Context-Guided Adaptive Network for Efficient Human Pose Estimation

Lei Zhao[1,2], **Jun Wen**[1,2], **Pengfei Wang**[1,2], **Nenggan Zheng**[1,2,3,4*]
[1] Qiushi Academy for Advanced Studies, Zhejiang University, Hangzhou, China
[2] College of Computer Science and Techology, Zhejiang University, Hangzhou, China
[3] Collaborative Innovation Center for Artificial Intelligence by MOE and Zhejiang Provincial Government (ZJU)
[4] Zhejiang Lab, Hangzhou, China
{zlchn, junwen, pfei}@zju.edu.cn, zng@cs.zju.edu.cn

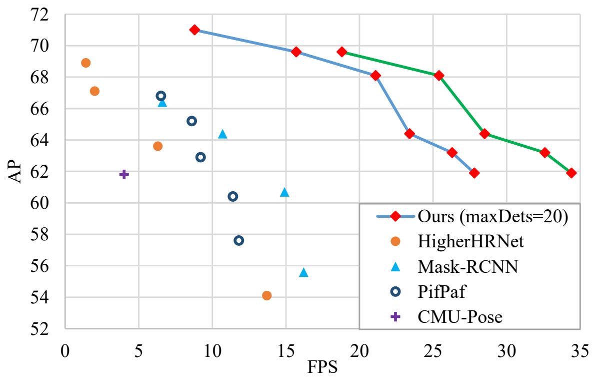
Abstract

Although recent work has achieved great progress in human pose estimation (HPE), most methods show limitations in either inference speed or accuracy. In this paper, we propose a fast and accurate end-to-end HPE method, which is specifically designed to overcome the commonly encountered jitter box, defective box and ambiguous box problems of box-based methods, e.g. Mask R-CNN. Concretely, 1) we propose the ROIGuider to aggregate box instance features from all feature levels under the guidance of global context instance information. Further, 2) the proposed Center Line Branch is equipped with a Dichotomy Extended Area algorithm to adaptively expand each instance box area, and Ambiguity Alleviation strategy to eliminate duplicated keypoints. Finally, 3) to achieve efficient multi-scale feature fusion and real-time inference, we design a novel Trapezoidal Network (TNet) backbone. Experimenting on the COCO dataset, our method achieves 68.1 AP at 25.4 fps, and outperforms Mask-RCNN by 8.9 AP at a similar speed. The competitive performance on the HPE and person instance segmentation tasks over the state-of-the-art models show the promise of the proposed method. The source code will be made available at https://github.com/zlcnup/CGANet.

Introduction

Human pose estimation (HPE) is a challenging yet fundamental computer vision problem whose goal is to determine the location of each body keypoint (e.g. eyes, shoulders, knees and so on). The provided informative knowledge is critical for tasks such as intelligent video surveillance, human-computer interaction, virtual reality, etc.

There are generally two HPE approaches: bottom-up or top-down. The bottom-up approach firstly regresses all keypoints of an image, and then assigns them to each person instances (Insafutdinov et al. 2016; Cao et al. 2017; Kreiss, Bertoni, and Alahi 2019; Cheng et al. 2020). Because of the global regressing manner of keypoints, it is difficult to adaptively handle scale variation of each instance and obtain high accuracy. Meanwhile, it typically adopts complicated post processing that could significantly reduce the inference speed (as shown in Fig. 1 and Table 4). In contrast, the top-down approach firstly detects all person instances using the detection model, and then runs the HPE model to regress the



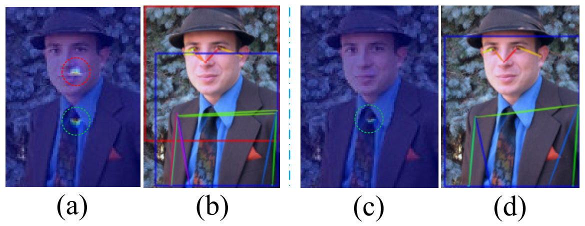


Figure 1: Top: The speed-accuracy trade-off of single-model methods on the COCO (Lin et al. 2014) *val2017* set. The green line is the speed with fixed input size (such as video), which can better reflect the actual speed. But for testing convenience and fair comparisons, we also test the average speed (blue line) on the whole COCO *val2017* set (used in the experiments). The inference details please refer to section *Experiments*. In particular, the result of CMU-Pose is achieved on *test-dev* set with refinement. Bottom: (a) The ambiguous center point heatmap predicted by the original CenterNet, which has two Gaussian peaks with the same score for one target (red and green dotted circle). (b) Illustrate the *ambiguous box* (blue and red boxes) caused by (a), and the *defective box* (blue box) that excludes the keypoints of eyes. (c) The center point heatmap after alleviating ambiguity by center line heatmap. (d) The visualization after Dichotomy Extended Area algorithm based on center line heatmap.

[*]Corresponding author.

keypoints for each instance (Huang, Gong, and Tao 2017; Papandreou et al. 2017; Fang et al. 2017; Chu et al. 2017; Sun et al. 2019). These methods normalize all cropped instances to regress the local keypoint heatmaps, thus usually with higher accuracy. However, most of them are not end-to-end trainable (*i.e.*, separated model), which limits their inference speed.

The framework of end-to-end trainable Mask-RCNN (He et al. 2017) makes a good balance between accuracy and speed, by normalizing the scale of each detected instance to regress the local heatmaps and sharing the feature extractor among all instances. However, (i) the ROIPooler of Mask-RCNN fails to fully utilize multi-scale box instance information. It only extracts instance features from specific feature levels according to the box size and discards information from the other feature levels. In addition, such box-based methods still have the commonly encountered limitations in the HPE task. (ii) *Jitter box* sensitivity, which leads the predicted keypoints to be greatly influenced and also to be jittered. (iii) *Defective box*, which means that some keypoints are excluded from the box and can never be regressed, *e.g.*, the eyes excluded by the blue box in Fig. 1(b). (iv) *Ambiguous box*, that is an instance detected in multiple boxes with each of high confidence score, *e.g.,*, the two center points of the same person shown in Fig. 1(a). In this case, it is difficult to eliminate the redundant boxes with thresholding or NMS, therefore, duplicated or ambiguous keypoint detection occurs.

To overcome the above limitations, we propose a simple yet efficient Context-Guided Adaptive Network, as shown in Fig. 2, which mainly consists of three parts: (i) ROIGuider which fuses features hierarchically under global context guidance, (ii) Center Line Branch (CLB) which adaptively extends each instance box area and alleviates ambiguity, and (iii) Trapezoidal Network (TNet) for efficient multi-scale feature aggregation. Concretely, to utilize multi-level information for robust keypoint regression, ROIGuider adaptively fuses multi-level box features and re-weights different channels under the guidance of global context information via the Guided Fusion Module (GFM). To address the *defective box* or *ambiguous box* problems, CLB adaptively adjusts the extended area for each box instance and alleviates the ambiguity of center point by utilizing the center line heatmaps in different directions. Further, we design a new backbone TNet that efficiently aggregates multi-scale features with only a small number of channels. On the COCO dataset, the results in person instance segmentation and separated-model experiments show its competitive effectiveness. We also conduct single-model experiments to show that our method achieves state-of-the-art performance at both speed and accuracy (as shown in Fig.1). The main contributions of our work can be summarized as follows:

- We propose ROIGuider to aggregate box instance features from multiple levels under the guidance of global context instance information. It effectively improves the robustness of keypoint regression with inaccurately detected boxes, *e.g. jitter box*.
- To tackle the *defective box* and *ambiguous box*, we further propose Center Line Branch with a simple Dichotomy Extended Area algorithm and Ambiguity Alleviation strategy to adaptively extend each box instance area and alleviate prediction ambiguity.
- To generalize the benefits of ROIGuider for real-time HPE, we design a novel backbone TNet to fuse multi-scale features in a highly efficient manner.
- Experimental results show that our method outperforms all single-model approaches significantly in terms of speed and accuracy, and generalizes well to other visual tasks, *e.g.*, instance segmentation.

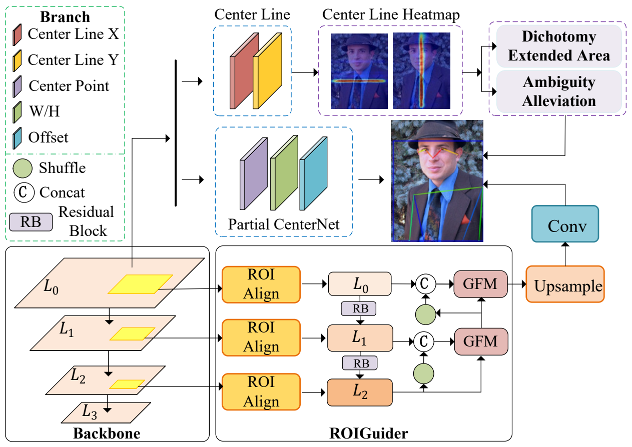


Figure 2: Overview of the proposed approach, which is mainly composed of anchor-free detection branch, center line branch and keypoint branch (ROIGuider).

Related Work

Single-Model Methods. Single-model methods (Nie et al. 2019; Jin et al. 2020) are an end-to-end way to solve the HPE task. OpenPose (Cao et al. 2017) proposes Part Affinity Fields (PAFs) to learn a 2D vector field between two keypoints to group into person instance. Associative Embedding (AE) (Newell, Huang, and Deng 2017) introduces the tag map, which labels each keypoint a tag. The keypoints with similar tags are considered to belong to the same person instance. HigherHRNet (Cheng et al. 2020) uses higher resolution keypoint heatmaps to improve accuracy, and the keypoints grouping method is the same as AE. Mask-RCNN (He et al. 2017) predicts the bounding box of each instance, and then use ROIPooler to select different level features to infer the keypoints. Different from the anchor-based detection method of Mask-RCNN, we apply the anchor-free strategy to our pipeline for pose estimation.

Separated-Model Methods. Separated-model methods (Ke et al. 2018; Tang, Yu, and Wu 2018; Moon, Chang, and Lee 2019; Zhang et al. 2020; Zhang, Tang, and Wu 2019; Huang et al. 2020) use two or more models for person detection and pose estimation, respectively. Hourglass (Newell, Yang, and Deng 2016) repeatedly uses the bottom-up and top-down structure to improve performance. Simple

(Xiao, Wu, and Wei 2018) proposes a network that adds some deconvolution layers on the ResNet (He et al. 2016) and achieves better performance. HRNet (Sun et al. 2019) achieves higher accuracy by maintaining high-resolution representations throughout the whole network. Due to the maintenance of high-resolution features, it does not perform well in speed.

The Proposed Approach

In this section, we firstly give a brief overview of the proposed approach. Next, we present ROIGuider for global context-guided multi-level feature fusion and describe how the Center Line Branch to tackle the more critical *defective box* and *ambiguous box* problems. Finally, we demonstrate the structure of the designed backbone TNet to make the idea of ROIGuider more general.

Architecture

We illustrate the proposed approach in Fig. 2. The backbone network, such as TNet or Feature Pyramid Networks (FPN) (Lin et al. 2017), produces pyramid features. The highest resolution features (L_0, stride=4) followed by five branches are used to predict center point heatmaps, width and height map, offset map and center line heatmaps of two directions. The first three are the same as CenterNet (Zhou, Wang, and Krähenbühl 2019). The center line heatmaps combine the Dichotomy Extended Area algorithm and Ambiguity Alleviation strategy to extend and refine each instance boxes dynamically. After passing through the keypoint branch of ROIGuider, the extracted hierarchical box features are employed to predict the keypoints of each instance. Considering the size of L_3 in the pyramid features is too small, only the features $L_0 \sim L_2$ in the keypoint branch are used.

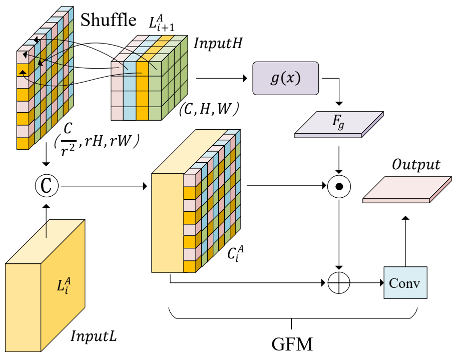


Figure 3: The structure of GFM and shuffle. "$\odot$" and "$\oplus$" denote element-wise multiplication and element-wise addition, respectively.

ROIGuider

The box-based HPE model is very sensitive to the box accuracy, such as *jitter box* which appears most frequently in the human detector. In order to improve the robustness of the HPE model to inaccurate boxes, we propose the ROIGuider. Instead of selecting aligned features from a certain level according to the box size, as done in Mask-RCNN, we integrate all levels of features under the guidance of high-level features. This is because that high-level box instance features are not sensitive to location, and have more global context semantic information. By using the high-level instance features to guide the weighting of position-sensitive low-level features, the influence of box location on the low-level features can be reduced, and the robustness can be improved. The ROIGuider also frees the model from having to carefully select different levels of aligned features for different datasets or outputs.

As shown in Fig. 2, we use ROIAlign and instance boxes to get the aligned features $L_i^A \in R^{N_b \times C \times H_i \times W_i}$ from the pyramid features $L_0 \sim L_2$, where N_b is the number of boxes and i denotes the level of features. C is the channel number of features. H_i and W_i are the height and width of the corresponding aligned features. Then, L_i^A is subsampled by a series of Residual Blocks (RB) and gradually merged to obtain the new L_i^A. Finally, inspired by (Hu, Shen, and Sun 2017), the L_i^A is adaptively weighted under the guidance of L_{i+1}^A to enhance features discriminability in the *reverse* path.

As shown in Fig. 3, the *reverse* path is mainly composed of *shuffle* (Shi et al. 2016) and Guided Fusion Module (*GFM*). Unlike previous methods of using deconvolution (Xiao, Wu, and Wei 2018), we use the shuffle to increase the feature size. The operation is more efficient, and it can alleviate the aliasing effect caused by the deconvolution. When the input feature size is (C, H, W) and the scale factor is r, the output feature size is $(\frac{C}{r^2}, rH, rW)$ after the shuffle.

The mutual guidance of different level features is mainly completed by GFM. GFM has two inputs, called guided features L_{i+1}^A and concatenate features C_i^A respectively. In order to acquire the global context information of L_{i+1}^A, we use pooling operation to generate statistics for each channel in L_{i+1}^A. The L_{i+1}^A will be squeezed to $C \times 1 \times 1$ global information descriptor d_{i+1}. Each channel is 1×1 number with global information for that channel. The value of c-th descriptor d_{i+1}^c for the feature map $L_{i+1}^A \in R^{C \times H_{i+1} \times W_{i+1}}$ can be calculated by

$$d_{i+1}^c = \frac{1}{H_{i+1} \times W_{i+1}} \sum_{x=1}^{H_{i+1}} \sum_{y=1}^{W_{i+1}} L_{i+1}^A(x, y). \tag{1}$$

We adopt the *sigmoid* function $g(x)$ to learn the nonlinear interactions between channels. The guided features F_g can be generated by $F_g = f(g(Wd))$, where $f(x)$ is convolution operation. To reduce the gradient degradation and improve the optimization speed, we also adopt the identity mapping like ResNet (He et al. 2016). The final output of GFM is calculated by

$$O_{GFM} = F_g \odot C_i^A + C_i^A. \tag{2}$$

where $\odot$ denotes element-wise multiplication.

Center Line Branch

Although ROIGuider can improve the robustness of keypoints regression under the inaccurate boxes, there is nothing to do with more critical *defective box* and *ambiguous box* problems. So we further address them by proposing Center Line Branch (CLB), which is equipped with the simple Dichotomy Extended Area algorithm and Ambiguity Alleviation strategy.

Given an image $I \in R^{H\times W\times 3}$, the CLB separately predicts features $\hat{M}_x \in R^{\frac{H}{s}\times\frac{W}{s}\times 1}$ and $\hat{M}_y \in R^{\frac{H}{s}\times\frac{W}{s}\times 1}$, where H, W are the height and width of the input image, and $s = 4$ is the output stride. $\hat{M}_x$ and $\hat{M}_y$ represent the predicted Gaussian center lines in the x and y directions, respectively. We do not use simultaneous prediction $\hat{M}_{xy} \in R^{\frac{H}{s}\times\frac{W}{s}\times 2}$, because we find it difficult to regress to the ideal Gaussian lines. Since the Gaussian lines in the x and y directions have the same form, in the following subsections, we will only introduce the y direction in detail.

Center Line Regression. Given an annotated box (x_1, y_1, x_2, y_2) in an image, firstly it is transformed to the feature-map scale. We get the center line $\{(x, y)|x = \lfloor\frac{x_1+x_2}{2}\rfloor, y_1 \le y \le y_2, y \in N\}$ with line width $d = \lceil\frac{r}{6} + 0.5\rceil$. We calculate the radius of the center point $r = \frac{b-\sqrt{b^2-4c}}{2}$ so that the minimum $IOU = \frac{(H-r)(W-r)}{2HW-(H-r)(W-r)}$ between ground-truth box and box generated within r is not less than $miniou = 0.3$, where $b = \frac{H+W}{s}, c = \frac{HW(1-miniou)}{s^2(1+miniou)}$. Then, a 2D Gaussian kernel $K(x, y) = exp(-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2})$ is applied to generate the ground-truth center line heatmap M_y^{gt}, where $\sigma = 1.4$ in our paper. In order to make the center line regression continuous and have clear boundaries, we follow (Lin et al. 2018; Li, Su, and Wang 2020) to automatically penalize the weight of easy positive samples during training. The loss of y direction can be calculated by

$$L_y = \frac{1}{N}\sum_M \begin{cases} \hat{M}_y^{\alpha} \cdot \|\hat{M}_y - M_y^{gt}\|_2^2, \; M_y^{gt} < thre \\ (1-\hat{M}_y)^{\alpha} \cdot \|\hat{M}_y - M_y^{gt}\|_2^2, \; else, \end{cases} \quad (3)$$

where $thre$ denotes the threshold of easy positive samples, N is the number of easy positive samples, and α is the hyper parameter to control the degree of penalty. We set $thre = 0.1$ and $\alpha = 1$ in our work. It should be noted that *sigmoid* function is applied to $\hat{M}_y = \frac{1}{1+exp(-\hat{M}_y)}$.

Dichotomy Extended Area. During the inference stage, the easiest way to alleviate the *defective box* problem is to extend all the box instance areas at a certain ratio. However, this way cannot make dynamic adjustments for each box instance, so it may have negative impacts on some boxes. We propose a simple Dichotomy Extended Area algorithm to solve the problem with the help of center line heatmap. For a coarse box ($\hat{x}_1$, $\hat{y}_1$, $\hat{x}_2$, $\hat{y}_2$) predicted by partial CenterNet, its *top point* and *bottom point* lie at $\hat{pt} = (\frac{\hat{x}_1+\hat{x}_2}{2}, \hat{y}_1)$ and $\hat{pb} = (\frac{\hat{x}_1+\hat{x}_2}{2}, \hat{y}_2)$, respectively. As shown in Algorithm 1, take the extended upper boundary (*top point*) as an example. First of all, the maximum extended range $ratio$ and the threshold ($thre$) of center line heatmap are set according to the height and width of the box. The upper boundary is then extended directly to the maximum $offset$. Under the $offset$, it is determined whether the score of the center line map is less or more than the $thre$, and then according to it to decide whether the upper boundary moves down or up to $\frac{offset}{2}$. In our work, we only need less than $MAX\_ITERS = 10$ iterations to get a good result, and the process can be easily extended to multi-box and multi-boundary forms to reduce the computing time. Fig. 1(d) shows the area extended result from Fig. 1(b) (blue box) by employing the proposed algorithm.

Algorithm 1 Dichotomy Extended Area (one box, upper boundary).

```
Input: $(\hat{x}_1, \hat{y}_1, \hat{x}_2, \hat{y}_2)$, $\hat{M}_y$, $ratio$ and $thre$
Output: The extended box $(\hat{x}_1, \hat{y}_1, \hat{x}_2, \hat{y}_2)$
 1: $offset = ratio \cdot \frac{\hat{y}_2-\hat{y}_1}{2}; n\_iter = 0$
 2: $\hat{y}_1 = \hat{y}_1 - offset$
 3: while $n\_iter \le MAX\_ITERS$ do
 4:     $offset = \frac{offset}{2}; n\_iter = n\_iter + 1$
 5:     if $\hat{M}_y(\frac{\hat{x}_1+\hat{x}_2}{2}, \hat{y}_1) \le thre$ then
 6:         $\hat{y}_1 = \hat{y}_1 - offset$
 7:     else
 8:         $\hat{y}_1 = \hat{y}_1 + offset$
 9:     end if
10: end while
11: return $(\hat{x}_1, \hat{y}_1, \hat{x}_2, \hat{y}_2)$
```

Ambiguity Alleviation. The *ambiguous box* problem is mainly caused by the duplication of the Gaussian center point in the heatmap $\hat{H}_{det}$, as shown in Fig. 1(a). With the help of center line heatmap $\hat{M}$, there is a simple and clear way to tackle this problem through

$$\hat{H}_{det} = \hat{H}_{det}^{\alpha} \cdot (\hat{M}_x \cdot \hat{M}_y)^{\beta}, \quad (4)$$

where α and β are used to control the balance $\hat{H}_{det}$ and $\hat{M}$. Fig. 1(c) shows the result of multiplying Fig. 1(a) by center line heatmap. Obviously, the heatmap in the red dotted circle is significantly weakened.

Efficient Trapezoidal Network

More generally, we do not constrain the idea of ROIGuider to only box instances. We extend it to the more general situation and design an efficient backbone to improve the overall performance of the model, named Trapezoidal Network (TNet). With the help of high-level features to guide the weighting of low-level features, the low-level features can adaptively learn the features discriminability. In this way, the backbone does not need too many channels to achieve high performance.

The structure of TNet is shown in Fig. 4. The input image I is firstly passed through feature extractor f_{RB} to extract features $L_I = f_{RB}(I)$, where f_{RB} consists of a number of Residual Blocks (RB). Then L_I passes through 1×1 and 3×3 convolutions respectively to generate the next depth input features L_i^d and L_{i+1}^d, where d and i denote the depth of network and the level of scale. For any depth d, we can generate features $\{L_0^d, L_1^d, ..., L_i^d\}$ by employing the same manner. It should be noted that each feature L_i^d is merged with the previous RB downsampled features L_{i-1}^d. That is,

$$L_i^d = f_{RB}(L_{i-1}^d) + L_i^d. \quad (5)$$

Finally, the outputs $\{O_0^d, O_1^d, ..., O_i^d\}$ of depth d can be calculated by

$$C_i^d = \mathcal{C}(\mathcal{S}(L_{i+1}^d), L_i^d); \tag{6}$$

$$O_i^d = GFM(L_{i+1}^d, C_i^d) + C_i^d, \tag{7}$$

where $\mathcal{C}$ and $\mathcal{S}$ represent concatenate and shuffle operation respectively.

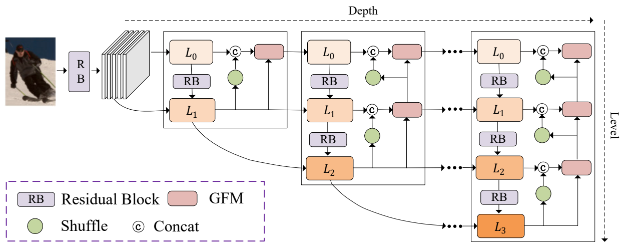


Figure 4: The structure of TNet.

Total Loss

The keypoint ground-truth heatmaps are generated by applying the 2D Gaussian kernel around the keypoint locations. In the keypoint heatmap loss L_{km}, the peak of the Gaussian distribution is treated as the positive sample while any other pixel is treated as the negative sample. We use the modified focal loss like (Law and Deng 2018; Zhou, Wang, and Krähenbühl 2019) as the objective function. Given the predicted keypoint heatmap $\hat{K}$ and ground-truth heatmap K, the L_{km} can be calculated by

$$L_{km} = \frac{-1}{N} \sum_p \begin{cases} (1-\hat{K}_p)^\alpha log(\hat{K}_p), \; K_p = 1 \\ (1-K_p)^\beta \hat{K}_p^\alpha log(1-\hat{K}_p), \; else, \end{cases} \tag{8}$$

where N is the number of annotated keypoints, and p denotes the each position of $\hat{K}$ and K. We set the hyper-parameters $\alpha = 2$ and $\beta = 4$ in our work.

The final total loss L is composed of center point heatmap loss L_{det}, width-height map loss L_{wh}, offset map loss L_{off}, center line heatmap loss L_x and L_y, keypoint heatmap loss L_{km} and their corresponding weights. That is,

$$\begin{aligned} L = \omega_{det} L_{det} + \omega_{wh} L_{wh} + \omega_{off} L_{off} \\ + \omega_x L_x + \omega_y L_y + \omega_{km} L_{km}, \end{aligned} \tag{9}$$

where $\omega_{km} = 3.0$, $\omega_{wh} = 0.1$, and other weights are 1.0 in our work. The L_{det}, L_{wh} and L_{off} are the same as CenterNet.

Experiments

In this section, we evaluate our approach on the COCO dataset (Lin et al. 2014), which contains over 200, 000 images and 250, 000 person instances labeled with 17 keypoints. It is divided into *train2017/val2017/test-dev2017* sets with 57k, 5k and 20k images respectively. Our models are trained on the *train2017* set without using extra data. We adopt the typical average precision (AP) and average recall (AR) as the evaluation metric. We report the inference time (speed) of models using one batch size on the same environment equipped with a single NVIDIA GTX 2080Ti GPU, CUDA V10.0 and PyTorch 1.4. The inference time includes model inference time with COCO max detections (maxDets=20) and the post-processing time of getting the final keypoint coordinates. In addition, we also conduct the HPE separated model and person instance segmentation experiments to verify the effectiveness and generality of our approach.

Implementation Details

We use data augmentation with random scale between $0.6 \sim 1.5$, random rotation between $-45° \sim +45°$, random translation between $-40 \sim +40$ and random flip to crop an input image patch. The aligned feature sizes are $\frac{1}{16}$, $\frac{1}{32}$ and $\frac{1}{64}$ of the training input size, respectively. We use the SGD optimizer for 95 epochs, with an initial learning rate of 1e-2 (dropped to 1e-3 and 1e-4 at the 70th and 85th epochs, respectively). For the pre-training TNet, we only modify the prediction layer to accommodate the classification problem on the ImageNet (Krizhevsky, Sutskever, and Hinton 2012) dataset.

	Component	**Params**	**Speed**	***AP***
(a)	FPN-50 + ROIPooler	34.7M	13.5fps	57.5
(b)	FPN-50 + ROIGuider	37.3M	18.0fps	60.5
(c)	TNet-D1W64 + ROIGuider	20.1M	27.8fps	61.9
(d*)	+ Fixed Extend Area	20.1M	27.2fps	62.2
(d)	+ Dichotomy Extended Area	20.2M	26.4fps	63.0
(e)	+ Ambiguity Alleviation	20.2M	26.4fps	63.2
(f)	+ ROIGuider (× 2)	26.9M	23.4fps	64.4

Table 1: Ablation study of different components on the COCO *val2017* set. ROIPooler represents the original component with a series of convolutions used in Mask-RCNN. It should be noted that experiments (d*) and (d) are mutually exclusive. All results are obtained with input size *384*.

Ablation Study

The ablation study is conducted with input size *384* on the COCO *val2017* set. We present the number of parameters, speed and AP to illustrate the effectiveness of each component. The results are shown in Table 1.

ROIPooler *vs.* ROIGuider. To verify the effectiveness of the proposed ROIGuider, we consider using Mask-RCNN's ROIpooler with a series of 3×3 convolutions as the baseline. Compared with ROIPooler (a), our proposed ROIGuider (b) has a considerable improvement (+3.0 AP) with faster speed (+4.5 fps). This demonstrates that guiding and fusing corresponding instance features at different levels can effectively improve accuracy.

Backbone. We further design the backbone TNet based on the idea of ROIGuider and we replace FPN to the TNet-D1W64. It can be seen from (b) and (c) that, compared with FPN, TNet can obtain 1.4 AP improvement by using only half of the parameters, and it has considerable real-time inference speed (27.8 fps). It is worth mentioning that the time to complete the training is only 1/4 $\sim$ 1/3 of the FPN.

Center Line Branch (CLB). Experiments (d) and (e) show the effect of CLB to assist in extending the instance area and alleviating the ambiguity of keypoints. The comparison with (c) shows that the branch is very lightweight, only increasing the parameter amount of 0.1M. The proposed Dichotomy Extended Area algorithm (d) can improve 1.1 AP with a slight reduction in speed (1.4 fps). However, using a fixed ideal ratio for all box instances to expand the area (d*) can only increase by 0.3 AP. Alleviating the ambiguity can further increase 0.2 AP without reducing the speed.

Cascade ROIGuider. From the experiment (f), it can be verified that our cascade ROIGuider can further improve accuracy. Since the input feature size of ROIGuider is very low (lower than 24×24), it does not bring a large amount of calculation. The results show that the operation can improve 1.2 AP, only 3.0 fps speed drop.

$ratio$ \ $thre$	0.05	0.06	0.07	0.08	0.09	0.1	0.15
0.05	62.6	62.6	62.6	62.7	62.6	62.6	62.5
0.07	62.8	62.8	62.8	62.9	62.8	62.7	62.6
0.08	62.8	62.8	62.9	**63.0**	62.9	62.8	62.7
0.09	62.8	62.8	**63.0**	**63.0**	62.9	62.9	62.7
0.1	62.7	62.7	62.8	62.9	62.9	62.8	62.7
0.15	62.3	62.4	62.6	62.7	62.6	62.6	62.6

Table 2: The effect of *ratio* and *thre* in different settings. Results are obtained with input size *384* on the TNet-D1W64.

	Method	Params	Speed	AP
(a)	Mask-RCNN (R50-FPN)	56.4M	14.9fps	60.7
(b)	PifPaf (ResNet-50)	24.1M	9.2fps	62.9
(c)	SimplePose + Flip	123.0M	2.2fps	65.8
(d)	HigherHRNet-W32	28.6M	13.7fps	54.1
(e)	+ Adjust_Refine	28.6M	6.3fps	63.6
(f)	+ Flip	28.6M	2.0fps	67.1
(g)	Ours (TNet-D1W64)	26.9M	21.1fps	68.1
(h)	+ Flip	26.9M	12.4fps	69.5
(i)	+ 55 epochs (150 epochs)	26.9M	12.4fps	**70.3**

Table 3: Comparisons of running speed and parameters with state-of-the-art single-model methods on the COCO *val2017* set. HigherHRNet is trained 300 epochs. All results are obtained with input size *512*.

Different Settings in Dichotomy Extended Area

In the Dichotomy Extended Area algorithm, both the max extended range (*ratio*) and the threshold (*thre*) of the center line heatmap are the main parameters that affect the accuracy. Table 2 shows the model accuracy (AP) of *ratio* and *thre* in different settings. In order to reduce the number of iterations, the $ratio$ we set is relatively small. It can be seen from the results that AP first rises and then falls with the increase of *ratio* and *thre*, and the $thre = 0.08$ and $ratio = 0.09$ are a set of ideal parameters. In our experiments, this settings of *ratio* and *thre* are robust under different models and input sizes. It can increase $1.1 \sim 1.4$ AP steadily, without setting them separately for each situation.

Method	Params / FLOPs	Speed / Memory	AP
Hourglass (8-stage)	25.1M / 19.5G	- / -	66.9
Simple (ResNet-50)	34.0M / 4.0G	423sps / 7829M	70.5
HRNet-W32	28.5M / 7.7G	235sps / 10955M	74.5
Ours (D3W64)	24.7M / 7.2G	503sps / 6645M	**74.9**

Table 4: Comparisons with state-of-the-art methods on the HPE separated-model experiments during training. The results are obtained on 3*NVIDIA GTX 2080Ti GPUs with a batch size of 64 per GPU and an input image size of 256×192.

Comparisons of Running Speed and Parameters

We compare the performance of our model in terms of parameters, speed and accuracy with other state-of-art methods (Mask-RCNN (He et al. 2017), PifPaf (Kreiss, Bertoni, and Alahi 2019), SimplePose (Li, Su, and Wang 2020), HigherHRNet (Cheng et al. 2020)) in the same environment. The compared results are shown in Table 3. Our method has a similar pipeline with Mask-RCNN (a). Compared with it, we achieve a considerable improvement of 7.4 AP. However, the amount of parameters (26.9M) is only half of its, and the speed is faster (+6.2 fps). PifPaf's network (b) is lightweight, but the post processing of getting the final keypoints takes a lot of time, accounting for more than 60% of the total time. Compared with it, our method does not require complex post processing and achieves the improvement of 5.2 AP and 2.3x speedup. SimplePose (c) makes improvements to CMU-pose (Cao et al. 2017), but uses a larger backbone. Our method only uses about 1 / 5 of the parameters, and achieves 5.6x speedup and higher accuracy (+3.7 AP). The post processing of HigherHRNet (d $\sim$ f) significantly improves its accuracy, which will lead to a great reduction in speed. For example, the officially provided model trained 300 epochs with input size *512* has the accuracy/speed of 54.1 AP/13.7 fps, 63.6 AP/6.3 fps after Refine operation (identifying missing points), 67.1 AP/2.0 fps after flipping test. Compared with HigherHRNet, the results show that our method is significantly better than it in terms of accuracy and speed, especially in the absence of refining operations (+14.0 AP). It is worth noting that HigherHRNet is trained 300 epochs, and our model is only trained 95 epochs. We further increase the number of epochs (total 150 epochs), and our model can be further improved.

TNet on Separated-Model Experiments

In order to verify the effectiveness of the designed TNet, we follow the general top-down process to conduct the experiments on the separated model. We compare the memory

Method	Backbone	Input Size	AP	AP_{50}	AP_{75}	AP_M	AP_L	AR
separated-model methods								
Integral Pose (Sun et al. 2018)	ResNet-101	256×256	67.8	88.2	74.8	63.9	74.0	-
CPN (Chen et al. 2018)	ResNet-Inception	384×288	72.1	91.4	80.0	68.7	77.2	78.5
RMPE (Fang et al. 2017)	PyraNet (Yang et al. 2017)	320×256	72.3	89.2	79.1	68.0	78.6	-
Simple (Xiao, Wu, and Wei 2018)	ResNet-152	384×288	73.7	91.9	81.1	70.3	80.0	79.0
HRNet-W32 (Sun et al. 2019)	HRNet-W32	384×288	74.9	92.5	82.8	71.3	80.9	80.1
HRNet-W48 (Sun et al. 2019)	HRNet-W48	384×288	75.5	92.5	83.3	71.9	**81.5**	80.5
Ours	TNet-D3W96	384×288	**75.8**	**92.6**	**83.6**	**72.7**	81.4	**81.1**
single-model methods								
CMU-Pose (Cao et al. 2017) *	-	-	61.8	84.9	67.5	57.1	68.2	66.5
Mask R-CNN (He et al. 2017)	ResNet-50-FPN	~ 800	63.1	87.3	68.7	57.8	71.4	-
AE (Newell, Huang, and Deng 2017) *†	Hourglass	~ 512	65.5	86.8	72.3	60.6	72.6	70.2
PifPaf (Kreiss, Bertoni, and Alahi 2019)	ResNet-152	~ 641	66.7	-	-	62.4	72.9	72.2
SPM (Nie et al. 2019) *†	Hourglass	-	66.9	88.5	72.9	62.6	73.1	-
HigherHRNet (Cheng et al. 2020) *	HRNet-W32	~ 512	66.4	87.5	72.8	61.2	74.2	-
HigherHRNet (Cheng et al. 2020) *^{1}	HRNet-W48	~ 512	67.8	**88.3**	74.8	62.5	75.2	72.3
Ours	TNet-D3W96	~ 512	**68.8**	88.0	**75.8**	**64.2**	**76.0**	**78.0**

Table 5: Comparisons between different methods on the COCO *test-dev2017* set. "+" involving ensemble models for inferring. "*" indicates using refinement. "†" means using multi-scale test. "^{1}" using the improved version in *mmpose* (MMPose 2020).

usage and speed of training with the recent state-of-the-art methods(Hourglass (Newell, Yang, and Deng 2016), Simple (Xiao, Wu, and Wei 2018), HRNet (Sun et al. 2019)). For fair comparisons with the previous top-down work, we use the human detection results (55.8 AP) similar to (Xiao, Wu, and Wei 2018; Sun et al. 2019) (56.0 AP).

The compared results are shown in Table 4. Compared to the Simple (ResNet-50) model, our TNet-D3W64 model uses fewer parameters while with an improvement of 4.4 AP, a faster training speed (+80 sps) and a lower memory usage (-1184 MB). Compared with HRNet-W32, our TNet-D3W64 model outperforms it by 0.4 AP but with nearly half the memory consumption and a 2.2x speedup.

Method	Input Size	Params	AP
Mask-RCNN (FPN-50)	~ 384	42.3M	38.6
Mask-RCNN (FPN-50)	~ 512	42.3M	41.6
Ours (TNet-D1W64)	~ 384	20.2M	41.6
Ours (TNet-D1W64)	~ 512	20.2M	**45.1**

Table 6: The results of person instance segmentation on the COCO *val2017* set.

Person Instance Segmentation

We also conduct instance segmentation (person category) experiments to verify the generality of our method. We simply change the number of channels in the output layer to fit the person segmentation task. Therefore, there is no significant change in the complexity of our model. The parameters of training are not refined and most of them are inherited from the pose estimation task. Table 6 shows the results of our method in comparison with the Mask-RCNN baseline. Our models can outperform Mask-RCNN 3.0 $\sim$ 3.5 AP with only half of the parameters. This demonstrates the generality of our model to handle different tasks with competitive performance.

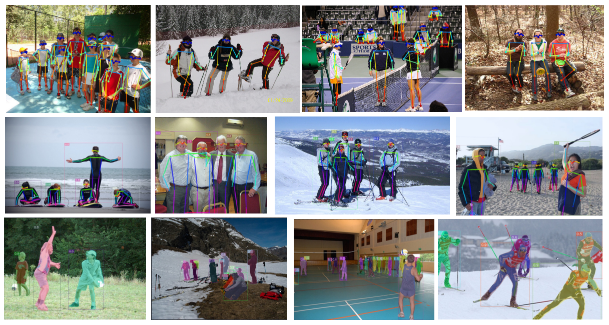


Figure 5: Qualitative results of our model on the COCO dataset (drawn with different color maps). From top to bottom are the results of the separated-model HPE, single-model HPE and person instance segmentation, respectively.

Results on the COCO *tes-dev2017* Set

Table 5 shows the results on the COCO *test-dev2017* set. Most methods use refinement or multi-scale test to improve accuracy. Without bells and whistles, our models can achieve state-of-the-art results on both single-model and separated-model experiments. It is worth noting that our models also have great advantages in speed. We also show some qualitative results on the COCO dataset in Fig. 5.

Conclusion

In this paper, we propose a simple yet efficient pose estimation method, which is equipped with ROIGuider, Center Line Branch (CLB) and TNet. The ROIGuider can enhance the robustness of regressing keypoints. We propose the CLB to solve the *defective box* and *ambiguous box* problems. We further design a new backbone TNet to improve the overall performance of our model. The experiments are conducted on the COCO dataset and the results show that our model can achieve state-of-the-art accuracy and speed.

Acknowledgments

This work is supported by Zhejiang Provincial Natural Science Foundation (LR19F020005), National Natural Science Foundation of China (61572433, 61972347) and thanks for a gift grant from Baidu Inc.

References

Cao, Z.; Simon, T.; Wei, S.-E.; and Sheikh, Y. 2017. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In *CVPR*.

Chen, Y.; Wang, Z.; Peng, Y.; Zhang, Z.; Yu, G.; and Sun, J. 2018. Cascaded pyramid network for multi-person pose estimation. In *CVPR*, 7103–7112.

Cheng, B.; Xiao, B.; Wang, J.; Shi, H.; Huang, T. S.; and Zhang, L. 2020. HigherHRNet: Scale-Aware Representation Learning for Bottom-Up Human Pose Estimation. In *CVPR*.

Chu, X.; Yang, W.; Ouyang, W.; Ma, C.; Yuille, A. L.; and Wang, X. 2017. Multi-context attention for human pose estimation. In *CVPR*, 1831–1840.

Fang, H.; Xie, S.; Tai, Y.-W.; and Lu, C. 2017. Rmpe: Regional multi-person pose estimation. In *ICCV*, volume 2.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *ICCV*, 2980–2988. IEEE.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

Hu, J.; Shen, L.; and Sun, G. 2017. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507* 7.

Huang, J.; Zhu, Z.; Guo, F.; and Huang, G. 2020. The Devil is in the Details: Delving into Unbiased Data Processing for Human Pose Estimation. In *CVPR*, 5700–5709.

Huang, S.; Gong, M.; and Tao, D. 2017. A coarse-fine network for keypoint localization. In *ICCV*, 3028–3037.

Insafutdinov, E.; Pishchulin, L.; Andres, B.; Andriluka, M.; and Schiele, B. 2016. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In *ECCV*, 34–50. Springer.

Jin, S.; Xu, L.; Xu, J.; Wang, C.; Liu, W.; Qian, C.; Ouyang, W.; and Luo, P. 2020. Whole-Body Human Pose Estimation in the Wild. *arXiv preprint arXiv:2007.11858* .

Ke, L.; Chang, M.-C.; Qi, H.; and Lyu, S. 2018. Multi-scale structure-aware network for human pose estimation. In *ECCV*, 713–728.

Kreiss, S.; Bertoni, L.; and Alahi, A. 2019. Pifpaf: Composite fields for human pose estimation. In *CVPR*, 11977–11986.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1097–1105.

Law, H.; and Deng, J. 2018. Cornernet: Detecting objects as paired keypoints. In *ECCV*, 734–750.

Li, J.; Su, W.; and Wang, Z. 2020. Simple Pose: Rethinking and Improving a Bottom-up Approach for Multi-Person Pose Estimation. In *AAAI*, 11354–11361.

Lin, T.-Y.; Dollár, P.; Girshick, R. B.; He, K.; Hariharan, B.; and Belongie, S. J. 2017. Feature Pyramid Networks for Object Detection. In *CVPR*, volume 1, 4.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2018. Focal loss for dense object detection. *TPAMI* .

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *ECCV*, 740–755. Springer.

MMPose. 2020. OpenMMLab Pose Estimation Toolbox and Benchmark. https://github.com/open-mmlab/mmpose.

Moon, G.; Chang, J. Y.; and Lee, K. M. 2019. Posefix: Model-agnostic general human pose refinement network. In *CVPR*, 7773–7781.

Newell, A.; Huang, Z.; and Deng, J. 2017. Associative embedding: End-to-end learning for joint detection and grouping. In *NIPS*, 2277–2287.

Newell, A.; Yang, K.; and Deng, J. 2016. Stacked hourglass networks for human pose estimation. In *ECCV*, 483–499. Springer.

Nie, X.; Feng, J.; Zhang, J.; and Yan, S. 2019. Single-stage multi-person pose machines. In *ICCV*, 6951–6960.

Papandreou, G.; Zhu, T.; Kanazawa, N.; Toshev, A.; Tompson, J.; Bregler, C.; and Murphy, K. 2017. Towards accurate multi-person pose estimation in the wild. In *CVPR*, volume 3, 6.

Shi, W.; Caballero, J.; Huszár, F.; Totz, J.; Aitken, A. P.; Bishop, R.; Rueckert, D.; and Wang, Z. 2016. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 1874–1883.

Sun, K.; Xiao, B.; Liu, D.; and Wang, J. 2019. Deep High-Resolution Representation Learning for Human Pose Estimation. In *CVPR*.

Sun, X.; Xiao, B.; Wei, F.; Liang, S.; and Wei, Y. 2018. Integral human pose regression. In *ECCV*, 529–545.

Tang, W.; Yu, P.; and Wu, Y. 2018. Deeply learned compositional models for human pose estimation. In *ECCV*, 190–206.

Xiao, B.; Wu, H.; and Wei, Y. 2018. Simple baselines for human pose estimation and tracking. In *ECCV*, 466–481.

Yang, W.; Li, S.; Ouyang, W.; Li, H.; and Wang, X. 2017. Learning feature pyramids for human pose estimation. In *ICCV*, 1281–1290.

Zhang, F.; Zhu, X.; Dai, H.; Ye, M.; and Zhu, C. 2020. Distribution-aware coordinate representation for human pose estimation. In *CVPR*, 7093–7102.

Zhang, Z.; Tang, J.; and Wu, G. 2019. Simple and Lightweight Human Pose Estimation. *arXiv preprint arXiv:1911.10346* .

Zhou, X.; Wang, D.; and Krähenbühl, P. 2019. Objects as points. *arXiv preprint arXiv:1904.07850* .