

# PoseFix: Model-agnostic General Human Pose Refinement Network

Gyeongsik Moon  
 Department of ECE, ASRI  
 Seoul National University  
 mks0601@snu.ac.kr

Ju Yong Chang  
 Department of EI  
 Kwangwoon University  
 juyong.chang@gmail.com

Kyoung Mu Lee  
 Department of ECE, ASRI  
 Seoul National University  
 kyoungmu@snu.ac.kr

## Abstract

*Multi-person pose estimation from a 2D image is an essential technique for human behavior understanding. In this paper, we propose a human pose refinement network that estimates a refined pose from a tuple of an input image and input pose. The pose refinement was performed mainly through an end-to-end trainable multi-stage architecture in previous methods. However, they are highly dependent on pose estimation models and require careful model design. By contrast, we propose a model-agnostic pose refinement method. According to a recent study, state-of-the-art 2D human pose estimation methods have similar error distributions. We use this error statistics as prior information to generate synthetic poses and use the synthesized poses to train our model. In the testing stage, pose estimation results of any other methods can be input to the proposed method. Moreover, the proposed model does not require code or knowledge about other methods, which allows it to be easily used in the post-processing step. We show that the proposed approach achieves better performance than the conventional multi-stage refinement models and consistently improves the performance of various state-of-the-art pose estimation methods on the commonly used benchmark. The code is available in this [https URL](https://github.com/mks0601/PoseFix_RELEASE)<sup>1</sup>.*

## 1. Introduction

The goal of human pose estimation is to localize semantic keypoints of a human body. It is an essential technique for human behavior understanding and human-computer interaction. Recently, many methods [5, 7, 11, 14, 15, 18, 20, 22, 23, 26, 30] utilize deep convolutional neural networks (CNNs) and achieved noticeable performance improvement. They are also updating performance limits in annual competitions for 2D human keypoint detection such as MS COCO keypoint detection challenge [19].

In this paper, we propose a human pose refinement net-

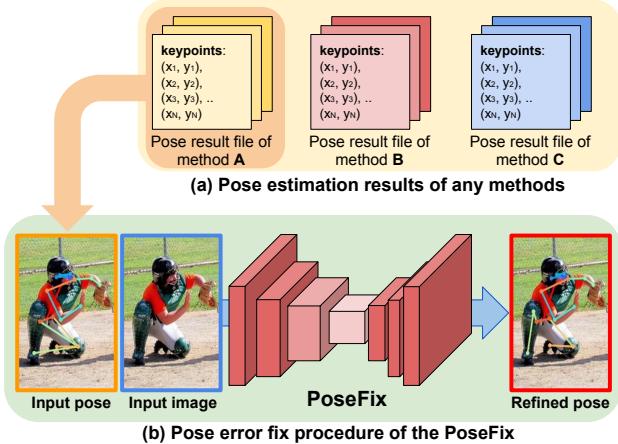


Figure 1: Testing pipeline of the PoseFix. It takes pose estimation results of any other method with an input image and outputs a refined pose. Note that the PoseFix does not require any code or knowledge about other methods.

work that estimates a refined pose from a tuple of an input image and a pose. Conventionally, the pose refinement has been mainly performed by multi-stage architectures [4, 7, 21, 29]. In other words, the initial pose and image features generated in the first stage go through subsequent stages, and each stage outputs a refined pose. These multi-stage architectures are usually trained in an end-to-end manner. However, the conventional multi-stage architecture-based refinement approach is highly dependent on the pose estimation model and requires careful design for successful refinement. By contrast, in this work, we propose a model-agnostic pose refinement method that does not depend on the pose estimation model.

Recent research by Ronchi *et al.* [24] gave us a clue on how to design a general model-agnostic pose refiner. They analyzed the results of the MS COCO 2016 keypoint detection challenge winners [5, 22] by using new pose estimation evaluation metrics, *i.e.*, keypoint similarity (KS) and object keypoint similarity (OKS). They taxonomized pose estimation errors into several types such as *jitter*, *inversion*, *swap*,

<sup>1</sup>[https://github.com/mks0601/PoseFix\\_RELEASE](https://github.com/mks0601/PoseFix_RELEASE)

and *miss* and described how frequently these errors occur and how much they can negatively affect performance. Although the winners [5, 22] used very different approaches, their pose error distributions are very similar, which indicates that common issues exist for more accurate pose estimation.

Our basic idea is to use this error statistics as prior information to generate synthetic poses and use the synthesized poses to train the proposed pose refinement model (PoseFix). To train our model, we generate each type of the errors (*i.e.*, jitter, inversion, swap, and miss) based on the pose error distributions from Ronchi *et al.* [24], and construct diverse and realistic poses. The generated input pose is fed to the PoseFix with the input image, and the PoseFix learns to refine the pose. We design our PoseFix as a single-stage architecture with a coarse-to-fine estimation pipeline. It takes the input pose in a coarse form and estimates the refined pose in a finer form. The coarse input pose enables the proposed model to focus not only on an exact location of the input pose but also around it, allowing our model to fix the error of the input pose. Furthermore, the finer form of the output pose enables the proposed model to localize the location of the pose more exactly compared to existing methods. After training, our PoseFix can be applied to and refine the pose estimation results of any single- or multi-person pose estimation method. Figure 1 shows such a pose refinement pipeline of the proposed PoseFix.

Our contributions can be summarized as follows.

- We show that model-agnostic general pose refinement is possible. The PoseFix is trained independently of the pose estimation model. Instead, it is based on error statistics obtained through empirical analysis.
- Our PoseFix can take the pose estimation result of any pose detection method as the input. As the PoseFix does not require any code or knowledge about other methods, our model has very high flexibility and accessibility.
- We design the PoseFix as a coarse-to-fine estimation system. We empirically observed that this coarse-to-fine pipeline is crucial for successful pose refinement.
- Our PoseFix achieves a better result than the conventional multi-stage architecture-based refinement methods. Also, the PoseFix consistently improves the performance of various state-of-the-art pose estimation methods on the commonly used benchmark.

## 2. Related works

**Single-person pose estimation.** Toshev *et al.* [28] directly estimated the Cartesian coordinates of body joints by using a multi-stage deep network and achieved state-of-the-art performance. Tompson *et al.* [27] jointly trained

a CNN and a graphical model. The CNN estimated 2D heatmaps for each joint, and they were used as the unary term for the graphical model. Liu *et al.* [29] used multi-stage CNN which progressively enlarges receptive fields and refines the pose estimation result. Newell *et al.* [21] proposed a stacked hourglass network which repeats down-sampling and upsampling to exploit multi-scale information effectively. Carreria *et al.* [6] proposed an iterative error feedback-based human pose estimation system. Chu *et al.* [8] enhanced the stacked hourglass network [21] by integrating it with a multi-context attention mechanism. Ke *et al.* [16] proposed a multi-scale structure-aware network which achieved leading position in the publicly available human pose estimation benchmark [3].

**Multi-person pose estimation.** There are two main approaches in the multi-person pose estimation. The first one, top-down approach, relies on a human detector that predicts bounding boxes of humans. The detected human image is cropped and fed to the pose estimation network. The second one, bottom-up approach, localizes all human body keypoints in an input image and assembles them using proposed clustering algorithms in each work.

[7, 11, 14, 22, 26, 30] are based on the top-down approach. He *et al.* [11] proposed Mask R-CNN that can perform human detection and keypoint localization in a single model. Instead of cropping the detected humans in the input image, it crops human features from a feature map via the differentiable RoIAlign layer. Chen *et al.* [7] proposed a cascaded pyramid network (CPN) which consists of two networks. The first one, GlobalNet, is based on deep backbone network and upsampling layers with skip connections. The second one, RefineNet, is built to refine the estimation results from the GlobalNet by focusing on hard keypoints. Xiao *et al.* [30] used a simple pose estimation network that consists of a deep backbone network and several upsampling layers. Although it is based on a simple network architecture, it achieved state-of-the-art performance on the commonly used benchmark [19].

[5, 15, 18, 20, 23] are based on the bottom-up approach. DeepCut [23] assigned the detected keypoints to each person in an image by formulating the assignment problem as an integer linear program. DeeperCut [23] improves the DeepCut [23] by introducing image-conditioned pair-wise terms. Cao *et al.* [5] proposed part affinity fields (PAFs) that directly expose the association between human body keypoints. They assembled the localized keypoints of all persons in the input image by using the estimated PAFs. Newell *et al.* [20] introduced a pixel-wise tag value to assign localized keypoints to a certain human. Kocabas *et al.* [18] proposed a pose residual network to assign detected keypoints to each person. Their model can jointly handle person detection, keypoint detection, and person segmentation.

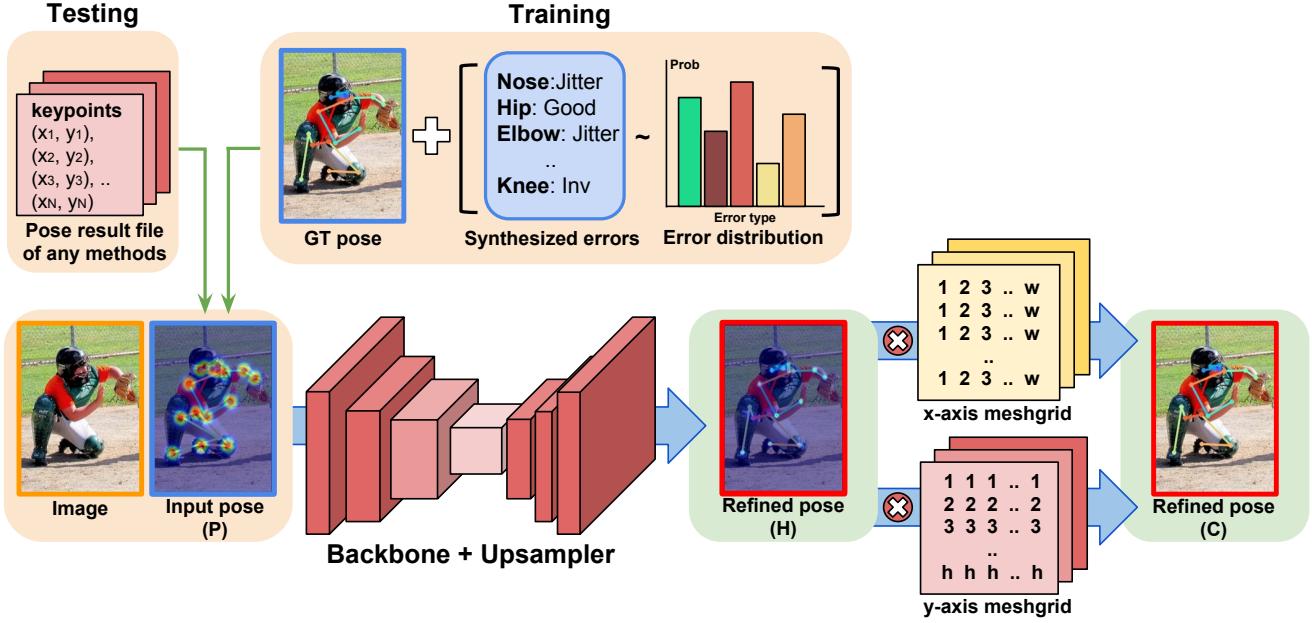


Figure 2: Overall pipeline of the PoseFix. In the training stage, the input pose is generated by synthesizing the pose errors based on the real pose error distributions on the groundtruth pose. In the testing stage, pose estimation results of any other methods become the input pose. The heatmaps are visualized by performing max pooling along the channel axis.

**Human pose refinement.** Many methods attempted to refine the estimated keypoint for more accurate performance. Newell *et al.* [21], Bulat and Tzimiropoulos [4], Liu *et al.* [29], and Chen *et al.* [7] utilized an end-to-end trainable multi-stage architecture-based network. Each stage tries to refine the pose estimation results of the previous stage via end-to-end learning. Carreria *et al.* [6] iteratively estimated error feedback from a shared weight model. The output error feedback of the previous iteration is transformed into the input pose of the next iteration, which is repeated several times for progressive pose refinement. All of these methods combine pose estimation and refinement into a single model, and each refinement module is dependent on estimation. Therefore, the refinement modules have different structures, and they are not guaranteed to work successfully when they are combined with other estimation methods. On the other hand, our pose refinement method is independent of the estimation, and therefore the results can be consistently improved regardless of the prior pose estimation method.

Recently, Fieraru *et al.* [10] proposed a post-processing network to refine the pose estimation results of other methods, which is conceptually similar to ours. They synthesized pose for training and employed simple network architecture that estimates refined heatmaps and offset vectors for each joint. While their method follows ad-hoc rules to generate input pose, our method is based on actual error statistics obtained through empirical analysis. Also, our network with coarse-to-fine structure achieves a much

stronger refinement performance than their simpler one.

### 3. Overview of the proposed model

The goal of the PoseFix is to refine the input 2D coordinates of the human body keypoints of all persons in an input image. To address this problem, our system is constructed based on the top-down pipeline which processes a tuple of a cropped human image and a given pose estimation result of that human instead of processing an entire image including multiple persons. In the training stage, the input pose is synthesized on the groundtruth pose realistically and diversely. In the testing stage, pose estimation results of any other methods can be the input pose to our system. The overall pipeline of the PoseFix is illustrated in Figure 2.

### 4. Synthesizing poses for training

To train the PoseFix, we generate synthesized poses using the groundtruth poses. As the PoseFix should cover different pose estimation results from various methods in the testing stage, synthesized poses need to be diverse and realistic. To satisfy these properties, we generate synthesized poses randomly based on the error distributions of real poses as described in [24]. The distributions include the frequency of each pose error (*i.e.*, jitter, inversion, swap, and miss) according to the joint type, number of visible keypoints, and overlap in the input image. There may also be joints that do not have any error, which should be synthesized very close to the groundtruth to simulate correct esti-

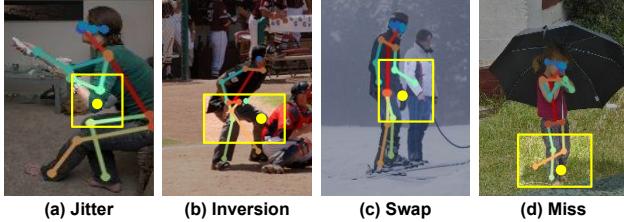


Figure 3: Visualization of synthesized pose errors for each type. The keypoint with pose error is highlighted by a yellow rectangle, and the groundtruth keypoints are drawn in a yellow circle.

mations. Ronchi *et al.* [24] called this status *good*. Considering most of the empirical distributions in [24], we compute the probability that each joint will have one of the pose errors or be in the good status.

The detailed error synthesis procedure on each groundtruth keypoint  $\theta_j^p$  of joint  $j$  which belongs to a person  $p$  is described in below. For more clear description, we define  $j'$  as a left/right inverted joint from the  $j$ , and  $p'$  as a different person from the  $p$  in the input image. Also,  $d_j^k$  is defined as a  $L_2$  distance that makes KS with the groundtruth keypoint becomes  $k$  for joint  $j$ . Note that  $d_j^k$  depends on the type of joint  $j$  because the error distribution of each joint has different scale [24]. For example, eyes require more precise localization than hips to obtain the same KS  $k$ . Figure 3 visualizes examples of synthesized pose errors of each type.

**Good.** Good status is defined as a very small displacement from the groundtruth keypoint. An offset vector whose angle and length are uniformly sampled from  $[0, 2\pi]$  and  $[0, \theta_j^{0.85}]$ , respectively, is added to the groundtruth  $\theta_j^p$ . The synthesized keypoint position should be closer to the original groundtruth  $\theta_j^p$  than  $\theta_{j'}^p$ ,  $\theta_j^{p'}$ , and  $\theta_{j'}^{p'}$ .

**Jitter.** Jitter error is defined as a small displacement from the groundtruth keypoint. An offset vector whose angle and length are uniformly sampled from  $[0, 2\pi]$  and  $[d_j^{0.85}, d_j^{0.5}]$ , respectively, is added to the groundtruth  $\theta_j^p$ . Similar to the *good* status, the synthesized keypoint position should be closer to the original groundtruth  $\theta_j^p$  than  $\theta_{j'}^p$ ,  $\theta_j^{p'}$ , and  $\theta_{j'}^{p'}$ .

**Inversion.** Inversion error occurs when a pose estimation model is confused between semantically similar parts that belong to the same instance. We restrict the inversion error to the left/right body part confusion following [24]. The *jitter* error is added to  $\theta_{j'}^p$ . The synthesized keypoint position should be closer to the  $\theta_{j'}^p$  than  $\theta_j^p$ ,  $\theta_j^{p'}$ , and  $\theta_{j'}^{p'}$ .

**Swap.** Swap error represents a confusion between the same or similar parts which belong to different persons. The *jitter* is added to  $\theta_j^{p'}$  or  $\theta_{j'}^{p'}$ . The closest keypoint from the synthesized keypoint should be  $\theta_j^{p'}$  or  $\theta_{j'}^{p'}$ , not any of  $\theta_j^p$  and  $\theta_{j'}^{p'}$ .

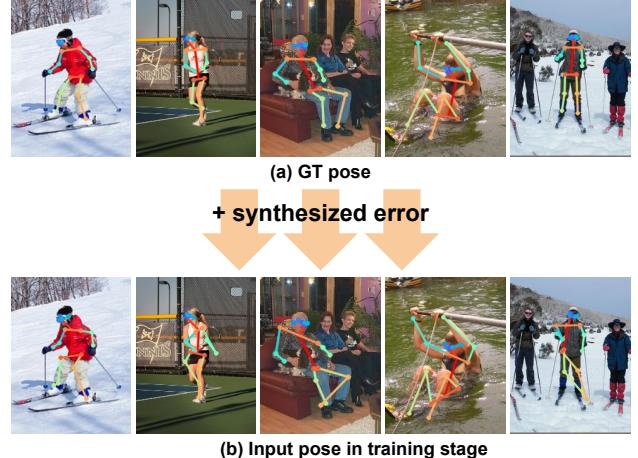


Figure 4: Visualization of the groundtruths and synthesized input poses. The synthesized poses are generated by adding errors to the groundtruth poses, which are used for training PoseFix.

**Miss.** Miss error represents a large displacement from the groundtruth keypoint position. An offset vector whose angle and length are uniformly sampled from  $[0, 2\pi]$  and  $[d_j^{0.5}, d_j^{0.1}]$ , respectively, is added to one of  $\theta_j^p$ ,  $\theta_{j'}^p$ ,  $\theta_j^{p'}$ , and  $\theta_{j'}^{p'}$ . The synthesized keypoint position should be at least  $d_j^{0.5}$  away from all of  $\theta_j^p$ ,  $\theta_{j'}^p$ ,  $\theta_j^{p'}$ , and  $\theta_{j'}^{p'}$ .

Some examples of synthesized input poses are shown in Figure 4.

## 5. Architecture and learning of PoseFix

### 5.1. Model design

We design the PoseFix to directly estimate a refined pose from a tuple of an input image and an input pose as shown in Figure 2. The input image and the input pose provide contextual and structured information to the PoseFix, respectively, and the PoseFix learns to use these information to fix pose errors in the input pose. Although some errors exist in the input pose, it still provides useful structured information because, as indicated by Ronchi *et al.* [24], most keypoints in the input pose are in *good* status or have *jitter* error which represent a small displacement from the groundtruth pose. This rough structured information acts like attention which tells the PoseFix where to focus on at the human body.

We observed that by learning to fix the pose errors in the input pose, the PoseFix learns where to focus on at the human body as in Figure 5. As it shows, although some errors exist in the input pose, the PoseFix initially focuses well on the reliable keypoint locations of the input pose. And then, it successfully localizes correct keypoints without being influenced by the errors of the input pose.

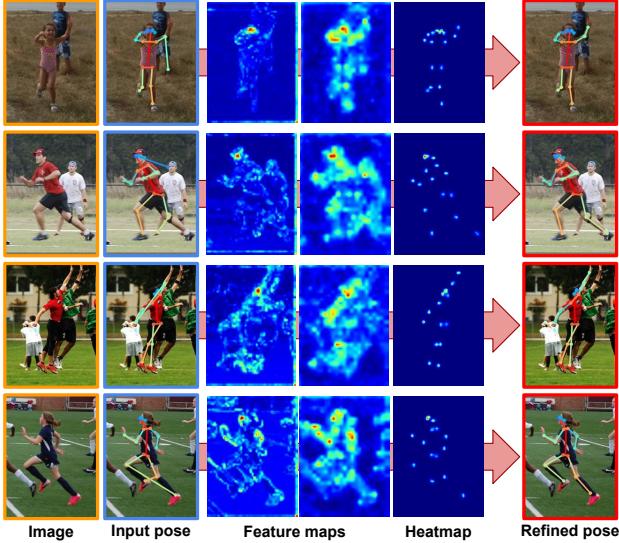


Figure 5: Visualization of feature maps and final heatmaps of the PoseFix. The feature maps and heatmaps are reduced into one channel by max pooling along the channel axis for visualization. The order of the feature maps and heatmaps in the figure is the same with that of the feedforward.

## 5.2. Coarse-to-fine estimation

To make it more robust to errors, we design the proposed PoseFix to operate in a coarse-to-fine manner. We use the terms “coarse” and “fine” by the degree of uncertainty in representing the pose. For example, in representing the position of each joint constituting a pose, a Gaussian blob has a high uncertainty as much as the size of its standard deviation. On the other hand, a one-hot vector has relatively low uncertainty up to the size of a quantized grid. The coordinates of a keypoint has the least amount of uncertainty because it provides the exact information about the location itself. Therefore, in our work, the coarse-to-fine estimation implies that the coarse input pose ( $P = \{P_n\}_{n=1}^N$ ) represented by the set of Gaussian blobs is fed to the network, producing the finer pose in the form of the one-hot vector ( $H = \{H_n\}_{n=1}^N$ ), and then the finest pose in terms of the keypoint coordinates ( $C = \{C_n\}_{n=1}^N$ ) is generated as the final output as illustrated in Figure 2.  $N$  denotes the number of keypoints. In this subsection, we describe this coarse-to-fine estimation in more detail.

The input pose is constructed in a coarse form by a single-mode Gaussian heatmap representation as follows:

$$P_n(i, j) = \exp \left( -\frac{(i - i_n)^2 + (j - j_n)^2}{2\sigma^2} \right), \quad (1)$$

where  $P_n$  and  $(i_n, j_n)$  are the input heatmap and 2D coordinates of  $n$ th keypoint, respectively, and  $\sigma$  is the standard deviation of the Gaussian peak. The generated input pose is concatenated with the input image and fed into the PoseFix.

This Gaussian heatmap representation is suitable for subsequent convolutional operations because it is pixel-wise aligned with the input image. Moreover, as the input pose can contain some errors, non-zero values around the center of the blob can be used to encourage the PoseFix to focus not only on the exact location of the input pose, but also around it.

From the input Gaussian heatmap in a coarse form, the proposed network generates the heatmap  $H_n$  and the keypoint coordinates  $C_n$  for the  $n$ th keypoint, sequentially. To make  $H_n$  a finer form, we supervise it using a one-hot vector. Then, soft-argmax operation [26] is applied to  $H_n$  to generate  $C_n$  in a differentiable manner. Soft-argmax is defined as the element-wise product between the input heatmap and the meshgrid followed by the summation, as shown in Figure 2. More precisely, the 2D coordinates are calculated from  $H_n$  as follows:

$$C_n = \left( \sum_{i=1}^w \sum_{j=1}^h i H_n(i, j), \sum_{i=1}^w \sum_{j=1}^h j H_n(i, j) \right)^T, \quad (2)$$

where  $w$  and  $h$  are the width and height of  $H_n$ , respectively. Our network is trained by minimizing the cross-entropy-based integral loss [26], which is defined as follows:

$$L = L_H + L_C, \quad (3)$$

where  $L$  is the cross-entropy-based integral loss, and two losses  $L_H$  and  $L_C$  are described below.

The  $L_H$  is a cross-entropy loss which is calculated after applying the softmax function to the output heatmap along the spatial axis. The definition of the  $L_H$  is as follows:

$$L_H = -\frac{1}{N} \sum_{n=1}^N \sum_{i,j} H_n^*(i, j) \log H_n(i, j), \quad (4)$$

where  $H_n^*$  and  $H_n$  are the groundtruth and estimated heatmaps with softmax applied, respectively. The groundtruth heatmap  $H_n^*$  is a one-hot vector if the groundtruth keypoint coordinates are integers. Otherwise, two grids for each  $x$  and  $y$  axis are selected by floor and ceil operations and are filled with probabilities by linear extrapolation. The  $L_C$  is the sum of all  $L_1$  losses applied to the coordinates as follows:

$$L_C = \frac{1}{N} \sum_{n=1}^N \|C_n^* - C_n\|_1, \quad (5)$$

where  $C_n^*$  is the groundtruth coordinates vector for  $n$ th keypoint. The  $L_H$  forces the PoseFix to select a single grid point in the estimated heatmap, and the  $L_C$  enables the PoseFix to localize keypoints more precisely because it is calculated in the continuous space which is free from quantization errors.

### 5.3. Network architecture

We used network architecture of Xiao *et al.* [30] which consists of a deep backbone network (*i.e.*, ResNet [13]) and several upsampling layers. The final upsampling layer becomes heatmaps ( $H$ ) after applying the softmax function. The soft-argmax operation extracts coordinates ( $C$ ) from the heatmaps ( $H$ ), and it becomes the final estimation of the PoseFix.

## 6. Implementation details

**Training.** The proposed PoseFix is trained in an end-to-end manner. The weights of the backbone part are initialized with the publicly released ResNet model pre-trained on the ImageNet dataset [25], and the weights of the remaining part are initialized from the zero-mean Gaussian distribution with  $\sigma = 0.01$  and as in He *et al.* [12]. The weights are updated by Adam optimizer [17] with a mini-batch size of 128. The initial learning rate is set to  $5 \times 10^{-4}$  and reduced by a factor of 10 at 90 and 120th epoch. We perform data augmentation including scaling ( $\pm 30\%$ ), rotation ( $\pm 40^\circ$ ), and flip. To crop humans from an input image, groundtruth human bounding boxes are extended to a fixed aspect ratio (*i.e.*, height:width = 4:3) and then cropped without distorting the aspect ratio. The cropped bounding box is resized to a fixed size, which becomes the input image. We train the PoseFix 140 epochs with four NVIDIA 1080 Ti GPUs, which took two days.

**Testing.** In the testing stage, the pose estimation result of other pose estimation methods becomes the input pose. To crop human bounding box from an image with multiple persons, we calculate bounding box coordinates from the keypoints coordinates of the input pose. Following [7, 21], we used testing time flip augmentation.

Our model is implemented using TensorFlow [1] deep learning framework.

## 7. Experiment

### 7.1. Dataset and evaluation metric

The proposed PoseFix is trained and tested on the MS COCO [19] 2017 keypoint detection dataset, which consists of training, validation, and test-dev sets. The training set includes 57K images and 150K person instances. The validation set and the test-dev sets include 5K and 20K images, respectively. The OKS-based AP metric is used to evaluate the accuracy of the keypoint localization.

### 7.2. Ablation study

To validate each component of the PoseFix, we tested the PoseFix on the validation set. The backbone of all the models are ResNet-50, and the size of the input image is

Methods	$AP$	$AP_{.50}$	$AP_{.75}$	$AP_M$	$AP_L$
E2E-refine	70.1 (+0.4)	87.3 (-1.0)	76.8 (-0.2)	66.8 (+0.6)	76.3 (+0.2)
<b>MA-refine (Ours)</b>	<b>72.1 (+2.4)</b>	<b>88.5 (+0.2)</b>	<b>78.3 (+1.3)</b>	<b>68.6 (+2.4)</b>	<b>78.2 (+2.1)</b>

Table 1: AP comparison between the conventional end-to-end trainable multi-stage refinement model (E2E-refine) and the proposed model-agnostic refinement model (MA-refine) on the validation set. The number in the parenthesis denotes the AP change from the input pose (*i.e.*, CPN).

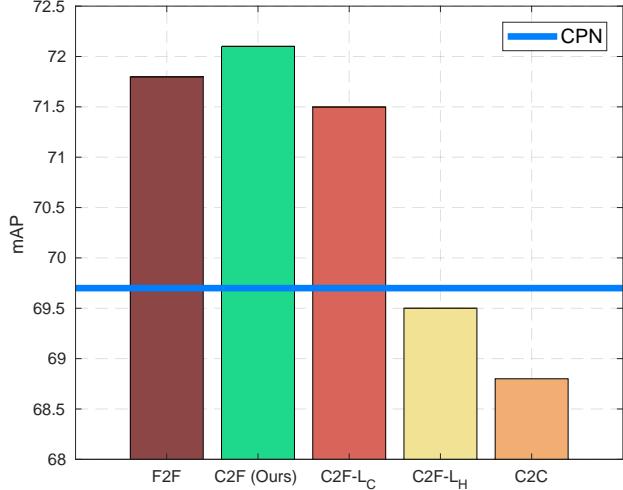


Figure 6: mAP comparison of various pipelines. The mAP is calculated on the validation set.

set to  $256 \times 192$ . We used the CPN [7] which is a state-of-the-art human pose estimation method to generate the input poses.

**Model-agnostic pose refinement.** We compared the accuracy of the conventional end-to-end trainable multi-stage architecture-based pose refinement model (E2E-refine) and the proposed model-agnostic refinement model (MA-refine) in Table 1. To train the E2E-refine, we added a refinement module which has the same network architecture as the PoseFix at the end part of the pre-trained CPN. And then, we fine-tuned it by additionally giving the cross-entropy-based integral loss to the added module in an end-to-end manner. Both the input image and the output pose of the CPN are fed into the refinement module similarly to the PoseFix. We used a pre-trained CPN instead of training it from scratch because fine-tuning the pre-trained model yielded better performance.

As Table 1 shows, the MA-refine trained in a model-agnostic manner improves the accuracy greatly more than the conventional refinement model does. We believe that this is because the added refinement module can be easily overfitted to the output pose of the CPN when training the E2E-refine. In contrast, various input poses that are realistic

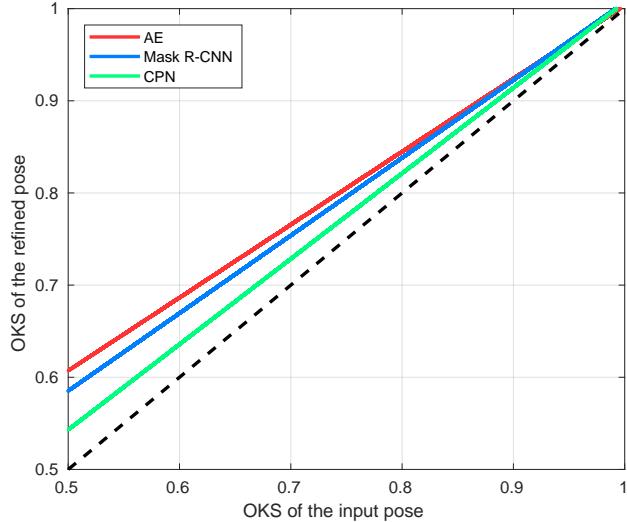


Figure 7: OKS change when the PoseFix is applied to state-of-the-art methods. The dotted line denotes identity function. OKS is calculated on the validation set.

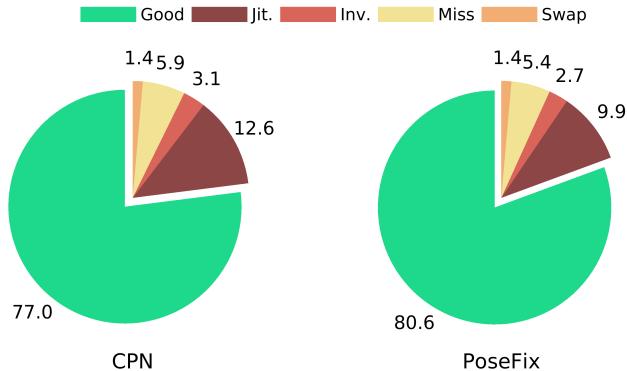


Figure 8: Frequency of each error type change when the PoseFix is applied to the CPN. The frequency is calculated on the validation set.

cially synthesized in the training stage of the PoseFix lead to the effect of data augmentation, which makes the PoseFix more robust to unseen input poses in the testing stage.

Instead of using the same network architecture of the PoseFix like in the E2E-refine, one can design their own refinement module. However, this approach requires careful network design because the amount of GPU memory available at one time is limited. By contrast, since the PoseFix is a decoupled model in both of the training and testing stages, it can serve as an add-on module, and thus provides more flexibility when building pose estimation models.

This analysis clearly demonstrates the benefits of using the model-agnostic pose refinement model compared with the conventional end-to-end trainable multi-stage architecture-based ones.

**Coarse-to-fine estimation.** To demonstrate the validity

of the coarse-to-fine estimation, we compared the performance of fine-to-fine (*i.e.*, F2F), coarse-to-fine (*i.e.*, C2F, ours), and coarse-to-coarse (*i.e.*, C2C) estimation pipelines in Figure 6. As described in Section 5.2, the Gaussian heatmap and one-hot vector are used as coarse and fine forms of the input pose, respectively. To estimate the refined pose in a coarse form, the model learns to estimate the Gaussian heatmap by minimizing mean square error following [7, 21, 29]. For the fine-form estimation, cross-entropy-based integral loss is used as a loss function like ours.

As Figure 6 shows, the C2F (*i.e.*, ours) exhibits a more accurate performance than F2F, which indicates that coarse input pose representation is more beneficial than fine input pose representation. Also, C2C fails to improve the input pose whereas F2F and C2F successfully refine the input pose. These results indicate that the fine-form estimation is crucial for a successful refinement.

To further analyze the benefit of the fine-form estimation, we additionally trained two models (C2F- $L_H$  and C2F- $L_C$ ). Instead of using both of the  $L_C$  and  $L_H$  like the C2F does, they are trained by minimizing only either  $L_H$  or  $L_C$ . The C2F- $L_H$  learns to estimate one-hot vector ( $H$ ) by minimizing  $L_H$ , and C2F- $L_C$  is supervised to estimate coordinate ( $C$ ) by minimizing  $L_C$ . Among C2C, C2F- $L_H$ , and C2F- $L_C$ , the target form of the C2C is the most coarse representation. On the other hand, that of C2F- $L_C$  is the finest representation as described in Section 5.2. Figure 6 shows that C2C yields the worst performance while C2F- $L_C$  achieves the best among them. This finding shows that as the output representation of the PoseFix becomes a finer form, the performance improves. Thus, by integrating the two loss functions (*i.e.*,  $L_H$  and  $L_C$ ) together, we can improve the performance much, as C2F shows.

This analysis clearly shows the benefit of the coarse-to-fine estimation pipeline.

### 7.3. Performance improvement of the state-of-the-art methods by PoseFix

We report the performance improvement when the PoseFix is applied to the recent state-of-the-art human pose estimation methods. PAFs [5], AE [20], Mask R-CNN [11], CPN [7], and Simple [30] are used to generate the input pose. To obtain the pose estimation results of the previous methods, we used their released codes and pre-trained models. We tested them by ourselves without ensembling and testing time augmentation. We also trained a pose estimation model (IntegralPose) with the same network architecture and loss function with the PoseFix to show that the PoseFix can improve a model trained from the same architecture. To analyze how the PoseFix changes the OKS and frequency of each error type, we tested the PoseFix on the validation set. We also report how much the PoseFix improves AP on the test-dev set. The ResNet-152 is used as

Methods	$AP$	$AP_{.50}$	$AP_{.75}$	$AP_M$	$AP_L$	$AR$	$AR_{.50}$	$AR_{.75}$	$AR_M$	$AR_L$
AE [20]	56.6	81.7	62.1	48.1	69.4	62.5	84.9	67.2	52.2	76.5
<b>+ PoseFix (Ours)</b>	<b>63.9</b>	<b>83.6</b>	<b>70.0</b>	<b>56.9</b>	<b>73.7</b>	<b>69.1</b>	<b>86.6</b>	<b>74.2</b>	<b>61.1</b>	<b>79.9</b>
PAFs [5]	61.7	84.9	67.4	57.1	68.1	66.5	87.2	71.7	60.5	74.6
<b>+ PoseFix (Ours)</b>	<b>66.7</b>	<b>85.7</b>	<b>72.9</b>	<b>62.9</b>	<b>72.3</b>	<b>71.3</b>	<b>88.0</b>	<b>76.7</b>	<b>66.3</b>	<b>78.1</b>
Mask R-CNN (ResNet-50) [11]	62.9	87.1	68.9	57.6	71.3	69.7	91.3	75.1	63.9	77.6
<b>+ PoseFix (Ours)</b>	<b>67.2</b>	<b>88.0</b>	<b>73.5</b>	<b>62.5</b>	<b>75.1</b>	<b>74.0</b>	<b>92.2</b>	<b>79.6</b>	<b>68.8</b>	<b>81.1</b>
Mask R-CNN (ResNet-101)	63.4	87.5	69.4	57.8	72.0	70.2	91.8	75.6	64.3	78.2
<b>+ PoseFix (Ours)</b>	<b>67.5</b>	<b>88.4</b>	<b>73.8</b>	<b>62.6</b>	<b>75.5</b>	<b>74.3</b>	<b>92.6</b>	<b>79.9</b>	<b>69.1</b>	<b>81.4</b>
Mask R-CNN (ResNeXt-101-64)	64.9	88.6	71.0	59.6	73.3	71.4	92.4	76.8	65.9	78.9
<b>+ PoseFix (Ours)</b>	<b>68.7</b>	<b>89.3</b>	<b>75.2</b>	<b>64.1</b>	<b>76.4</b>	<b>75.2</b>	<b>93.1</b>	<b>80.9</b>	<b>70.3</b>	<b>81.9</b>
Mask R-CNN (ResNeXt-101-32)	64.9	88.4	70.9	59.5	73.2	71.3	92.2	76.7	65.8	78.9
<b>+ PoseFix (Ours)</b>	<b>68.5</b>	<b>88.9</b>	<b>75.0</b>	<b>64.0</b>	<b>76.2</b>	<b>75.0</b>	<b>92.9</b>	<b>80.7</b>	<b>70.1</b>	<b>81.8</b>
IntegralPose	66.3	87.6	72.9	62.7	72.7	73.2	91.8	79.1	68.3	79.8
<b>+ PoseFix (Ours)</b>	<b>69.5</b>	<b>88.3</b>	<b>75.9</b>	<b>65.7</b>	<b>76.1</b>	<b>75.9</b>	<b>92.4</b>	<b>81.8</b>	<b>71.1</b>	<b>82.5</b>
CPN (ResNet-50) [7]	68.6	89.6	76.7	65.3	74.6	75.6	93.7	82.6	70.8	82.0
<b>+ PoseFix (Ours)</b>	<b>71.8</b>	<b>89.8</b>	<b>78.9</b>	<b>68.3</b>	<b>78.1</b>	<b>78.2</b>	<b>93.9</b>	<b>84.3</b>	<b>73.5</b>	<b>84.6</b>
CPN (ResNet-101)	69.6	89.9	77.6	66.3	75.6	76.6	93.9	83.5	72.0	82.9
<b>+ PoseFix (Ours)</b>	<b>72.6</b>	<b>90.2</b>	<b>79.7</b>	<b>69.0</b>	<b>78.9</b>	<b>78.9</b>	<b>94.1</b>	<b>85.0</b>	<b>74.2</b>	<b>85.1</b>
Simple (ResNet-50) [30]	69.4	90.1	77.4	66.2	75.5	75.1	93.9	82.4	70.8	81.0
<b>+ PoseFix (Ours)</b>	<b>72.5</b>	<b>90.5</b>	<b>79.6</b>	<b>68.9</b>	<b>79.0</b>	<b>78.0</b>	<b>94.1</b>	<b>84.4</b>	<b>73.4</b>	<b>84.1</b>
Simple (ResNet-101)	70.5	90.7	78.8	67.5	76.3	76.2	94.3	83.7	72.1	81.9
<b>+ PoseFix (Ours)</b>	<b>73.3</b>	<b>90.8</b>	<b>80.7</b>	<b>69.8</b>	<b>79.8</b>	<b>78.7</b>	<b>94.4</b>	<b>85.3</b>	<b>74.3</b>	<b>84.8</b>
Simple (ResNet-152)	71.1	90.7	79.4	68.0	76.9	76.8	<b>94.4</b>	84.3	72.6	82.4
<b>+ PoseFix (Ours)</b>	<b>73.6</b>	<b>90.8</b>	<b>81.0</b>	<b>70.3</b>	<b>79.8</b>	<b>79.0</b>	<b>94.4</b>	<b>85.7</b>	<b>74.8</b>	<b>84.9</b>

Table 2: Improvement of APs when the PoseFix is applied to the state-of-the-art methods. The APs are calculated on the test-dev set.

the backbone of the PoseFix, and the size of the input image is set to  $384 \times 288$ .

**OKS change.** The graph in Figure 7 shows the change of the OKS of the same instance when the PoseFix is applied to the baseline state-of-the-art methods.

**Error frequency change.** Figure 8 shows how the frequency of each status or error type changes when the PoseFix is applied to the CPN.

**AP improvement.** Table 2 shows the improvements in AP when the PoseFix is applied to the recent state-of-the-art human pose estimation methods. We also included the results of using different backbone networks [13, 31] for the Mask R-CNN, CPN, and Simple.

As Figures 7, 8 and Table 2 show, the PoseFix consistently improves the performance of the state-of-the-art methods. The PoseFix corrects not only the small displacement error (*i.e.*, jitter), but also the large displacement errors (*i.e.*, inversion, miss, and swap) as in Figure 8. Taking into account the fact that the state-of-the-art methods used in the experiments vary in structure and learning strategies, we believe that our model has generalizability that can be applied to other pose estimation methods. It is also noticeable that

the PoseFix does not require any code or knowledge of the pose estimation methods, which makes our model very easy and convenient to use in practice.

## 8. Conclusion

We proposed a novel and powerful network, PoseFix, for human pose refinement. Unlike conventional end-to-end multi-stage architecture models, the proposed PoseFix is a model-agnostic pose refinement network. To train the PoseFix, we generate the input pose by synthesizing pose errors according to empirical pose error distributions on the groundtruth pose. The PoseFix takes an input pose in a coarse form and estimates the refined pose in a finer form. Since PoseFix is model-agnostic, it does not require any code or knowledge about the target models. So, it can be used as a post-processing add-on module conveniently. We showed that the PoseFix achieves better performance than the conventional multi-stage architecture-based pose refinement module. Furthermore, the PoseFix consistently improves the accuracy of other methods on the commonly used pose estimation benchmark.

## Supplementary Material of “PoseFix: Model-agnostic General Human Pose Refinement Network”

In this supplementary material, we present more experimental results that could not be included in the main manuscript due to the lack of space.

### 1. Comparison with conventional end-to-end trainable multi-stage refinement

In Table 1 of the main manuscript, we compared the accuracy of the conventional end-to-end trainable multi-stage refinement model (E2E-refine) and the proposed model-agnostic refinement model (MA-refine). We tried to show the effectiveness of the proposed model-agnostic refinement model by making the number of parameters of the E2E-refine and MA-refine same.

However, as the conventional refinement requires careful model design, simply adding a refinement module which has the same network architecture with the PoseFix can result in sub-optimal performance. Therefore, we compare the accuracy of the refinement module of the state-of-the-art refinement-based method (*i.e.*, CPN [7]) and the PoseFix. The CPN consists of two parts. The first one, GlobalNet, is the baseline of the CPN. The second one, RefineNet, refines the pose estimation results of the GlobalNet. We use the GlobalNet as the pose estimation model and compare the accuracy improvement of the RefineNet and PoseFix. We trained and tested the CPN with GlobalNet only and both of the GlobalNet and RefineNet, using their released code.

Table 3 shows our PoseFix improves AP more than state-of-the-art refinement module (*i.e.*, RefineNet) by a large margin. This comparison demonstrates the benefit of the model-agnostic refinement over conventional end-to-end trainable multi-stage refinement more clearly.

### 2. Performance improvement of the state-of-the-art methods by PoseFix

In Figure 8 of the main manuscript, we showed how the frequency of each error type changes when the PoseFix is applied to the state-of-the-art method (*i.e.*, CPN [7]). We additionally show the changes of the AE [20] and Mask R-CNN [11] in Figure 9 and 10, respectively. As the Figures show, our PoseFix improves the performance by fixing all types of pose errors.

We demonstrate more generalizability by showing performance improvement on another 2D multi-person pose estimation dataset (*i.e.*, PoseTrack 2018 [2]). The PoseTrack 2018 dataset includes 66K frames, and they are split into training, validation and testing set. The state-of-the-art human pose estimation method, Simple [30], is re-

Methods	$AP$	$AP_{.50}$	$AP_{.75}$	$AP_M$	$AP_L$
RefineNet [7]	69.1 (+1.8)	87.9 (+0.4)	76.6 (+2.2)	65.7 (+1.6)	75.5 (+2.2)
<b>PoseFix (Ours)</b>	<b>71.5</b> <b>(+4.2)</b>	<b>88.0</b> <b>(+0.5)</b>	<b>77.6</b> <b>(+3.2)</b>	<b>68.0</b> <b>(+3.9)</b>	<b>78.1</b> <b>(+4.8)</b>

Table 3: AP comparison between state-of-the-art conventional end-to-end trainable multi-stage refinement model (RefineNet [7]) and the proposed model-agnostic refinement model (PoseFix) on the MS COCO [19] validation set. The number in the parenthesis denotes the AP change from the input pose (*i.e.*, GlobalNet of the CPN [7]).

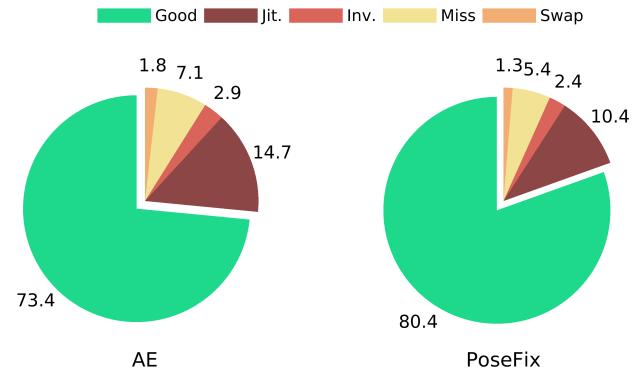


Figure 9: Frequency of each error type change when the PoseFix is applied to the AE. The frequency is calculated on the MS COCO [19] validation set.

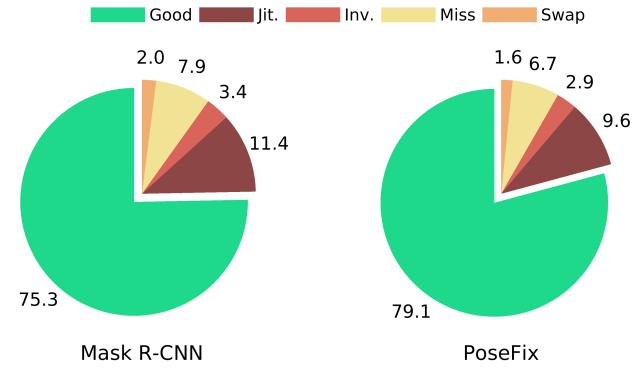


Figure 10: Frequency of each error type change when the PoseFix is applied to the Mask R-CNN. The frequency is calculated on the MS COCO [19] validation set.

implemented by ours <sup>2</sup> and its testing result is used as the input pose of the PoseFix. Both of the Simple [30] and PoseFix is pre-trained on the COCO dataset and trained again on the PoseTrack 2018 training set without hyperparameter changes following [30]. Figure 11 and Table 5 show performance improvement on the PoseTrack 2018 validation set. As they show, the PoseFix significantly improves the perfor-

<sup>2</sup><https://github.com/mks0601/TF-SimpleHumanPose>

Methods	$AP$	$AP_{.50}$	$AP_{.75}$	$AP_M$	$AP_L$	$AR$	$AR_{.50}$	$AR_{.75}$	$AR_M$	$AR_L$
RMPE [9]	61.0	82.9	68.8	57.9	66.5	-	-	-	-	-
PAFs [5]	61.8	84.9	67.5	57.1	68.2	66.5	87.2	71.8	60.6	74.6
Mask R-CNN [11]	63.1	87.3	68.7	57.8	71.4	-	-	-	-	-
AE [20]	65.5	86.8	72.3	60.6	72.6	70.2	89.5	76.0	64.6	78.1
Integral [26]	67.8	88.2	74.8	63.9	74.0	-	-	-	-	-
G-RMI [22]	64.9	85.5	71.3	62.3	70.0	69.7	88.7	75.5	64.4	77.1
G-RMI* [22]	68.5	87.1	75.5	65.8	73.3	73.3	90.1	79.5	68.1	80.4
MultiPoseNet [18]	69.6	86.3	76.6	65.0	76.3	73.5	88.1	79.5	68.6	80.3
CFN [14]	72.6	86.1	69.7	<b>78.3</b>	64.1	-	-	-	-	-
CPN [7]	72.1	91.4	80.0	68.7	77.2	78.5	<b>95.1</b>	85.3	74.2	84.3
CPN++ [7]	73.0	91.7	80.9	69.5	78.1	79.0	<b>95.1</b>	85.9	74.8	84.7
Simple [30]	73.7	<b>91.9</b>	81.1	70.3	80.0	79.0	-	-	-	-
Simple [30]	73.3	91.2	80.9	69.8	79.7	78.7	94.8	85.4	74.2	84.8
<b>+ PoseFix (Ours)</b>	<b>74.9</b>	91.2	<b>81.9</b>	71.1	<b>81.2</b>	<b>79.9</b>	94.8	<b>86.3</b>	<b>75.5</b>	<b>86.0</b>

Table 4: Comparison of APs with the state-of-the-art methods on the test-dev set. “\*” means that the method involves extra data for training. “++” indicates results using ensemble.

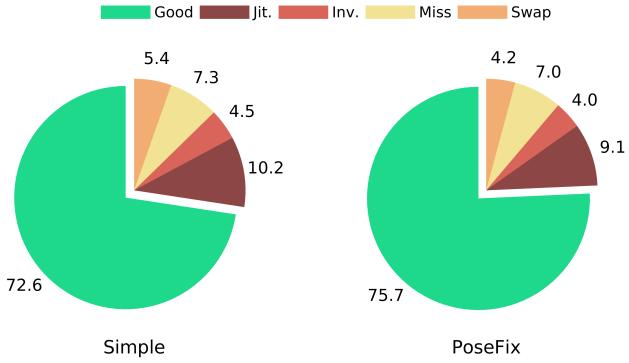


Figure 11: Frequency of each error type change when the PoseFix is applied to the Simple. The frequency is calculated on the PoseTrack 2018 [2] validation set.

mance of the input pose. They show the proposed PoseFix can improve the input pose on variable datasets.

### 3. Comparison with state-of-the-art methods

We compare the performance of the PoseFix with state-of-the-art methods, which include PAFs [5], G-RMI [22], AE [20], RMPE [9], Mask R-CNN [11], CFN [14], CPN [7], Integral [26], MultiPoseNet [18], and Simple [30] on the MS COCO [19] test-dev set. All the performance are from their papers. We used Simple [30] as the input pose of the PoseFix. As they did not release the human detection model and result, we used our human detection model which achieves 57.2 AP for the human category on the test-dev set. The Simple [30] with our human detection model outputs slightly worse performance (73.3 AP) than the original Simple [30] (73.7 AP).

As shown in Table 4, our PoseFix outperforms all exist-

Methods	Head	Shou	Elb	Wri	Hip	Knee	Ankl	Total
Simple [30]	74.4	76.9	72.2	65.2	69.2	70.0	62.9	70.4
<b>+ PoseFix (Ours)</b>	<b>79.0</b>	<b>81.6</b>	<b>76.4</b>	<b>69.7</b>	<b>75.2</b>	<b>74.3</b>	<b>67.0</b>	<b>75.0</b>

Table 5: Improvement of APs when the PoseFix is applied to the state-of-the-art method. The APs are calculated on the PoseTrack 2018 validation set.

Methods	Head	Shou	Elb	Wri	Hip	Knee	Ankl	Total
PoseRefiner [10]	74.0	76.8	72.2	65.4	70.5	69.7	63.7	70.6
<b>PoseFix (Ours)</b>	<b>79.0</b>	<b>81.6</b>	<b>76.4</b>	<b>69.7</b>	<b>75.2</b>	<b>74.3</b>	<b>67.0</b>	<b>75.0</b>

Table 6: AP comparison between PoseRefiner [10] and PoseFix on the PoseTrack 2018 validation set. The number in the parenthesis denotes the AP change from the input pose (*i.e.*, Simple).

ing methods. It is noticeable that our method can achieve better performance when a new state-of-the-art method is proposed by using it as the input pose of our method. We also compare the performance of the PoseFix with PoseRefiner [10] which has a similar approach to ours in Table 6. Table shows our PoseFix improve input pose significantly more than PoseRefiner [10].

### 4. Qualitative results

We show some qualitative results on the MS COCO [19] test-dev set. Figure 12 and 13 show the input images, input poses, and refined poses when the PoseFix is applied to Mask R-CNN [11]. Figure 14 shows final results when the PoseFix is applied to Simple [30].

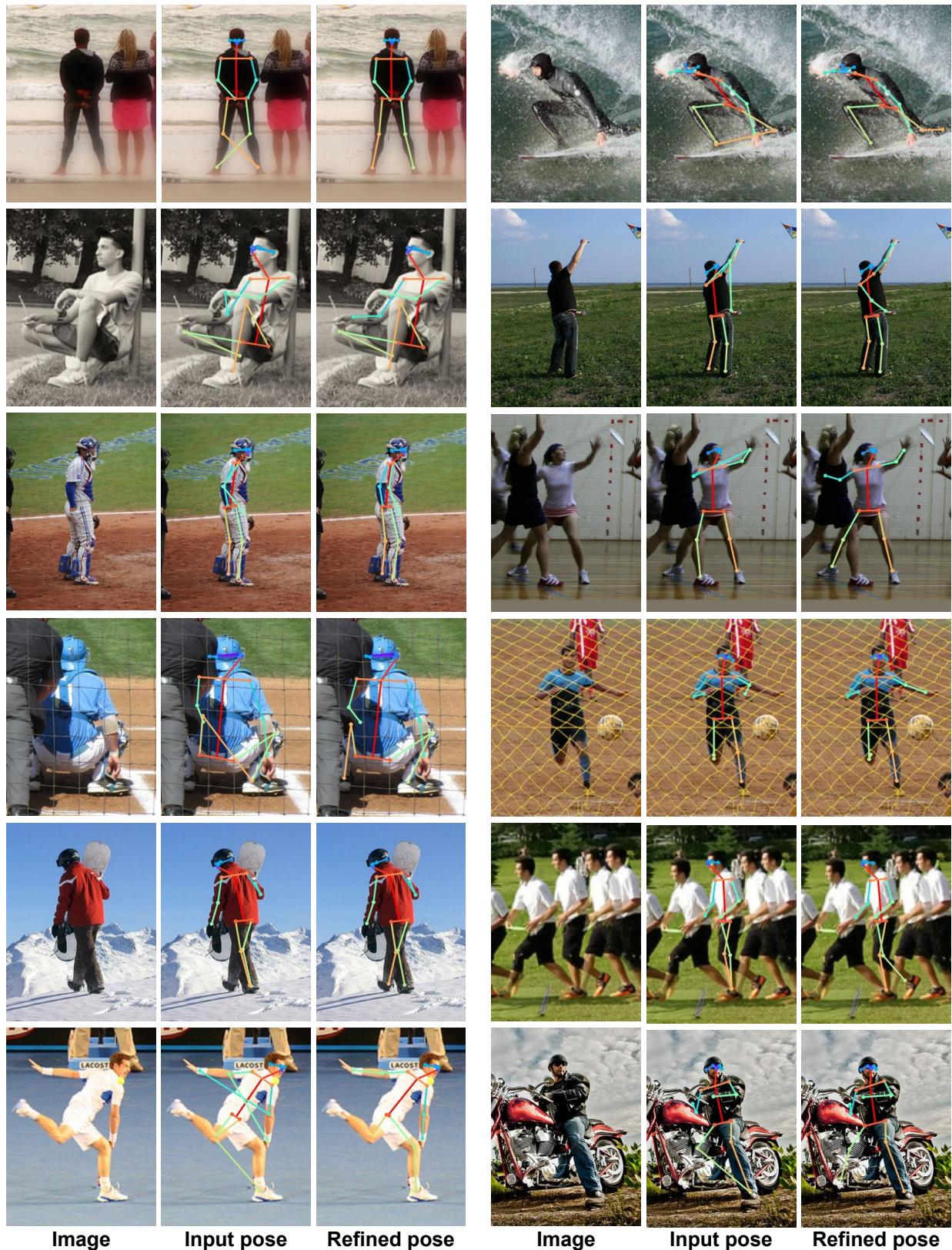


Figure 12: Qualitative results of the PoseFix on the test-dev set.

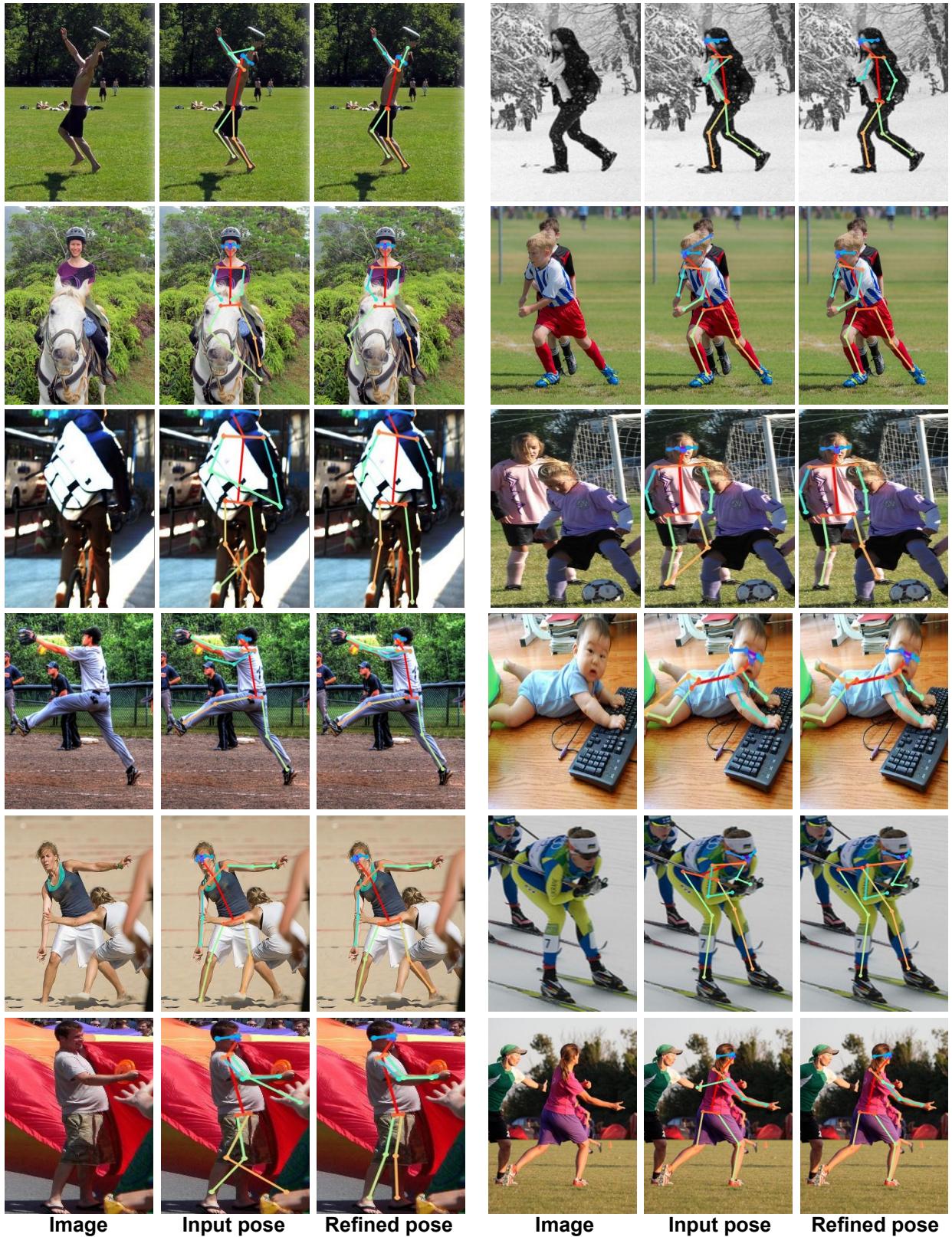


Figure 13: Qualitative results of the PoseFix on the test-dev set.



Figure 14: Qualitative results of the PoseFix on the test-dev set.

## References

- [1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, 2016.
- [2] M. Andriluka, U. Iqbal, E. Insafutdinov, L. Pishchulin, A. Milan, J. Gall, and B. Schiele. Posetrack: A benchmark for human pose estimation and tracking. In *CVPR*, 2018.
- [3] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *CVPR*, 2014.
- [4] A. Bulat and G. Tzimiropoulos. Human pose estimation via convolutional part heatmap regression. In *ECCV*, 2016.
- [5] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. *CVPR*, 2017.
- [6] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. In *CVPR*, 2016.
- [7] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun. Cascaded pyramid network for multi-person pose estimation. *CVPR*, 2018.
- [8] X. Chu, W. Yang, W. Ouyang, C. Ma, A. L. Yuille, and X. Wang. Multi-context attention for human pose estimation. In *CVPR*, 2017.
- [9] H. Fang, S. Xie, Y.-W. Tai, and C. Lu. Rmpe: Regional multi-person pose estimation. In *ICCV*, 2017.
- [10] M. Fieraru, A. Khoreva, L. Pishchulin, and B. Schiele. Learning to refine human pose estimation. *CVPRW*, 2018.
- [11] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017.
- [12] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.
- [13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [14] S. Huang, M. Gong, and D. Tao. A coarse-fine network for keypoint localization. In *ICCV*, 2017.
- [15] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele. Deepcut: A deeper, stronger, and faster multi-person pose estimation model. In *ECCV*, 2016.
- [16] L. Ke, M.-C. Chang, H. Qi, and S. Lyu. Multi-scale structure-aware network for human pose estimation. *ECCV*, 2018.
- [17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2014.
- [18] M. Kocabas, S. Karagoz, and E. Akbas. Multiposenet: Fast multi-person pose estimation using pose residual network. In *ECCV*, 2018.
- [19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.
- [20] A. Newell, Z. Huang, and J. Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *NIPS*, 2017.
- [21] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.
- [22] G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, and K. Murphy. Towards accurate multi-person pose estimation in the wild. In *CVPR*, 2017.
- [23] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *CVPR*, 2016.
- [24] M. R. Ronchi and P. Perona. Benchmarking and error diagnosis in multi-instance pose estimation. In *ICCV*, 2017.
- [25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.
- [26] X. Sun, B. Xiao, S. Liang, and Y. Wei. Integral human pose regression. *ECCV*, 2018.
- [27] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *NIPS*, 2014.
- [28] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, 2014.
- [29] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *CVPR*, 2016.
- [30] B. Xiao, H. Wu, and Y. Wei. Simple baselines for human pose estimation and tracking. *ECCV*, 2018.
- [31] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.