Rectify the Regression Bias in Long-Tailed Object Detection

Ke Zhu, Minghao Fu, Jie Shao, Tianyu Liu, and Jianxin Wu*

State Key Laboratory for Novel Software Technology
Nanjing University, Nanjing, China
School of Artificial Intelligence, Nanjing University, China
{zhuk, fumh, shaoj, liuty}@lamda.nju.edu.cn, wujx2001@nju.edu.cn

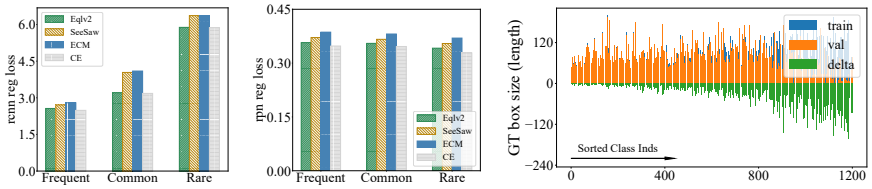
Abstract. Long-tailed object detection faces great challenges because of its extremely imbalanced class distribution. Recent methods mainly focus on the classification bias and its loss function design, while ignoring the subtle influence of the regression branch. This paper shows that the regression bias exists and does adversely and seriously impact the detection accuracy. While existing methods fail to handle the regression bias, the class-specific regression head for rare classes is hypothesized to be the main cause of it in this paper. As a result, three kinds of viable solutions to cater for the rare categories are proposed, including adding a class-agnostic branch, clustering heads and merging heads. The proposed methods brings in consistent and significant improvements over existing long-tailed detection methods, especially in rare and common classes. The proposed method achieves state-of-the-art performance in the large vocabulary LVIS dataset with different backbones and architectures. It generalizes well to more difficult evaluation metrics, relatively balanced datasets, and the mask branch. This is the first attempt to reveal and explore rectifying of the regression bias in long-tailed object detection.

1 Introduction

Long-tailed object detection draws great attention [12] for its practical utility recently. Numerous efforts [1,7,34,35,37] have been made to tackle this challenging task, such as re-weighting [33,35,37], over-sampling [11,12,39], and balanced grouping [19, 24]. These methods are proposed to prevent the tail classes from being overwhelmed due to discouraging gradients [33, 37], inferior predicting scores [11, 24] or insufficient samples [11, 28].

Long-tailed detection often involves both classification and regression heads. While almost all existing methods focus on mitigating the classification bias (*e.g.*, adjusting the classification structure in detection branches), little or *no attention has been paid to the regression branch*. We will show in this paper that the regression bias *has significant adverse effects in long-tailed object detection*, but previous methods failed to identify this important issue.

* J. Wu is the corresponding author



(a) RCNN loss **(b)** RPN loss **(c)** Mean scale in LVIS train and val set

Fig. 1: 1a shows the RCNN regression train loss of frequent, common and rare categories. 1b shows the RPN regression loss. 1c shows distribution of per class mean object scales in LVIS1.0. 'delta' in 1c is the *negative* of difference between train and validation set sizes for different classes.

Fig. 1 clearly showcases this issue. For different methods (EQLv2 [33], SeeSaw [35], ECM [7] and CrossEntropy CE [12]) trained on the LVIS1.0 [12] dataset, we plot the regression branch's losses for the final detection RCNN head in Fig. 1a. It is clear that the regression losses of rare categories are significantly higher than those of frequent and common categories, which inevitably will lead to poor regression results (and hence detection results) for rare classes. We name this finding as the *regression bias*, but existing long-tailed detection methods all fail to deal with or even identify this issue.

To further demonstrate the importance of regression, we calculate the class-wise mean scale of GT (groundtruth) boxes in LVIS train and validation sets, as well as their differences (*cf*. Fig. 1c). The scale shift of rare classes is much larger than that of frequent classes. Since regression is highly correlated with box scale [30], it is thus inherently difficult for a rare class to learn a good bounding box regressor with both few samples and large scale shift. In short, *it is crucial to properly handle the regression bias* in long-tailed object detection.

Our solution to rectify the regression bias is motivated by Fig. 1b. We find that the regression loss in RPN is *balanced* where rare, common and frequent categories have almost the same regression loss, which is almost immune to the regression bias when comparing Fig. 1b with Fig. 1a.

The key difference between RPN and RCNN regression is that the former is *class-agnostic* (i.e., all classes share the same regression parameters), while the latter is *class-specific*. Then, one natural question arises: Can class-agnostic head improve a tail class's generalization ability and handle the regression bias? Our hypothesis is that *rare classes do favor class-agnostic regression heads*.

Our hypothesis is supported by experiments in Table 1, where we compared class-specific and -agnostic regression heads in RCNN. By replacing the classification head with groundtruth class labels, experiments in Table 1 disentangled the impact of the classification head and focus on the regression heads. It is clear that the class-agnostic head possesses substantial advantages: AP_r^b (for rare) surges from 0.7 to 54.6, even surpassing AP_f^b (for frequent classes)! However, the agnostic head will bring a small drop in frequent classes (*e.g.*, from 40.7 to 40.0 in AP_f^b). *The final version of our conjecture* is: the rare (and possibly

common) classes indeed favor class-agnostic regression, while the frequent classes prefers a class-specific regression, and there should be a *trade-off* between the two to optimize for all three types of categories.

Table 1: Results on LVIS1.0 based on the CE baseline. We trained a Mask-RCNN R50-FPN detector [14] with class-specific (the default setting) and class-agnostic regression heads in the RCNN head. Please note that during test, the predicted proposals were provided groundtruth (GT) classification results.

Reg. Head	GT	AP	AP^b	AP^b_r	AP^b_c	AP^b_f
Specific	*no*	18.7	19.7	0.3	16.5	31.7
	yes	**37.2**	**39.8**	**34.1**	**41.3**	**40.7**
Agnostic	*no*	18.0	19.3	0.7	15.5	31.5
	yes	**38.4**	**45.6**	**54.6**	**47.0**	**40.0**

Accordingly, we design three different methods to fully rectify the regression bias, including adding a class-agnostic head, clustering similar heads, or merging heads. All three methods bring in positive effects (*cf*. Table 2), which verifies that rectifying regression bias is indeed crucial. We choose to adopt 'adding a class-agnostic head' in our main experiment for its simplicity, which leads to consistent and significant improvements over previous long-tailed detection pipeline, and has achieved state-of-the-art performance with various backbones and architectures. Moreover, our method shows robust generalization ability (*cf*. Tables 5-8) under varying settings, including different datasets (COCO/COCO-LT [36]), different evaluation metrics and even adapts to the mask branch design. Furthermore, visualizations show that the proposed method leads to more accurate bounding box predictions (*cf*. Fig. 4) and indeed alleviates the regression bias (*cf*. Fig. 5). In summary, our contributions are:

1. For the first time, we reveal and successfully handle the regression bias in long-tailed object detection.
2. We propose three remedies to alleviate this bias, all of which produce consistent improvements over existing methods.
3. Our method achieves state-of-the-art results on LVIS as well as generalizing across datasets, metrics and even the mask branch. Visualizations qualitatively verify our hypothesis, too.

2 Related Work

Vast number of pipelines and techniques have been invented in the field of general purpose object detection. Although most of these detectors can possibly be adapted to balanced detection datasets [25], they can nevertheless hardly handle extremely long-tailed distribution. Dedicated methods are hence required.

Modern long-tailed learning methods aim to solve two important tasks: image classification [31] and object detection/segmentation [12, 30].

The first task, long-tailed image classification [26], is well explored and displays great diversity in terms of specific methods [3,29], training pipelines [20,44], post-hoc analysis [22, 27] and network architecture [16, 23, 38]. Early attempts in this family focused on the details of re-weighting [3] or re-sampling [8] techniques to provide a relatively balanced (data) distribution for tail classes. Later on, the decoupling pipeline [20] for long-tailed learning becomes popular, paving the way for various successors [43, 44]. This decoupled paradigm are based on the insight that instance sampling is beneficial for representation, while the classifier needs to be calibrated to alleviate its bias. Some works try to unify long-tailed learning with theoretical analysis [22, 27], which involves label distribution [29] and generalization bound [27]. Most recent methods are much more diverse in ideas: they involve knowledge distillation [16, 38, 46], self-supervised learning [23, 45] or statistical approach [10] to better handle long-tailed recognition.

The combination of long-tailed learning and object detection or instance segmentation proves to be more challenging [12], because the class distribution in LVIS [12] is extremely imbalanced and naively applying common object detection techniques leads to un-satisfactory results [18, 24, 34, 35, 37]. Dominating solutions in this area are re-sampling [12, 39] and re-weighting [17, 21, 33, 35, 37]. There are variants of them that adopt balanced grouping [24], class incremental learning [19], augmented feature sampling [11, 36] or extra data source [41]. A common characteristic of these long-tailed detection methods is that they only *focus on the classification task, and ignore the subtle influence of the regression branch*. In this paper, we thoroughly explore the previously unvisited regression bias in long-tailed object detection, and propose a simple yet novel method to tackle long-tailed object detection.

In [36], there is an experiment with similar design as ours in Table 1. But, we emphasize that they are fundamentally different. On one hand, conclusions in [36] is that 'performance drop in LVIS is mainly caused by the proposal classification [36]', and hence they focused on *classification*. On the other hand, their class agnostic-head is a supplementary to support their classification hypothesis, while our design and experiments lead to a novel finding: the regression bias.

3 Method

Now we elaborate three different remedies to alleviate the regression bias, and start from the background on long-tailed object detection.

3.1 Preliminaries

We take Faster-RCNN [14] as an example. For a scene image I, it is first fed into a backbone network $\phi(\cdot)$ (*e.g.*, ResNet [15]) to get the image feature $f \in \mathbb{R}^{d\times w\times h}$: $f = \phi(I)$, with dimensionality, width and height denoted as d, w, h, respectively. A region proposal network (RPN) [30] that contains both *agnostic*

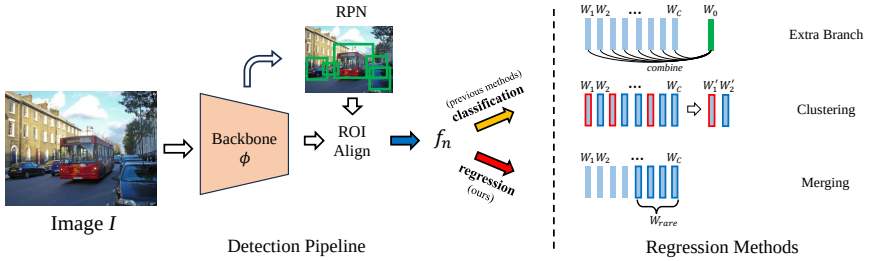


Fig. 2: Illustration of the regular two-stage detection pipeline and the proposed regression methods. Previous methods (the left figure) generally focus on the final classification branch (the yellow arrow), while we focus on rectifying the regression bias (the red arrow). The right part shows our three regression methods, including adding an extra branch W_0, clustering regression heads (*e.g.*, from $W_1, \ldots, W_C$ to W_1', W_2') and merging (*e.g.*, merging rare categories into one regression head W_{rare}, *cf*. Table 2). In our main experiments, we choose 'adding an extra branch' for its simplicity. This figure needs to be viewed in color.

classification and regression branches is then applied on the feature tensor f to produce proposals p from pre-defined anchors. The ROIAlign [14] then extracts proposal features:

$$p = \text{RPN}(f)\,, \quad f_p = \text{ROIAlign}(f, p)\,, \tag{1}$$

where p represents a large set of proposals, and f_p is the set of aligned proposal features. f_p goes into post-process modules (*e.g.*, NMS [30]) before being sent to the RCNN head to get the final features set f_n: $f_n = \text{RCNN}(f_p)$, which is fed to classification and regression branches (*e.g.*, linear layers) to produce the prediction results.

3.2 Our three remedies for the regression bias

In Faster RCNN, there is a dedicated regression head for each class:

$$r_i = W_i^T f_n, \quad i = 1, 2, \ldots, C\,, \tag{2}$$

where $r_i = (\delta x_i, \delta y_i, \delta w_i, \delta h_i)$ represents the regression offset for class i, and W_i is the class-specific regression head (a linear layer). We will now present our approaches to rectify the regression bias. The existing pipeline and our proposed methods are shown in Fig. 2.

Extra class-agnostic branch. This is a simple remedy to cope with the regression bias. Since rare classes favor a class-agnostic head while class-specific ones are slightly preferable to frequent classes in our hypothesis, we propose to use both heads. For class i, its regression head changes to:

$$W_i' = \alpha W_0 + (1 - \alpha) W_i\,, \tag{3}$$

Table 2: Our three different methods to alleviate the regression bias. In 2a, we add an extra class-agnostic regression head during training and testing, where the shared class-agonistic head and the class-specific head are combined using a weight α. In 2b, we perform regression head clustering according to the number of instances ('n') or mean box scale ('s'). In 2c, we directly merge the regression heads of rare, common or frequent classes. The original class-specific heads in 2b and 2c are replaced by the new heads. Note that 'base' means the baseline method, i.e., with only class-specific regression heads. In 2a, $\alpha = 0$ is equivalent to the baseline method, and $\alpha = 1.0$ is equivalent to merging all r, c, f heads into a single head (*i.e.*, the last row in 2c).

(α)	AP	AP_r	AP^b	AP^b_r
0.0	23.7	14.2	24.7	13.4
0.2	24.1	15.8	25.4	15.1
0.5	**25.1**	**17.5**	**27.0**	18.0
0.8	24.4	17.0	25.9	16.4
1.0	24.7	16.7	26.7	**18.3**

(a) class-agnostic head

k	n	s	AP	AP_r	AP^b	AP^b_r
base			23.7	14.2	24.7	13.4
10	✓		24.5	14.5	26.3	14.4
10		✓	24.2	12.8	26.0	12.6
100	✓		24.4	13.0	26.2	12.9
100		✓	**25.2**	**16.7**	**26.9**	**16.7**

(b) Clustering heads

merge	AP	AP_r	AP^b	AP^b_r
base	23.7	14.2	24.7	13.4
r	25.1	16.3	26.7	16.7
c	**25.5**	**17.7**	27.2	17.2
r, c	25.3	17.1	**27.3**	18.2
r, c, f	24.7	16.7	26.7	**18.3**

(c) Merging heads

where W_0 is a shared class-agnostic regression head for *all* classes, and α is a hyper-parameter to balance class-agnostic and class-specific heads. We empirically find that this simple change leads to consistent improvements over the the default class-specific regression head (*cf*. Table 2a), while simply setting $\alpha = 0.5$ gives the optimal trade-off.

Clustering heads. This method is motivated by the analysis in Fig. 1c. Since some categories have similar statistics, we can assign them a *shared* regression head to improve the generalization ability. We implement it by following three steps: sorting, grouping and assigning. First, we use the number of instance or mean box scale to sort the original categories in descending order ($C = 1203$ in LVIS):

$$W_1, \ldots, W_C \overset{sort}{\Longrightarrow} W^s_1, \ldots, W^s_C \,. \tag{4}$$

Then, we cluster them into K groups. During clustering, we do not rely on time-cost algorithms such as K-means [4] or GMM [40]), but simply put adjacent classes into one group, and each group has the same number of classes $N = \frac{C}{K}$:

$$\{(W^s_{N\times i+1}, W^s_{N\times i+2}, \ldots, W^s_{N\times i+N})\}^{K-1}_{i=0} \,, \tag{5}$$

Finally, each group i share one regression head:

$$W_{gi} \overset{replace}{\Longleftarrow} (W^s_{N\times i+1}, W^s_{N\times i+2}, \ldots, W^s_{N\times i+N}) \,. \tag{6}$$

These shared regression matrix are then used for both training and testing. As shown in Table 2b, clustering heads with similar scale statistics brings robust improvements over the baseline methods (see more discussion in Sec. 4.4).

Merge heads. This approach has similar motivation to the previous one, which clusters regression heads, but is even more straightforward. We simply

group the regression heads in pre-defined clusters. For example, we let all rare categories share a common regression head W_{rare}, and the same for common and frequent classes. Experimental results can be found in Table 2c, where we try four different combination (note that r, c means merging rare and common classes all together into *one* regression head). The results indicate that merging always leads to performance gains, especially for the rare categories. The most significant improvements come from merging only the common class (*i.e.*, the row denoted as 'c' in Table 2c, also *cf*. Sec. 4.4 for discussion on partition shift).

All our proposed remedies improve the accuracy of long-tailed object detection and instance segmentation, which verifies the importance of rectifying the regression bias. We will show that all these methods lead to consistent and non-trivial margin over various existing approaches (*cf*. Table 3). We then choose *adding a class-agnostic branch* (the first approach, *cf*. Table 2a) to conduct the rest of our experiments for its simplicity.

Attentive readers might advocate that all proposed three methods are to be verified on all datasets. However, we argue that this is not necessary and our choice has its advantages. On one hand, it does not need any dataset statistics, which is especially appreciated when the number of categories and the data distribution is unknown. On the other hand, by combining both class-agnostic and class-specific heads, it fully exploits object priors and per class knowledge: each type of head has its own merit, as Table 1 shows. Moreover, the validity of all three methods are already verified in Table 2-3.

4 Experiment

We first combine 'Class-Agnostic Branch' (CAB), 'Clustering Head' (CH) and 'Merging Head' (MH) with existing long-tailed methods (*cf*. Table 3), then choose 'SeeSaw [35] + CAB' as 'Our' to compete with state-of-the-art methods (*cf*. Table 4). Although SeeSaw is a lower baseline than ECM [7], it is more stable in reproducing. Finally, we generalize our methods to various evaluation metrics, different datasets and the mask branch.

4.1 Experimental settings

Datasets. We use the large vocabulary dataset LVIS1.0 [12] as our main dataset, which contains 100k training and 20k validation images. Rare (r), common (c) and frequent (f) classes are defined by how many images they occur [12]: [0, 10] for rare, [11, 100] for common, and (100, $+\infty$) for frequent, respectively. We also adopt COCO-LT [25] and COCO2017 [36] to verify the generalization ability of our approach. COCO2017 is a large object detection dataset, containing 118k training and 5k validation images. It is relatively balanced in comparison with LVIS1.0. The COCO-LT dataset is an artificially sampled subset of COCO, with the same validation set but a long-tailed training set. It has about 99k training and 5k validation images. Following previous works [36], we partition COCO-LT

Table 3: Experiments on LVIS1.0. We combine four existing methods with our approach 'adding a class-agnostic branch' (CAB), 'Clustering Head' (CH) or 'Merging Head' (MH). We reproduced RFS [12], EQLv2 [33], SeeSaw [35] and ECM [7] using their official code. For clarity, here we list the object detection metrics AP^b in the first four columns while putting the instance segmentation metrics AP in the latter ones.

Method	+Our	AP^b	AP^b_r	AP^b_c	AP^b_f	AP	AP_r	AP_c	AP_f
RFS [12]	*base*	24.7	13.4	23.1	31.4	23.7	14.2	22.9	29.3
	CAB	27.0	18.0	25.3	**32.9**	25.1	**17.5**	23.9	29.7
	CH	26.9	17.2	25.5	33.0	25.2	16.7	24.3	29.8
	MH	**27.3**	**18.2**	**25.8**	32.8	**25.3**	17.1	**24.5**	**29.8**
EQLv2 [33]	*base*	26.0	16.1	24.0	32.5	25.2	17.4	24.1	29.9
	CAB	**28.1**	**20.4**	**26.3**	**33.5**	**26.0**	**19.5**	**24.9**	**30.2**
	CH	27.3	18.2	25.8	33.1	25.7	17.7	24.8	30.2
	MH	27.1	17.3	25.7	33.2	25.6	17.5	24.7	30.2
SeeSaw [35]	*base*	27.3	18.2	26.5	32.3	26.9	19.6	26.8	30.5
	CAB	28.9	**19.9**	28.3	**33.6**	27.7	20.2	27.3	**31.3**
	CH	**29.1**	19.8	**28.9**	33.4	**27.8**	**20.6**	**27.7**	31.1
	MH	28.5	18.4	28.4	33.7	27.4	19.8	27.4	31.0
ECM [7]	*base*	27.7	17.7	26.6	33.1	27.2	19.6	26.6	31.3
	CAB	**29.1**	18.4	**28.9**	**33.9**	**27.8**	19.1	**28.0**	**31.8**
	CH	28.7	18.0	28.3	33.7	27.8	19.6	27.7	31.6
	MH	28.6	**18.6**	28.2	33.5	27.7	**19.8**	27.7	31.5

into 4 evaluation subsets according to the number of training instances per class, with bins of [1, 20), [20, 400), [400, 8000) and [8000, -), respectively.

Training details. We reproduce four different methods as our baselines, including RFS [12], EQLv2 [33], SeeSaw [35] and ECM [7], following their default experiment settings. We employ MMDetection [5] as our detection framework to conduct our experiment, and train detection models of Faster-RCNN, Mask-RCNN and Cascade R-CNN for 1x or 2x scheduler (except Swin-Transformer based detectors), following previous works [7, 17]. The batch size and learning rate are set as 16 and 0.02, and the data augmentation strictly follows previous long-tailed detection methods [7, 33, 35]. During training, we use FP16 mixed precision training and the warmup strategy to stabilize the learning process. For the evaluation metrics, we adopt AP and AP^b for instance segmentation and object detection, respectively, and adopt AP^b_1, AP^b_2, AP^b_3 and AP^b_4 on COCO-LT, corresponding to its 4 different subsets. With the suggested practice in LVIS official website, we run all our experiment 3 times on 8 RTX3090 GPUs to reduce the variance. Please refer to our supplementary material for more detailed information.

4.2 LVIS detection and segmentation

Consistent improvements. we first evaluate the effectiveness of our method on the LVIS1.0 dataset by combing the proposed approach 'adding a class-agnostic

Table 4: Comparison with state-of-the-art methods on the LVIS1.0 dataset. The results of 'Our' means our proposed CAB regression method combined with the SeeSaw loss ('SeeSaw + CAB', *cf*. Sec. 4). We show the metrics of instance segmentation AP and object detection AP^b. The results with a † are copied from [7], while others are reproduced by us using their official released code.

Architect	Backbone	Method	AP	AP_r	AP_f	AP_c	AP^b
Mask-RCNN [14]	R50-FPN	CE	18.7	0.4	16.5	29.3	19.7
		RFS [13]	23.7	14.2	22.9	29.3	24.7
		BSCE [29]	24.4	15.7	23.6	29.1	25.5
		EQLv2 [33]	25.2	17.4	24.1	29.9	26.0
		ECM [7]†	27.4	19.7	27.0	31.1	27.9
		ECM	27.2	19.6	26.6	31.3	27.7
		SeeSaw [35]	26.9	19.6	26.8	30.5	27.3
		ROG [42]	26.9	20.1	26.8	30.0	27.2
		Our	**27.7**	**20.2**	**27.3**	**31.3**	**28.9**
Mask-RCNN [14]	R101-FPN	CE [7]†	25.5	16.6	24.5	30.6	26.6
		EQLv2 [33]†	27.2	20.6	25.9	31.4	27.9
		ECM [7]†	28.7	**21.9**	27.9	32.3	29.4
		ECM	28.6	20.9	28.3	32.2	29.3
		SeeSaw [35]	28.2	20.3	28.1	31.8	29.0
		Our	**29.0**	21.0	**28.9**	**32.4**	**30.7**
Cascade R-CNN [2]	Swin-T	ECM [7]	34.1	24.0	34.9	38.0	37.6
		SeeSaw [35]	34.2	24.6	34.7	37.8	37.8
		Our	**34.6**	**24.7**	**35.3**	**38.1**	**38.2**
Cascade R-CNN [2]	Swin-B	ECM [7]†	39.7	33.5	40.6	**41.4**	43.6
		Our	**39.9**	**34.6**	**40.7**	41.2	**44.2**

branch' (CAB) with existing long-tailed object detection methods. Since our main focus is on bounding box regression, we list the object detection results in early columns and segmentation results in later ones. As shown in Table 3, using CAB leads to consistent AP^b and AP improvement over existing *classification-based* methods, surpassing all of them with large margins. For object detection, Our CAB benefits the rare class a lot, with an increase of 4.6 AP^b and 4.3 AP^b on RFS and EQLv2. The same is true for instance segmentation, where a growing trend can be observed on all metrics, showing that CAB is also beneficial for later mask pixel predictions. Interestingly, the method 'RFS+CAB' (which uses a CE loss) can almost achieve the same object detection accuracy as the SeeSaw method, and surpasses EQLv2 for about 1 AP^b. We thus conjecture that: besides merely focusing on classification, *our regression methods* can serve as strong alternatives that also *strongly boost* long-tailed detection accuracy.

Comparison with SOTA. We then compare our method with state-of-the-art methods using different object detection framework (Mask-RCNN, Cascade R-CNN) and backbones (ResNet-50, ResNet-101, Swin-T and Swin-B). Note

Table 5: More evaluation metrics in LVIS1.0 dataset with a 1x scheduler using Mask RCNN. $AP^{fixed}_{boundary}$ and AP^{fixed}_{bbox} are two newly proposed challenging metrics [6, 9].

Method	AP	AP^b	$AP^{fixed}_{boundary}$	AP^{fixed}_{bbox}
BAGS [24]	23.1	23.7	-	26.2
EQLv2 [33]	23.9	24.0	20.3	25.9
SeeSaw [35]	25.2	25.4	19.8	26.5
ECM [7]	26.3	26.7	21.4	27.4
Our	**26.9**	**28.0**	**22.1**	**28.2**

that 'Our' means 'SeeSaw + CAB'. For fair comparison, we reproduce majority of existing methods using their official released code unless specialized symbols (*e.g.*, †) appears after a method's name. As can be seen in Table 4, our method achieves the overall highest accuracy in AP and AP^b. For ResNet series models, our regression technique easily surpass the best competitor ECM [7], especially in AP^b (an increase of 1.2 AP^b for ResNet-50 and 1.1 AP^b for ResNet-101). The advantage also holds for ViT-based object detectors, where we surpassed the best competitor ECM [7] with both Swin-Tiny and Swin-Base backbone architectures. Following LVIS's common practice, we didn't list AP^b_r and AP^b_c here, but we want to emphasize that the advantage of our regression methods can be further enlarged when more metrics are listed (*cf*. Table 3). This is also true if we replace 'adding a Class-Agnostic Branch' (CAB) with the 'merging heads' regression alternative (*cf*. the best accuracy in Table 2).

4.3 Generalization ability

In this section, we will show the generalization ability of our regression methods in various aspects, including different evaluation metrics, datasets and its benefits on the mask branch.

Different metrics. We first explore how different metrics affect our model's accuracy. Two additional metrics are $AP_{boundary}$ (a more strict metric in instance segmentation [6]) and AP^{fixed}_{bbox} (constraining 10,000 predicted bounding boxes per class across the dataset [9]). Object detector that obtain decent results in traditional metrics may not perform as well in these criterions. As shown in Table 5, our regression methods adapts well and surpasses all existing methods on both traditional and these challenging metrics.

The COCO-LT dataset. We also transfer our regression to another long-tailed dataset: COCO-LT. This is an artificially sampled [36] subset of original COCO [25]. We calculate the over bounding box metrics AP^b and more fine-grained results: AP^b_1, AP^b_2 AP^b_3 and AP^b_4 (ranging from the rarest to the most frequent class). As shown in Table 6, our CAB leads to consistent improvements on all metrics (especially for the rarest categories AP^b_1) under different repeat-factor-sampling rates, showing the great power of CAB to help the rare

Table 6: Experiments on COCO-LT [36]. Following LVIS [12], we use a similar repeat-factor-sampling (RFS) strategy with different sampling rates (3e-3, 5e-3 and 1e-2).

Rate	+CAB	AP^b_1	AP^b_2	AP^b_3	AP^b_4	AP^b
3e-3		2.3	16.9	26.7	30.4	23.5
	✓	**6.2**	**19.1**	**27.0**	**30.5**	**24.4**
5e-3		3.6	19.3	26.7	30.6	24.3
	✓	**7.5**	**20.0**	**27.3**	**30.7**	**25.0**
1e-2		8.7	20.2	27.8	30.3	25.2
	✓	**10.8**	**21.6**	**28.0**	**30.5**	**25.8**

Table 7: Experiments on the (relatively) balanced COCO dataset. We adopted the Faster RCNN [30] detector and tried three different backbones.

Backbone	+CAB	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
Res-50		37.3	58.3	40.3	21.7	41.0	48.2
	✓	**38.3**	**58.9**	**42.1**	**22.4**	**41.4**	**50.4**
Res-101		39.4	60.3	43.0	22.9	43.4	51.0
	✓	**39.9**	**60.3**	**43.6**	**22.7**	**43.7**	**52.8**
Res-32x4d		41.0	62.2	44.6	**23.9**	45.3	52.9
	✓	**41.7**	**62.5**	**45.4**	23.6	**45.9**	**54.8**

classes. In fact, this conclusion generally holds true for all sampling rates in our experiments. We only listed three here for simplicity and clarity.

On balanced training set. Furthermore, we validate how our method perform on the relatively balanced dataset MS-COCO2017. We adopted Faster RCNN with three different backbones (ResNet-50, ResNet-101 and ResNext101-32x4d). Results in Table 7 clearly shows that CAB generalizes well to datasets with more balanced distributions. Interestingly, CAB boosts metrics of large and medium objects while shows similar accuracy on small objects. This is possibly because that small objects occupy over 60% of the total instances while large objects only consumes roughly 15% [32]. Since CAB is more beneficial to less-frequent classes, it brings higher gains in large objects than in small ones.

The mask branch. Finally, we apply our CAB to the segmentation branch to test whether adding an class-agnostic prior is suitable for mask prediction. Experimental results are in Table 8, where we add a class-agnostic mask prediction head and combines it with each class-specific mask head. As shown in the table, our CAB generalizes well to segmentation tasks. Our main experimental results may be further improved if CAB was applied in both box and mask predictions.

Table 8: Generalization to mask prediction. We added a class-agnostic mask prediction branch and combines it with each class-specific mask prediction results.

Branch	+CAB	AP	AP_r	AP^b	AP^b_r
Box		23.7	14.2	24.7	13.4
	✓	**25.1**	**17.5**	**27.0**	**18.0**
Mask		23.7	14.2	24.7	13.4
	✓	**24.1**	**14.6**	**25.0**	**13.9**

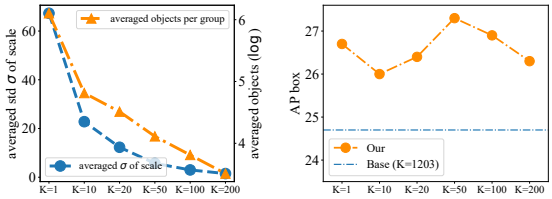
4.4 Discussion of our three remedies

Now we further discuss the proposed three methods (CAB, CH and MH) in detail, hoping to provide a more in-depth understanding of our hypothesis.

Relationship of CAB to the objectness branch. At start, we want to clarify the differences between our CAB and the objectness branch approach [7, 35] adopted in the *classification* head. We have argued that although they seemingly share similar structures with ours, they are essentially *different*. First, the purpose of objectness branch is to deal with the imbalanced distribution between foreground and background *classification* samples [35], while our CAB aim to tackle *the regression bias*. Second, unlike classification where each class must preserve its own classifier, the regression heads can be clustered or merged, which provide more diverse solutions to reduce the regression bias besides CAB (*cf*. Table 2). Lastly, the effect of the objectness branch is yet to be proved because it will *decrease* the performance of rare classes even when combined with CE loss [35], while CAB leads to consistent improvements, especially in rare class.

How many clusters are the best? Attentive readers might have questions why larger clusters K (such as 100) in CH achieve better results in Table 2b for rare class, since enlarging K means the (useful) class-agnostic information is reduced. We thus empirically verify that: besides *objects numbers per group*, the *scale variance* could potentially affect the final AP. We test CH with 'object scales' in more K. For each K, we calculate *standard deviation* of scales in each group, average, and get representative scale std. The same is done for average object. As Fig. 3a shows: as K decreases, training objects per group increases, which benefit rare classes. But, the variation of scales increases a lot, which makes regression head learning more difficult—imagine one head to fit offsets that vary a lot, and we also observed higher training loss for smaller K. Hence, max AP^b_r is *not* at K=1 or 10, but at K=50 in our experiments (*cf*. Fig. 3b).

Why merging 'Common' class is the best?. We observe in Table 2c that merging common classes leads to the best improvement in AP_r, which is somehow counter-intuitive. We conjecture that this is due to the *partition shift* of rare, common and frequent in LVIS train and validation sets. In the LVIS1.0 train dataset, the size ranges for frequent, common and rare are [0, 404], [405, 865] and [866, 1202], respectively. While for the validation set, they become [0, 212], [213, 536] and [537, 1202], respectively (*cf*. Fig. 1c). Hence, when we



(a) statistics of each K **(b)** results of different K

Fig. 3: The statistics (averaged standard deviation, averaged objects per group) of different K and its corresponding AP box.

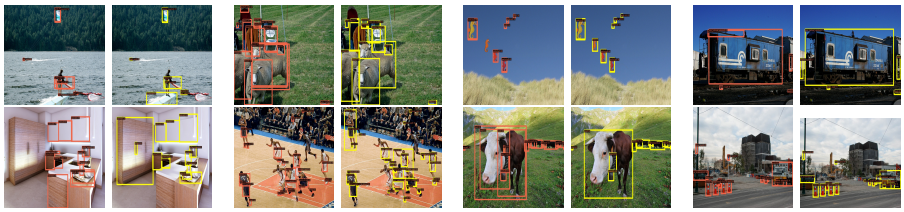


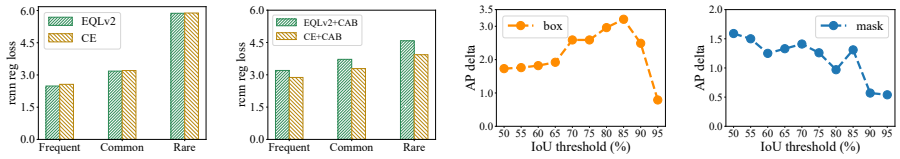
Fig. 4: Visualizations of detection results before (in the left of each group) and after (in the right) using our CAB. We adopted RFS [12] as the baseline in LVIS1.0 and combine it with our CAB regression method. In comparison, the proposed method is good at detecting missing objects, filtering duplicated objects away, as well as rectifying bounding box predictions. This figure needs to be viewed in color.

use training set statistics to merge the common class, a large portion of rare categories in the validation set are also potentially merged, thus contributing to the validation improvement on both AP_r and AP_c.

4.5 Visualization and ablation

Flatten distribution. We first plot the RCNN regression loss of each category before and after combining with our regression CAB. The baseline methods we choose are EQLv2 and CE. As shown in Fig. 5, the regression loss for rare has seen a noticeable drop after adding our CAB, and the overall loss distribution has become much more balanced, too. We thus believe that the pipeline of our regression remedies can relieve the regression bias, further verifying that our hypothesis in Sec. 1 in indeed valid.

More accurate box/mask. We then calculate AP50-AP95 of boxes and masks to find whether our CAB behaves well on stricter IoU thresholds. As show in Fig. 5c-5d, adding CAB achieves consistent accuracy gains over all IoU thresholds for both box and mask prediction, and is especially helpful for those hard IoU threshold (*e.g.*, AP75-90) in box evaluation (*cf*. Fig. 5c). Since a higher threshold requires more precise box prediction, these results have shown that our regression methods is capable of producing precise boxes with better quality.



(a) w/o CAB **(b)** w/ CAB **(c)** box delta **(d)** mask delta

Fig. 5: The loss distribution shift before and after combining with our CAB (Fig. 5a, 5b), and AP improvement of combining our CAB with baseline in LVIS (Fig. 5c, 5d).

Qualitative results. Last but not least, we provide example detection images sampled from the LVIS1.0 validation dataset. For simplicity, we choose RFS [12] with CE loss as baseline and combine it with our CAB method. As shown in Fig. 4, both baseline and our CAB detect most of the objects in an image, but CAB generally caters for more details. For example, CAB can help discover missed boxes, like the cabinet in the lower left images. It is also clearly illustrated that CAB helps filter duplicate boxes away in the final predictions (*e.g.*, the pictures with elephant and cow with grass background). Since CAB brings in better predictions, duplicate boxes will have larger overlap that may be suppressed by NMS (non-maximum suppression). If we zoom this figure (*cf*. the last colum in Fig. 4), we will find that CAB is capable of rectifying the predicted boxes, and this can empirically explain why our CAB achieves much better AP under higher and more difficult IoU thresholds (*cf*. Fig. 5c).

5 Conclusions and Limitations

In this paper, we discovered that the regression bias (imbalanced regression loss distribution on the RCNN head) exists in long-tailed object detection, and adversely affects detection results. We thus proposed three remedies for rectifying the regression bias. The proposed method significantly boosts the performance of rare class AP^b, and achieves state-of-the-art results. We also generalize our regression methods to balanced dataset, different evaluation metrics and the mask branch. Finally, visualizations show that our method indeed produces better predicted bounding boxes.

As for the limitations, it remains unclear why the boosted accuracy become lower when the baseline are higher (*e.g.*, the improvement of CAB on ECM is lower than that on RFS, *cf*. Table 3). This may relate to the upper bound a backbone model can achieve. Since large network generally better fit the dataset (*cf*. Table 4), the performance a ResNet-50 can achieve is limited. We thus call for involving larger vision models on the LVIS dataset to better handle the long-tailed object detection problem. Besides involving stronger vision backbones, we will try to testify the adaptability of the proposed methods when different pre-training information is integrated into the backbone network (e.g. self-supervised learning), which is left as our future work.

Acknowledgements

We thank Yin-Yin He (M.Sc. degree in Nanjing University, now in ByteDance) for providing helpful discussions in long-tailed object detection.

References

1. Alexandridis, K.P., Deng, J., Nguyen, A., Luo, S.: Long-tailed instance segmentation using gumbel optimized loss. In: ECCV. Lecture Notes in Computer Science, vol. 13670, pp. 353–369. Springer (2022)
2. Cai, Z., Vasconcelos, N.: Cascade r-cnn: Delving into high quality object detection. In: CVPR. pp. 6154–6162 (2018)
3. Cao, K., Wei, C., Gaidon, A., Arechiga, N., Ma, T.: Learning imbalanced datasets with label-distribution-aware margin loss. In: NeurIPS. pp. 1565–1576 (2019)
4. Caron, M., Bojanowski, P., Joulin, A., Douze, M.: Deep clustering for unsupervised learning of visual features. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV. Lecture Notes in Computer Science, vol. 11218, pp. 139–156. Springer (2018)
5. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: MMDetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155 (2019)
6. Cheng, B., Girshick, R., Dollar, P., Berg, A.C., Kirillov, A.: Boundary iou: Improving object-centric image segmentation evaluation. In: CVPR. pp. 15334–15342 (2021)
7. Cho, J.H., Krähenbühl, P.: Long-tail detection with effective class-margins. In: ECCV. Lecture Notes in Computer Science, vol. 13668, pp. 698–714. Springer (2022)
8. Cui, Y., Jia, M., Lin, T.Y., Song, Y., Belongie, S.: Class-balanced loss based on effective number of samples. In: CVPR. pp. 9268–9277 (2019)
9. Dave, A., Dollár, P., Ramanan, D., Kirillov, A., Girshick, R.: Evaluating large-vocabulary object detectors: The devil is in the details. arXiv preprint arXiv:2102.01066 (2021)
10. Du, Y., Wu, J.: No one left behind: Improving the worst categories in long-tailed learning. In: CVPR. pp. 15804–15813 (2023)
11. Feng, C., Zhong, Y., Huang, W.: Exploring classification equilibrium in long-tailed object detection. In: ICCV. pp. 3417–3426 (2021)
12. Gupta, A., Dollar, P., Girshick, R.: Lvis: A dataset for large vocabulary instance segmentation. In: CVPR. pp. 5356–5364 (2019)
13. Gupta, A., Dollar, P., Girshick, R.: Lvis: A dataset for large vocabulary instance segmentation. In: CVPR. pp. 5356–5364 (2019)
14. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: ICCV. pp. 2961–2969 (2017)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
16. He, Y.Y., Wu, J., Wei, X.S.: Distilling virtual examples for long-tailed recognition. In: ICCV. pp. 235–244 (2021)

17. He, Y.Y., Zhang, P., Wei, X.S., Zhang, X., Sun, J.: Relieving long-tailed instance segmentation via pairwise class balance. In: CVPR. pp. 7000–7009 (2022)
18. Hsieh, T., Robb, E., Chen, H., Huang, J.: Droploss for long-tail instance segmentation. In: AAAI. pp. 1549–1557 (2021)
19. Hu, X., Jiang, Y., Tang, K., Chen, J., Miao, C., Zhang, H.: Learning to segment the tail. In: CVPR (2020)
20. Kang, B., Xie, S., Rohrbach, M., Yan, Z., Gordo, A., Feng, J., Kalantidis, Y.: Decoupling representation and classifier for long-tailed recognition. In: ICLR (2020)
21. Li, B., Yao, Y., Tan, J., Zhang, G., Yu, F., Lu, J., Luo, Y.: Equalized focal loss for dense long-tailed object detection. In: CVPR. pp. 6990–6999 (2022)
22. Li, M., Cheung, Y.m., Lu, Y.: Long-tailed visual recognition via gaussian clouded logit adjustment. In: CVPR. pp. 6929–6938 (2022)
23. Li, T., Wang, L., Wu, G.: Self supervision to distillation for long-tailed visual recognition. In: ICCV. pp. 630–639 (2021)
24. Li, Y., Wang, T., Kang, B., Tang, S., Wang, C., Li, J., Feng, J.: Overcoming classifier imbalance for long-tail object detection with balanced group softmax. In: CVPR (2020)
25. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV. Lecture Notes in Computer Science, vol. 8693, pp. 740–755. Springer (2014)
26. Liu, Z., Miao, Z., Zhan, X., Wang, J., Gong, B., Yu, S.X.: Large-scale long-tailed recognition in an open world. In: CVPR. pp. 2537–2546 (2019)
27. Menon, A.K., Jayasumana, S., Rawat, A.S., Jain, H., Veit, A., Kumar, S.: Long-tail learning via logit adjustment. In: ICLR (2021)
28. Pang, J., Chen, K., Shi, J., Feng, H., Ouyang, W., Lin, D.: Libra r-cnn: Towards balanced learning for object detection. In: CVPR. pp. 821–830 (2019)
29. Ren, J., Yu, C., sheng, s., Ma, X., Zhao, H., Yi, S., Li, h.: Balanced meta-softmax for long-tailed visual recognition. In: NeurIPS. pp. 4175–4186 (2020)
30. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: NeurIPS. pp. 91–99 (2015)
31. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A., Fei-Fei, L.: ImageNet large scale visual recognition challenge. International Journal of Computer Vision **115**(3), 211–252 (2015)
32. Singh, B., Davis, L.S.: An analysis of scale invariance in object detection SNIP. In: CVPR. pp. 3578–3587 (2018)
33. Tan, J., Lu, X., Zhang, G., Yin, C., Li, Q.: Equalization loss v2: A new gradient balance approach for long-tailed object detection. In: CVPR. pp. 1685–1694 (2021)
34. Tan, J., Wang, C., Li, B., Li, Q., Ouyang, W., Yin, C., Yan, J.: Equalization loss for long-tailed object recognition. In: CVPR (2020)
35. Wang, J., Zhang, W., Zang, Y., Cao, Y., Pang, J., Gong, T., Chen, K., Liu, Z., Loy, C.C., Lin, D.: Seesaw loss for long-tailed instance segmentation. In: CVPR. pp. 9695–9704 (2021)
36. Wang, T., Li, Y., Kang, B., Li, J., Liew, J.H., Tang, S., Hoi, S.C.H., Feng, J.: The devil is in classification: A simple framework for long-tail instance segmentation. In: ECCV. Lecture Notes in Computer Science, vol. 12359, pp. 728–744. Springer (2020)
37. Wang, T., Zhu, Y., Zhao, C., Zeng, W., Wang, J., Tang, M.: Adaptive class suppression loss for long-tail object detection. In: CVPR. pp. 3103–3112 (2021)
38. Wang, X., Lian, L., Miao, Z., Liu, Z., Yu, S.: Long-tailed recognition by routing diverse distribution-aware experts. In: ICLR (2021)

39. Wu, J., Song, L., Wang, T., Zhang, Q., Yuan, J.: Forest R-CNN: large-vocabulary long-tailed object detection and instance segmentation. In: ACM MM. pp. 1570–1578 (2020)
40. Xuan, G., Zhang, W., Chai, P.: EM algorithms of gaussian mixture model and hidden markov model. In: ICIP. pp. 145–148 (2001)
41. Zhang, C., Pan, T.Y., Li, Y., Hu, H., Xuan, D., Changpinyo, S., Gong, B., Chao, W.L.: Mosaicos: A simple and effective use of object-centric images for long-tailed object detection. In: ICCV. pp. 417–427 (2021)
42. Zhang, S., Chen, C., Peng, S.: Reconciling object-level and global-level objectives for long-tail detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 18982–18992 (2023)
43. Zhong, Z., Cui, J., Liu, S., Jia, J.: Improving calibration for long-tailed recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 16489–16498 (2021)
44. Zhou, B., Cui, Q., Wei, X.S., Chen, Z.M.: Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In: CVPR. pp. 9716–9725 (2020)
45. Zhu, K., Fu, M., Wu, J.: Multi-label self-supervised learning with scene images. In: ICCV. pp. 6694–6703 (2023)
46. Zhu, K., He, Y., Wu, J.: Quantized feature distillation for network quantization. In: AAAI. pp. 11452–11460 (2023)