FCPose: Fully Convolutional Multi-Person Pose Estimation with Dynamic Instance-Aware Convolutions

Weian Mao[1], Zhi Tian[1], Xinlong Wang[1], Chunhua Shen[1,2*]
[1] The University of Adelaide, Australia [2] Monash University, Australia

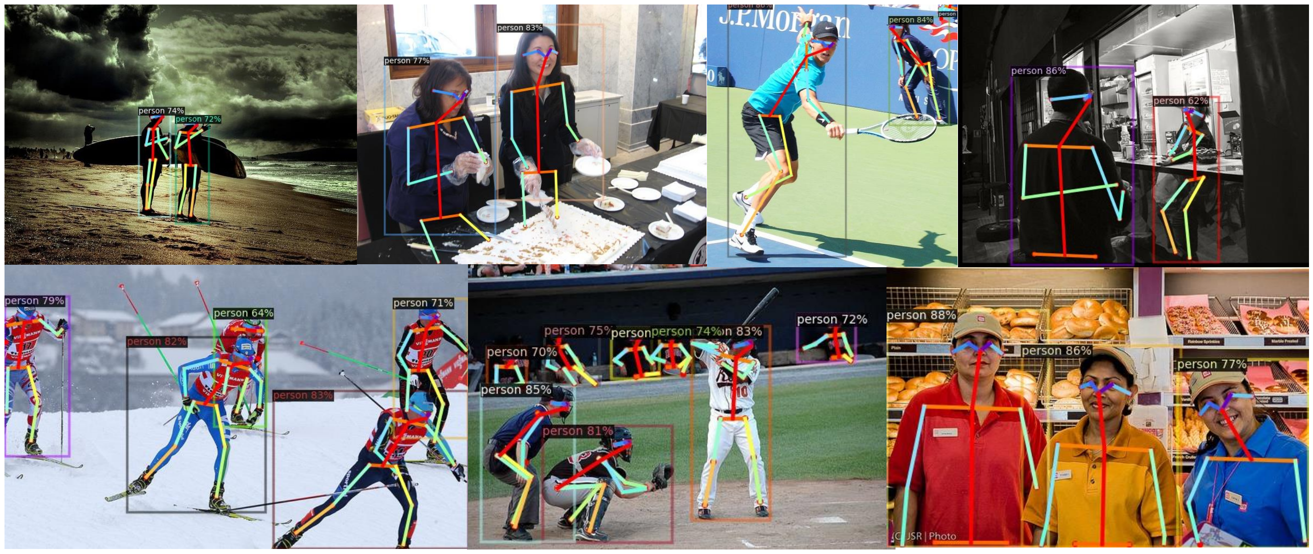

Figure 1 – Qualitative results of FCPose. The results are obtained using the ResNet-101 based FCPose, achieving 65.6% AP^{kp} on the MS-COCO `test-dev` split.

Abstract

We propose a fully convolutional multi-person pose estimation framework using dynamic instance-aware convolutions, termed FCPose. Different from existing methods, which often require ROI (Region of Interest) operations and/or grouping post-processing, FCPose eliminates the ROIs and grouping post-processing with dynamic instance-aware keypoint estimation heads. The dynamic keypoint heads are conditioned on each instance (person), and can encode the instance concept in the dynamically-generated weights of their filters. Moreover, with the strong representation capacity of dynamic convolutions, the keypoint heads in FCPose are designed to be very compact, resulting in fast inference and making FCPose have almost constant inference time regardless of the number of persons in the image. For example, on the COCO dataset, a real-time version of FCPose using the DLA-34 backbone infers about 4.5× faster than Mask R-CNN (ResNet-101) (41.67 FPS vs. 9.26 FPS) while achieving improved performance (64.8% AP^{kp} vs. 64.3% AP^{kp}). FCPose also offers better speed/accuracy trade-off than other state-of-the-art methods. Our experiment results show that FCPose is a simple yet effective multi-person pose estimation framework. Code is available at: `https://git.io/AdelaiDet`

1. Introduction

Multi-person pose estimation (*a.k.a.*, keypoint detection) aims to obtain the keypoint locations of the persons in an image, which is one of the fundamental computer vision tasks with many downstream applications.

The key challenge in multi-person keypoint detection is how to obtain the *instance-level* keypoints. In other words, the detected keypoints need to be grouped according to the instance that they belong to. Currently, the mainstream methods tackle this challenge with bottom-up or top-down

*Corresponding author (email: `chunhua@icloud.com`).

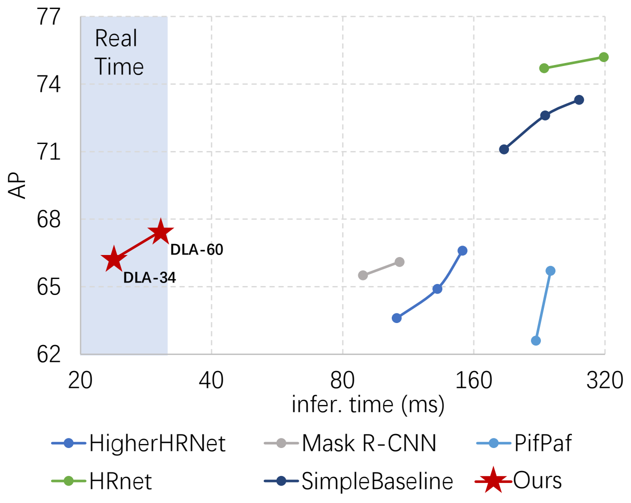

Figure 2 – The trade-off between speed and accuracy. Inference time is measured on a single 1080Ti GPU. We report the AP^{kp} on the COCO `val2017` dataset. FCPose (DLA-34 backbone [33]) is up to 13$\times$ faster than previous state-of-the-art methods and achieves real-time speed with competitive accuracy. It is worth noting that in the fast/real-time keypoint detection realm, which is previously dominated by bottom-up methods, FCPose surpasses them both in speed and accuracy. This suggests that FCPose serve as a new strong baseline for real-time keypoint detection.

approaches. Top-down methods [8, 27] first detect each individual instance with a person detector. These detected boxes form the ROIs and an ROI operation is used to crop the person from either the feature maps or the original image. Next, single person keypoint detection is performed within a ROI for each person, individually. The ROI-based pipeline may come with some drawbacks. First, the ROIs are forwarded separately and thus the convolutional computation cannot be shared. As a result, the inference time of these methods heavily depends on the number of instances in the image, which impedes these methods from being real-time, as shown in our experiments. Second, top-down methods usually are not end-to-end trainable since the ROIs are often obtained from an isolated person detector such as Faster R-CNN [25]. Moreover, the use of the isolated detector also results in significantly longer end-to-end inference time (*i.e.*, from the raw image to the final keypoint results). Third, these ROI-based methods also rely on the localization quality of the ROIs. This may harm the keypoint detection performance if the detector yields inaccurate boxes. On the other hand, bottom-up methods [2, 20] do not rely on ROIs. They first detect instance-agnostic keypoints and then employ grouping post-processing to obtain the full-body results for each instance. The processing of assembling the keypoints is usually heuristic and can involves many hyper-parameters, making these methods complicated.

In this work, we propose a new solution to keypoint detection. Our solution is simple yet effective, and is able to avoid the shortcomings of the previous ROI-based or grouping-based methods. *The key idea of our solution is to use the keypoint heads whose convolution filters/weights are dynamically generated.* More specifically, for each instance, we dynamically generate a keypoint head. The generated keypoint head is applied to convolutional feature maps in the fashion of fully convolutional networks. As a result, we are able to obtain the keypoint detection results *only for that specific target instance*, as shown in Fig. 3. This is made possible as the keypoint head can encode the instance's characteristics in the filters' weights. Thus, this keypoint head can distinguish the instance's keypoints from that of other instances, hence *instance-specific convolution filters*. If we consider that the ROI operations in top-down methods are the operations making the model attend to an instance, then in our method, so do the dynamic instance-aware keypoint heads. This idea eliminates the need for ROIs and the grouping post-processing, thus bypassing the drawbacks mentioned before. Additionally, the dynamically-generated keypoint head is very compact and only has several thousand coefficients. Thus, it can infer very fast, making the overall inference time almost remain the same regardless of the number of persons in a test image. This is particularly valuable for real-time applications.

Moreover, it is easy to see that the localization precision of keypoint detection is tightly related to the output resolution of the FCN. Typically, the output resolution of the fully convolutional keypoint heads is designed to that of the input feature maps (*e.g.*, $^1/_8$ of the input image), which is not sufficient for keypoint detection. Simply using deconvolutions, as in Mask R-CNN [8], to upsample the outputs would inevitably result in significantly increased computation overhead. Here, we tackle this dilemma of accuracy vs. computation complexity by proffering a new keypoint refinement module. As shown in Table 4, compared with the baseline, our proposed keypoint refinement can dramatically improve the accuracy (56.2% *vs.* 63.0% AP^{kp}) with negligible computation overhead (69 ms *vs.* 71 ms).

We summarize our contributions as follows.

- We propose an efficient and accurate human pose estimation framework, termed **FCPose**, built upon dynamic filters [13]. For the first time, we demonstrate that an ROI-free and grouping-free end-to-end trainable human pose estimator can achieve even better accuracy and speed, comparing favourably with recent top-down and bottom-up methods.
- The elimination of ROIs enables FCPose to be implemented by only convolution operations, resulting in much easier deployment in cases such as on mobile devices. Moreover, not using ROIs also avoids that the keypoint prediction is truncated by the inaccurate

detected boxes as in ROI-based frameworks such as Mask R-CNN [8] (*e.g.*, see Fig. 5).

- The core of FCPose is the use of the dynamic filters in our keypoint heads. Dynamically generated filters have demonstrated strong representation capacities. Thus, we only need a small number of such convolutions for achieving top results. As a result, the keypoint heads are very compact and thus the overall inference time is fast and almost constant regardless of the number of the instances in the image. We also present a real-time end-to-end keypoint detection models with competitive performance. The trade-off between speed and accuracy is shown in Fig. 2. We believe that FCPose can be a new strong baseline for keypoint detection, particularly in the real-time realm.

2. Related Work

Multi-person pose estimation. Multi-person pose estimation is often solved using either top-down or bottom-up approaches. Almost all top-down methods first obtain the boxes of the person instances with an object detector [7,8,11,12,23,27,31]. Then, the boxes are used to crop the image patches (*i.e.*, ROIs) from the input image. It is expected that the cropped image patch should include only one person instance. A single-person estimation method is applied to the cropped image patch to attain keypoint locations. These methods can work well. The main drawback is the slow inference speed because they do not share the computation and features with the person detector. Thus the second-stage pose estimation can be very slow when the number of person instances in the image is large. Instead of cropping the ROIs from the original image and recomputing the features for them, Mask R-CNN [8] proposes the ROIAlign operation, which can directly obtain the features of the ROIs from the feature maps of the detector. Thus, it can share the features between the ROIs and the detector, significantly speeding up the inference.

Bottom-up methods [3,5,6,24] usually detect all the keypoints in an instance-agnostic fashion, and then a grouping post-processing is used to obtain the instance-level keypoints. For example, CMU-Pose [2] proposes Part Affinity Fields (PAFs) to group the keypoints,. The authors of [20] employ the associative embedding (AE) to assemble them. Compared to top-down methods, bottom-up methods are often faster because it computes all the convolutional features once, being fully convolutional models.

Furthermore, built upon the anchor-free detector FCOS [29], DirectPose [28] proposes to directly regress the instance-level keypoints by considering the keypoints as a special bounding-box with more than two corners. DirectPose can also eliminate the ROIs and grouping post-processing. Compared to DirectPose, *FCPose takes advantage of the dynamic keypoint head and achieves significantly better performance.*

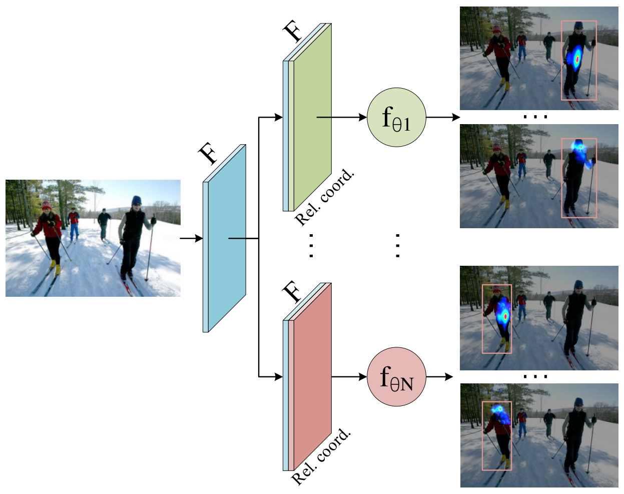

Figure 3 – The core idea of the dynamic keypoint head in FCPose. F denotes a level of feature maps. "Rel. Coord." means the relative coordinates, denoting the relative offsets from the locations of F to the location where the filters are generated. Refer to the text for details. f_{θ_i} is the dynamically-generated keypoint head for the i-th person instance. Note that each person instance has its own keypoint head.

Dynamic filters and conditional convolutions. The core idea of dynamic filter networks [13] and CondConv [32] is to dynamically generate the weights of the convolutions. This is different from the traditional convolutional networks, whose weights are fixed once trained. Since the weights are dynamically-generated and only used once, the model can have strong representation capacity, even with fewer parameters. Moreover, the dynamic filters can be conditioned on each instance in the image, which can make the filters only fire for the target instance. Thus, it can be viewed as a new operation that makes a model attend to the instance, thus replacing the previous ROI operations.

3. Our Approach

3.1. Overall Architecture

In previous works, multi-person keypoint detection is often solved as per-pixel heatmap prediction with FCNs [19]. Since the vanilla FCNs cannot produce instance-aware keypoints, which poses the key challenge in multi-person keypoint detection. As mentioned before, some of methods use an ROI to crop the person of interest and then reduce the multi-person keypoint detection to the single-person one. Formally, let $G \in \mathbb{R}^{h \times w \times c}$ be the features of an ROI, and f_θ be the keypoint head, where θ is the learnable network weights. The predicted heatmaps $H \in \mathbb{R}^{h \times w \times K}$ are

$$H = f_\theta(G). \tag{1}$$

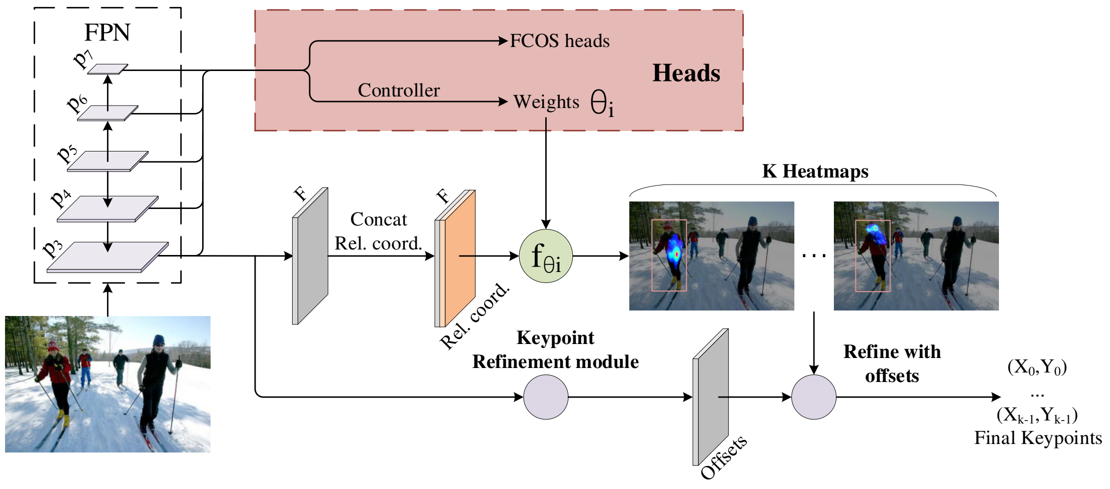

Figure 4 – The overall framework of FCPose. FCPose is built on the one-stage object detector FCOS. The controller that generates the weights of the keypoint heads is attached to the FCOS heads. The weights θ_i generated by the controller is used to fulfill the keypoint head f for the instance i. Moreover, a keypoint refinement module is introduced to predict the offsets from each location of the heatmaps to the ground-truth keypoints. Finally, the coordinates derived from the predicted heatmaps are refined by the offsets predicted by the keypoint refinement module, resulting in the final keypoint results. "Rel. coord." is a map of the relative coordinates from all the locations of the feature maps F to the location where the weights are generated. The relative coordinate map is concatenated to F as the input to the keypoint head.

Note that K being 17 on COCO is the number of keypoints for an instance. Then, the final keypoint coordinates can be obtained by finding the peak on each channel of the heatmaps. *The ROI operation is the core operation making the model attend to an instance.* In this work, we propose to employ the instance-aware keypoint heads to make the model attend to an instance. For each instance, i, a new set of weights θ_i of the keypoint head will be generated. The keypoint head with weights θ_i is applied to full-image feature maps.

Formally, let $F \in \mathbb{R}^{H\times W\times 32}$ be a level of feature maps, which are generated by applying a few conv. layers to the FPN output feature maps [16] and have the same resolution of P_3 in the FPN, as shown in Fig. 4. For the instance i, the predicted heatmaps $H \in \mathbb{R}^{H\times W\times K}$ are

$$H = f_{\theta_i}(F). \tag{2}$$

Note that F is the full-image feature maps without any cropping operations. The filters' weights θ_i are conditioned on the features of the instance i, and thus it can encode the characteristics of the target instance. This makes it possible that the keypoint head only fires at the keypoints of the instance, as shown in Fig. 3.

In this work, we use FCOS [29] to generate the dynamic weights θ for each instance. To this end, we add a new output branch to the box regression branch of FCOS (*i.e.*, the controller shown in Fig. 4). Recall that in FCOS, each location (if considered positive) on the feature maps is associated to an instance. Thus, if a location is associated to the instance i, the controller can generate the weights θ_i that are used to detect the keypoints of the instance i. In this paper, the controller is a single convolutional layer with kernel size 1×1. The number of outputs of the layer is equal to the number of the weights in the keypoint head (*i.e.*, the cardinality of θ_i). In this paper, the keypoint head has $3\times$ conv. layer with channel 32 and kernel size 1×1, followed by ReLU, as well as a K-channel final prediction layer (one channel for one keypoint), which has 2, 737 weights in total. Therefore, the final prediction layer of the controller has 2, 737 output channels. It is worth noting that our keypoint head is much more compact than other top-down methods, for example, Mask R-CNN has $8\times$ conv. layers with channel 512 and kernel size 3×3 as well as deconv. layers in the keypoint head. The very compact keypoint head makes our method take negligible inference time on the keypoint head and thus the overall inference time is almost constant regardless of the number of the instances in the image, as shown in Fig. 6.

Moreover, provided that the model is predicting the instance with the filters generated at the location (x, y), it is obvious that the locations on F that are far from (x, y) are less likely to be the keypoints for the instance. Thus, inspired by CoordConv [18], we append the relative offsets to the feature maps F, as shown in Fig. 3 and Fig. 4, which denote the distances from each location of F to the location (x, y). The keypoint head takes as input the augmented feature maps. This improves the performance remarkably.

3.2. Keypoint Refinement with Regression

As mentioned before, the FPN feature maps P_3 is used to generate the heatmaps, and thus the resolution of the heatmaps is $1/8$ resolution of the input image. Since keypoint detection requires high localization precision, the $1/8$ resolution is not sufficient for keypoint detection. In most of previous methods, an upsampling operation such as deconvolution is often used to upsample the heatmaps. However, upsampling the heatmaps comes with high computational overheads in FCPose. Specifically, in FCPose, for an image, we output N heatmaps with channel K, height H, and width W, where K is the number of keypoints of an instance and N is the number of the instances in the image. These heatmaps will occupy $N \times K \times H \times W$ memory footprint. If we upsample the heatmaps by 8 times, the memory footprint will be increased by 64 times. Also, this will result in much longer computational time.

Here, we address this issue by introducing a regression-based keypoint refinement module. As shown in Fig. 4, the keypoint refinement module is also applied to the FPN level P_3, which is a single conv. layer with output channel $2K$ (*i.e.*, 34 on COCO). Let $O \in \mathbb{R}^{H \times W \times 2K}$ be the output feature maps of this module. $O_{i,j} = (\Delta x, \Delta y)$ predicts the offsets from the location (i, j) to the nearest ground-truth keypoint. As a result, for a keypoint, if its heatmap's peak is at (i, j), the final coordinates of the keypoint will be $(i + \Delta x, j + \Delta y)$. Experiments show that the refinement module can greatly improve the keypoint detection performance with negligible computational overheads. Note that in our experiments, all instances share the same keypoint refinement module. Although it is possible to dynamically generate the module and make each instance have its own one, we empirically find that using one shared keypoint refinement module is sufficient.

3.3. Training Targets and Loss Functions

Training targets. FCPose is built on the detector FCOS. First, we use the same processing to associate each location on the feature maps with an instance or label the location negative. The classification and box regression training targets of each location are computed as in FCOS. As mentioned before, a location is also required to generate the keypoint head's filters for the associated instance. The generated filters are not explicitly supervised. Instead, we supervise the heatmaps predicted by the keypoint head with the filters, which implicitly supervise the generated filters. Except for the classification outputs, all the other outputs are only supervised at the positive locations. In FCOS, for each batch of images (on the same GPU), we might have up to $\sim$ 500 positive locations. If all these locations are used to generate the filters, it will come with high computational overheads. Therefore, for each batch, we only sample at most $M = 50$ positive locations to generate filters. The M locations are averaged over all the ground-truth instances. For each instance, the positive locations with high confidence will be chosen, and the rest of positive locations will be discarded in the keypoint loss computation.

Loss functions. The loss functions of FCPose consist of three parts. The first part is the original losses of FCOS, which are kept as they are. We refer readers to the paper of FCOS [29] for the details. The second part is the loss function for the heatmap learning. As mentioned before, one heatmap only predicts one keypoint. Therefore, we can use one-hot training target for the heatmap, and the cross entropy (CE) loss with softmax is used as the loss function. To be specific, assume a ground-truth keypoint's coordinates are (x^*, y^*), and the heatmap's resolution is $1/8$ resolution of the input image. Then, for this keypoint, the location $(\lfloor \frac{x-4}{8} \rfloor, \lfloor \frac{y-4}{8} \rfloor)$ on its ground-truth heatmap will be set 1 and other locations will be zeros. Let $H_i^* \in \mathbb{R}^{H \times W}$ be the ground-truth heatmap for the keypoint. The loss function can be formulated as

$$L_{heatmap} = \text{CE}(\text{softmax}(H_i), H^*), \tag{3}$$

where $H_i \in \mathbb{R}^{H \times W}$ is the heatmap predicted by the dynamic keypoint head for this keypoint. Here, both H_i and H_i^* are flatten to a vector, and the cross entropy and softmax are applied to each vector. Finally, for the keypoint offset regression, the mean square error (MSE) is used to compute the difference between the predicted offsets and the ground-truth ones. The overall loss function is the summation of these loss functions. Formally, we have

$$L_{overall} = L_{fcos} + \alpha L_{heatmap} + \beta L_{reg}, \tag{4}$$

where α and β are the loss weights, respectively.

3.4. Inference

Given an input image I, FCPose first forwards the image through the network and obtain the network outputs. Following FCOS, the locations with the classification score greater than 0.05 are chosen as positive locations. One positive location corresponds to one predicted instance. Next, for each location, we compute the generated filters and apply the filters to the feature maps F (as shown in Fig. 4) to obtain the keypoint heatmaps of the instance associated with the location. For each heatmap, we find the coordinates of its peak. Then, we refine the coordinates of the peak by the offsets of the keypoint regression module, and obtain the resulting keypoint coordinates. Finally, non-maximum suppression (NMS) is used to remove the duplicates.

4. Experiments

We train and evaluate FCPose on the COCO 2017 Keypoint Detection benchmark [17], which has $57K$ images

# channels	time	AP^{kp}	AP$^{kp}_{50}$	AP$^{kp}_{75}$	AP$^{kp}_{M}$	AP$^{kp}_{L}$
16	**71**	62.8	85.7	68.6	59.0	69.9
32	**71**	**63.0**	**85.9**	**68.9**	**59.1**	**70.3**
64	**71**	62.6	**85.9**	68.3	58.8	69.9

Table 1 – The effect of the number of channels of the input feature maps to the keypoint head (*i.e.*, the feature maps F in the text). "time": the total inference time per image in milliseconds.

depth	time	AP^{kp}	AP$^{kp}_{50}$	AP$^{kp}_{75}$	AP$^{kp}_{M}$	AP$^{kp}_{L}$
2	**69**	62.8	**86.0**	68.7	59.0	70.0
3	71	**63.0**	85.9	**68.9**	**59.1**	**70.3**
4	72	62.6	85.4	68.5	58.9	69.8

Table 2 – Varying the number of the layers in the dynamic keypoint head (*i.e.*, depth). "time": the total inference time per image in milliseconds.

for training, $5K$ images for validation, and $20K$ images for testing. The dataset includes more than $250K$ person instances with 17 annotated keypoints per person. The Average Precision (AP) based on Object Keypoint Similarity (OKS) is used as the evaluation metric. The ablation studies are evaluated on the `val2017` split. Our main results are reported on the `test-dev` split.

Implementation details. We implement FCPose using `Detecton2` [30]. The models are trained with stochastic gradient descent (SGD) over 8 GPUs. Unless specified, all the experiments use the following training details. Following FCOS [29], ResNet-50 [10] with FPNs [16] is used as the feature extractor. The weights pre-trained on ImageNet are used to initialize the backbone ResNet-50. The newly added layers are initialized with the method in [9]. The learning rate is initially set to 0.01, and it is reduced by a factor of 10 at iteration $60K$ and $80K$ in the $1\times$ training schedule (*i.e.*, $90K$ iterations), or at $180K$ and $240K$ in the $3\times$ training schedule (*i.e.*, $270K$ iterations). The weight decay, batch size, and momentum are set as 0.0001, 16, and 0.9, respectively. For data augmentation, we apply random crop $[0.4, 1.0]$ (relative range), random flip, and random resizing (the short size of the image is sampled from $[320, 800]$). For inference, we only use single scale of the image. The shorter side of the image is resized to 800 and the longer side is resized to less than or equal to 1333. All the inference time is measured on a single 1080 Ti GPU.

4.1. Ablation Experiments

4.1.1 Architecture of the Dynamic Keypoint Head

Here, we study the effect of the architecture of the dynamic keypoint head on the final keypoint detection. Specifically, we conduct experiments by varying the number of channels of the input feature maps, the number of channels of the keypoint head, and the number of conv. layers in the keypoint head.

width	time	AP^{kp}	AP$^{kp}_{50}$	AP$^{kp}_{75}$	AP$^{kp}_{M}$	AP$^{kp}_{L}$
16	**70**	62.5	85.5	68.0	58.3	70.1
32	71	**63.0**	**85.9**	**68.9**	**59.1**	**70.3**
64	72	62.6	85.5	68.4	**59.1**	69.5

Table 3 – The effect of the number of channels in the dynamic keypoint head (*i.e.*, width). "time": the total inference time per image in milliseconds.

	time	AP^{kp}	AP$^{kp}_{50}$	AP$^{kp}_{75}$	AP$^{kp}_{M}$	AP$^{kp}_{L}$
none	**69**	56.2	83.6	60.8	49.8	66.3
deconv.	135	60.1	84.8	65.7	56.4	67.3
proposed	71	**63.0**	**85.9**	**68.9**	**59.1**	**70.3**

Table 4 – Comparison of various upsampling methods on the COCO `val2017` split. "none": no upsamling methods used. "deconv.": using deconvolutions to upsample the heatmaps. "proposed": using the proposed keypoint refinement module. As shown here, the proposed module achieves much better performance while keeping almost the same inference time as the baseline model without using any upsampling methods.

shared	time	AP^{kp}	AP$^{kp}_{50}$	AP$^{kp}_{75}$	AP$^{kp}_{M}$	AP$^{kp}_{L}$
	71	62.7	85.2	**68.9**	58.8	**70.3**
✓	**71**	**63.0**	**85.9**	**68.9**	**59.1**	**70.3**

Table 5 – Share the keypoint refinement module between instances or not. As shown in the table, both can have the similar performance. We use the shared one in other experiments due to the slightly better performance.

First, we attempt to change the number of channels of the input feature maps (*i.e.*, F mentioned before) of the keypoint head. As shown in Table 1, 16-, 32- and 64-channel input feature maps have roughly the same performance. Among, using 32 channels achieves the best performance. Moreover, we change the number of conv. layers in the keypoint head. As shown in Table 2, the number of conv. layers does not significantly affect the final performance but using 3 convlayers is the best. Finally, Table 3 shows the performance is also insensitive to the number of channels of the keypoint head. We use 32 channels in the keypoint head in other experiments since it has the best performance.

4.1.2 Keypoint Refinement Module

As mentioned before, the proposed keypoint refinement module can largely improve the localization precision while without introducing large computational overheads. We confirm this in this section.

First, if none of the upsampling methods is used, FCPose can only localize the keypoints with the heatmaps that are $1/8$ resolution of the input image. Unsurprisingly, this has low keypoint detection performance (56.22% AP^{kp}), as shown in Table 4. Most of previous methods such as Mask R-CNN employ deconvolutions to improve the resolution of the heatmaps. However, as mentioned before, deconvolutions will significantly increase the com-

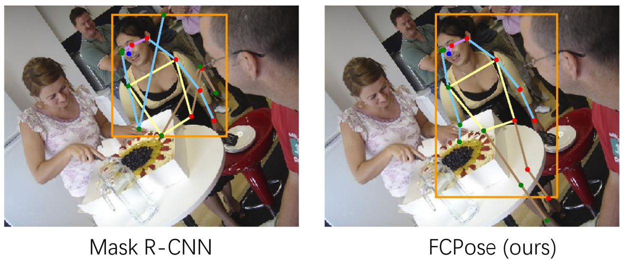

Figure 5 – Comparison of FCPose and the ROI-based Mask R-CNN. As shown in the figure, Mask R-CNN misses some keypoints if the box predicted by the detector is not accurate. In contrast, FCPose can still detect these keypoints since it does not rely on the box.

putational overheads. As shown in Table 4, using deconvolutions in FCPose increases the inference time from 69 ms to 135 ms per image with only 4 points better performance. In contrast, the proposed keypoint refinement module can achieve even better performance than deconvolutions (63.03% AP^{kp}) while keeping almost the same computational time as the model without any upsampling.

Additionally, as mentioned before, we share the keypoint refinement module between all the instances. In principle, it is more reasonable that each instance has its own keypoint refinement module. This is possible by using the dynamic filter technique to generate the keypoint refinement module. However, in Table 5, we empirically show that the shared keypoint refinement module can achieve slightly better performance (62.7% vs. 63.0% AP^{kp}), and thus the shared one is used in all the other experiments. We conjecture that this is because that the situation where multiple persons crowd together is relatively rare, and thus the unshared one does not show remarkable superiority.

4.2. Comparisons with State-of-the-art Methods

In this section, we compare FCPose with other state-of-the-art multi-person pose estimation methods. Unless specified, all the experiments of FCPose in this section use the 3× training schedule, and the performance is reported on the COCO `test-dev` split.

Comparisons with top-down methods. As shown in Table 6, compared with the previous top-down method Mask R-CNN, with ResNet-50, FCPose has better performance (64.3% AP^{kp} vs. 63.9% AP^{kp}). FCPose also has lower computational cost (191.7 GFLOPs vs. 212.7 GFLOPs) and can infer faster (68ms vs. 89ms per image). If ResNet-101 is used as the backbone, the performance of Mask R-CNN can be only improved from 63.9% AP^{kp} to 64.3% AP^{kp} while the performance of FCPose can be boosted from 64.3% to 65.6%. We conjecture that the low resolution of ROIs in Mask R-CNN hampers the performance. On the contrary, FCPose eliminates the ROIs and thus can bypass the issue.

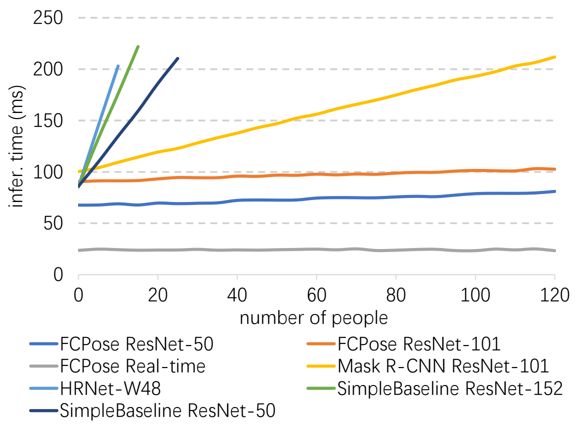

Figure 6 – **End-to-end inference time w.r.t. the number of the persons in an input image.** As shown here, the inference time of previous ROI-based methods significantly increases in the number of instances in the image. In sharp contrast, the inference time needed for FCPose remains almost constant, which is desirable for real-time applications.

There are some other top-down methods such as HR-Net [27] which first employ an isolated object detector to detect the person box and then crop the ROIs on the original image. The features of these ROIs are computed separately by another network. These methods often have high performance but very slow if we measure the end-to-end inference time (*i.e.*, from the input image to the keypoint results). As shown in Table 6, compared the top-down method HRNet, FCPose can significantly reduce the end-to-end inference time from from 337ms to 68ms (ResNet-50) or 488ms to 93ms (ResNet-101) per image, making the keypoint detection nearly real-time. Additionally, since these ROI-based methods use a relatively cumbersome network to obtain the heatmaps for each ROI *separately*, their total inference time heavily depends on the number of the instances. For example, as shown in Fig. 6, the inference time of the model HRNet-W48 significantly increases with the number of the instances. In a sharp contrast, FCPose keeps almost constant inference time. This advantage of FCPose is of great importance to real-time applications.

It is also important to note that FCPose does not rely the boxes predicted by the underlying detector. As a result, the keypoint detection results will not be affected by the inaccurate boxes of the detector. In contrast, the ROI-based methods can only predict the keypoints inside the ROIs. If the detector does not yield an accurate box, some keypoints will be missing. As shown in Fig. 5(left), in Mask R-CNN, the keypoints outside the box are missing. However, as shown in the right figure, FCPose can still correctly detect all the keypoints even if the box is not accurate.

Comparisons with bottom-up methods. Moreover, we also compare FCPose to bottom-up methods [2, 15, 20, 22].

method	backbone	input size	infer. time (ms)	AP^{kp} (%)	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}
Top-down methods								
DirectPose [28]	ResNet-50	800	**74**	62.2	86.4	68.2	56.7	69.8
Mask R-CNN [8]	ResNet-50	800	-	62.7	87.0	68.4	57.4	71.1
Mask R-CNN*	ResNet-50	800	**89**	63.9	87.7	69.9	59.7	71.5
Mask R-CNN*	ResNet-101	800	108	64.3	88.2	70.6	60.1	71.9
G-RMI [23]	ResNet-101	800	-	64.9	85.5	71.3	62.3	70.0
CPN [4]	ResNet-Inc.	384×288	282	72.1	91.4	80.0	68.7	77.2
RSN† [1]	RSN-50	256×192	-	72.5	93.0	81.3	69.9	76.5
SimpleBaseline† [31]	ResNet-152	384×288	430	73.7	91.9	81.1	70.3	80.0
HRNet† [27]	HRNet-W32	384×288	337	74.9	92.5	82.8	71.3	80.9
HRNet†	HRNet-W48	384×288	488	**75.5**	**92.5**	**83.3**	**71.9**	**81.5**
Bottom-up methods								
CMU-Pose [2]	VGG-19 [26]	-	**74**	64.2	86.2	70.1	61.0	68.8
AE [20]	HourGlass [21]	512	-	56.6	81.8	61.8	49.8	67.0
MultiPoseNet‡ [14]	ResNet	800	43	69.6	86.3	76.6	65.0	76.3
HigherHRNet†‡ [6]	HRNet-W48	640	1153	**70.5**	**89.3**	**77.2**	**66.6**	**75.8**
HigherHRNet†	HRNet-W48	640	579	68.4	88.2	75.1	64.4	74.2
HigherHRNet†	HRNet-W32	512	400	66.4	87.5	72.8	61.2	74.2
HigherHRNet	HRNet-W32	640	128	64.7	86.9	71.0	60.2	71.2
Our methods								
FCPose	ResNet-50	800	**68**	64.3	87.3	71.0	61.6	70.5
FCPose	ResNet-101	800	93	**65.6**	**87.9**	**72.6**	**62.1**	**72.3**

Table 6 – Comparisons with recent state-of-the-art methods. † and ‡ denote flipping and multi-sacle testing, respectively. We measure the inference time of other methods on the same hardware if possible. Mask R-CNN* are the results from `Detectron2` [30], which are better than the original results reported in the Mask R-CNN paper [8].

method	time (ms)	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}
CMU-Pose [2]	74	64.2	86.2	70.1	**61.0**	68.8
FCPose (DLA-34)	**24**	64.8	88.4	71.4	59.6	73.3
FCPose (DLA-60)	30	**65.9**	**89.1**	**72.6**	60.9	**74.1**

Table 7 – Comparison of the real-time models on the COCO `test-dev` split. Ours (DLA-34 backbone [33] with 736×512 input resize) is more than 3× faster than the previous strong real-time baseline CMU-Pose while obtaining better performance. With DLA-60, our performance can be boosted by 1.1% AP^{kp} while the inference speed is increased by 6ms.

As shown in Table 6, CMU-Pose [2] takes 74ms per image to infer and achieves 64.2% AP^{kp}, while FCPose takes 67ms per image and has even better performance (64.2% AP^{kp}). FCPose also achieves better or competitive performance with other bottom-up methods but it can infer much faster.

Finally, some qualitative results are shown in Fig. 1, demonstrating that FCPose can work reliably in many challenging cases. These results are based on the ResNet-101 based model.

4.3. Real-time Keypoint Detection with FCPose

We also present a real-time FCPose using DLA-34 [33] as the backbone. For the real-time model, following FCOS [29], the model is trained with 4× training schedule (*i.e.*, 360K iterations). The initial learning is set to 0.01 and it is decayed by a factor of 10 at 300K and 340K, respectively. The shorter side's size of the input images is reduced from 800 to 512. Moreover, the FPN feature levels P_6 and P_7 are removed. We also make use of more aggressive data augmentation during training, *i.e.*, the shorter side' size of the input images is sampled from $[128, 736]$.

The performance of the real-time model is shown in Table 7. Compared to previous strong real-time baseline CMU-Pose [2], our real-time model can run at ∼42 FPS on a single 1080Ti GPU and it is 3 times faster (24ms vs. 74ms) while having better performance (64.8% vs. 64.2% AP^{kp}).

5. Conclusions

We have proposed a novel keypoint detection framework, termed FCPose. It can eliminate the ROI operations in top-down methods and the grouping post-processing in bottom-up methods, solving keypoint detection in the fully convolutional fashion. The core idea of FCPose is to use the dynamic keypoint head instead of ROIs to make the model attend to instances. Extensive experiments demonstrate that FCPose offers a simple, fast and effective keypoint detection framework. Additionally, we have presented a real-time FCPose that can execute at ∼42 FPS on a single 1080Ti GPU with 64.8% AP^{kp} on the COCO dataset, outperforming previous strong real-time baseline CMU-Pose [2] by a large margin.

References

[1] Yuanhao Cai, Zhicheng Wang, Zhengxiong Luo, Binyi Yin, Angang Du, Haoqian Wang, Xinyu Zhou, Erjin Zhou, Xiangyu Zhang, and Jian Sun. Learning delicate local representations for multi-person pose estimation. *arXiv preprint arXiv:2003.04030*, 2020.

[2] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: realtime multi-person 2d pose estimation using part affinity fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.

[3] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 7291–7299, 2017.

[4] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 7103–7112, 2018.

[5] Bowen Cheng, Yunchao Wei, Honghui Shi, Rogerio Feris, Jinjun Xiong, and Thomas Huang. Revisiting RCNN: On awakening the classification power of faster RCNN. In *Proc. Eur. Conf. Comp. Vis.*, pages 453–468, 2018.

[6] Bowen Cheng, Bin Xiao, Jingdong Wang, Honghui Shi, Thomas Huang, and Lei Zhang. HigherHRNet: Scale-aware representation learning for bottom-up human pose estimation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 5386–5395, 2020.

[7] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. RMPE: Regional multi-person pose estimation. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 2334–2343, 2017.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 2961–2969, 2017.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 1026–1034, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 770–778, 2016.

[11] Shaoli Huang, Mingming Gong, and Dacheng Tao. A coarse-fine network for keypoint localization. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 3028–3037, 2017.

[12] Umar Iqbal and Juergen Gall. Multi-person pose estimation with local joint-to-person associations. In *Proc. Eur. Conf. Comp. Vis.*, pages 627–642. Springer, 2016.

[13] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V. Gool. Dynamic filter networks. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 667–675, 2016.

[14] Muhammed Kocabas, Salih Karagoz, and Emre Akbas. Multiposenet: Fast multi-person pose estimation using pose residual network. In *Proc. Eur. Conf. Comp. Vis.*, pages 417–433, 2018.

[15] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. Pifpaf: Composite fields for human pose estimation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 11977–11986, 2019.

[16] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 2117–2125, 2017.

[17] 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 *Proc. Eur. Conf. Comp. Vis.*, pages 740–755. Springer, 2014.

[18] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Proc. Advances in Neural Inf. Process. Syst.*, volume 31, pages 9605–9616, 2018.

[19] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 3431–3440, 2015.

[20] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 2277–2287, 2017.

[21] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *Proc. Eur. Conf. Comp. Vis.*, pages 483–499. Springer, 2016.

[22] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proc. Eur. Conf. Comp. Vis.*, pages 269–286, 2018.

[23] George Papandreou, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, Chris Bregler, and Kevin Murphy. Towards accurate multi-person pose estimation in the wild. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 4903–4911, 2017.

[24] 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 *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 4929–4937, 2016.

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 91–99, 2015.

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[27] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 5693–5703, 2019.

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

[29] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 9627–9636, 2019.

[30] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[31] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proc. Eur. Conf. Comp. Vis.*, pages 466–481, 2018.

[32] Brandon Yang, Gabriel Bender, Quoc V. Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 1307–1318, 2019.

[33] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 2403–2412, 2018.