Poseur: Direct Human Pose Regression with Transformers

Weian Mao[1] Yongtao Ge[1,2] Chunhua Shen[3] Zhi Tian[1] Xinlong Wang[1]
Zhibin Wang[2] Anton van den Hengel[1]

[1] The University of Adelaide [2] Alibaba Damo Academy [3] Zhejiang University

1 The Effect of Training Schedules

In this section, we conduct experiments to show the effect of training schedules on the Poseur's performance, as shown in Tab. 1. In our paper, we use a longer training schedule (i.e., 325 epochs in total) than other methods, e.g., RLE [2] (270 epochs in total). In Tab. 1, we show that Poseur trained by 275 epochs or 250 epochs can also achieve impressive performance, which is only slightly lower than the fully-trained one in our paper. Thus, a longer training schedule is not the main reason for our superior performance.

Epoch	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}
150	74.1	90.1	81.3	67.4	76.8
175	74.6	90.2	81.7	67.9	77.2
200	74.8	90.3	81.7	68.0	77.6
225	75.0	90.3	81.8	68.2	77.8
250	75.2	90.7	82.3	68.4	78.0
275	75.3	90.3	82.3	68.5	78.2
300	75.4	90.4	82.6	68.6	78.4
325	75.5	90.7	82.7	68.7	78.3

Table 1: The effect of training schedules on the COCO *val* set

2 The Effect of Self-attention

In this section, we perform experiments to explore the effect of the self-attention module in the Poseur decoder. As shown in Tab. 2, the performance drops significantly from 75.5 AP to 74.0 AP when the self-attention module is removed from the decoder. Thus, we conjecture that the self-attention module can effectively model the relationship between different keypoints, improving Poseur's performance.

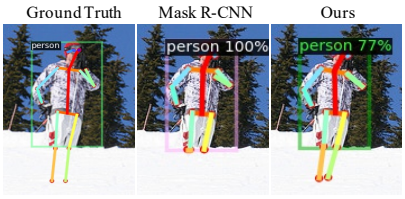

Fig. 1: Qualitative comparison on truncations. Heatmap-based methods (e.g.Mask R-CNN) can only predict keypoints within the bounding box, while Poseur can predict keypoints outside the bounding box

Fig. 2: Visualization of the self-attention weights between keypoint queries for left shoulder. Dots represent the keypoints. Lines depict attention weights between different joints. Thicker line indicates larger attention weight

Self-Attn.	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}
	74.0	90.2	80.9	66.8	77.2
✓	75.5	90.7	82.7	68.7	78.3

Table 2: The effect of self-attention module on the COCO *val* set

Share weight	Param.	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}
	32.3M	75.5	90.7	82.7	68.7	78.3
✓	26.2M	75.0	90.3	81.9	68.0	77.7

Table 3: The parameter reduction technique on the COCO *val* set

Moreover, We also visualize the self-attention weights across queries in Fig. 2. The left shoulder query attends to the most relevant keypoints, including left elbow, left wrist and left ear.

3 Reducing the Number of Parameters

Former works, e.g. Deeppose [5] and RLE [3], use fully-connected layers as decoder to regress keypoints, while Poseur has a transformer-based decoder. As the number of decoder layers increases, the model parameters increases rapidly, which may limit the deployment of Poseur for real-time applications that run on mobile devices.

In this section, we explore reducing the parameters of Poseur by sharing weights between different decoder layers. As shown in Tab. 3, the number of parameters of Poseur is significantly reduced, while the performance of Poseur only drops by 0.5 AP. Notably, the number of parameters of the backbone (ResNet-50) is 23.5 M, which means Poseur with weight sharing only introduces 2.7 M parameters.

type	GFLOPs (Dec.)	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}
MSDA	1.25	73.6	89.8	80.6	66.6	75.5
EMSDA	0.44	73.6	89.6	80.1	66.7	75.4

Table 4: Comparison between EMSDA and MSDA on the COCO *val* set. "GFLOPs (Dec.)": computatuional cost of the decoder

Method	backbone	GFLOPs	FPS	Mem. Consumption	AP^{kp}
RLE	HRNet-w32	7.1	62	1456M	74.3
Poseur	R-50	**4.6**	**94**	**1386M**	**75.4**

Table 5: Comparison between RLE and Poseur on the COCO *val* set. "Mem. Consumption": memory consumption of one image during the training stage

4 Computational Cost of EMSDA

Let us denote the number of queries by K, and denote the number of pixels in the input feature maps $\{\mathbf{x}^l\}_{l=1}^{L}$ as P, and other notations follow our paper. The complexity of MSDA can be written as $O(KC^2 + PC^2 + 5KSC)$. Since P is much larger than K, C and S (i.e., $P = 4080$ when the input image resolution is 256×192 and the feature maps from Res2 to Res5 are taken as the input), the computational cost mostly comes from the factor $O(PC^2)$. In our design, the EMSDA module significantly reduces the complexity to $O(KC^2+KC^2+5KSC)$, where $K \ll P$ (17 *vs.* 4080). As shown in Tab. 4, the performance of EMSDA is almost the same with that of MSDA, while EMSDA significantly reduces the computational cost from 1.25 GFLOPs to 0.44 GFLOPs.

5 Comparing the Performance of Poseur and RLE

As shown in Tab. 5, Poseur with ResNet-50 backbone achieves higher performance than RLE with HRNet-w32 backbone (75.4 AP *vs.* 74.3 AP), and has a faster inference speed than RLE (94 FPS *vs.* 62 FPS). The memory consumption of Poseur during the training is lower then that of RLE (1386 M *vs.* 1456 M). Although the memory consumption of Poseur during the testing is sightly higher than that of RLE (86.25 M *vs.* 68.12 M), the memory consumption of the whole system during the test (human detector and pose estimator) is exactly the same ($\sim$ 2000 M) for most of methods in Tab.10 of the paper, including both Poseur and RLE.

6 Verifying the Effect of Keypoint Encoder and Query Decoder in Poseur

Compared to RLE [3], the proposed keypoint encoder and query decoder (without uncertainty estimation) can boost the performance by 3.8 AP on COCO [4].

This ablation study is performed with ResNet-50 [1]; all the settings are strictly aligned.

7 The Explanation of the Positional Encoding in Keypoint Encoder

Positional encoding in the proposed keypoint encoder transforms the coarse proposal $\hat{\boldsymbol{\mu}}_f \in \mathbb{R}^{K\times 2}$ from the x-y coordinates to the sine-cosine positional embedding. Denote an element in $\hat{\boldsymbol{\mu}}_f$ as pos, which is normalized to $[0, 2\pi]$, The positional encoding function can be written as $PE(pos, 2i) = sin(pos/10000^{2i/d})$; $PE(pos, 2i+1) = cos(pos/10000^{2i/d})$, where $d = 128$, $2i$ and $2i+1$ are the $2i^{th}$ and $(2i+1)^{th}$ dimension. In this way, a pair of x-y coordinates is transferred to two positional embeddings representing x and y axis respectively, which are concatenated to be the final encodings $\hat{\boldsymbol{\mu}}^*_f \in \mathbb{R}^{K\times 256}$.

8 Robustness to Truncation

Truncation is very common in real world scenes. We conduct qualitative visualization to show the superiority of our method. As depicted in Fig. 1, heatmap-based Mask R-CNN can only detect the joints inside the predicted boxes, while our method can infer the joints outside the boxes since the queries can attend to the whole input image.

References

1. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. pp. 770–778 (2016)
2. Li, J., Bian, S., Zeng, A., Wang, C., Pang, B., Liu, W., Lu, C.: Human pose regression with residual log-likelihood estimation. In: Proc. IEEE Int. Conf. Comp. Vis. (2021)
3. Li, J., Bian, S., Zeng, A., Wang, C., Pang, B., Liu, W., Lu, C.: Human pose regression with residual log-likelihood estimation. In: Proc. IEEE Int. Conf. Comp. Vis. (2021)
4. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Proc. Eur. Conf. Comp. Vis. pp. 740–755. Springer (2014)
5. Toshev, A., Szegedy, C.: Deeppose: Human pose estimation via deep neural networks. In: Proc. IEEE Conf. Comp. Vis. Patt. Recogn. pp. 1653–1660 (2014)