Joint COCO and Mapillary Workshop at ICCV 2019: COCO Keypoint Challenge Track

Technical Report: Res-Steps-Net for Multi-Person Pose Estimation

Yuanhao Cai[1,2*] Zhicheng Wang[1*] Binyi Yin[1,3] Ruihao Yin[1,3] Angang Du[1,4] Zhengxiong Luo[1,5] Zeming Li[1] Xinyu Zhou[1] Gang Yu[1] Erjin Zhou[1] Xiangyu Zhang[1] Yichen Wei[1] Jian Sun[1]

[1]Megvii Inc. [2]Tsinghua University [3]Beihang University
[4]Ocean University of China [5]Chinese Academy of Sciences
[1]{*caiyuanhao, wangzhicheng, yinbinyi, yinruihao, duangang, luozhengxiong, lizeming, zxy, yugang, zej, zhangxiangyu, weiyichen, sunjian*}*@megvii.com*

Abstract

Human poses vary greatly in viewpoints and scales in the image. Thus, it is necessary to extract both good global and local features to obtain accurate prediction. In this paper, we propose a novel approach called Res-Steps-Net (RSN) towards this purpose. It outperforms the state-of-the-art methods by a large margin with only COCO labeled data for training and less computation cost with input size 256 × 192. Our single model achieves 78.0 in test-dev. Ensemble models achieve 79.2 in test-dev and 77.1 in test-challenge, which surpasses the winner of COCO Keypoint challenge 2018.

Figure 1: The ResNet-18 and RSN-18 are used as backbone, respectively. The prediction results of ResNet-18 is on the top line, and the prediction results of RSN-18 is on the bottom line.

1. Introduction

Multi-person pose estimation is a fundamental task in computer vision. Existing deep learning based methods fall into two categories: top-down methods and bottom-up methods. Top-down methods first detect the position of all people, then estimate the pose of each person. Bottom-up methods first detect all the human skeletal points in an image and then assemble these points into groups to form different individuals.

The task of human pose estimation contains both localization and classification factors. Local features benefit the localization task, and global features are good for the classification task. Thus, both local and global features should be treated properly. We propose a novel architecture named Res-Steps-Net (RSN) for this problem. We follow the top-down approach and use MegDet[8] as the human detector. Figure 1 illustrates the results comparison between ResNet-18[4] (with 2.3 GFLOPs) and RSN-18 (with 2.5 GFLOPs) as backbone. It is shown that RSN-18 surpasses ResNet-18 by a large extent in performance, especially, in "hard" cases. Our method achieves state-of-the-art performance in COCO Keypoint dataset.

2. Method

The overall pipeline of our method is illustrated in Figure 2. It is based on the pipeline in as [10], with some modifications. The first stage is called init-stage. The latter stages are called refine-stages. The init-stage incorporates refine block (RB) and feature fusion block as shown in Fig-

*The first two authors contribute equally to this work. This work is done when Yuanhao Cai, Binyi Yin, Ruihao Yin, Angang Du and Zhengxiong Luo are interns at Megvii Research.

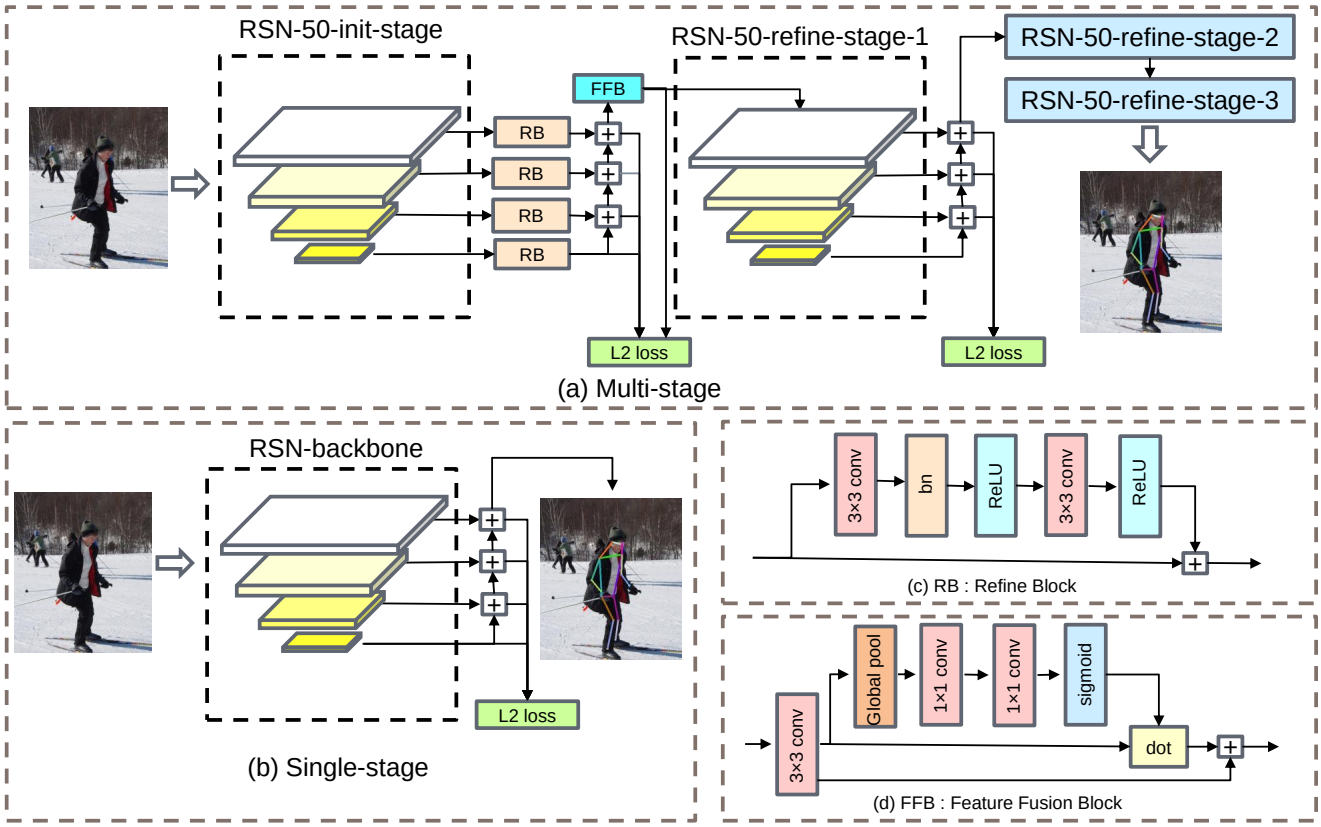

Figure 2: The whole pipeline of RSN. (a) The pipeline of Multi-stage Network. (b) The pipeline of Single-stage network. (c) The components of Refine Block. (d) The components of Feature Fusion Block.

ure 2. The backbone of a single stage is the proposed Res-Steps-Net (RSN), which differs from ResNet in the structure of bottleneck, as shown in Figure 3.

2.1. Res-Steps-Net

Res-Steps-Net is designed for extracting better local and global features. As shown in Figure 3, RSN bottleneck firstly implements a conv1×1 (Convolutional layer with kernel size 1×1), then divides the feature into four splits. Each split feature f_i (i = 1, 2, 3, 4) undergoes incremental numbers of conv3×3 (Convolutional layer with kernel size 3×3). The output features y_i(i = 1, 2, 3, 4) are then concatenated to go through a conv1×1. An identity connection is employed as the ResNet bottleneck. Because the incremental numbers of conv3×3 look like steps, the network is therefore named Res-Steps-Net. RSN is deeply connected in bottleneck, so the low-level features are better supervised, resulting in richer local features in network. The receptive fields of RSN bottleneck are across several scales, and the max one is 15. Compared with one scale in ResNet bottleneck as shown in Table 1, RSN bottleneck indicates more global features are viewed in the network.

We have explored the number of different branch architectures, as shown in Figure 4. The effect of different branch numbers is investigated in the experiment section. Four branches are eventually adopted.

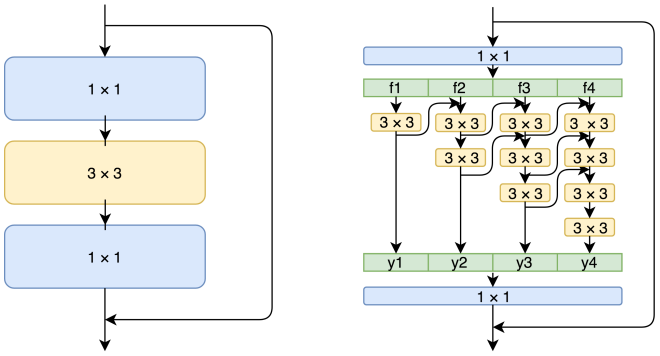

Figure 3: The architecture of the bottleneck of ResNet and RSN. The left one is of ResNet and the right one is of RSN.

2.2. Recursive Formula

As shown in Figure 2, we define the output feature of each branch after passing through certain conv3×3 as $y_{(i,j)}$.

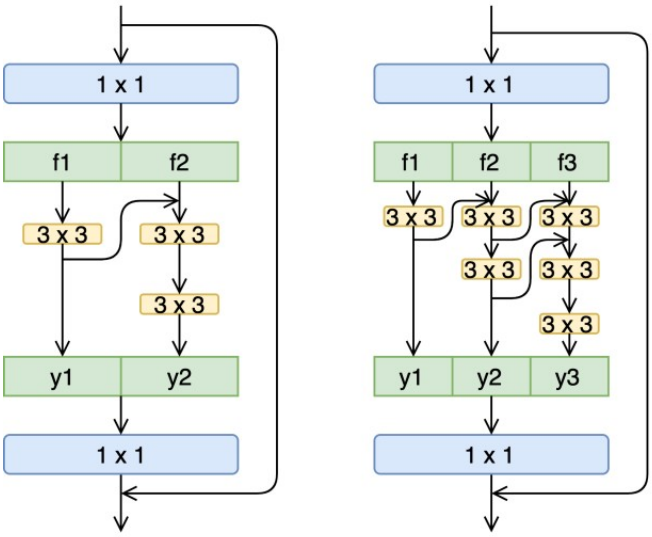

Figure 4: The architecture of different number of branches. On the left is the architecture of two branches, on the right is the architecture of three branches, and so on.

i denotes the serial number of branches and j denotes the number of layers passed through conv3×3. $K_{(i,j)}()$ denotes the j^{th} conv3×3 of i^{th} branch. Define $y_{(i,0)} = f_i$. Then $y_{(i,j)}$ $(j \leq i)$ can be written as equation 1

$$y_{(i,j)} = \begin{cases} K_{(i,j)}(y_{(i,j-1)}) & , j = i \\ K_{(i,j)}(y_{(i,j-1)} + y_{(i-1,j)}) & , j < i \end{cases} \quad (1)$$

2.3. Receptive Field Calculation

In this part, we will calculate the receptive field range of bottleneck of ResNet and RSN. First, the formula for calculating the receptive field of the k^{th} convolution layer is as equation 2

$$l_k = l_{k-1} + [(f_k - 1) * \prod_{i=0}^{n} s_i] \quad (2)$$

l_k denotes the size of the receptive field corresponding to layer k, f_k denotes kernel size of layer k, and s_i denotes the stride of layer i.

Using this formula, we calculate the receptive fields of bottleneck of ResNet and RSN, as shown in Table 1. The results show that the receptive field of feature obtained by ResNet is 3 and the receptive field of feature obtained by RSN is 3-15. Thus, RSN has a better fusion and non-linear representation for different levels of features. In addition, larger receptive field results in greater global features in the network.

Architecture	y1	y2	y3	y4
ResNet	3	3	3	3
RSN	3	5,7	7,9,11	9,11,13,15

Table 1: The receptive fields of bottleneck of ResNet and RSN.

2.4. Multi Stage Network

As shown in the Figure 2 (a), the backbone of multi-stage is 4×RSN-50, the refine-stage is the same with Single-stage. In init-stage, we use extra two blocks to refine the output features: Refine Block (RB) and Feature Fusion Block (FFB).

Refine Block: The components of RB is shown in Figure 2 (c). RB consists of two paths. The upper one passes through two layers of conv3×3. The lower one is an identity connection. The two branches are then summed to output features. The motivation of RB comes from ResNet. This block refines the feature of each stage of RSN-50.

Feature Fusion Block: The components of FFB is shown in Figure 2 (d). The first component of the block is a conv3×3, then the block is divided into three paths. The top path passes through global pool, two conv1×1 and a sigmoid activation. Element-wise product is implemented between the top two paths followed by element-wise sum with the bottom identity path. This block is designed to change the weight of each channel of the feature.

As for the output features of RSN-50, features of different levels are mixed together. These features contain both low-level spatial texture information and high-level global semantic information. Different levels of features contribute differently to the final performance, so it is necessary to change the proportion of each channel to make different levels of features play a better role.

Set the input of FFB to f_{in}, the number of its channel is c, and set the output of FFB to f_{out}. When f_{in} passes through the top path, a weight vector α can be obtained, the number of its channel is also c. $K()$ denotes the conv3×3. Then we can get the equation 3.

$$f_{out} = K(f_{in}) + K(f_{in}) * \alpha \quad (3)$$

3. Experiments

COCO dataset [5] contains over 200000 images and 250000 person instances labeled with 17 joints. We only use the COCO train17 dataset for training including 57K images and 150K person instances. We evaluate our approach on the COCO minival dataset (includes 5K images) and the testing sets includes test-dev set (20K images) and test-challenge set (20K images).

Method	AP	GFLOPs
4×ResNet-50	76.8	19.9
4×ResNet-50 + RB	77.0	22.0
4×ResNet-50 + FFB	77.0	21.8
4×ResNet-50 + RB + FFB	77.2	23.9
4×RSN-50	78.6	26.5
4×RSN-50 + RB	78.8	28.6
4×RSN-50 + FFB	78.8	28.4
4×RSN-50 + RB + FFB	79.0	30.5

Table 2: Results of ablation experiments of RB and FFB on 4×ResNet-50 and 4×RSN-50

Backbone	AP	GFLOPs
RSN-2branch-50	74.4	6.9
RSN-3branch-50	74.5	6.5
RSN-4branch-50	**74.7**	**6.3**
RSN-5branch-50	74.3	6.2
RSN-6branch-50	73.8	5.7

Table 3: Results on different number of branches

backbone	input size	AP	GFLOPs
ResNet-18	256×192	70.7	2.3
RSN-18	**256×192**	**73.2**	**2.5**
ResNet-50	256×192	72.2	4.4
RSN-50	**256×192**	**74.7**	**6.3**
ResNet-hourglass-2stage	256×192	72.3	6.1
RSN-hourglass-2stage	**256×192**	**75.0**	**9.3**
ResNet-101	256×192	73.2	7.5
RSN-101	**256×192**	**75.8**	**11.5**
4×ResNet-50	256×192	77.2	23.9
4×**RSN-50**	**256×192**	**79.0**	**30.5**
4×ResNet-50	384×288	77.9	53.8
4×**RSN-50**	**384×288**	**79.4**	**68.6**

Table 4: Results on ResNet and RSN

We follow the standard evaluation metric. Our approach is evaluated in OKS (Object Keypoints Similarity) based mAP, where OKS defines the similarity between different human poses.

In this part, we validate experiments on ResNet and RSN. Except that the experiments of 4×ResNet-50 and 4×RSN-50 were carried out with the architecture of Multi-stage, the others were implemented with the architecture of Single-stage and RB and FFB are not used in the hourglass network.

Note that, in all the comparative experiments on ResNet and RSN, the only variable is the architecture of bottleneck. The human detector we used in ablation study is MegDet[8] which has 49.4 AP on COCO minival set.

Training Details. Single-stage network is trained on 8 Nvidia GTX 2080Ti GPUs with mini-batch size 32 per GPU. There are 60k iterations per epoch and 400 epochs in total. Adam optimizer is adopted and the linear learning rate gradually decreases from 3e-4 to 0. The weight decay is 1e-5. Multi-stage network is trained on 8 V100 GPUs with mini-batch size 48 per GPU. There are 140k iterations per epoch and 200 epochs in total. Adam optimizer is adopted and the linear learning rate gradually decreases from 5e-4 to 0. The weight decay is 1e-5.

Each image goes through a series of data augmentation operations including cropping, flipping, rotation and scaling. The range of rotation is $-45° \sim +45°$. The range of scaling is 0.7∼1.35. The input size in Section 3.1 and Section 3.2 is 256×192.

Testing Details. We apply a post-Gaussian filter to the estimated heat maps. We average the prediected heat maps of original image with results of corresponding flipped image following the strategy of [6]. Then a quarter offset from the highest response to the second highest one is implemented to obtain the locations of key points. Same as in [2], the pose score is the multiplication of the average score of key points and the bounding box score.

3.1. RB and FFB

The effect of FFB and RB on performance in the Multi-stage architecture. We perform ablation experiments on 4×ResNet-50 and 4×RSN-50. The results are shown in Table 2.

From Table 2, we can see that the use of RB and FFB in the Multi-stage architecture each obtains 0.2 AP gain in a relative high baseline. Both 4×ResNet-50 and 4×RSN-50 in the rest of this paper employ RB and FFB in Multi-stage architecture as default setting.

3.2. The Architecture Choice of the RSN Bottleneck

Ablation experiments are done in order to validate The architecture choice of the RSN bottleneck. The experiment results are shown in Table 3.

Comparing AP and GFLOPs, we can find that when the number of branches is 4, the network has the best performance, so we finally adopt this architecture as RSN.

3.3. Ablation study of RSN

In order to verify the high performance of RSN, we have done extensive experiments on ResNet[4] and RSN. The experimental results are shown in Table 4. The experiment results in Table 4 plotted as polygons are shown in Figure 5. RSN always works better than ResNet with the same GFLOPs. In addition, it is worth noting that RSN has a considerable performance when the computation complexity is

Method	Backbone	Input Size	GFLOPs	AP	AP.5	AP.75	AP(M)	AP(L)	AR
CMUpose[1]	-	-	-	61.8	-	-	-	-	-
G-MRI[7]	ResNet-101	353×257	57.0	64.9	85.5	71.3	62.3	70.0	69.7
G-RMI*[7]	ResNet-101	353×257	57.0	68.5	87.1	75.5	65.8	73.3	73.3
CPN[2]	ResNet-Inception	384×288	-	72.1	91.4	80.0	68.7	77.2	78.5
CPN+ [2]	ResNet-Inception	384×288	-	73.0	91.7	80.9	69.5	78.1	79.0
SimpleBase[11]	ResNet-152	384×288	35.6	73.7	91.9	81.1	70.3	80.0	79.0
HRNet-W32[9]	HRNet-W32	384×288	16.0	74.9	92.5	82.8	71.3	80.9	80.1
HRNet-W48[9]	HRNet-W48	384×288	32.9	75.5	92.5	83.3	71.9	81.5	80.5
Simple Base* + [11]	Res-152	384×288	-	76.5	92.4	84.0	73.0	82.7	81.5
HRNet-W48*[9]	HRNet-W48	384×288	32.9	77.0	92.7	84.5	73.4	**83.1**	82.0
MSPN*[10]	4×ResNet-50	384×288	46.4	77.1	93.8	84.6	73.4	82.3	82.3
Ours(RSN)	**4×RSN-50**	256×192	**30.5**	**78.0**	**94.2**	**86.5**	**75.3**	82.2	**83.4**
Ours(RSN+)	**4×RSN-50**	-	-	**79.2**	**94.4**	**87.1**	**76.1**	**83.8**	**84.1**

Table 5: Results on COCO test-dev sets compared with other methods. "+" means using a ensemble model and "*" means using external data

Method	Backbone	Input Size	AP	AP.5	AP.75	AP(M)	AP(L)	AR
Mask R-CNN*[3]	ResX-101-FPN	-	68.9	89.2	75.2	63.7	76.8	75.4
G-RMI[7]	Res-152	353×257	69.1	85.9	75.2	66.0	74.5	75.1
CPN+[2]	Res-Inception	384×288	72.1	90.5	78.9	67.9	78.1	78.7
Sea Monsters*+	-	-	74.1	90.6	80.4	68.5	82.1	79.5
Simple Base* + [11]	Res-152	384×288	74.5	90.9	80.8	69.5	82.9	80.5
MSPN* + [10]	4×ResNet-50	384×288	76.4	92.9	82.6	71.4	83.2	82.2
Ours(RSN+)	**4×RSN-50**	384×288	**77.1**	**93.3**	**83.6**	**72.2**	**83.6**	**82.6**

Table 6: Results on COCO test-challenge sets compared with other methods. "+" means using a ensemble model and "*" means using external data.

relatively low. This indicates that RSN can maintain both high accuracy and low computation cost. From the experimental results, we can draw a conclusion that RSN is much more effective than ResNet as for the backbone of human pose estimation.

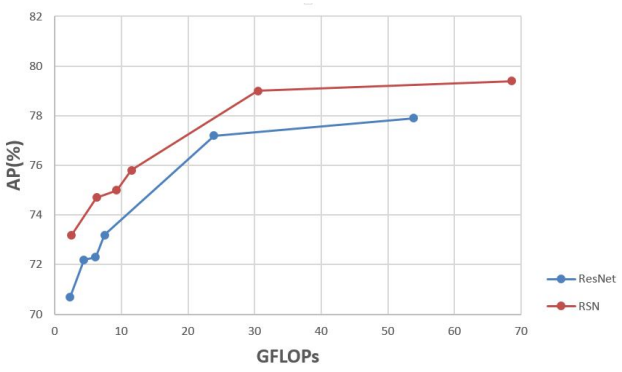

Figure 5: Illustrating how the performances of RSN and Resnet are affected by GFLOPs

3.4. Results on COCO test-dev and test-challenge datasets

Our final model is 4×RSN-50. The results on test-dev and test-challenge are shown in Table 5. and Table 6. Our single model trained only by COCO train17 dataset with input size 256 × 192 achieves 78.0 AP on test-dev datasets and outperforms state-of-art methods by a large margin. The ensemble model obtains 79.2 AP on test-dev dataset and 77.1 AP on test-challenge dataset.

4. Conclusion

In this paper, we propose a novel network Res-Steps-Net (RSN) for human pose estimation. This network is designed for better extracting both local and global features. RSN is more effective in feature representation, thus, it remains high performance with a relatively low computation cost. For example, RSN-18 with 2.5 GFLOPS can achieve 73.2 AP COCO minival. Our single model trained only by coco train17 dataset with 256×192 input size outperforms the state-of-art methods with input size 384×288 with extra

data by a large margin.

References

[1] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 5

[2] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 4, 5

[3] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 5

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1, 4

[5] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: common objects in context. In *ECCV*, pages 740–755, June 2014. 3

[6] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *The European Conference on Computer Vision (ECCV)*, 2016. 4

[7] 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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 5

[8] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. Megdet: A large mini-batch object detector. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 4

[9] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019. 5

[10] Li Wenbo, Wang Zhicheng, Yin Binyi, Peng Qixiang, Du Yuming, Xiao Tianzi, Yu Gang, Lu Hongtao, Wei Yichen, and Sun Jian. Rethinking on multi-stage networks for human pose estimation. In *arxiv 1901.00148*, 2019. 1, 5

[11] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *The European Conference on Computer Vision (ECCV)*, September 2018. 5