[19] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. *CVPR*, pages 8759–8768, 2018.

[20] Xin Lu, Buyu Li, Yuxin Yue, Quanquan Li, and Junjie Yan. Grid r-cnn. In *CVPR*, 2019.

[21] Jing Nie, Rao Muhammad Anwer, Hisham Cholakkal, Fahad Shahbaz Khan, Yanwei Pang, and Ling Shao. Enriched feature guided refinement network for object detection. *ICCV*, pages 9537–9546, 2019.

[22] Junhyug Noh, Wonho Bae, Wonhee Lee, Jinhwan Seo, and Gunhee Kim. Better to follow, follow to be better: Towards precise supervision of feature super-resolution for small object detection. *ICCV*, pages 9725–9734, 2019.

[23] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra r-cnn: Towards balanced learning for object detection. *CVPR*, 2019.

[24] Jiangmiao Pang, Cong Li, Jianping Shi, Zhihai Xu, and Huajun Feng. R^2-cnn: Fast tiny object detection in large-scale remote sensing images. *TGRS*, 2019.

[25] Bharat Singh and Larry S Davis. An analysis of scale invariance in object detection - snip. *CVPR*, 2018.

[26] Bharat Singh, Mahyar Najibi, and Larry S Davis. Sniper: Efficient multi-scale training. *NeurIPS*, 2018.

[27] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. *CVPR*, 2019.

[28] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. *CVPR*, 2019.

[29] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, 2019.

[30] Xinjiang Wang, Shilong Zhang, Zhuoran Yu, Litong Feng, and Wayne Zhang. Scale-equalizing pyramid convolution for object detection. *CVPR*, 2020.

[31] Christian Wojek, Stefan Walk, and Bernt Schiele. Multi-cue onboard pedestrian detection. In *CVPR*, pages 794–801, 2009.

[32] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *CVPR*, pages 5525–5533, 2016.

[33] Xuehui Yu, Yuqi Gong, Nan Jiang, Qixiang Ye, and Zhenjun Han. Scale match for tiny person detection. In *WACV*, pages 1246–1254, 2020.

[34] Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele. Citypersons: A diverse dataset for pedestrian detection. In *CVPR*, pages 3213–3221, 2017.

[35] Xiaosong Zhang, Fang Wan, Chang Liu, Rongrong Ji, and Qixiang Ye. Freeanchor: Learning to match anchors for visual object detection. *NeurIPS*, 2019.