RTMO: Towards High-Performance One-Stage Real-Time Multi-Person Pose Estimation

Peng Lu[1,2], Tao Jiang[2], Yining Li[2], Xiangtai Li[2,3], Kai Chen[2*], Wenming Yang[1*]

[1]Tsinghua Shenzhen International Graduate School
[2]Shanghai AI Laboratory [3]Nanyang Technological University
{lupeng, jiangtao, liyining, chenkai}@pjlab.org.cn, yang.wenming@sz.tsinghua.edu.cn

Abstract

Real-time multi-person pose estimation presents significant challenges in balancing speed and precision. While two-stage top-down methods slow down as the number of people in the image increases, existing one-stage methods often fail to simultaneously deliver high accuracy and real-time performance. This paper introduces RTMO, a one-stage pose estimation framework that seamlessly integrates coordinate classification by representing keypoints using dual 1-D heatmaps within the YOLO architecture, achieving accuracy comparable to top-down methods while maintaining high speed. We propose a dynamic coordinate classifier and a tailored loss function for heatmap learning, specifically designed to address the incompatibilities between coordinate classification and dense prediction models. RTMO outperforms state-of-the-art one-stage pose estimators, achieving 1.1% higher AP on COCO while operating about 9 times faster with the same backbone. Our largest model, RTMO-l, attains 74.8% AP on COCO `val2017` *and 141 FPS on a single V100 GPU, demonstrating its efficiency and accuracy. The code and models are available at https://github.com/open-mmlab/mmpose/tree/main/projects/rtmo.*

1. Introduction

Multi-person pose estimation (MPPE) is essential in the field of computer vision, with applications ranging from augmented reality to sports analytic. Real-time processing is particularly crucial for applications requiring instant feedback, such as coaching for athlete positioning. Although numerous real-time pose estimation techniques have emerged [3, 16, 17, 31], achieving a balance between speed and accuracy remains challenging.

Current real-time pose estimation methods fall into two categories: top-down [3, 16] and one-stage [17, 31]. Top-down methods employ pre-trained detectors to create bounding boxes around subjects, followed by pose estimation for each individual. A key limitation is that their inference time scales with the number of people in the image (see Figure 1). On the other hand, one-stage methods di-

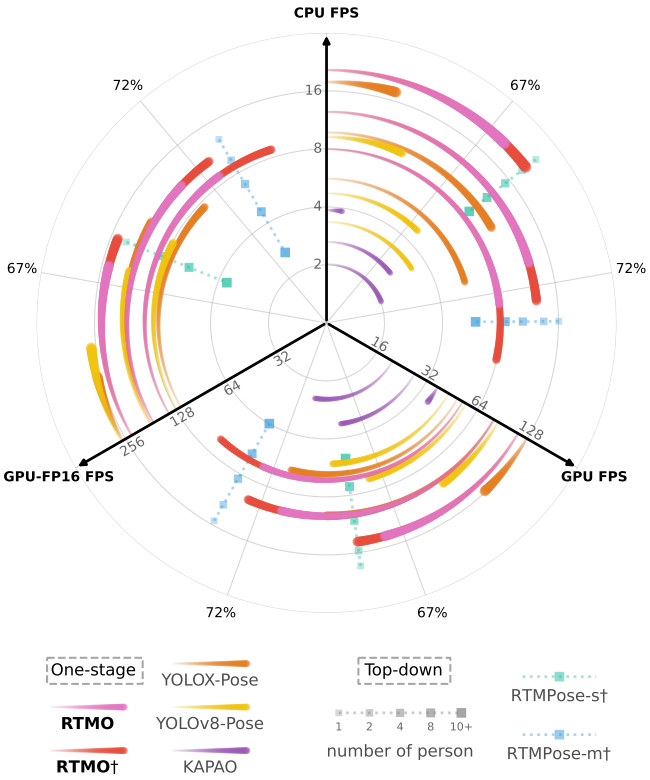

Figure 1. Efficiency and efficacy comparison among real-time pose estimation methods across different inference backends and devices. The radial axes indicate inference speed in Frames Per Second (FPS). The outer circular axis shows Average Precision (AP) on the COCO `val2017` dataset. Models marked with † were trained with additional data beyond the COCO `train2017`.

*Corresponding authors.

rectly predict the locations of keypoints for all individuals in the image. However, current real-time one-stage methods [17, 31, 34] lag in accuracy compared to top-down approaches (see Figure 1). These methods, relying on the YOLO architecture, directly regress the keypoint coordinates, which hinders performance since this technique resembles using a Dirac delta distribution for each keypoint, neglecting the inherent ambiguity and uncertainty [21].

Alternatively, the coordinate classification methods employ dual 1-D heatmaps that increase spatial resolution by spreading the probability of keypoint locations over two sets of bins spanning the entire image. This offers more accurate predictions with minimal extra computational cost [16, 23]. However, directly applying coordinate classification to dense prediction scenarios like one-stage pose estimation leads to inefficient bin utilization due to global bin distribution across the image and each person occupying only a minor region. Additionally, conventional Kullback–Leibler divergence (KLD) losses treat all samples equally, which is suboptimal for one-stage pose estimation where instance difficulty varies significantly across grids.

In this work, we overcome the above challenges and incorporate the coordinate classification approach within the YOLO-based framework, leading to the development of ***R**eal-**T**ime **M**ulti-person **O**ne-stage* (RTMO) pose estimation models. RTMO introduces a Dynamic Coordinate Classifier (DCC) that includes dynamic bin allocation localized to bounding boxes and learnable bin representations. Furthermore, we propose a novel loss function based on Maximum Likelihood Estimation (MLE) to effectively train the coordinate heatmaps. This new loss allows learning of per-sample uncertainty, automatically adjusting task difficulty and balancing optimization between hard and easy samples for more effective and harmonized training.

Consequently, RTMO achieves accuracy comparable to real-time top-down methods and exceeds other lightweight one-stage methods, as shown in Figure 1. Moreover, RTMO demonstrates superior speed when processing multiple instances in an image, outpacing top-down methods with similar accuracy. Notably, RTMO-l attains a 74.8% Average Precision (AP) on the COCO `val2017` dataset [24] and exhibited 141 frames per second (FPS) on the NVIDIA V100 GPU. On the CrowdPose benchmark [19], RTMO-l achieves 73.2% AP, a new state of the art for one-stage methods. The key contributions of this work include:

- An innovative coordinate classification technique tailored for dense prediction, utilizing coordinate bins for precise keypoint localization while addressing varying instance sizes and complexities.
- A new real-time one-stage MPPE approach that seamlessly integrates coordinate classification with the YOLO architecture, achieving an optimal balance of performance and speed among existing MPPE methods.

2. Related Works

2.1. One-Stage Pose Estimator

Inspired by advancements in one-stage object detection algorithms [8, 10, 25, 41, 52], a series of one-stage pose estimation methods have emerged [11, 31, 35, 40, 52]. These methods perform MPPE in a single forward pass and directly regress instance-specific keypoints from predetermined root locations. Alternative approaches such as PETR [38] and ED-Pose [47] treat pose estimation as a set prediction problem for end-to-end keypoint regression. Beyond regression-based solutions, techniques like FCPose [32], InsPose [36], and CID [43] utilize dynamic convolution or attention mechanisms to generate instance-specific heatmaps for keypoint localization.

Compared to two-stage pose estimation methods, one-stage approaches eliminate the need for pre-processing (e.g., human detection for top-down methods) and post-processing (e.g., keypoint grouping for bottom-up methods). This results in two benefits: 1) consistent inference time, irrespective of the number of instances in the image; and 2) a simplified pipeline that facilitates deployment and practical use. Despite these advantages, the existing one-stage methods struggle to balance high accuracy with real-time inference. High-accuracy models [43, 47] often depend on resource-intensive backbones like HRNet [39] or Swin [26], making real-time estimation challenging. Conversely, real-time models [31, 34] compromise on performance. Our model addresses this trade-off, delivering both high accuracy and fast real-time inference.

2.2. Coordinate Classification

SimCC [23] and RTMPose [16] have adopted coordinate classification for pose estimation, classifying keypoints into sub-pixel bins along horizontal and vertical axes to achieve spatial discrimination without high-resolution features, balancing accuracy and speed. However, spanning bins across the entire image for dense prediction methods is impractical due to the vast number of bins needed to diminish quantization error, which leads to inefficiency from many bins being superfluous for individual instances. DFL [21] sets bins around a predefined range near each anchor, which may not cover the keypoints of large instances and could induce significant quantization errors for small instances. Our approach assigns bins within localized regions scaled to each instance's size, which optimizes bin utilization, ensures keypoint coverage, and minimizes quantization error.

2.3. Transformer-Enhanced Pose Estimation

Transformer-based architectures have become ubiquitous in pose estimation, leveraging state-of-the-art transformer backbones to improve accuracy as in ViTPose [46], or combining transformer encoders with CNNs to capture spatial

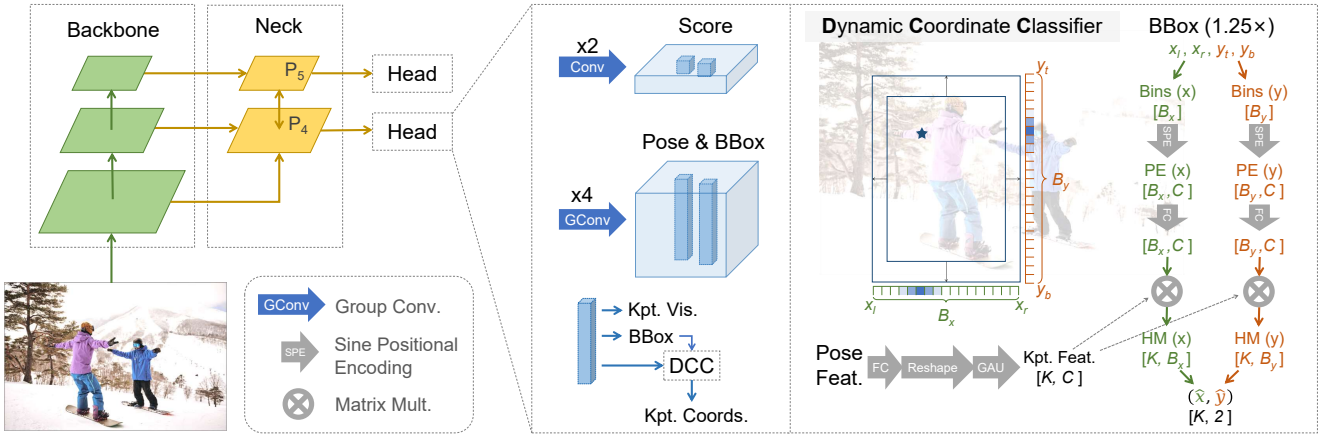

Figure 2. Overview of the RTMO Network Architecture. Its head outputs predictions for the score, bounding box, keypoint coordinates and visibility for each grid cell. The Dynamic Coordinate Classifier translates pose features into K pairs of 1-D heatmaps for both the horizontal and vertical axes, encompassing an expanded region 1.25 times the size of the predicted bounding boxes. From these heatmaps, keypoint coordinates are precisely extracted. K denotes the total number of keypoints.

relationships [48]. TokenPose [22] and Poseur [33] demonstrate the efficacy of token-based keypoint embedding in both heatmap and regression-based methods, leveraging visual cues and anatomical constraints. Frameworks like PETR [38] and ED-Pose [47] introduce transformers in end-to-end multi-person pose estimation, and RTMPose [16] integrates self-attention with a SimCC-based [23] framework for keypoint dependency analysis, an approach also adopted by RTMO. While positional encoding is standard in attention to inform query and key positions, we innovatively employ it to form representation vectors for each spatial bin, enabling computation of bin-keypoint similarity that facilitates accurate localization predictions.

3. Methodology

In our model, we adopt a YOLO-like architecture, illustrated in Figure 2. The backbone is CSPDarknet [10], and we process the last three feature maps using a Hybrid Encoder [29], yielding spatial features P_4 and P_5 with respective downsampling rates of 16 and 32. Each pixel in these features maps to a grid cell uniformly distributed on the original image plane. The network head, utilizing dual convolution blocks at each spatial level, generates a score and corresponding pose features for every grid cell. These pose features are utilized to predict bounding boxes, keypoint coordinates, and visibility. The generation of 1-D heatmap predictions through the Dynamic Coordinate Classifier is detailed in Sec. 3.1 while the proposed heatmap loss based on MLE is presented in Sec. 3.2. The complete training and inference procedures are outlined in Sec. 3.3.

3.1. Dynamic Coordinate Classifier

The pose features associated with each grid cell encapsulate the keypoint displacement from the grid. Previous works [11, 31, 35] directly regress these displacement, and thus fall short in performance. Our study explores the integration of coordinate classification with a one-stage pose estimation framework to improve keypoint localization accuracy. A notable limitation of existing coordinate classification methods is their static strategy for bin assignment. To address this problem, we introduce the Dynamic Coordinate Classifier (DCC), which *dynamically assigns ranges and forms representations for bins* in two 1-D heatmaps, effectively resolving the incompatibilities of coordinate classification in dense prediction contexts.

Dynamic Bin Allocation. Coordinate classification technique employed in top-down pose estimators allocates bins across the entire input image [16, 23], leading to bin wastage in one-stage methods since each subject occupies only a small portion. DFL [21] sets bins within a predefined range near each anchor, which may miss keypoints in larger instances and cause significant quantization errors in smaller ones. DCC addresses these limitations by dynamically assigning bins to align with each instance's bounding box, ensuring localized coverage. The bounding boxes are initially regressed using a pointwise convolution layer and then expanded by 1.25x to cover all keypoints, even in cases of inaccurate predictions. These expanded bounding boxes are uniformly divided into B_x, B_y bins along horizontal and vertical axes. The x-coordinate for each horizontal bin is calculated using:

$$x_i = x_l + (x_r - x_l)\,\frac{i-1}{B_x - 1},$$

where x_r, x_l are the left and right sides of the bounding box, and index i varies from 1 to B_x. The y-axis bins are computed similarly. We empirically use $B_x = 192$ and $B_y = 256$ for all models.

Dynamic Bin Encoding. In the context of DCC, the position of each bin varies across grids since their predicted bounding boxes differ. This differs from previous methods [16, 23] where bin coordinates are fixed. Rather than a shared representation for bins across grids used in these methods, DCC generates tailored representations on-the-fly. Specifically, we encode each bin's coordinates into positional encodings to create bin-specific representations. We utilize sine positional encoding [42] defined as:

$$[\boldsymbol{PE}(x_i)]_c = \begin{cases} \sin\left(\frac{x_i}{t^{c/C}}\right), & \text{for even } c \\ \cos\left(\frac{x_i}{t^{(c-1)/C}}\right), & \text{for odd } c \end{cases},$$

where t denotes the temperature, c is the index, and C represents the total number of dimension. We refine the positional encoding's adaptability for our task using a fully connected layer, which applies a learnable linear transformation $\boldsymbol{\phi}$, thereby optimizing its effectiveness in DCC.

The primary objective of DCC is to accurately predict keypoint occurrence probabilities at each bin, informed by bin coordinates and keypoint features. Keypoint features are extracted from the pose feature and refined via a Gated Attention Unit (GAU) module [14] following RTMPose [16], to enhance inter-keypoint consistency. The probability heatmap is generated by multiplying the keypoint features $\boldsymbol{f}_k$ with the positional encodings of each bin $\boldsymbol{PE}(x_i)$, followed by a softmax:

$$\hat{p}_k(x_i) = \frac{e^{\boldsymbol{f}_k \cdot \boldsymbol{\phi}(\boldsymbol{PE}(x_i))}}{\sum_{j=1}^{B_x} e^{\boldsymbol{f}_k \cdot \boldsymbol{\phi}(\boldsymbol{PE}(x_j))}},$$

where $\boldsymbol{f}_k$ is the k-th keypoint's feature vector.

3.2. MLE for Coordinate Classification

In classification tasks, one-hot targets and cross-entropy loss are commonly utilized. Label smoothing, like Gaussian label smoothing used in SimCC [23] and RTMPose [16], along with KLD, can improve performance. The Gaussian mean μ_x, μ_y and variance σ^2 are set to the annotated coordinates and a predefined parameter. The target distribution is defined as:

$$p_k(x_i \mid \mu_x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu_x)^2}{2\sigma^2}} \sim \mathcal{N}(x_i; \mu_x, \sigma^2).$$

Importantly, we note that $p_k(x_i \mid \mu_x)$ is mathematically identical to the likelihood $p_k(\mu_x \mid x_i)$ of the annotation μ_x under a Gaussian error model with true value x_i. This symmetrical property arises because the Gaussian distribution is symmetric with respect to its mean. Treating the predicted $\hat{p}_k(x_i)$ as the prior of x_i, the annotation likelihood for the k-th keypoint is:

$$\begin{aligned} P(\mu_x) &= \sum_{i=1}^{B_x} P(\mu_x \mid x_i) P(x_i) \\ &= \sum_{i=1}^{B_x} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu_x)^2}{2\sigma^2}} \hat{p}_k(x_i). \end{aligned}$$

Maximizing this likelihood models the true distribution of the annotations.

In practice, we employ a Laplace distribution for $P(\mu_x \mid x_i)$ and a negative log-likelihood loss:

$$\mathcal{L}_{mle}^{(x)} = -\log\left[\sum_{i=1}^{B_x} \frac{1}{\hat{\sigma}} e^{-\frac{|x_i-\mu_x|}{2\hat{\sigma}s}} \hat{p}_k(x_i)\right],$$

where the instance size s normalizes the error and $\hat{\sigma}$ is the predicted variance. The constant factor is omitted as it does not affect the gradient. The total Maximum Likelihood Estimation (MLE) loss is $\mathcal{L}_{mle} = \mathcal{L}_{mle}^{(x)} + \mathcal{L}_{mle}^{(y)}$.

Unlike KLD, our MLE loss allows for learnable variance, representing uncertainty. This uncertainty learning framework automatically adjusts the difficulty of various samples [5, 13]. For hard samples, the model predicts a large variance to ease optimization. For simple samples, it predicts a small variance, aiding in accuracy improvement. With KLD, adopting a learnable variance is problematic - the model leans to predict a large variance to flatten the target distribution as this simplifies learning. More discussion can be found in Sec. 4.4.

3.3. Training and Inference

Training. Our model, adhering to a YOLO-like structure, employs dense grid prediction for human detection and pose estimation. It is crucial for the model to differentiate between positive and negative grids. We extend SimOTA [10] for training, assigning positive grids based on grid scores, bounding box regression, and pose estimation accuracy. The head's score branch classifies these grids, supervised by varifocal loss [51] $\mathcal{L}_{cls}$, with target scores being the Object Keypoint Similarity (OKS) between the predicted pose and the assigned ground truth for each grid.

Positive grid tokens yield bounding box, keypoint coordinates, and visibility predictions. Keypoint coordinates are derived via the DCC, while other predictions come from pointwise convolution layers. The losses applied are IoU loss for bounding boxes $\mathcal{L}_{bbox}$, MLE loss for keypoints $\mathcal{L}_{mle}$, and BCE loss for visibility $\mathcal{L}_{vis}$.

Given the DCC's computational demands, we implement a pointwise convolution layer for preliminary coordinate regression, similar to YOLO-Pose [31], to mitigate out-of-memory issues. This regressed keypoints kpt_{reg} serves as

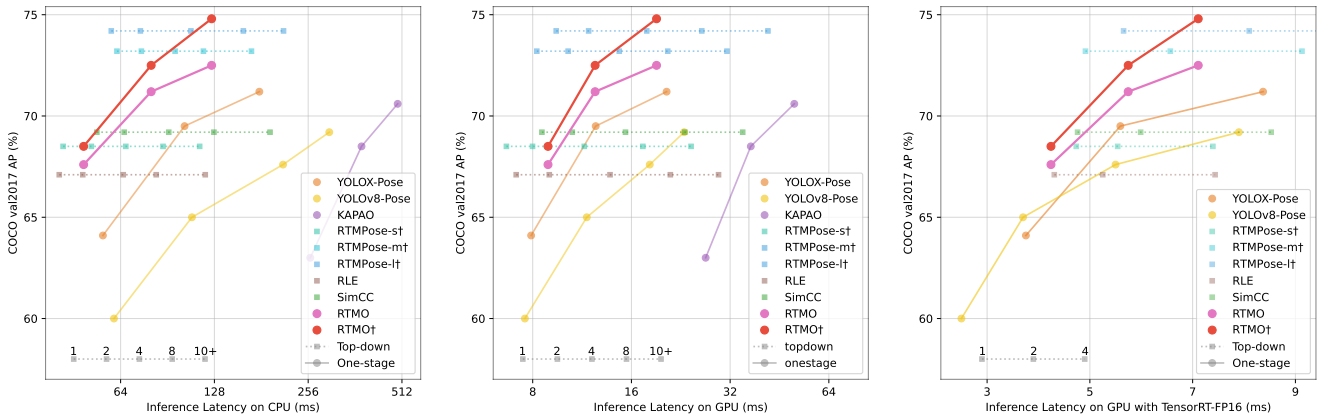

Figure 3. Comparison of RTMO with other real-time multi-person pose estimators. The latency for top-down methods varies depending on the number of instances in the image, as indicated by numerical values in the figures. All models are evaluated without test-time augmentation. † indicates that the model was trained using additional data beyond the COCO `train2017` dataset.

a proxy in SimOTA for positive grid selection, with the decoded keypoints kpt_{dec} later used to calculate OKS. The regression branch's loss is OKS loss [31]:

$$\mathcal{L}_{proxy} = 1 - \text{OKS}\left(\text{kpt}_{reg}, \text{kpt}_{dec}\right).$$

The total loss for the proposed model is

$$\mathcal{L} = \lambda_1 \mathcal{L}_{bbox} + \lambda_2 \mathcal{L}_{mle} + \lambda_3 \mathcal{L}_{proxy} + \lambda_4 \mathcal{L}_{cls} + \mathcal{L}_{vis},$$

with hyperparameters $\lambda_1, \lambda_2, \lambda_3$, and λ_4 set at $\lambda_1 = \lambda_2 = 5, \lambda_3 = 10$, and $\lambda_4 = 2$.

Inference. During inference, our model employs a score threshold of 0.1 and non-maximum suppression for grid filtering. It then decodes pose features from selected grids into heatmaps, using integral over the heatmaps to derive keypoint coordinates. This selective decoding approach minimizes the number of features for processing, reducing computational cost.

4. Experiments

4.1. Settings

Datasets. Experiments were primarily conducted on the COCO2017 Keypoint Detection benchmark [24], comprising approximately 250K person instances with 17 keypoints. Performance comparisons were made with state-of-the-art methods on both the `val2017` and `test-dev` sets. To explore our model's performance ceiling, training was also extended to include additional datasets: CrowdPose [19], AIC [44], MPII [1], JHMDB [15], Halpe [9], and PoseTrack18 [2]. These annotations were converted to COCO format. RTMO was further evaluated on the CrowdPose benchmark [19], which is known for its high complexity due to crowded and occluded scenes, comprising 20K images and approximately 80K persons with 14 key points. OKS-based Average Precision (AP) served as the evaluation metric for both datasets.

Implementation Details. During training, we adopt the image augmentation pipeline from YOLOX [10], incorporating mosaic augmentation, random color adjustments, geometric transformations, and MixUp [50]. Training images are resized to dimensions [480, 800]. Epoch counts are set to 600 and 700 for the COCO and CrowdPose datasets, respectively. The training process is divided into two stages: the first involves training both the proxy branch and DCC using pose annotations, and the second shifts the target of the proxy branch to the decoded pose from DCC. The AdamW optimizer [27] is used with a weight decay of 0.05, and training is performed on Nvidia GeForce RTX 3090 GPUs with batch size 256. Initial learning rates are set to 4×10^{-3} and 5×10^{-4} for the two training phases, decaying to 2×10^{-4} via cosine annealing. For inference, images are resized to 640. CPU latency is measured on an Intel Xeon Gold CPU using ONNXRuntime. GPU latency is tested on an NVIDIA V100 GPU with ONNXRuntime and with TensorRT using half-precision floating-point (FP16) format. MMPose [7] toolbox is used to implement RTMO models.

4.2. Benchmark Results

COCO To assess RTMO against other real-time pose estimators, we measured AP and inference latency on the COCO `val2017` dataset. For one-stage methods, we considered KAPAO [34], YOLOv8-Pose [17], and YOLOX-Pose—an adaptation of YOLO-Pose [31] on YOLOX [10]. For top-down approaches, RLE [20], SimCC [23] and RTMPose [16] were selected for comparison. RTMDet-nano [30], a highly efficient object detection model, served as the human detector for top-down models. Since top-

Method	Backbone	#Params	Time (ms)	AP	AP_{50}	AP_{75}	AP_M	AP_L	AR
DirectPose [40]	ResNet-50	-	74	62.2	86.4	68.2	56.7	69.8	-
DirectPose [40]	ResNet-101	-	-	63.3	86.7	69.4	57.8	71.2	-
FCPose [32]	ResNet-50	41.7M	68	64.3	87.3	71.0	61.6	70.5	-
FCPose [32]	ResNet-101	60.5M	93	65.6	87.9	72.6	62.1	72.3	-
InsPose [36]	ResNet-50	50.2M	80	65.4	88.9	71.7	60.2	72.7	-
InsPose [36]	ResNet-101	-	100	66.3	89.2	73.0	61.2	73.9	-
CenterNet [8]	Hourglass	194.9M	160	63.0	86.8	69.6	58.9	70.4	-
PETR [37]	ResNet-50	43.7M	89	67.6	89.8	75.3	61.6	76.0	-
PETR [37]	Swin-L	213.8M	133	70.5	91.5	78.7	65.2	78.0	-
CID [43]	HRNet-w32	29.4M	*84.0*	68.9	89.9	76.9	63.2	77.7	74.6
CID [43]	HRNet-w48	65.4M	*94.8*	70.7	90.4	77.9	66.3	77.8	76.4
ED-Pose [47]	ResNet-50	50.6M	*135.2*	69.8	90.2	77.2	64.3	77.4	-
ED-Pose [47]	Swin-L	218.0M	*265.6*	72.7	92.3	80.9	67.6	80.0	-
KAPAO-s [34]	CSPNet	12.6M	*26.9*	63.8	88.4	70.4	58.6	71.7	71.2
KAPAO-m [34]	CSPNet	35.8M	*37.0*	68.8	90.5	76.5	64.3	76.0	76.3
KAPAO-l [34]	CSPNet	77.0M	*50.2*	70.3	91.2	77.8	66.3	76.8	77.7
YOLO-Pose-s [31]	CSPDarknet	10.8M	*7.9*	63.2	87.8	69.5	57.6	72.6	67.6
YOLO-Pose-m [31]	CSPDarknet	29.3M	*12.5*	68.6	90.7	75.8	63.4	77.1	72.8
YOLO-Pose-l [31]	CSPDarknet	61.3M	*20.5*	70.2	91.1	77.8	65.3	78.2	74.3
RTMO-r50	ResNet-50	41.7M	*15.5*	70.9	91.0	78.2	65.8	79.1	75.0
RTMO-s	CSPDarknet	9.9M	*8.9*	66.9	88.8	73.6	61.1	75.7	70.9
RTMO-s†	CSPDarknet	9.9M	*8.9*	67.7	89.4	74.5	61.5	77.2	71.9
RTMO-m	CSPDarknet	22.6M	*12.4*	70.1	90.6	77.1	65.1	78.1	74.2
RTMO-m†	CSPDarknet	22.6M	*12.4*	71.5	91.0	78.6	66.1	79.9	75.6
RTMO-l	CSPDarknet	44.8M	*19.1*	71.6	91.1	79.0	66.8	79.1	75.6
RTMO-l†	CSPDarknet	44.8M	*19.1*	73.3	91.9	80.8	68.3	81.1	77.4

Table 1. Performance comparison of state-of-the-art one-stage methods on the COCO `test-dev` dataset. The symbol † denotes models trained with additional data beyond the COCO `train2017` dataset. Inference time in *italic* are obtained using a single NVIDIA Tesla V100 GPU, while times without this emphasis are from PETR's paper [37] and evaluated using the same device. Times underlined were measured using PyTorch due to ONNX exportation incompatibilities with those models.

down models slow down as more people appear in the image, we partitioned the COCO `val2017` set based on person counts and assessed top-down model speeds accordingly. As shown in Fig. 3, the RTMO series outperform comparable lightweight one-stage methods in both performance and speed. Against top-down models, RTMO-m and RTMO-l are as accurate as RTMPose-m and RTMPose-l, and faster when more people are in the image. With ONNXRuntime, RTMO matches RTMPose in speed with around four people, and with TensorRT FP16, RTMO is quicker with two or more people. This demonstrates RTMO's advantage in multi-person scenarios. Importantly, although the number of tokens processed varies with the number of people in the image, the difference in inference latency is marginal. For example, the latency of RTMO-l on a GPU in a subset with more than 10 persons is only about 0.1 ms higher than in a subset with a single person, accounting for roughly 0.5% of the total latency.

Our evaluation of RTMO against leading one-stage pose estimators on the COCO `test-dev` is presented in Table 1. RTMO showcases significant advancements in speed and precision. Specifically, RTMO-s outperforms PETR [37] using a ResNet-50 [12] backbone, being 10x faster while maintaining similar accuracy. Compared to lightweight models like KAPAO and YOLO-Pose, RTMO consistently outperforms in accuracy across different model sizes. When trained on COCO `train2017`, RTMO-l has the second-best performance among all tested models. The highest-performing model, ED-Pose [47] with a Swin-L [26] backbone, is quite heavy and not deployment-friendly. RTMO, using the same ResNet-50 backbone, surpassed ED-Pose by 1.1% in AP and was faster. Additionally, transferring ED-Pose to the ONNX format resulted in a higher latency than its PyTorch model, about 1.5 seconds per frame. By contrast, the ONNX model of RTMO-l processes an image in just 19.1ms. With further training on additional human pose datasets, RTMO-l performs best among one-stage pose estimators in terms of accuracy.

CrowdPose To evaluate RTMO under challenging scenarios, we test it on the CrowdPose [19] benchmark, characterized by images with dense crowds, significant person

Figure 4. Visualization of estimated human pose (top) and corresponding heatmaps (bottom).

Method	#Params	AP	AP_E	AP_M	AP_H
Top-down methods					
Sim.Base. [45]	34.0M	60.8	71.4	61.2	51.2
HRNet [39]	28.5M	71.3	**80.5**	71.4	62.5
TransPose-H [48]	-	71.8	79.5	72.9	62.2
HRFormer-B [49]	43.2M	72.4	80.0	73.5	62.4
RTMPose-m [16]	13.5M	70.6	79.9	71.9	58.2
Bottom-up methods					
OpenPose [4]	-	-	62.7	48.7	32.3
HrHRNet [6]	63.8M	65.9	73.3	66.5	57.9
DEKR [11]	65.7M	67.3	74.6	68.1	58.7
SWAHR [28]	63.8M	71.6	78.9	72.4	63.0
One-stage methods					
PETR [37]	220.5M	71.6	77.3	72.0	**65.8**
CID [43]	65.4M	72.3	78.7	72.1	64.8
KAPAO-l [34]	77.0M	68.9	76.6	69.9	59.5
ED-Pose [47]	218.0M	73.1	**80.5**	73.8	63.8
ED-Pose† [47]	218.0M	76.6	83.3	77.3	68.3
RTMO-s	9.9M	67.3	73.7	68.2	59.1
RTMO-m	22.6M	71.1	77.4	71.9	63.4
RTMO-l	44.8M	**73.2**	79.2	**74.1**	65.3
RTMO-l†	44.8M	83.8	88.8	84.7	77.2

Table 2. Performance comparisons with state-of-the-art methods on CrowdPose. The highest and second-highest performances are highlighted in bold and underlined, respectively. † indicates that the model was trained using additional data beyond CrowdPose.

overlap, and occlusion. The results are summarized in Table 2. Among bottom-up and single-stage approaches, RTMO-s has accuracy comparable to DEKR [11], yet it uses only 15% of the parameters. When trained on the CrowdPose dataset, RTMO-l surpasses ED-Pose [47] which uses a Swin-L [26] backbone, despite having a smaller model size. Notably, RTMO-l exceeds ED-Pose primarily on medium and hard samples, demonstrating its effectiveness in challenging situations. Moreover, with additional training data, RTMO-l achieves a state-of-the-art 81.7% AP, highlighting the model's capacity.

Decoding	loss	COCO		CrowdPose	
		AP	AR	AP	AR
Regression	OKS	65.6	69.9	66.1	70.9
CC+DBA+DBE	KLD	64.4	68.2	62.5	67.5
CC	MLE	66.7	70.6	65.8	70.7
CC+DBA	MLE	65.6	70.0	65.2	70.2
CC+DBA+DBE	MLE	**67.6**	**71.4**	**67.2**	**72.3**

Table 3. Comparison of decoding and supervision methods on COCO `val2017` and CrowdPose. The base model is RTMO-s. The term CC denotes Coordinate Classification; DBA and DBE denote Dynamic Bin Allocation and Dynamic Bin Encoding.

4.3. Qualitative Results

RTMO utilizes coordinate classification and demonstrates strong performance in challenging multi-person scenarios with small individuals and frequent occlusions. Figure 4 reveals that RTMO generates spatially accurate heatmaps even under these difficult conditions, facilitating robust and context-aware predictions for each keypoint.

4.4. Ablation Study

Classification v.s. Regression. To assess the effectiveness of coordinate classification against regression, we replaced the model's 1-D heatmap generation with a fully connected layer for regression, supervised by the OKS loss [31]. Table 3 compares the performances. Using the DCC module and MLE loss, coordinate classification outperforms regression with 2.0% AP on the COCO.

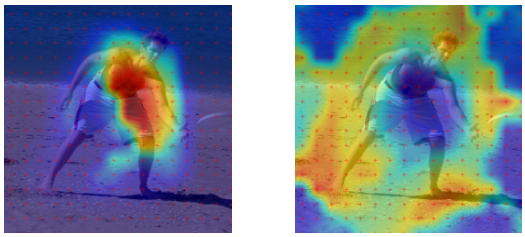

Figure 5. Visualization of (left) OKS showing sample difficulty and (right) learned variance in MLE loss. Red crosses mark the position of grids.

Figure 6. Heatmaps decoded without (left) and with (right) DBE.

Losses for Coordinate Classification. Compare to other pose estimation methods with coordinate classification that use KLD loss, our research indicates its inadequacy for RTMO. Table 3 demonstrates our MLE loss achieves higher accuracy than KLD. This improvement stems from the learnable variance in the MLE loss, which helps to balance the learning between hard and easy samples. In a one-stage pose estimator, the difficulty varies per grid due to factors like instance pose, size, and relative grid position, as visualized in Figure 5. Grids with higher OKS (easier) have lower variance in MLE loss, and vice versa. KLD fails to account for this variability, making it less effective in this context.

Dynamic Strategy in Coordinate Classification. We firstly adopt a static coordinate classification strategy similar to DFL [21] where bins are distributed in a fixed range around each grid. This approach outperformed the regression method on the COCO dataset but underperformed on CrowdPose. Introducing the Dynamic Bin Allocation (DBA) strategy to this baseline resulted in decreased performance on both datasets. This is reasonable, as the semantics of each bin vary across different samples without corresponding representation adjustments. This issue was rectified by incorporating Dynamic Bin Encoding (DBE). With DBE, our DCC methods exceeded the efficacy of the static strategy on both datasets. Furthermore, Without dynamic bin encoding (DBE), the probabilities of nearby bins can vary significantly, as shown in Figure 6, contradicting the expectation that adjacent spatial locations should have similar probabilities. In contrast, incorporating DBE leads to smoother output heatmaps, indicating improved decoder training by enabling representation vectors that better capture similarities between nearby locations.

Model	features	Latency (ms)		Accuracy	
		CPU	GPU	AP	AR
RTMO-s	{P3, P4, P5}	65.3	8.96	67.6	71.8
	{P4, P5}	48.7	8.91	67.6	71.4
RTMO-m	{P3, P4, P5}	108.6	16.87	71.4	75.1
	{P4, P5}	80.2	12.40	71.2	75.2
RTMO-l	{P3, P4, P5}	186.3	22.01	72.7	76.9
	{P4, P5}	125.4	19.09	72.5	76.6

Table 4. Comparison of performance and latency for models using 2 or 3 features. Accuracy metrics are based on the COCO `val2017` dataset. Latency measurements for both CPU and GPU are taken using ONNXRuntime.

Feature Maps Selection. Feature pyramids [18] leverage multi-scale features for detecting instances of varying sizes; deeper features typically detect larger objects. Our initial model used P3, P4, P5 features with strides of 8, 16, and 32 pixels. However, P3 contributed 78.5% of the FLOPs in the model head while accounting for 10.7% of correct detections. To improve efficiency, we focused on P4, P5. As shown in Table 4, omitting P3 led to significant speed gains with minimal accuracy loss, indicating that P4 and P5 alone are effective for multi-person pose estimation. This suggests that the role of P3 in detecting smaller instances can be compensated by the remaining features.

5. Conclusion

In conclusion, our RTMO model significantly improves the speed-accuracy tradeoff in one-stage multi-person pose estimation. By integrating coordinate classification within a YOLO-based framework, we achieve both real-time processing and high precision. Our approach, featuring a dynamic coordinate classifier and a loss function based on maximum likelihood estimation, effectively improves the location precision in dense prediction models. This breakthrough not only enhances pose estimation, but also establishes a robust foundation for future advancements in the scope of dense prediction for visual detection tasks.

Acknowledgments

We thank the reviewers for their helpful comments. This work was supported by the National Key R&D Program of China (No. 2022ZD0161600) and the Special Foundations for the Development of Strategic Emerging Industries of Shenzhen (Nos. JCYJ20200109143035495 & CJGJZD20210408092804011)

References

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter V. Gehler, and Bernt Schiele. 2D human pose estimation: New benchmark and state of the art analysis. In *CVPR*, 2014. 5

[2] Mykhaylo Andriluka, Umar Iqbal, Eldar Insafutdinov, Leonid Pishchulin, Anton Milan, Juergen Gall, and Bernt Schiele. Posetrack: A benchmark for human pose estimation and tracking. In *CVPR*, 2018. 5

[3] PaddlePaddle Authors. Paddledetection, object detection and instance segmentation toolkit based on paddlepaddle. https://github.com/PaddlePaddle/PaddleDetection. 1

[4] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *TPAMI*, 2019. 7

[5] Jie Chang, Zhonghao Lan, Changmao Cheng, and Yichen Wei. Data uncertainty learning in face recognition. In *CVPR*, 2020. 4

[6] 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 *CVPR*, 2020. 7

[7] MMPose Contributors. Openmmlab pose estimation toolbox and benchmark. https://github.com/open-mmlab/mmpose, 2020. 5

[8] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *ICCV*, 2019. 2, 6

[9] 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. *TPAMI*, 2022. 5

[10] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021. 2, 3, 4, 5

[11] Zigang Geng, Ke Sun, Bin Xiao, Zhaoxiang Zhang, and Jingdong Wang. Bottom-up human pose estimation via disentangled keypoint regression. In *CVPR*, 2021. 2, 3, 7

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

[13] Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang. Bounding box regression with uncertainty for accurate object detection. In *CVPR*, 2019. 4

[14] Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. Transformer quality in linear time. In *ICML*, 2022. 4

[15] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J Black. Towards understanding action recognition. In *ICCV*, 2013. 5

[16] Tao Jiang, Peng Lu, Li Zhang, Ningsheng Ma, Rui Han, Chengqi Lyu, Yining Li, and Kai Chen. RTMPose: Real-time multi-person pose estimation based on mmpose. *arXiv preprint arXiv:2303.07399*, 2023. 1, 2, 3, 4, 5, 7

[17] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. YOLO by Ultralytics. https://github.com/ultralytics/ultralytics, 2023. Accessed: February 30, 2023. 1, 2, 5

[18] Seung-Wook Kim, Hyong-Keun Kook, Jee-Young Sun, Mun-Cheon Kang, and Sung-Jea Ko. Parallel feature pyramid network for object detection. In *ECCV*, 2018. 8

[19] Jiefeng Li, Can Wang, Hao Zhu, Yihuan Mao, Hao-Shu Fang, and Cewu Lu. Crowdpose: Efficient crowded scenes pose estimation and a new benchmark. In *CVPR*, 2019. 2, 5, 6

[20] 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. 5

[21] Xiang Li, Chengqi Lv, Wenhai Wang, Gang Li, Lingfeng Yang, and Jian Yang. Generalized focal loss: Towards efficient representation learning for dense object detection. *TPAMI*, 2022. 2, 3, 8

[22] 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 *ICCV*, 2021. 3

[23] 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. In *ECCV*, 2022. 2, 3, 4, 5

[24] 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. 2, 5

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 2

[26] 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 *ICCV*, 2021. 2, 6, 7

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[28] 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*, 2021. 7

[29] Wenyu Lv, Shangliang Xu, Yian Zhao, Guanzhong Wang, Jinman Wei, Cheng Cui, Yuning Du, Qingqing Dang, and Yi Liu. Detrs beat yolos on real-time object detection. *arXiv preprint arXiv:2304.08069*, 2023. 3

[30] 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. *arXiv preprint arXiv:2212.07784*, 2022. 5

[31] Debapriya Maji, Soyeb Nagori, Manu Mathew, and Deepak Poddar. Yolo-pose: Enhancing yolo for multi person pose estimation using object keypoint similarity loss. In *CVPR Workshop*, 2022. 1, 2, 3, 4, 5, 6, 7

[32] Weian Mao, Zhi Tian, Xinlong Wang, and Chunhua Shen. Fcpose: Fully convolutional multi-person pose estimation with dynamic instance-aware convolutions. In *CVPR*, 2021. 2, 6

[33] 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 *ECCV*, 2022. 3

[34] William McNally, Kanav Vats, Alexander Wong, and John McPhee. Rethinking keypoint representations: Modeling keypoints and poses as objects for multi-person human pose estimation. In *ECCV*, 2022. 2, 5, 6, 7

[35] Xuecheng Nie, Jiashi Feng, Jianfeng Zhang, and Shuicheng Yan. Single-stage multi-person pose machines. In *ICCV*, 2019. 2, 3

[36] Dahu Shi, Xing Wei, Xiaodong Yu, Wenming Tan, Ye Ren, and Shiliang Pu. Inspose: instance-aware networks for single-stage multi-person pose estimation. In *ACMMM*, 2021. 2, 6

[37] Dahu Shi, Xing Wei, Liangqi Li, Ye Ren, and Wenming Tan. End-to-end multi-person pose estimation with transformers. In *CVPR*, 2022. 6, 7

[38] Dahu Shi, Xing Wei, Liangqi Li, Ye Ren, and Wenming Tan. End-to-end multi-person pose estimation with transformers. In *CVPR*, 2022. 2, 3

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

[40] Zhi Tian, Hao Chen, and Chunhua Shen. Directpose: Direct end-to-end multi-person pose estimation. *arXiv preprint arXiv:1911.07451*, 2019. 2, 6

[41] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *CVPR*, 2019. 2

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 4

[43] Dongkai Wang and Shiliang Zhang. Contextual instance decoupling for robust multi-person pose estimation. In *CVPR*, 2022. 2, 6, 7

[44] Jiahong Wu, He Zheng, Bo Zhao, Yixin Li, Baoming Yan, Rui Liang, Wenjia Wang, Shipei Zhou, Guosen Lin, Yanwei Fu, et al. Ai challenger: A large-scale dataset for going deeper in image understanding. *arXiv preprint arXiv:1711.06475*, 2017. 5

[45] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *ECCV*, 2018. 7

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

[47] Jie Yang, Ailing Zeng, Shilong Liu, Feng Li, Ruimao Zhang, and Lei Zhang. Explicit box detection unifies end-to-end multi-person pose estimation. In *ICLR*, 2023. 2, 3, 6, 7

[48] Sen Yang, Zhibin Quan, Mu Nie, and Wankou Yang. Transpose: Keypoint localization via transformer. In *ICCV*, 2021. 3, 7

[49] Yuhui Yuan, Rao Fu, Lang Huang, Weihong Lin, Chao Zhang, Xilin Chen, and Jingdong Wang. HRFormer: High-resolution vision transformer for dense predict. In *NeurIPS*, 2021. 7

[50] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 5

[51] Haoyang Zhang, Ying Wang, Feras Dayoub, and Niko Sunderhauf. Varifocalnet: An iou-aware dense object detector. In *CVPR*, 2021. 4

[52] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 2