Appendix

This appendix is organized as follows.

- In Section A, we provide details of the downstream datasets we used.
- In Section B, we list the prompt templates used for zero-shot classification experiments.
- In Section C, we describe the training details of our linear probe experiments.
- In Section D, we provide more implementation details of the `MultiheadAttn` used in the proposed Retrieval Augmented Module (RAM).
- In Section E, we provide more visualization to show that RAM is robust to the quality of the retrieved image-text pairs.

A. Downstream Datasets

We have 12 widely used downstream datasets: ImageNet [4], ImageNet V2 [13], CIFAR 10 [8], CIFAR 100 [8], Caltech 101 [5], Oxford Pets [11], SUN 397 [14], Food 101 [3], Stanford Dogs [7], COCO [2] and LVIS [6]. Table 1 summarizes the details of these datasets. For ImageNet V2, we use the same training data of ImageNet for the linear probe classification experiments. For COCO and LVIS, we only use them to evaluate the zero-shot ROI classification, thus we don’t need training data. The classification datasets use classification accuracy as evaluation metric, except for Caltech 101 and Oxford Pets, which use averaged per-class accuracy. The detection datasets use average precision as evaluation metric.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Classes</th>
<th>#Train</th>
<th>#Test</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet</td>
<td>1,000</td>
<td>1,281,167</td>
<td>50,000</td>
<td>accuracy</td>
</tr>
<tr>
<td>ImageNet V2</td>
<td>1,000</td>
<td>–</td>
<td>50,000</td>
<td>accuracy</td>
</tr>
<tr>
<td>CIFAR 10</td>
<td>10</td>
<td>50,000</td>
<td>10,000</td>
<td>accuracy</td>
</tr>
<tr>
<td>CIFAR 100</td>
<td>100</td>
<td>50,000</td>
<td>10,000</td>
<td>accuracy</td>
</tr>
<tr>
<td>Caltech 101</td>
<td>102</td>
<td>3,060</td>
<td>6,085</td>
<td>mean-per-class</td>
</tr>
<tr>
<td>Oxford Pets</td>
<td>37</td>
<td>3,680</td>
<td>3,669</td>
<td>mean-per-class</td>
</tr>
<tr>
<td>SUN 397</td>
<td>397</td>
<td>19,850</td>
<td>19,850</td>
<td>accuracy</td>
</tr>
<tr>
<td>Food 101</td>
<td>102</td>
<td>75,750</td>
<td>25,250</td>
<td>accuracy</td>
</tr>
<tr>
<td>DTD</td>
<td>47</td>
<td>3,760</td>
<td>1,880</td>
<td>accuracy</td>
</tr>
<tr>
<td>Stanford Dogs</td>
<td>120</td>
<td>12,000</td>
<td>8,580</td>
<td>accuracy</td>
</tr>
<tr>
<td>COCO</td>
<td>81</td>
<td>–</td>
<td>5,000</td>
<td>average precision</td>
</tr>
<tr>
<td>LVIS</td>
<td>1,203</td>
<td>–</td>
<td>5,000</td>
<td>average precision</td>
</tr>
</tbody>
</table>

B. Prompt Engineering

Following previous works [9, 10, 12], we extend the category names into sentences with prompts such as “a photo of `{label}`.” before feeding them into the text encoders. For a fair comparison, we adopt the same prompts used in CLIP [12]. Specifically, for Oxford Pets, we use “a photo of a `{label}`, a type of pet.”, while for Food 101 dataset, we use “a photo of a `{label}`, a type of food.”. For the other datasets, we use 80 prompt templates as shown in Figure 1. For a given category name, we average the embeddings of different prompted sentences, and conduct L2 normalization to obtain the final category embedding.

C. Training Details of Linear Probe

We freeze the pre-trained image encoder and append a linear classifier after it for linear probe classification. During training, we apply data augmentation to the input image. Concretely, we random crop a 224×224 patch from input image, then conduct random horizontal flip. During testing, we resize the shorter size to 224 then center crop a 224×224 patch as input image. We train the classifier for 90 epochs except for the ImageNet dataset, for which we train 10 epochs in total due to the large data volume. The learning rate follows a cosine decay schedule with initial learning rate equal to 0.1. We use SGD with momentum for
optimization. Weight decay is not used in our experiments. The batch size is set to 128.

D. More Implementation Details of RAM

Equation 2 and Equation 3 in Section 3.2 of the main paper adopt MultiheadAttn blocks to aggregate reference embeddings \( \{e_k^T\}_{k=1}^K \) and \( \{e_k^{i,T}\}_{k=1}^K \) for input \( v_i \). In this section, we take Equation 2 for example and provide more implementation details of it. As shown in Figure 2, the MultiheadAttn block used in Equation 2 contains not only multi-head attention layer, but also layer normalization, feed forward block and short-cut connections. Given \( v_i \), RAM scans all reference image embeddings \( \{e_k^T\}_{k=1}^K \) and gathers related textual information from \( \{e_k^{i,T}\}_{k=1}^K \) into a new embedding \( \alpha_i^{T} \). This process can be repeated for several times to iteratively refine \( \alpha_i^{T} \), each block takes \( v_i \), \( \{e_k^T\}_{k=1}^K \), \( \{e_k^{i,T}\}_{k=1}^K \) and previous block’s output \( \alpha_i^{T} \) as inputs, then update \( \alpha_i^{T} \) as output. The initial \( \alpha_i^{T} \) fed into the first block is set to all-zero embedding.

E. More Visualization

In this section, we provide more visualization results of reference retrieval described in Section 3.2. As shown in Figure 3, the retrieval process may not return image-text pairs that provide description about the ground-truth category. However, the proposed RAM is robust to the quality of the retrieval results and still produces correct
predictions. Specifically, for the first retrieval example in Figure 3, the input image is a photo of a *taxicab*, although the retrieved images are similar to the input image, their corresponding texts are not descriptions about *taxicab*. Since the proposed RAM not only depends on the retrieved image-text pairs, but also takes the embedding of input image into consideration, we can still give correct prediction for this case.

![Image Retrieval](image1)

GT: Taxicab
Prediction: Taxicab

![Image Retrieval](image2)

GT: Vulture
Prediction: Vulture

![Image Retrieval](image3)

GT: Small White Butterfly
Prediction: Small White Butterfly

Figure 3. More visualization results of reference retrieval.

References


