1. Implementation Details

Retentive R-CNN. As a transfer learning based method, Retentive R-CNN is trained in two steps: the first step is trained on $D^b$, which follows the same hyperparameters and learning schedule as in TFA[2]; the second step is trained on a balanced dataset of both $C^b \cup C^n$. During the finetuning stage, we set learning rate to 0.05, coefficient for consistency loss to 0.1 across all settings, and only the finetuned RPN objectness is used. Note that the model is trained until full convergence; thus, the learning schedule for finetuning may vary from different datasplits. During inference, the base detector’s classification logits are padded with 0 on the novel class entries; then, softmax operation is conducted on the padded logits to produce classification scores. As all activation in the network is ReLU and the base detector utilizes an fc classifier, logits with zero value can make good prior probabilities for novel classes, thus balance the scale of scores as the number of class entries are less than the novel detector. This improves base class AP and overall AP, e.g., overall AP increases from 32.0 to 32.1 under
MS-COCO 10-shot setting. The novel detector also predicts base class probabilities, so we include these predictions for the non-maximum suppression procedure as well. Though consistency loss enhances the similarity between the prediction of the base detector and novel detector on base classes, the novel detector’s base class predictions show ensembling effect to a certain extent, improving 0.05-0.1 base class AP upon base class AP of the pretrained model.

**Meta R-CNN**[3] & **FsDetView**[1]. These two meta-learning methods are originally finetuned on randomly selected samples; we fix the samples to be the same as ours for fintuning for a fair comparison. Note that in both works, they finetune with base class samples as much as three times more than novel class samples to maintain base class performance, while we use the same number of samples to make a fair comparison. As Meta R-CNN[3] does not provide code for training on MS-COCO in the published implementation, we train Meta R-CNN[3] with identical hyperparameters and settings as FsDetView[1] on MS-COCO, which is implemented on the top of Meta R-CNN[3].

**2. Examples for the Base Detector Rejecting Novel Class Instances**

Here we show more detection results from the pretrained base model in Figure1 to better demonstrate the property that the pretrained detector can reject novel class instances even if they are of great saliency to humans. The images are randomly selected from the first 100 images ordered by image id of MS-COCO 2014 minival set without cherry-picking. We bound the unrecognized novel class instances with black boxes and the detected objects with green boxes and their corresponding predicted category. Obviously, the base detector has a strong ability to ignore novel classes, thus false positives seldom occur from the base detector when encountering unseen classes. This property is utilized in Retentive R-CNN to maintain base class performance.

**3. More Detection Results & Failure Case Analysis**

In this section, we show some extra detection results for further demonstration of the effectiveness of our method and a qualitative failure case analysis. Figure2(a) shows representative results for comparing our method and TFA w/cos[2] under Pascal VOC split1 2-shot setting. The conclusion is consistent with the qualitative comparison shown in the main paper that our method typically performs better on base classes due to the non-forgetting property and reduces object confusion on novel class instances, successfully detecting many of the ignored objects by TFA w/cos[2]. Some extra detection visualization of our method is shown in Figure3.

Nevertheless, both our method and previous works have a vast metrics gap between few-shot classes and classes trained from abundant data, indicating that few-shot object
Figure 3: Extra visualization of the detection results from Retentive R-CNN. The first row shows results under MS-COCO 10-shot settings while the second row shows results under Pascal VOC split1 2-shot settings.

<table>
<thead>
<tr>
<th>Methods</th>
<th>5 shot</th>
<th></th>
<th></th>
<th>10 shot</th>
<th></th>
<th></th>
<th>30 shot</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AP</td>
<td>bAP</td>
<td>nAP</td>
<td>AP</td>
<td>bAP</td>
<td>nAP</td>
<td>AP</td>
<td>bAP</td>
</tr>
<tr>
<td>Retentive R-CNN</td>
<td>31.4</td>
<td>39.3</td>
<td>7.7</td>
<td>31.8</td>
<td>39.2</td>
<td>9.5</td>
<td>32.6</td>
<td>39.3</td>
</tr>
<tr>
<td>FRCN-ft-full[2]</td>
<td>14.4</td>
<td>17.6</td>
<td>4.6</td>
<td>13.4</td>
<td>16.1</td>
<td>5.5</td>
<td>13.5</td>
<td>15.6</td>
</tr>
<tr>
<td>TFA w/ fc[2]</td>
<td>25.6</td>
<td>31.8</td>
<td>6.9</td>
<td>26.2</td>
<td>32.0</td>
<td>9.1</td>
<td>28.4</td>
<td>33.8</td>
</tr>
<tr>
<td>TFA w/ cos[2]</td>
<td>25.9</td>
<td>32.3</td>
<td>7.0</td>
<td>26.6</td>
<td>32.4</td>
<td>9.1</td>
<td>28.7</td>
<td>34.2</td>
</tr>
</tbody>
</table>

Table 1: Results over 10 random runs on COCO dataset under 5, 10, 30-shot settings. Note that we use the same samples as TFA[2] so that the metrics are directly comparable. We obtain better performance in terms of all metrics.

4. Results over Multiple Runs

To show the effectiveness of our method without random effect, we ran our model over 10 sets of random samples under 5, 10, 30-shot settings on COCO dataset, using exactly the same samples as TFA[2]. The results are shown in Table1. We obtain better performance in terms of all metrics (AP, bAP, nAP) under each of these settings.

References

