A. Other Effective Attempts in Baseline++

A.1. Data Augmentation

When facing scarce instances of tailed classes, data augmentation is the most straightforward method to battle against data hunger and alleviate overfitting. After trying several simple data augmentation methods, we found that zooming up or down images, randomly cropping the instance and color jitter positively affect the result.

A.2. Regression Loss Function

Object detection is the combination of classification and regression, and the regression loss function always plays an vital role in better localization performance. Nowadays, IoU loss [13] is widely used in object detection for box regression. For a prediction box $A$ and the corresponding ground-truth box $B$, IoU loss is defined as follows:

$$ L_{IoU} = IoU = \frac{|A \cap B|}{|A \cup B|} \quad (1) $$

Compared with IoU loss function, GIoU [9] loss, another excellent box regression loss function, is feasible to optimize in the case of non overlapping bounding boxes and sensitive to the alignment of proposed boxes and ground-truth boxes. Thus, we adopt GIoU as our box regression loss function for better performance:

$$ L_{GIoU} = IoU - \frac{|C \setminus (A \cup B)|}{|C|} \quad (2) $$

where $C$ is the smallest enclosing convex box of $A$ and $B$.

A.3. Loss Balance

Loss balance technique is a common attempt towards long-tailed object detection problem. Among these approaches [1, 8, 10], we adopt EQL [10], a simple yet effective method, to address loss balance in long-tailed object detection. EQL [10] simply ignores the suppression to tailed classes when they act as the negative samples, aiming to make the network training fairer for each class. EQL is formulated as follows:

$$ \hat{p}_j = \begin{cases} 1 - p_j, & y_j \neq 1 \\ p_j, & y_j = 1 \end{cases} \quad (3) $$

$$ L_{EQL} = - \sum_j (1 - E(r)T_{\lambda_r}(f_j)) \log(\hat{p}_j) \quad (4) $$

where $r$ denotes a given region proposal, $E(r)$ outputs 1 when $r$ is a foreground region proposal, $f_j$ is the frequency of category $j$ in the dataset, $T_{\lambda_r}(f_j)$ will be 1 only if $f_j$ is larger than $\lambda_r$. Please refer to [10] for more details.

B. Experiment Details

B.1. 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 [2], 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(AP$_f$), common(AP$_c$), rare(AP$_r$) groups will be reported respectively for both object detection and instance segmentation results.

B.2. Implementation Details

We use Mask R-CNN [3] as our base detector and ResNet-50 [4] with a Feature Pyramid Network [7] 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.
Figure 1. Mask R-CNN [3] vs. AHRL. Different colors stand for different frequency groups. Blue, green and red masks/boxes represent prediction objects from frequent, common and rare groups, respectively. Comparing with Mask R-CNN, AHRL can effectively eliminate the misclassification or missing detection problem, especially for scarce classes. For visualization, we apply the NMS with a threshold of 0.5 and filter out the predictions with a score lower than 0.05.

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 [12] to set $\lambda_c$ and $\lambda_p$ as 20 in our experiments, respectively. We set $\lambda_l$ to 1 to balance the scale of the losses. Following [10], $\lambda_r$ is set to be $1.76 \times 10^{-3}$.

B.3. Detailed Performance on LVIS v1.0

In this section, we report the performance on LVIS [2] v1.0 for each sub-category, i.e., $AP_f$, $AP_c$ and $AP_r$. We can find that AHRL outperforms all other SOTA methods for a large margin on each sub-category as well as the overall performance.

B.4. Visualization on LVIS

As shown in Figure 1, we make an intuitive visualization comparison between our AHRL and Mask R-CNN on LVIS v0.5 dataset. It is obvious that AHRL can achieve a superior performance than Mask R-CNN.

B.5. Visualization for Figure 2(b).

Detailed class information for those dots in Figure 2(b) in the paper can be found in Figure 2.

References


Table 1. Performance comparisons with the state-of-the-art methods on LVIS v1.0 [2].

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>$AP_{bh}$</th>
<th>$AP_{bs}$</th>
<th>$AP_c$</th>
<th>$AP_f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQL [10]</td>
<td>ResNet-50-FPN</td>
<td>22.5</td>
<td>21.6</td>
<td>3.8</td>
<td>21.7</td>
</tr>
<tr>
<td>EQL [10]</td>
<td>ResNet-101-FPN</td>
<td>24.2</td>
<td>22.9</td>
<td>3.7</td>
<td>23.6</td>
</tr>
<tr>
<td>AHRL (ours)</td>
<td>ResNet-50-FPN</td>
<td>26.4</td>
<td>25.7</td>
<td>16.6</td>
<td>25.4</td>
</tr>
<tr>
<td>AHRL (ours)</td>
<td>ResNet-101-FPN</td>
<td>28.7</td>
<td>27.6</td>
<td>19.3</td>
<td>27.6</td>
</tr>
</tbody>
</table>

Figure 2. Details of t-SNE [11] visualization of class weights. Three magnified areas are shown at the bottom of the image. And in every magnified area, we show the concrete index of each class for a more detailed description.


[10] Jingru Tan, Changbao Wang, Buyu Li, Quanquan Li, Wanli

