J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr

Empirical Comparison of Deep Learning Networks on Backbone Method of Human Pose Estimation☆

Beanbonyka Rim[1] Junseob Kim[1] Yoo-Joo Choi[2] Min Hong[3*]

ABSTRACT

Accurate estimation of human pose relies on backbone method in which its role is to extract feature map. Up to dated, the method of backbone feature extraction is conducted by the plain convolutional neural networks named by CNN and the residual neural networks named by Resnet, both of which have various architectures and performances. The CNN family network such as VGG which is well-known as a multiple stacked hidden layers architecture of deep learning methods, is base and simple while Resnet which is a bottleneck layers architecture yields fewer parameters and outperform. They have achieved inspired results as a backbone network in human pose estimation. However, they were used then followed by different pose estimation networks named by pose parsing module. Therefore, in this paper, we present a comparison between the plain CNN family network (VGG) and bottleneck network (Resnet) as a backbone method in the same pose parsing module. We investigate their performances such as number of parameters, loss score, precision and recall. We experiment them in the bottom-up method of human pose estimation system by adapted the pose parsing module of openpose. Our experimental results show that the backbone method using VGG network outperforms the Resent network with fewer parameter, lower loss score and higher accuracy of precision and recall.

☞ keyword : Deep learning, human pose estimation, CNN, VGG, Resnet

1. Introduction

Accurate estimation of human pose brings benefits for various applications such as augmented reality [1] in the field of computer vision. Those benefits are from an understanding a person in image to a recognizing an activity in video. In this context, the human pose estimation system refers to, by given a single RGB image, the precise 2D coordinates of a person's posture known as keypoints or body-joints and limb association known as joint-pairs or skeleton are wished to automatically be localized and generated. The robust pose estimation system must be able to accurately and quickly predict keypoints and joint-pairs even though under occlusion, illumination variation and deformation diversity. To tackle such challenges, significant image features and sophisticated prediction methods have been proposed over recent years.

With the successful promising rates of Deep Learning methods [2-14] outperform over the traditional graphical based methods [15-19], the human pose estimation has been conducted in two methods: bottom-up and top-down. The bottom-up method is ideally to predict keypoints and form joint-pairs in which they will be assigned to individual person instance later. However, it leads to wrongly pair the keypoints when two or more people instances are overlapped due to it could not define the boundary of each person instance. On the other hand, the top-down method is ideally to detect people instance before hand. Then, the bounding box of each person instance will be estimated the keypoints and joint-pairs by a single pose estimator later. It achieves

1 Department of Computer Science, Soonchunhyang University, Asan, 31538, Korea.
2 Department of Newmedia, Seoul Media Institute of Technology, Seoul, 07590, Korea.
3 Department of Computer Software Engineering, Soonchunhyang University, Asan, 31538, Korea.
* Corresponding author: mhong@sch.ac.kr.
[Received 9 June 2020, Reviewed 15 June 2020, Accepted 26 July 2020]
☆ This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2019R1F1A1062752), by the Ministry of Education (NRF-2017R1D1A1B03035718) and by the Soonchunhyang University Research Fund.
☆ A preliminary version of this paper was presented at ICONI 2019.

higher accuracy but slower in test time comparing to the bottom-up method [11].

Both methods are firstly conducted by the backbone network. The backbone network serves as a feature extractor. Most of state-of-the-art human pose estimation systems have exploited the plain convolutional neural network named by CNN and residual neural network named by Resnet as the backbone network. The CNN family network such as VGG [20] is multi-stacked layers and uses max-pooling to lower the convolution filters in order to extract the features. The Resnet network [21] is a bottleneck network in which layers are non-linear stacked. Stated by literature, the deeper of layers, the higher accuracy will be gained. As well, the numerous amount of parameters and computational complexity will intuitively be increased. To solve this, Resnet introduced the identity shortcut connections which they are concatenated by feedforward convolution mapping and identity mapping. Stated in its paper, Resnet of 50 layers performs error rate of 3.66% and 3.75% lower than VGG on ImageNet [22] validation and testing, respectively. Up to dated, both of backbone networks (VGG and Resnet) have achieved inspired results to extract features before fed into pose estimation network. However, they are exploited by different pose estimation networks named by pose parsing module.

Therefore, in this paper, we present a comparison between VGG and Resnet as the backbone network, then fed into the same pose parsing module. We experiment them in the bottom-up method of human pose estimation system. We adapt openpose [7] as the pose parsing module. We investigate their performance on number of parameters, loss score, precision and recall. Our main contribution is to compare and find the best deep learning model for backbone network of 2D human pose estimation.

2. Related work

In this section, we review the works who used the plain CNN network such as ConNets, AlexNet and VGG and residual network such as Resnet as the backbone network for feature extraction in human pose estimation.

Toshev et al. [2] initially used AlexNet [23] to extract the features in which consists of 5 ConNets and 2 fully-connected layers. Then, spatial pose coordinates were directly detected by cascade pose regressors. Wei et al. [3] learned image features which composed of 5 ConNets and other two 1x1 ConNets, then directly operated on belief map to localize partial joint coordinates. Newell et al. [4] introduced hourglass model which composed of ConNets and connected by fully-connected layers for end-to-end learning the keypoints coordinates. Chu et al. [5] adopted an 8-stack hourglass network in which each stack composed of ConNets to extract the features. Belagiannis et al. [6] used ConNets with Recurrent network model to extract the features and learned to regress the location of the keypoints. Cao et al. [7] exploited the first 12 layers of VGG-19 to generate a set of features. Then, the stacked ConNets were used to localize the keypoints and joint-pairs.

Papandreou et al. [8] used Resnet with 101 layers to produce feature heatmaps and combined with offset prediction composed of ConNets to localize the keypoints. Xiao et al. [9] used Resnet as the backbone network to extract the image features and added a few deconvolutional layers over the last convolution layers of Resnet to generate predicted keypoints locations. Chen et al. [10] proposed cascaded pyramid network to estimate human pose by exploited Resnet as the backbone network and combining with RefineNet to efficiently localize the keypoints. Kocabas et al. [11] proposed pose residual network to produce accurate keypoints location by used Resnet as the backbone network to extract the features for keypoints localization. Hu et al. [12] proposed learning feature integration to localize the keypoints by exploited Resnet as the backbone network and integrated with pose parsing model in which composed of up-sampling and deconvolutional layers.

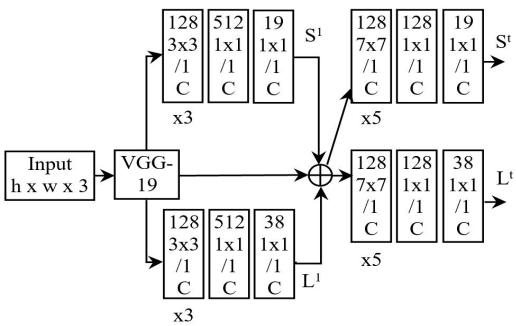
3. Methodology

In order to compare these commonly used backbone networks (the plain convolutioinal layers and residual network layers), we conduct an experiment of applying them in the same pose parsing module. We choose VGG-19 and Resnet-50 representing for the plain convolutional layers and the residual network layers, respectively. We exploit the

openpose [7] as the pose parsing module to localize the keypoints and generate the joint-pairs.

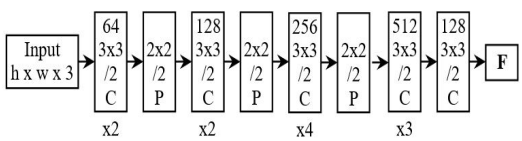
3.1 VGG-19 as the backbone network

The work of openpose [7] was originally used the last 12 layers of VGG-19 for feature extraction. Then, there are two networks for pose paring module. One is confident map for localizing the keypoints. Second is part-assoicated map for paring the keypoints (joint-pairs). Figure 1 illustrates the original network architecture of openpose.



(Figure 1) An architecture of openpose [7]

The network feeds the input image with size of height x width with three color channels (i.e. R, G, B) into the backbone network (the last 12 layers of VGG-19) to produce a set of feature maps F, as depicted in Figure 2.



(Figure 2) The first 12 layers of VGG-19 [18]

The first two convolutional layers (C) filter the input image with 64 kernels of size 3x3 with a stride of 1x1, followed by max-pooling (P) 2x2 with a stride of 2x2. The second two convolutional layers filter with 128 kernels of size 3x3 with a stride of 1x1, followed by max-pooling 2x2 with a stride of 2x2. The third four convolutional layers filter with 256 kernels with size of 3x3 with a stride of 1x1, followed by max-pooling 2x2 with a stride of 2x2. Then, three convolutional layers filter with 512 kernels with size of 3x3 with a stride of 1x1, and last convolutional layer filters with 128 kernels with size of 3x3 with a stride of 1x1.

Then, the feature maps **F** is used as input for pose paring module to generate confident maps **S** and part-associated maps (part affinity fields) **L**, as depicted in Figure 1. There are two steps in the pose paring module: ($\mathbf{S^1}$, $\mathbf{L^1}$) and ($\mathbf{S^t}$, $\mathbf{L^t}$). The first step is the network to produce ($\mathbf{S^1}$, $\mathbf{L^1}$) named as stage 1, in which the set of confident maps $\mathbf{S^1}$ and part-associated maps $\mathbf{L^1}$ as shown in equation 1 and 2.

$$S^1 = \rho^1(F) \tag{1}$$

$$L^1 = \phi^1(F) \tag{2}$$

where ρ^1 is the ConNets composed of three convolutional layers with 128 filter kernels of size 3x3 with a stride of 1x1, one convolutional layer with 512 filter kernels of size 1x1 with a stride of 1x1, and one covolutional layer with 19 (number of keypoints) filter kernels of size 1x1 with a stride of 1x1 for $\mathbf{S^1}$. While ϕ^1 is the same ConNets as ρ^1 except the last covolutional layer in which composed of a convolutional layer with 38 (number of keypoints x 2) filter kernels of size 1x1 with a stride of 1x1 for $\mathbf{L^1}$.

The second step is the network to produce ($\mathbf{S^t}$, $\mathbf{L^t}$) named as stage **t**, in which the set of confident maps $\mathbf{S^t}$ and part-associated maps $\mathbf{L^t}$ as shown in equation 3 and 4.

$$S^t = \rho^t(F, S^{t-1}, L^{t-1}), t \geq 2 \tag{3}$$

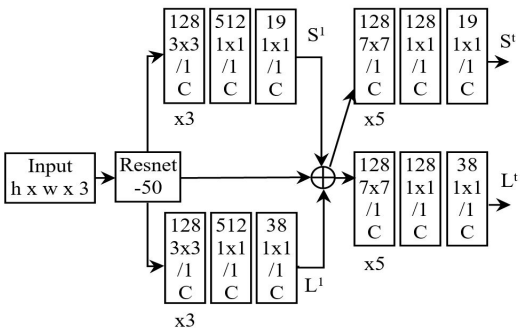
$$L^t = \phi^t(F, S^{t-1}, L^{t-1}), t \geq 2 \tag{4}$$

where ρ^t is the ConNets composed of five convolutional layers with 128 filter kernels of size 7x7 with a stride of 1x1, and one covolutional layer with 19 (number of keypoints) filter kernels of size 1x1 with a stride of 1x1 for $\mathbf{S^t}$. While ϕ^t is the same ConNets as ρ^t except the last convolutional layer in which composed of a covolutional layer with 38 (number of keypoints x 2) filter kernels of size 1x1 with a stride of 1x1 for $\mathbf{L^t}$. **t** is the number of stages in which it is greater than 2. In our experiment, we used **t** up to 7 stages.

3.2 Resnet-50 as the backbone network

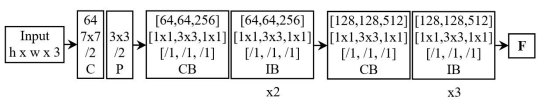
We modify the architecture of openpose [7] by replacing the backbone network by Resnet-50. We keep exploiting the

pose paring module of openpose (confident map S for keypoint localization and pari-associated map L for joint-pairs). We use only one ConNet and two residual blocks of Resnet-50. Figure 3 illustrates the modified network architecture of openpose.



(Figure 3) A modified architecture of openpose [7]

The network feeds the input image with size of height x width with three color channels (i.e. R, G, B) into the Resnet-50 backbone network to produce a set of feature maps F, as depicted in figure 4.



(Figure 4) The first 3 blocks of Resnet-50 [21]

One ConNet and two residual blocks of Resnet-50 consists of 22 convolutional layers. The first convolutional layer (C) filters the input image with 64 kernels of size 7x7 with a stride of 2x2, followed by max-pooling (P) 3x3 with a stride of 2x2. The first residual block consists of one convolutional block (CB) and two identify blocks (IB) which filters [64, 64, 128] kernels of size [1x1, 3x3, 1x1], and the second residual block consists of one convolutional block (CB) and three identify blocks (IB) which filters [128, 128, 512] kernels of size [1x1, 3x3, 1x1].

Then, the feature maps **F** is used as input for pose paring module to generate confident maps **S** and part-associated maps **L**. The network of pose paring module for generating the first step ($\mathbf{S}^1$, $\mathbf{L}^1$) and second step ($\mathbf{S}^t$, $\mathbf{L}^t$) is the same as described in section 3.1.

4. Experimental Results

4.1 Experimental setup

Our experiment is conducted on a system with processor Intel Core™i7-9700 CPU @3.00GHZ, RAM 32.0GB, GPU of NVIDIA GeForce RTX 2070 SUPER. The code is written in Python language and Tensorflow framework with Tensorlayer library [24-25]. The CUDA and CuDNN library are used to accelerate the training. The training is carried by optimizer SGD with momentum set to 0.9, an initial learning rate of 4e-5 with weight decay factor of 5e-4, batch size of 4 and number of iteration of 220,000, 440,000 and 550,000.

The network of original openpose [7] was exploited the pre-trained weight of VGG-19 on ImageNet [22]. To fairly for comparison, we re-train the original openpose and train the modified openpose with Resnet as the backbone network from scratch on COCO 2017 dataset [26]. The COCO 2017 dataset consists of 118K images and 5K for train set and validation set, respectively.

4.2 Training results

The confident map ($\mathbf{S}^t$) and part-associated map ($\mathbf{L}^t$) are the heatmaps from the last unit output. The heatmaps are scores of the loss function. For confident map, the loss function f_S^t is defined by summing the L2 losses of all keypoints. For part-associated map, the loss function f_L^t is defined by summing the L2 losses of all joint-pairs. We adapted the loss function from openpose [7]. The overall loss score f is defined as shown in equation 5.

$$f = \sum_{t=1}^{T} (f_S^t + f_L^t) \tag{5}$$

where t is the number of stage in T = {1,2,..,7}.

Table 1 describes the training performances on COCO train2017 sets. The training were conducted on two backbone networks (VGG-19 and Resnet-50) and pose parsing module (openpose). The network model with VGG-19 backbone generated about 52M parameters, while network model with

Resnet-50 generated about 205M parameters. In order to strive for best loss score, both network models were trained for three iterations (220000, 440000 and 550000). The lowest value of loss score is the best score. The network model with VGG-19 backbone generated the best loss score of 1.1271 for the iteration of 440,000. The network model with Resnet-50 backgone generated the best loss score of 1.5517 for the same iteration.

(Table 1) Training performances on COCO train2017 sets.

Backbone network	Pose parsing module	Number of parameters	Iteration	Loss score
VGG-19	openpose	~52M	220,000	3.6237
			440,000	**1.1271**
			550,000	2.6123
Resnet-50	openpose	~205M	220,000	3.3100
			440,000	**1.5517**
			550,000	3.3196

4.3 Evaluation Results

Our performances are evaluated using COCO assessment library named Object Keypoint Similarity (OKS) [24] with the formula as shown in equation 6.

$$OKS = \frac{\sum_i \exp(-d_i^2/2s^2k_i^2)\delta(v_i > 0)}{\sum_i \delta(v_i > 0)} \quad (6)$$

where d_i is the Euclidean distances between predicted keypoints to the cooresponding ground truth keypoints. s is the scale of person instance within image. k_i is contants defining the distance ratio of all keypoints. v_i is visibility flag of keypoints. The OKS library produces 10 evaluation metrics which measures in Average Precision (AP) and Average Recall (AR). However, we reported only on mean average precision (mAP), mean average recall (mAR), average precision and average recall of threshold OKS greater than 0.5.

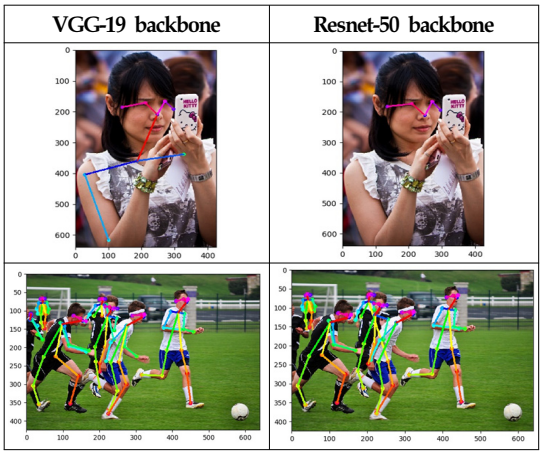
The evaluation performances on COCO val2017 sets of two backbone network models are reported in Table 2. The VGG-19 backbone network predicted the mAP of 33.6% and mAR of 38.2%. Additionally, with the threshold of OKS greater than 0.5, the AP was predicted of 60.9% and AR was predicted of 63.8%. On the other hand, the Resnet-50 backbone network predicted mAP of 28.9% and mAR of 33.8%. With the threshold of OKS greater than 0.5, the AP was predicted of 56.3% and the AR was predicted of 59.4%.

(Table 2) Evaluation performances on COCO val2017 sets.

Backbone network	Pose parsing module	mAP	mAR	$AP_{0.5}$	$AR_{0.5}$
VGG-19	openpose	33.6	38.2	60.9	63.8
Resnet-50	openpose	28.9	33.8	56.3	59.4

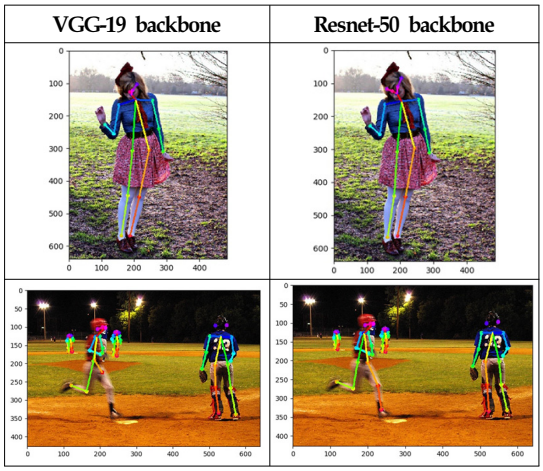
5. Discussion

By training and evaluation results, VGG-19 backbone network is better than Resnet-50 with pose parsing module of openpose. The VGG-19 backbone generated fewer parameters (~52M) comparing to Resnet-50 (~205M). It means VGG-19 backbone was trained faster than Resnet-50. Additionally, the loss score of VGG-19 backbone (1.1271) was also better than Resenet-50 (1.5517). In the test time, the VGG-19 backbone predicted higher accuracy of mAP 4.7% and mAR 4.4% than mAP and mAR of Resnet-50, respectively.



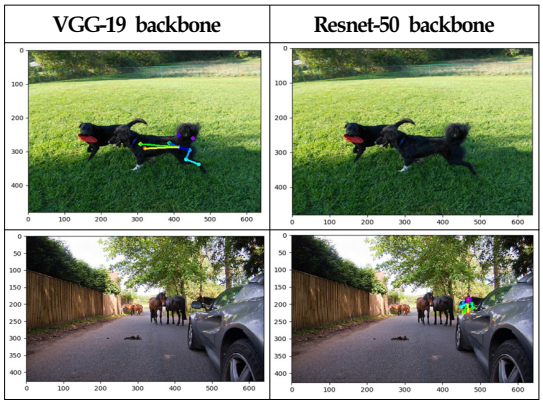
(Figure 5) Example visualizations of VGG-19 backbone outperformed Resnet-50 on COCO val2017 sets.

Figure 5 illustrates some examples visualizing the pose estimation using VGG-19 and Resnet-50 backbone on COCO val2017 sets. In the examples, the Resnet-50 backbone missed some keypoints while VGG-19 backbone performed better.



(Figure 6) Example visualizations of Resnet-50 backbone outperformed VGG-19 on COCO val2017 sets.

Figure 6 illustrates some examples visualizing the pose estimation using VGG-19 and Resnet-50 backbone on COCO val2017 sets. In the examples, the VGG-19 backbone missed some keypoints while Resnet-50 backbone performed better.



(Figure 7) Example visualizations of VGG-19 and Resnet-50 backbone over predicted on COCO val2017 sets.

Figure 7 illustrates some examples visualizing the pose estimation using VGG-19 and Resnet-50 backbone on COCO val2017 sets. In the top row of examples, the VGG-19 backbone over predicted the dog as a person and localized some keypoints while Resnet-50 backbone performed better. On the other hand, the bottom row of examples showed the VGG-19 backbone is better since Resnet-50 backbone over predicted the house as a person and generated some keypoints.

6. Conclusion

In this paper, we presented an empirical analysis and comparison on deep learning network of backbone method of human pose estimation. The plain convolutional neural network (VGG-19) and residual neural network (Resnet-50) were exploited as the backbone network with openpose as the pose parsing module for this experiment. Our experiment showed that VGG-19 network outperformed the Resnet-50 as the backbone with openpose. The reason is we exploited only one ConNet and the first two residual blocks of Resnet-50 while we exploited all convolutional layers of VGG-19. We did not exploit full residual blocks of Resnet-50 because we need the output size of backbone network to fit with the input size of pose parsing module (openpose).

In conclusion, the pose parsing module (openpose) performs better with VGG-19 backbone network which produce fewer parameter, lower loss score, higher accuracy of precision and recall.

Reference

[1] SK. Kim, S. Kang, YJ. Choi, MH. Choi, and M. Hong, “Augmented-Reality Survey: from Concept to Application”, In KSII Transactions on Internet and Information Systems, Vol. 11, No. 2, pp. 982-1004, 2017.
https://doi.org/10.3837/tiis.2017.02.019

[2] A. Toshev, and C. Szegedy, “Deeppose: Human Pose Estimation via Deep Neural Networks”, In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 1653-1660, 2014.
https://doi.org/10.1109/CVPR.2014.214

[3] SE. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, “Convolutional Pose Machines”, In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 4724-4732, 2016. https://doi.org/10.1109/CVPR.2016.511

[4] A. Newell, K. Yang, and J. Deng, “Stacked Hourglass Networks for Human Pose Estimation”, In European conference on Computer Vision, Springer, Cham, pp. 483-499, 2016. https://doi.org/10.1007/978-3-319-46484-8_29

[5] X. Chu, W. Yang, W. Ouyang, C. Ma, AL. Yuille, and X. Wang, “Multi-context Attention for Human Pose Estimation”, In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 1831-1840, 2017. https://doi.org/10.1109/CVPR.2017.601

[6] V. Belagiannis, and A. Zisserman, “Recurrent Human Pose Estimation”, In Proceedings of the IEEE conference on Automatic Face and Gesture Recognition, pp. 468-475, 2017. https://doi.org/10.1109/FG.2017.64

[7] Z. Cao, T. Simon, SE. Wei, and Y. Sheikh, “Realtime Multi-person 2D Pose Estimation using Part Affinity Fields”, In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 7291-7299, 2017. https://doi.org/10.1109/CVPR.2017.143

[8] G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, and K. Murphy, “Towards Accurate Multi-person Pose Estimation in the Wild”, In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 4903-4911, 2017. https://doi.org/10.1109/CVPR.2017.395

[9] B. Xiao, H. Wu, and Y. Wei, “Simple Baseline for Human Pose Estimation and Tracking”, In Proceeding of European Conference on Computer Vision, pp. 466-481, 2018. https://doi.org/10.1007/978-3-030-01231-1_29

[10] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun, “Cascaded Pyramid Network for Multi-person Pose Estimation”, In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 7103-7112, 2018. https://doi.org/10.1109/CVPR.2018.00742

[11] M. Kocabas, S. Karagoz, and E. Akbas, “Multiposenet: Fast Multi-person Pose Estimation using Pose Residual Network”, In Proceeding of European conference on Computer Vision, pp. 417-433, 2018. https://doi.org/10.1007/978-3-030-01252-6_26

[12] X. Hu, J. Tian, H. Qiao, and M. Wang, “Multi-Person Pose Estimation via Learning Feature Integration”, In Journal of Physics, Vol. 1302, No. 3, pp. 032015, 2019. https://doi.org/10.1088/1742-6596/1302/3/032015

[13] B. Rim, N. Sung, J. Ma, Y. Choi and M. Hong, ″Real-time Human Pose Estimation using RGB-D images and Deep Learning,″ Journal of Internet Computing and Services, vol. 21, no. 3, pp. 113-121, 2020. https://doi.org/10.7472/jksii.2020.21.3.113

[14] S. Park, M. Ji and J. Chun, ″A Method for 3D Human Pose Estimation based on 2D Keypoint Detection using RGB-D information,″ Journal of Internet Computing and Services, vol. 19, no. 6, pp. 41-51, 2018. https://doi.org/10.7472/jksii.2018.19.6.41

[15] N. Dalal, and B. Triggs, “Histograms of Oriented Gradients for Human Detection”, In Proceedings of the IEEE computer society conference on Computer Vision and Pattern Recognition, Vol. 1, pp. 886-893, 2005. https://doi.org/10.1109/CVPR.2005.177

[16] Y. Yang, and D. Ramanan, “Articulated Human Detection with Flexible Mixtures of Parts”, In IEEE Transactions on Pattern Analysis and Machine Intelligence,Vol. 35, No. 12, pp. 2878-2890, 2012. https://doi.org/10.1109/TPAMI.2012.261

[17] L. Pishchlin, M. Andriluka, P. Gehler, and B. Schiele, “Poselet Conditioned Pictorial Structures”, In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 588-595, 2013. https://doi.org/10.1109/CVPR.2013.82

[18] M. Dantone, J. Gall, C. Leistner, and LV. Gool, “Human Pose Estimation using Body Parts Dependent Joint Regressors”, In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 3041-3048, 2013. https://doi.org/10.1109/CVPR.2013.391

[19] G. Gkioxari, B. Hariharan, R. Grishick, and J. Malik, “Using k-poselets for Detecting People and Localizing Their Keypoints”, In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 3582-3589, 2014.
https://doi.org/10.1109/CVPR.2014.458

[20] K. Simonyan, and A. Zisserman, “Very Deep Convolutional Networks for Large-scale Image Recognition”, In International Conference on Learning Representations, 2014.
https://arxiv.org/abs/1409.1556

[21] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition”, In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 770-778, 2016.
https://doi.org/10.1109/CVPR.2016.90

[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, AC. Berg, and L. Fei-Fei, “Imagenet Large Scale Visual Recognition Challenge”, In International Journal of Computer Vision, Vol. 115, No. 3, pp. 211-252, 2015.
https://doi.org/10.1007/s11263-015-0816-y

[23] A. Krizhevsky, I. Sutskever, and GE. Hinton, “Imagenet Classification with Deep Convolutional Neural Networks”, In Advances in Neural Information Processing systems, pp. 1097-1105, 2012.
https://doi.org/10.1145/3065386

[24] H. Dong, A. Supratak, L. Mai, F. Liu, A. Oehmichen, S. Yu, and Y. Guo, “Tensorlayer: A Versatile Library for Efficient Deep Learning Development”, In Proceedings of the 25th ACM International Conference on Multimedia, pp. 1201-1204, 2017.
https://doi.org/10.1145/3123266.3129391

[25] LG. Lgarithm, L. Mai, H. Zsdonghao, J. Lui, Boldjoel, Mandeman, S. Fujita, D. Chew, N. Era, and J. Zhang, “High-Performance and Flexible Pose Estimation Framework using TensorFlow, Openpose and TensorRT”, In GitHub repository, 2020.
https://github.com/trustcoinmining/openpose-plus

[26] TY. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and CL. Zitnick, “Microsoft COCO: Common Objects in Context”, In European Conference on Computer Vision, Springer, Cham, pp. 740-755, 2014.
https://doi.org/10.1007/978-3-319-10602-1_48

◐ 저 자 소 개 ◑

림빈보니카(Beanbonyka Rim)

2007: Royal University of Phnom Penh (Cambodia), Computer Science (Bachelor degree)
2010: GroupT – KU Leuven University (Belgium), E-Media (Master degree)
2018 ~ Present: Soonchunhyang University (Korea), Computer Science (PhD degree)
Research Interests : Deep Learning, Image Processing, Computer Graphics, Augmented Reality
E-mail: rim.beanbonyka@sch.ac.kr

김 준 섭(Junseob Kim)

2019: Soonchunhyang University (Korea), Computer Software Engineering (Bachelor degree)
2019 ~ Present: Soonchunhyang University (Korea), Computer Science (Master degree)
Research Interests: Deep Learning, Database, Image processing
E-mail: 20197015@sch.ac.kr

최 유 주(Yoo-Joo Choi)

1991: Ewha Womans University (Korea), Computer Science (Master degree)
2005: Ewha Womans University (Korea), Computer Science (PhD degree)
2006 ~ Present: Professor at Department of Newmedia, Seoul Media Institute of Technology (SMIT), Korea
Research Interests: Image Processing, Computer Vision, Computer Graphics, Augmented Reality, Human-Computer Interaction
E-mail: yjchoi@smit.ac.kr

홍 민(Min Hong)

1995: Soonchunhyang University (Korea), Computer Science (Bachelor degree)
2001: University of Colorado at Boulder (USA), Computer Science (Master degree)
2005: University of Colorado at HSC (USA), Bioinformatics (PhD degree)
2006 ~ Present: Professor at Department of Computer Software Engineering, Soonchunhyang University, Korea
Research Interests: Computer Graphics, Dynamic Simulation, Bioinformatics, Image Processing
E-mail: mhong@sch.ac.kr