Supplementary Material: CapeNext: Rethinking and refining dynamic support information for category-agnostic pose estimation

Yu Zhu[1,2], Dan Zeng[1*], Shuiwang Li[3], Qijun Zhao[4], Qiaomu Shen[5], Bo Tang[2]

[1]School of Artificial Intelligence, Sun Yat-sen University, Zhuhai 510275, China
[2]Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China
[3]College of Computer Science and Engineering, Guilin University of Technology, Guilin 541004, China
[4]College of Computer Science, Sichuan University, Chengdu 610065, China
[5]Aerospace and Informatics Domain, Beijing Institute of Technology, Zhuhai 519088, China

1 Additional Experiments

In this section, we extend the result of the manuscript and conduct additional experiments on the proposed modules to validate our effectiveness. The experiments settings of network backbone are consistent with the settings in the manuscript. We will make our project codes public in the final version.

1.1 Extended Main Results.

Results with Different PCK Thresholds. Following the setting of previous CAPE methods, we mainly report CapeNext's PCK@0.2 performance. Here, we also show the average PCK results of CapeNext and CapeX for the comparison with the threshold of 0.05, 0.1, 0.15, 0.2, 0.25 in Table 1. Our method consistently outperforms CapeX, the SOTA method, to a large margin. The performance gain is even larger at smaller threshold, exceeding it by 1.89% in PCK@0.05.

Additional Qualitative results. In order to comprehensively show the superior performance of our model, we have provided additional visualization results compared to CapeX in Figure 2, which complement and enhance the visual evidence presented earlier.

Due to the challenges of the CAPE task, we also present some failure cases in Figure 3. Although the prediction results are inaccurate, they still show improvements compared to the baseline.

Results on Sub-datasets MP-100 is composed of 13 distinct pose estimation datasets, serving as a representative benchmark that includes 5 distinct data splits. To conduct a refined performance analysis, we first traced the original source dataset of each test sample within MP-100's 5 splits. Subsequently, we computed the performance of the models on each dataset separately for every individual split and averaged the performance scores obtained for each dataset across the 5 splits to derive the final evaluation results, which are presented in Table 6.

Analysis of DSFR's Intermediate Embeddings and Dynamic Adjustment To investigate DSFR's dynamic adjustment capability for image and text embeddings, we conducted experiments on the MP-100 test set. Taking Split-1 as an example, we randomly sampled 300 cases from this split and computed the average cosine similarity between DSFR's inputs (joint embedding, image embedding, text embedding) and its output enhanced joint embedding. As shown in Table 2, when incorrect category descriptions are used as text inputs, the cosine similarity of text embeddings decreases (4.22% vs. 3.95%), while the image embeddings increases (1.11% vs. 1.96%). This observation directly confirms DSFR's ability to dynamically adjust the contribution of image and text embeddings based on input quality.

1.2 Polysemy Handling and Fine-grained Category Control Evaluation.

We check the performance of our method and CapeX when meeting the polysemy issue. We evaluate models on the super-category of animal body(average PCK results on 5 splits of different thresholds) in Table 3. The categories under "animal body" share identical keypoint definitions and prompts but exhibit distinct visual cues due to inherent category differences (e.g., dog, horse, rabbit). This makes them well-suited for verifying the model's ability to resolve polysemy issue when keypoint texts are identical across different categories.

To examine the model's capability for fine-grained category control, we evaluate the performance of our method and CapeX on the "cat body" category. Below cat examples (standing black cat vs. curled white cat) show the large intra-class variations in Figure 1. Added experiments show that ours solves these issues better.

1.3 Detailed Results of Robustness to Noisy Keypoint Templates and Category Description Text.

To verify the model's adaptability and robustness when using incorrect keypoint prompts or different keypoint prompt

*Corresponding author: zengd8@mail.sysu.edu.cn
Codes and supplementary materials of this paper are available at https://github.com/yzrs/CapeNext

Methods	PCK@0.05	PCK@0.1	PCK@.15	PCK@0.2	PCK@0.25	Avg
CapeX	46.62	71.25	81.96	87.61	90.95	75.68
CapeNext	48.51	72.66	83.13	88.37	91.51	**76.83**(+1.15)

Table 1: PCK performance of CapeX and CapeNext of different thresholds.

Settings	Noise Type	joint	text	img
CapeNext	-	81.14%	4.22%	1.11%
CapeNext	random cls	81.09%	3.95%↓	1.96%↑

Table 2: Cosine similarity score between DSFR's inputs and output when using different category description text.

Figure 1: Cats in MP-100 with large intra-class variations.

templates, we conducted experiments on MP-100, comparing the performance of CapeX and CapeNext (same as the manuscript). Extended results are presented in Table 4. It is evident that CapeNext outperforms CapeX both when using misspelled keypoint prompts and when employing keypoint prompt templates different from those used during training.

We also replaced the category information in the category description text with null values and random categories during testing. The results in Table 5 show that the model's PCK metric decreases minimally, demonstrating the model's robustness against erroneous category description text.

1.4 Effect of Refinement Module.

Layer Number Experiment Our proposed innovative module of HCMI and DSFR can be encapsulated into an independent refinement layer. To explore the optimal usage of this layer, we conducted experiments to investigate the effects of stacking it multiple times. This layer takes three inputs: image embedding e_{img}, class embedding e_{cls}, and joint embedding e_{joint}, and produces an enhanced joint embedding $e^{'}_{joint}$. When stacking multiple layers, each subsequent layer takes the original inputs e_{img} and e_{cls}, along with the enhanced output $e^{'}_{joint}$ from the previous layer, and generates a further refined output $e^{''}_{joint}$. Formally, the operation of the i-th layer can be described as follows:

$$e^{i}_{joint} = Layer(e_{img}, e_{cls}, e^{i-1}_{joint}), \quad (1)$$

where $e^{0}_{joint} = e_{joint}$ represents the initial input, and e^{i}_{joint} denotes the enhanced output after the i-th layer.

The corresponding experimental results in Table 7 demonstrate that the model achieves its best performance when this layer is stacked only once. Specifically, with a single layer, the model achieves a PCK@0.2 accuracy of 88.37% when the model achieves a lower average accuracy with double or triple layers. This is because a single enhancement layer strikes the right balance between enriching the joint embedding with additional class/instance information and preserving the integrity of the original keypoint features. Further enhancements risk introducing redundancy or suppressing critical feature expressions, which could degrade the model performance. Therefore, according to Equation 7, enhancing joint embedding with multimodal embedding only once proves to be the most appropriate for our model in the MP-100 benchmark.

Module Experiment We replaced the HCMI/DSFR module with a standard Transformer Encoder to identify the source of performance gains. Results in Table 8 confirm that our proposed new module is the core driver behind the improved performance.

1.5 Ablation Study on Image Masking.

To investigate the potential of image embedding in improving CAPE, we also considered applying a mask to the images using the bounding box information provided in the dataset.

Specifically, we set the pixels outside the bounding box to zero, retaining only the target instance. This process aims to minimize noise unrelated to the target instance in the images, thereby increasing the proportion of effective information related to the target instance in the image embedding. However, it is worth-noting that for face-related data, the mask also obscures significant portions of the body that are relevant to the subject, leaving only the facial region.

We conducted an ablation study for our multimodal input data in this setting. The experimental settings, including the backbone network, remains consistent with previous settings, except for the mask process during training and test process. As we can see in Table 9, the application of mask processing improves the model's accuracy across multiple settings. Specifically, the model employing our multimodal input achieves the highest accuracy, reaching 88.53% in terms of PCK@0.2. However, it should be note that since mask processing was applied during both training and testing, these results are not directly comparable to the MP-100 benchmark.

Methods	Species	PCK@0.05	PCK@0.1	PCK@0.15	PCK@0.2	PCK@0.25	Avg
CapeX	animal body	44.28	73.84	86.39	92.01	94.84	78.27
CapeNext	animal body	46.12	74.52	86.80	92.19	94.94	**78.91**(+0.64)
CapeX	cat body	44.61	71.85	85.58	91.91	94.89	77.77
CapeNext	cat body	45.74	74.37	87.10	92.42	95.33	**78.99**(+1.22)

Table 3: PCK performance of different thresholds on specific super-category and category from MP-100 test dataset.

Methods	Noise Type	Split1	Split2	Split3	Split4	Split5	Avg
CapeX	keypoint typo	64.01	65.32	59.51	64.24	67.41	64.10
CapeNext	keypoint typo	64.39	68.37	64.87	65.84	68.26	**66.35**(+2.25)
CapeX	paw→foreleg	90.83	86.6	83.75	85.1	87.57	86.77
CapeNext	paw→foreleg	90.21	86.6	85.07	85	89.61	**87.30**(+0.53)
CapeX	paw→foot	91.55	86.91	84.34	85.86	88.24	87.38
CapeNext	paw→foot	91.82	86.93	85.43	86.11	90	**88.06**(+0.68)
CapeX	eye→eyeball	91.9	86.97	84.39	86.11	88.62	87.60
CapeNext	eye→eyeball	92.01	86.95	85.5	86.29	90.18	**88.19**(+0.59)
CapeX	elbow→forearm	91.69	86.89	84.11	85.83	88.32	87.37
CapeNext	elbow→forearm	92.11	86.82	85.27	86.02	90.05	**88.05**(+0.68)
CapeX	shoulder→upper arm	91.53	86.62	82.99	85.25	87.68	86.81
CapeNext	shoulder→upper arm	92.04	86.7	84.04	85.65	89.45	**87.58**(+0.77)

Table 4: PCK@0.2 results on MP-100 when using different keypoint prompt templates.

1.6 CapeNext With Support Image.

To explore whether including a support image can lead to furthe improvements, we encode one support image and the query image with CLIP to get their embeddings and fused them with gating mechanism mentioned in DSFR module as the new query image embedding. This new query image embedding will be taken as one input of DSFR module to perform the subsequent computation, same as the original CapeNext. PCK results are shown in Table 10, indicating that our multi-modal inputs are almost enough for CAPE task, while the support images are expensive to collect.

1.7 Error-induced Scenario Predictions.

To rigorously validate the effectiveness of image embeddings and model robustness, we also show the qualitative results comparison between ours with e_{img} and e_{cls} and ours but only with e_{cls} in Figure 4. The textual class information is randomly replaced with the wrong class. For example, the text prompt "a photo of chair" for a photo of a chair may be replaced with "a photo of pademelon face" where the incorrect class ("pademelon face") is randomly sampled from MP-100 to create intentional semantic mismatching. Visualization results in the second column show that when exclusively employing incorrect class embedding e_{cls}, the predicted keypoints exhibited substantial positional deviation from the ground truth, particularly out of target body regions. The introduction of e_{img} effectively mitigates this problem, offsetting the semantic ambiguity in the incorrect class embedding, demonstrating the robustness of CapeNext in this error-induced scenario, as shown in the third column.

1.8 Computational Overhead.

To assess CapeNext's computational overhead, we compared it with baseline CapeX via one-epoch training on an NVIDIA TITAN Xp (batch size=8). With Swinv2-T as image backbone, CapeNext (116.5M params) requires 10% more GPU memory (3588MB) and longer computation time (676.7s) than CapeX (100.7M params, 3293MB, 606.3s), but delivers superior performance (verified in subsequent sections).

We also tested the computational overhead of other backbones in Table 11. Results show lightweight backbones (DINOv2-ViT-S: 22.06M params, 544s) still achieve high Avg PCK@0.2 (87.95%). Swinv2-T (28.35M params, 676s) hits the highest Avg PCK@0.2 (88.37%), while heavier Vit-Base-16 (86.39M params) gives no proportional gains, validating our backbone selection and CapeNext's efficiency.

Methods	Noise Type	Split1	Split2	Split3	Split4	Split5	Avg
CapeNext	-	92.44	87.31	85.44	86.47	90.17	88.37
CapeNext	null cls string	91.85	86.95	85.50	85.98	90.30	88.116
CapeNext	random cls string	91.95	86.95	85.48	86.13	90.27	88.156

Table 5: PCK@0.2 results on MP-100 when using different category description text.

Methods	AnimalWeb	AP-10K	CarFusion	DeepFashion	Deeppose
CapeX	85.85	78.55	48.14	79.06	45.50
CapeNext	86.26	78.92	48.74	79.72	46.91

Table 6: Performance comparison between CapeX and CapeNext on constituent datasets of MP-100.

Layer Number	Split1	Split2	Split3	Split4	Split5	Avg
0	91.9	86.97	84.41	86.13	88.64	87.61
1	92.44	**87.31**	85.44	86.47	**90.17**	**88.37**
2	92.42	86.38	**85.8**	**86.6**	89.95	88.23
3	**92.62**	85.8	85.44	86.57	89.53	87.99

Table 7: Ablation study for the layer number on MP-100 test dataset. Layer number of "0" corresponds to the baseline and "1" corresponds to CapeNext setting. "2","3" corresponds to repeating the joint embedding refinement operation two to three times, respectively.

Methods	Params(M)	Avg
CapeX	100.7M	87.61
CapeX+Trans.Encoder	120.7M	86.96
CapeNext	116.5M	88.37

Table 8: Average PCK@0.2 performance comparison of different module configurations.

Settings	Split1	Split2	Split3	Split4	Split5	Avg
baseline	92.59	87.53	83.85	87.2	88.46	87.92
baseline + e_{cls}	92.51	87.17	84.28	86.47	89.84	88.05
baseline + e_{img}	92.56	87.4	83.63	86.54	88.63	87.75
CapeNext	92.24	87.58	85.57	87.15	90.13	**88.53**

Table 9: Quantitative results using mask operation. We show the performance of the models when different inputs are used and the query images are masked.

Settings	Split1	Split2	Split3	Split4	Split5	Avg
Ours w/o SuppImg	92.44	87.31	85.44	86.47	90.17	88.37
Ours w/ SuppImg	92.31	87.48	85.59	86.29	90.3	**88.39**

Table 10: PCK@0.2 results on MP-100 when using additional support image.

Methods	Backbone	Params (M)	Time (s)	Avg
CapeX	Swinv2-T	28.35	603	87.61
CapeNext	HRNet-w32	28.54	774	86.30
CapeNext	Vit-Base-16	86.39	719	86.88
CapeNext	DINOv2-ViT-S	22.06	544	87.95
CapeNext	Swinv2-T	28.35	676	88.37

Table 11: PCK@0.2 performance and overhead of different backbones.

Figure 2: **Qualitative results.** We visualize additional joint predictions on MP-100 test dataset.

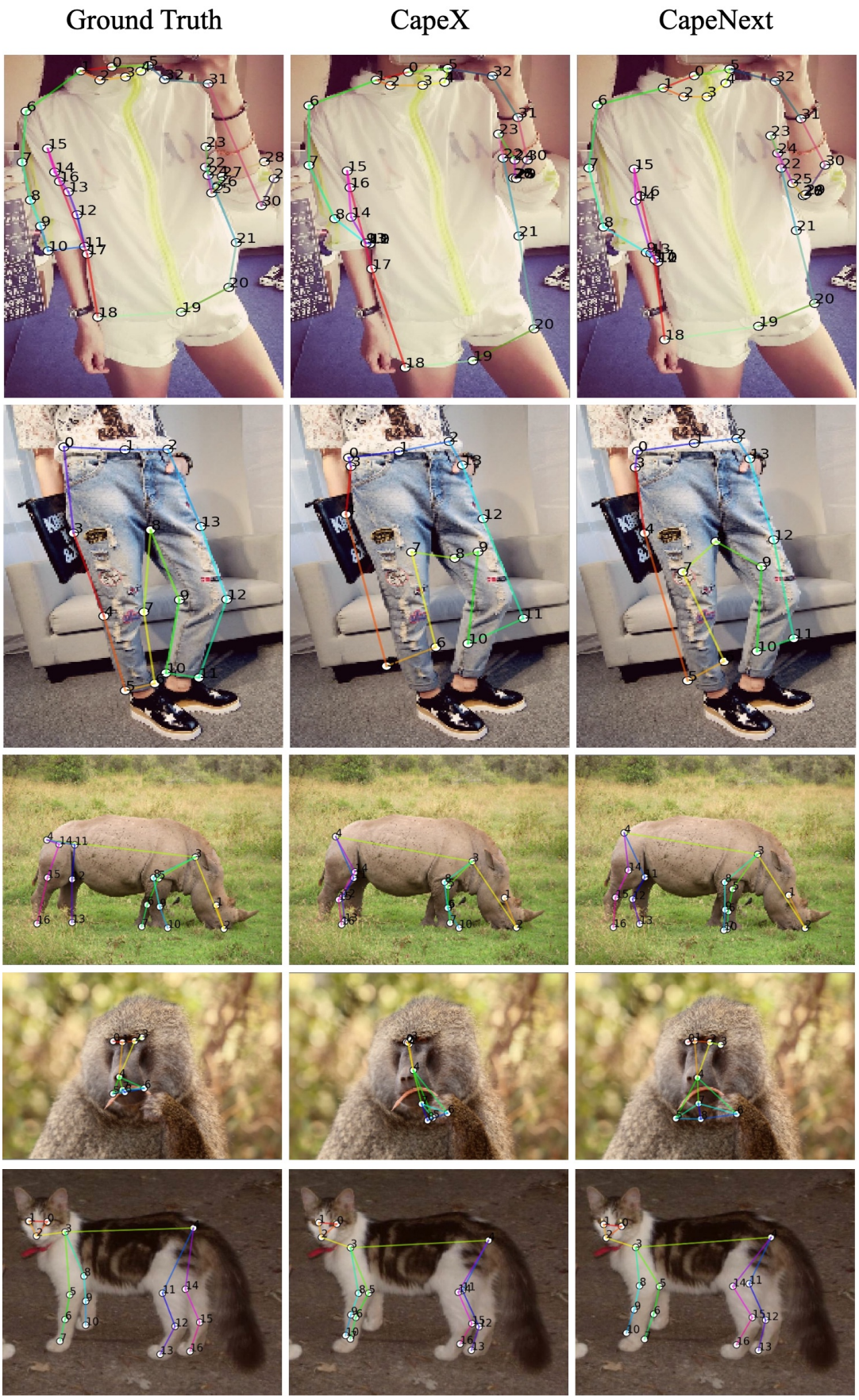

Figure 3: **Failure case.** Although the model sometimes makes inaccurate predictions, it still achieves improvements compared to the baseline.

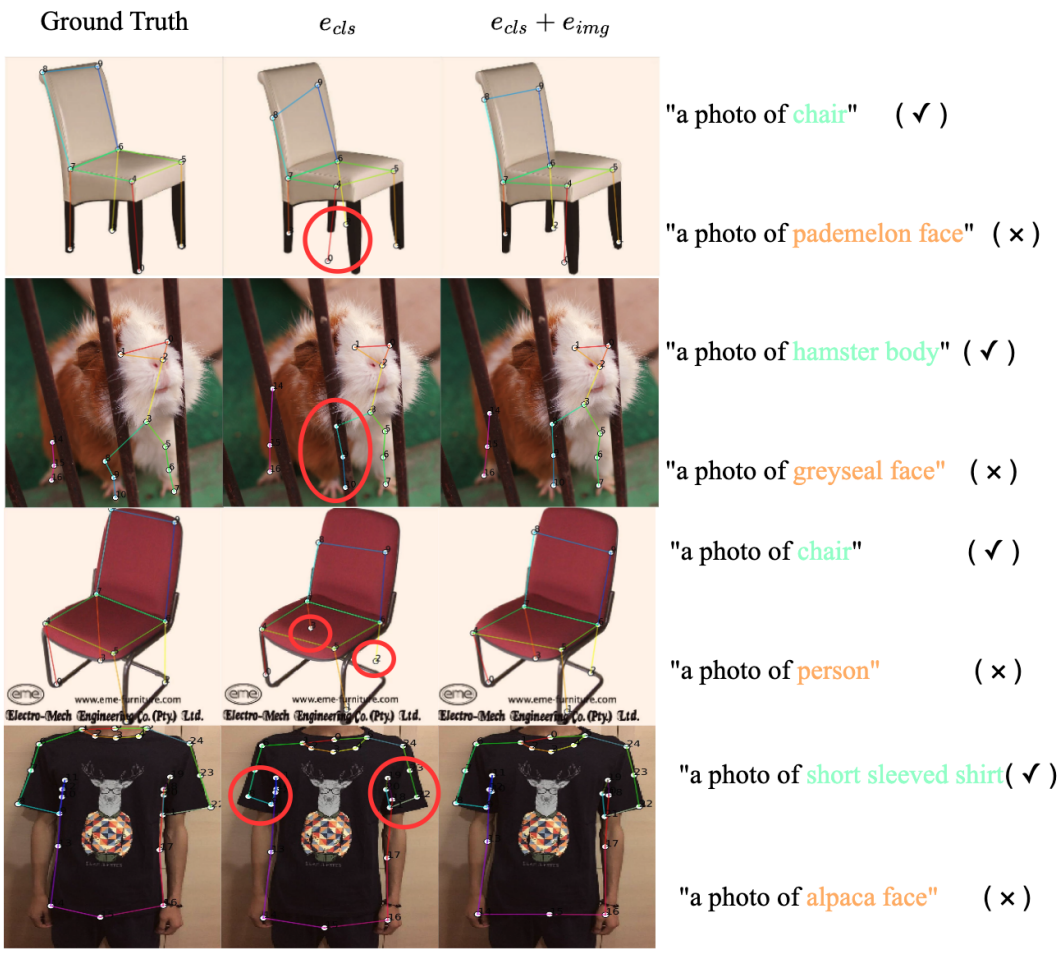

Figure 4: **Qualitative results.** We visualize joint predictions when using incorrect class descriptions on MP-100 test dataset. The second column shows the prediction with only incorrect class descriptions as input, followed by the predictions with multimodal input. Zoom in for more details.