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

Yu Li\textsuperscript{1,2,3*}, Tao Wang\textsuperscript{3,4}, Bingyi Kang\textsuperscript{3}, Sheng Tang\textsuperscript{1,2}, Chunfeng Wang\textsuperscript{2}, Jintao Li\textsuperscript{1,2}, Jiashi Feng\textsuperscript{3}

\textsuperscript{1}Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
\textsuperscript{2}University of Chinese Academy of Sciences, Beijing, China
\textsuperscript{3}Department of Electrical and Computer Engineering, National University of Singapore, Singapore
\textsuperscript{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

Abstract

Solving long-tail large vocabulary object detection with deep learning based models is a challenging and demanding task, which is however under-explored. In this work, we provide the first systematic analysis on the underperformance of state-of-the-art models in front of long-tail distribution. We find existing detection methods are unable to model few-shot classes when the dataset is extremely skewed, which can result in classifier imbalance in terms of parameter magnitude. Directly adapting long-tail classification models to detection frameworks can not solve this problem due to the intrinsic difference between detection and classification. In this work, we propose a novel balanced group softmax (BAGS) module for balancing the classifiers within the detection frameworks through group-wise training. It implicitly modulates the training process for the head and tail classes and ensures they are both sufficiently trained, without requiring any extra sampling for the instances from the tail classes. Extensive experiments on the very recent long-tail large vocabulary object recognition benchmark LVIS show that our proposed BAGS significantly improves the performance of detectors with various backbones and frameworks on both object detection and instance segmentation. It beats all state-of-the-art methods transferred from long-tail image classification and establishes new state-of-the-art. Code is available at https://github.com/FishYuLi/BalancedGroupSoftmax.

\*This work was supported by the National Key Research and Development Program of China (2017YFC08020605), and the National Natural Science Foundation of China (61525206, 61572472, U1703261, 61871004), and 242 project (2019A010). Jiashi Feng was partially supported by ALSG R-263-000-D97-490, NUS ECRA R-263-000-C87-133 and MOE Tier-II R-263-000-D17-112. Yu Li was partially supported by the program of China Scholarships Council (No.201904910801).†: corresponding author.

Figure 1. Sorted number of training instances (#ins) for categories in COCO and LVIS training set, and the corresponding classifier weight norm $\|w\|$ from Faster R-CNN model trained on COCO and LVIS. The x-axis represents the sorted category index of COCO and LVIS. We align 80 classes of COCO with 1230 classes of LVIS for better visualization. Category 0 indicates background.

1. Introduction

Object detection [31, 29, 25, 23, 21, 1] is one of the most fundamental and challenging tasks in computer vision. Recent advances are mainly driven by large-scale datasets that are manually balanced, such as PASCAL VOC [9] and COCO [24]. However in reality, the distribution of object categories is typically long-tailed [30]. Effective solutions that adapt state-of-the-art detection models to such class-imbalanced distribution are highly desired yet still absent. Recently, a long-tail large vocabulary object recognition dataset LVIS [14] is released, which greatly facilitates object detection research in much more realistic scenarios.

A straightforward solution to long-tail object detection is to train a well-established detection model (e.g., Faster R-CNN [31]) on the long-tail training data directly. However, big performance drop would be observed when adapting detectors designed for fairly balanced datasets (e.g., COCO) to a long-tail one (e.g., LVIS), for which the rea-
sons still remain unclear due to multiple entangled factors. Inspired by [20], we decouple the representation and classification modules within the detection framework, and find the weight norms of the proposal classifier corresponding to different categories are severely imbalanced, since low-shot categories get few chances to be activated. Through our analysis, this is one direct cause of the poor long-tail detection performance, which is intrinsically induced by data imbalance. As shown in Figure 1, we sort the category-wise classifier weight norms of models trained on COCO and LVIS respectively by the number of instances in the training set. For COCO, the relatively balanced data distribution leads to relatively balanced weight norms for all categories, except for background class (CID=0, CID for Category ID). For LVIS, it is obvious that the category weight norms are imbalanced and positively correlated with the number of training instances. Such imbalanced classifiers (w.r.t. their parameter norm) would make the classification scores for low-shot categories (tail classes) much smaller than those of many-shot categories (head classes). After standard softmax, such imbalance would be further magnified thus the classifier wrongly suppresses the proposals predicted as low-shot categories.

The classifier imbalance roots in data distribution imbalance—classifiers for the many-shot categories would see more and diverse training instances, leading to dominating magnitude. One may consider using solutions to long-tail classification to overcome such an issue, including re-sampling training instances to balance the distribution [15, 7, 32, 26] and re-weighting classification loss at category level [5, 2, 18] or instance level [23, 33]. The re-sampling based solutions are applicable to detection frameworks, but may lead to increased training time and over-fitting risk to the tail classes. Re-weighting based methods are unfortunately very sensitive to hyper-parameter choices and not well applicable to detection frameworks due to difficulty in dealing with the special background class, an extremely many-shot category. We empirically find none of these methods works well on long-tail detection problem.

In this work, to address the classifier imbalance, we introduce a simple yet effective balanced group softmax (BAGS) module into the classification head of a detection framework. We propose to put object categories with similar numbers of training instances into the same group and compute group-wise softmax cross-entropy loss separately. Treating categories with different instance numbers separately can effectively alleviate the domination of the head classes over tail classes. However, due to the lack of diverse negative examples for each group training, the resultant model suffers too many false positives. Thus, BAGS further adds a category others into each group and introduces the background category as an individual group, which can alleviate the suppression from head classes over tail classes by keeping their classifiers balanced while preventing false positives by categories background and others.

We experimentally find BAGS works very well. It improves by 9% – 19% the performance on tail classes of various frameworks including Faster R-CNN [31], Cascade R-CNN [1], Mask R-CNN [16] and HTC [4] with ResNet-50-FPN [17, 22] and ResNeXt-101-x64x4d-FPN [37] backbones consistently on the long-tail object recognition benchmark LVIS [14], with the overall mAP lifted by around 3% – 6%.

To sum up, this work makes following contributions:

• Through comprehensive analysis, we reveal the reason why existing models perform not well for long-tail detection, i.e. their classifiers are imbalanced and not trained equally well, reflected by the observed imbalanced classifier weight norms.

• We propose a simple yet effective balanced group softmax module to address the problem. It can be easily combined with object detection and instance segmentation frameworks to improve their long-tail recognition performance.

• We conduct extensive evaluations with state-of-the-art long-tail classification methods for object detection. Such benchmarking not only deepens our understandings of these methods as well as the unique challenges of long-tail detection, but also provides reliable and strong baselines for future research in this direction.

2. Related Works

Compared with balanced distribution targeted object detection [11, 31, 1], and few-shot object detection [19, 3, 38, 10], the challenging and practical long-tail detection problem is still underexplored. Though Ouyang et al. [28] proposes the concept of long-tail object detection, their work focuses on the imbalanced training data distribution on ILSVRC DET dataset [5] without few-shot setting for tail classes like LVIS [14]. [14] proposes repeat factor sampling (RFS) serving as a baseline. Classification calibration [36] enhances RFS by calibrating classification scores of tail classes with another head trained with ROI level class-balanced sampling strategy. Below we first review general object detection methods, and then long-tail classification methods.

General object detection Deep learning based object detection frameworks are divided into anchor-based and anchor-free ones. Anchor-based approaches [12, 11, 31, 29, 23] explicitly or implicitly extract features for individual regions thus convert object detection into proposal-level classification which have been largely explored. In contrast, anchor-free approaches focus on detecting key points of objects and construct final detection boxes by properly combining detected key points [21, 8, 40] or expanding the rep-
representation of key points [39, 35]. For such detectors, proposal classification is achieved by classifying the key points.

These popular object detection frameworks all employ a softmax classifier for either proposal classification or key-point classification. Our proposed balanced group softmax module can be easily plugged into such mainstream detectors by simply replacing the original softmax classifier. For simplicity, we mainly experiment with anchor-based detectors Faster R-CNN [31] and Cascade R-CNN [1] as well as their corresponding instance segmentation approaches Mask R-CNN [16] and HTC [4].

**Long-tail classification** Long-tail classification is attracting increasing attention due to its realistic applications. Current works leverage data re-sampling, cost-sensitive learning, or other techniques. For data re-sampling methods, training samples are either over-sampled (adding copies of training samples for tail classes) [15], under-sampled (deleting training samples for head classes) [7], or class-balanced sample [32, 26], which motivates RFS [14]. For cost-sensitive learning, the network losses are re-weighted at category level by multiplying different weights on different categories to enlarge the influence of tail-class training samples [5, 2, 18] or at instance level by multiplying different weights on different training samples for more fine-grained control [23, 33]. Some other approaches optimize the classifier trained with long-tail data such as Nearest Class Mean classifier (NCM) [27, 13], and τ-normalized classifier [20]. These methods are usually sensitive to hyper-parameters and do not perform well when transferred to detection frameworks due to the inherent difference between classification and detection as stated in Sec. 1.

Therefore, an approach specifically designed for long-tail object detection is desirable, and our work is the first successful attempt to overcome classifier imbalance through group-wise training without extra sampling from tail classes.

### 3. Preliminary and Analysis

#### 3.1. Preliminary

We first revisit the popular two-stage object detection framework, by taking Faster R-CNN [31] as an example. We adopt such a two-stage framework to develop and implement our idea.

The backbone network \( f_{\text{back}} \) takes an image \( I \) as input, and generates a feature map \( F = f_{\text{back}}(I) \). The feature map is then passed to ROI-align [16] or ROI-pooling [11] to produce \( K \) proposals with their own feature \( F_k = \text{ROIAlign}(F, b_k) \). Here \( b_k \) denotes proposal \( k \). The classification head \( f_{\text{head}} \) then extracts a \( d \)-dimensional feature \( h = f_{\text{head}}(F_k) \) for each of the proposals. Finally, one FC (fully connected) layer is used to transfer \( h \) to the \((C+1)\)-category prediction \((C \) object classes plus background\) by \( z = Wh + b \), where \( W \in \mathbb{R}^{d \times (C+1)} \) is the classifier weights with each column \( w_j \in \mathbb{R}^d \) related to one specific category \( j \), and \( b \) is the bias term.

During training, with ground truth label \( y \in \{0, 1\}^{C+1} \), softmax cross entropy is applied to compute loss for a specific proposal:

\[
\mathcal{L}_k(p, y) = - \sum_{j=0}^{C} y_j \log \left( p_j \right),
\]

\[
p_j = \text{softmax} \left( z_j \right) = \frac{e^{z_j}}{\sum_{i=0}^{C} e^{z_i}}.
\]

Here \( z_j \) denotes the \( i \)-th element of \( z \) and \( p_j \) is the predicted probability of the proposal being an instance of category \( j \).

#### 3.2. Analysis

Current well-performing detection models often fail to recognize tail classes when the training set follows a long-tailed distribution. In this section, we try to investigate the underlying mechanism behind such performance drop from balanced dataset to long-tailed dataset, by conducting contrast experiments on their representative examples, i.e., COCO and LVIS.

We adopt a Faster R-CNN [11] model with R50-FPN backbone. By directly comparing the mAP on the two datasets, the performance drops notably from 36.4%(COCO) to 20.9%(LVIS). Despite the unfairness as LVIS contains much more classes than COCO (1230 vs. 80), we can still draw some interesting observations. On head classes, the LVIS model achieves comparable results with COCO. However, when it comes to tail classes, the performance decreases to 0 rapidly. Such a phenomenon implies current detection models are indeed challenged by data imbalance. To further investigate how the performance degradation is induced by data imbalance, we decouple the detection framework into proposal feature extraction stage and proposal classification stage, following [20].

Specifically, following the notations in Sec. 3.1, we deem the operations used to generate \( h \) as proposal feature extraction, and the last FC layer and softmax in Eqn. (2) as a softmax classifier. Then, we investigate the correlation between the number of training instances and the weight norm \(|w_j|\) in the classifier for each category. The results are visualized in Figure 1. We can see for COCO dataset, most categories contain \(10^3 - 10^4 \) training instances (at least \(10^2\)); and classifier weight norms are also relatively balanced (0.75-1.25) for all foreground categories \(^1\). In contrast, for the LVIS dataset, a weight norm \(|w_j|\) is highly related to the number of training instances in the corresponding category \(j\); the more training examples there are, the larger weight magnitude it will be. For the extreme few-shot categories (tail classes), their corresponding weight norms are extremely

\(^1\)Note that the first class is background(CID=0).
small, even close to zero. Based on such observations, one can foresee that prediction scores for tail classes will be congenitally lower than head classes, and proposals of tail classes will be less likely to be selected after competing with those of head categories within the softmax computation. This explains why current detection models often fails on tail classes.

Why would the classifier weights be correlated to the number of training instances per-class? To answer this question, let us further inspect the training procedure of Faster R-CNN. When proposals from a head class \( j \) are selected as training samples, \( z_j \) should be activated, while the predictions for other categories should be suppressed. As the training instances for head classes are much more than those of tail classes (e.g., 10,000 vs. 1 in some extreme cases), classifier weights of tail classes are much more likely (frequent) to be suppressed by head class ones, resulting in imbalanced weight norms after training.

Therefore, one may see why re-sampling method [14, 36] is able to benefit tail classes on long-tail instance classification and segmentation. It simply increases the sampling frequency of tail class proposals during training so that the weights of different classes can be equally activated or suppressed, thus balance the tail and head classes to some degree. Also, loss re-weighting methods [5, 2, 18, 23, 33] can take effect in a similar way. Though the resampling strategy is able to alleviate data imbalance, it actually introduces new risks like overfitting to tail classes and extra computation overhead. Meanwhile, loss re-weighting is sensitive to per-class loss weight design, which usually varies across different frameworks, backbones and datasets, making it hardly deployable in real-world applications. Moreover, re-weighting based methods cannot handle the background class well in detection problems. Therefore, we propose a simple yet effective solution to balance the classifier weight norms without heavy hyper-parameter engineering.

4. Balanced Group Softmax

Our novel balanced group softmax module is illustrated in Figure 2. We first elaborate on its formulation and then explain the design details.

4.1. Group softmax

As aforementioned, detector performance is harmed by the positive correlation between weight norms and number of training examples. To solve this problem, we propose to divide classes into several disjoint groups and perform the softmax operation separately, such that only classes with similar numbers of training instances are competing with each other within each group. In this way, classes containing significantly different numbers of instances can be isolated from each other during training. The classifier weights of tail classes would not be substantially suppressed by head classes.

Concretely, we divide all the \( C \) categories into \( N \) groups according to their training instance numbers. We assign category \( j \) to group \( G_n \) if

\[
s_n^j \leq N(j) < s_n^h, \quad n > 0
\]

where \( N(j) \) is the number of ground-truth bounding boxes for category \( j \) in the training set, and \( s_n^l \) and \( s_n^h \) are hyper-parameters that determine minimal and maximal instance numbers for group \( n \). In this work, we set \( s_n^{l+1} = s_n^h \) to ensure there is no overlap between groups, and each category can only be assigned to one group. \( N \) and \( s_n^l \) are set empirically to make sure that categories in each group contain similar total numbers of training instances. Throughout this paper, we set \( N = 4, s_1^l = 0, s_2^l = 10, s_3^l = 10^2, s_4^l = 10^3, s_4^h = +\infty \).

Besides, we manually set the \( G_0 \) to contain only the background category, because it owns the most training instances (typically 10-100 times more than object categories). We adopt sigmoid cross entropy loss for \( G_0 \) here because it only contains one prediction, while for the other
4.2. Calibration via category “others”

However, we find the above group softmax design suffers from the following issue. During testing, for a proposal, all groups will be used to predict since its category is unknown. Thus, at least one category per group will receive a high prediction score, and it will be hard to decide which group-wise prediction we should take, leading to a large number of false positives. To address this issue, we add a category others into every group to calibrate predictions among groups and suppress false positives. This category others contains categories not included in the current group, which can be either background or foreground categories in other groups. For \( G_0 \), category others also represents foreground classes.

To be specific, for a proposal \( b_k \) with ground-truth label \( c \), the new prediction \( z \) should be \( z \in \mathbb{R}^{(C+1) \times (N+1)} \). The probability of class \( j \) is calculated by

\[
p_j = \frac{e^{z_j}}{\sum_{i \in \mathcal{G}_n} e^{z_i}}, \quad \{ n \mid j \in \mathcal{G}_n \}. \tag{4}
\]

The ground-truth labels should be re-mapped in each group. In groups where \( c \) is not included, class others will be defined as the ground-truth class. Then the final loss function is

\[
\mathcal{L}_k = -\sum_{n=1}^{N} \sum_{j \in \mathcal{G}_n} y^n_i \log (p^n_i), \tag{5}
\]

where \( y^n \) and \( p^n \) represent the label and probability in \( \mathcal{G}_n \).

4.3. Balancing training samples in groups

In the above treatment, the newly added category others will again become a dominating outlier with overwhelming many instances. To balance training sample number per group, we only sample a certain number of others proposals for training, which is controlled by a sampling ratio \( \beta \). For \( G_0 \), all training samples of others will be used since the number of background proposals is very large. For \( \{ \mathcal{G}_n \mid n \in \mathbb{R}, 1 \leq n \leq N \} \), \( m_n \) others instances will be randomly sampled from all others instances, where \( m_n = \beta \sum_{i \in \mathcal{G}_n} N_{batch}(i) \). \( \beta \in [0, +\infty) \) is a hyperparameter and we conduct an ablation study in Sec. 5.4 to show the impact of \( \beta \). Normally, we set \( \beta = 8 \). \( N_{batch}(i) \) indicates the instances for category \( i \) in current batch.

Namely, within the groups that contain the ground-truth categories, others instances will be sampled proportionally based on a mini-batch of \( K \) proposals. If there is no normal categories activated in one group, all the others instances will not be activated. This group is ignored. In this way, each group can keep balanced with a low ratio of false positives. Adding category others brings 2.7% improvement over the baseline.

4.4. Inference

During inference, we first generate \( z \) with the trained model, and apply softmax in each group using Eqn. (4). Except for \( G_0 \), all nodes of others are ignored, and probabilities of all categories are ordered by the original category IDs. \( p^n_0 \) in \( G_0 \) can be regarded as the probability of foreground proposals. Finally, we rescale all probabilities of normal categories with \( \bar{p}_j = p^n_0 \times p_j \). This new probability vector is fed to the following post-processing steps like NMS to produce final detection results. It should be noticed that the \( \bar{p} \) is not a real probability vector technically since the summation of it does not equal to 1. It plays the role of the original probability vector which guides the model through selecting final boxes.

5. Experiments

5.1. Dataset and setup

We conduct experiments on the recent Large Vocabulary Instance Segmentation (LVIS) dataset [14], which contains 1,230 categories with both bounding box and instance mask annotations. For object detection experiments, we use only bounding box annotation for training and evaluation. When exploring BAGS’s generalization to instance segmentation, we use mask annotations. Please refer to our supplementary materials for implementation details.

Following [36], we split the categories in the validation set of LVIS into 4 bins according to their training instance numbers to evaluate the model performance on the head and tail classes more clearly. Bin, contains categories that have \( 10^{i-1} \) to \( 10^i \) instances. We refer categories in the first two bins as “tail classes”, and categories in the other two bins as “head classes”. Besides the official metrics mAP, APr, (AP for rare classes), APC (AP for common classes), and APf (AP for frequent classes) that are provided by LVIS-api\(^2\), we also report AP on different bins. AP1 denotes the mean AP over the categories from Bin1.

5.2. Main results on LVIS

We transfer multiple state-of-the-art methods for long-tail classification to the Faster R-CNN framework, including fine-tuning on tail classes, repeat factor sampling (RFS) [26], category loss re-weighting, Focal Loss [23], NCM [20, 34], and \( \tau \)-normalization [20]. We carefully adjust the hyperparameter settings to make them suitable for

\(^2\)https://github.com/lvis-dataset/lvis-api
object detection. Implementation details are provided in our supplementary material. We report their detection performance and proposal classification accuracy in Table 1.

How well does naive baseline perform? We take Faster R-CNN with ResNet-50-FPN backbone as the baseline (model (1) in the table) that achieves 20.98% mAP but 0 AP1. The baseline model misses most tail categories due to domination of other classes. Consider other models are initialized by model (1) and further fine-tuned by another 12 epochs. To make sure the improvement is not from longer training schedule, we train model (1) with another 12 epochs for fair comparison. This gives model (2). Comparing model (2) with model (1), we find longer training mainly improves on AP2, but AP1 remains around 0. Namely, longer training hardly helps improve the performance for low-shot categories with instances less than 10. Fine-tuning model (1) on tail-class training samples (model (3)) only increases AP2 notably while decreases AP1 by 2.5%, and AP3 remains 0. This indicates the original softmax classifier cannot perform well when the number of training instances is too small.

Do long-tail classification methods help? We observe sampling-based method RFS (model (4)) improves the overall mAP by 2.5%. The AP for tail classes is improved while maintaining AP for head classes. However, RFS increases the training time cost by 1.7×. We also try to initialize the model with model (1), obtaining model (5). But mAP drops by 0.8% due to over-fitting.

For cost sensitive learning methods, both model (6) and (7) improve the performance, while model (7) works better. This confirms the observation in [20] that decoupling feature learning and classifier benefits long-tail recognition still applies for object detection. For focal loss, we directly apply sigmoid focal loss at proposal level. It is worth noting that in terms of proposal classification, the accuracy over all the object classes (ACC_{1,2,3,4}) increases notably. However, for background proposals, ACCbg drops from 95.8% to 0.16%, leading to a large number of false positives and low AP. This phenomenon again highlights the difference between long-tail detection and classification—the very special background class should be carefully treated.

For NCM, we try to use both FC feature just before classifier (model (10)), and Conv feature extracted by ROI-align (model (11)). However, our observation is NCM works well for extremely low-shot classes, but is not good for head classes. Moreover, NCM can provide a good 1-nearest-neighbour classification label. But for detection, we also need the whole probability vector to be meaningful so that scores of different proposals on the same categories can be used to evaluate the quality of proposals.

The τ-normalization model (12) suffers similar challenge as Focal Loss model (8). The many-shot background class is extremely dominating. Though foreground proposal accuracy is greatly increased, ACCbg drops hugely. Consequently, for model (13), the proposals categorized to background inherit prediction of the original model while the others take τ-norm results. However, the improvement is limited. We should notice that AP1 and ACC1 are still 0 after τ-norm, but AP2 and ACC2 are improved.

How well does our method perform? For our model, ex-

<table>
<thead>
<tr>
<th>ID</th>
<th>Models</th>
<th>mAP</th>
<th>AP1</th>
<th>AP2</th>
<th>AP3</th>
<th>AP4</th>
<th>APr</th>
<th>APc</th>
<th>APf</th>
<th>ACC</th>
<th>ACC1</th>
<th>ACC2</th>
<th>ACC3</th>
<th>ACC4</th>
<th>ACCbg</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>R50-FPN</td>
<td>20.98</td>
<td>0.00</td>
<td>17.34</td>
<td>24.00</td>
<td>29.99</td>
<td>4.13</td>
<td>19.70</td>
<td>29.30</td>
<td>92.78</td>
<td>0.00</td>
<td>2.47</td>
<td>25.30</td>
<td>45.87</td>
<td>95.91</td>
</tr>
<tr>
<td>(2)</td>
<td>x2</td>
<td>21.93</td>
<td>0.64</td>
<td>20.94</td>
<td>23.54</td>
<td>28.92</td>
<td>5.79</td>
<td>22.02</td>
<td>28.26</td>
<td>92.62</td>
<td>0.00</td>
<td>5.60</td>
<td>26.51</td>
<td>45.71</td>
<td>95.69</td>
</tr>
<tr>
<td>(3)</td>
<td>Finetune tail</td>
<td>22.28</td>
<td>0.27</td>
<td>22.58</td>
<td>23.89</td>
<td>27.43</td>
<td>5.67</td>
<td>23.54</td>
<td>27.34</td>
<td>94.81</td>
<td>0.00</td>
<td>5.04</td>
<td>5.86</td>
<td>99.85</td>
<td></td>
</tr>
<tr>
<td>(4)</td>
<td>RFS [14]</td>
<td>23.41</td>
<td>7.80</td>
<td>24.18</td>
<td>23.14</td>
<td>28.33</td>
<td>14.59</td>
<td>22.74</td>
<td>27.77</td>
<td>92.71</td>
<td>0.60</td>
<td>7.50</td>
<td>25.62</td>
<td>44.39</td>
<td>95.84</td>
</tr>
<tr>
<td>(5)</td>
<td>RFS-finetune</td>
<td>22.66</td>
<td>8.06</td>
<td>23.07</td>
<td>22.43</td>
<td>27.73</td>
<td>13.44</td>
<td>22.06</td>
<td>27.09</td>
<td>92.77</td>
<td>0.60</td>
<td>7.14</td>
<td>25.08</td>
<td>43.79</td>
<td>95.91</td>
</tr>
<tr>
<td>(6)</td>
<td>Re-weight</td>
<td>23.48</td>
<td>6.34</td>
<td>22.91</td>
<td>23.88</td>
<td>30.12</td>
<td>11.47</td>
<td>22.41</td>
<td>29.61</td>
<td>94.84</td>
<td>0.00</td>
<td>0.82</td>
<td>9.57</td>
<td>17.40</td>
<td>99.53</td>
</tr>
<tr>
<td>(7)</td>
<td>Re-weight-cls</td>
<td>24.66</td>
<td>10.04</td>
<td>24.12</td>
<td>24.57</td>
<td>31.07</td>
<td>14.16</td>
<td>23.51</td>
<td>30.28</td>
<td>94.76</td>
<td>0.00</td>
<td>0.34</td>
<td>7.72</td>
<td>16.02</td>
<td>99.64</td>
</tr>
<tr>
<td>(8)</td>
<td>Focal loss [23]</td>
<td>11.12</td>
<td>0.00</td>
<td>10.24</td>
<td>13.36</td>
<td>13.17</td>
<td>2.74</td>
<td>11.13</td>
<td>14.46</td>
<td>3.87</td>
<td>0.00</td>
<td>17.45</td>
<td>40.11</td>
<td>49.31</td>
<td>1.35</td>
</tr>
<tr>
<td>(9)</td>
<td>Focal loss-cls</td>
<td>19.29</td>
<td>1.64</td>
<td>19.30</td>
<td>20.64</td>
<td>23.70</td>
<td>6.60</td>
<td>19.81</td>
<td>23.71</td>
<td>2.90</td>
<td>0.00</td>
<td>27.67</td>
<td>48.53</td>
<td>48.89</td>
<td>0.16</td>
</tr>
<tr>
<td>(10)</td>
<td>NCM-fc [20]</td>
<td>16.02</td>
<td>5.87</td>
<td>14.13</td>
<td>16.97</td>
<td>21.40</td>
<td>10.31</td>
<td>13.92</td>
<td>20.92</td>
<td>94.29</td>
<td>0.00</td>
<td>0.02</td>
<td>0.23</td>
<td>0.15</td>
<td>100.00</td>
</tr>
<tr>
<td>(11)</td>
<td>NCM-conv [20]</td>
<td>12.56</td>
<td>4.20</td>
<td>9.71</td>
<td>13.75</td>
<td>18.46</td>
<td>6.11</td>
<td>10.39</td>
<td>17.85</td>
<td>94.29</td>
<td>0.00</td>
<td>0.00</td>
<td>0.20</td>
<td>0.10</td>
<td>100.00</td>
</tr>
<tr>
<td>(12)</td>
<td>τ-norm [20]</td>
<td>11.01</td>
<td>0.00</td>
<td>11.71</td>
<td>12.01</td>
<td>12.36</td>
<td>2.07</td>
<td>12.30</td>
<td>12.97</td>
<td>5.91</td>
<td>0.00</td>
<td>30.32</td>
<td>39.49</td>
<td>49.14</td>
<td>3.42</td>
</tr>
<tr>
<td>(13)</td>
<td>τ-norm-select</td>
<td>21.61</td>
<td>0.35</td>
<td>20.07</td>
<td>23.43</td>
<td>29.16</td>
<td>6.18</td>
<td>20.99</td>
<td>28.54</td>
<td>92.43</td>
<td>0.00</td>
<td>13.19</td>
<td>20.62</td>
<td>38.98</td>
<td>95.91</td>
</tr>
<tr>
<td>(14)</td>
<td>Ours</td>
<td>25.96</td>
<td>11.33</td>
<td>27.64</td>
<td>25.14</td>
<td>29.90</td>
<td>17.65</td>
<td>25.75</td>
<td>29.54</td>
<td>93.71</td>
<td>0.26</td>
<td>7.50</td>
<td>22.07</td>
<td>35.88</td>
<td>97.41</td>
</tr>
</tbody>
</table>

Table 1. Comparison with state-of-the-art methods transferred from long-tail image classification on LVIS val set. Bold numbers denote the best results among all models. Model (1) and (4) are initialized with model pre-trained on COCO dataset. All the others are initialized with model (1). “-cls” denotes only train the classification FC layer the best results among all models. Model (1) and (4) are initialized with model pre-trained on COCO dataset. All the others are initialized with model (1). “-cls” denotes only train the classification FC layer.
except for \( G_0 \), we split the normal categories into 4 groups for group softmax computation, with \( s^1 \) and \( s^h \) to be \((0, 10), (10, 10^2), (10^2, 10^3), (10^3, +\infty)\) respectively, and \( \beta = 8 \). Our model is initialized with model (1), and the classification FC layer is randomly initialized since the output shape changed. Only this FC layer is trained for another 12 epochs, and all other parameters are frozen. Our results surpass all the other methods by a large margin. \( \text{AP}_1 \) increases 11.3\%, \( \text{AP}_2 \) increases 10.3\%, with \( \text{AP}_3 \) and \( \text{AP}_4 \) almost unchanged. This result verifies the effectiveness of our designed balanced group softmax module.

**Extension of our method to stronger models.** To further verify the generalization of our method, we change Faster R-CNN backbone to ResNeXt-101-64x4d [37]. Results are shown in Table 2. On this much stronger backbone, our method still gains 3.2\% improvement. Then, we apply our method to state-of-the-art Cascade R-CNN [1] framework with changing all 3 softmax classifiers in 3 stages to our BAGS module. Overall mAP is increased significantly by 5.6\%. Our method brings persistent gain with 3 heads.

### 5.3. Results for instance segmentation

We further evaluate our method benefits for instance segmentation models including Mask R-CNN [16] and state-of-the-art HTC [4] on LVIS. Here HTC models are trained with COCO stuff annotations for a segmentation branch. Results are shown in Table 3. First, comparing our models (8)(10)(12) with their corresponding baseline models (7)(9)(11), mAPs of both bounding box and mask increase largely. Our models fit the tail classes much better while APs on head classes drop slightly. Second, we compare our results with state-of-the-art results [36, 14] on LVIS instance segmentation task. With both Mask R-CNN framework and ResNet-50-FPN backbone, our model (8) surpass RFS (1) and Calib (4) by at least 1.8\%. With both HTC framework and ResNeXt-101-FPN backbone, our model (10) is 1.4\% better than Calib (5). With ResNeXt-101-FPN-DCN backbone and multiscale training, our model (12) is 2.3\% better than Calib (6). Our method establishes new state-of-the-arts in terms of both bounding box and mask criterion.

**Table 2. Results with stronger backbone ResNeXt-101-64x4d and stronger framework Cascade R-CNN. All Ours models are initialized with their plain counterparts.**

<table>
<thead>
<tr>
<th>Models</th>
<th>( m\text{AP} )</th>
<th>( \text{AP}_1 )</th>
<th>( \text{AP}_2 )</th>
<th>( \text{AP}_3 )</th>
<th>( \text{AP}_r )</th>
<th>( \text{AP}_c )</th>
<th>( \text{AP}_f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster R50</td>
<td>20.98</td>
<td>17.34</td>
<td>24.00</td>
<td>29.99</td>
<td>4.13</td>
<td>19.70</td>
<td>29.30</td>
</tr>
<tr>
<td>Ours</td>
<td>25.96</td>
<td>11.33</td>
<td>27.64</td>
<td>25.14</td>
<td>29.90</td>
<td>17.65</td>
<td>25.75</td>
</tr>
<tr>
<td>Faster X101</td>
<td>24.63</td>
<td>0.79</td>
<td>22.37</td>
<td>27.45</td>
<td>32.73</td>
<td>5.80</td>
<td>24.54</td>
</tr>
<tr>
<td>Ours</td>
<td>27.83</td>
<td>14.99</td>
<td>28.07</td>
<td>27.93</td>
<td>32.02</td>
<td>18.78</td>
<td>27.32</td>
</tr>
<tr>
<td>Cascade X101</td>
<td>27.16</td>
<td>0.79</td>
<td>24.06</td>
<td>31.09</td>
<td>36.17</td>
<td>4.84</td>
<td>27.22</td>
</tr>
<tr>
<td>Ours</td>
<td>32.77</td>
<td>19.03</td>
<td>36.10</td>
<td>31.13</td>
<td>34.96</td>
<td>28.24</td>
<td>32.11</td>
</tr>
</tbody>
</table>

### 5.4. Model analysis

**Does our method balance classifiers well?** We visualize the classifier weight norm \( W \) of model (1)(4)(7) and our model (14) of Table 1 in Figure 3. Weights on head classes are obviously enlarged. Re-weighting method suppresses the weights of head classes and lifts weights of the tail classes. For ours, since we decouple the relationships of different group of categories, weights of \( G_1, G_2 \) and \( G_3 \) are almost at the same level. Though weights of \( G_4 \) are still smaller, they have been better balanced than the original model. Noting that the weights norm of our model are less related to the training instance numbers in each group, implying such decoupling benefits network training.

**How much background and others contribute?** See Table 4. With baseline model (0), directly grouping normal categories to 4 sets without adding background \( G_0 \) and others in each group, we get results of (1). For model (1), during inference, scores of each group are fed to softmax respectively, and concatenated directly for NMS. Though \( \text{AP}_1 \) improves 5.7\%, performance on all the other Bins drops significantly. This is because we do not have any constraints for FPs. For a single proposal, at least one category will be activated in each group, leading to many FPs. When we add \( G_0 \) (model (2)), and use \( p_0 \) to rescale scores of normal categories, we get 1.9\% improvement over model (1), but still worse than model (0). For model (3), we add category others into each groups, and not using \( G_0 \), we obtain 2.7\% performance gain.

**How many groups to use in BAGS?** With rescaling with \( G_0 \), another 2.2\% improvement is obtained (model (5)). If we reduce the group number from 4 to 2, as shown in model (4), the overall mAP drops 0.6. However, specifically, it should be noticed that \( \text{AP}_1 \) becomes much worse, while \( \text{AP}_4 \) increases a little. Using more groups does not help as well (model(6)). Since \#ins for Bin1 is too small for \( N = 4 \), dividing Bin1 into 2 bins further decreases \#ins of per group, leading to highly insufficient training for tails.
To sum up, adding category others into each group matters a lot, and using specially trained $p_o$ to suppress background proposals works better than just others. Finally, grouping categories into bins and decoupling the relationship between tail and head classes benefits a lot for learning of tail classes.

**Impact of $\beta$ in BAGS.** After adding category others to all groups, we need to sample training instances for others. Using all others proposals will lead to imbalance problem in each group. Thus, our strategy is to sample others with ratio $\beta$, so that $\#\text{ins} \text{others} : \#\text{ins normal} = \beta$. As shown in Fig. 4, mAP continuously increases when we increase $\beta$ until $\beta = 8$. If we use all others proposals in activated group (indicated as $n$ in x-axis), the performance for head classes keep increasing, but that for tail classes drops a lot. If we train all others proposals no matter whether there are normal categories being activated (indicated as all in x-axis), mAP gets worse. This verifies our opinion that another imbalance problem could worsen the results.

### 5.5. Results on COCO-LT

To further verify the generalization ability of our method, we conduct a long-tail distribution COCO-LT dataset by
References

[18] Chen Huang, Yining Li, Change Loy Chen, and Xiaou Tang. Deep imbalanced learning for face recognition and attribute prediction. IEEE transactions on pattern analysis and machine intelligence, 2019. 2, 3, 4


