Supplementary Material:
Rethinking pose estimation in crowds: overcoming the detection information bottleneck and ambiguity

Mu Zhou* Lucas Stoffl* Mackenzie Weygandt Mathis Alexander Mathis
École Polytechnique Fédérale de Lausanne (EPFL)
alexander.mathis@epfl.ch

Contents

We present a two-stage (hybrid) framework for improving multi-instance pose estimation that is designed to tackle crowded scenarios. Our bottom-up conditioned top-down pose estimation (BUCTD) framework reaches state-of-the-art performance on both human [9, 18] and animal benchmarks [8], outperforming bottom-up, single-stage, top-down, and the recently proposed hybrid top-down methods, such as MIPNet [6].

In the following, we provide additional implementation details, compare the general detection performance of the bottom-up stage, and evaluate the type of conditional training data that gives best CTD results. We conclude by showing that the computational costs of BUCTD compare favorably to the standard detector+TD pipeline, as well as success and failure cases of the BUCTD model.

*Authors contributed equally to this work.

Overall, our work demonstrated that providing individual detections and conditional pose input derived from a bottom-up pose detector to a Conditional Pose Estimator (CTD) can effectively boost performance in crowded scenes.

A. Implementation details

We report training and implementation details for the bottom-up and the conditional top-down models applied to both human benchmarks and animal benchmarks.

A.1. Training settings for bottom-up pose detectors

In order to create pose proposals from bottom-up models, we trained DLCRNet-ms4 [8] on the animal benchmarks, and HigherHRNet-W32 using mmpose [3] on the human benchmarks. We show the default training settings in Table S1.

Hyperparameters	Animal Benchmarks DLCRNet	Human Benchmarks HrHRNet-W32
Optimizer	Adam [7]	Adam [7]
Base learning rate	0.0001	0.0015
Learning rate sched.	step [7500, 12000]	step [200, 260]
Learning rate drop (γ)	[0.5, 0.2]	0.1
Training epochs	-	300
Training iterations	60,000	-
Warmup iterations	-	500
Warmup ratio	-	0.001
Batch size	8	40
Input resolution	400×400	512×512
Rotation	24°	30°
Scale	[0.5, 1.25]	[0.75, 1.5]
RandomFlip	-	0.5

Table S1. **Default training settings for bottom-up models.** We applied these hyperparameters and training data settings to the bottom-up models (DLCRNet for animal datasets, and HigherHRNet for human benchmarks).

Note that the HigherHRNet-W32 is trained with the same settings for COCO [10] (used also for inference on OCHuman [18]) and CrowdPose [9]. We furthermore used

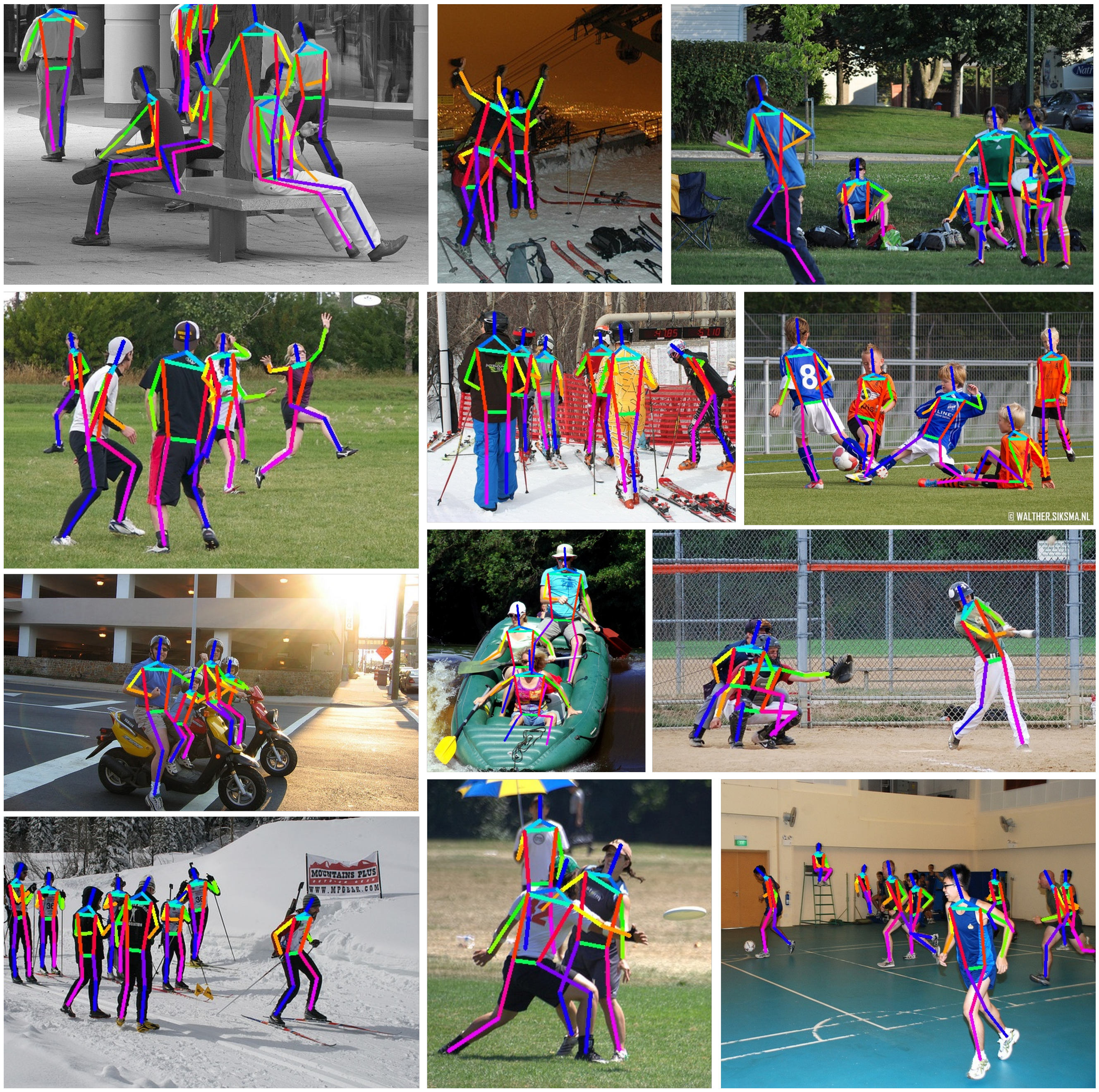

Figure S1. Additional predictions using BUCTD-CoAM-W48 with conditional inputs from PETR on the CrowdPose *test* set.

the same settings of the DLCRNet as in Table S1 on all three animal datasets.

A.2. Training settings for conditional top-down (CTD) pose estimators

Here we provide the default parameter settings for CTD-CoAM-W32/48 and CTD-TP-H-A6 trained on human benchmarks (Table S2).

During training on the animal benchmarks, we used the same settings for CTD-CoAM-W32/W48 (Table S2). However, to keep the aspect ratio for animals, we padded the cropped individuals to the input resolution of 256×256 (and the batch size is 16). For training CTD-CoAM-W32 on COCO [10], we trained for 110 epochs, with an initial learning rate of 0.02 and a learning rate drop at epochs 70 and 110, respectively.

Hyperparameters	CTD-CoAM-W32/48	CTD-TP-H-A6
Optimizer	Adam [7]	Adam [7]
Base learning rate	0.001	0.0001
Weight decay	0.0001	0.1
Learning rate sched.	step [170,200]	step [100,150,200,220]
Learning rate drop (γ)	0.1	0.25
Training epochs	210	240
Batch size	96/48	64
Input resolution	256×192 / 384×288	256×192
Rotation	45°	45°
Scale	[0.65, 1.35]	[0.65, 1.35]
RandomFlip	0.5	0.5

Table S2. **Default training settings for conditional top down models.** We apply these hyperparameters and training data settings to a CTD-CoAM-W32.

A.3. Details for CTD with a Conditional Attention Module

We introduced a new Conditional Attention Module (CoAM) that is inspired by Fu et al. [4] and contains spatial and channel attention sub-modules, which are defined as follows.

Position Attention Module.

Given a local feature f, we first feed it into a convolution layer to obtain $F \in \mathbb{R}^{C\times H\times W}$ and embed it linearly to generate two new features maps K and V (keys and values) with $K, V \in \mathbb{R}^{C\times H\times W}$. The condition heatmap $c \in \mathbb{R}^{3\times H\times W}$ at the corresponding stage is also processed by a convolution layer to create $C \in \mathbb{R}^{3\times H\times W}$ and embedded linearly into Q (queries) with $Q \in \mathbb{R}^{C\times H\times W}$. We reshape queries, keys and values to $\mathbb{R}^{C\times N}$, where $N = H \times W$ is the number of pixels. A softmax layer is applied after a matrix multiplication between the transpose of Q and K, to generate the spatial attention map $S \in \mathbb{R}^{N\times N}$:

$$s_{ji} = \frac{\exp(Q_i \cdot K_j)}{\sum_{i=1}^{N}\exp(Q_i \cdot K_j)} \tag{1}$$

where s_{ji} measures the impact of condition position i on the feature position j. Then, we perform a matrix multiplication between V and the transpose of S and reshape the result to $\mathbb{R}^{C\times H\times W}$ to obtain the final output P of the position attention submodule:

$$P_j = \sum_{i=1}^{N}(s_{ji}V_i) \tag{2}$$

The resulting feature from the position attention submodule has a global contextual view and aggregates the conditional context according to the spatial attention map.

Channel Attention Module.

Each channel map of high level features can be regarded as a keypoint-specific response while the condition itself is a keypoint-specific map. Hence, it is beneficial to learn the associations between these different semantic representations.

Different from the position attention submodule, the channel attention submodule directly calculates the channel attention map $X \in \mathbb{R}^{C\times C}$ from the original features F (treated as key and value) and the condition C (processed by convolution layer to be in $\mathbb{R}^{C\times H\times W}$ and treated as query). Specifically, we reshape both F and C to $\mathbb{R}^{C\times N}$, and then perform a matrix multiplication between F and the transpose of C, followed by a softmax layer to retain the channel attention map $X \in \mathbb{R}^{C\times C}$:

$$x_{ji} = \frac{\exp(C_i \cdot F_j)}{\sum_{i=1}^{C}\exp(C_i \cdot F_j)} \tag{3}$$

where x_{ji} measure the impact of the condition channel i on the feature channel j. Afterwards, we perform a matrix multiplication between the transpose of X and F and reshape the result to $\mathbb{R}^{C\times H\times W}$ to obtain the final output E of the channel attention submodule:

$$E_j = \sum_{i=1}^{C}(x_{ji}F_i) \tag{4}$$

The final feature of the channel attention submodule models the long-range semantic dependencies between conditional keypoints and feature maps.

To obtain the final output M of CoAM, we perform an element-wise sum operation between the original feature map F and the outputs of the respective submodules P and E:

$$M_j = F_j + (P_j + E_j) \tag{5}$$

A.4. Design of the conditional input to CTD

The condition fed to the CTD stage of our BUCTD framework is created as follows:

With the predictions coming from the first stage, we generate a conditional heatmap in ($c \in \mathbf{R}^{H\times W\times 3}$) by using a Gaussian distribution with a standard deviation σ.

We tried several designs for this conditional input: (3D) color heatmap (CM), (1D) gray-scale heatmap (GM), and K-channel single Gaussian heatmaps (KM). We achieved +1.3 mAP with CM, +0.9 mAP with GM, vs. KM BUCTD-CoAM-W32 on CrowdPose.

Therefore, we applied color heatmap as conditions for all models.

A.5. Details of generative sampling scheme during conditional training

Similar to PoseFix [11], during training, we synthesized the pose by using the error statistics described in [14] as prior information to generate noisy pose as conditional inputs. We generated the conditional pose with the four error types of jitter, inversion, swap and miss. For human benchmark, i.e. CrowdPose, we applied the same error probabilities as in PoseFix (which are estimated from COCO and are likely slightly different [14]; despite this we achieve excellent results). For animal benchmarks, we utilized the same error types and tuned the error distribution by running a few different cases; we ended up using jitter error: 0.15 or 0.2 (depending on keypoint validity), miss error: 0.05 or 0.2 (depending on keypoint validity), inversion error: 0.03, swap error: 0.04 or 0.1(depending on keypoint validity). Additionally, we allow swapping keypoints between individuals that do not have any overlap, to simulate wrong assemblies in the bottom-up stage.

Our results demonstrate that the CoAM module leads to improved performance on some animal benchmarks when applying generative sampling (Table S3). However, the preNet module underperforms on the SchoolingFish dataset compared to the baseline results. We further ablate the error types and find that the performance with fewer error types on preNet-W48 on the SchoolingFish dataset is slightly higher than the performance on the models with all error types. Specifically, when we use two types of errors (jitter and swap), we achieve 71.7 AP, while using jitter error only results in 77.0 AP.

From the different results, we observed that the generative sampling strategy is not as stable as empirical sampling on small-scale datasets, likely due to different error statistics between human and animal pose estimation methods (or different body plans). However, combined generative and empirical sampling could be a great strategy to explore in the future.

methods	Marmosets	Sch.Fish	Tri-Mouse
BUCTD-preNet-W48 (DLCRNet)	91.6	62.1	98.4
BUCTD-CoAM-W48 (DLCRNet)	91.6	81.9	99.1
BUCTD-preNet-W48 (DLCRNet)σ	90.4	88.7	98.5

Table S3. Results on animal benchmarks with generative sampling and empirical sampling. σ denotes empirical sampling.

B. Comparisons to MIPNet

In this section, we compare our method with previous SOTA (MIPNet). based on precision and recall. We find higher performance on both metrics with BUCTD, also strong error-correcting capabilities to improve the performance of BU models.

Furthermore, to ablate the influence of the number of detections (which vary widely across different BU models), we only provide the same (amount of) detected bounding boxes as in MIPNet to our CTD models. We notice that our method still outperforms MIPNet, independent of the bottom-up model applied, and with especially large gains on hard frames (i.e. frames with higher crowdedness level).

B.1. Evaluation on ground-truth bounding boxes

First, to take the detectors completely out of the equation, we simply evaluated different models on ground truth bounding boxes (i.e., the same pixel input).

We compare the performance of our BUCTD-CoAM-W32 model on CrowdPose to HRNet and MIPNet when evaluated using ground-truth bounding boxes (Table S4). Note that these models were trained on $train$ and validated on val (as done in [6]). During training we matched the conditions to the GT keypoints and then fed it to the CTD model together with the cropped input image. The same approach is used during testing. Our method outperforms the HRNet baseline and improves upon the MIPNet baseline, that was designed to better handle crowded scenarios. While MIPNet only achieves small improvements over HRNet, our method substantially boosts the AP values, especially on the hard, highly crowded cases (+ 9.0 AP over HRNet and + 6.9 AP over MIPNet). This directly corroborates our choice to provide conditional pose input to boost performance vs. an index.

Method	AP	AP_{easy}	AP_{med}	AP_{hard}
HRNet-W32 [17]	70.0	78.8	70.3	61.7
MIPNet-W32 [6]	71.2	78.8	71.5	63.8
BUCTD-CoAM-W32 (Ours)	**75.2**	**81.4**	**75.3**	**70.7**

Table S4. Our BUCTD model outperforms HRNet and MIPNet on CrowdPose val (using ground-truth bounding boxes). All models are trained on input resolutions of 256x192.

B.2. Performance details - precision and recall

To gain better insights into the performance gains of BUCTD, we computed precision and recall on the CrowdPosetest set (Figure S2) . We compared our model (trained with empirical sampling) to the previous SOTA on CrowdPose: MIPNet. Importantly, we have higher recall and precision than MIPNet for all BU models. Thus, due to its design, BUCTD improves the precision *and recall* for all BU models we tested.

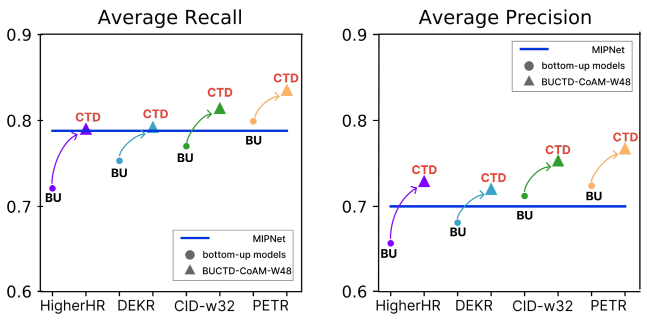

Figure S2. Comparison of recall and precision curves for different BU and CTD models vs. MIPNet. CTD boosts both precision and recall (of BU models), and can thus "recover" more poorly predicted persons than MIPNet.

B.3. Robustness to number of detections

Next, we wanted to fairly compare our BUCTD in terms of the number of detections that the first stage provides, in order to exclude that simply a higher number of detections, made by the bottom-up pose detector in comparison to commonly used object detectors, would lead to our superior performance.

We hence provided the *same number* of detections from the bottom-up models, as provided by the object detector. Despite this artificial constraint the performance of BUCTD was still significantly higher than the one of MIPNet [6] (Figure S3).

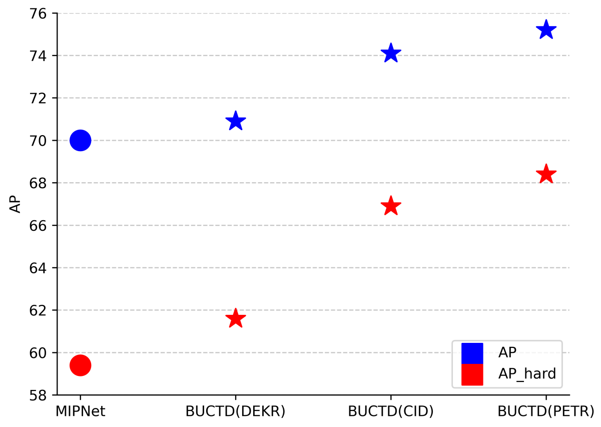

Figure S3. **Performance of BUCTD when provided with same amount of detections (65,044) as the object detector of MIPNet.** When constraining the number of detections we pass from the bottom-up pose detector (DEKR, CID, PETR) to the CTD model to the same amount of detections MIPNet receives from the object detector on the CrowdPose test set, the BUCTD framework still significantly outperforms MIPNet on AP and AP_H (i.e., AP hard).

C. Computational costs of training BUCTD

There are three components for creating a BUCTD model (and comparing it to the standard pipeline either BU or detector + TD). Naturally, simply using a BU model is more efficient, but we also achieve more accuracy.

In contrast, BU models are more efficient than detectors (as we show below). Creating and storing the empirical predictions as well as matching them to ground-truth also comes with some cost (while one does not have these costs with generative sampling; however, one might need to estimate the error distribution). Training CTD or TD models is comparable. Furthermore, the inference of TD and CTD is also comparable (with an advantage for TD). However, those costs come at the benefit of stronger performance – depending on the application performance or speed might be differently relevant.

We compared the parameters and GFLOPs on object detectors, bottom-up models, top-down models, and our methods (Table S5). Bottom-up models generally have fewer parameters and GFLOPs than widely-used object detectors.

We further compare the overall training time of object detectors vs. bottom-up pose detectors. We trained two commonly used object detectors (i.e., Faster R-CNN [13] and YOLOv3 [12]) with the default parameter settings from mmdetection [1] and a DLCRNet [8] as the bottom-up pose detector on the animal benchmarks. For training on the marmosets datasets, training the FasterRCNN detector with 90 epochs took 26.5 hours (saturated around 60 epochs), and training the YOLOv3 with 273 epochs took 33 hours (saturated around 258 epochs). However, training the DLCRNet took only 2.25 hours for 90 epochs. Experiments were performed on a single Titan RTX.

Method	#params	GFLOPs
FasterRCNN [13]	60.0M	246.0
YOLOv3 [12]	62.0M	65.9
HigherHRnet-W32 [2]	28.6M	47.9
DEKR [5]	28.6M	44.5
CID [16]	29.4M	43.2
PETR [15]	220.5M	-
HRNet-W48 [6]	63.6M	19.5
MIPNet-W48 [17]	63.7M	64.5
PoseFix [11]	68.7M	36.6
BUCTD-preNet-W32	28.5M	7.6
BUCTD-TP-H-A6	17.0M	8.4
BUCTD-CoAM-W32	39.1M	8.6
BUCTD-CoAM-W48	115.6M	43.5

Table S5. Number of parameters and GFLOPs on object detectors, bottom-up models, top-down models and our methods.

D. Success and failure cases

To illustrate the power of our method, we show additional qualitative results of success (Figure S1) and failure cases (Figure S4).

References

[1] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 5

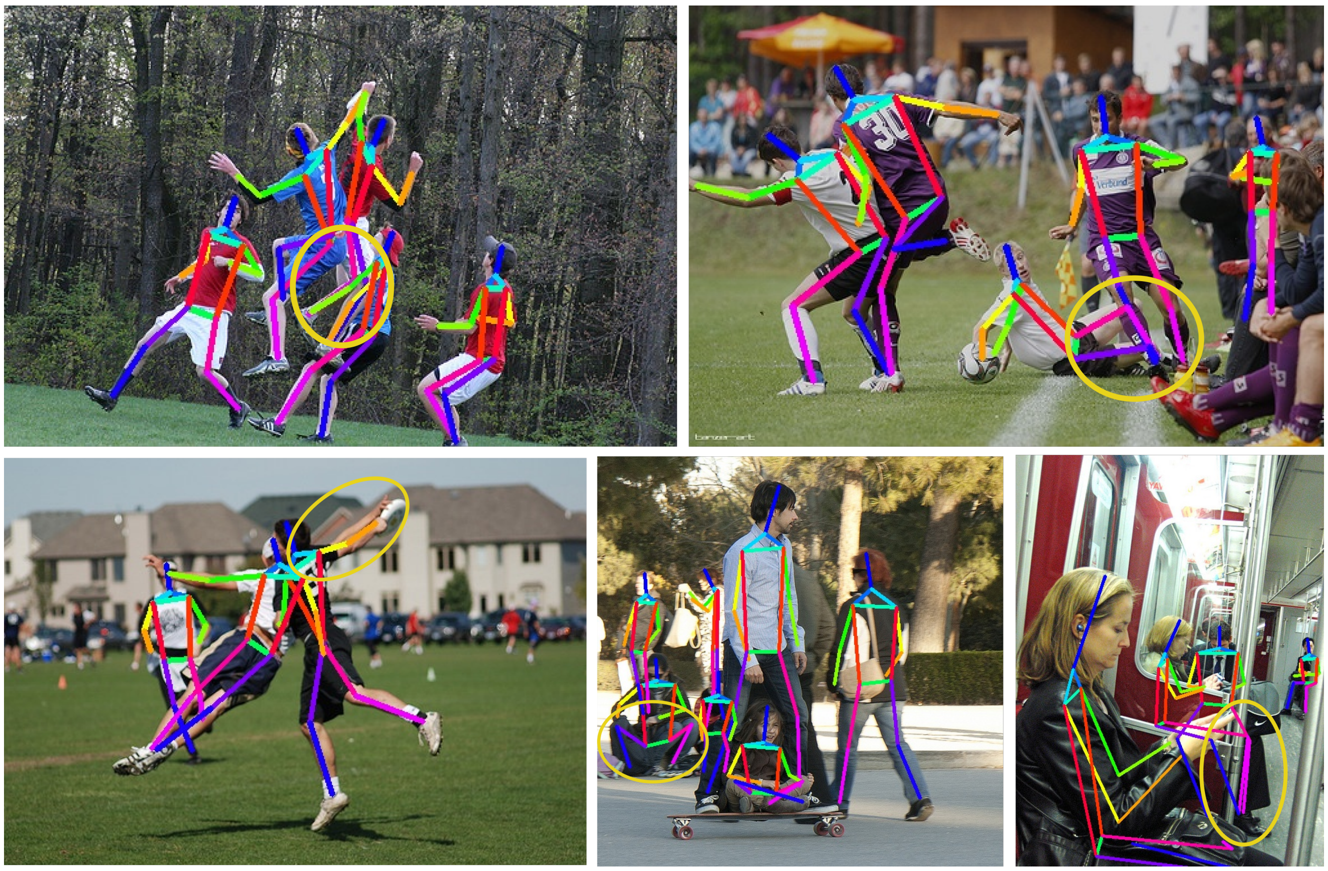

Figure S4. Example failure cases from the CrowdPose *test* set. Especially in complex sport scenes, our BUCTD-CoAM-W48 model sometimes shows inaccuracies in estimating the correct position of the extremities. Errors are highlighted by yellow ellipses.

[2] 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. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5386–5395, 2020. 5

[3] MMPose Contributors. Openmmlab pose estimation toolbox and benchmark. https://github.com/open-mmlab/mmpose, 2020. 1

[4] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3146–3154, 2019. 3

[5] 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*, 2021. 5

[6] Rawal Khirodkar, Visesh Chari, Amit Agrawal, and Ambrish Tyagi. Multi-instance pose networks: Rethinking top-down pose estimation. In *ICCV*, 2021. 1, 4, 5

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1, 3

[8] Jessy Lauer, Mu Zhou, Shaokai Ye, William Menegas, Steffen Schneider, Tanmay Nath, Mohammed Mostafizur Rahman, Valentina Di Santo, Daniel Soberanes, Guoping Feng, et al. Multi-animal pose estimation, identification and tracking with deeplabcut. *Nature Methods*, 19(4):496–504, 2022. 1, 5

[9] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10863–10872, 2019. 1

[10] 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 *European conference on computer vision*, pages 740–755. Springer, 2014. 1, 2

[11] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. Posefix: Model-agnostic general human pose refinement network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7773–7781, 2019. 3, 5

[12] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 5

[13] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun.

Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 5

[14] Matteo Ruggero Ronchi and Pietro Perona. Benchmarking and error diagnosis in multi-instance pose estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 369–378, 2017. 3

[15] 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. 5

[16] Dongkai Wang and Shiliang Zhang. Contextual instance decoupling for robust multi-person pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11060–11068, 2022. 5

[17] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition, 2020. 4, 5

[18] Song-Hai Zhang, Ruilong Li, Xin Dong, Paul Rosin, Zixi Cai, Xi Han, Dingcheng Yang, Haozhi Huang, and Shi-Min Hu. Pose2seg: Detection free human instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 889–898, 2019. 1