PoseFix: Model-agnostic General Human Pose Refinement Network (2018)

Seho Kim

https://arxiv.org/pdf/1812.03595v3.pdf

Introduction

- Localize semantic keypoints of a human body
- A human pose refinement network

  : estimate a refined pose from a tuple of an input image and a pose
- Conventional multi-stage architecture
  - First stage: initial pose and image features
  - N stage: refined pose
  - Usually trained in an end-to-end manner

  → Highly dependent on the pose estimation model and requires careful design for successful refinement

Introduction

- Propose a model agnostic pose refinement method that does not depend on the pose estimation model

- Use error statistics; prior information → synthetic poses → train the proposed pose refinement model (PoseFix)

- Generate errors based on the pose error distributions

  ;Input: Generated pose + image

  → PoseFix; single-stage architecture with coarse-to-fine estimation pipeline

Introduction

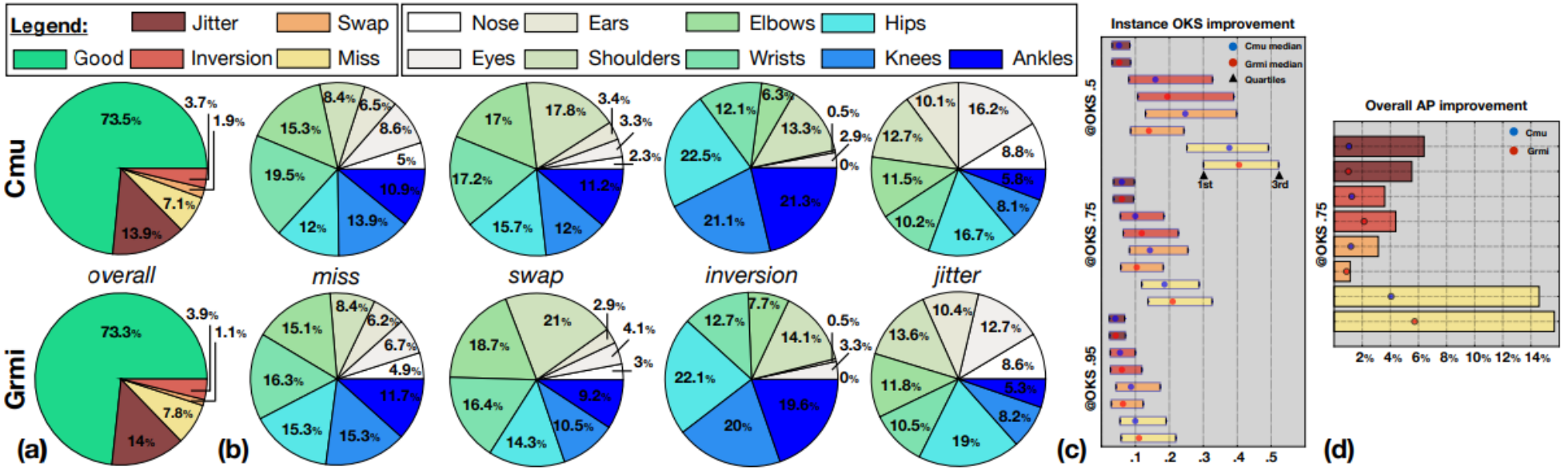


Figure 5. **Distribution and Impact of Localization Errors.** (a) Outcome for the predicted keypoints: *Good* indicates correct localization. (b) The breakdown of errors over body parts. (c) The algorithm's detections OKS improvement obtained after separately correcting errors of each type; evaluated over all the instances at OKS thresholds of .5, .75 and .95; the dots show the median, and the bar limits show the first and third quartile of the distribution. (d) The AP improvement obtained after correcting localization errors; evaluated at OKS thresholds of .75 (bars) and .5 (dots). A larger improvement in (c) and (d) shows what errors are more impactful. See Sec. 3.1 for details.

https://arxiv.org/pdf/1707.05388.pdf

Introduction

- Contributions
  - Model-agnostic general pose refinement is possible

    ;Independent of the pose estimation model,

    Based on error statistics(empirical analysis)
  - High flexibility and accessibility
  - Coarse-to-fine pipeline is crucial for successful pose refinement
  - Improve the performance of various SOTA pose estimation methods

Introduction

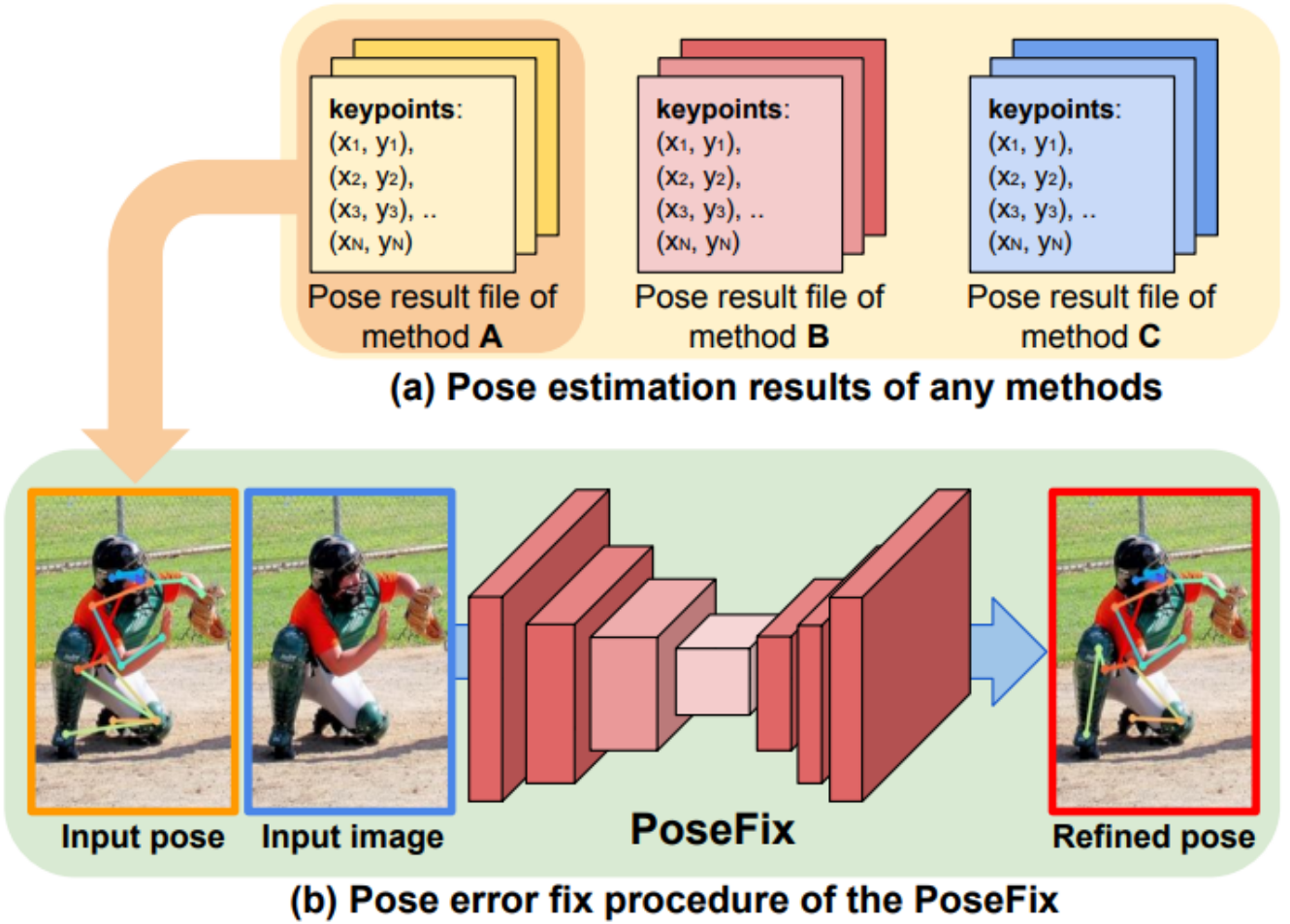


Figure 1: Testing pipeline of the PoseFix. It takes pose estimation results of any other method with an input image and outputs a refined pose. Note that the PoseFix does not require any code or knowledge about other methods.

Overview of the proposed model

- To refine th input 2D coordinates of the human body keypoints of all persons in an input image
- Based on the top-down pipeline

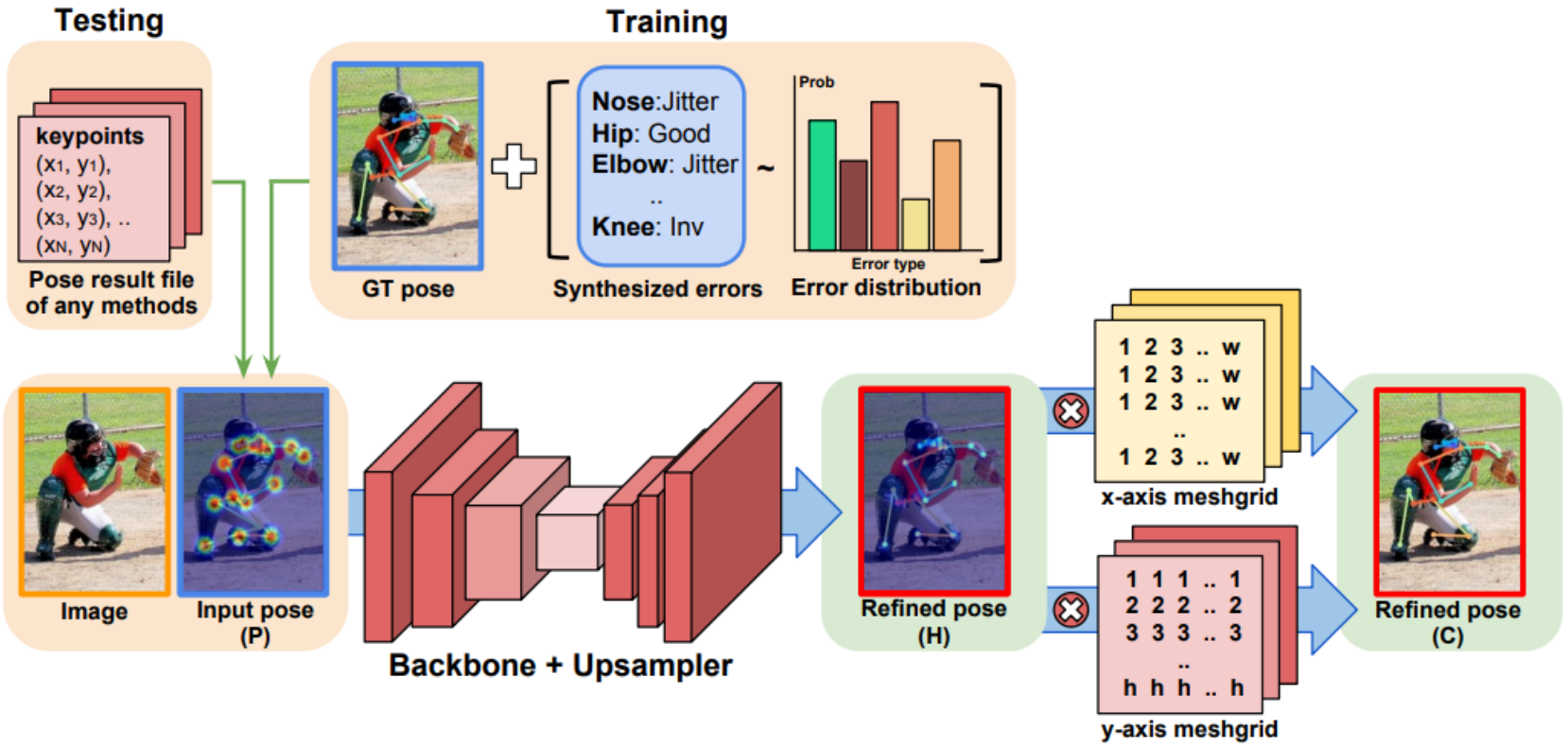


Figure 2: Overall pipeline of the PoseFix. In the training stage, the input pose is generated by synthesizing the pose errors based on the real pose error distributions on the groundtruth pose. In the testing stage, pose estimation results of any other methods become the input pose. The heatmaps are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

Synthesizing poses for training

- Generate synthesized poses randomly based on the error distributions of real poses
- Status and Errors

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 $[0, d_j^{0.85})$, respectively, is added to the groundtruth θ_j^p. The synthesized keypoint position should be closer to the original groundtruth θ_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 θ_j^p. Similar to the *good* status, the synthesized keypoint position should be closer to the original groundtruth θ_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 θ_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 θ_j^p and $\theta_{j'}^p$.

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 θ_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 θ_j^p, $\theta_{j'}^p$, $\theta_j^{p'}$, and $\theta_{j'}^{p'}$.

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

Synthesizing poses for training

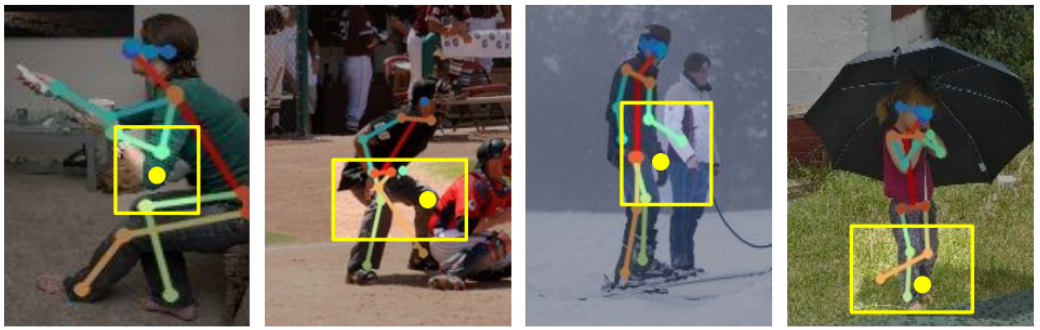


Figure 3: Visualization of synthesized pose errors for each type. The keypoint with pose error is highlighted by a yellow rectangle, and the groundtruth keypoints are drawn in a yellow circle.

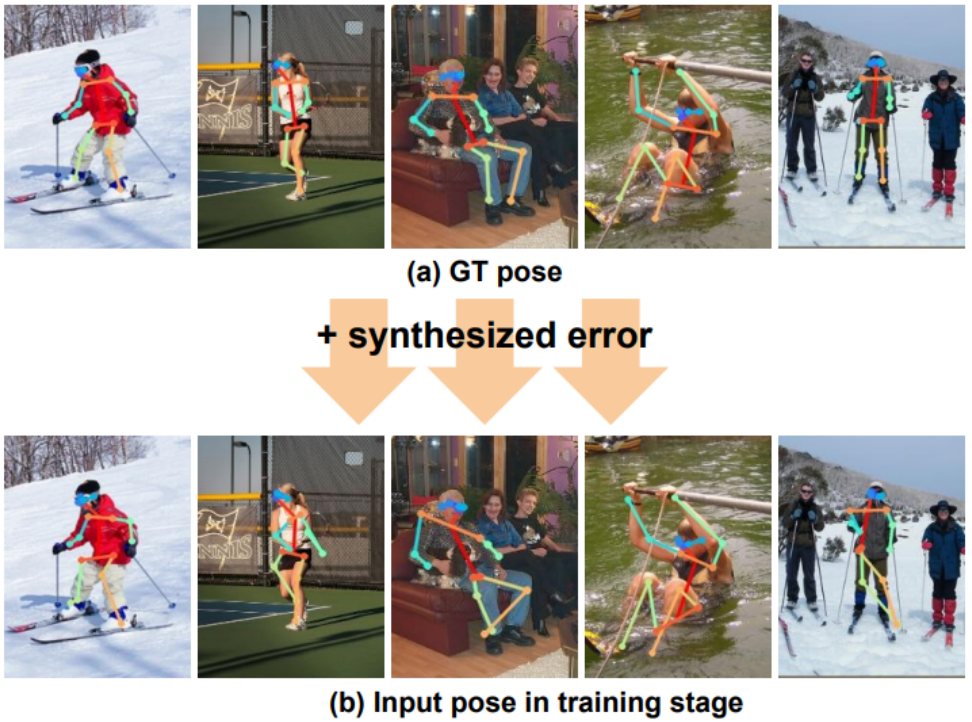


Figure 4: Visualization of the groundtruths and synthesized input poses. The synthesized poses are generated by adding errors to the groundtruth poses, which are used for training PoseFix.

Architecture and learning of PoseFix

- Model design
  - Input(image + pose) provide contextual and structured information(; acts like attention) → Fix pose errors

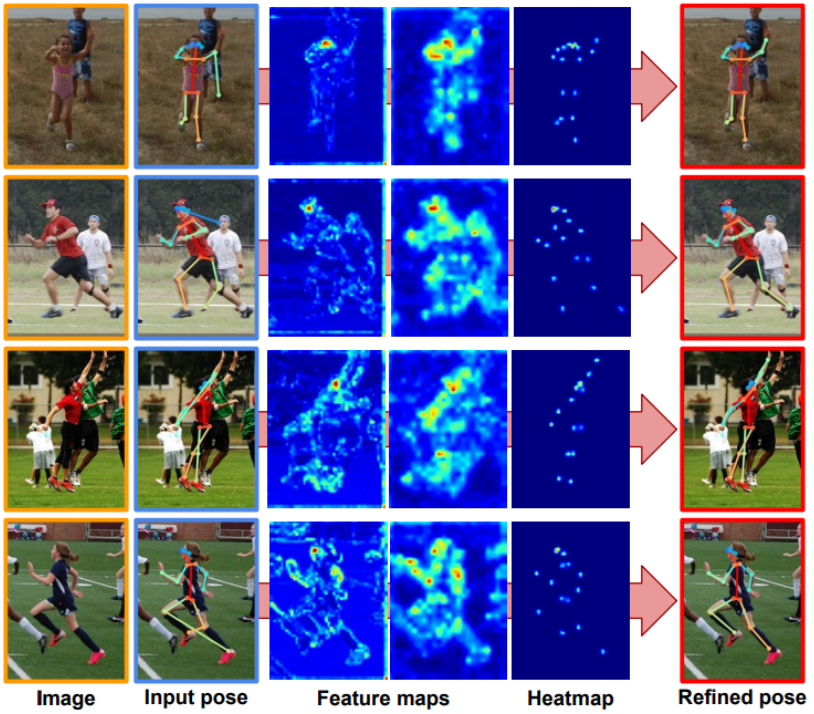


Figure 5: Visualization of feature maps and final heatmaps of the PoseFix. The feature maps and heatmaps are reduced into one channel by max pooling along the channel axis for visualization. The order of the feature maps and heatmaps in the figure is the same with that of the feedforward.

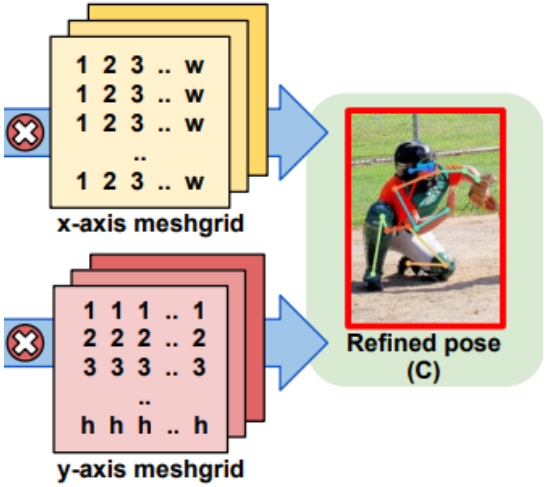
Architecture and learning of PoseFix

- Coarse-to-fine estimation
  - To make it more robust to errors
  - The coordinated of a keypoints has the least amount of uncertainty

$$\text{the coarse input pose } (P = \{P_n\}_{n=1}^{N})$$

$$\text{one-hot vector } (H = \{H_n\}_{n=1}^{N}),$$

$$\text{keypoint coordinates } (C = \{C_n\}_{n=1}^{N})$$



Architecture and learning of PoseFix

- Coarse-to-fine estimation
  - The generated input pose is concatenated with the input image and fed into the PoseFix
  - Gaussian heatmap representation
  - Cross-entropy based integral loss
  - Soft-argmax operation(Hn)
  - Lc enable PoseFix to localize keypoints more precisely

    ; free from quantization errors(calculated in continuous space)

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

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

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

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

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

Architecture and learning of PoseFix

- Network architecture
  - Deep backbone network(ResNet) and several upsampling layers
  - Final upsampling layer + softmax function → heatmaps(H) → coordinates(C)

Implementation details

- Using Tensorflow deep learning framework
- Training
  - ResNet model(pre-trained on ImageNet)
  - Adam optimizer with a mini-batch size of 128
  - Initial learning rate: 5x10^-4 (step: 10, 90, 120th epoch)
  - Augmentation: scaling(±30%), rotation(±40%), and flip
  - Groundtruth bounding box
    ; aspect ratio(h:w = 4:3) without distorting
    resize to a fixed size
  - 140 epochs with four NVIDIA 1080 Ti GPUs (two days)
- Testing
  - Flip augmentation

Experiments

- Dataset and evaluation metric
  - Trained and tested on the MS COCO 2017 keypoint detection dataset

    ;training(57K images, 150K person instances), validation(5K images), test-dev(20K images)
  - OKS-based AP metric for evaluation

Experiments

- Ablation study

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)
MA-refine (Ours)	**72.1 (+2.4)**	**88.5 (+0.2)**	**78.3 (+1.3)**	**68.6 (+2.4)**	**78.2 (+2.1)**

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).

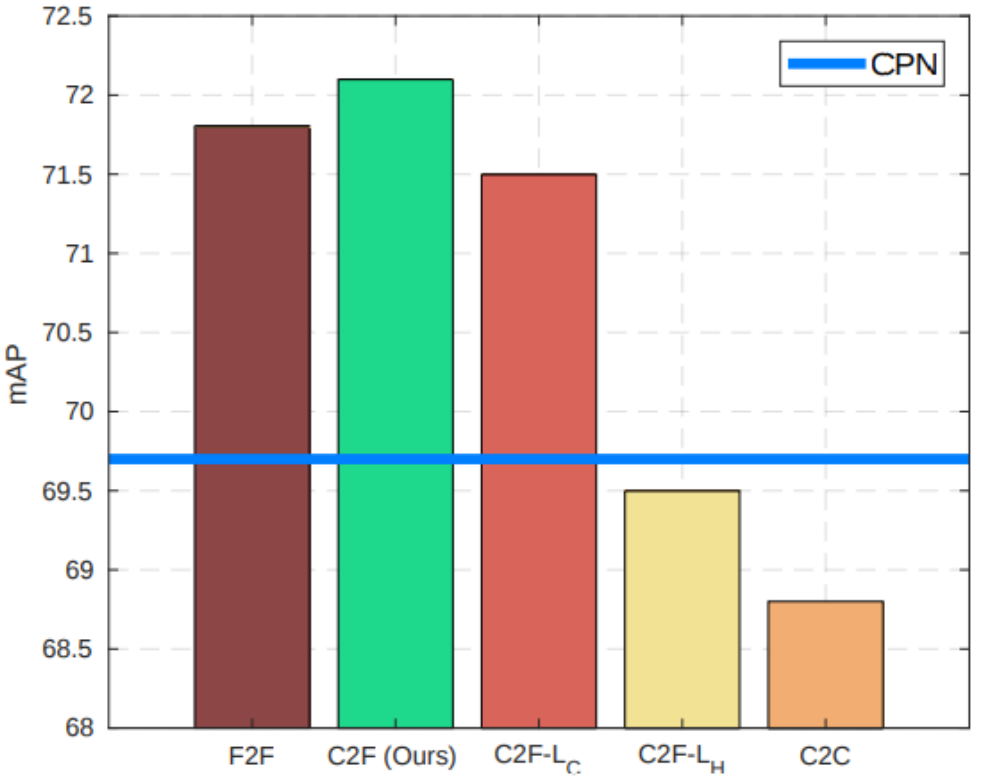


Figure 6: mAP comparison of various pipelines. The mAP is calculated on the validation set.

Experiments

- Performance improvement of the state-of-the-art methods by PoseFix

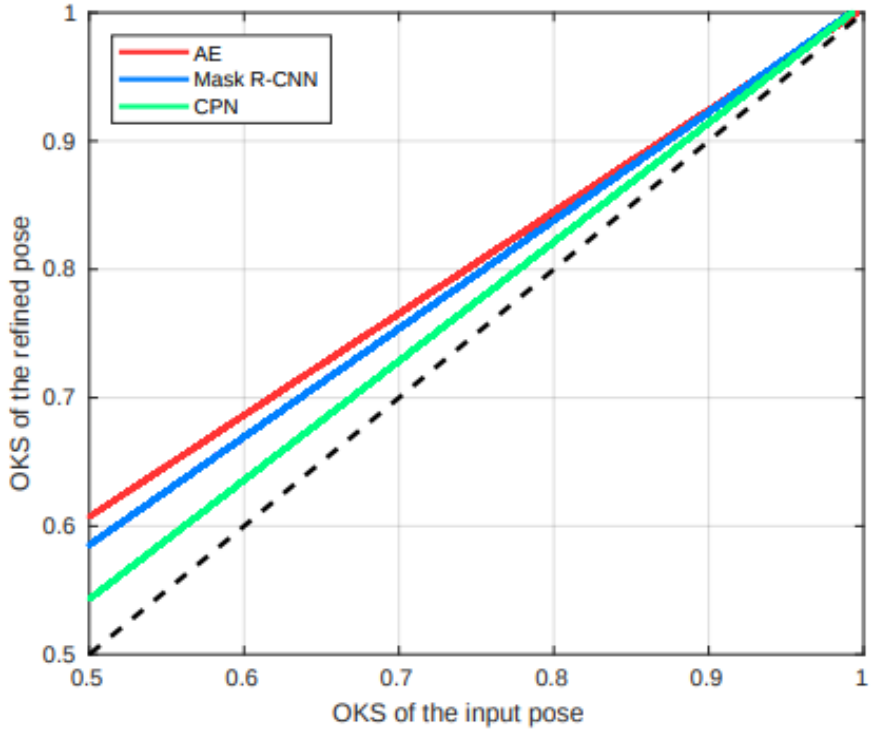


Figure 7: OKS change when the PoseFix is applied to state-of-the-art methods. The dotted line denotes identity function. OKS is calculated on the validation set.

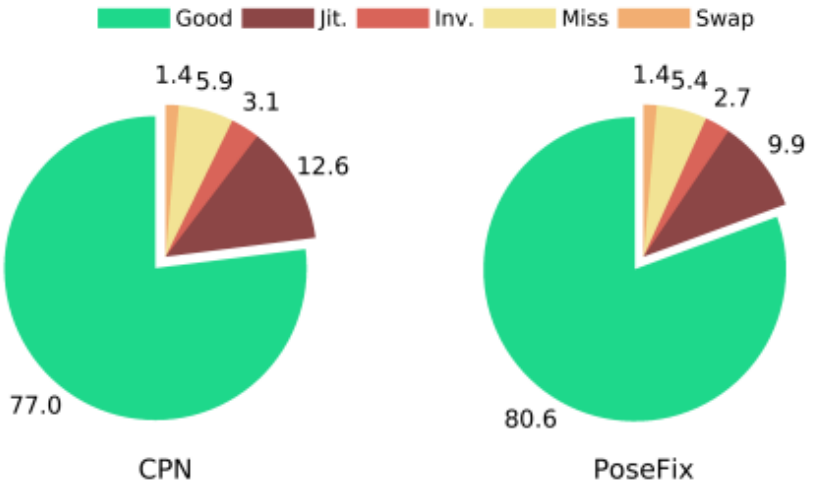


Figure 8: Frequency of each error type change when the PoseFix is applied to the CPN. The frequency is calculated on the validation set.

Experiments

- Performance improvement of the state-of-the-art methods by PoseFix

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
+ PoseFix (Ours)	**63.9**	**83.6**	**70.0**	**56.9**	**73.7**	**69.1**	**86.6**	**74.2**	**61.1**	**79.9**
PAFs [5]	61.7	84.9	67.4	57.1	68.1	66.5	87.2	71.7	60.5	74.6
+ PoseFix (Ours)	**66.7**	**85.7**	**72.9**	**62.9**	**72.3**	**71.3**	**88.0**	**76.7**	**66.3**	**78.1**
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
+ PoseFix (Ours)	**67.2**	**88.0**	**73.5**	**62.5**	**75.1**	**74.0**	**92.2**	**79.6**	**68.8**	**81.1**
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
+ PoseFix (Ours)	**67.5**	**88.4**	**73.8**	**62.6**	**75.5**	**74.3**	**92.6**	**79.9**	**69.1**	**81.4**
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
+ PoseFix (Ours)	**68.7**	**89.3**	**75.2**	**64.1**	**76.4**	**75.2**	**93.1**	**80.9**	**70.3**	**81.9**
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
+ PoseFix (Ours)	**68.5**	**88.9**	**75.0**	**64.0**	**76.2**	**75.0**	**92.9**	**80.7**	**70.1**	**81.8**
IntegralPose	66.3	87.6	72.9	62.7	72.7	73.2	91.8	79.1	68.3	79.8
+ PoseFix (Ours)	**69.5**	**88.3**	**75.9**	**65.7**	**76.1**	**75.9**	**92.4**	**81.8**	**71.1**	**82.5**
CPN (ResNet-50) [7]	68.6	89.6	76.7	65.3	74.6	75.6	93.7	82.6	70.8	82.0
+ PoseFix (Ours)	**71.8**	**89.8**	**78.9**	**68.3**	**78.1**	**78.2**	**93.9**	**84.3**	**73.5**	**84.6**
CPN (ResNet-101)	69.6	89.9	77.6	66.3	75.6	76.6	93.9	83.5	72.0	82.9
+ PoseFix (Ours)	**72.6**	**90.2**	**79.7**	**69.0**	**78.9**	**78.9**	**94.1**	**85.0**	**74.2**	**85.1**
Simple (ResNet-50) [30]	69.4	90.1	77.4	66.2	75.5	75.1	93.9	82.4	70.8	81.0
+ PoseFix (Ours)	**72.5**	**90.5**	**79.6**	**68.9**	**79.0**	**78.0**	**94.1**	**84.4**	**73.4**	**84.1**
Simple (ResNet-101)	70.5	90.7	78.8	67.5	76.3	76.2	94.3	83.7	72.1	81.9
+ PoseFix (Ours)	**73.3**	**90.8**	**80.7**	**69.8**	**79.8**	**78.7**	**94.4**	**85.3**	**74.3**	**84.8**
Simple (ResNet-152)	71.1	90.7	79.4	68.0	76.9	76.8	**94.4**	84.3	72.6	82.4
+ PoseFix (Ours)	**73.6**	**90.8**	**81.0**	**70.3**	**79.8**	**79.0**	**94.4**	**85.7**	**74.8**	**84.9**

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.

Conclusion