Real-Time Human Pose Estimation via Cascaded Neural Networks Embedded with Multi-task Learning

Satoshi Tanabe(✉), Ryosuke Yamanaka, Mitsuru Tomono, Makiko Ito, and Teruo Ishihara

FUJITSU LABORATORIES LTD., Kawasaki, Japan
tanabe.s@jp.fujitsu.com

Abstract. Deep convolutional neural networks (DCNNs) have recently been applied to Human pose estimation (HPE). However, most conventional methods have involved multiple models, and these models have been independently designed and optimized, which has led to suboptimal performance. In addition, these methods based on multiple DCNNs have been computationally expensive and unsuitable for real-time applications. This paper proposes a novel end-to-end framework implemented with cascaded neural networks. Our proposed framework includes three tasks: (1) detecting regions which include parts of the human body, (2) predicting the coordinates of human body joints in the regions, and (3) finding optimum points as coordinates of human body joints. These three tasks are jointly optimized. Our experimental results demonstrated that our framework improved the accuracy and the running time was 2.57 times faster than conventional methods.

Keywords: Human pose estimation · Neural networks · Multi-task learning · End-to-end learning

1 Introduction

Human pose estimation (HPE) from images is a challenging task in computer vision. It predicts the coordinates of human body joints in images. It has many applications, such as those in gesture recognition, clothing parsing, and human tracking. This task is still challenging due to camera viewpoints, complicated backgrounds, occlusion, and running time. Image recognition has recently been improved with deep convolutional neural networks (DCNNs) [1]. Krizhevsky et al. [1] achieved the best recognition rate and attracted a great deal of attention. State-of-the-art performance of HPE has also been achieved with DCNNs [2–10]. However, because the computational cost of DCNNs is very high, the number of calculations should be reduced as much as possible. Furthermore, in order to achieve state-of-the-art performance, end-to-end learned models should be used.

This paper proposes a novel end-to-end framework for HPE implemented with cascaded neural networks. Figure 1 overviews the architecture of our framework, which includes three tasks: (1) detecting region proposals [14] which

M. Felsberg et al. (Eds.): CAIP 2017, Part II, LNCS 10425, pp. 241–252, 2017.
DOI: 10.1007/978-3-319-64698-5_21

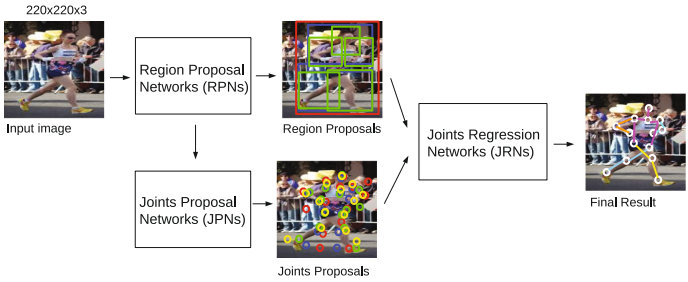
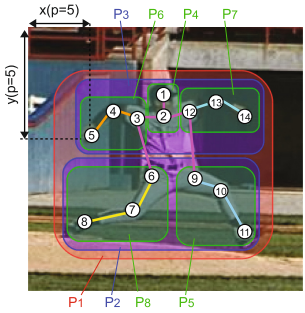


Fig. 1. Overview of the proposed framework

include parts of the human body via region proposal networks (RPNs), (2) predicting the coordinates of human body joints in region proposals via joints proposal networks (JPNs), and (3) finding optimum points as the coordinates of human body joints via joints regression networks (JRNs). These three tasks are jointly optimized. We demonstrated the efficiency of our framework on the Leeds sports pose (LSP) dataset [11]. Our experiments revealed that our framework improved accuracy and reduced the running time compared to conventional methods. The remainder of the paper discusses related works in Sect. 2, and then introduces our framework in Sect. 3. The experimental results are presented in Sect. 4. Section 5 concludes the paper.

2 Related Work

A number of different approaches using DCNNs have been proposed for HPE. DeepPose [2] proposes a cascade of DCNNs-based pose predictors. Such a cascade allows for increased precision of joint localization, which achieves very high levels of accuracy. However, this model includes multiple DCNNs that are computationally expensive, and each pose predictor is independently designed and optimized. Chen et al. [10] use DCNNs to learn conditional probability for the presence of parts. The conditional probability is also called a heat map. The human pose is predicted using graphical models with prior knowledge such as geometric relationships among body parts. However, the DCNNs and the graphical models are independently optimized. Yang et al. [12] propose a model, which combines the DCNNs for generating a heat map with the graphical models, and these models are jointly optimized. This approach also achieves high levels of accuracy. However, generating a heat map requires the use of many DCNNs, which leads to large computational costs. Wang et al. [13] propose a model which handles two tasks: (1) it generates a heat map from depth images via a fully convolutional network (FCN) [15] and (2) it seeks an optimal configuration of body parts via an inference built-in MatchNet [16]. However, MatchNet imposes large computational costs due to the use of chains in multiple convolutional layers [17].



Set of Joint Numbers	Joint Number p
P1	1,2,3,4,5,6,7,8,9,10,11,12,13,14
P2	6,7,8,9,10,11
P3	1,2,3,4,5,12,13,14
P4	1,2
P5	9,10,11
P6	3,4,5
P7	12,13,14
P8	6,7,8

Joint Number p	Joint Name
1	Head
2	Neck
3	Right Shoulder
4	Right Elbow
5	Right Wrist
6	Right Hip
7	Right Knee
8	Right Ankle
9	Left Hip
10	Left Knee
11	Left Ankle
12	Left Shoulder
13	Left Elbow
14	Left Wrist

Fig. 2. Representing a human body as a graph

3 Our Framework

This section presents our framework, which consists of three stages. The first stage is region proposal networks (RPNs), the second stage is joints proposal networks (JPNs), and the third stage is joints regression networks (JRNs). These are described in Subsects. 3.1, 3.2, and 3.3, respectively. In Subsect. 3.4, the multi-task learning procedure of our model is described.

3.1 Region Proposal Networks (RPNs)

Our model predicts region proposals $\mathbf{R}$ via RPNs in the first stage. Region proposals $\mathbf{R}$ denote a vector that consists of bounding boxes, which include multiple parts of the human body. Region proposals $\mathbf{R}$ are obtained as follows:

$$\mathbf{R}(\mathbf{I}) = \left(\mathbf{B}_1,\ \mathbf{B}_2,\ \ldots,\ \mathbf{B}_K\right) \tag{1}$$

$$\mathbf{B}_k = \left(b_1^k,\ b_2^k,\ b_3^k,\ b_4^k\right) = \left(\min_{p\in P_k} x(p),\ \min_{p\in P_k} y(p),\ \max_{p\in P_k} x(p),\ \max_{p\in P_k} y(p)\right), \tag{2}$$

where $\mathbf{I}$ denotes an input image, p denotes a joint number, and P_k denotes a set of joint numbers. Here, $1 \leq k \leq K$, K denotes the number of bounding boxes which is set to eight, and $x(p)$ and $y(p)$ denote the coordinates of human body joints with joint number p in the input image. Figure 2 outlines the relationship between a joint number p and P_k. Figure 3 shows an example of the architecture for RPNs, where feature map 1, 2, and 3 are used in joints proposal networks (JPNs) as input.

We adopted two architectures which have been widely used for image classification. The first was VGG-16 [18], and the second was GoogLeNet [23], because these architectures have provided outstanding results for image classification. For example, Faster-RCNN [14] predicts region proposals via VGG-16 [18] for object detection. The performance impact of each RPNs architecture is described in Sect. 4.

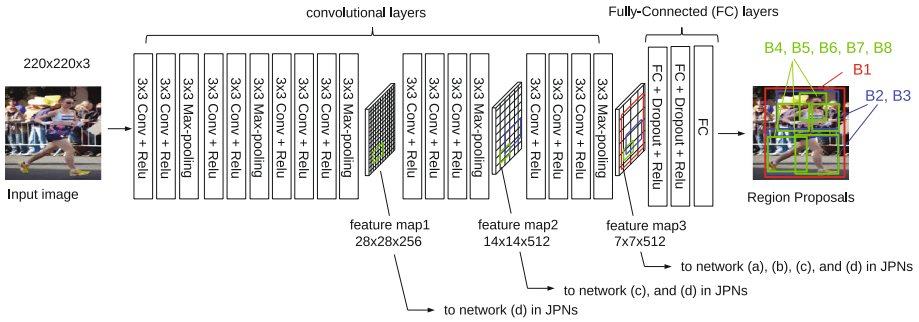


Fig. 3. The architecture of RPNs by using VGG-16 [18]

3.2 Joints Proposal Networks (JPNs)

Our model takes feature maps and region proposals $\mathbf{R}$ as input in the second stage, and it predicts joints proposals $\mathbf{J}$ via JRNs, where joints proposals $\mathbf{J}$ are defined as the coordinates of human body joints in region proposals $\mathbf{R}$. Joints proposals $\mathbf{J}$ are obtained as follows:

$$\mathbf{J}(\mathbf{I}) = \left(\mathbf{J}_0,\ \mathbf{J}_1,\ \mathbf{J}_2 \ldots,\ \mathbf{J}_K\right) \tag{3}$$

$$\mathbf{J}_0 = \left(x(1),\ y(1),\ \ldots,\ x(L),\ y(L)\right) \tag{4}$$

$$\mathbf{J}_k = \left(j_1^k(p_1),\ j_2^k(p_1), \ldots,\ j_1^k(p_{s(k)}),\ j_2^k(p_{s(k)})\right) \tag{5}$$

$$j_1^k(p) = (x(p) - b_1^k)/(b_3^k - b_1^k) \tag{6}$$

$$j_2^k(p) = (y(p) - b_2^k)/(b_4^k - b_2^k), \tag{7}$$

where $1 \leq k \leq K$, $p_1, \ldots, p_{s(k)} \in P_k$, b_1^k, b_2^k, b_3^k, and b_4^k denote vertices of bounding box $\mathbf{B}_k$ defined in Eq. (2), and L denotes the number of human body parts, which is set to 14 as described in Fig. 2.

Figure 4 shows the architectures for JPNs, which consists of four types of networks. Network (a) takes a feature map from the middle layer in RPNs as input, and predicts $\mathbf{J}_0$. Networks (b), (c), and (d) take feature maps from multiple middle layers in RPNs and region proposals $\mathbf{R}$ as input, and predict $\mathbf{J}_1$, $\mathbf{J}_2$, ..., and $\mathbf{J}_K$.

The purpose of the region-of-interest (RoI) pooling layers [14] is to extract the area indicated by region proposals $\mathbf{R}$ from feature maps and to produce a fixed-length feature vector because full-connected (FC) layers [1] require it as input. DeepPose [2] extracts the area from an input image. However, such an approach requires many calculations using convolutional layers for the feature extraction, which leads to large computational costs. In our model, the calculation is performed only once because the area is extracted from feature maps. This approach is computationally efficient and suitable for real-time applications. Moreover, our model takes feature maps from multiple middle layers as input to increase the resolution of feature maps. For example, feature maps with

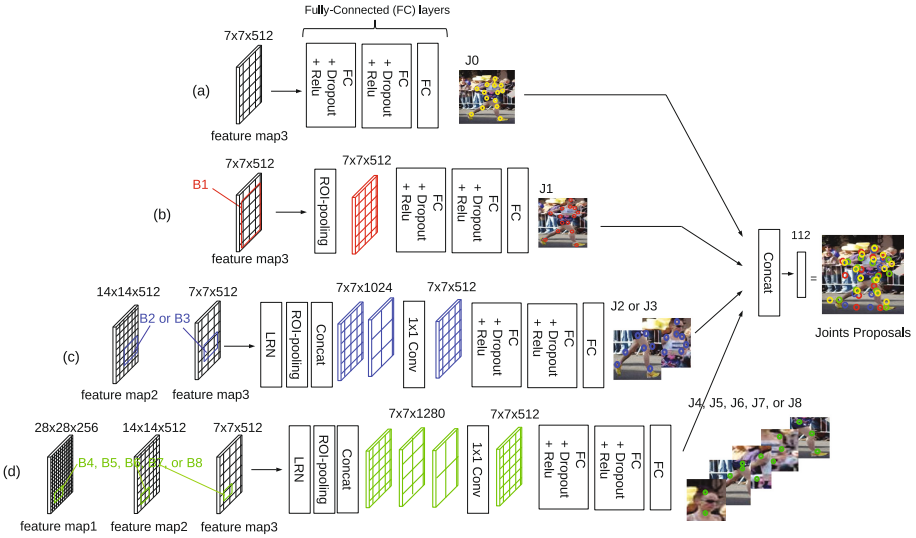


Fig. 4. The architecture of JPNs.

high resolution are required to calculate $\mathbf{J}_4$, $\mathbf{J}_5$, $\mathbf{J}_6$, $\mathbf{J}_7$, and $\mathbf{J}_8$, because $\mathbf{B}_4$, $\mathbf{B}_5$, $\mathbf{B}_6$, $\mathbf{B}_7$, and $\mathbf{B}_8$ have a small area (see Fig. 3). On the other hand, feature maps with high resolution are not required for calculating $\mathbf{J}_1$ because $\mathbf{B}_1$ has a large area. Our model changes the number of feature maps depending on the size of the bounding box. As shown in Fig. 4, Network (d) takes feature map 3, 2, and 1 as input. Network (c) takes feature map 3 and 2 as input. Network (b) and (a) only take feature map 3 as input.

The purpose of the 1×1 convolutional layers is to reduce the channel dimensions of feature maps. They enable the training time to shorten by reducing the dimensions. The Local Response Normalization (LRN) layers are used to align the amplitude of feature maps in each convolutional layer.

3.3 Joints Regression Networks (JRNs)

Our model takes region proposals $\mathbf{R}$ and joints proposals $\mathbf{J}$ as input in the third stage, and it predicts the coordinates of human body joints via JRNs. Figure 5 shows the architectures for JRNs, whose purpose is finding the optimum points as human body joints.

These layers can be replaced with a linear function under ideal conditions, in the case that region proposals $\mathbf{R}$ and joints proposals $\mathbf{J}$ are correct values. The coordinates of human body joints are obtained as follows:

$$x(p) = (b_3^k - b_1^k)\, j_1^k(p) + b_1^k \tag{8}$$

$$y(p) = (b_4^k - b_2^k)\, j_2^k(p) + b_2^k, \tag{9}$$

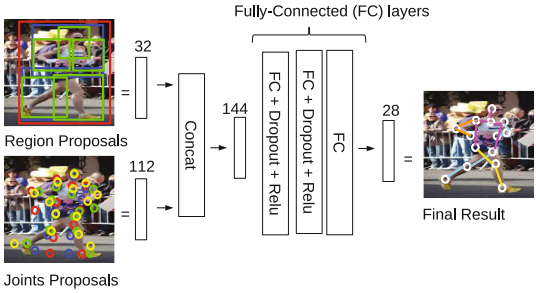


Fig. 5. The architecture of JRNs

where p denotes a joint number, b_1^k, b_2^k, b_3^k, and b_4^k denote vertices of bounding box $\mathbf{B}_k$ defined in Eq. (2), and $j_1^k(p)$ and $j_2^k(p)$ denote elements of joints proposals $\mathbf{J}_k$ in Eq. (5). However, as b_1^k, b_2^k, b_3^k, b_4^k, $j_1^k(p)$, and $j_2^k(p)$ fluctuate randomly, Eqs. (8) and (9) do not work well. We used fully-connected (FC) layers [1] because they are a nonlinear function that leads to universal approximation property [20].

3.4 Multi-task Learning

We define the loss function of the entire network as:

$$l(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) = l_1(\mathbf{w}_1) + l_2(\mathbf{w}_1, \mathbf{w}_2) + l_3(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3), \tag{10}$$

where $\mathbf{w}_1$, $\mathbf{w}_2$, and $\mathbf{w}_3$ denote weight parameters in RPNs, JPNs, and JRNs. $l_1(\mathbf{w}_1)$, $l_2(\mathbf{w}_1, \mathbf{w}_2)$, and $l_3(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)$ correspond to the mean-squared-error (MSE) [21] for RPNs, JPNs, and JRNs.

The loss function, $l(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)$, is minimized with respect to $\mathbf{w}_1$, $\mathbf{w}_2$, and $\mathbf{w}_3$. We employ an Adaptive Moment Estimation (Adam) [27] to optimize $\mathbf{w}_1$, $\mathbf{w}_2$, and $\mathbf{w}_3$. The entire algorithm for multi-task learning is summarized in Algorithm 1. First, $\mathbf{w}_1$, $\mathbf{w}_2$, and $\mathbf{w}_3$ are initialized randomly in step 1. Then, RPNs, JPNs and JRNs are independently optimized in step 2, 3, and 4. Finally, the entire network is trained in step 5. Step 2, 3, and 4 are pre-training techniques [22] to shorten training time in step 5. The end-to-end learning is performed in step 5. The details about values of the several parameters are described in Sect. 4.

4 Experiments

4.1 Experimental Settings

Datasets. We evaluated the proposed methods on well-known public pose estimation benchmarks: Leeds sports poses (LSP) dataset [11] and Leeds sports pose extended training (LSPET) dataset [12]. The LSP dataset consists of 1,000 training and 1,000 testing images, and the LSPET dataset consists of 10,000 training images. However, a lot of data are required for training DCNNs. We

Algorithm 1. Optimize $\mathbf{w}_1$, $\mathbf{w}_2$, and $\mathbf{w}_3$

Input: Training Samples $\left(\mathbf{I}_1, \mathbf{R}(\mathbf{I}_1), \mathbf{J}(\mathbf{I}_1)\right) \ldots \left(\mathbf{I}_N, \mathbf{R}(\mathbf{I}_N), \mathbf{J}(\mathbf{I}_N)\right)$
Output: $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$
1: Initialize $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$ randomly.
2: $\mathbf{w}_1 \leftarrow \arg\min_{\mathbf{w}_1} l_1(\mathbf{w}_1)$
3: $\mathbf{w}_2 \leftarrow \arg\min_{\mathbf{w}_2} l_2(\mathbf{w}_1, \mathbf{w}_2)$
4: $\mathbf{w}_3 \leftarrow \arg\min_{\mathbf{w}_3} l_3(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)$
5: $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3 \leftarrow \arg\min_{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3} l_1(\mathbf{w}_1) + l_2(\mathbf{w}_1, \mathbf{w}_2) + l_3(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)$

also used our 3D-CAD models to increase the amount of data. The 3D-CAD models consist of 11,000 training images that are automatically generated by using open source 3D-CAD tools [28,29], and the human motions are created by using the motion capture database [30]. We combined these datasets with the LSP and the LSPET datasets. As a result, the combination contained 22,000 training images. Peng et al. [19] augment the training images with synthetic images generated from 3D-CAD models for image classification. We used this approach.

We augmented the training images to reduce overfitting by horizontally mirroring the images, rotating them through 360 degrees for every 9 degrees, cropping them randomly, and injecting white noise into them. The final amount of training samples was 5,000,000.

Metrics. We used widely accepted evaluation metrics called the percent of detected joints (PDJ) [12], which calculates the detection rate of human body joints, where a joint is considered as being detected if the distance between the predicted joint and the correct joint is less than a fraction of the torso diameter. The torso diameter is defined as the distance between the left shoulder and the right hip. We also computed an Area-Under-the-Curve (AUC) to compare our work with other approaches (see Figs. 6, 7, and 8).

Person-Centric/Observer-Centric. In person-centric [24] annotations, right/left body parts are marked according to the viewpoint of the person in an image. For example, the right wrist of a person facing the camera is left in the image. However, if the person faces away from the camera, it is right in the image. On the other hand, in observer-centric [24] annotations, right/left body parts are marked regardless of the viewpoint. Person-centric annotations are more difficult than observer-centric annotations because it is necessary to recognize the viewpoint. The information of the viewpoint is important for action recognition. Therefore, we used person-centric annotations.

DCNN Architectures. We investigated two DCNN architectures in RPNs. The first was VGG-16 [18] that consists of 16 convolutional layers and FC layers.

Table 1. Running time on an Intel Xeon CPU at 3.50 GHz and a NVIDIA Tesla K40 GPU. Note that the Heat Map [12] only indicates the running time for generating a heat map, and it does not include the processing time of other tasks.

Method	Running time [s]
DeepPose [2]	0.18
Heat map [12]	0.50
Our work (VGG-16 [18])	**0.094**
Our work (GoogLeNet [23])	**0.070**

The second was GoogLeNet [23] that consists of three convolutional layers, nine inception layers, and FC layers.

Implementation Details. All of our experiments were carried out on an Intel Xeon CPU at 3.50 GHz and a NVIDIA Tesla K40 GPU. Our model was implemented on the Chainer library [31]. In order to optimize our model, we used pre-training models in Model Zoo [32] for fine-tuning [22]. The learning rate and the batch size were set to 0.0001 and 24, respectively, for training RPNs, JPNs and JRNs. On the other hand, the learning rate and the batch size were set to 0.00001 and 20 for the end-to-end learning. The total training time was about 2 weeks.

4.2 Experimental Results

Table 1 lists the running time results. Our model was 2.57 times faster than DeepPose [2]. As described in Sect. 3.2, the conventional methods use a lot of DCNNs that are computationally expensive. However, our model does not use them, so the running time of our model was fast.

Figure 6 shows the PDJ results on the LSP dataset. We used person-centric annotations for fair comparison with related work [2,25]. Our model achieved the best performance compared with the conventional methods. Our results were particularly better in the low precision domain. The AUC of our model was 7.34% – 29.66% higher than that of DeepPose [2].

We analyzed how different RPNs architectures affected performance. Figure 7 shows the PDJ results with different RPNs architectures. Our best performance was achieved by using the architecture of VGG-16 [18] in RPNs. The AUC of VGG-16 [18] was 8.65% – 19.62% higher than that of GoogLeNet [23].

Figure 8 compares JPNs with JRNs. The PDJ of JRNs was higher than that of JPNs, especially for ankle. The AUC was improved from 0.66% to 12.56%. Here, we can observe that JRNs have an effect on improving accuracy. Figure 9 shows some pose estimation results.

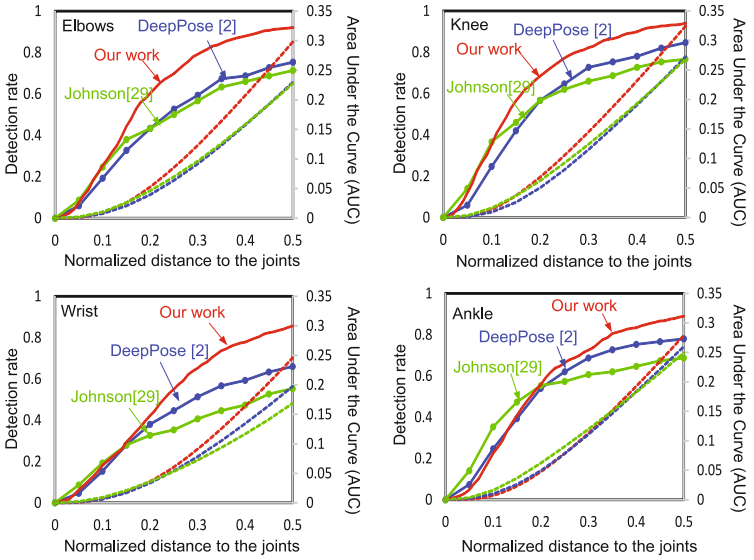


Fig. 6. PDJ comparison of our work and other approaches on the LSP dataset. The solid lines and the dashed lines represent PDJ and AUC on the LSP dataset, respectively. All results are from author's papers, and these are the person-centric results. The architecture of RPNs was VGG-16 [18]. Note that we adopted only person-centric results as related work. For example, Chen and Yuille [10] and Yang et al. [12] used observer-centric annotations, therefore these were excluded from comparisons.

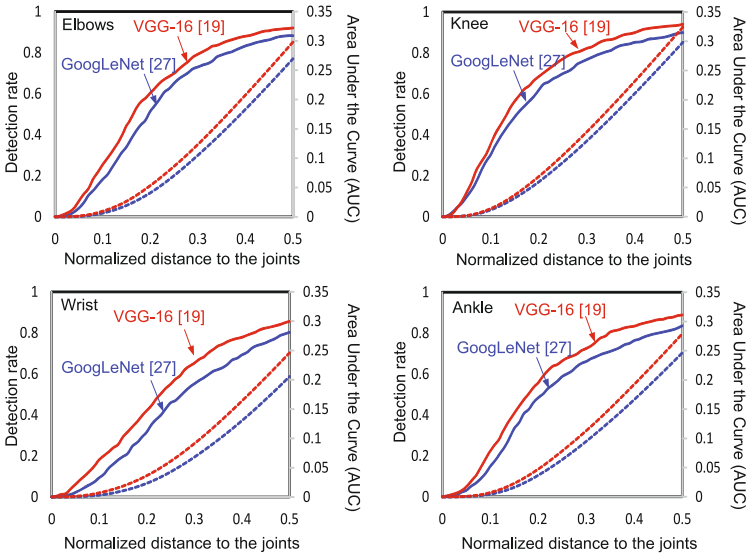


Fig. 7. Influences of different RPNs architectures are plotted. The solid lines and the dashed lines represent PDJ and AUC on the LSP dataset, respectively.

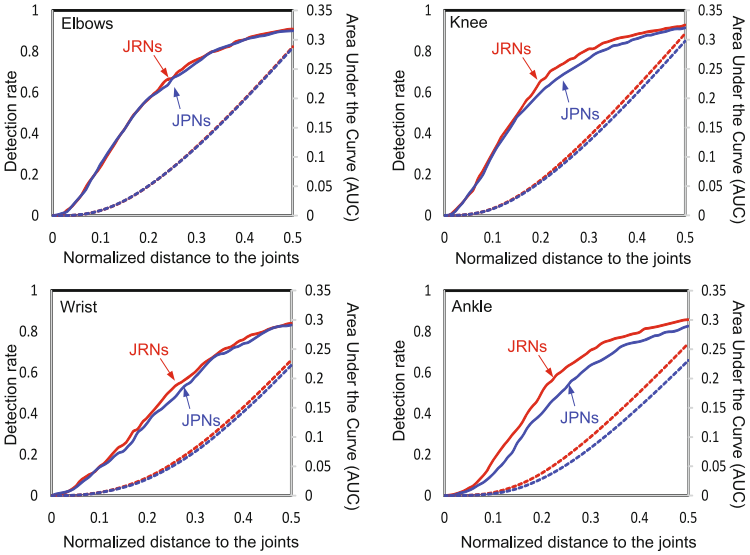


Fig. 8. Figures compare JPNs with JRNs. Results for JPNs were calculated by using output of network (a) in Fig. 4. The architecture of RPNs was VGG-16 [18]. The solid lines and the dashed lines represent PDJ and AUC on the LSP dataset, respectively.

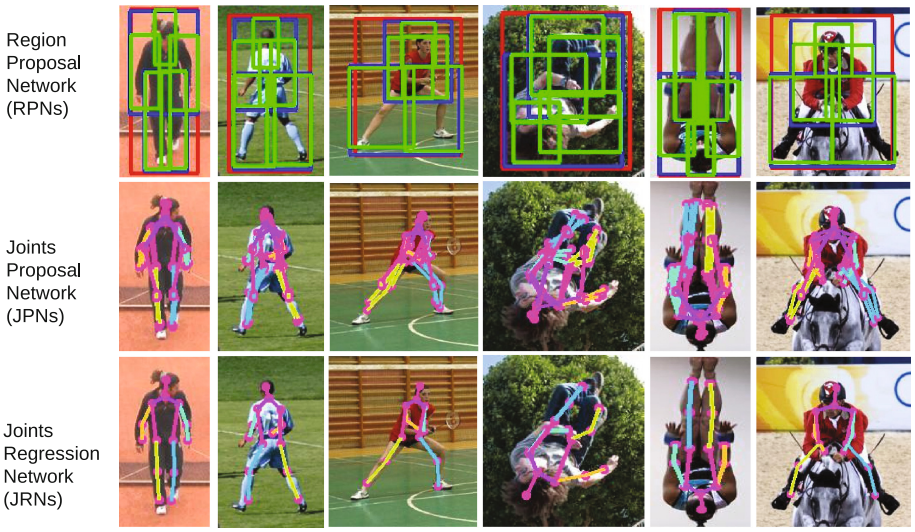


Fig. 9. HPE results we obtained. The first row shows the outputs of RPNs. The second row shows the outputs of JPNs. The third row shows the outputs of JRNs.

5 Conclusion and Future Work

We proposed a novel end-to-end framework for HPE implemented with cascaded neural networks. We demonstrated the efficiency of our framework on the LSP dataset [11]. As a result, our model achieved accuracy that was higher than that of conventional models, and the running time was 2.57 times faster than conventional methods.

As a future work, we plan to evaluate our method on other datasets, for example, Frames Labeled In Cinema dataset (FLIC) [33], Kinect2 Human Gesture Dataset (K2HGD) [13], and MPII Human Pose Dataset [34]. Furthermore, we apply other methods of speeding up HPE to our model, such as binarized weights [26] or low rank approximation [17].

Acknowledgments. The authors would like to thank Professor Hironobu Fujiyoshi at Chubu University for his forthright comments and valuable suggestions.

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: NIPS (2012)
2. Toshev, A., Szegedy, C.: DeepPose: human pose estimation via deep neural networks. In: CVPR (2014)
3. Tompson, J., Jain, A., LeCun, Y., Bregler, C.: Joint training of a convolutional network and a graphical model for human pose estimation. In: NIPS (2014)
4. Tompson, J., Goroshin, R., Jain, A., LeCun, Y., Bregler, C.: Efficient object localization using convolutional networks. In: CVPR (2015)
5. Jain, A., Tompson, J., LeCun, Y., Bregler, C.: MoDeep: a deep learning framework using motion features for human pose estimation. In: Cremers, D., Reid, I., Saito, H., Yang, M.-H. (eds.) ACCV 2014. LNCS, vol. 9004, pp. 302–315. Springer, Cham (2015). doi:10.1007/978-3-319-16808-1_21
6. Jain, A., Tompson, J., Andriluka, M., Taylor, G.W., Bregler, C.: Learning human pose estimation features with convolutional networks. In: ICLR (2014)
7. Fan, X., Zheng, K., Lin, Y., Wang, S.: Combining local appearance and holistic view: dual-source deep neural networks for human pose estimation. In: CVPR (2015)
8. Chu, X., Ouyang, W., Li, H., Wang, X.: Structured feature learning for pose estimation. In: CVPR (2016)
9. Chen, X., Yuille, A.: Parsing occluded people by flexible compositions. In: CVPR (2015)
10. Chen, X., Yuille, A.L.: Articulated pose estimation by a graphical model with image dependent pairwise relations. In: NIPS (2014)
11. Johnson, S., Everingham, M.: Clustered pose and nonlinear appearance models for human pose estimation. In: BMVC (2010)
12. Yang, W., et al.: End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation. In: CVPR (2016)
13. Wang, K., et al.: Human pose estimation from depth images via inference embedded multi-task learning. In: Multimedia Conference (2016)

14. Ren, S., et al.: Faster R-CNN: towards real-time object detection with region proposal networks. In: NIPS (2016)
15. Long, J., et al.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
16. Han, X., et al.: MatchNet: unifying feature and metric learning for patch-based matching. In: CVPR (2015)
17. Jaderberg, M., et al.: Speeding up convolutional neural networks with low rank expansions. BMVA Press (2014)
18. Simonyan, K., et al.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
19. Peng, X., et al.: Learning deep object detectors from 3D models. In: CVPR (2015)
20. Sonoda, S., et al.: Neural network with unbounded activation functions is universal approximator. In: Applied and Computational Harmonic Analysis (2015)
21. Bishop, C.M.: Pattern Recognition and Machine Learning. Information Science and Statistics, 2nd edn. Springer, New York (2006)
22. Reyes, A., Caicedo, J., Camargo, J.: Fine-tuning Deep Convolutional Networks for Plant Recognition. In: Proceedings of CEUR Workshop, CLEF (Working Notes), vol. 1391 (2015). CEUR-WS.org
23. Szegedy, C., et al.: Going deeper with convolutions. In: CVPR (2014)
24. Eichner, M., Ferrari, V.: Appearance sharing for collective human pose estimation. In: Lee, K.M., Matsushita, Y., Rehg, J.M., Hu, Z. (eds.) ACCV 2012. LNCS, vol. 7724, pp. 138–151. Springer, Heidelberg (2013). doi:10.1007/978-3-642-37331-2_11
25. Johnson, S., et al.: Learning effective human pose estimation from inaccurate annotation. In: CVPR (2011)
26. Courbariaux, M., et al.: BinaryConnect: training deep neural networks with binary weights during propagations. In: NIPS (2015)
27. Kingma, D.P., et al.: Adam: a method for stochastic optimization. In: ICLR (2015)
28. Website of MakeHuman. http://www.makehuman.org/. Accessed 18 Apr 2017
29. Website of Blender. https://www.blender.org. Accessed 18 Apr 2017
30. Website of motion capture database. http://mocap.cs.cmu.edu/. Accessed 18 Apr 2017
31. Website of Chainer. http://chainer.org/. Accessed 18 Apr 2017
32. Website of Model Zoo. https://github.com/BVLC/caffe/wiki/Model-Zoo. Accessed 18 Apr 2017
33. Benjamin, S., et al.: MODEC: multimodal decomposable models for human pose estimation. In: CVPR (2013)
34. Andriluka, M., et al.: 2D human pose estimation: new benchmark and state of the art analysis. In: CVPR (2014)