The Devil is in Classification: A Simple Framework for Long-tail Instance Segmentation

Tao Wang[1,4][0000−0002−2480−878X], Yu Li[2,4], Bingyi Kang[4], Junnan Li[3], Junhao Liew[4], Sheng Tang[2], Steven Hoi[3], and Jiashi Feng[4]

[1] NGS, National University of Singapore, Singapore twangnh@gmail.com
[2] Institute of Computing Technology, Chinese Academy of Sciences, China {liyu,ts}@ict.ac.cn
[3] Salesforce Research Asia, Singapore {junnan.li,shoi}@salesforce.com
[4] ECE Department, National University of Singapore, Singapore {kang,liewjunhao}@u.nus.edu elefjia@nus.edu.sg

Abstract. Most existing object instance detection and segmentation models only work well on fairly balanced benchmarks where per-category training sample numbers are comparable, such as COCO. They tend to suffer performance drop on realistic datasets that are usually long-tailed. This work aims to study and address such open challenges. Specifically, we systematically investigate performance drop of the state-of-the-art two-stage instance segmentation model Mask R-CNN on the recent long-tail LVIS dataset, and unveil that a major cause is the inaccurate classification of object proposals. Based on such an observation, we first consider various techniques for improving long-tail classification performance which indeed enhance instance segmentation results. We then propose a simple calibration framework to more effectively alleviate classification head bias with a bi-level class balanced sampling approach. Without bells and whistles, it significantly boosts the performance of instance segmentation for tail classes on the recent LVIS dataset and our sampled COCO-LT dataset. Our analysis provides useful insights for solving long-tail instance detection and segmentation problems, and the straightforward *SimCal* method can serve as a simple but strong baseline. With the method we have won the 2019 LVIS challenge [5]. Codes and models are available at **https://github.com/twangnh/SimCal**.

Keywords: Long-tail Distribution; Instance Segmentation; Object Detection; Long-tail Classification

1 Introduction

Object detection and instance segmentation aim to localize and segment individual object instances from an input image. The widely adopted solutions to such

[5] **Importantly**, after the challenge submission [40], we find significant improvement can be further achieved by modifying the head from 2fc_rand to 3fc_ft (refer to Sec. 5.4 and Table 6 for details), which is expected to generates much higher test set result. We also encourage readers to read our following work [29] that more effectively calibrates the last classification layer with a re-designed softmax module.

tasks are built on region-based two-stage frameworks, *e.g.*, Faster R-CNN [36] and Mask R-CNN [18]. Though these models have demonstrated remarkable performance on several class-balanced benchmarks, such as Pascal VOC [12], COCO [32] and OpenImage [1], they are seldom evaluated on datasets with long-tail distribution that is common in realistic scenarios [35] and dataset creation [12],[27],[32]. Recently, Gupta et al. [16] introduce the LVIS dataset for large vocabulary long-tail instance segmentation model development and evaluation. They observe the long-tail distribution can lead to severe performance drop of the state-of-the-art instance segmentation model [16]. However, the reason for such performance drop is not clear yet.

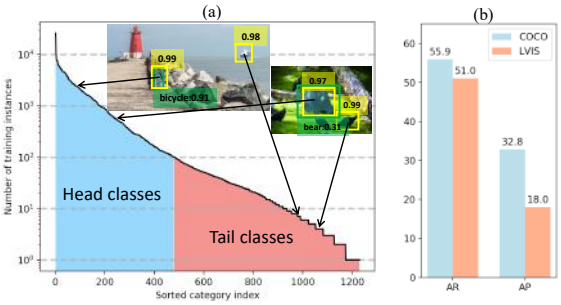
In this work, we carefully study why existing models are challenged by long-tailed distribution and develop solutions accordingly. Through extensive analysis on Mask R-CNN in Sec. 3, we show one major cause of performance drop is the inaccurate classification of object proposals, which is referred to the bias of classification head. Fig. 1 shows a qualitative example. Due to long-tail distribution, under standard training schemes, object instances from the tail classes are exposed much less frequently to the classifier than the ones from head classes[6], leading to poor classification performance on tail classes.

To improve proposal classification, we first consider incorporating several common strategies developed for long-tail classification into current instance segmentation frameworks, including loss re-weighting [19],[38], adaptive loss adjustment (focal loss [30], class-aware margin loss [7]), and data re-sampling [17,37]. We find such strategies indeed improve long-tail instance segmentation performance, but their improvement on tail classes is limited and facing the trade-off problem of largely sacrificing performance on head classes. We thus propose a simple and efficient framework after a thorough analysis of the above strategies. Our method, termed *SimCal*, aims to correct the bias in the classification head with a decoupled learning scheme. Specifically, after normal training of an instance segmentation model, it first collects class balanced proposal samples with a new bi-level sampling scheme that combines image-level and instance-level sampling, and then uses these collected proposals to calibrate the classification head. Thus performance on tail classes can be improved. *SimCal* also incorporates a simple dual head inference component that effectively mitigates performance drop on head classes after calibration.

Based on our preliminary findings, extensive experiments are conducted on LVIS [16] dataset to verify the effectiveness of our methods. We also validate the proposed method with SOTA multi-stage instance segmentation model HTC [9] and our sampled long-tail version of COCO dataset (COCO-LT). From our systematic study, we make the following intriguing observations:

– Classification is the primary obstacle preventing state-of-the-art region-based object instance detection and segmentation models from working well on long-tail data distribution. There is still a large room for improvement along this direction.

[6] we use head classes and many-shot classes interchangeably.



(c)

	Model	AP_r	AP_c	AP_f	AP
bbox	mrcnn-ag*	0.0	10.5	26.5	14.9
	mrcnn-ag	2.8	16.4	27.5	18.6
	props-gt	**48.3**	**44.5**	36.5	42.1
	mrcnn	3.3	19.0	30.0	20.8
	props-gt	**35.4**	**43.0**	40.2	40.8
mask	mrcnn-ag*	0.0	10.5	24.7	14.3
	mrcnn-ag	2.7	16.8	25.6	18.0
	props-gt	**44.2**	**40.2**	29.0	36.6
	mrcnn	3.7	20.0	28.4	20.7
	props-gt	**42.4**	**41.8**	32.6	38.4

Fig. 1: (a) Examples of object proposal and instance segmentation results from ResNet50-FPN Mask R-CNN, trained on long-tail LVIS dataset. The RPN can generate high-quality object proposals (yellow bounding boxes with high confidence scores) even on long-tail distribution, *e.g.*, cargo ship (7 training instances) and vulture (4 training instances). However, they are missed in final detection and segmentation outputs (green bounding boxes and masks) due to poor proposal classification performance. Other proposal candidates and detection results are omitted from images for clarity. (b) Comparison of proposal recall (COCO style Average Recall) and AP between COCO and LVIS dataset with Mask R-CNN model. (c) Pilot experiment results on Mask R-CNN with class-agnostic and class-wise box and mask heads on ResNet50-FPN backbone, evaluated with LVIS v0.5 val set. *mrcnn-ag** denotes standard inference with 0.05 confidence threshold as in optimal settings of COCO, while *mrcnn-ag* means inference with threshold 0.0. Note for all later experiments we use 0.0 threshold. AP^{bb} denotes box AP. *props-gt* means testing with ground truth labels of the proposals

– By simply calibrating the classification head of a trained model with a bi-level class balanced sampling in the decoupled learning scheme, the performance for tail classes can be effectively improved.

2 Related Works

Object Detection and Segmentation Following the success of R-CNN [14], Fast R-CNN [13] and Faster R-CNN [36] architectures, the two-stage pipeline has become prevailing for object detection. Based on Faster R-CNN, He et al. [18] propose Mask R-CNN that extends the framework to instance segmentation with a mask prediction head to predict region based mask segments. Lots of later works try to improve the two-stage framework for object detection and instance segmentation. For example, [21],[22] add IOU prediction branch to improve confidence scoring for object detection and instance segmentation respectively. Feature augmentation and various training techniques are thoroughly examined by [34]. Recently, [6] and [9] further extend proposal based object detection and instance segmentation to multi-stage and achieve state-of-the-art performance.

In this work, we study how to improve proposal-based instance segmentation models over long-tail distribution.

Long-tailed Recognition Recognition on long-tail distribution is an important research topic as imbalanced data form a common obstacle in real-world applications. Two major approaches tackling long-tail problems are sampling [4],[5], [17],[37] and loss re-weighting [11],[19],[20]. Sampling methods over-sample minority classes or under-sample majority classes to achieve data balance to some degree. Loss re-weighting assigns different weights to different classes or training instances adaptively, *e.g.*, by inverse class frequency. Recently, [11] proposes to re-weight loss by the number of inversed effective samples. [7] explores class aware margin for classifier loss calculation. In addition, [24] tries to examine the relation of feature and classifier learning in an imbalanced setting. [41] develops a meta learning framework that transfers knowledge from many-shot to few-shot classes. Existing works mainly focus on classification, while the crucial tasks of long-tail object detection and segmentation on remain largely unexplored.

3 Analysis: Performance Drop on Long-tail Distribution

We investigate the performance decline phenomenon of popular two-stage frameworks for long-tail instance detection and segmentation.

Our analysis is based on experiments on LVIS v0.5 train and validation sets. The LVIS dataset [16] is divided into 3 sets: *rare, common, and frequent*, among which *rare* and *common* contain tail classes and *frequent* includes head classes. We report AP on each set, denoted as AP_r, AP_c, AP_f. For simplicity, we train a baseline Mask R-CNN with ResNet50-FPN backbone and *class agnostic box and mask prediction heads*. As shown in Fig. 1 (c), our baseline model (denoted as mrcnn-ag*) performs poorly, especially on tail categories (*rare* set, AP_r, AP^b_r), with 0 box and mask AP.

Usually, the confidence threshold is set to a small positive value (*e.g.* 0.05 for COCO) to filter out low-quality detections. Since LVIS contains 1,230 categories, the softmax activation gives much lower average confidence scores, thus we minish the threshold here. However, even lowering the threshold to 0 (mrcnn-ag), the performance remains very low for tail classes, and improvement on *rare* is much smaller than that of *common* (6.1 vs 2.7 for segmentation AP, 5.9 vs 2.8 for bbox AP). This reveals the Mask R-CNN model trained with the normal setting is heavily biased to the head classes.

We then calculate proposal recall of *mrcnn-ag* model and compare with the one trained on COCO dataset with the same setting. As shown in Fig. 1 (b), the same baseline model trained on LVIS only has a drop of 8.8% (55.9 to 51.0) in proposal recall compared with that on COCO, but notably, has a 45.1% (32.8 to 18.0) drop in overall mask AP. Since the box and mask heads are class agnostic, we hypothesize that the performance drop is mainly caused by the degradation of proposal classification.

To verify this, for the proposals generated by RPN [36], we assign their ground truth class labels to the second stage as its classification results. Then we evaluate the AP. As shown in Fig. 1 (c), the mask AP for tail classes is increased by a large margin, especially on *rare* and *common* sets. Such findings also hold for the box AP. Surprisingly, with normal class-wise box and mask heads (standard version of Mask R-CNN), performance on tail classes is also boosted significantly. This suggests the box and mask head learning are less sensitive to long-tail training data than classification.

The above observations indicate that the low performance of the model over tail classes is mainly caused by poor proposal classification on them. We refer to this issue as *classification head bias*. Addressing the bias is expected to effectively improve object detection and instance segmentation results.

4 Solutions: Alleviating Classification Bias

Based on the above analysis and findings, we first consider using several existing strategies of long-tail classification, and then present a new calibration framework to correct the classification bias for better detection and segmentation on long-tail distribution.

4.1 Using Existing Long-tail Classification Approaches

We adapt some popular approaches of image classification to solving our long-tail instance detection and segmentation problem, as introduced below. We conduct experiments to see how our adapted methods work in Sec. 5.2. Given a sample x_i, the model outputs logits denoted as y_i, and p_i is probability prediction on the true label z.

Loss Re-weighting [11], [19], [26], [38], [39] This line of works alleviate the bias by applying different weights to different samples or categories, such that tail classes or samples receive higher attention during training, thus improving the classification performance. For LVIS, we consider a simple and effective inverse class frequency re-weighting strategy adopted in [19,41]. Concretely, the training samples of each class are weighted by $w = N/N_j$ where N_j is the training instance number of class j. N is a hyperparameter. To handle noise, the weights are clamped to $[0.1, 10.0]$. The weight for the background is also a hyperparameter. During training, the second stage classification loss is weighted as $L = -w_i \log(p_i)$.

Focal Loss [30] Focal loss can be regarded as loss re-weighting that adaptively assigns a weight to each sample by the prediction. It was originally developed for foreground-background class imbalance for one-stage detectors, and also applicable to alleviating the bias in long-tail problems since head-class samples tend to get smaller losses due to sufficient training, and the influence of tail-class samples would be again enlarged. Here we use the multi-class extension of Focal loss $L = -(1 - p_i)^\gamma \log(p_i)$.

Class-aware Margin Loss [7] This method assigns a class dependent margin to loss calculation. Specifically, a larger margin will be assigned for the tail classes, so they are expected to generalize better with limited training samples. We adopt the margin formulation $\Delta_j = C/N_j^{1/4}$ [7] where N_j is the training instance number N_j for class j as above and plug the margin into cross entropy loss $L = -\log e^{y_{iz}-\Delta_z}/(e^{y_{iz}-\Delta_z} + \sum_{c\neq z} e^{y_{ic}})$.

Repeat Sampling [17], [37] Repeat sampling directly over-samples data (images) with a class-dependent repeating factor, so that the tail classes can be more frequently involved in optimization. Consequently, the training steps for each epoch will be increased due to the over-sampled instances. However, this type of methods are not trivially applicable to detection frameworks since multiple instances from different classes frequently exist in one image. [16] developed a specific sampling strategy for LVIS dataset, calculating a per-image repeat factor based on a per-category repeat threshold and over-sampling each training image according to the repeat factor in each epoch. Note that box and mask learning will also be affected by this method.

We implement the adapted version of [7], [11], [16], [31] for experiments. See Sec. 5.2 for details. From the results, we find the above approaches indeed bring some performance improvements over the baselines, which however are very limited. **Re-weighting methods** tend to complicate the optimization of deep models with extreme data imbalance [11], which is the case for object detection with long-tail distribution, leading to poor performance on head classes. **Focal loss** well addresses the imbalance between foreground and easy background samples, but has difficulty in tackling the imbalance between foreground object classes with more similarity and correlation. For **class-aware margin loss**, the prior margin enforced in loss calculation also complicates the optimization of a deep model, leading to larger drop of performance on head classes. The **repeat sampling** strategy suffers from overfitting since it repeatedly samples from tail classes. Also, it additionally samples more data during training, leading to increased computation cost. In general, the diverse object scale and surrounding context in object instance detection further complicate above-discussed limitations, making these methods hardly suitable for our detection tasks.

4.2 Proposed *SimCal*: Calibrating the Classifier

We find in Sec. 3 that significant performance gain on tail classes can be achieved with merely ground truth proposal labels, and as discussed in Sec. 4.1, exiting classification approaches are not very suitable for tackling our long-tail instance segmentation task. Here, we propose a new *SimCal* framework to calibrate the classification head by retraining it with a new bi-level sampling scheme while keeping the other parts frozen after standard training. This approach is very simple and incurs negligible additional computation cost since only the classification head requires gradient back-propagation. The details are given as follows.

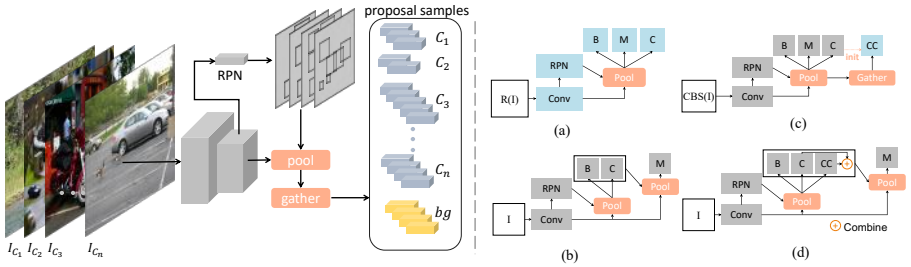


Fig. 2: Left: Illustration of proposed bi-level sampling scheme. Refer to Sec. 4.2 for more details. Right: Architecture of proposed method. I: training or test image sets; R: random sampling; CBS: class-balanced sampling; C: classification head; B: box regression head; M: mask prediction head; CC: calibrated classification head. *Blue* modules are in training mode while *grey* modules indicate frozen. (a) (b) show standard Mask R-CNN training and inference respectively. (c) (d) show proposed calibration and dual head inference respectively

Calibration Training with Bi-level Sampling As shown in Fig. 2, we propose a bi-level sampling scheme to collect training instances for calibrating the classification head through retraining. To create a batch of training data, first, n object classes (*i.e.*, c_1 to c_n) are sampled uniformly form all the classes (which share the same probability). Then, we randomly sample images that contain the categories respectively (*i.e.*, I_{c_1} to I_{c_n}), and feed them to the model. At the object level, we only collect proposals that belong to the sampled classes and background for training. Above, we only sample 1 image for each sampled class for simplicity, but note that the number of sampled images can also be larger. As shown in Fig. 2 right (a), after standard training, we freeze all the model parts (including backbone, RPN, box and mask heads) except for the classification head, and employ the bi-level sampling to retrain the classification head, which is initialized with the original head. Then, the classification head is fed with fairly balanced proposal instances, thus enabling the model to alleviate the bias. Different from conventional fine-tuning conducted on a small scale dataset after pretraining on a large one, our method only changes the data sample distribution. Refer to supplementary material for more implementation details, including foreground and background ratio and matching IOU threshold for proposals. Formally, the classification head is trained with loss:

$$L = \frac{1}{\sum_{i=0}^{N} n_i} \sum_{i=0}^{N} \sum_{j=1}^{n_i} L_{cls}(p_{ij}, p_{ij}^*) \tag{1}$$

where N is the number of sampled classes per batch, n_i is the number of proposal samples for class i, $i = 0$ is for background, L_{cls} is cross entropy loss, and p_{ij} and p_{ij}^* denotes model prediction and ground truth label.

Dual Head Inference After the above calibration, the classification head is now balanced over classes and can perform better on tail classes. However, the performance on head classes drops. To achieve optimal overall performance, here we consider combining the new balanced head and the original one that have higher performance respectively on tail classes and on head classes. We thus propose a dual head inference architecture.

An effective combining scheme is to simply average the models' classification predictions [2], [3], [28], but we find this is not optimal as the original head is heavily biased to many-shot classes. Since the detection models adopt class-wise post-processing (*i.e.*, NMS) and the prediction does not need to be normalized, we propose a new combining scheme that directly selects prediction from the two classifiers for the head and tail classes:

$$p[z] = \begin{cases} p^{cal}[z] & N_z \leq T \\ p^{orig}[z] & \text{otherwise,} \end{cases} \tag{2}$$

where $z \in [0, C]$ indexes the classes, C is the number of classes, $z = 0$ stands for background, p^{cal} and p^{orig} denote the $(C+1)$-dimensional predictions of calibrated and original heads respectively, p is the combined prediction, N_z is the training instance number of class z, and T is the threshold number controlling the boundary of head and tail classes. Other parts of inference remain the same (Fig. 2 (d)). Our dual head inference is with small overhead compared to the original model.

Bi-level Sampling vs. Image Level Repeat Sampling Image level repeat sampling (*e.g.*, [16]), which is traditionally adopted, balances the long-tail distribution at the image level, while our bi-level sampling alleviates the imbalance at the proposal level. Image level sampling approaches train the whole model directly, while we decouple feature and classification head learning, and adjust the classification head only with bi-level class-centric sampling and keep other parts the freezed after training under normal image-centric sampling. We also empirically find the best setting (t=0.001) of IS [16] additionally samples about 23k training images (56k in total) per epoch, leading to more than 40% increase of training time. Comparatively, our method incurs less than 5% additional time and costs much less GPU memory since only a small part of the model needs backpropagation.

5 Experiments

In this section, we first report experiments of using exiting classification approaches to solve our long-tail instance segmentation problem. Then we evaluate our proposed solution, *i.e.* the *SimCal* framework, analyze its model designs and test its generalizality.

Our experiments are mainly conducted on LVIS dataset [16]. Besides, to check the generalizability of our method, we sample a new COCO-LT dataset

from COCO [33]. We devise a complimentary instance-centric category division scheme that helps to more comprehensively analyze model performance. For each experiment, we report result with median overall AP over 3 runs.

5.1 Datasets and Metrics

Datasets 1) LVIS [16]. It is a recent benchmark for large vocabulary long-tail instance segmentation [16]. The source images are from COCO dataset, while the annotation follows an iterative object spotting process that captures the long-tail category statistic naturally appearing in images. Current released version v0.5 contains 1,230 and 830 object classes respectively in its train and validation set, with test set unknown. Refer to Fig. 1 (a) for train set category distribution. The three sets contain about 50k, 5k and 20k images correspondingly. 2) COCO-LT. We sample it from COCO [33] by following an exponential distribution on training instance statistics to create a long-tail version. COCO-LT contains 80 classes and about 100k images. Fig. 3 shows the category distribution. Due to space limitations, we defer details of sampling process to supplement.

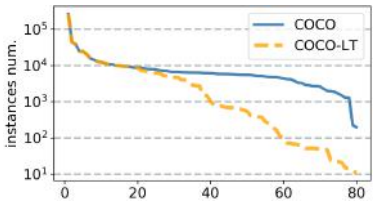


Fig. 3: Category distribution of COCO (2017) and sampled COCO-LT datasets. The categories are sorted in descending numbers of training instances

Table 1: Diffenrent category division scheme, with LVIS v0.5 dataset [16]. The left part is division based on training image number as in [16], the right part is proposed scheme based on training intance number. Train-on-val means categories that appear in the validation set

Set	Total	Divided by #image			Divided by #instance			
		rare	*common*	*frequent*	$(0, 10)$	$[10, 100)$	$[100, 1000)$	$[1000, -]$
Train	1230	454	461	315	294	453	302	181
Train-on-val	830	125	392	313	67	298	284	181

Metrics We adopt AP as overall evaluation metric. Object categories in LVIS are divided into *rare*, *common*, *frequent* sets [16], respectively containing <10, 10-100, and >=100 training images. We show in Table 1 the category distribution of training and validation sets. Besides data splitting based on image number, we devise a complimentary instance-centric category division scheme, considering number of instances is a widely adopted measurement for detection in terms

of benchmark creation, model evaluation [1], [12], [33]. In particular, we divide all the categories into four bins[7] based on the number of training instances, with #instances <10, 10-100, 100-1000, and >=1000, as shown in Table 1. Accordingly, we calculate AP on each bin as complementary metrics, denoted as AP_1, AP_2, AP_3, and AP_4. Such a division scheme offers a finer dissection of model performance. For example, AP_1 corresponds to the commonly referred few-shot object detection regime [8], [23], [25]. *rare* set ($\leq$10 training images) contains categories that have up to 219 training instances ('chickpea'), so AP_r cannot well reflect model's few-shot learning capability. AP_4 reflects performance on classes with COCO level training data, while most classes in *frequent* set (>100 images) have much less than 1,000 training instances (*e.g.*, 'fire-alarm': 117). With the two division schemes, we can report AP on both image-centric (AP_r, AP_c, AP_f) and instance-centric (AP_1, AP_2, AP_3, AP_4) bins for LVIS. For COCO-LT, since the per-category training instance number varies in a much larger range, we divide the categories into four bins with <20, 20-400, 400-8000, and >=8000 training instances and report performance as AP_1, AP_2, AP_3, AP_4 on these bins. Unless specified, AP is evaluated with COCO style by mask AP.

5.2 Evaluating Adapted Existing Classification Methods

We apply adapted discussed methods in Sec. 4.1 to classification head of Mask R-CNN for long-tail instance segmentation, including [7], [11], [16], [31]. Results are summarized in Table 2. We can see some improvements have been achieved on tail classes. For example, 6.0, 6.2, 7.7 absolute margins on AP_1 and 10.1, 8.7, 11.6 on AP_r for loss re-weighting (LR), focal loss (FL) and image level repeat sampling (IS) are observed, respectively. However, on the other hand, they inevitably lead to drop of performance on head classes, *e.g.*, more than 2.0 drop for all methods on AP_4 and AP_f. Performance drop on head classes is also observed in imbalanced classification [17], [38]. Overall AP is improved by at most 2.5 in absolute value (*i.e.*, IS). Similar observation holds for box AP.

5.3 Evaluating Proposed *SimCal*

In this subsection, we report the results of our proposed method applied on mask R-CNN. We evaluate both class-wise and class-agnostic versions of the model. Here T for dual head inference is set to 300.

Calibration Improves Tail Performance From results in Table 3, we observe consistent improvements on tail classes for both class-agnostic and class-wise version of Mask R-CNN (more than 10 absolute mask and box AP improvement on tail bins). Overall mask and box AP are boosted by a large margin. But we also observe a significant drop of performance on head class bins, *e.g.*, 23.7 to 21.9 on AP_3 and 29.6 to 25.3 on AP_4 for the class-wise version of Mask R-CNN. With calibration, the classification head is effectively balanced.

[7] Note we use "bin" and "set" interchangeably.

Table 2: Results on LVIS by adding common strategies in long-tail classification to Mask R-CNN in training. r50 means Mask R-CNN on ResNet50-FPN backbone with class-wise box and mask heads (standard version). CM, LR, FL and IS denote discussed class aware margin loss, loss re-weighting, Focal loss and image level repeat sampling respectively. AP^b denotes box AP. We report result with median overall AP over 3 runs

Model	AP_1	AP_2	AP_3	AP_4	AP_r	AP_c	AP_f	AP	AP^b_1	AP^b_2	AP^b_3	AP^b_4	AP^b_r	AP^b_c	AP^b_f	AP^b
r50	0.0	17.1	23.7	29.6	3.7	20.0	28.4	20.7	0.0	15.9	24.6	30.5	3.3	19.0	30.0	20.8
CM	2.6	21.0	21.8	26.6	8.4	21.2	25.5	21.0	2.8	20.0	22.0	26.6	6.8	20.5	26.4	20.7
LR	6.0	23.3	22.0	25.1	13.8	22.4	24.5	21.9	6.0	21.2	22.3	25.5	11.3	21.5	24.9	21.4
FL	6.2	21.0	22.0	27.0	12.4	20.9	25.9	21.5	5.8	20.5	22.7	28.0	10.5	21.0	27.0	21.7
IS	7.7	25.6	21.8	27.4	15.3	23.7	25.6	23.2	6.7	22.8	22.1	27.4	11.6	22.2	26.7	22.0

Table 3: Results on LVIS by applying *SimCal* to Mask R-CNN with ResNet50-FPN. r50-ag and r50 denote models with class-agnostic and class-wise heads (box/mask) respectively. cal and dual means calibration and dual head inference. Refer to supplementary file for an anlaysis on LVIS result mean and std

	Model	cal	dual	AP_1	AP_2	AP_3	AP_4	AP_r	AP_c	AP_f	AP
bbox	r50-ag			0.0	12.8	22.8	28.3	2.8	16.4	27.5	18.6
		✓		12.4	23.2	20.6	23.5	17.7	21.2	23.4	21.5
		✓	✓	**12.4**	**23.4**	21.4	27.3	**17.7**	**21.3**	26.4	**22.7**
	r50			0.0	15.9	24.6	30.5	3.3	19.0	30.0	20.8
		✓		8.1	21.0	22.4	25.5	13.4	20.6	25.7	21.4
		✓	✓	**8.2**	**21.3**	23.0	29.5	**13.7**	**20.6**	28.7	**22.6**
mask	r50-ag			0.0	13.3	21.4	27.0	2.7	16.8	25.6	18.0
		✓		13.2	23.1	20.0	23.0	18.2	21.4	22.2	21.2
		✓	✓	**13.3**	**23.2**	20.7	26.2	**18.2**	**21.5**	24.7	**22.2**
	r50			0.0	17.1	23.7	29.6	3.7	20.0	28.4	20.7
		✓		10.2	23.5	21.9	25.3	15.8	22.4	24.6	22.3
		✓	✓	**10.2**	**23.9**	22.5	28.7	**16.4**	**22.5**	27.2	**23.4**

Table 4: Results for augmenting discussed long-tail classification methods with proposed decoupled learning and dual head inference.

Model	AP_1	AP_2	AP_3	AP_4	AP_r	AP_c	AP_f	AP
r50	0.0	17.1	23.7	29.6	3.7	20.0	28.4	20.7
CM	4.6	21.0	22.3	28.4	10.0	21.1	27.0	21.6
LR	6.9	23.0	22.1	28.8	13.4	21.7	26.9	22.5
FL	7.1	21.0	22.1	28.4	13.1	21.5	26.5	22.2
IS	6.8	23.2	22.5	28.0	14.0	22.0	27.0	22.7
ours	**10.2**	**23.9**	22.5	28.7	**16.4**	**22.5**	27.2	**23.4**

Table 5: Comparison between proposed combining scheme (sel) and averaging (avg).

	AP_1	AP_2	AP_3	AP_4	AP
orig	0.0	13.3	21.4	27.0	18.0
cal	8.5	20.8	17.6	19.3	18.4
avg	8.5	20.9	19.6	24.6	20.3
sel	**8.6**	**22.0**	19.6	26.6	**21.1**

Dual Head Inference Mitigates Performance Drop on Head Classes The model has a minor performance drop on the head class bins but an enormous boost on the tail class bins. For instance, we observe 0.8 drop of AP_4 but 13.3 increase on AP_1 for r50-ag model. It can be seen that with the proposed combination method, the detection model can effectively gain the advantage of both calibrated and original classification heads.

Class-wise Prediction is Better for Head Classes While Class-agnostic One is Better for Tail Classes We observe AP_1 of r50-ag with cal and dual is 3.1 higher (13.3 vs 10.2) than that of r50 while AP_4 is 2.5 lower (26.2 vs 28.7), which means class-agnostic heads (box/mask) have an advantage on tail classes, while class-wise heads perform better for many-shot classes. This phenomenon suggests that a further improvement can be achieved by using class-agnostic head for tail classes so they can benefit from other categories for box and mask prediction, and class-wise head for many-shot classes as they have abundant training data to learn class-wise prediction, which is left for future work.

Comparing with Adapted Existing Methods For fair comparison, we also consider augmenting the discussed imbalance classification approaches with proposed decoupled learning framework. With the same baseline Mask R-CNN trained in the normal setting, we freeze other parts except for classification head, and use these methods to calibrate the head. After that, we apply the dual head inference for evaluation. As shown in Table 4, they have similar performance on head classes as dual head inference is used. They nearly all get improved on tail classes than the results in Table 2 (*e.g.*, 4.6 vs 2.6, 6.9 vs 6.0, and 7.1 vs 6.2 on AP_1 for CM, LR, and FL methods respectively), indicating the effectiveness of the decoupled learning scheme for recognition of tail classes. The image level repeat sampling (IS) gets worse performance than that in Table 2, suggesting box and mask learning also benefits a lot from the sampling. Our method achieves higher performance, *i.e.*, 10.2 and 23.9 for AP_1 and AP_2, which validates effectiveness of the proposed bi-level sampling scheme.

5.4 Model Design Analysis of *SimCal*

Calibration Dynamics As shown in Fig. 4 (a), with the progress of calibration, model performance is progressively balanced over all the class bins. Increase of AP on tail bins (*i.e.*, AP_1, AP_2) and decrease of AP on the head (*i.e.*, AP_3, AP_4) are observed. With about 10-20k steps, AP on all the bins and overall AP converge to a steady value.

Design Choice of Calibration Head While the proposed calibration method tries to calibrate the original head, we can perform the calibration training on other head choices. As shown in Fig. 4 (b), we have tried different instantiations instead of the original head. It is interesting that with random initialization,

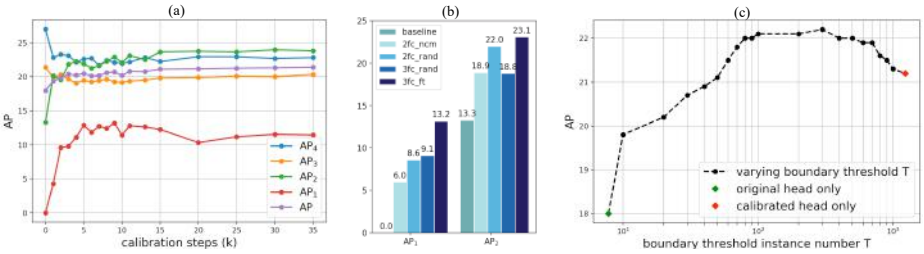


Fig. 4: (a) Model performance as a function of calibration steps. The result is obtained with r50-ag model (Table 3). (b) Effect of design choice of calibration head. Baseline: original model result; 2fc_ncm [15]: we have tried to adopt the deep nearest class mean classifier learned with 2fc representation. 2fc_rand: 2-layer fully connected head with 1024 hidden units, random initialized; 3fc-rand: 3-layer fully connected head with 1024 hidden units, random initialized. 3fc-ft: 3fc initialized from original head. (c) Effect of boundary number T (with r50-ag)

3-layer fully connected head performs worse than 2-layer head on AP_1 (*i.e.*, 2fc_rand vs 3fc-rand). But when it is initialized from the original 3-layer head, the performance is significantly boosted by 4.1 and 4.3 AP respectively on AP_1 and AP_2 (*i.e.*, 3fc_ft). This phenomenon indicates that training under random sampling can help the classification head learn general features and perform well when calibrating with balanced sampling. We only compare them on the tail class bins since they perform on par on head class bins with dual head inference.

Combining Scheme and Head/Tail Boundary for Dual Heads As shown in Table 8. Our combination approach achieves much higher performance than simple averaging. Refer to supplementary material for more alternative combining choices. We also examine the effect of head/tail boundary as in Fig. 4 (c). For the same model, we vary the boundary threshold instance number T from 10 to 1000. The AP is very close to optimal ($T = 300$) in $T \in [90, 500]$. Thus dual head is insensitive to the exact value of hyperparameter T in a wide range.

5.5 Generalizability Test of *SimCal*

Performance on SOTA Models We further apply the proposed method to state-of-the-art multi-stage cascaded instance segmentation model, Hybrid Task Cascade [9] (HTC), by calibrating classification heads at all the stages. As shown in Table 6, our method brings significant gain on tail classes and minor drop on many-shot classes. Notably, the proposed approach leads to much higher gain than the image level repeat sampling method (IS), (*i.e.*, 8.5 and 2.5 higher on AP_1 and AP_2 respectively). We achieve state-of-the-art single model performance on LVIS, which is 6.3 higher in absolute value than the best single model reported in [16] (33.4 vs 27.1). And with test set, a consistent gain is observed.

Table 6: Results with Hybrid Task Cascade (HTC) on LVIS. With backbone of ResNeXt101-64x4d-FPN. best denotes best single model performance reported in [16]. The remaining rows are our experiment results with HTC. 2fc_rand and 3fc_ft are different design choices of classification head (Sec. 5.4). Only 2fc_rand is available on test set as the evaluation server is closed

Model	Val								Test			
	AP_1	AP_2	AP_3	AP_4	AP_r	AP_c	AP_f	AP	AP_r	AP_c	AP_f	AP
best [16]	–	–	–	–	15.6	27.5	31.4	27.1	9.8	21.1	30.0	20.5
htc-x101	5.6	33.0	33.7	37.0	13.7	34.0	36.6	31.9	5.9	25.7	35.3	22.9
IS	10.2	32.3	33.2	36.6	17.6	33.0	36.1	31.9	–	–	–	–
2fc_rand	12.9	32.2	33.5	37.1	18.5	33.3	36.1	32.1	10.3	25.3	35.1	23.9
3fc_ft	**18.8**	**34.9**	33.0	36.7	**24.7**	33.7	36.4	**33.4**	–	–	–	–

Table 7: Results on COCO-LT, evaluated on minival set. AP_1, AP_2, AP_3, AP_4 correspond to bins of [1, 20), [20, 400), [400, 8000), [8000, -) training instances

Model	cal	dual	AP_1	AP_2	AP_3	AP_4	AP	AP^b_1	AP^b_2	AP^b_3	AP^b_4	AP^b
r50-ag			0.0	8.2	24.4	26.0	18.7	0.0	9.5	27.5	30.3	21.4
r50-ag	✓		15.0	16.2	22.4	24.1	20.6	14.5	17.9	24.8	27.6	22.9
r50-ag	✓	✓	**15.0**	**16.2**	24.3	26.0	**21.8**	**14.5**	**18.0**	27.3	30.3	**24.6**

Performance on COCO-LT As shown in Table 7, similar trend of performance boost as LVIS dataset is observed. On COCO-LT dual head inference can enjoy nearly full advantages of both the calibrated classifier on tail classes and the original one on many shot classes. But larger drop of performance on many-shot classes with LVIS is observed. It may be caused by the much stronger inter-class competition as LVIS has much larger vocabulary.

6 Conclusions

In this work, we carefully investigate two-stage instance segmentation model's performance drop with long-tail distribution data and reveal that the devil is in proposal classification. Based on this finding, we first try to adopt several common strategies in long-tail classification to improve the baseline model. We also propose a simple calibration approach, *SimCal*, for improving the second-stage classifier on tail classes. It is demonstrated that *SimCal* significantly enhances Mask R-CNN and SOTA multi-stage model HTC. A large room of improvement still exists along this direction. We hope our pilot experiments and in-depth analysis together with the simple method would benefit future research.

Acknowledgement Jiashi Feng was partially supported by MOE Tier 2 MOE 2017-T2-2-151, NUS_ECRA_FY17_P08, AISG-100E-2019-035.

G Implementation Details

We use PyTorch to implement the proposed method. Our implementation is based on the mmdetection [10]. All the models are trained with SGD and momentum of 0.9, on 8 NVIDIA Tesla V100 GPUs.

G.1 Standard Model Training

For standard model training, i.e., normal training of whole model with random sampling as shown in Fig.2 (a), 8 images per mini-batch are sampled.

For the Mask-RCNN standard model training, the learning rate starts with 0.01 and decays at the 8th and 11th epochs by a factor of 0.1. The training ends at the 12th epoch. The short edge of images is fixed at 800 pixels and the long edge is capped at 1,333 pixels, without changing the aspect ratio.

For the HTC standard model training, the learning rate starts with 0.01 and decays at the 16th and 19th epochs by a factor of 0.1. The training ends at the 20th epoch. We also add multi-scale augmentation for HTC model training. More specifically, the size of image short edge is randomly sampled from [400, 1,400], and the long edge is capped at 1,600 pixels, without changing the aspect ratio.

G.2 Calibration Training

For calibration training, we sample 16 classes and 1 image per class in one mini-batch. proposals are matched to ground truth bounding boxes with threshold of 0.5, as in [36]. We sample the same number of background ROIs as the the foreground ROIs (i.e., # background : # foreground = 1:1). For both Mask-RCNN and HTC calibration, the learning rate starts with 0.01 and decays at the 8000th (i.e., 0.001) and 11000th (i.e., 0.0001) steps by a factor of 0.1. The calibration training ends at 12000 steps.

G.3 Other Hyperparameters

All the hyperparameters for the adopted long-tail classification methods are tuned by grid search. (i) For re-weighting method, N (i.e., the numerator of class dependent loss weight, as in Sec. 4.1) and background loss weight are set as 100 and 1 respectively. (ii) For Focal loss, γ and α are set as 3 and 0.5 respectively. Different from the observation made in [31] that one-stage detector's performance is relatively robust to the value of γ in a wide range, we find the performance of Mask R-CNN model on LVIS is sensitive to the value of γ and $\gamma = 3$ gives the best performance. The hyperparameter C for class-aware margin loss is set as 6.0.

H Qualitative Results

Some qualitative results can be found in Fig. 5.

Fig. 5: Qualitative results for low-shot categories (in [1, 10) bin) on LVIS with r50-ag model. Only relevant detections of low-shot classes are visualized for a better view. Although there are some false positives, the model after calibration can detect and segment those object instances. For the original model without calibration, all those objects are missed. Note some detections have 0.00 score as we only round the score to 2 decimal places

I COCO-LT Sampling

Here we explain how we created the COCO-LT dataset. We evenly divide the 80 categories into 4 subsets according to their category index, i.e. (1-20, 21-40, 41-60, 61-80) respectively, each with 20 classes. For the ith subset if $i \neq 1$, we randomly sample n_i instances with $n_i \in (8 * 10^{4-i}, 8 * 10^{5-i})$. For the first subset (with category indices of 1-20, i.e., i=1) we do not perform sampling. If an instance is not sampled, we remove it from the annotation of its image. For training images without sampled instances, we remove these images. The category distribution after sampling (COCO-LT as shown as in Fig. 3) follows a long-tail distribution. The validation set is kept as the original and used for evaluation.

J How to Combine the Dual Heads

In addition to the proposed simple threshold selection scheme for combining the calibrated and original heads' prediction, we also explored some other possible schemes. As shown in Table 8, we compare the proposed combination scheme with other alternatives, including (i) *cal-only*: using only the prediction from the calibrated head; (ii) *avg*: averaging predictions of the original head and calibrated head, this is widely adopted way of ensembling two classification models; (iii) *det*: using the two heads separately for detection outputs and combining them afterward (i.e., with NMS), this is most simple but effective way of ensembling detection models; (iv) *sel*: the proposed output combining scheme; (v) *sel-thr*: filtering the calibrated head predictions with 0.05 threshold before *sel*, aiming to reduce low quality detections from calibrated head with low confidence score; (vi) *sel-scale*: scaling calibrated head's predictions by ratio of average background score between calibrated and original head's predictions before *sel*, since the

calibrated head is trained with different background and foreground sample ratio, it has different average foreground score compared to original head, *sel-scale* aims to scale alleviate the difference; (vii) *sel-norm*: normalizing the prediction by the summed score over classes after *sel*, aiming to convert the prediction to a normalized classification score.

The proposed combining scheme achieves the best overall result (i.e., 21.1 AP) compared with the other alternatives. Those prior strategies of *sel-thr*, *sel-scale* and *sel-norm* all have similar but slightly lower performance. The result verifies the simplicity and effectiveness of proposed combining method.

Table 8: Ablation result for different ways of combining calibrated and original heads' predictions. The model is Mask R-CNN with ResNet50-FPN backbone and class agnostic box and mask heads. The experiment is with 2 layer fully connected head with random initialization (i.e., 2fc)

Model	AP_1	AP_2	AP_3	AP_4	AP
orig	0.0	13.3	**21.4**	**27.0**	18.0
cal-only	8.5	20.8	17.6	19.3	18.4
avg	8.5	20.9	19.6	24.6	20.3
det	8.6	22.0	16.7	25.2	19.8
sel	**8.6**	**22.0**	19.6	26.6	**21.1**
sel-thr	8.5	20.8	20.1	26.7	20.9
sel-scale	8.5	21.3	19.9	26.7	21.0
sel-norm	8.5	21.9	19.5	26.2	20.9

K How Calibration Learning Rate Affects Performance

While we use the same initial learning rate for calibration as standard model training (i.e., starting from 0.01), we examine the results when varying the initial learning rate for calibration, to see if optimal learning rate for calibration is different from standard model training. We measure model performance with overall AP. As shown in Table 9, we tested the following learning rates of 0.001, 0.002, 0.004, 0.008, 0.01, 0.02, 0.04 and 0.08. The decaying step and factor remains the same. The best performance is achieved at learning rate of 0.01, same as standard model training. The phenomenon also stands in contrast to the observation in conventional fine-tuning that a much lower learning rate compared to pre-training is required for optimal performance.

L Layers to Perform Calibration

While we calibrate the whole 3-layer fully connected classification head, we tried to only perform the calibration on last layer, and last 2 layers, to see if we can

Table 9: Ablation study for calibration learning rate. The model is Mask R-CNN with ResNet50-FPN backbone and class agnostic box and mask heads

lr	0.001	0.002	0.004	0.008	0.01	0.02	0.04	0.08	baseline
AP	19.5	21.8	21.9	22.0	**22.2**	21.5	21.1	21.0	18.0

achieve better performance. We measure model performance with overall AP. Table 10 shows the result comparison of those settings. Best result is obtained when we calibrate the whole classification head.

Table 10: Ablation study for calibration layers. lastfc means only calibrate the last fc layer of classification layerl; last2fc indicates calibrating the last 2fc layers; all 3fc means normal setting of calibrating the full 3 layer classification head. The model is Mask R-CNN with ResNet50-FPN backbone and class agnostic box and mask heads

layer	$lastfc$	$last2fc$	$all3fc$	baseline
AP	21.9	21.8	**22.2**	18.0

M LVIS Mean and Std Analysis

As shown in Tab 11, we report mean and std analysis for the major two *SimCal* models on LVIS dataset. As shown from the results, the variance of AP result is much larger for the tail classes while smaller for many-shot classes as they have ample training instances.

Table 11: Mean and standard deviation analysis for *SimCal* models on LVIS dataset. The result is obtained by repeat each experiments 5 times and calculating the mean and standard deviation of each metric

Model	AP_1	AP_2	AP_3	AP_4	AP_r	AP_c	AP_f	AP
r50-ag-lvis	$13.0_{\pm 1.5}$	$23.2_{\pm 0.8}$	$20.7_{\pm 0.3}$	$26.2_{\pm 0.2}$	$18.0_{\pm 0.9}$	$21.3_{\pm 0.3}$	$24.8_{\pm 0.1}$	$22.4_{\pm 0.3}$
r50-lvis	$10.2_{\pm 1.3}$	$23.9_{\pm 0.6}$	$22.5_{\pm 0.4}$	$28.7_{\pm 0.1}$	$16.4_{\pm 0.7}$	$22.5_{\pm 0.4}$	$27.2_{\pm 0.2}$	$23.4_{\pm 0.2}$

References

1. Openimage dataset. **https://storage.googleapis.com/openimages/web/factsfigures.html**
2. Alpaydin, E.: Multiple networks for function learning. In: IEEE International Conference on Neural Networks. pp. 9–14. IEEE (1993)
3. Breiman, L.: Bagging predictors. Machine learning **24**(2), 123–140 (1996)
4. Buda, M., Maki, A., Mazurowski, M.A.: A systematic study of the class imbalance problem in convolutional neural networks. Neural Networks **106**, 249–259 (2018)
5. Byrd, J., Lipton, Z.: What is the effect of importance weighting in deep learning? In: International Conference on Machine Learning. pp. 872–881 (2019)
6. Cai, Z., Vasconcelos, N.: Cascade r-cnn: Delving into high quality object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6154–6162 (2018)
7. Cao, K., Wei, C., Gaidon, A., Arechiga, N., Ma, T.: Learning imbalanced datasets with label-distribution-aware margin loss. arXiv preprint arXiv:1906.07413 (2019)
8. Chen, H., Wang, Y., Wang, G., Qiao, Y.: Lstd: A low-shot transfer detector for object detection. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
9. Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., et al.: Hybrid task cascade for instance segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4974–4983 (2019)
10. 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)
11. Cui, Y., Jia, M., Lin, T.Y., Song, Y., Belongie, S.: Class-balanced loss based on effective number of samples. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9268–9277 (2019)
12. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International journal of computer vision **88**(2), 303–338 (2010)
13. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 1440–1448 (2015)
14. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 580–587 (2014)
15. Guerriero, S., Caputo, B., Mensink, T.: Deepncm: Deep nearest class mean classifiers (2018)
16. Gupta, A., Dollar, P., Girshick, R.: LVIS: A dataset for large vocabulary instance segmentation. In: CVPR (2019)
17. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Transactions on knowledge and data engineering **21**(9), 1263–1284 (2009)
18. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
19. Huang, C., Li, Y., Change Loy, C., Tang, X.: Learning deep representation for imbalanced classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5375–5384 (2016)

20. Huang, C., Li, Y., Chen, C.L., Tang, X.: Deep imbalanced learning for face recognition and attribute prediction. IEEE transactions on pattern analysis and machine intelligence (2019)
21. Huang, Z., Huang, L., Gong, Y., Huang, C., Wang, X.: Mask scoring r-cnn. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6409–6418 (2019)
22. Jiang, B., Luo, R., Mao, J., Xiao, T., Jiang, Y.: Acquisition of localization confidence for accurate object detection. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 784–799 (2018)
23. Kang, B., Liu, Z., Wang, X., Yu, F., Feng, J., Darrell, T.: Few-shot object detection via feature reweighting. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 8420–8429 (2019)
24. Kang, B., Xie, S., Rohrbach, M., Yan, Z., Gordo, A., Feng, J., Kalantidis, Y.: Decoupling representation and classifier for long-tailed recognition. arXiv preprint arXiv:1910.09217 (2019)
25. Karlinsky, L., Shtok, J., Harary, S., Schwartz, E., Aides, A., Feris, R., Giryes, R., Bronstein, A.M.: Repmet: Representative-based metric learning for classification and few-shot object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5197–5206 (2019)
26. Khan, S.H., Hayat, M., Bennamoun, M., Sohel, F.A., Togneri, R.: Cost-sensitive learning of deep feature representations from imbalanced data. IEEE transactions on neural networks and learning systems **29**(8), 3573–3587 (2017)
27. Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.J., Shamma, D.A., et al.: Visual genome: Connecting language and vision using crowdsourced dense image annotations. International Journal of Computer Vision **123**(1), 32–73 (2017)
28. Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation, and active learning. In: Advances in neural information processing systems. pp. 231–238 (1995)
29. 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: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10991–11000 (2020)
30. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017)
31. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. IEEE transactions on pattern analysis and machine intelligence (2018)
32. 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: European conference on computer vision. pp. 740–755. Springer (2014)
33. 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 (2014)
34. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8759–8768 (2018)
35. Reed, W.J.: The pareto, zipf and other power laws. Economics letters **74**(1), 15–19 (2001)
36. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)

37. Shen, L., Lin, Z., Huang, Q.: Relay backpropagation for effective learning of deep convolutional neural networks. In: European conference on computer vision. pp. 467–482. Springer (2016)
38. Tang, Y., Zhang, Y.Q., Chawla, N.V., Krasser, S.: Svms modeling for highly imbalanced classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **39**(1), 281–288 (2008)
39. Ting, K.M.: A comparative study of cost-sensitive boosting algorithms. In: In Proceedings of the 17th International Conference on Machine Learning. Citeseer (2000)
40. Wang, T., Li, Y., Kang, B., Li, J., Liew, J.H., Tang, S., Hoi, S., Feng, J.: Classification calibration for long-tail instance segmentation. arXiv preprint arXiv:1910.13081 (2019)
41. Wang, Y.X., Ramanan, D., Hebert, M.: Learning to model the tail. In: Advances in Neural Information Processing Systems. pp. 7029–7039 (2017)