RTMPose: Real-Time Multi-Person Pose Estimation based on MMPose

Tao Jiang Peng Lu Li Zhang Ningsheng Ma Rui Han
Chengqi Lyu Yining Li Kai Chen

Shanghai AI Laboratory

{jiangtao, lupeng, zhangli, maningsheng, hanrui, lvchengqi, liyining, chenkai}@pjlab.org.cn

Abstract

Recent studies on 2D pose estimation have achieved excellent performance on public benchmarks, yet its application in the industrial community still suffers from heavy model parameters and high latency. To bridge this gap, we empirically explore key factors in pose estimation including paradigm, model architecture, training strategy, and deployment, and present a high-performance real-time multi-person pose estimation framework, ***RTMPose****, based on MMPose. Our RTMPose-m achieves* ***75.8% AP on COCO*** *with* ***90+ FPS*** *on an Intel i7-11700 CPU and* ***430+ FPS*** *on an NVIDIA GTX 1660 Ti GPU, and RTMPose-x achieves* ***65.3% AP on COCO-WholeBody****. To further evaluate RTMPose's capability in critical real-time applications, we also report the performance after deploying on the mobile device. Our RTMPose-s model achieves* ***72.2% AP on COCO*** *with* ***70+ FPS*** *on a Snapdragon 865 chip, outperforming existing open-source libraries. Our code and models are available at* https://github.com/open-mmlab/mmpose/tree/main/projects/rtmpose.

1. Introduction

Real-time human pose estimation is appealing to various applications such as human-computer interaction, action recognition, sports analysis, and VTuber techniques. Despite the stunning progress [52, 61] on academic benchmarks [31, 37], it remains a challenging task to perform robust and real-time multi-person pose estimation on devices with limited computing power. Recent attempts narrow the gap with efficient network architecture [3, 56, 63] and detection-free paradigms [19, 29, 51], which is yet inadequate to reach satisfactory performance for industrial applications.

In this work, we empirically study key factors that affect the performance and latency of 2D multi-person pose estimation frameworks from five aspects: paradigm, backbone network, localization method, training strategy, and deployment. With a collection of optimizations, we introduce **RTMPose**, a new series of **R**eal-**T**ime **M**odels for **Pose** estimation.

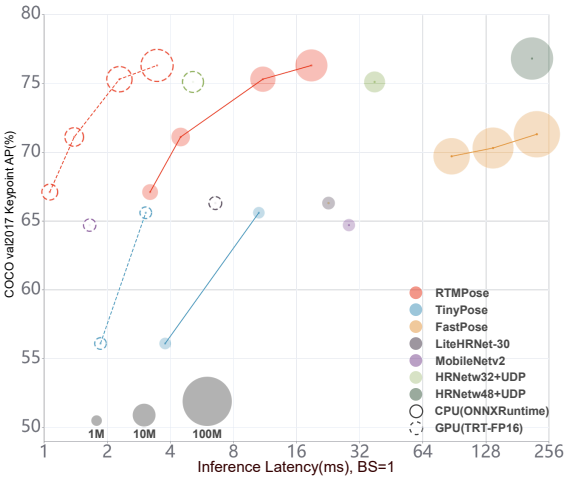

Figure 1. Comparison of RTMPose and open-source libraries on COCO val set regarding model size, latency, and precision. The circle size represents the relative size of model parameters.

First, RTMPose employs a top-down approach by using an off-the-shelf detector to obtain bounding boxes and then estimating the pose of each person individually. Top-down algorithms have been stereotyped as accurate but slow, due to the extra detection process and increasing workload in crowd scenes. However, benefiting from the excellent efficiency of real-time detectors [42, 46], the detection part is no longer a bottleneck of the inference speed of top-down methods. In most scenarios (within 6 persons per image), the proposed lightweight pose estimation network is able to perform multiple forward passes for all instances in real time.

Second, RTMPose adopts CSPNeXt [42] as the backbone, which is first designed for object detection. Backbones designed for image classification [20, 49] are suboptimal for dense prediction tasks like object detection, pose estimation and semantic segmentation, etc. Some back-

bones leveraging high-resolution feature maps [52, 63] or advanced transformer architectures [15] achieve high accuracy on public pose estimation benchmarks, but suffer from high computational cost, high inference latency, or difficulties in deployment. CSPNeXt shows a good balance of speed and accuracy and is deployment-friendly.

Third, RTMPose predicts keypoints using a SimCC-based [35] algorithm that treats keypoint localization as a classification task. Compared with heatmap-based algorithms [24, 59, 61, 65], the SimCC-based algorithm achieves competitive accuracy with lower computational effort. Moreover, SimCC uses a very simple architecture of two fully-connected layers for prediction, making it easy to deploy on various backends.

Fourth, we revisit the training settings in previous works [3, 30, 42], and empirically introduce a collection of training strategies applicable to the pose estimation task. Our experiments demonstrate that this set of strategies bring significant gains to proposed RTMPose as well as other pose estimation models.

Finally, we jointly optimize the inference pipeline of the pose estimation framework. We use the skip-frame detection strategy proposed in [3] to reduce the latency and improve the pose-processing with pose Non-Maximum Suppression (NMS) and smoothing filtering for better robustness. In addition, we provide a series of RTMPose models with t/s/m/l/x sizes to cover different application scenarios with the optimum performance-speed trade-off.

We deploy RTMPose with different inference frameworks (PyTorch, ONNX Runtime, TensorRT, ncnn) and hardwares (i7-11700, GTX1660Ti, Snapdragon865) to test the efficiency.

As shown in Fig. 1, We evaluate the efficiency of RTMPose with various inference frameworks (PyTorch, ONNX Runtime, TensorRT, ncnn) and hardwares (Intel i7-11700, GTX 1660Ti, Snapdragon 865). Our RTMPose-m achieves 75.8% AP (with flipping) on COCO val set with 90+ FPS on an Intel i7-11700 CPU, 430+ FPS on an NVIDIA GeForce GTX 1660 Ti GPU, and 35+ FPS on a Snapdragon 865 chip. Using the high-performance real-time object detection model RTMDet-nano in our pose estimation pipeline, RTMPose-m can achieve 73.2% AP. With the help of MMDeploy [12], RTMPose can also be easily deployed to various backends like RKNN, OpenVINO, PPLNN, etc.

2. Related Work

Bottom-up Approaches. Bottom-up algorithms [7, 10, 19, 27, 29, 41, 44, 45] detect instance-agnostic keypoints in an image and partition these keypoints to obtain the human pose. The bottom-up paradigm is considered suitable for crowd scenarios because of the stable computational cost regardless the number of people increases. However, these algorithms often require a large input resolution to handle various person scales, making it challenging to reconcile accuracy and inference speed.

Top-down Approaches. Top-down algorithms use off-the-shelf detectors to provide bounding boxes and then crop the human to a uniform scale for pose estimation. Algorithms [5, 38, 52, 59, 61] of the top-down paradigm have been dominating public benchmarks. The two-stage inference paradigm allows both the human detector and the pose estimator to use relatively small input resolutions, which allows them to outperform bottom-up algorithms in terms of speed and accuracy in non-extreme scenarios (i.e. when the number of people in the image is no more than 6). Additionally, most previous work has focused on achieving state-of-the-art performance on public datasets, while our work aims to design models with better speed-accuracy trade-offs to meet the needs of industrial applications.

Coordinate Classification. Previous pose estimation approaches usually regard keypoint localization as either coordinate regression (e.g. [30, 43, 54]) or heatmap regression (e.g. [25, 59, 61, 65]). SimCC [35] introduces a new scheme that formulates keypoint prediction as classification from sub-pixel bins for horizontal and vertical coordinates respectively, which brings about several advantages. First, SimCC is freed from the dependence on high-resolution heatmaps, thus allowing for a very compact architecture that requires neither high-resolution intermediate representations [52] nor costly upscaling layers [59]. Second, SimCC flattens the final feature map for classification instead of involving global pooling [54] and therefore avoids the loss of spatial information. Third, the quantization error can be effectively alleviated by coordinate classification at the sub-pixel scale, without the need for extra refinement post-processing [65]. These qualities make SimCC attractive for building lightweight pose estimation models. In this work, we further exploit the coordinate classification scheme with optimizations on model architecture and training strategy.

Vision Transformers. Transformer-based architectures [55] ported from modern Natural Language Processing (NLP) have achieved great success in various vision tasks like representation learning [15, 39], object detection [8, 34, 67], semantic segmentation [66], video understanding [4, 17, 40], as well as pose estimation [36, 43, 51, 61, 62]. ViTPose [61] leverages the state-of-the-art transformer backbones to boost pose estimation accuracy, while TransPose [62] integrates transformer encoders with CNNs to efficiently capture long-range spatial relationships. Token-based keypoint embedding is introduced to incorporate visual cue querying and anatomic constraint learning, shown effective in both

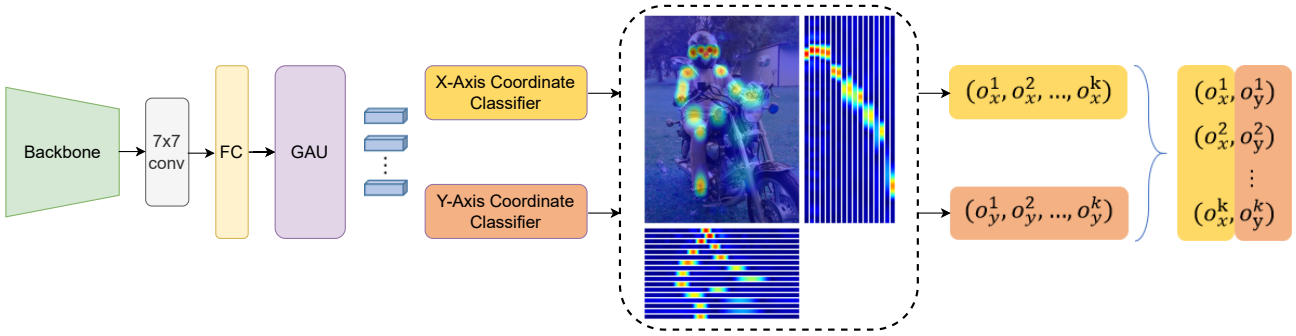

Figure 2. The overall architecture of RTMPose, which contains a convolutional layer, a fully-connected layer and a Gated Attention Unit (GAU) to refine K keypoint representations. After that 2d pose estimation is regarded as two classification tasks for x-axis and y-axis coordinates to predict the horizontal and vertical locations of keypoints.

heatmap-based [36] and regression-based [43] approaches. PRTR [33] and PETR [51] propose end-to-end multi-person pose estimation frameworks with transformers, inspired by the pioneer in detection [8]. Previous pose estimation approaches with transformers either use a heatmap-based representation or retained both pixel tokens and keypoint tokens, which results in high computation costs and makes real-time inference difficult. In contrast, we incorporate the self-attention mechanism with a compact SimCC-based representation to capture the keypoint dependencies, which significantly reduces the computation load and allows real-time inference with advanced accuracy and efficiency.

3. Methodology

In this section, we expound the roadmap we build RTMPose following the coordinate classification paradigm. We start by refitting SimCC [35] with more efficient backbone architectures, which gives a lightweight yet strong baseline (3.1). We adopt the training strategies proposed in [42] with minor tweaks to make them more effective on the pose estimation task. The model performance is further improved with a series of delicate modules (3.3) and micro designs (3.4). Finally, we jointly optimize the entire top-down inference pipeline toward higher speed and better reliability. The final model architecture is shown in Fig. 2, and Fig. 3 illustrates the step-by-step gain of the roadmap.

3.1. SimCC: A lightweight yet strong baseline

Preliminary SimCC [35] formulates the keypoint localization as a classification problem. The core idea is to divide the horizontal and vertical axes into equal-width numbered bins and discretize continuous coordinates into integral bin labels. Then the model is trained to predict the bin in which the keypoint is located. The quantization error can be reduced to a subpixel level by using a large number of bins.

Thanks to this novel formulation, SimCC has a very simple structure that uses a 1×1 convolution layer to convert features extracted by the backbone into vectorized keypoint representations, and two fully-connected layers to perform classification, respectively.

Inspired by label smoothing in traditional classification tasks [53], SimCC proposes a Gaussian label smoothing strategy that replaces the one-hot label with Gaussian distributed soft label centered at the ground-truth bin, which integrates the inductive bias in the model training and brings about significant performance improvement. We find this technique also coincides with the idea of SORD [16] in the ordinal regression task. The soft label naturally encapsulates the rank likelihoods of the keypoint locations given the inter-class penalty distance defined by the label distribution.

Baseline We first remove the costly upsampling layers from the standard SimCC. Results in Table 1 show that the trimmed SimCC has significantly lower complexity compared to the SimCC and heatmap-based baselines [59], and still achieves promising accuracy. This indicates the efficiency of encoding global spatial information into disentangled one-dimension representations in localization tasks. By replacing the ResNet-50 [21] backbone with the more compact CSPNext-m [42], we further reduce the model size and obtain a lightweight yet strong baseline, 69.7% AP.

Table 1. Computational costs and accuracy of baseline methods. We show FLOPs and model parameters of prediction heads for a detailed comparison. "SimCC*" denotes the removal of upsampling layers from the standard SimCC head.

	Heatmap	SimCC	SimCC*
Repr. Size	64×48	512+384	512+384
AP	71.8	72.1	71.3
Total FLOPs(G)	5.45	5.50	4.03
Total Params(M)	34.00	36.75	23.59
Head FLOPs(G)	1.425	1.472	0.002
Head Params(M)	10.492	13.245	0.079

3.2. Training Techniques

Pre-training Previous works [3, 30] show that pre-training the backbone using the heatmap-based method can improve the model accuracy. We adopt UDP [24] method for the backbone pre-training. This improves the model from 69.7% AP to 70.3% AP. We use this technique as a default setting in the following sections.

Optimization Strategy We adopt the optimization strategy from [42]. The Exponential Moving Average (EMA) is used for alleviating overfitting (70.3% to 70.4%). The flat cosine annealing strategy improves the accuracy to 70.7% AP. We also inhibit weight decay on normalization layers and biases.

Two-stage training augmentations Following the training strategy in [42], we use a strong-then-weak two-stage augmentation. First using strong data augmentations to train 180 epochs and then a weak strategy for 30 epochs. During the strong stage, we use a large random scaling range [0.6, 1.4], and a large random rotation factor, 80, and set the Cutout [14] probability to 1. According to AID [26], Cutout helps to prevent the model from overfitting to the image textures and encourages it to learn the pose structure information. In the weak strategy stage, we turn off the random shift, use a small random rotation range, and set the Cutout probability to 0.5 to allow the model to fine-tune in a domain that more closely matches the real image distribution.

3.3. Module Design

Feature dimension We observe that the model performance increases along with higher feature resolution. Therefore, we use a fully connected layer to expand the 1D keypoint representations to a desired dimension controlled by the hyper-parameter. In this work, we use 256 dimensions and the accuracy is improved from 71.2% AP to 71.4% AP.

Self-attention module To further exploit the global and local spatial information, we refine the keypoint representations with a self-attention module, inspired by [36, 62]. We adopt the transformer variant, Gated Attention Unit (GAU) [23], which has faster speed, lower memory cost, and better performance compared to the vanilla transformer [55]. Specifically, GAU improves the Feed-Forward Networks (FFN) in the transformer layer with Gated Linear Unit (GLU) [50], and integrates the attention mechanism in an elegant form:

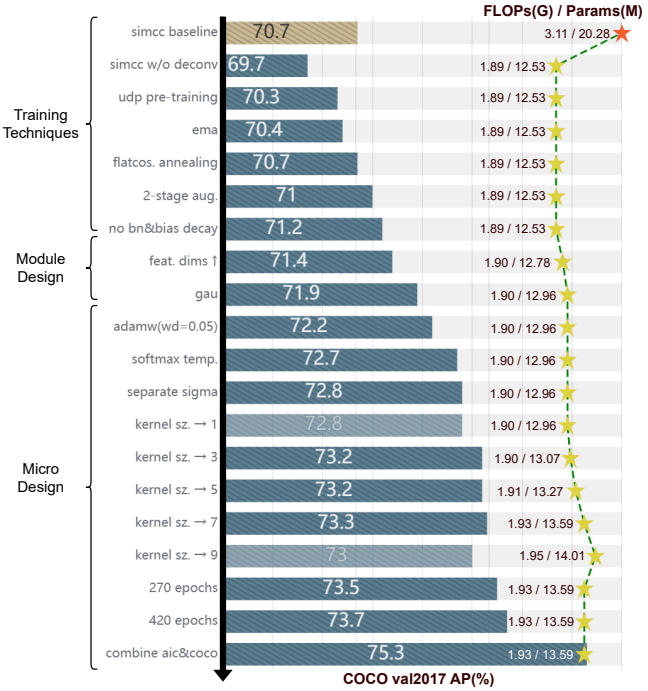

Figure 3. Step-by-step improvements from a SimCC baseline.

$$
\begin{aligned}
U &= \phi_u(XW_u) \\
V &= \phi_v(XW_v) \\
O &= (U \odot AV)W_o
\end{aligned} \tag{1}
$$

where $\odot$ is the pairwise multiplication (Hadamard product) and ϕ is the activation function. In this work we implement the self-attention as follows:

$$
A = \frac{1}{n}\, relu^2\left(\frac{Q(X)K(Z)^\top}{\sqrt{s}}\right), Z = \phi_z(XW_z) \tag{2}
$$

where $s = 128$, Q and K are simple linear transformations, and $relu^2(\cdot)$ is ReLU then squared. This self-attention module brings about a 0.5% AP (71.9%) improvement to the model performance.

3.4. Micro Design

Loss function We treat the coordinate classification as an ordinal regression task and follow the soft label encoding proposed in SORD [16]:

$$
y_i = \frac{e^{\phi(r_t, r_i)}}{\sum_{k=1}^{K} e^{\phi(r_t, r_k)}} \tag{3}
$$

where $\phi(r_t, r_i)$ is a metric loss function of our choice that penalizes how far the true metric value of r_t is from

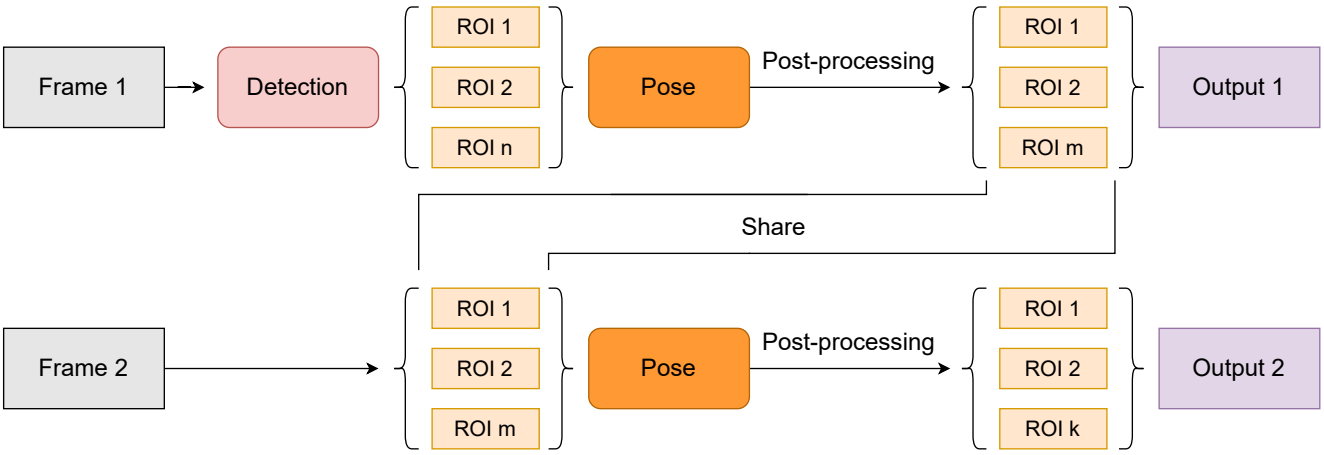

Figure 4. Inference pipeline of RTMPose.

the rank $r_i \in Y$. In this work, we adopt the unnormalized Gaussian distribution as the inter-class distance metric:

$$\phi(r_t, r_i) = e^{\frac{-(r_t - r_i)^2}{2\sigma^2}} \tag{4}$$

Note that Eq. 3 can be seen as computing Softmax for all $\phi(r_t, r_i)$. We add temperatures in the Softmax operation for both model outputs and soft labels further adjust the normalized distribution shape:

$$y_i = \frac{e^{\phi(r_t, r_i)/\tau}}{\sum_{k=1}^{K} e^{\phi(r_t, r_l)/\tau}} \tag{5}$$

According to the experimental results, using $\tau = 0.1$ can improve the performance from 71.9% to 72.7%.

Separate σ In SimCC, the horizontal and vertical labels are encoded using the same σ. We empirically explore a simple strategy to set separate σ for them:

$$\sigma = \sqrt{\frac{W_S}{16}} \tag{6}$$

where W_S is the bin number in the horizontal and vertical directions respectively. This step improves the accuracy from 72.7% to 72.8%.

Larger convolution kernel We experiment with different kernel sizes of the last convolutional layer and find that using a larger kernel size gives a performance improvement over using 1×1 kernel. Finally, we chose to use a 7×7 convolutional layer, which achieves 73.3% AP. We compare model performances with different kernel sizes in Table 2. Additionally, we also compare the effect of different temperature factors in Table 3 using the final model architecture.

Table 2. Comparison of different kernel sizes

Kernel Size	mAP
1x1	72.8
3x3	73.2
5x5	73.2
7x7	73.3
9x9	73.0

Table 3. Comparison of different temperature factors.

$1/\tau$	mAP
1	unstable
5	73.1
10	73.3
15	73.0

More epochs and multi-dataset training Increasing the training epochs brings extra gains to the model performance. Specifically, training 270 and 420 epochs reach 73.5% AP and 73.7% AP respectively. To further exploit the model's potential, we enrich the training data by combining COCO [37] and AI Challenger [58] datasets together for pre-training and fine-tuning, with a balanced sampling ratio. The performance finally achieves 75.3% AP.

3.5. Inference pipeline

Beyond the pose estimation model, we further optimize the overall top-down inference pipeline for lower latency and better robustness. We use the skip-frame detection mechanism as in BlazePose [3], where human detection is performed every K frames, and in the interval frames the bounding boxes are generated from the last pose estimation results. Additionally, to achieve smooth prediction over frames, we use OKS-based pose NMS and OneEuro [9] filter in the post-processing stage.

4. Experiments

4.1. Settings

The training settings in our experiments are shown in Tabel. 7. As described in Sec. 3.2, we conduct a heatmap-based pre-training [24] which follows the same training

Table 4. Body pose estimation results on COCO validation set. We only report GFLOPs of pose model and ignore the detection model. Flip test is not used.

Methods		Backbone	Detector	Det. Input Size	Pose Input Size	GFLOPs	AP	Extra Data
PaddleDetection [2]	TinyPose	Wider NLiteHRNet	YOLOv3	608 × 608	128 × 96	0.08	52.3	AIC(220K) +Internal(unknown)
	TinyPose	Wider NLiteHRNet	YOLOv3	608 × 608	256 × 192	0.33	60.9	
	TinyPose	Wider NLiteHRNet	Faster-RCNN	N/A	128 × 96	0.08	56.1	
	TinyPose	Wider NLiteHRNet	Faster-RCNN	N/A	256 × 192	0.33	65.6	
	TinyPose	Wider NLiteHRNet	PicoDet-s	320 × 320	128 × 96	0.08	48.4	
	TinyPose	Wider NLiteHRNet	PicoDet-s	320 × 320	256 × 192	0.33	56.5	
AlphaPose [18]	FastPose	ResNet 50	YOLOv3	608 × 608	256 × 192	5.91	71.2	-
	FastPose(DUC)	ResNet-50	YOLOv3	608 × 608	256 × 192	9.71	71.7	
	FastPose(DUC)	ResNet-152	YOLOv3	608 × 608	256 × 192	15.99	72.6	
	FastPose	ResNet 50	Faster-RCNN	N/A	256 × 192	5.91	69.7	
	FastPose(DUC)	ResNet-50	Faster-RCNN	N/A	256 × 192	9.71	70.3	
	FastPose(DUC)	ResNet-152	Faster-RCNN	N/A	256 × 192	15.99	71.3	
MMPose [11]	RTMPose-t	CSPNeXt-t	Faster-RCNN	N/A	256 × 192	0.36	65.8	-
	RTMPose-s	CSPNeXt-s	Faster-RCNN	N/A	256 × 192	0.68	69.6	
	RTMPose-m	CSPNeXt-m	Faster-RCNN	N/A	256 × 192	1.93	73.6	
	RTMPose-l	CSPNeXt-l	Faster-RCNN	N/A	256 × 192	4.16	74.8	
	RTMPose-t	CSPNeXt-t	YOLOv3	608 × 608	256 × 192	0.36	66.0	AIC(220K)
	RTMPose-s	CSPNeXt-s	YOLOv3	608 × 608	256 × 192	0.68	70.3	
	RTMPose-m	CSPNeXt-m	YOLOv3	608 × 608	256 × 192	1.93	74.7	
	RTMPose-l	CSPNeXt-l	YOLOv3	608 × 608	256 × 192	4.16	75.7	
	RTMPose-t	CSPNeXt-t	Faster-RCNN	N/A	256 × 192	0.36	67.1	
	RTMPose-s	CSPNeXt-s	Faster-RCNN	N/A	256 × 192	0.68	71.1	
	RTMPose-m	CSPNeXt-m	Faster-RCNN	N/A	256 × 192	1.93	75.3	
	RTMPose-l	CSPNeXt-l	Faster-RCNN	N/A	256 × 192	4.16	76.3	
	RTMPose-t	CSPNeXt-t	PicoDet-s	320 × 320	256 × 192	0.36	64.3	
	RTMPose-s	CSPNeXt-s	PicoDet-s	320 × 320	256 × 192	0.68	68.8	
	RTMPose-m	CSPNeXt-m	PicoDet-s	320 × 320	256 × 192	1.93	73.2	
	RTMPose-l	CSPNeXt-l	PicoDet-s	320 × 320	256 × 192	4.16	74.2	
	RTMPose-t	CSPNeXt-t	RTMDet-nano	320 × 320	256 × 192	0.36	64.4	
	RTMPose-s	CSPNeXt-s	RTMDet-nano	320 × 320	256 × 192	0.68	68.5	
	RTMPose-m	CSPNeXt-m	RTMDet-nano	320 × 320	256 × 192	1.93	73.2	
	RTMPose-l	CSPNeXt-l	RTMDet-nano	320 × 320	256 × 192	4.16	74.2	
	RTMPose-m	CSPNeXt-m	RTMDet-m	640 × 640	256 × 192	1.93	75.7	
	RTMPose-l	CSPNeXt-l	RTMDet-m	640 × 640	256 × 192	4.16	76.6	

strategies used in the fine-tuning except for shorter epochs. All our models are trained on 8 NVIDIA A100 GPUs. And we evaluate the model performance by mean Average Precision (AP).

4.2. Benchmark Results

COCO COCO [37] is the most popular benchmark for 2d body pose estimation. We follow the standard splitting of `train2017` and `val2017`, which contains 118K and 5k images for training and validation respectively. We extensively study the pose estimation performance with different off-the-shelf detectors including YOLOv3 [47], Faster-RCNN [48], and RTMDet [42]. To conduct a fair comparison with AlphaPose [18] which doesn't use extra training data, we also report the performance of RTMPose only trained on COCO. As shown in Table 4, RTMPose outperforms competitors by a large margin with much lower complexity and shows strong robustness for detection.

COCO-SinglePerson Popular pose estimation open-source algorithms like BlazePose [3], MoveNet [56], and PaddleDetection [2] are designed primarily for single-person or sparse scenarios, which are practical in mobile applications and human-machine interactions. For a fair comparison, we construct a COCO-SinglePerson dataset that contains 1045 single-person images from the COCO `val2017` set to evaluate RTMPose as well as other approaches. For MoveNet, we follow the official inference pipeline to apply a cropping algorithm, namely using the coarse pose prediction of the first inference to crop the input image and performing a second inference for better pose estimation results. The evaluation results in Table 5 show that RTMPose archives superior performance and efficiency even compared to previous solutions tailored for the single-person scenario.

COCO-WholeBody We also validate the proposed RTMPose model on the whole-body pose estimation task with

Table 5. Body pose estimation results on COCO-SinglePerson validation set. We sum up top-down methods' GFLOPs of detection and pose for a fair comparison with bottom-up methods. "*" denotes double inference. Flip test is not used.

Methods		Backbone	Detector	Det. Input Size	Pose Input Size	GFLOPs	AP	Extra Data
MediaPipe [3]	BlazePose-Lite	BlazePose	N/A	256 × 256	N/A	N/A	29.3	Internal(85K)
	BlazePose-Full	BlazePose	N/A	256 × 256	N/A	N/A	35.4	
MoveNet [56]	Lightning	MobileNetv2	N/A	192 × 192	N/A	0.54	53.6*	Internal(23.5K)
	Thunder	MobileNetv2 depth×1.75	N/A	256 × 256	N/A	2.44	64.8*	
PaddleDetection [2]	TinyPose	Wider NLiteHRNet	PicoDet-s	320 × 320	128 × 96	0.55	58.6	AIC(220K)
	TinyPose	Wider NLiteHRNet	PicoDet-s	320 × 320	256 × 192	0.80	69.4	+Internal(unknown)
MMPose [11]	RTMPose-t	CSPNeXt-t	RTMDet-nano	320 × 320	256 × 192	0.67	72.1	AIC(220K)
	RTMPose-s	CSPNeXt-s	RTMDet-nano	320 × 320	256 × 192	0.91	77.1	
	RTMPose-m	CSPNeXt-m	RTMDet-nano	320 × 320	256 × 192	2.23	82.4	
	RTMPose-l	CSPNeXt-l	RTMDet-nano	320 × 320	256 × 192	4.47	83.5	

Table 6. Whole-body pose estimation results on COCO-WholeBody [28, 60] V1.0 dataset. We only report the input size and GFLOPs of pose models in top-down approaches and ignore the detection model. "*" denotes the model is pre-trained on AIC+COCO. "†" indicates multi-scale testing. Flip test is used.

	Method	Input Size	GFLOPs	whole-body		body		foot		face		hand	
				AP	AR	AP	AR	AP	AR	AP	AR	AP	AR
Whole-body	SN† [22]	N/A	272.3	32.7	45.6	42.7	58.3	9.9	36.9	64.9	69.7	40.8	58.0
	OpenPose [6]	N/A	451.1	44.2	52.3	56.3	61.2	53.2	64.5	76.5	84.0	38.6	43.3
Bottom-up	PAF† [7]	512×512	329.1	29.5	40.5	38.1	52.6	5.3	27.8	65.6	70.1	35.9	52.8
	AE [44]	512×512	212.4	44.0	54.5	58.0	66.1	57.7	72.5	58.8	65.4	48.1	57.4
Top-down	DeepPose [54]	384×288	17.3	33.5	48.4	44.4	56.8	36.8	53.7	49.3	66.3	23.5	41.0
	SimpleBaseline [59]	384×288	20.4	57.3	67.1	66.6	74.7	63.5	76.3	73.2	81.2	53.7	64.7
	HRNet [52]	384×288	16.0	58.6	67.4	70.1	77.3	58.6	69.2	72.7	78.3	51.6	60.4
	PVT [57]	384×288	19.7	58.9	68.9	67.3	76.1	66.0	79.4	74.5	82.2	54.5	65.4
	FastPose50-dcn-si [18]	256×192	6.1	59.2	66.5	70.6	75.6	70.2	77.5	77.5	82.5	45.7	53.9
	ZoomNet [28]	384×288	28.5	63.0	74.2	**74.5**	**81.0**	60.9	70.8	88.0	92.4	57.9	73.4
	ZoomNAS [60]	384×288	18.0	**65.4**	**74.4**	74.0	80.7	61.7	71.8	88.9	**93.0**	**62.5**	**74.0**
	RTMPose-m*	256×192	2.2	58.2	67.4	67.3	75.0	61.5	75.2	81.3	87.1	47.5	58.9
	RTMPose-l*	256×192	4.5	61.1	70.0	69.5	76.9	65.8	78.5	83.3	88.7	51.9	62.8
	RTMPose-l*	384×288	10.1	64.8	73.0	71.2	78.1	**69.3**	**81.1**	88.2	91.9	57.9	67.7
	RTMPose-x	384×288	18.1	65.2	73.2	71.2	78.0	68.1	80.4	**89.0**	92.2	59.3	68.7
	RTMPose-x*	384×288	18.1	65.3	73.3	71.4	78.4	69.2	81.0	88.9	92.3	59.0	68.5

Table 7. Training settings for RTMPose models.

optimizer	AdamW
base learning rate	0.004
learning rate schedule	Flat-Cosine
batch size	1024
warm-up iterations	1000
weight decay	0.05 (RTMPose-m/l) 0 (RTMPose-t/s)
EMA decay	0.9998 (RTMPose-s/m/l) no EMA (RTMPose-t)
training epochs	210 (pre-train) 420 (fine-tune)

COCO-WholeBody [28, 60] V1.0 dataset. As shown in Table 6, RTMPose achieves superior performance and well balances accuracy and complexity. Specifically, our RTMPose-m model outperforms previous open-source libraries [6, 18, 63] with significantly lower GFLOPs. And by increasing the input resolution and training data we obtain competitive accuracy with SOTA approaches [28, 60].

Other Datasets As shown in Table 8 and Table 9, we further evaluate RTMPose on AP-10K [64], CrowdPose [32] and MPII [1] datasets. We report the model performance using ImageNet [13] pre-training for a fair comparison with baselines. Besides we also report the performance of our models pre-trained using a combination of COCO [37] and AI Challenger (AIC) [58], which achieves higher accuracy and can be easily reproduced by users with our provided pre-trained weights.

4.3. Inference Speed

We perform the export, deployment, inference, and testing of models by MMDeploy [12] to test the inference speed on CPU and GPU respectively. Table 10 demonstrates the

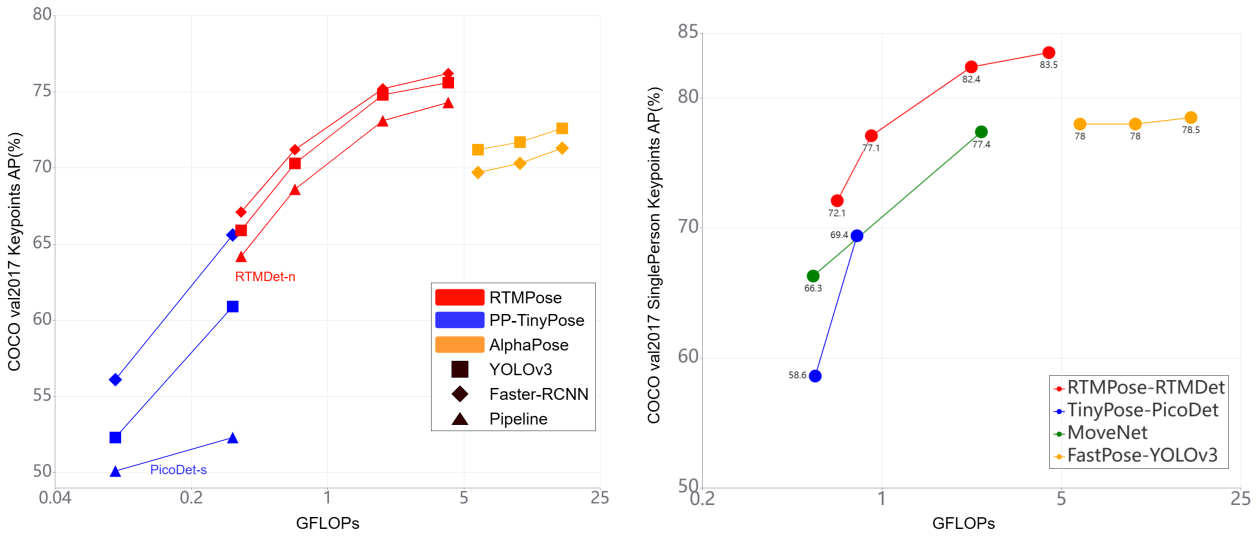

Figure 5. Comparison of GFLOPs and accuracy. Left: Comparison of RTMPose and other open-source pose estimation libraries on full COCO val set. Right: Comparison of RTMPose and other open-source pose estimation libraries on COCO-SinglePerson val set.

comparison of inference speed on the mobile device. We deploy RTMPose on the Snapdragon 865 chip with ncnn and inference with 4 threads. The TensorRT inference latency is tested in the half-precision floating-point format (FP16) on an NVIDIA GeForce GTX 1660 Ti GPU, and the ONNX latency is tested on an Intel I7-11700 CPU with ONNXRuntime with 1 thread. The inference batch size is 1. All models are tested on the same devices with 50 times warmup and 200 times inference for fair comparison. For TinyPose [2], we test it with both MMDeploy and FastDeploy, and note that ONNXRuntime speed on MMDeploy is slightly faster (10.58 ms vs. 12.84 ms). The results are shown in Table 11 and Table 12.

Table 8. Performance on different datasets. "*" denotes the model is pre-trained on AIC+COCO and fine-tuned on the corresponding dataset. Flip test is used.

Dataset	Methods	Backbone	Input Size	GFLOPs	AP
AP-10K [64]	SimpleBaseline [59]	ResNet-50	256 × 256	7.28	68.0
	HRNet [52]	HRNet-w32	256 × 256	10.27	72.2
	RTMPose-m	CSPNeXt-m	256 × 256	**2.57**	**68.4**
	RTMPose-m*	CSPNeXt-m	256 × 256	**2.57**	**72.2**
CrowdPose [32]	SimpleBaseline [59]	ResNet-50	256 × 192	5.46	63.7
	HRNet [52]	HRNet-w32	256 × 192	7.7	67.5
	RTMPose-m	CSPNeXt-m	256 × 192	**1.93**	**66.9**
	RTMPose-m*	CSPNeXt-m	256 × 192	**1.93**	**70.6**

Table 9. Comparison on MPII [1] validation set. "*" denotes the model is pre-trained on AIC+COCO and fine-tuned on MPII. Flip test is used.

Dataset	Methods	Backbone	Input Size	GFLOPs	PCKh@0.5
MPII [1]	SimpleBaseline [59]	ResNet-50	256 × 256	7.28	88.2
	HRNet [52]	HRNet-w32	256 × 256	10.27	90.0
	SimCC [35]	HRNet-w32	256 × 256	10.34	90.0
	TokenPose [36]	L/D24	256 × 256	11.0	90.2
	RTMPose-m	CSPNeXt-m	256 × 256	**2.57**	**88.9**
	RTMPose-m*	CSPNeXt-m	256 × 256	**2.57**	**90.7**

Table 10. Comparison of inference speed on Snapdragon 865. RTMPose models are deployed and tested using ncnn.

Methods		Input Size	GFLOPs	AP(GT)	FP32(ms)	FP16(ms)
PaddleDetection [2]	TinyPose	128 × 96	0.08	58.4	4.57	3.27
	TinyPose	256 × 192	0.33	68.3	14.07	8.33
MMPose [11]	RTMPose-t	256 × 192	0.36	68.4	15.84	9.02
	RTMPose-s	256 × 192	0.68	72.8	25.01	13.89
	RTMPose-m	256 × 192	1.93	77.3	49.46	26.44
	RTMPose-l	256 × 192	4.16	78.3	85.75	45.37

Table 11. Inference speed on CPU and GPU. RTMPose models are deployed and tested using ONNXRuntime and TensorRT respectively. Flip test is not used in this table.

Results		Input Size	GFLOPs	AP	CPU(ms)	GPU(ms)
COCO [37]	TinyPose	256 × 192	0.33	65.6	10.580	3.055
	LiteHRNet-30	256 × 192	0.42	66.3	22.750	6.561
	RTMPose-t	256 × 192	**0.36**	**67.1**	**3.204**	**1.064**
	RTMPose-s	256 × 192	**0.68**	**71.2**	**4.481**	**1.392**
	HRNet-w32+UDP	256 × 192	7.7	75.1	37.734	5.133
	RTMPose-m	256 × 192	**1.93**	**75.3**	**11.060**	**2.288**
	RTMPose-l	256 × 192	**4.16**	**76.3**	**18.847**	**3.459**
COCO-WholeBody [28]	HRNet-w32+DARK	256 × 192	7.72	57.8	39.051	5.154
	RTMPose-m	256 × 192	**2.22**	**59.1**	**13.496**	**4.000**
	RTMPose-l	256 × 192	**4.52**	**62.2**	**23.410**	**5.673**
	HRNet-w48+DARK	384 × 288	35.52	65.3	150.765	13.974
	RTMPose-l	384 × 288	**10.07**	**66.1**	**44.581**	**7.678**

Table 12. Pipeline Inference speed on CPU, GPU and Mobile device.

Model	Input Size	GFLOPs	Pipeline AP	CPU(ms)	GPU(ms)	Mobile(ms)
RTMDet-nano RTMPose-t	320 × 320 256 × 192	0.31 0.36	64.4	12.403	2.467	18.780
RTMDet-nano RTMPose-s	320 × 320 256 × 192	0.31 0.42	68.5	16.658	2.730	21.683
RTMDet-nano RTMPose-m	320 × 320 256 × 192	0.31 1.93	73.2	26.613	4.312	32.122
RTMDet-nano RTMPose-l	320 × 320 256 × 192	0.31 4.16	74.2	36.311	4.644	47.642

5. Conclusion

This paper empirically explores key factors in pose estimation such as the paradigm, model architecture, training strategy, and deployment. Based on the findings we present a high-performance real-time multi-person pose estimation framework, RTMPose, which achieves excellence in balancing model performance and complexity and can be deployed on various devices (CPU, GPU, and mobile devices) for real-time inference. We hope that the proposed algorithm alone with its open-sourced implementation can meet some of the demand for applicable pose estimation in industry, and benefit future explorations on the human pose estimation task.

References

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter V. Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. *Computer Vision and Pattern Recognition*, 2014. 7, 8

[2] PaddlePaddle Authors. Paddledetection, object detection and instance segmentation toolkit based on paddlepaddle. https://github.com/PaddlePaddle/PaddleDetection. 6, 7, 8

[3] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. Blazepose: On-device real-time body pose tracking, 2020. 1, 2, 4, 5, 6, 7

[4] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, volume 2, page 4, 2021. 2

[5] Yuanhao Cai, Zhicheng Wang, Zhengxiong Luo, Binyi Yin, Angang Du, Haoqian Wang, Xiangyu Zhang, Xinyu Zhou, Erjin Zhou, and Jian Sun. Learning delicate local representations for multi-person pose estimation. In *ECCV*, pages 455–472. Springer, 2020. 2

[6] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 7

[7] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017. 2, 7

[8] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 2, 3

[9] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 1€ filter: a simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2527–2530, 2012. 5

[10] Bowen Cheng, Bin Xiao, Jingdong Wang, Honghui Shi, Thomas S Huang, and Lei Zhang. Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5386–5395, 2020. 2

[11] MMPose Contributors. Openmmlab pose estimation toolbox and benchmark. https://github.com/open-mmlab/mmpose, 2020. 6, 7, 8

[12] MMDeploy Contributors. Openmmlab's model deployment toolbox. https://github.com/open-mmlab/mmdeploy, 2021. 2, 7

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *Computer Vision and Pattern Recognition*, 2009. 7

[14] Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv: Computer Vision and Pattern Recognition*, 2017. 4

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 2

[16] Raúl Díaz and Amit Marathe. Soft labels for ordinal regression. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3, 4

[17] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6824–6835, 2021. 2

[18] Hao-Shu Fang, Jiefeng Li, Hongyang Tang, Chao Xu, Haoyi Zhu, Yuliang Xiu, Yong-Lu Li, and Cewu Lu. Alphapose: Whole-body regional multi-person pose estimation and tracking in real-time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 6, 7

[19] Zigang Geng, Ke Sun, Bin Xiao, Zhaoxiang Zhang, and Jingdong Wang. Bottom-up human pose estimation via disentangled keypoint regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14676–14686, 2021. 1, 2

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 1

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Cornell University - arXiv*, 2015. 3

[22] Gines Hidalgo, Yaadhav Raaj, Haroon Idrees, Donglai Xiang, Hanbyul Joo, Tomas Simon, and Yaser Sheikh. Single-network whole-body pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6982–6991, 2019. 7

[23] Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc V. Le. Transformer quality in linear time. *ArXiv*, abs/2202.10447, 2022. 4

[24] Junjie Huang, Zheng Zhu, Feng Guo, and Guan Huang. The devil is in the details: Delving into unbiased data processing for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 4, 5

[25] Junjie Huang, Zheng Zhu, Feng Guo, and Guan Huang. The devil is in the details: Delving into unbiased data processing for human pose estimation. In *CVPR*, pages 5700–5709, 2020. 2

[26] Junjie Huang, Zheng Zhu, Guan Huang, and Dalong Du. Aid: Pushing the performance boundary of human pose estimation with information dropping augmentation, 2020. 4

[27] Sheng Jin, Wentao Liu, Enze Xie, Wenhai Wang, Chen Qian, Wanli Ouyang, and Ping Luo. Differentiable hierarchical graph grouping for multi-person pose estimation. In *European Conference on Computer Vision*, pages 718–734. Springer, 2020. 2

[28] Sheng Jin, Lumin Xu, Jin Xu, Can Wang, Wentao Liu, Chen Qian, Wanli Ouyang, and Ping Luo. Whole-body human pose estimation in the wild, 2020. 7, 9

[29] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. Pifpaf: Composite fields for human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11977–11986, 2019. 1, 2

[30] Jiefeng Li, Siyuan Bian, Ailing Zeng, Can Wang, Bo Pang, Wentao Liu, and Cewu Lu. Human pose regression with residual log-likelihood estimation. In *ICCV*, 2021. 2, 4

[31] Jiefeng Li, Can Wang, Hao Zhu, Yihuan Mao, Hao-Shu Fang, and Cewu Lu. Crowdpose: Efficient crowded scenes pose estimation and a new benchmark. *arXiv preprint arXiv:1812.00324*, 2018. 1

[32] Jiefeng Li, Can Wang, Hao Zhu, Yihuan Mao, Hao-Shu Fang, and Cewu Lu. Crowdpose: Efficient crowded scenes pose estimation and a new benchmark. *Cornell University - arXiv*, 2018. 7, 8

[33] Ke Li, Shijie Wang, Xiang Zhang, Yifan Xu, Weijian Xu, and Zhuowen Tu. Pose recognition with cascade transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1944–1953, 2021. 3

[34] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. *arXiv preprint arXiv:2203.16527*, 2022. 2

[35] Yanjie Li, Sen Yang, Peidong Liu, Shoukui Zhang, Yunxiao Wang, Zhicheng Wang, Wankou Yang, and Shu-Tao Xia. Simcc: a simple coordinate classification perspective for human pose estimation, 2021. 2, 3, 8

[36] Yanjie Li, Shoukui Zhang, Zhicheng Wang, Sen Yang, Wankou Yang, Shu-Tao Xia, and Erjin Zhou. Tokenpose: Learning keypoint tokens for human pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11313–11322, 2021. 2, 3, 4, 8

[37] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 1, 5, 6, 7, 9

[38] Huajun Liu, Fuqiang Liu, Xinyi Fan, and Dong Huang. Polarized self-attention: towards high-quality pixel-wise regression. *arXiv preprint arXiv:2107.00782*, 2021. 2

[39] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 2

[40] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3202–3211, 2022. 2

[41] Zhengxiong Luo, Zhicheng Wang, Yan Huang, Liang Wang, Tieniu Tan, and Erjin Zhou. Rethinking the heatmap regression for bottom-up human pose estimation. In *CVPR*, pages 13264–13273, 2021. 2

[42] Chengqi Lyu, Wenwei Zhang, Haian Huang, Yue Zhou, Yudong Wang, Yanyi Liu, Shilong Zhang, and Kai Chen. Rtmdet: An empirical study of designing real-time object detectors, 2022. 1, 2, 3, 4, 6

[43] Weian Mao, Yongtao Ge, Chunhua Shen, Zhi Tian, Xinlong Wang, Zhibin Wang, and Anton van den Hengel. Poseur: Direct human pose regression with transformers. In *European Conference on Computer Vision*, pages 72–88. Springer, 2022. 2, 3

[44] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. *NIPS*, 30, 2017. 2, 7

[45] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *CVPR*, pages 4929–4937, 2016. 2

[46] RangiLyu. Nanodet-plus: Super fast and high accuracy lightweight anchor-free object detection model. https://github.com/RangiLyu/nanodet, 2021. 1

[47] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv: Computer Vision and Pattern Recognition*, 2018. 6

[48] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Cornell University - arXiv*, 2015. 6

[49] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *Cornell University - arXiv*, 2018. 1

[50] Noam Shazeer. Glu variants improve transformer, 2020. 4

[51] Dahu Shi, Xing Wei, Liangqi Li, Ye Ren, and Wenming Tan. End-to-end multi-person pose estimation with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11069–11078, 2022. 1, 2, 3

[52] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019. 1, 2, 7, 8

[53] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 3

[54] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 2, 7

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 4

[56] Ronny Votel, Na Li, and Google Research. Next-generation pose detection with movenet and tensorflow.js. 2023. 1, 6, 7

[57] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv: Computer Vision and Pattern Recognition*, 2021. 7

[58] Jiahong Wu, He Zheng, Bo Zhao, Yixin Li, Baoming Yan, Rui Liang, Wenjia Wang, Shipei Zhou, Guosen Lin, Yanwei Fu, Yizhou Wang, and Yonggang Wang. Ai challenger : A large-scale dataset for going deeper in image understanding. *arXiv: Computer Vision and Pattern Recognition*, 2017. 5, 7

[59] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *European Conference on Computer Vision (ECCV)*, 2018. 2, 3, 7, 8

[60] Lumin Xu, Sheng Jin, Wentao Liu, Chen Qian, Wanli Ouyang, Ping Luo, and Xiaogang Wang. Zoomnas: Searching for whole-body human pose estimation in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 7

[61] Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. Vitpose: Simple vision transformer baselines for human pose estimation, 2022. 1, 2

[62] Sen Yang, Zhibin Quan, Mu Nie, and Wankou Yang. Transpose: Keypoint localization via transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11802–11812, 2021. 2, 4

[63] Changqian Yu, Bin Xiao, Changxin Gao, Lu Yuan, Lei Zhang, Nong Sang, and Jingdong Wang. Lite-hrnet: A lightweight high-resolution network. In *CVPR*, 2021. 1, 2, 7

[64] Hang Yu, Yufei Xu, Jing Zhang, Wei Zhao, Ziyu Guan, and Dacheng Tao. Ap-10k: A benchmark for animal pose estimation in the wild. *Cornell University - arXiv*, 2021. 7, 8

[65] Feng Zhang, Xiatian Zhu, Hanbin Dai, Mao Ye, and Ce Zhu. Distribution-aware coordinate representation for human pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[66] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6881–6890, 2021. 2

[67] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 2