Supplementary Material:

Matching Is Not Enough: A Two-Stage Framework for Category-Agnostic Pose Estimation

This supplementary material includes the following five parts.

**Part 1** presents more implementation details, including network architecture and the training recipe (in main paper, Sec. 4.1, line 559).

**Part 2** presents the comparison with the semantic correspondence approaches.

**Part 3** presents more qualitative results (Fig. 5 in main paper).

**Part 4** shows more visualizations of the attention maps (Fig. 7 in the main paper).

**Part 5** includes the visualizations of the similarity maps and the similarity-aware position proposals.
Network architecture. Here, we specify more details on network architecture. For all the encoder and the decoder layers, the embedding dimension is set to 256. The raw feature maps with 2048 channels from the backbone are first compressed to 256 channels with a 1 × 1 convolution. The number of heads for all the self- and the cross-attention layers are set to eight. The detailed architecture for the feed-forward network and the offset prediction head (in line 483 of the main paper) are illustrated in Fig. A1. Note that the overall design of the encoder and decoder has been illustrated in Fig. 3 and Fig. 4 of the main paper.

For the similarity-aware proposal generator, we use the same configuration as in BMNet [7]. The detailed architecture is illustrated in Fig. A2.

Training recipe. We use pytorch [6] and MMPose [4] framework to develop the method. We set the dropout rate of attention layers and feed-forward networks to be 0.1 following DETR [1]. We use a linear warm-up strategy for 1,000 iterations. The warm-up ratio is set to be 0.001.

To generate the keypoint heatmaps, we set the variance of the Gaussian kernel to 2.0. Note that, for the support keypoints, the heatmaps are used as the soft masks to obtain the support keypoint features (Sec. 3.2 in the main paper). Meanwhile, the heatmaps for keypoints in the query images are used as the ground truth for the heatmap loss (see Eq. (5) in the main paper).

Minor design choice. Here we ablate some minor design choices, including hyper-parameters and parameter sharing. For hyper-parameters, we analyze the impact of different $\lambda_h$, i.e., the weight for heatmap loss in Table A1. We test the PCK under 1-shot setting on MP-100 split1. The results show that 2.0 is an optimal choice, and the impact of $\lambda_h$ is insignificant. The other hyper-parameters are not further fine-tuned.
We also notice that POMNet [9] adopts separated backbones for the support and query images, while the proposed CapeFormer adopts a shared one. The offset prediction head can also be shared across the decoder layers. We compare different parameter-sharing settings in Table A2. Although using separated backbones, as in POMNet, can increase the performance slightly, we adopt a shared backbone considering the efficiency.

Table A1. The impact of different heatmap loss weights ($\lambda_h$).

<table>
<thead>
<tr>
<th>$\lambda_h$</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCK</td>
<td>89.36</td>
<td>89.45</td>
<td>89.26</td>
<td>89.03</td>
<td>88.58</td>
</tr>
</tbody>
</table>

Table A2. The impact of parameter sharing for backbone and offset prediction head.

<table>
<thead>
<tr>
<th>Backbone</th>
<th>Offset Head</th>
<th>PCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>×</td>
<td>89.45</td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>89.74</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>89.08</td>
</tr>
</tbody>
</table>
Part 2: Comparison with Semantic Correspondence  Here we show the comparison between CAPE and the semantic correspondence models [3, 5], and then discuss their differences. As mentioned in the main paper, SC can naturally conduct CAPE. However, we find that directly applying SC models is not an optimal choice for CAPE. As shown in Table A3 (CATs++ [2] is currently one of the best practices on SC), both the pre-trained and the fine-tuned CATs++ (denoted by *) fall behind POMNet [9] and CapeFormer in terms of accuracy and efficiency, but outperform the metric learning baseline ProtoNet [8]. We find the fine-tuned CATs++ hard to converge during training, thus failing to achieve better results.

Table A3. Quantitative comparison with the semantic correspondence models. * denotes that the model is fine-tuned on MP-100 dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>ProtoNet</th>
<th>POMNet</th>
<th>CapeFormer</th>
<th>CATs++</th>
<th>CATs++*</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCK</td>
<td>46.05</td>
<td>84.23</td>
<td>89.45</td>
<td>66.76</td>
<td>59.18</td>
</tr>
<tr>
<td>GFLOPs</td>
<td>–</td>
<td>38.01</td>
<td>23.68</td>
<td>36.23</td>
<td>36.23</td>
</tr>
</tbody>
</table>

Then we discuss the differences, which are three folds: task, methodology, and data bias. For task, SC aims to predict a dense correspondence field for every pixel in the source (support) image. The core of this process is to compute a correlation map with the size of $hw \times hw$. In contrast, CAPE only computes similarity maps for input support keypoints. Hence, directly using SC is inefficient.

In terms of methodology, SC researchers find coarse correlation maps unreliable and noisy due to similar appearance and repetitive patterns. Hence, similar to our two-stage design, post-processing techniques are designed for correlation results. Unfortunately, most of these techniques in SC models can not be adapted to CAPE as they heavily rely on the full-size $hw \times hw$ correlation map, such as the commonly-used geometry consistency constraints.

Another difference can be the bias on the training data, most training data for SC models is rigid objects, e.g., aeroplanes, faces, or bicycles. For CAPE, one needs to consider non-rigid deformation and occlusions.
More qualitative results are shown in Fig. A3 and Fig. A4. Note that here we also add the qualitative comparison with the previous best method POMNet [9].

Figure A3. More qualitative results #1. One-stage removes the decoder in the proposed CapeFormer.
Figure A4. More qualitative results #2. One-stage removes the decoder in the proposed CapeFormer.
Part 4: Attention Visualization

More visualizations of the encoder attention maps. We visualize more attention maps for the query-support joint refine encoder in Fig. A5. We only visualize the support-to-query attention of the last encoder layer. The encoder attention indicates which location in the query image the support keypoints will attend to during information fusion.

Figure A5. Visualizations of the encoder attention map. The left column shows the support image with support keypoints. The other columns represent the attention maps for different support keypoints. The support keypoint for each attention map is marked in the attention maps, whose color is identical to the corresponding keypoints in the support images.
More visualizations for the decoder attention maps. More visualizations of the cross-attention in the last proposal refine decoder layer are shown in Fig. A6. The attention shows which features will be extracted for each support keypoint during position update.

Support Image | Decoder Attention (last layer)
--- | ---

![Visualizations of the decoder attention map.](image)

The left column shows the support image with support keypoints. The other columns represent the attention maps for different support keypoints. The support keypoint for each attention map is marked in the attention visualizations, whose color is identical to the corresponding keypoints in the support images.
Part 5: Visualizations of Similarity Map and Similarity-Aware Proposals

We visualize the similarity maps and similarity-aware proposals in Fig. A7.

![Support Image](image1)
![Similarity Map](image2)

Figure A7. Visualizations of similarity maps and similarity-aware proposals. The left column shows the support images with support keypoints. The other columns visualize the similarity maps for different support keypoints. The keypoints are marked on the similarity maps, whose colors are identical to the corresponding keypoints in the support images. We mark the ground truth annotations with red points and the final predictions with orange points.
References


