Overcoming Classifier Imbalance for Long-tail Object Detection with Balanced Group Softmax – Supplementary Material

Yu Li[1,2,3], Tao Wang[3,4], Bingyi Kang[3], †Sheng Tang[1,2], Chunfeng Wang[2], Jintao Li[1,2], Jiashi Feng[3]

[1]Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China

[2]University of Chinese Academy of Sciences, Beijing, China

[3]Department of Electrical and Computer Engineering, National University of Singapore, Singapore

[4]Institute of Data Science, National University of Singapore, Singapore

{liyu,ts,jtli}@ict.ac.cn, twangnh@gmail.com, kang@u.nus.edu, wangchunfeng14@mails.ucas.ac.cn, elefjia@nus.edu.sg

1. Implementation details

1.1. Experiment setup

Our implementations are based on the MMDetection toolbox [1] and Pytorch [8]. All the models are trained with 8 V100 GPUs, with a batch size of 2 per GPU, except for HTC models (1 image per GPU). We use SGD optimizer with learning rate = 0.01, and decays twice at the 8_{th} and 11_{th} epochs with factor = 0.1. Weight decay = 0.0001. Learning rate warm-up are utilized. All *Ours* models are initialized with their corresponding baseline models that directly trained on LVIS with softmax, and only the last FC layer is trained of another 12 epochs, with learning rate = 0.01, and decays twice at the 8_{th} and 11_{th} epochs with factor = 0.1. All other parameters are frozen.

1.2. Transferred methods

Here, we elaborate on the detailed implementation for transferred long-tail image classification methods in Table 1 of the main text.

Repeat factor sampling (RFS) *RFS* [7] is applied to LVIS instance segmentation in [3]. It increases the sampling rate for tail class instances by oversampling images containing these categories. We implement RFS with its best practice settings given by [3] with $t = 0.001$.

Re-weight *Re-weight* is a category-level cost sensitive learning method. Motivated by [2], we re-weight losses of different categories according to their corresponding number of training instances. We calculate $\{\alpha_j = 1/\mathcal{N}(j) | j \in [1, 2, ..., C]\}$, where $\mathcal{N}(j)$ denotes the number of instance for category j. We normalize α_j by dividing the mean of all α, namely μ_α and cap their values between 0.01 and 5. α_0 is set to 1 for *background* class. The model (6) and (7) are both

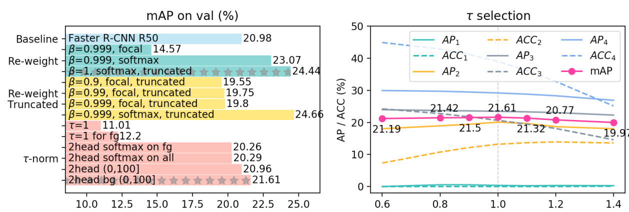

Figure 1. Settings we tried for [2] and [4].

initialized with model (1). Model (6) fine-tunes all parameters in the network except for Conv1 and Conv2. Model (7) only fine-tunes the last fully connected layer, namely W and b in Sec.3.1 in the main text, and β is set to 0.999.

Fig.1 left shows more settings we have tried for loss reweighting. we tried [2]'s best practice $\{\beta$=0.999, *focal*, γ=0.5$\}$ by setting #bg=3×#fg, but only got 14.57% mAP. $\{\beta$=0.999, *softmax*$\}$=23.07% indicates softmax works better for Faster R-CNN. So our (6) in Tab.1 are improved version of $\{\beta$=1, *softmax*$\}$ with weights truncated to [0.01,5]. We further try to add weight truncation to β=$\{$0.9, 0.99, 0.999$\}$, loss=$\{$*softmax*, *focal*$\}$, and set w_{bg}=1, γ=2 (loss for γ=0.5 is too small), and finally found that $\{\beta$=1, *softmax*, truncated$\}$ (model 7) works best.

Focal loss *Focal loss* [5] re-weights the cost at image-level for classification. We directly apply Sigmoid focal loss at proposal-level. Similar to models (6) and (7), models (8) and (9) are initialized with model (1). Then we finetune the whole backbone and classifier (W, b) respectively.

Nearest class mean classifier (NCM) *NCM* is another commonly used approach that first computes the mean feature for each class on training set. During inference, 1-NN algorithm is applied with cosine similarity on L_2 normal-

ID	Mode	Part	**mAP**	AP_1	AP_2	AP_3	AP_4	AP_r	AP_c	AP_f
(1)	train	fc-cls	*23.79*	8.16	24.42	23.35	29.26	14.36	23.04	28.50
(2)	train	head	*21.18*	9.34	21.32	20.94	25.69	12.39	20.67	25.31
(3)	tune	head	*23.88*	8.90	23.96	23.78	29.44	14.19	23.08	28.75
(4)	tune	all	***24.02***	8.91	24.86	23.49	29.06	14.81	23.36	28.52

Table 1. Different ways to train models. Mode "train" means train from random initialization. Mode "tune" means finetune from trained model (1). Part *fc-cls*, *head*, and *all* indicate the last classification FC layer, the whole classification head (2FC+fc-cls), and the whole backbone except for Conv1 and Conv2. β is set to 1 here so that the results are lower than that in the main paper where $\beta = 8$.

ized mean features [4, 9]. Thus, for object detection, with the trained Faster R-CNN model (1), we first calculate the mean feature for proposals of each class on training set except for *background* class. At inference phase, features for all the proposals are extracted. We then calculate cosine similarity of all the proposal features with the class centers. We apply softmax over similarities of all categories to get a probability vector p_n for normal classes. To recognize background proposals we directly take the probability p_0 of background class from model (1), and update p_n with $p_n \times (1 - p_0)$. We try both FC feature just before classier (model (10)), and Conv feature extracted by ROI-align (model (11)) as proposal features.

τ-normalization τ-normalization [4] directly scale the classifier weights $W = \{w_j\}$ by $\widetilde{w_i} = \frac{w_i}{\|w_i\|^\tau}$, where $\tau \in (0, 1)$ and $\|\cdot\|$ denotes L_2 norm. It achieves state-of-the-art performance on long-tail classification [4]. For model (13), we first obtain results from both the original model and the τ-normed model. The original model is good at categorizing background. Thus, if the proposal is categorized to *background* by the original model, we select the results of the original model for this proposal. Otherwise, the τ-norm results will be selected. In spite of this, we designed multiple ways to deal with *bg* (background class) (Fig 1 red bars), and found the above way perform best. We also searched τ value on *val* set, and found τ=1 is the best (Fig 1 right).

2. How to train our model

There are several options to train a model with our proposed BAGS module. As shown in Tab.1, we try different settings with $\beta = 1$. Since adding categories *others* changes the dimension of classification outputs, we need to randomly initialize the classifier weights W and bias b. So for model (1), following [4] to decouple feature learning and classifier, we fix all the parameters for feature extraction and only train the classifier with parameters W and b. For model (2), we fix the backbone parameters and train the whole classification head together (2 FC and W, b). It is worth noticing that the 2 FC layers are initialized by model (1), while W, b are randomly initialized. This drops mAP by 2.6%, which may be caused by the inconsistent initialization of feature and classifier. Therefore, we try to train W and b first with settings for model (1), and fine-tune the classification head (model (3)) and all backbones except for Conv1 and Conv2 (model (4)) respectively. Fine-tuning improves mAP slightly. However, taking the extra training time into consideration, we choose to take the setting of model (1) to directly train parameters for classifier only in all the other experiments.

3. Comparison with winners of LVIS 2019

Since the evaluation server for LVIS *test* set is closed, all results in this paper are obtained on *val* set. There are two winners: *lvlvisis* and *strangeturtle*. We compared with *lvlvisis* in Tab.3 based on their report [11], and our results surpass theirs largely. For *strangeturtle*, their Equalization Loss [10] (released on 12/11/2019) replaces softmax with sigmoid for classification and blocks some back-propagation for tail classes. With Mask R-CNN R50 baseline (mAP 20.68%), Equalization Loss achieves 23.90% with COCO pre-training (vs 26.25% of ours). Our method performs much better on tail classes (AP_r 11.70% [10] vs 17.97% ours). They also tried to decrease the suppression effect from head over tail classes, but using sigmoid completely discards all suppression among categories, even though some of them are useful for suppressing false positives. Without bells and whistles, our method outperforms both winners on *val* set.

4. Results on COCO-LT

To further verify the generalization ability of our BAGS, we construct a long-tail distribution dataset COCO-LT by sampling images and annotations from COCO [6] train 2017 split.

4.1. Dataset construction

To get a similar long-tail data distribution as LVIS, we first sort all categories of LVIS and COCO by their corresponding number of training instances. As shown in Fig. 2, we align 80 categories of COCO with 1230 categories of LVIS, and set the target training instance number per category in COCO as the training instance number of its corresponding category in LVIS. Then, we sample target number of instances for each COCO category. We make use of as many instances in a sampled image as possible. Training instances in a sampled image will only be ignored when there are plenty of instances belonging to that category.

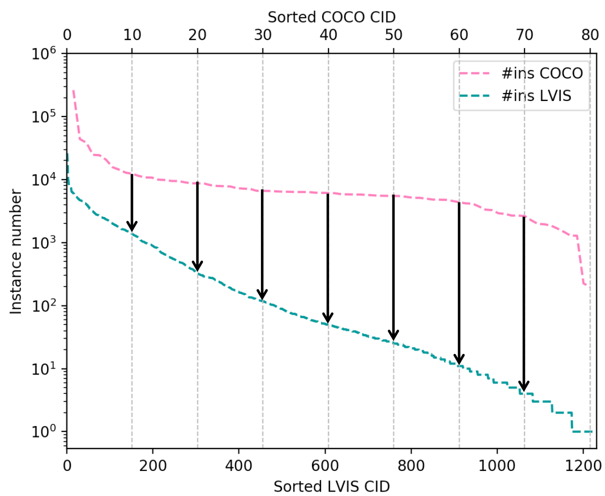

Figure 2. We align 80 categories of COCO with 1230 categories of LVIS, and sample corresponding number of instances for each COCO category.

	mAP	AP_1	AP_2	AP_3	AP_4
Faster R-CNN	*20.3*	0.1	12.9	24.3	26.7
Ours	***22.5***	13.0	18.6	24.1	26.4
Mask R-CNN bbox	*19.1*	0.0	11.1	22.9	26.4
Ours	***21.5***	13.4	17.7	22.5	26.0
Mask R-CNN segm	*18.0*	0.0	11.5	21.8	23.3
Ours	***20.3***	3.4	18.9	21.7	23.0

Table 2. Results on COCO-LT dataset. ResNet50-FPN backbone are used for both Faster R-CNN and Mask R-CNN.

In this way, we sample a subset of COCO that follows long-tail distribution just like LVIS. COCO-LT only contains 9100 training images of 80 categories, which includes 64504 training instances. For validation, we use the same validation set as COCO val 2017 split, which includes 5000 images.

4.2. Main results

We compare with Faster R-CNN and Mask R-CNN (R50-FPN backbone) on the above COCO-LT dataset. The results are shown in Tab. 2. Since the number of training images is small, we initialize baseline models with model trained on LVIS. As we can see, our models introduce more than 2% improvements on mAP of both bounding box and mask. Importantly, it gains large improvement on tail classes.

References

[1] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 1

[2] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9268–9277, 2019. 1

[3] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5356–5364, 2019. 1

[4] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*, 2019. 1, 2

[5] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2

[7] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 181–196, 2018. 1

[8] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017. 1

[9] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017. 2

[10] Jingru Tan, Changbao Wang, Quanquan Li, and Junjie Yan. Equalization loss for large vocabulary instance segmentation. *arXiv preprint arXiv:1911.04692*, 2019. 2

[11] Tao Wang, Yu Li, Bingyi Kang, Junnan Li, Jun Hao Liew, Sheng Tang, Steven Hoi, and Jiashi Feng. Classification calibration for long-tail instance segmentation. *arXiv preprint arXiv:1910.13081*, 2019. 2