Adaptive Hierarchical Representation Learning for Long-Tailed Object Detection

Banghuai Li
MEGVII Technology
libanghuai@gmail.com

Abstract

General object detectors are always evaluated on hand-designed datasets, e.g., MS COCO and Pascal VOC, which tend to maintain balanced data distribution over different classes. However, it goes against the practical applications in the real world which suffer from a heavy class imbalance problem, known as the long-tailed object detection. In this paper, we propose a novel method, named Adaptive Hierarchical Representation Learning (AHRL), from a metric learning perspective to address long-tailed object detection. We visualize each learned class representation in the feature space, and observe that some classes, especially under-represented scarce classes, are prone to cluster with analogous ones due to the lack of discriminative representation. Inspired by this, we propose to split the whole feature space into a hierarchical structure and eliminate the problem in a coarse-to-fine way. AHRL contains a two-stage training paradigm. First, we train a normal baseline model and construct the hierarchical structure under the unsupervised clustering method. Then, we design an AHR loss that consists of two optimization objectives. On the one hand, AHR loss retains the hierarchical structure and keeps representation clusters away from each other. On the other hand, AHR loss adopts adaptive margins according to specific class pairs in the same cluster to further optimize locally. We conduct extensive experiments on the challenging LVIS dataset and AHRL outperforms all the existing methods. We report different task results (object detection and instance segmentation) on both ResNet-50 red and ResNet-101 blue backbones. APs stands for the segmentation AP, while APb means the box AP. Our proposed AHRL outperforms all the existing methods.

1. Introduction

The emerging of convolutional neural networks (CNNs) leads to prosperity in object detection. With effort of researchers, recent advances in object detection achieve encouraging results in manually balanced datasets, like Pascal VOC [5] and MS COCO [18]. However, in reality, we always need to face long-tailed distributed data [25], where head classes(classes with plenty of instances) and tailed/scarcely classes(classes with few instances) significantly differ in the number of instances. Nevertheless, many traditional detection models are hard to take care of head classes and tailed classes in the same time, resulting in the desire for an adaptive solution.

Different from long-tailed object recognition, an additional localization sub-task makes long-tailed object detection more challenging. Extreme imbalance of the instance number for each class still restricts its performance. Almost all the past works [3, 12, 15, 31, 35, 37] on long-tailed object detection reach a consensus that classifier is the major bottleneck for further improvements. As shown in Figure
In this work, we present a simple yet effective method, named Adaptive Hierarchical Representation Learning (AHRL), from a metric learning perspective to address the long-tailed object detection problem. As shown in Figure 2b, we take Mask R-CNN [10] as an example model to train on LVIS v0.5 [8] dataset and utilize t-SNE [33] to visualize each class weight. Each dot in Figure 2b stands for a specific class center, and we select 247 out of 1230 classes for better illustration. Moreover, rare, common, and frequent classes are marked in red, green, and blue, respectively (detailed class information for those dots can be found in our supplementary materials). We can find an interesting phenomenon that some classes, especially under-represented scarce classes, are prone to cluster with analogous ones due to the lack of discriminative representation. Thus, these classes tend to be misclassified and result in poor performance. Go a step further. We adopt K-Means to group all the class centers into 50 clusters and ignore the misclassification in the same cluster to re-evaluate the performance. As depicted in Figure 2c, we distinguish this evaluation method and the standard one as coarse and fine classification results, respectively, and we observe that coarse results are much better than fine results, especially for scarce classes, which also verifies our assumption above. This discovery opens up room to optimize the long-tailed object detection performance and inspires us to handle this tough problem in a coarse-to-fine way.

Motivated by the observation above, we resort to a coarse-to-fine strategy to tackle this problem and design a two-stage training procedure AHRL from a hierarchical representation learning perspective. In the first stage, we follow standard settings in [8, 10, 31] to train a typical baseline model, i.e., Mask R-CNN. Then, we adopt unsupervised clustering algorithms, i.e., K-Means, to build the hierarchical feature space based on the classification weights of the pre-trained model. In the second stage, we propose a novel loss function, named Adaptive Hierarchical Representation loss (or AHR loss), to implement our coarse-to-fine design. AHR loss involves two optimization objectives, one for coarse-grained classification and the other one for fine-grained classification. On the one hand, AHR loss retains the constructed hierarchical structure and prompts all clusters to repel each other. On the other hand, AHR loss adopts dynamic and adaptive margins according to the specific relationship between each class pair in the same cluster, the more similar pairs they are and the larger margins between them are performed during the whole training process, to make indistinguishable classes more discriminative. We conduct extensive experiments on LVIS dataset and achieve new state-of-the-art results with both ResNet-50 [11] and ResNet-101 [11] backbones, as shown in Figure 1.

To sum up, the contributions of this work are as follow:

- We delve deep into the long-tailed object detection problem and present a strong baseline to help ease future research, which already beats the most state-of-the-art methods.
• We present a simple and effective approach, named Adaptive Hierarchical Representation Learning (AHRL), from a metric learning perspective to eliminate long-tailed object detection in a coarse-to-fine way. A novel AHR loss is also proposed to make AHRL work better.

• Compared with other existing state-of-the-art methods, our proposed method outperforms them and achieves a new state-of-the-art performance on LVIS benchmarks with various backbones.

2. Related Work

General Object Detection and Instance Segmentation. The rise of deep learning improves the performance of object detection in recent years. These deep learning-based frameworks can be divided into two categories. One-stage approaches [17, 20, 24] chase faster inference speed, while two-stage frameworks [6, 27] prefer a higher accuracy. With the appearance of Mask R-CNN [10], the gap between object detection and instance segmentation disappeared by adding an extra segmentation branch upon Faster R-CNN [27].

Long-tailed Recognition. Common methods towards long-tailed recognition can be summarized as follows. 1) Data re-sampling. It is the most intuitive way by duplicating tailed samples [8, 9] or under-sampling head samples [4] to deal with the long-tailed distribution. [38] goes a step further by changing the ratio of head and tailed classes over time. But all of them cannot avoid under-fitting in head classes or over-fitting in tailed classes. 2) Data augmentation. Generating or synthesizing new samples is always used to enlarge the limited dataset. Recent studies [1, 2, 19] manage to create fake samples for tailed classes to address long-tailed distribution. However, these methods are limited to the diversity of tailed classes. 3) Loss re-weighting. Instead of modifying input, modifying loss function directly is also a popular way to settle down this problem. Recently, several works [30, 31, 35] seek many ways to adapt the loss weight for both head and tailed classes to prevent severe and frequent punishment to tailed classes.

Long-tailed Object Detection. As long-tailed recognition becomes mature, researchers start to pay attention to long-tailed detection. Meanwhile, Facebook start a long-tailed detection challenge with dataset LVIS [8]. EQL loss [31] easily decreases the times to suppress punishment to tailed classes to conquer this problem. Following EQL, ACSL [35] prevents tailed classes from suppressing of head classes and preserves the discrimination between similar classes. Besides focusing on loss function, some methods deliberately design the last layer in the classifier. Forest R-CNN [37] constructs a classification forest with different prior knowledge to incorporates relations. BAGS [15] uses a cascade-like softmax layer to alleviate the difference between head classes and tailed classes in quantity. These structures avoid the imbalance between reward and punishment in a specific part of the model. Moreover, some adaptive methods [26, 32] provide in long-tailed classification still have an fantastic result in long-tailed object detection.

In this paper, we tackle the long-tailed object detection problem from the metric learning perspective. By splitting the whole feature space into the hierarchical structure, AHRL can handle the tough problem in a divide-and-conquer way and achieves superior results. It should be noted that Forest R-CNN [37] adopts an analogous hierarchical split method. However, it achieves this by adding a separate classification branch to distinguish parent classes, which results in a severe inconsistency between parent and fine-grained classification as these two branches are projected to different feature spaces. On the contrary, our proposed AHRL adopts unsupervised clustering algorithms based on fine-grained classes to construct the hierarchical
structure and optimize both coarse-grained and fine-grained classes at the same time.

3. Proposed Method

In this section, we first introduce a strong baseline model to help ease future research in long-tailed object detection and further verify the effectiveness of our proposed method. Then, we discuss our proposed method AHRL, followed by the details of the AHR loss.

3.1. Preliminary and A Strong Baseline

In the past works, the naive Mask R-CNN [10] is invariably adopted as the baseline model to conduct experiments and verify the superiority of their proposed methods. However, with the development of modern deep learning methods, some intuitive and mature techniques can boost naive baseline performance to a certain extent. This section delves deep into the long-tailed object detection problem and presents a strong baseline based on naive Mask R-CNN, named baseline++. Baseline++ serves as the baseline model to further verify the effectiveness of our proposed method. Its details are described as follows.

Proposal Oversampling. As shown in Figure 3a and Figure 3b, we observe a clear gap between tailed classes and head classes on the average objectness score during the model training. The proposals of tailed classes tend to achieve lower objectness score and be filtered out before the ROI head in Mask R-CNN. Figure 3b can well illustrate the phenomenon, average proposals per instance for tailed classes are much smaller than head classes, which results in less optimization for tailed classes. According to these findings, we directly double the maximum number of tailed class proposals allowed to keep after non-max suppression (NMS), bringing more foreground proposals for tailed classes.

Cosine Similarity Classifier. Generally speaking, the fully-connected (FC) layer is the default choice to implement classifiers in most object detectors. However, the fully-connected layer will result in an obvious bias towards head classes, when it comes to the long-tailed object detection problem. Supposing $F$ is the feature extractor, $W^c \in \mathbb{R}^{d \times k}$ is the final classification weight matrix, $k$ is the number of whole classes, and $W^c = [w^c_1, w^c_2, ..., w^c_k]$, where $w^c_i \in \mathbb{R}^d$ is the corresponding classification weight vector for $i$-th class. When given an input sample vector $x$, we can get the raw classification score $s^c_i$ by dot-product operation:

$$s^c_i = \frac{F(x)^T \cdot w^c_i}{\|F(x)\| \|w^c_i\|} \cos(\theta) \quad (1)$$

We can find that all things being equal, weight vectors with larger magnitude tend to yield higher scores. As shown in Figure 3c, we take Mask R-CNN [10] as an example and observe that weight vectors of head classes have a much larger magnitude comparing with tailed classes, which results in prediction preference towards head classes. Inspired by [36], we adopt a cosine similarity classifier to replace the original linear classifier to reduce the intra-class variance, which is defined as follows:

$$s^c_i = \lambda_c \cdot \frac{F(x)^T \cdot w^c_i}{\|F(x)\| \|w^c_i\|} \quad (2)$$

where $F(x)$ is the feature of a given proposal, $w^c_i$ is the weights for class $i$ and $\lambda_c$ is the scaling factor.

Moreover, compared with weights in a fully-connected classifier, weights from a cosine classifier, without preference and bias, can respond to the relationship among classes better, which can lay the foundation for subsequent excellent clustering results.

Other Effective Attempts. According to our discussion in the related work section, we adopt EQL [31] as the loss re-weighting method and GIoU [28] is utilized to replace the default Smooth L1 loss for more accurate bounding box regression. Besides, we also try several simple data augmentation methods to increase the data diversity. Due to the space limit, please refer to our Appendix for more detailed description about these attempts.

3.2. Adaptive Hierarchical Representation Learning

Motivated by our findings in Section 1, we design a simple yet effective method, named Adaptive Hierarchical Representation Learning (AHRL), from a metric learning perspective. An overview of AHRL’s pipeline can be found in Figure 4. AHRL contains a two-stage training paradigm. In the first stage, we follow standard settings in [10] to train a normal baseline++ model, illustrated in Section 3.1. Then, we construct the hierarchical feature space based on the classification nodes of pre-trained baseline++. An intuitive way to achieve this is with the help of modern clustering algorithms. In this section, without loss of generality, we only cluster these classification nodes for once and di-vide the whole classification feature space into two levels for better illustration. Assuming we get $n$ clusters finally, e.g., $P = \{P_1, P_2, ..., P_n\}$, and the cluster representation $w^c_j$ is defined as the mean of each classification node $w^c_i$ in cluster $P_j$:

$$w^c_j = \frac{\sum_{i \in P_j} w^c_i}{\|P_j\|} \quad (3)$$

where $\|P_j\|$ equals to the number of nodes in set $P_j$.

When it comes to clustering algorithms, there are two typical choices we can take. One is based on unsupervised clustering methods like K-Means to aggregate similar classification nodes, which is deemed to utilize visual information to some extend. The other one is based on lexical infor-
Figure 4. Overview of our adaptive hierarchical representation learning. In the first stage, we train a simple baseline model and each blue dot in the first circle represents a specific class center. Next, clustering algorithms, i.e., K-Means, are adopted to construct the hierarchical structure and each cluster representation (triangle in the second circle) is defined as the mean of each node in the same cluster. Finally, we fine-tune the model and optimize it with our AHR loss. \( M_{ij} \) is the adaptive margin between each class pair.

Table 1. Ablation for each component in our proposed strong baseline. \( AP_r \), \( AP_c \) and \( AP_f \) denote segmentation average precision for rare, common and frequent classes, respectively. * indicates that the reported result is based on our own implementation.

<table>
<thead>
<tr>
<th>Method</th>
<th>EQL*</th>
<th>GloU</th>
<th>Proposal Oversample</th>
<th>Data Augmentation</th>
<th>Cosine Similarity Head</th>
<th>( AP_r )</th>
<th>( AP_c )</th>
<th>( AP_f )</th>
<th>( AP_{\pi} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mask R-CNN+ [8]</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>23.6</td>
<td>24.2</td>
<td>14.0</td>
<td>24.2</td>
</tr>
<tr>
<td>Baseline++ (ours)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>25.5</td>
<td>25.9</td>
<td>17.1</td>
<td>27.1</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>25.8</td>
<td>26.2</td>
<td>17.8</td>
<td>27.1</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>26.1</td>
<td>26.4</td>
<td>17.7</td>
<td>27.6</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>26.5</td>
<td>26.7</td>
<td>17.8</td>
<td>28.0</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>26.7</td>
<td>26.8</td>
<td>17.9</td>
<td>28.2</td>
</tr>
</tbody>
</table>

loss can be defined as follows:

\[
\mathcal{L}_{AHR} = \mathcal{L}_{fine} + \lambda \mathcal{L}_{coarse} \quad (4)
\]

where \( \lambda \) is the hyperparameters to balance the scale of \( \mathcal{L}_{coarse} \).

More specifically, \( \mathcal{L}_{coarse} \) serves as a coarse-grained classification loss to distinguish each cluster clearly, and we adopt a simple cross-entropy loss to achieve this goal:

\[
L_{coarse} = -\sum_i p_i \log \sigma(s_i^p) + (1 - p_i) \log (1 - \sigma(s_i^p)) \quad (5)
\]

where:

\[
s_i^p = \lambda_p \cdot \frac{\mathcal{F}(x)^T \cdot w_i^p}{\|\mathcal{F}(x)\| \cdot \|w_i^p\|} \quad (6)
\]

\[
p_i = \begin{cases} 1, & x \in \mathcal{P}_{\pi_i} \\ 0, & x \notin \mathcal{P}_{\pi_i} \end{cases} \quad (7)
\]

where \( w_i^p \) is the weight for cluster \( i \), which is defined in Eq. (3), \( \pi_i \) is the cluster index corresponding to class \( i \), \( \sigma \) is
the Sigmoid operation, $\lambda_p$ is the scaling factor for cluster predictions.

$\mathcal{L}_{\text{coarse}}$ only focuses on coarse-grained cluster classification. Thus $\mathcal{L}_{\text{fine}}$ is necessary for further fine-grained classification in the single cluster. It is worth mentioning that our proposed $\mathcal{L}_{\text{fine}}$ adopts adaptive margin mechanism to make each class more discriminative:

$$
\mathcal{L}_{\text{fine}} = - \sum_i (y_i \log \sigma_{ad}(s_{i,y_i}^c) + (1 - y_i) \sum_j \log(1 - \sigma_{ad}(s_{i,j}^c)))
$$

where:

$$
\sigma_{ad}(s_{i,j}^c) = \frac{1}{1 + e^{-(s_{i,j}^c + M_{i,j})}}
$$

(8)

where $y_i$ is the ground-truth label of $i$-th proposal, $s_{i,j}^c$ is the raw score of $j$-th class for proposal $i$ and $M$ is a matrix measuring the specific margin values between each class pair. This paper adopts the cosine similarity between each class pair to reflect the margin between them. As we have discussed in Section 3.1, the classifier weights of our proposed method are all normalized. Thus the $M$ can be directly defined as follows:

$$
M_{i,j} = \begin{cases} 
0, & i = j \\
\lambda_m \max(0, (w_i^c \cdot (w_j^c)^T)) & i \neq j 
\end{cases}
$$

(9)

where $\lambda_m$ is a hyperparameter to control the degree of mutual exclusion, which is set as 2 by default. Moreover, because the purpose of $\mathcal{L}_{\text{fine}}$ is to optimize classification performance locally in the single cluster, $M$ is restricted to kick in for those classes in the same cluster:

$$
I_{i,j} = \begin{cases} 
1, & \pi_i = \pi_j \\
0, & \pi_i \neq \pi_j
\end{cases}
$$

(10)

$$
M = M \cdot I
$$

(11)

It is noteworthy that $M$ is calculated dynamically during the model training until all the class nodes achieve an optimal status.

Consequently, $\mathcal{L}_{\text{AHR}}$, the combination of $\mathcal{L}_{\text{coarse}}$ and $\mathcal{L}_{\text{fine}}$, works in a coarse-to-fine way to effectively address the long-tailed object detection problem. Besides, $\mathcal{L}_{\text{AHR}}$ is easy to extend to Softmax version, and we implement $\mathcal{L}_{\text{AHR}}$ in this paper based on Sigmoid for simplicity.

### 3.4. Training Objective

In the first stage, the base detector is trained with a standard Mask R-CNN [10] i.e., a typical loss $\mathcal{L}_{\text{rpn}}$ to improve the qualification of foreground proposals, a EQL [31] loss and GIoU loss for box classification and box regression respectively in ROI head. In the second stage, $\mathcal{L}_{\text{fine}}$ is adopted to our proposed baseline++, and it can be reformulated based on [31] and Eq. (8):

$$
\mathcal{L}_{\text{fine}} = - \sum_i (y_i \log \sigma_{ad}(s_{i,y_i}^c) + \sum_j E(r)T_{\lambda_c}(f_i)(1 - y_i)\log(1 - \sigma_{ad}(s_{i,j}^c)))
$$

(13)

Finally, the overall objective function in the second stage is as follow, and $\lambda$ is set as 1 by default:

$$
\mathcal{L} = \mathcal{L}_{\text{rpn}} + \mathcal{L}_{\text{reg}} + \mathcal{L}_{\text{AHR}} \\
= \mathcal{L}_{\text{rpn}} + \mathcal{L}_{\text{GIoU}} + \mathcal{L}_{\text{fine}} + \lambda \mathcal{L}_{\text{coarse}}
$$

(14)

### 4. Experiments

#### 4.1. Experiment Setup

**Datasets.** Large Vocabulary Instance Segmentation(LVIS) dataset,a large long-tailed vocabulary dataset in long-tailed detection, consists of 1230 categories in v0.5 and 1203 categories in v1.0. Since LVIS is a federated dataset [8], a few annotations are missing and few annotations are ambiguous. All categories are officially divided into three groups: frequent(more than 100 images), common(10 to 100 images), and rare.less than 10 images). Following the official guideline, we train our model on the train set and evaluate the result on the val set. Besides widely-used AP across IoU threshold from 0.5 to 0.95, AP for frequent(APf), common(APc), rare(APr) groups will be reported respectively for both object detection and instance segmentation results.

**Implementation Details.** We use Mask R-CNN [10] as our base detector and ResNet-50 [11] with a Feature Pyramid Network [16] as the backbone. We use 8 GPUs with a batch size 16 for training. Our model is trained using stochastic gradient descent(SGD) with momentum 0.9 and weight decay 0.0001 for 90k steps, with an initial learning rate of 0.02, which is decay to 0.002 and 0.0002 at 60k and 80k respectively. We adopt a class-specific branch for both mask and bounding box regression. The threshold of the prediction score is set to be 0.05. We follow [36] to set $\lambda_r$ and $\lambda_p$ as 20 in our experiments, respectively. We set $\lambda$ to 1 to balance the scale of the losses. Following [31], $\lambda_r$ is set to be $1.76 \times 10^{-3}$.

#### 4.2. Ablation Studies

In this section, we take Mask R-CNN [10] based on ResNet-50 [11] as the baseline model to perform ablation studies on LVIS v0.5 [8] unless otherwise specified.

**Ablation for each component in Baseline++.** We follow standard settings in [8, 15, 31, 37] and adopt [8] equipped with the Repeat Factor Sampling (RFS) method
Table 2. Performance comparisons with the state-of-the-art methods on LVIS v0.5 [8]. ResNet-50 and ResNet-101 are adopted as the backbones respectively for fair comparisons. ∗ indicates the reported result is based on its official implementation under Pytorch [22] framework.

<table>
<thead>
<tr>
<th>Method</th>
<th>Conference</th>
<th>Backbone</th>
<th>AP (b)</th>
<th>AP (s)</th>
<th>AP (r)</th>
<th>AP (c)</th>
<th>AP (f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQL [31]</td>
<td>CVPR 2020</td>
<td>ResNet-50-FPN</td>
<td>23.3</td>
<td>22.8</td>
<td>11.3</td>
<td>24.7</td>
<td>25.1</td>
</tr>
<tr>
<td>LST [13]</td>
<td>CVPR 2020</td>
<td>ResNet-50-FPN</td>
<td>-</td>
<td>23.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SimCal [34]</td>
<td>ECCV 2020</td>
<td>ResNet-50-FPN</td>
<td>-</td>
<td>23.4</td>
<td>16.4</td>
<td>22.5</td>
<td>27.2</td>
</tr>
<tr>
<td>Forest R-CNN [37]</td>
<td>ACMMM 2020</td>
<td>ResNet-50-FPN</td>
<td>25.9</td>
<td>25.6</td>
<td>18.3</td>
<td>26.4</td>
<td>27.6</td>
</tr>
<tr>
<td>BALMS* [26]</td>
<td>NeurIPS 2020</td>
<td>ResNet-50-FPN</td>
<td>26.4</td>
<td>27.0</td>
<td>17.3</td>
<td>28.1</td>
<td>29.5</td>
</tr>
</tbody>
</table>

|                     |            |          |       |       |       |       |       |
| Forest R-CNN [37]     | ACMMM 2020 | ResNet-101-FPN | 27.5     | 26.9    | 20.1    | 27.9    | 28.3    |

| AHRL(ours)            | N/A        | ResNet-50-FPN | 27.4     | 27.3    | 17.5    | 29.0    | 29.1    |
| AHRL(ours)            | N/A        | ResNet-101-FPN | 29.3     | 29.1    | 21.3    | 30.7    | 30.3    |

Table 3. Comparisons between various clustering strategies.

<table>
<thead>
<tr>
<th>Method</th>
<th>Clusters</th>
<th>AP (b)</th>
<th>AP (s)</th>
<th>AP (r)</th>
<th>AP (c)</th>
<th>AP (f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordNet</td>
<td>108</td>
<td>26.7</td>
<td>26.9</td>
<td>16.7</td>
<td>28.3</td>
<td>29.2</td>
</tr>
<tr>
<td>K-Means</td>
<td>108</td>
<td>26.9</td>
<td>27.0</td>
<td>16.9</td>
<td>28.6</td>
<td>28.9</td>
</tr>
<tr>
<td>K-Means</td>
<td>200</td>
<td>27.4</td>
<td>27.3</td>
<td>17.5</td>
<td>29.0</td>
<td>29.1</td>
</tr>
<tr>
<td>K-Means</td>
<td>400</td>
<td>27.1</td>
<td>26.8</td>
<td>16.1</td>
<td>28.5</td>
<td>29.3</td>
</tr>
</tbody>
</table>

Table 4. Comparisons between our proposed method and the baseline Mask R-CNN based on various backbones.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>AP (b)</th>
<th>AP (s)</th>
<th>AP (r)</th>
<th>AP (c)</th>
<th>AP (f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHRL(ours)</td>
<td>ResNet-50-FPN</td>
<td>27.4</td>
<td>27.3</td>
<td>17.5</td>
<td>29.0</td>
<td>29.1</td>
</tr>
<tr>
<td>AHRL(ours)</td>
<td>ResNet-101-FPN</td>
<td>29.3</td>
<td>29.2</td>
<td>21.3</td>
<td>30.7</td>
<td>30.3</td>
</tr>
</tbody>
</table>

as our baseline model. In Section 3.1, we propose an effective strong baseline baseline++ and Table 1 shows that each component of baseline++ can effectively promote the overall performance.

General object detectors always suffer from heavy class imbalance in long-tailed object detection problem, and EQL [31] alleviates this by ignoring the suppression to tails when they act as the negative samples. Finally, it can achieve around \(1.9\)% segmentation Average Precision (AP) and \(2.0\)% box AP gains in our implementation. GIoU [28], a more advanced IoU-based regression loss, achieves about \(0.3\)% segmentation AP and \(0.3\)% box AP gains against the default Smooth L1 loss in our settings. Unlike the common data sampling methods [3, 8], our proposed proposal oversampling for rare classes can eliminate the class imbalance problem more intrinsically, and it achieves \(0.2\)% segmentation AP and \(0.3\)% box AP improvements, respectively.

As we all know, it is almost common sense to perform data augmentation on scarce classes to ease class imbalance. In this paper, we try several simple data augmentation methods and find random cropping and color jitter can contribute to the performance, which achieves \(0.3\)% segmentation AP and \(0.4\)% box AP improvements, respectively. In addition, we replace the final fully-connected layer with the cosine similarity head for the classification and it achieves about \(0.2\)% box AP and \(0.1\)% segmentation gains, which keeps consistent with our discussion about the magnitude of weight vectors in Section 3.1. In summary, our proposed strong baseline achieves about \(2.8\)% segmentation AP and \(2.2\)% box AP improvements comparing with the original baseline in [8].

**Effectiveness of our proposed method.** We adopt two typical backbones, i.e., ResNet-50 [11] and ResNet-101 [11], to implement AHRL based on Mask-RCNN [10] to verify the effectiveness of our method. Table 4 shows the detailed comparisons. We can find that AHRL outperforms the baseline model by a large margin, whether it is based on ResNet-50 or ResNet-101. Concretely, AHRL achieves about \(3.1\)% segmentation AP and \(3.8\)% box AP gains with
Table 5. Comparisons between different training strategies.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>AP&lt;sub&gt;b&lt;/sub&gt;</th>
<th>AP&lt;sub&gt;r&lt;/sub&gt;</th>
<th>AP&lt;sub&gt;c&lt;/sub&gt;</th>
<th>AP&lt;sub&gt;f&lt;/sub&gt;</th>
<th>AP&lt;sub&gt;s&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline++</td>
<td>ResNet-50-FPN</td>
<td>26.7</td>
<td>17.9</td>
<td>28.2</td>
<td>28.7</td>
<td></td>
</tr>
<tr>
<td>Baseline++†</td>
<td>ResNet-50-FPN</td>
<td>26.8</td>
<td>15.4</td>
<td>29.0</td>
<td>29.2</td>
<td></td>
</tr>
</tbody>
</table>

ResNet-50, while it also achieves about 3.0% segmentation AP and 3.3% box AP gains with ResNet-101. This experiment proves that AHRL can work very well with various backbones and achieve promising results. We randomly sample several images from LVIS v0.5 to intuitively depict the effect of our AHRL, visualization results can be found in our appendix.

Different clustering strategies. Clustering algorithms play an important role in our proposed AHRL. In this section, we conduct extensive experiments on unsupervised K-Means and WordNet [21]. As shown in Table 3, we follow WordNet settings in [37] and group all the classification nodes into 108 clusters. We can find that the overall performance of K-Means is slightly better than WordNet under the same settings, which keeps consistent with our discussion in Section 3.2. Moreover, AHRL achieves optimal results when we group all the classes into 200 clusters. We have to emphasize that we do not pay much attention to fine-tuning the cluster hyperparameter otherwise we believe AHRL can achieve further improvements.

Discussion about our training paradigm. As we have described in Section 3.2, our proposed AHRL involves a two-stage training paradigm. To eliminate the doubt that whether the gain is brought by the 2x training time, in the second stage, we follow the same settings in AHRL to fine-tune the pre-trained model without any extra modifications, and we mark the result in Table 5 as baseline++†. We observe that baseline++† achieves comparable performance with the pre-trained model. 2x training time leads to even worse prediction bias towards head classes. It is noteworthy that we strictly share the same settings between the two-stage training paradigm, e.g., learning rate, batch size, etc. Thus, we attribute it to the influence of different initial statuses. So far, we can conclude that the improvements brought by AHRL benefit from our novel design instead of the training time.

4.3. Comparisons with State-of-the-art Methods

As shown in Table 2 and Table 6, we compare our proposed method with all the published state-of-the-art methods. It is obvious that AHRL achieves superior performance and sets up a new state-of-the-art record on both LVIS v0.5 [8] and LVIS v1.0 [8] dataset. Moreover, it is worth mentioning that our proposed AHRL is free to boost long-tailed object detection performance without any extra inference cost. Due to the space limit, detailed results of LVIS v1.0 for each sub-category are reported in our supplementary materials.

Table 6. Performance comparisons with the state-of-the-art methods on LVIS v1.0 [8].

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>AP&lt;sub&gt;b&lt;/sub&gt;</th>
<th>AP&lt;sub&gt;s&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQL [31]</td>
<td>ResNet-50-FPN</td>
<td>22.5</td>
<td>21.6</td>
</tr>
<tr>
<td>DropLoss [12]</td>
<td>ResNet-50-FPN</td>
<td>22.9</td>
<td>22.3</td>
</tr>
<tr>
<td>EQL [31]</td>
<td>ResNet-101-FPN</td>
<td>24.2</td>
<td>22.9</td>
</tr>
<tr>
<td>AHRL(ours)</td>
<td>ResNet-50-FPN</td>
<td>26.4</td>
<td>25.7</td>
</tr>
<tr>
<td>AHRL(ours)</td>
<td>ResNet-101-FPN</td>
<td>28.7</td>
<td>27.6</td>
</tr>
</tbody>
</table>

5. Conclusions

In this paper, we propose a novel yet effective method from a metric learning perspective to address the long-tailed object detection problem. Our proposed AHRL splits the whole classification feature space into a hierarchical structure and eliminates this tough problem in a coarse-to-fine way. More specifically, AHRL builds the hierarchical structure based on the classification weights of the pre-trained model in the first stage, then AHR loss retains the hierarchical structure and prompts all clusters to repel each other. In addition, according to the relationship between each class pair, an adaptive and dynamic margin mechanism is designed to make similar classes more discriminative. We conduct extensive experiments to verify the effectiveness of our proposed method, and we achieve a new state-of-the-art result on the challenging LVIS dataset based on various backbones without bells and whistles.

6. Broad Impact

Our contributions focus on the hierarchical representation learning for long-tailed object detection, which can be extended to other computer vision tasks. Also, it may provide new ideas for follow-up research. It therefore has the potential to advance both the beneficial and harmful applications of object detectors, such as autonomous vehicles, intelligent video surveillance, robotics and so on. As for ethical aspects and future societal consequences, this technology can bring harmful or beneficial effects to the society, which depends on the citizens who have evil or pure motivation and who can make good use of this technological progress.
References


