TFPose: Direct Human Pose Estimation with Transformers

Weian Mao†∗, Yongtao Ge†∗, Chunhua Shen†∗, Zhi Tian‡, Xinlong Wang‡, Zhibin Wang‡
† The University of Adelaide ‡ Alibaba Group

Abstract

We propose a human pose estimation framework that solves the task in the regression-based fashion. Unlike previous regression-based methods, which often fall behind those state-of-the-art methods, we formulate the pose estimation task into a sequence prediction problem that can effectively be solved by transformers. Our framework is simple and direct, bypassing the drawbacks of the heatmap-based pose estimation. Moreover, with the attention mechanism in transformers, our proposed framework is able to adaptively attend to the features most relevant to the target keypoints, which largely overcomes the feature misalignment issue of previous regression-based methods and considerably improves the performance. Importantly, our framework can inherently take advantages of the structured relationship between keypoints. Experiments on the MS-COCO and MPII datasets demonstrate that our method can significantly improve the state-of-the-art of regression-based pose estimation and perform comparably with the best heatmap-based pose estimation methods.

Code is available at: https://git.io/AdelaiDet

1. Introduction

Human pose estimation requires the computer to obtain the human keypoints of interest in an input image and plays an important role in many computer vision tasks such as human behavior understanding.

Existing mainstream methods solving the task can be generally categorized into heatmap-based (Figure 1 top) and regression-based methods (Figure 1 bottom). Heatmap-based methods often first predict a heatmap or a classification score map with fully convolutional networks (FCNs), and then the body joints are located by the peak’s locations in the heatmap or the score map. Most pose estimation methods are heatmap-based because it has relatively higher accuracy. However, the heatmap-based methods may suffer the following issues. 1) A post-processing (e.g., the “taking-maximum” operation) is needed. The post-processing might not be differentiable, making the framework not end-to-end trainable. 2) The resolution of heatmaps predicted by the FCNs is usually lower than the resolution of the input image. The reduced resolution results in a quantization error and limits the precision of the keypoint’s localization. This quantization error might be solved by shifting the output coordinates according to the value of the pixels near the peak, but it makes the framework much more complicated and introduces more hyperparameters. 3) The ground truth heatmaps need to be manually designed and heuristically tuned, which might cause many noises and ambiguities contained in the ground-truth maps, as show in [21, 31, 41].

In contrast, the regression-based methods usually directly map the input image to the coordinates of body joints with a FC (fully-connected) prediction layer, eliminating the need for heatmaps. The pipeline of regression-based methods is much more straightforward than heatmap-based methods as in principle pose estimation is a kind of regression tasks such as object detection. Moreover, the regression-based method can bypass the aforementioned drawbacks of heatmap-based methods, thus being more promising.

Figure 1 – Comparison of mainstream pose estimation pipelines. (a) Heatmap-based methods. (b) Regression-based methods.

*First two authors contributed equally. CS is the corresponding author.
However, there are only a few research works focusing on regression-based methods because regression-based methods often have inferior performance to heatmap-based methods. The reasons may be four-fold. First, in order to reduce the network parameters in the FC layer, in the DeepPose [46], a global average pooling is applied to reduce the feature map resolution before the FC layer. This global average pooling destroys the spatial structure of the convolutional feature maps, and significantly deteriorates the performance. Next, as shown in DirectPose [44] and SPM [35], in regression-based methods, the convolutional features and predictions are misaligned, which results in low localization precision of the keypoints. Moreover, regression-based methods only regress the coordinates of body joints and does not take account of the structured dependency between these keypoints [41].

Recently, we have witnessed the rise of vision transformers [6, 15, 54]. The transformers are originally designed for the sequence-to-sequence tasks, which inspires us to formulate the single person pose estimation to the problem of predicting K-length sequential coordinates, where K is the number of body joints for one person. This leads to a simple and novel regression-based pose estimation framework, termed TFPose (i.e., Transformer-based Pose Estimation). As shown in Figure 2, taking as inputs the feature maps of CNNs, the transformer sequentially predict K coordinates. TFPose can bypass the aforementioned difficulties of regression-based methods. First, it does not need the global average pooling as in DeepPose [46]. Second, due to the multi-head attention mechanism, our method can avoid the feature misalignment between the convolutional features and predictions. Third, since we predict the keypoints in the sequential way, the transformer can naturally capture the structured dependency between the keypoints, resulting in improved performance.

We summarize the main contributions as follows.

• TFPose is the first transformer-based pose estimation framework. Our proposed framework adapts to the simple and straightforward regression-based methods, which is end-to-end trainable and can overcome many drawbacks of the heatmap-based methods.

• Moreover, our TFPose can naturally learn to exploit the structured dependency between the keypoints without heuristic designs, e.g., in [41]. This results in improved performance and better interpretability.

• TFPose achieves greatly advance the state-of-the-art of regression-based methods, making the regression-based methods comparable to the state-of-the-art heatmap-based ones. For example, we improve the previously best regression-based method Sun et al. [42] by 4.4% AP on the COCO keypoint detection task, and Aiden et al. [34] by 0.9% PCK on the MPII benchmark.

2. Related Work

Transformers in computer vision. After being proposed in [47], Transformers have achieved significant progress in NLP (Natural Language Processing) [2, 14]. Recently, Transformers have also attracted much attention in computer vision community. For basic image classification task, ViT [15] apply a pure Transformer to sequential image patches. Expect for image classification, vision Transformer is also widely applied to object detection [6, 54], segmentation [48, 49], pose estimation [22, 23, 28], low-level vision task [8]. More details, we refer to [16]. Specially, DETR [6] and Deformable DETR [54] formulate the object detection task to predict a box set so that object detection model can be trained end-to-end; the Transformer applications in both 3D Hand Pose Estimation [22, 23] and 3D human pose estimation [22, 23] show that Transformer is suitable for modeling human pose.

Heatmap-based 2D pose estimation. Heatmap-based 2D pose estimation methods [4, 5, 9, 10, 17, 26, 33, 40, 51] perform the state-of-the-art accuracy in 2D human pose estimation. Recently, most work, including both top-down and bottom up, are heatmap-based methods. [33] firstly propose a novel network architecture for heatmap-based 2D pose estimation and achieve a excellent performance. [10] propose a new bottom-up method achieve impressive performance in CrowdPose dataset [25] and improved by [31], [4] propose a efficient network achieving the the-state-of-art performance in COCO keypoint detection dataset [29]. However, [42, 44] argue that heatmap-based methods cannot be trained end-to-end, due to the "taking-maximum" operation. Recently, the noise and ambiguity in the ground truth heatmap are found by [31, 41]. [21] finds the heatmap data processing applied by most previous work is biased and proposed an new unbiased data processing method.

Regression-based 2D pose estimation. 2D human pose estimation is naturally a regression problem [42]. However, regression based methods are not accurate as well as heatmap-based methods, thus there are just a few works [7, 35, 41, 42, 44, 46] for it. Apart from that, although some methods, such as G-RMI [36], apply regression method to reduce the quantization errors caused by heatmap, they are essentially heatmap-based methods. There are some work point out the reason of the bad performance of regression-based method. Directpose [44] points out the feature misalignment issue and propose a mechanism to align the feature and the predictions; [41] indicates regression-based method cannot learn the structure-aware information well and proposal a hand-design model for pose estimation to
force regression-based method learn the structure-aware in-
formation better; Sun et al. [42] propose integral regression,
which shares the merits of both heatmap representation and
regression approaches, to avoid non-differentiable postpro-
cessing and quantization error issues.

3. Our Approach

3.1. TFPose Architecture

This work focuses on the single pose estimation task. Fol-
lowing previous works, we first apply a person detector to
obtain the bounding boxes of persons. Then, according to
the detected boxes, each person is cropped from the input
image. We denote the cropped image by \( I \in \mathbb{R}^{h \times w \times 3} \),
where \( h, w \) is the height and the width of the image, re-
spectively. With the cropped image with a single person,
the previous heatmap-based methods apply a convolutional
neural network \( F \) to the patch to predict keypoint heatmaps
\( H \in \mathbb{R}^{h \times w \times k} \) (for \( k^{th} \) joint) of this person, where \( k \) is
the number of the predicted keypoint. Formally, we have

\[
H = F(I),
\]

where \( F \) is composed of three main components: a stan-
dard CNN backbone to extract multi-level feature represen-
tations, a feature encoder to capture and fuse multi-level
features and a coarse-to-fine decoder to generate the a se-
quence of keypoint coordinates. It is illustrated in Figure 2.
Note that our TFPose is fully differentiable and the localiza-
tion precision is not limited by the resolution of the feature
maps.

3.2. Transformer Encoder

As shown in Figure 3, the backbone extracts multi-level
features of the input image. The multi-level feature maps

\[
J = F(I),
\]

where \( F \) is composed of three main components: a stan-
dard CNN backbone to extract multi-level feature represen-
tations, a feature encoder to capture and fuse multi-level
features and a coarse-to-fine decoder to generate the a se-
quence of keypoint coordinates. It is illustrated in Figure 2.
Note that our TFPose is fully differentiable and the localiza-
tion precision is not limited by the resolution of the feature
maps.

Figure 2 – Overall pipeline of TFPose. The model directly predicts a sequence of keypoint coordinates in parallel by combining a com-
mon CNN with a transformer architecture. A transformer decoder takes as input a fix number of keypoint queries and encoder output.
Then, we pass the output embedding of the decoder to a multi-layer feed forward network that predicts final keypoint coordinates.

Figure 3 – Positional encoding. This figure illustrates the po-
sitional embeddings to the input \( F_0 \) of the transformer. \( E_i \)
represents the level embeddings depicting which level a feature vector comes from. \( E_0^p \) represents the pixel embedding depict-
ing the spatial location of a feature vector on the feature maps.
We use \( F_0^E \) to denote \( F_0 \) with position embedding. Following
[54], both \( F_0 \) and \( F_0^E \) are the inputs of the transformer.
are denoted by $C_2$, $C_3$, $C_4$ and $C_5$, respectively, whose strides are 4, 8, 16 and 32, respectively. We separately apply a $1 \times 1$ convolution to these feature maps so that they have the same number of the output channels. These feature maps are flatten and concatenated together, which results in the input $F_0 \in \mathbb{R}^{n \times c}$ to the first encoder in the transformer, where $n$ is the number of the pixel in the $F_0$. Here, we use $F_i$ denotes the output to the $i$-th encoder in the transformer. Following [47, 54], $F_0$ is added with the positional embeddings and we denote $F_0$ with the positional embeddings by $F_0^E$. The details of the positional embeddings will be discussed in Section 3.2. Afterwards, both $F_0$ and $F_0^E$ are sent to the transformer to compute the memory $M \in \mathbb{R}^{n \times c}$. With the memory $M$, a query matrix $Q \in \mathbb{R}^{K \times c}$ will be used in the transformer decoder to obtain the K body joints’ coordinates $\mathbf{J} \in \mathbb{R}^{K \times 2}$.

We follow Deformable DETR [54] to design the encoder in our transformer. As mentioned before, before $F_0$ is taken as inputs, each feature vector of $F_0$ is added with the positional embeddings. Following Deformable DETR, we use both level embedding $E_l^L \in \mathbb{R}^{1 \times c}$ and pixel position embeddings $E^P \in \mathbb{R}^{n \times c}$. The former encodes the level where the feature vector comes from, and the latter is the feature vector’s spatial location on the feature maps. As shown in Figure 3, all the feature vectors from level $l$ are added with $E_l^L$ and then the feature vectors are added with their pixel position embeddings $E^P$, where $E^P$ is the 2-D cosine positional embeddings corresponding to the 2-D location of the feature vector on the feature maps.

In TFPose, we use $N_E = 6$ encoder layers. For $e^{th}$ encoder layer, as shown in Figure 4, the previous encoder layer’s outputs will be taken as the input of this layer. Following Deformable DETR, we also compute the pixel-to-pixel attention between the output vectors of each encoder layer (denoted by ‘p2p attention’). After $N_E$ transformer encoder layers are applied, we can obtain the memory $M$.

### 3.3. Transformer Decoder

In the decoder, we aim to decode the desired keypoint coordinates from the memory $M$. As mentioned before, we use a query matrix $Q \in \mathbb{R}^{K \times c}$ to achieve this. $Q$ is essentially an extra learnable matrix, which is jointly updated with the model parameters during training and each row of which corresponds to a keypoint. In TFPose, we have $N_D$ transformer decoder layers. As shown in Figure 4, each decoder layer takes as input the memory $M$ and the outputs of the previous decoder layer $Q_{d-1} \in \mathbb{R}^{K \times c}$. The first layer takes as inputs $M$ and the matrix $Q$. Similarly, $Q_{d-1}$ is added with the positional embeddings. The result is denoted by $Q^E_{d-1}$. The $Q_{d-1}$ and $Q^E_{d-1}$ will be sent to the query-to-query attention module (denoted as ‘q2q attention’), which aims to model the dependency between human body joints. The q2q attention module use $Q_{d-1}$, $Q^E_{d-1}$ and $Q^E_{d-1}$ as values, queries and keys, respectively. Later, the output of the q2q attention module and $M$ used to compute the pixel-to-query attention (denoted as ‘p2q attention’) with the value being the former and query being the latter. Then, an MLP will be applied to the output of p2q attention the output of the decoder $Q_d$. The keypoint coordinates are predicted by applying an MLP with output channels being 2 to each row of $Q_d$.

Instead of simply predicting the keypoint coordinates in the final decoder layer, inspired by [7, 20, 54], we require all the decoder layers to predict the keypoint coordinates. Specifically, we let the first decoder layer directly predict the target coordinates. Then, every other decoder layer refines the predictions of its previous decoder layer by predicting refinements $\hat{y}_{d-1} \in \mathbb{R}^{K \times 2}$. In that way, the keypoint coordinates can be progressively refined. Formally, let $y_{d-1}$ be the keypoint coordinates predicted by the $(d-1)$-th decoder layer, the predictions of the $d$-th decoder layer are

$$y_d = \sigma(\sigma^{-1}(y_{d-1}) + \Delta \hat{y}_{d}),$$

where $\sigma$ and $\sigma^{-1}$ denote the sigmoid and inverse sigmoid function, respectively. $\hat{y}_{0}$ is a randomly-initialized matrix and jointly updated with model parameters during training.

### 3.4. Training Targets and Loss Functions

The loss functions of TFPose consist of two parts. The first part is the $\mathcal{L}_1$ regression loss. Let $y \in \mathbb{R}^{K \times 2}$ be the ground-truth coordinates. The regression loss is formulated as,

![Figure 4 – Transformer architecture.](image-url)

During training, de-convolution modules are used to upsample transformer encoder output ($M_{C5}$) for for auxiliary loss. During testing, only output Transformer decoder. ‘Norm’ represent normalization; $(X_i, Y_i)$ represent the coordinate for $i^{th}$ keypoint.
where \( N_D \) is the number of the decoders, and every decoder layer is supervised with the target keypoint coordinates. The second part is an auxiliary loss \( L_{aux} \). Following DirectPose \([44]\), we use the auxiliary heatmap learning during training \(^1\), which can result in better performance. In order to use the heatmap learning, we gather the feature vectors that were \( C_5 \) from \( M \) and reshape these vectors into the original spatial shape. The result is denoted by \( M_{C_5} \in \mathbb{R}^{(h/32)\times(w/32)\times c} \). Similar to simple baseline \([51]\), we apply \( 3\times \) deconvolution to \( M_{C_5} \) to up-sample the feature maps by 8 and generate the heatmap \( \hat{H} \in \mathbb{R}^{(h/4)\times(w/4)\times K} \). Then, we compute the mean square error (MSE) loss between the predicted and ground-truth heatmaps. The ground-truth heatmaps are generated by following \([33, 52]\). Formally, the auxiliary loss function is

\[
L_{aux} = ||H - \hat{H}||^2, \tag{6}
\]

We sum the two loss functions to obtain the final overall loss

\[
L_{overall} = L_{reg} + \lambda L_{aux}, \tag{7}
\]

where \( \lambda \) is a constant and used to balance the two losses.

4. Experiments

4.1. Implementation details.

Datasets. We conduct a number of ablation experiments on two mainstream pose estimation datasets.

Our experiments are mainly conducted on COCO2017 Keypoint Detection \([50]\) benchmark, which contains about 250\(K \) person instances with 17 keypoints. Following common settings, we use the same person detector in Simple Baseline \([52]\) for COCO evaluation. We report results on the val set for ablation studies and compare with other state-of-the-art methods on the test-dev set. The Average Precision (AP) based on Object Keypoint Similarity (OKS) is employed as the evaluation metric.

Besides COCO dataset, we also report results on MPII dataset \([1]\). MPII is a popular benchmark for single person 2D pose estimation, which has 25\(K \) images. In total, there are 29\(K \) annotated poses for training, and another 7\(K \) poses for testing. The Percentage of Correct Keypoints (PCK) metric is used for evaluation.

Model settings. Unless specified, ResNet-18 \([18]\) is used as the backbone in ablation study. The size of input image is 256 \times 192 or 384 \times 288. The weights pre-trained on ImageNet \([13]\) are used to initialize the ResNet backbone. The rest parts of our network are initialized with random parameters. For the Transformer, we adopt Deformable Attention Module proposed in \([54]\) and the same hyper-parameters are used.

Training. All the models are optimized by AdamW \([30]\) with a base learning rate of \( 4 \times 10^{-3} \). \( \beta_1 \) and \( \beta_2 \) are set to 0.9 and 0.999. Weight decay is set to \( 10^{-4} \). \( \lambda \) is set to 50 by default for balancing the regression loss and auxiliary loss. Unless specified, all the experiments use a cosine learning schedule with base learning rate \( 4 \times 10^{-3} \). Learning rate of the Transformers and the linear projections for predicting the structure-aware information, query-to-query attention is designed to capture structure-aware information across all the keypoints. Unlike \([41]\) which uses a hand-design method to explicitly force the model to learn the structure-aware information, query-to-query attention models human body structure implicitly. To study the effect of query-to-query attention, we report the results of removing the query-to-query attention in all decoder layers. As shown in Table 1, the proposed query-to-query attention improve the performance by 1.3% AP with only 0.1 GFLOPs more computational cost.

Configurations of Transformer decoder. Here we study the effect of width and depth of the decoder. Specifically, we conduct experiments by varying the number of channels

\[ L_{reg} = \sum_{d=1}^{N_D} ||y - \hat{y}_d||, \tag{5} \]

\(^1\)The heatmap branch is removed in inference.
of the input features and the number of decoder layers in Transformer decoder.

As shown in Table 2, Transformers with 256-channel feature maps is 1.3% AP higher than 128-channels ones. Moreover, we change the number of decoder layers. As shown in Table 3, the performance grows at the first three layers and saturates at the fourth decoder layer.

Auxiliary loss. As shown in previous works [15,54,54], the transformer modules may converge slower. To mitigate this issue, we adopt the deformable attention module proposed in [54]. Apart from that, we propose an auxiliary loss to accelerate the convergence speed of TFPose. Here, we investigate the effect of the auxiliary loss. In this experiment, the first model is only supervised by regression loss; the second model is supervised by both regression loss and auxiliary loss. As shown in Figure 6 and Table 4, the auxiliary loss can significantly accelerates the convergence speed of TFPose and boost the performance by a large margin (+2.3% AP).

### 4.3. Discussions on TFPose

**Visualization of sampling keypoints.** To study how the Deformable Attention Module locate the body joints, we visualize the sampling locations of the module on the feature maps $C_3$. In Deformable Attention Module, there are 8 attention heads and every head will sample 4 points on every feature map. So for the $C_3$ feature map, there are 32 sampling points. As shown in Figure 7, the sampling points (red dot) are all densely located nearby the ground truth (yellow circle). This visualization shows that TFPose can address the feature mis-alignment issue in a sense, and supervises the CNN with dense pixel information.

**Visualization of query-to-query attention.** To further study how the query-to-query self-attention module works, we visualize the attention weights of the query-to-query self-attention. As shown in Figure 6, there are two obvious patterns of attention: the first attention pattern is that the symmetric joints (e.g. left shoulder and right shoulder) are more likely to attend to each other, and the second attention pattern is that the adjacent joints (e.g. eyes, nose, and mouth) are more likely to attend to each other.

To have a better understanding of this attention pattern, we also visualize the attention graph between each keypoint according to the attention maps in the supplementary. This attention pattern suggests that TFPose can employ the context and structured relationship between the body joints to locate and classify the types of body joints.

### 4.4. Comparison with State-of-the-art Methods

In this section, we compare TFPose with previous state-of-the-art 2D pose estimation methods on COCO val2017 split, COCO test-dev split and MPII [1]. We compare these method in terms of both accuracy and computational cost. The results of our proposed TFPose and other state-of-the-art methods are listed in Table 5, Table 6 and Table 7.

**Results on COCO val set.** As shown in Table 5, with similar computational cost, TFPose with 4 encoder layers and ResNet-50 surpass the previous regression-based method DeepPose with ResNet-101 (70.5% AP vs. 56.0% AP) by a large margin and even has much better performance than DeepPose with ResNet-152 (70.5% AP vs. 58.3% AP). Besides, TFPose also outperform many heatmap-based methods, for example, 8-stage Hourglass [33](70.5% AP vs. 56.0% AP) by a large margin.
Figure 6 – Visualization of the attention weights of the q2q attention. We average the attention maps over the whole COCO 2017 val dataset. The left map is the attention weights of the second decoder layer. The right map is the attention weights of the third decoder layer. ‘L’ means the joints are in the left. ‘R’ means the joints are in the right. The horizontal axis and the vertical axis represent the input query and key of the attention module, respectively. Multi-head attention computes the attention weights between each pair of the queries and keys. The query attends more to the key with a higher attention weight.

Figure 7 – Visualisation of the sampling point on feature map. There are 17 queries for 17 keypoints. We visualize 12 body joints queries (not including facial joints). Each image correspond to a body joints. Red dot represent the sampling point; yellow circle represent the ground truth.

67.1% AP), CPN [9](70.5% AP vs. 69.4% AP) by a large margin. It is also important to note that TFPose with 4 encoder layers and ResNet-50 outperforms the strong baseline SimpleBaseline [52] with ResNet-50 (70.5% AP vs. 70.4% AP) with lower computational cost (7.68 GFLOPs vs. 8.9 GFLOPs).

Results on COCO test-dev set. As shown in Table 6, TFPose achieves the best result among regression-based methods. Especially, TFPose with 6 encoder layers and ResNet-50 achieves 70.9% AP, which is higher than the Int. Reg. [42] (67.8% AP), and our computational cost is lower than the Int. Reg. (9.15 GFLOPs vs. 11.0 GFLOPs). Moreover, with the same backbone ResNet-50, our TFPose even achieves better performance than the strong heatmap-based method SimpleBaseline (70.5% vs. 70.0% AP) with less computational complexity (7.7 GFLOPS vs. 8.9 GFLOPS). Additionally, the results of TFPose are also close to the best reported pose estimation results. For example, the performance of TFPose (72.2% AP) is close to the ResNet-Inception based CPN(72.1% AP) and ResNet-152 based SimpleBaseline (73.7% AP). Note that they use much larger backbones than ours.
Table 5 – Comparisons with previous works on the COCO val split. For CPN, the results in the brackets are with online hard keypoints mining. All the reported methods use person detectors with similar performance. Specifically, Hourglass and CPN use the person detector with 55.3% AP on COCO. Others use the person detector with 56.4% AP. DeepPose is implemented by the mmpose [12]. Flipping test is applied for all model. $N_D$ represents the number of encoder layers.

Table 6 – Comparisons with state-of-the-art methods on COCO test-dev set. $^\dagger$ and $^\ddagger$ denote flipping and multi-scale testing, respectively. Input size and the GFLOPs are shown for the single person pose estimation methods. 'ResNet-Ince.' represent the ResNet inception. The Simple baseline (ResNet-50) is tested with the official code. $D$ represents the number of encoder layers.

Results on MPII test set. On the MPII benchmark, TFPos also achieves the best results among the regression-based methods. As shown in Table 7, TFPos with ResNet-50 is higher than the method proposed by Aiden et al. [34] (90.4% vs. 89.5%) with the same backbone. TFPos is also comparable to heatmap-based methods.

5. Conclusion

We have proposed a novel pose estimation framework named TFPos built upon Transformers, which largely improves the performance of the regression-based pose estimation and bypasses the drawbacks of heatmap-based methods such as the non-differentiable post-processing and quantization error. We have shown that with the attention mechanism, TFPos can naturally capture the structured relationship between the body joints, resulting in improved performance. Extensive experiments on the MS-COCO and MPII benchmarks show that TFPos can achieve state-of-the-art performance among regression-based methods and is comparable to the best heatmap-based methods.
| Method                   | Head | Sho. | Elb. | Wri. | Hip  | Knee | Ank. | Total |
|-------------------------|------|------|------|------|------|------|------|-------|
| Heatmap-based methods   |      |      |      |      |      |      |      |       |
| Pishchulin et al. [37]  | 74.3 | 49.0 | 40.8 | 34.1 | 36.5 | 34.4 | 35.2 | 44.1  |
| Tompson et al. [45]     | 95.8 | 90.3 | 80.5 | 74.3 | 77.6 | 69.7 | 62.8 | 79.6  |
| Hu et al. [19]          | 95.0 | 91.6 | 83.0 | 76.6 | 81.9 | 74.5 | 69.5 | 82.4  |
| Lifshitz et al. [27]    | 97.8 | 93.3 | 85.7 | 80.4 | 85.3 | 76.6 | 70.2 | 85.0  |
| Raf et al. [38]         | 97.2 | 93.9 | 86.4 | 81.3 | 86.8 | 80.6 | 73.4 | 86.3  |
| Bulat et al. [3]        | 97.9 | 95.1 | 89.9 | 85.3 | 89.4 | 85.7 | 81.7 | 89.7  |
| Chu et al. [11]         | 98.5 | 96.3 | 91.9 | 88.1 | 90.6 | 88.0 | 85.0 | 91.5  |
| Ke et al. [24]          | 98.5 | 96.8 | 92.7 | 88.4 | 90.6 | 89.3 | 86.3 | 92.1  |
| Tang et al. [43]        | 98.4 | 96.9 | 92.6 | 88.7 | 91.8 | 89.4 | 86.2 | 92.3  |
| Zhang et al. [53]       | 98.6 | 97.0 | 92.8 | 88.8 | 91.7 | 89.8 | 86.6 | 92.5  |
| Regression-based methods|      |      |      |      |      |      |      |       |
| Carreira et al. [7]     | 95.7 | 91.7 | 81.7 | 72.4 | 82.8 | 73.2 | 66.4 | 81.3  |
| Sun et al. [41]         | 97.5 | 94.3 | 87.0 | 81.2 | 86.5 | 78.5 | 75.4 | 86.4  |
| Aiden et al. (ResNet-50) | 97.8 | 96.0 | 90.0 | 84.3 | 89.8 | 85.2 | 79.7 | 89.5  |
| Ours (ResNet-50)        | 98.0 | 95.9 | 91.0 | 86.0 | 89.8 | 86.6 | 82.6 | 90.4  |

Table 7 – MPII human pose test set PCKh accuracies. For our model, the number of encoder layers $N_D$ is set to 6.

Acknowledgements

The authors would like to thank Alibaba Group for the donation of GPU cloud computing resources.

References

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In Proc. IEEE Conf. Comput. Vis. Pattern Recog., pages 3686–3693, 2014. 5, 6
[2] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In Proc. Adv. Neural Inform. Process. Syst., 2020. 2
[3] Adrian Bulat and Georgios Tzimiropoulos. Human pose estimation via convolutional part heatmap regression. In Proc. Eur. Conf. Comput. Vis., pages 717–732, 2016. 9
[4] Yuanhao Cai, Zhicheng Wang, Zhengxiong Luo, Binyi Yin, Angang Du, Haoqian Wang, Xiangyu Zhang, Xinyu Zhou, Erjnin Zhou, and Jian Sun. Learning delicate local representations for multi-person pose estimation. In Proc. Eur. Conf. Comput. Vis., pages 455–472. Springer, 2020. 2
[5] 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., 43(1):172–186, 2019. 2, 8
[6] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Proc. Eur. Conf. Comput. Vis., pages 213–229. Springer, 2020. 2
[7] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback. In Proc. IEEE Conf. Comput. Vis. Pattern Recog., pages 4733–4742, 2016. 2, 4, 9
[8] Hanting Chen, Yunhe Wang, Tianyu Guo, Yingpeng Deng, Zhenhua Liu, Siwei Ma, Chunjiang Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. arXiv preprint arXiv:2012.00364, 2020. 2
[9] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Crawled pyramid network for multi-person pose estimation. In Proc. IEEE Conf. Comput. Vis. Pattern Recog., pages 7103–7112, 2018. 2, 7, 8
[10] Bowen Cheng, Bin Xiao, Jingdong Wang, Honghui Shi, Thomas S Huang, and Lei Zhang. Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation. In Proc. IEEE Conf. Comput. Vis. Pattern Recog., pages 5386–5395, 2020. 2, 8
[11] Xiao Chu, Wei Yang, Wanli Ouyang, Cheng Ma, Alan L Yuille, and Xiaogang Wang. Multi-context attention for human pose estimation. In Proc. IEEE Conf. Comput. Vis. Pattern Recog., pages 1831–1840, 2017. 9
[12] MMPose Contributors. Openmmlab pose estimation toolbox and benchmark. https://github.com/open-mmlab/mmpose, 2020. 8
[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In Proc. IEEE Conf. Comput. Vis. Pattern Recog., pages 248–255. Ieee, 2009. 5
[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018. 2
[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020. 2, 6
[16] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunk
A. Qualitative Results of TFPose

We show more qualitative results in Figure 10. TFPose works reliably under various challenging cases.

B. Visualization of Transformer Attentions

B.1. Query-to-query Attention

We observe two obvious query-to-query attention patterns in different decoder layers, termed symmetric pattern and adjacent pattern, respectively. Both patterns exist in all decoder layers, we illustrate them separately for convenience. For symmetric pattern, Figure 8 demonstrates that the correlation between all pairs of symmetric joints in the third decoder layer. For adjacent pattern, Figure 9 explicitly shows that adjacent joints attend to each other in the second decoder layer.

B.2. Multi-scale Deformable Attention

We visualize the learned multi-scale deformable attention modules for better understanding. As shown in Figure 11 and Figure 12, the visualization indicates that TFPose looks at context information surround the ground truth joint. More concretely, the sampling points near the ground truth joint have higher attention weight (denoted as red), while the sampling points far from the ground truth joint own lower attention weight (denoted as blue).
Figure 8 – The pattern of symmetric joints. As shown in the right graph, left shoulder and right shoulder are symmetric joints and they attend to each other. The same pattern can be found in other body joints including left elbow and right elbow, left hip and right hip etc.

Figure 9 – The pattern of adjacent joints. As shown in the right graph, left shoulder attend to its adjacent joints including right shoulder, left elbow, and head. The same pattern can be found in other body joints, e.g., elbow and wrist.
Figure 10 – Qualitative results of TFPose with ResNet-50 on COCO2017 val set (single-model and single-scale testing). The joints in upper body are represented by green and the joints in lower body are represented by blue.
Figure 11 – Visualization of right shoulder’s pixel-to-query attention in the last decoder layer. For readability, we draw the sampling points and attention weights from $C_3$ feature map in different pictures. Each sampling point is marked as a filled circle whose color indicates its corresponding weight. The ground truth joint is shown as yellow cross marker.
Figure 12 – Visualization of right knee’s pixel-to-query attention in the last decoder layer. For readability, we draw the sampling points and attention weights from $C_3$ feature map in different pictures. Each sampling point is marked as a filled circle whose color indicates its corresponding weight. The ground truth joint is shown as yellow cross marker.