Self-Correctable and Adaptable Inference for Generalizable Human Pose Estimation
Supplemental Material

Zhehan Kan[1], Shuoshuo Chen[1], Ce Zhang[1], Yushun Tang[1], Zhihai He[1,2*]
[1]Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen, China
[2]Pengcheng Laboratory, Shenzhen, China
{kanzh2021,chenss2021,zhangc2019,tangys2022}@mail.sustech.edu.cn, hezh@sustech.edu.cn

In this Supplemental Material, we provide additional experimental results, ablation studies, more implementation details, and further discussion for in-depth understanding of the proposed SCAI method.

1. More Supporting Results for the Proposed SCAI Method

In this section, we provide additional experimental results to validate our proposed ideas of self-correctable and adaptable inference.

1.1. Further Understanding of Self-Correctable Inference

In Figure 1 (a), we compute the L_2 distance between the predicted heatmap by the prediction network $\mathbf{\Phi}$ and its ground-truth. We plot the distribution of distance for all training samples using the solid line.

The FFN (fitness feedback network) aims to evaluate if the corrected prediction of $\tilde{H}_D$ is accurate or not. If the predicted heapmap $\tilde{H}_D$ matches exactly the ground-truth H_D, the corresponding self-referential error should reach to the minimum. In this case, the self-referential feedback error generated by this FFN can successfully guide the correction process towards the ground-truth value. So, to evaluate the effectiveness of the FFN, we compute the L_2 distance $\|\hat{H}_A - H_A^*\|_2$ and plot its distribution using the dashed line in Figure 1(a). We can see that, on the training set, since the distributions of both distances by the prediction network $\mathbf{\Phi}$ and the FFN $\mathbf{\Gamma}$ share the same mean distance from the ground-truth. However, if we plot their distributions on the test set, we can see that the distribution for the FFN remains largely unchanged. However, the distribution of $\mathbf{\Phi}$ moves significantly towards the right with increased deviations from the ground-truth. This indicates that using self-referential feedback error obtained from the FFN as guidance for prediction correction, we can improve the generalization capability and prediction accuracy for the prediction network.

This process is further illustrated in Figure 1(b). The circle represents the ground-truth value of H_D. The square and triangle represent the predicted result and corrected result, respectively. On the training set, since both networks are well trained, they are both close to the ground-truth. However, on the test set, due to the generalization issue, the predicted result (square) deviates more significantly from the ground-truth. So, during the correction and refinement process, the self-referential feedback error can pull and correct the prediction result towards the ground-truth, resulting in improvement for human pose estimation performance.

1.2. Correction Examples of the Prediction Results

In Figure 2, we randomly choose five samples to demonstrate the unique correction ability of our SCAI method. The top row shows the original keypoints heatmaps predicted by the baseline method. The bottom row shows the corrected keypoints heatmaps by our SCAI method. We can see that SCAI can successfully correct keypoints from incorrect initial predictions and further enhance their accuracy. It should be noted that the above correction process can be repeated to further improve its accuracy. For example, Figure 3 shows the correction process of one example keypoint heatmap. When $t = 0$, the original result from the baseline is incorrect due to occlusion. After being corrected by our SCAI method, the heatmap is enhanced and becomes much more accurate.

2. Implementation Details

In this section, we provide more implementation details of our SCAI method.

*Corresponding author.

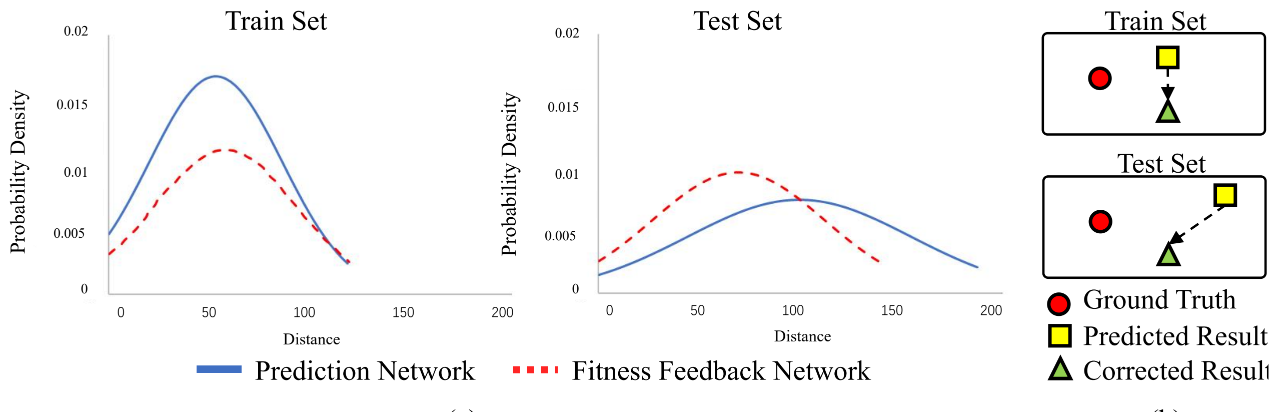


Figure 1. (a) shows average distribution of distance from the prediction to ground-truth. (b) shows a simple example on how self-referential feedback error by FFN reflects the generalization ability from train set to test set.

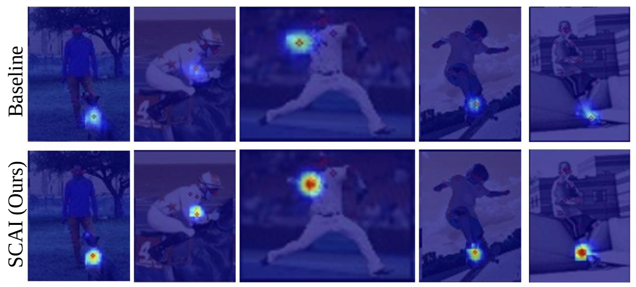


Figure 2. Five examples of refinement of keypoints heatmaps. The top row is the original keypoint heatmap. The bottom row is the keypoint heatmap from SCAI.

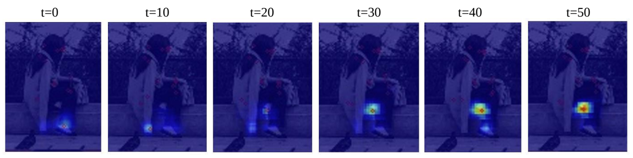


Figure 3. Correction process of keypoint heatmap.

2.1. Experimental Settings

We follow the commonly used standard Object Keypoint Similarity (OKS) in existing work [4] as our evaluation metric:

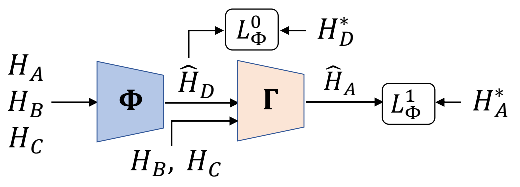
$$OKS = \frac{\sum_i e^{-d_i^2/2s^2k_i^2} \cdot \delta(v_i > 0)}{\sum_i \delta(v_i > 0)}, \quad (1)$$

where d_i denotes the Euclidean distance between the detected keypoint and the corresponding ground-truth, v_i represents the visibility flag of the ground-truth, s is the object scale, and k_i is a constant specific to each keypoint that controls the falloff. The function $\delta(\cdot)$ is 1 if $\cdot$ holds, and 0 otherwise. To evaluate the accuracy of the method, we report standard average precision and recall scores, including AP^{50}, AP^{75}, AP, AP^M, AP^L, AR, AP^{easy}, AP^{med}, AP^{hard} at various overlap thresholds, as defined in [1].

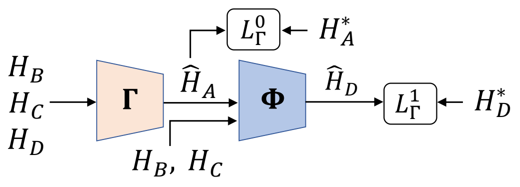
For the prediction network $\mathbf{\Phi}$, correction network $\mathbf{C}$ and FFN $\mathbf{\Gamma}$, we choose the fully convolutional network [3] of 13 layer and 7 layer respectively, during the actual prediction, we added a unit of delay, waiting for the feedback error needed by the correction network. The networks are trained with the Adam optimizer. When training the prediction network $\mathbf{\Phi}$ and the FFN $\mathbf{\Gamma}$, we choose a batch size of 32 and an initial learning rate of 0.001 with a decay rate of 0.97. The whole model is trained for 210 epochs. For the correction network $\mathbf{C}$, we choose a batch size of 64 and an initial learning rate of 0.00041 with a decay rate of 0.97. The whole model is trained for 210 epochs and $\mathbf{m}$ is set to be 120 epochs. The weights a, b and λ in (8) are respectively set as 0.85, 0.65, 0.45. During the inference process, we set the batch size to be 32, the maximum of training epochs to be 20, and an initial learning rate of 0.013 with a decay rate of 0.96. The correction iteration is set to be 20. All the experiments are conducted on two 32GB NVIDIA Tesla V100 GPUs.

2.2. Details on Training the Prediction Network and the Feedback Fitness Network (FFN)

We follow the process proposed in [2] to pre-train our networks. Figure 4 shows the pre-training process of the prediction network $\mathbf{\Phi}$ and the FFN $\mathbf{\Gamma}$. We denote the heatmaps of the proximal keypoints in each group as $\{H_A, H_B, H_C\}$ and the heatmap of the distal keypoint is denoted as H_D. All these heatmaps are generated by the baseline pose estimation network which is the HRNet [4] in our work. [2] shows that the distal keypoints are having much lower estimation accuracy than those proximal keypoints during pose estimation. In this case, as illustrated in Figure 4, the prediction network $\mathbf{\Phi}$ predicts the heatmap of the distal keypoint H_D from the non-distal key-



(a)



(b)

Figure 4. Pre-training of baseline prediction network $\mathbf{\Phi}$ and fitness feedback network $\mathbf{\Gamma}$.

points $\{H_A, H_B, H_C\}$ with feature map $\mathbf{f}$ as the visual context. The fitness feedback network (FFN) $\mathbf{\Gamma}$ shares the same structure as the prediction network. It performs the backward prediction of keypoint H_A from the rest three heatmaps $\{H_B, H_C, H_D\}$.

The prediction network and FFN are jointly trained iteratively during the pre-training process. The pre-training process for the prediction network $\mathbf{\Phi}$ is illustrated in Figure 4 (a), where FFN is fixed. The prediction network $\mathbf{\Phi}$ takes $\{H_A, H_B, H_C\}$ and the visual feature map $\mathbf{f}$ as its inputs with $\hat{H}_D$ as the output, using the distance between the ground truth $\hat{H}_D^*$ as the loss function $\mathcal{L}_{\mathbf{\Phi}}^0 = ||\hat{H}_D - H_D||_2$. Then, the backward prediction $\hat{H}_A$ is the output from the FFN with three inputs: $\hat{H}_D$, H_B, H_C and the visual feature map $\mathbf{f}$, which is compared with the ground-truth heatmap H_A^* to form another loss as $\mathcal{L}_{\mathbf{\Phi}}^1 = ||\hat{H}_A - H_A||_2$. These two losses are combined

$$\mathcal{L}_{\mathbf{\Phi}} = \mathcal{L}_{\mathbf{\Phi}}^0 + \mathcal{L}_{\mathbf{\Phi}}^1 \tag{2}$$

to pre-train the prediction network $\mathbf{\Phi}$. Similarly, when training the FFN, $\mathbf{\Phi}$ is fixed. We first predict $\hat{H}_A$ by the FFN from $\{H_B, H_C, H_D\}$ and feature map $\mathbf{f}$, which is then combined with $\{H_B, H_C\}$ and $\mathbf{f}$, being forwarded to $\mathbf{\Phi}$ to generate the output $\hat{H}_D$. Thus, the overall loss function for the training of FFN is given by

$$\mathcal{L}_{\mathbf{\Gamma}} = ||\hat{H}_A - H_A||_2 + ||\hat{H}_D - H_D||_2. \tag{3}$$

2.3. Training Details for the Correction Network and Feedback Fitness Network (FFN)

The FFN $\mathbf{\Gamma}$ and correction network $\mathbf{C}$ are jointly trained. Specifically, We first train $\mathbf{C}$ and then use the updated outputs $\tilde{H}_D, \tilde{H}_B, \tilde{H}_C$ as the new input to train FFN. After the training is completed, we use the new output $\hat{H}_A$ of FFN as the new input to train $\mathbf{C}$. The entire training process operates in an iterative manner. The reason for iterative training is that there are errors between the inputs H_B, H_C, $\tilde{H}_D$, $\hat{H}_A$ and their ground truth, which lead to the inaccuracy of FFN and $\mathbf{C}$. Through iterative training, the input can be continuously optimized to improve the performance of FFN and $\mathbf{C}$. The reason for not directly training with ground truth is that the FFN and $\mathbf{C}$ obtained in this way can successfully correct prediction errors from incorrect prediction results generated by the baseline estimation model.

As discussed in Section 3.2, similar to the refinement of keypoint X_D, we also develop correction networks for X_B and X_C to optimize heatmaps H_B and H_C respectively. The only difference between them is that the input $\hat{H}_D$ of correction network $\mathbf{C}_D$ is from the prediction network $\mathbf{\Phi}$, while the input H_B, H_C of correction network $\mathbf{C}_B$ and $\mathbf{C}_C$ are estimated by baseline pose estimation model instead of $\mathbf{\Phi}$.

2.4. Iterative Correction of the Prediction Results

The proposed correction can be performed multiple times to further improve the prediction accuracy. During the inference process, we can iteratively correct the keypoint heatmap for multiple times, which means that the correction module not only corrects the prediction results $\{H_B, H_C, \hat{H}_D\}$ by the baseline network and prediction network, but also corrects the results $\{\tilde{H}_B, \tilde{H}_C, \tilde{H}_D\}$ generated by the corresponding correction networks.

This iterative correction can be included during both training and inference. Specifically, considering the training of correction network for H_D as an example, the training process is divided into two stages, and runs for **n** epochs in total. For the first stage of **m** epochs, we keep using heatmaps $\{H_B, H_C, \hat{H}_D\}$ predicted by the baseline network and the prediction network. For the last stage of **n-m** epochs, we iteratively update the input of correction network with the corrected results from the last round. The reason for the two-stage training is that the correction network corrects both the baseline prediction and corrected results from the last round during the inference process, and the two stages of learning will allow the correction network to obtain knowledge of different rounds of correction.

During inference, the correction process runs for multiple iterations. After we have obtained the corrected $\{\tilde{H}_B, \tilde{H}_C, \tilde{H}_D\}$ through the corresponding correction networks, we decide whether to keep the correction result or not by checking if L_2-norm of self-referential error ${E_s}^t$ is smaller than ${E_s}^{t-1}$ or not. We will repeat this process multiple times until $|{E_s}^t - {E_s}^{t-1}|$ becomes smaller than a given threshold σ.

3. Pseudo Code for the SCAI Algorithm

The proposed SCAI algorithm is summarized by the pseudo code in Algorithm 1 and Algorithm 2.

Algorithm 1 Inference Stage

Input: Heatmaps H_A, H_B, H_C pre-predicted from training set, pretrained prediction network $\mathbf{\Phi_{pretrain}}$, trained FFN $\mathbf{\Gamma_{train}}$, trained correction network model $\mathbf{C_{train}}$
Output: Heatmaps $\tilde{H}_D$

```
1:  Initialization: $\mathbf{\Phi} \leftarrow \mathbf{\Phi_{pretrain}}$, $\mathbf{\Gamma} \leftarrow \mathbf{\Gamma_{train}}$
2:  for each test batch $i$ do
3:      // Self-Adaptable Inference
4:      $\mathbf{C}_i \leftarrow \mathbf{C_{train}}$
5:      $\hat{H}_D = \mathbf{\Phi}(H_A, H_B, H_C)$
6:      $\tilde{H}_D = \mathbf{C}_i(\hat{H}_D, \mathbf{e_s})$
7:      $\hat{H}_A = \mathbf{\Gamma}(H_B, H_C, \tilde{H}_D)$.
8:      $\mathbf{C}_i^*$ by updating $\mathbf{C}_i$ with $E_s = \|\hat{H}_A - H_A\|_2$
9:      // Inference
10:     $t = 0$
11:     $\tilde{H}_D^0 = \mathbf{C}_i^*(\hat{H}_D, \mathbf{e_s})$, with $\mathbf{e_s} = H_A - \hat{H}_A$
12:     while $\|E_s^{t+1} - E_s^t\| > \epsilon$ do
13:         $\tilde{H}_D^{t+1} = \mathbf{C}_i^*(\tilde{H}_D^t, \mathbf{e_s})$, with $\mathbf{e_s} = H_A - \hat{H}_A$
14:         $t = t + 1$
15:     end while
16: end for
17: return $\tilde{H}_D \leftarrow \tilde{H}_D^t$
```

Algorithm 2 Training Stage

Input: Heatmaps H_A, H_B, H_C pre-predicted from training set, feature maps $\mathbf{f}$, pretrained prediction network $\mathbf{\Phi_{pretrain}}$, pre-trained fitness feedback network model FFN $\mathbf{\Gamma_{pretrain}}$
Output: FFN model $\mathbf{\Gamma}$, correction network model $\mathbf{C}$

```
1:  Initialization: $\mathbf{\Phi} \leftarrow \mathbf{\Phi_{pretrain}}$, $\mathbf{\Gamma_0} \leftarrow \mathbf{\Gamma_{pretrain}}$
2:  for $e$ in epochs $n$ do
3:      if $e < m$ then
4:          for each training batch $i$ do
5:              $\hat{H}_D = \mathbf{\Phi}(H_A, H_B, H_C)$
6:              $\bar{H}_A = \mathbf{\Gamma_{pretrain}}(H_B, H_C, \hat{H}_D)$.
7:              $\tilde{H}_D = \mathbf{C_i}(\hat{H}_D, \mathbf{e_s})$, with $\mathbf{e_s} = H_A - \bar{H}_A$
8:              $\mathbf{C_{i+1}} \leftarrow \mathbf{C_i} - \eta \bigtriangledown \mathcal{L}_C(\tilde{H}_D, \bar{H}_A; H_D^*, H_A^*)$ by
                (8).
9:              $\tilde{H}_D = \mathbf{C_{i+1}}(\hat{H}_D, \mathbf{e_s})$, with $\mathbf{e_s} = H_A - \bar{H}_A$
10:             $\hat{H}_A = \mathbf{\Gamma_i}(H_B, H_C, \tilde{H}_D)$.
11:             $\mathbf{\Gamma_{i+1}} \leftarrow \mathbf{\Gamma_i} - \eta \bigtriangledown \mathcal{L}_\Gamma(\hat{H}_A; H_A^*)$
12:         end for
13:     else if $e \geq m$ then
14:         for each training batch $i$ do
15:             $\hat{H}_D = \mathbf{\Phi}(H_A, H_B, H_C)$
16:             $\hat{H}_A = \mathbf{\Gamma_{i+m}}(H_B, H_C, \tilde{H}_D)$.
17:             $\bar{H}_A = \mathbf{\Gamma_{i+m}}(H_B, H_C, \hat{H}_D)$.
18:             $\tilde{H}_D = \mathbf{C_{i+m}}(\hat{H}_D, \mathbf{e_s})$ with $\mathbf{e_s} = H_A - \hat{H}_A$
19:             $\mathbf{C_{i+m+1}} \leftarrow \mathbf{C_{i+m}} - \eta \bigtriangledown$
                $\mathcal{L}_C(\tilde{H}_D, \hat{H}_A, \bar{H}_A; H_D^*, H_A^*)$
20:             $\tilde{H}_D = \mathbf{C_{i+m+1}}(\hat{H}_D, \mathbf{e_s})$, with $\mathbf{e_s} = H_A - \hat{H}_A$

21:             $\hat{H}_A = \mathbf{\Gamma_i}(H_B, H_C, \tilde{H}_D)$.
22:             $\mathbf{\Gamma_{i+m+1}} \leftarrow \mathbf{\Gamma_{i+m}} - \eta \bigtriangledown \mathcal{L}_\Gamma(\hat{H}_A; H_A^*)$
23:         end for
24:     end if
25: end for
26: $\mathbf{\Gamma_{train}} \leftarrow \mathbf{\Gamma_{i+m+1}}$, $\mathbf{C_{train}} \leftarrow \mathbf{C_{i+m+1}}$
27: return $\mathbf{\Gamma_{train}}$, $\mathbf{C_{train}}$
```

References

[1] Zigang Geng, Ke Sun, Bin Xiao, Zhaoxiang Zhang, and Jingdong Wang. Bottom-up human pose estimation via disentangled keypoint regression. In *CVPR*, pages 14676–14686, 2021. 2

[2] Zhehan Kan, Shuoshuo Chen, Zeng Li, and Zhihai He. Self-constrained inference optimization on structural groups for human pose estimation. In *ECCV*, volume 13665, pages 729–745. Springer, 2022. 2

[3] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 2

[4] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, pages 5693–5703, 2019. 2