Multimedia Tools and Applications (2021) 80:34389–34402
https://doi.org/10.1007/s11042-021-11308-x

1135T: SOCIAL MULTIMEDIA PROCESSING

Real-time low-cost human skeleton detection

Eungyeol Song[1] · **Jinkyung Do**[1] · **Sunjin Yu**[2]

Received: 17 January 2020 / Revised: 27 May 2021 / Accepted: 20 July 2021 /
Published online: 9 August 2021

Abstract
The human skeleton or deep learning framework is useful for accurately recognizing human behavior and analyzing that behavior across different situations. This work introduces a low-cost human skeleton detection network for detecting human skeleton shapes in real time. The proposed network is divided into two parts: pattern extraction and multi-stage convolutional neural networks (CNNs). In the multi-stage CNN step, we use repeated stages, including two branches for the estimation of the heatmap and the part affinity fields (PAFs). In addition, the network consists of inverted bottleneck layers and separable convolutions to extract features efficiently. With test videos, our method achieved an average video analysis speed of approximately 10.45 fps, which is significantly higher than the value of 4.33 fps achieved by the OpenPose algorithm.

Keywords Skeleton detection · CNNs · Human skeleton · Multi-stage CNN

1 Introduction

The recognition performance of human detection algorithms has recently been increased because of improvements in artificial intelligence (AI) technology. As a result, more research into human skeleton detection for the recognition of human actions has been conducted.

Skeleton detection algorithms infer skeleton information from images of humans. A variety of techniques have been developed to recognize skeleton shapes using in-depth information involving, for example, 3D calculations and spatio-temporal information using

✉ Sunjin Yu
sjyu@cwnu.ac.kr

Eungyeol Song
song@codevision.kr

Jinkyung Do
jkdo0923@gmail.com

[1] Research and development department, Codevision Inc 50, Yonsei-ro, Seodaemun-gu, Seoul, Republic of Korea

[2] Department of Culture Technology, Changwon National University, Changwon, South Korea

convolutional neural networks (CNNs) and recurrent neural networks (RNNs) [1, 4, 13]. These methods require the x, y, and z coordinates of human positions, and therefore, they need to be executed on powerful devices due to the high computational load [6, 10]. However, in this study, we detected human skeletons using 2D images with only x and y coordinates, which significantly reduces the computational load [4].

Skeleton-based 2D human pose analysis using CNNs function by performing a localization and estimation of keypoints that represent human-body joints. Previous research presents two different types of methods: top-down and bottom-up. The top-down method typically results in greater accuracy but also higher inference times than the bottom-up method. The top-down method is composed of two steps: 1) detect a human region using a human detector and 2) recognize a human skeleton in the detected region. HRNet is an example of a top-down method that uses a high-resolution input image to achieve good performance [18], demonstrating an average precision (AP) of approximately 75.5% for the COCO dataset [14]. However, this method has a critical problem in that the inference time increases with the number of people in the image. In contrast, bottom-up methods are performed with a stable inference time regardless of the number of people but have comparatively lower accuracy. OpenPose encodes the global context of an input image using a heatmap and part affinity fields (PAFs) [3]. It is composed of a VGG-19 network and repeated CNN stages to extract the heatmap and PAFs. It has achieved a mean average precision (mAP) of approximately 60.6% but only requires an inference time of 0.005 s per image. Previous studies on skeleton detection have used repetition of deeper and more complex neural networks to achieve more accurate performance. However, in this study, we prioritize the networks with higher speed [17, 22].

Our proposed algorithm is divided into two parts: pattern extraction and multi-stage CNNs. In the pattern extraction step, the network calculates a proper feature. Multi-stage CNNs calculate an 18-dimensional heatmap and 36-dimensional PAFs.

The rest of this paper is organized as follows. Section 2 describes the reduced weighting of research data (research on reducing the number of parameters and the latency of skeleton detection has intensified recently, although much effort has been focused on improving the performance in joint detection technology as well). Section 3 introduces the algorithm proposed in this study. Section 4 describes the experimental environment and dataset and presents the evaluation methods and results. Finally, Section 5 concludes the paper.

2 Related work

Several studies have been conducted on image-based deep learning to achieve increased accuracy when solving detection, recognition, and classification problems. Yann LeCun's LeNet–5 first proposed the convolutional neural network and heralded the beginning of deep learning. Later, AlexNet was presented at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), to evaluate its classification performances on the ImageNet dataset. AlexNet uses a 224 × 224–sized RGB 3 channel image as an input image and has a memory–sharing structure at the middle of its network. This memory–sharing technique was designed to solve a memory shortage problem in the GPU's VRAM. AlexNet represented the idea that rectified linear units can be used to achieve rapid convergence in image classification. Additionally, it introduced the overlapping pooling technique, which renders the kernel size of pooling larger than that of the stride. AlexNet also

demonstrated its abilities through the incorporation of other techniques, including data augmentation using dropout and principal component analysis [11, 12, 15].

In 2014, the Oxford Visual Geometry Group introduced the VGG. Unlike other existing AI technologies, the VGG proposed the use of a 3×3 convolution filter. This filter is made up of three overlapping 3×3 convolutional layers, making it the equivalent of one 7×7 convergence with respect to its receptive field. However, the filter uses a larger number of activation functions, which results in greater nonlinearity. Furthermore, the filter has the added advantage of using fewer parameters than previous methods.

Around the same time, GoogLeNet (Inception) secured the first place at the 2014 ILSVRC. GoogLeNet is composed of twenty–two layers and has a long and complex structure. A main characteristic of GoogLeNet is its use of a block structure, referred to as an Inception module. Other existing models connect each layer through a single convolution and one pooling operation; however, the Inception module is characterized by its use of concatenation, whereby four different operations are employed followed by combining the feature map in the direction of the channel. Additionally, 1×1, 3×3, and 5×5 convolutions are intermixed to express various receptive fields [11, 15, 17].

One effective object detection algorithm is the one proposed by Jun Chu, based on multi-layer convolution feature fusion (MCFF) and online hard example mining (OHEM). The MCFF is effective for extracting and detecting candidate regions of interest from an image. Multi-layer data contain more complete information than single–layer data, meaning the former is better suited to detect objects of different scales, and as a result, MCFF is effective in object detection amongst various candidate regions [5].

Traditionally, studies on deep learning have developed networks based on recognition rates. These would later create derivations that focused on various other factors such as efficiency. Inception–v2, the follow-up technology of GoogLeNet, proposed different techniques including factorizing the convergence filter, rethinking the auxiliary classifier, and avoiding a representational bottleneck. There have since been more iterations of the Inception network, and currently, Inception-v4 consists of the following: a Stem Block, which directly connects to the input; three Inception blocks (Inception-A, Inception-B, Inception–C); and two reduction blocks (Reduction-A, Reduction-B), which reduce the feature map to half its size. Inception-v4 is the expanded version of Inception–v2 and Inception–v3 with an optimized and simplified architecture [20, 21]

Another method previous studies have used to obtain more sophisticated results is systems being equipped with expensive GPUs. However, more recent studies have presented novel techniques for creating lightweight AI architectures without such expensive GPUs. For example, at the 2017 CVPR, Xception introduced the depthwise separable convolution to make improvements upon the Inception structure. This method guides the model through a training process that separately maps the cross–channel correlations and the spatial correlations of images [20, 21].

The biggest contribution made toward creating a lightweight AI architecture is the MobileNet study. The MobileNet architecture was the most important in that it employed depthwise separable convolutions along with Xception. The difference between this study and the previous pose estimation studies is its use of a novel architecture and the addition of batch normalization and ReLU between the depthwise convolution and point convolution. Moreover, the Xception architecture employed depthwise separable convolutions to obtain greater accuracy when compared with Inception, whereas the MobileNet architecture used the same convolution to construct a lightweight structure that was operable on a mobile device [8, 16].

Subsequently, MobileNetv2 was released in 2018. Compared with its previous version, MobileNetv2 has been upgraded with lower accuracy and fewer parameters and operations in its convolution block. In addition, SqueezeNet was introduced to reduce its number of parameters through extensive use of the squeeze and expand modules and the 1×1 convolution [9, 16].

3 Proposed method

A human pose skeleton represents the orientation of a person in a graphical format. Essentially, it is a set of coordinates that can be connected to describe the pose of the person. Each coordinate in the skeleton is called a part, joint, or keypoint. A valid connection between two parts is called a pair or limb.

The simplest approach to skeleton detection is the top–down approach, wherein you use a person detector first, followed by estimating the parts and then calculating the pose for each person. The alternate approach is the bottom–up approach, wherein you detect all parts in the image (i.e., parts of every person) and then associate or group these parts as belonging to distinct persons.

There are three models used for model compression: MobileNetV1 and MobileNetV2. We used a separable convolution consisting of the inverted bottleneck layer used in MobileNetV2 and the depthwise and pointwise convolution proposed in MobileNetV1 [8, 16].

Figure 1 displays the overall pipeline of the proposed algorithm. The algorithm is divided into two steps: pattern extraction and multi–stage CNNs. The former is composed of inverted bottleneck layers, and the latter is composed of two branches: Branch 1, which calculates a heatmap for the image, and Branch 2, which calculates part affinity fields (PAFs). The heatmap includes the positions of human key points, and the PAFs include the association between pairs of keypoints. The process is conducted as follows:

1. The system takes the width and height of the image as input.
2. Ten inverted bottleneck layers are used to train features from the input image.
3. Using the output of step 2) as the input, the first stage of training is executed.
4. Repeat the stages composed of two branches. From the N^{th} stages (where N≤2), the output of the $(N–1)^{th}$ stage is used as the input of the N^{th} stage.

In the pattern extraction component, we used 10 inverted bottleneck layers. The inverted bottleneck layer was introduced in the MobileNetV2 architecture, and it is composed of three convolution layers with 1×1 and 3×3 kernels. As shown in Fig. 2a, the dimension of the input image channel was increased by a constant ratio using the

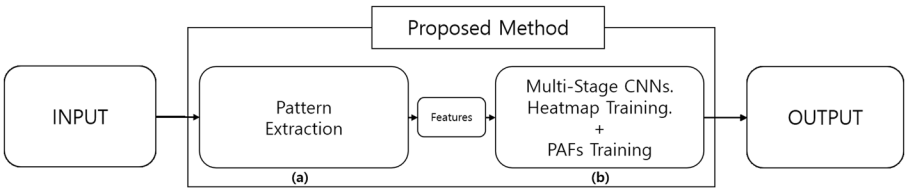

Fig. 1 Architecture of our proposed method: **(a)** Pattern Extraction and **(b)** Multi–Stage CNNs. Feature concatenates inverted bottleneck in pattern extraction and separable convolution in multi–stage CNNs

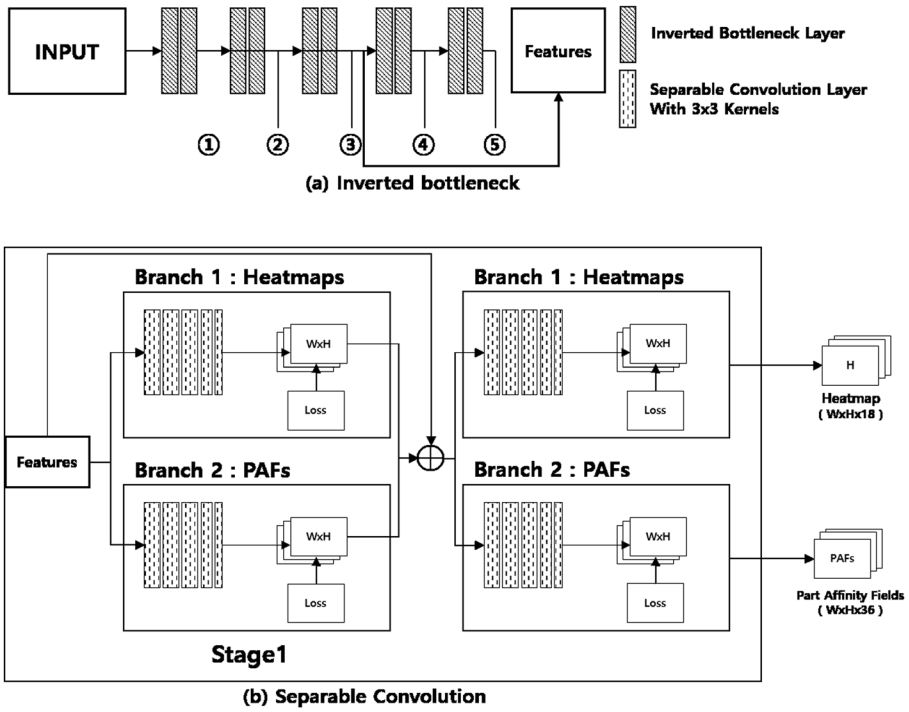

Fig. 2 Architecture of layers: **(a)** inverted bottleneck **(b)** separable convolution

1×1 kernel convolution layer. Then, we extracted features using the 3×3 kernel convolution layer. Finally, we used the 1×1 kernel convolution layer to adequately match the dimension of the output. From the network, we can use five intermediate outputs with different sizes, which are calculated from two continuous inverted bottleneck layers. The minimum size of the five intermediate outputs is 1/8 of the size of the input image. By using the pattern extraction step, we can create a high–efficiency network that can be used in restricted environments or environments with low computing power, such as those using only CPU or mobile devices [16].

The multi–stage CNN step generates a heatmap and PAFs using multiple repeated CNNs. One CNN stage is composed of two branches. Branch 1 produces an 18–dimensional heatmap, and Branch 2 produces 36–dimensional PAFs. Each branch has three separable convolution layers and two convolution layers. The three separable convolution layers are each composed of one 3×3 kernel and one 1×1 kernel, and the two convolution layers are each composed of 1×1 kernels. The separable convolution layers allow the system to operate with high efficiency by reducing the computational load. The separable convolution layer uses 3×3 kernels for depthwise operations and 1×1 kernels for pointwise operations, as shown in Fig. 2b. Additionally, each branch uses the output of the previous stage as its input for refinement. For Branch 1 and Branch 2, the convolution layers compute new outputs by refining the previous outputs and calculating the loss of the new heatmaps and PAFs.

3.1 Inverted bottleneck layer

The inverted bottleneck layer is used to reduce the depth and width of the architecture while maintaining strong performance. It is composed of a channel expansion operation and a channel reduction operation. The expansion operation creates features through non–linear calculations. Kernels that are of 3×3 dimension are used to extract feature maps and expand the channel dimension. The reduction operation, contrarily, helps remove useless features by reducing the channel dimension. This operation is required to maintain the overall training stream of the network for skeleton detection. In addition, in our model, we used scale–invariant feature maps, which can replace a residual network. These scale–invariant feature maps are necessary for detecting the existence of skeleton keypoints. From the input image, it is necessary to find keypoints of various skeletons. Therefore, we concatenated five outputs with different sizes for training the various types of keypoints.

The size of the concatenated outputs was matched to the intermediate output 3 in Fig. 2a: 1and 2 are resized to the size of 3 using max pooling, and 4 and 5 are resized to the size of 3 using upsampling and bilinear interpolation. These five concatenated outputs can replace a residual network, which results in a decrease in speed. By using the scale–invariant architecture with the inverted bottleneck layers, we used only 48 to 256–dimensional feature maps rather than a deeper network. This has the benefit of a less complicated operation, which thereby increases the speed in the training and test processes [7, 19].

3.2 Separable convolution

MobileNetV1 is composed of three kinds types of layers: convolution, depth-wise convolution, and pointwise convolution. A separable convolution is proposed for improving to improve the performance without increasing the depth and or width of the network. The proposed architecture optimizes the trade–off between performance and efficiency when compared to other networks, such as VGG[17], GoogLeNet[3], inception Inception modules [20], and ResNet [5]. The depthwise and pointwise convolution layers makes sparse connected layers, which reduces the gradient vanishing problem by extracting the optimal features. Through calculations , we can obtain highly correlated features with dense calculations. The depthwise convolution with 1×1 filter kernels are was used to control the dimension of the output feature maps, and the pointwise convolution with 3×3 filter kernels are was used to extract feature maps. The A comparison of between the number of operations between resulting from using a convolution layer and dividing it into pointwise and depthwise convolutions can be computed with expressed using Eq. 1. For this computation, we assume that the layer has size $m \times m$, an input channel of size n_1, and an output channel of size c_1.

$$m^2 3^2 n_1 c_1 \quad : \quad m^2 1^2 n_1 c_1 \quad + \quad m^2 3^2 n_1 c_1^2 \quad = \quad 9n_1 : n_1 + 9c_1 \quad (n_1 > c_1) \qquad (1)$$

Because reducing the number of dimensions also reduces the computational load by approximately a factor of two, this operation has the effect of improving performance while maintaining a high convergence speed during the training procedure.

3.3 Loss computation

Equation 2 is an L2 loss function, which calculates the difference between the ground truth y_i and the predicted value fx_i, and sum the squared values of the calculated differences. The network is trained to minimize the L2 loss.

$$L = \sum_{i=1}^{n} (y_i - f(x_i))^2 \tag{2}$$

3.4 Network architecture

Tables 1 and 2 show the overall structure of the proposed network. For the pattern extraction process, the deeper the network, the smaller the width and height of the output, but the larger the dimension of the outputs. For the multi–stage CNN step, the dimensions of the intermediate layers are different in Branch 1 and Branch 2. In Branch 1, the outputs of the three separable layers have 64, 64, and 128 dimensions, respectively, and the outputs of the two continuous convolution layers decrease the 128 dimensions to an adequate number of dimensions to match the number of heatmaps and PAFs. Branch 1 generates the heatmap, which has 18 dimensions. In Branch 2, 36–dimensional PAFs are generated. One CNN stage includes Branch 1 and Branch 2. The stages are repeated six times, and each stage uses a concatenated feature map as input. The feature map is composed of the output of the pattern extraction stage, which is composed of the output of the previous stage's Branch 1 and Branch 2. As shown in Table 2, the output channel size (i.e., the size of the output) has different values depending on the stage.

Table. 1 The architecture of pattern section

name	input channel size	Operator	up_ratio	Output channel size	k_s	stride
part0_1	3	Inverted bottleneck	1	24	3	1
part0_2		Inverted bottleneck	1	24	3	1
part1_1	24	Inverted bottleneck	6	48	3	2
part1_2		Inverted bottleneck	6	48	3	1
part2_1	48	Inverted bottleneck	6	64	3	2
part2_2		Inverted bottleneck	6	64	3	1
part3_1	64	Inverted bottleneck	6	128	3	2
part3_2		Inverted bottleneck	6	128	3	1
part4_1	128	Inverted bottleneck	6	256	3	2
part4_2		Inverted bottleneck	6	256	3	1
feat_concat		Concatenation [w/8, h/8, 520]		part_0_2.max_pool(4,4) part_1_2.max_pool(2,2) part_2_2 part_3_2.upsample(2) part_4_2.upsample(4)		

Table. 2 The architecture of multi-stage CNNs

Stage	Operator	k_s	Output channel size
Input	{feat_concat}		
0	Separable conv	3	128
	Separable conv	3	128
	Separable conv	3	128
	Conv	1	512
	Conv	1	19 (branch 1), 38 (branch 2)
Input	{concatenated {feat_concat, branch 1, branch 2}}		
1~5	Separable conv	3	64
	Separable conv	3	64
	Separable conv	3	64
	Conv	1	128
	Conv	1	19 (branch 1), 38 (branch 2)

4 Experimental results

4.1 Experimental environment

Our training was performed with an NVIDIA Quadro RTX 8000 on Windows 10 with Python 3.7 and Tensorflow r1.14. The experiment was tested in a CPU–only environment.

4.2 Datasets

We used the 2017 COCO keypoint dataset to train the network. The dataset is composed of 118,287 training images and 5,000 validation images. Figure 3a shows sample images of the COCO dataset. As shown in Fig. 3b, the dataset includes 18 keypoints, which have 19 pairs. Each keypoint indicates a part of the human body (0: nose, 1: neck, 2: right shoulder, 3: right elbow, 4: right wrist, 5: left shoulder, 6: left elbow, 7: left wrist, 8: right hip, 9:

Fig. 3 **(a)** Sample images from the COCO 2017 key point dataset. **(b)** Sample images from the dataset

right knee, 10: right ankle, 11: left hip, 12: left knee, 13: left ankle, 14: right eye, 15: left eye, 16: right ear, and 17: left ear) [3].

4.3 Evaluation metrics

4.3.1 Object keypoint similarity (OKS) based map

The Object Keypoint Similarity (OKS) metric uses the distance between a normalized coordinate of a predicted value and a ground truth value. It calculates the average similarity of visualized keypoints. The similarity is determined by a Gaussian distribution of ground truth keypoints and predicted keypoints. According to the association of keypoints, a smaller standard deviation signifies a more accurate result.

$$\begin{aligned} \mathrm{ks}(\hat{\theta}_{\mathrm{i}}^{(\mathrm{p})}, \theta_{\mathrm{i}}^{(\mathrm{p})}) &= \mathrm{e}^{-\frac{\|\theta_{\mathrm{i}}^{(\mathrm{p})} - \theta_{\mathrm{i}}^{(\mathrm{p})}\|_2^2}{2s2\mathrm{k}_{\mathrm{i}}^2}} \\ OKS((\hat{\theta}_i^{(p)}, \theta_i^{(p)}) &= \frac{\sum i \; ks(\hat{\theta}_i^{(p)}, \theta_i^{(p)}) \; \delta(v_i > 0)}{\sum i \; \delta(v_i > 0)} \end{aligned} \tag{3}$$

In Eq. 3, d_i is the Euclidian distance, u_i is the visibility flag, sk_i is the standard deviation of the keypoints, s is the scale of the keypoint region, and k_i is the keypoint constant.

Figure 4a, b explain how to calculate the distance distribution using the standard deviation. As shown in Figs. 4b and 5 each keypoint has different constants. If the distances between the

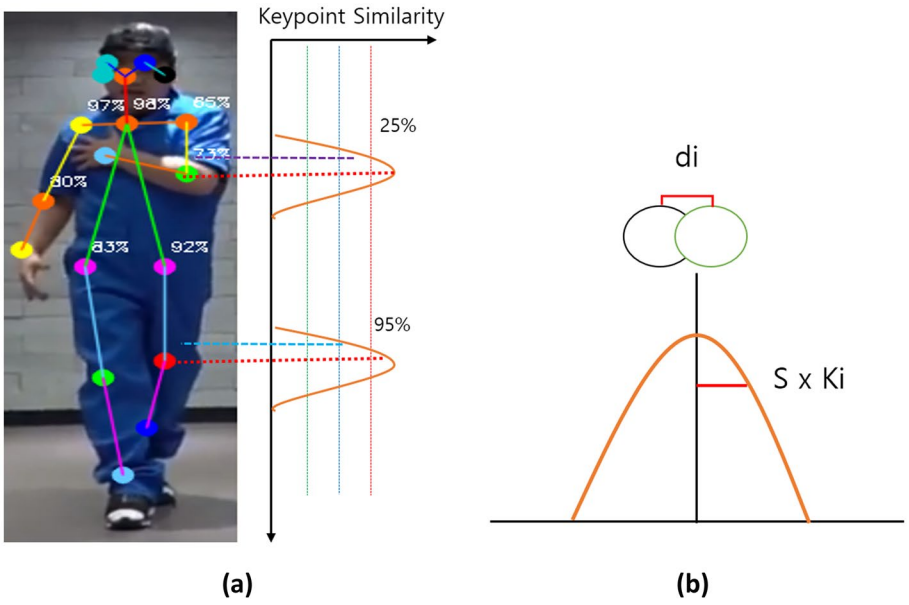

Fig. 4 **(a)** Keypoint similarity distribution that changes with the location of the predicted keypoint (left eye: 0.25, right hand: 0.75; the blue point is the ground truth, and the red and green points are the predicted values.) **(b)** Gaussian distribution with standard deviation $s \times k_i$

Keypoint	Ki
Hips	0.107
Ankles	0.089
Knees	0.087
Shoulders	0.079
Elbows	0.072
Wrists	0.062
Ears	0.035
Nose	0.026
Eyes	0.025

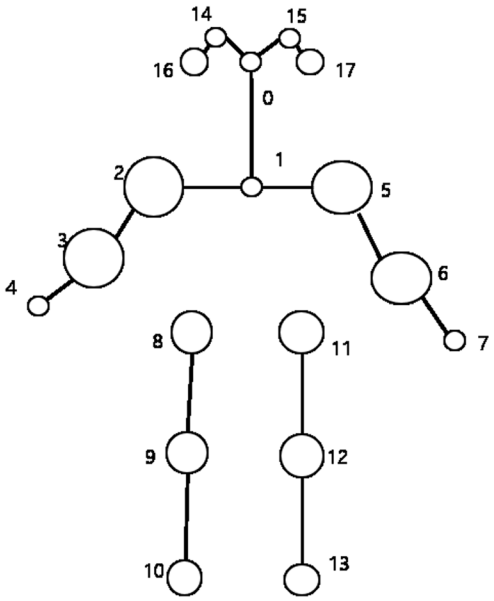

Fig. 5 Keypoint constants. k_i

center of the green circle, ground truth(black circle), and the predicted value are small, the value of OKS approaches 1.

4.3.2 AP (Average Precision)

We used the average precision (AP) for the OKS value. With this metric, we use AP$\hat{N}$ (N=50), which means averaging the AP with OKS values larger than 0.5. The total AP is then determined by the mean AP scores for 10 positions. In addition, we can define AP$\hat{M}$ as the AP for medium objects and AP$\hat{L}$ as the AP for large objects.

4.4 Evaluation r esults

To analyze the runtime performance of our system, we used videos that contained one or more people exercising. The speed was measured in the CPU environment during the evaluation to assess performance The original frame size of the videos was 2224×1080, and we resized the frame to 224×192 to infer the skeleton information. Figure 6 shows the example test outputs: Fig. 6a shows the input image, Fig. 6b displays the analyzed heatmap, Fig. 6c shows the analyzed PAFs with the x–direction vector information, and Fig. 6d displays the analyzed PAFs with the y–direction vector information. Each map indicates the keypoint existence probability. We tested the algorithm on videos containing multiple situations and compared the detection performance with those of state– of– the– art algorithms.

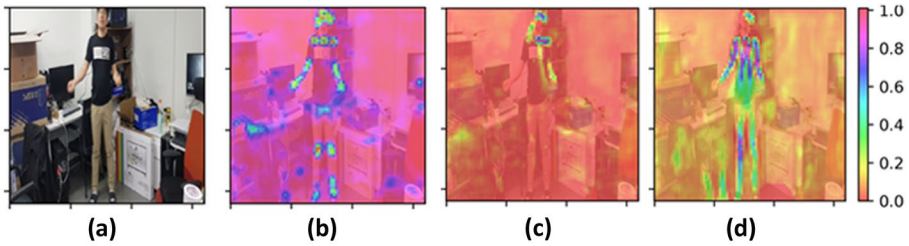

Fig. 6 Example test outputs for video frames: **(a)** input image, **(b)** heatmap, **(c)** PAFs – x– direction vector, and **(d)** PAFs– y – direction vector

4.4.1 Experiments on video

Figure 7 shows the visualized result of the proposed algorithm. Input for the test process was mp4 videos. The videos of Fig. 7b–d were selected from the ActivityNet dataset. Each video has labels regarding the specific situation taking place in the video, and we selected videos that portrayed common behavior situations. Each video had different labels: Fig. 7b is "Doing karate," Fig. 7c is "Platform diving," and Fig. 7d is "Long jump air and landing drills." To evaluate our system, we first used videos recorded in an indoor environment. For example, Fig. 7a shows a person standing in one place and the associated skeleton drawing output. Figure 7b shows a situation with multiple people indoors. There are six people in the video, and the video takes place in a poorly lit environment. Figure 7c shows a person diving into a pool. Despite the blurred frames, our algorithm was able to detect the person and analyze their skeleton information. Finally, we used a video showing people jogging.

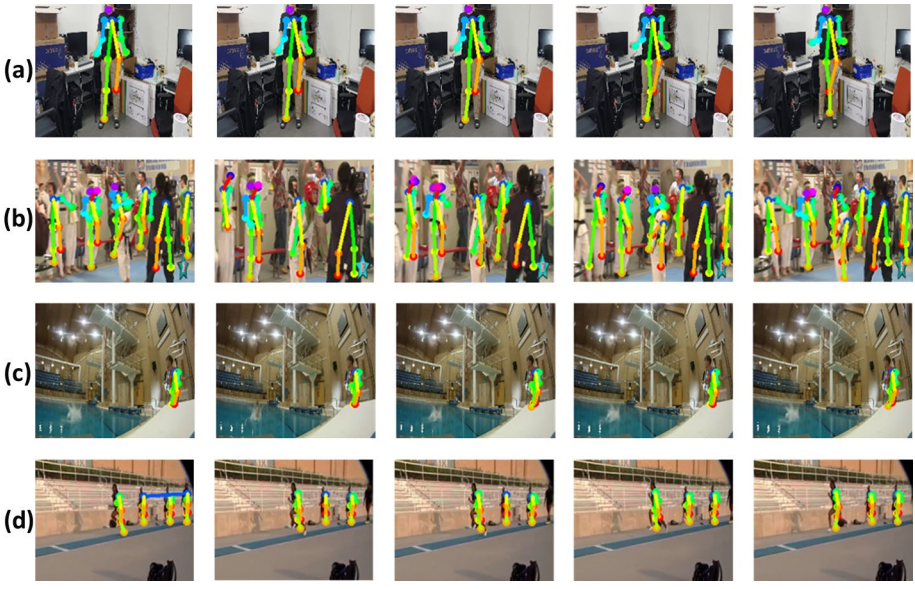

Fig. 7 Qualitative results on multiple videos: **(a)** video of one person, **(b)** video of multiple persons, **(c)** video of bright environment, and **(d)** video of dynamic situation

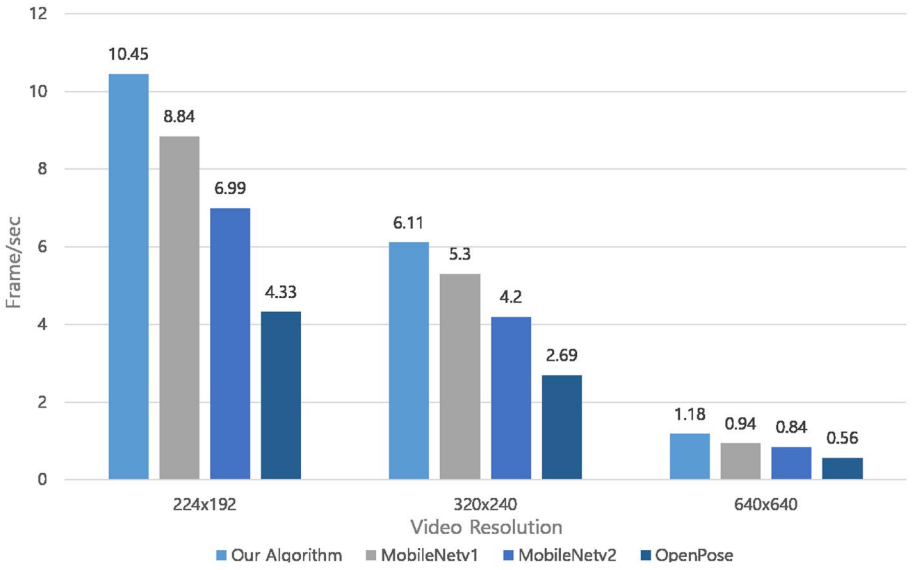

Fig. 8 Speed comparison of different algorithms on the sample videos

Figure 7d shows several people jogging forward at high speed. Even in this situation, our algorithm showed stable performance [2].

4.4.2 Comparison to state– of– the– art results on video speed

As shown in Fig. 8, we compared the speed of our algorithm to those of other state-of-the-art bottom-up detection methods. All tests were executed in the CPU environment (i.e., without a dedicated GPU). We used the same videos for all algorithms, resized them to measure frames per second (fps), and used the averaged fps value. The size of the videos was normalized to a width of 224 and a height of 192. When we tested our algorithm with the videos, we obtained an average of 10.45 fps, which was approximately 2.4 times higher than the value for OpenPose. Next, testing with a 320×240 video, our algorithm achieved 6.11 fps, which was approximately 2.3 times higher than the value for OpenPose. Finally, testing with a 640×640 video, our algorithm achieved 1.18 fps, which was approximately 2.1 times higher than the value for OpenPose. The results clearly indicate that the larger the size of the input image, the slower the processing speed for each algorithm. However, overall, our algorithm was more than twice as fast as OpenPose across all cases. Therefore, our method demonstrated an improved result over prior methods with respect to process time [14, 17, 21].

5 Conclusion

Our proposed artificial intelligence skeleton detection network demonstrated an improved processing speed in a CPU-only environment that does not use GPUs. One of the problems in skeleton detection is that the processing speed depends on the detection method.

Therefore, we proposed a bottom-up detection network that does not depend on different model hierarchies and training architectures. The proposed method is divided into two parts: pattern extraction and multi-stage CNNs. In the pattern extraction step, 10 inverted bottleneck layers are used to extract feature maps. For this step, we concatenated the outputs with five different sizes generated in the intermediate layers and used them as the input for the next step. The multi-stage CNN step extracts a heatmap (representing the probability distribution of keypoint locations) and PAFs (vector information representing the association of the keypoints). The output of our algorithm is an 18-dimensional heatmap and 36-dimensional PAFs. We evaluated our dataset using the 2017 COCO keypoint training set for training and videos from the ActivityNet dataset for testing. When we tested the algorithm with a 224×192 video, our algorithm achieved 10.45 fps, which was 2.41 times higher than the value for OpenPose. Additionally, we tested the algorithm with various videos, and those tests also demonstrated that our algorithm is faster than OpenPose under identical conditions.

In future, because our algorithm has shown efficiency in a CPU-only environment, we believe that it can be useful in the mobile environment as well. In mobile environments, there are no GPUs or other high-speed devices, and therefore, implementing our approach will contribute to accelerating real-time analysis performance.

Acknowledgements This work was supported by the Technology Innovation Program (20006697, multi sensor based artificial intelligence technology passenger recognition and air clean console for autonomous design companion animal family centered) funded By the Ministry of Trade, Industry & Energy (MOTIE, Korea).

References

1. Asadi-Aghbolaghi M, Kasaei S (2018) Supervised spatio-temporal kernel descriptor for human action recognition from RGB-depth videos. Multimed Tools Appl 77(11):14115–14135
2. Caba Heilbron F et al (2015) "Activitynet: A large-scale video benchmark for human activity understanding." Proc IEEE Conf Comput Vis Pattern Recognit
3. Cao Z et al (2017) "Realtime multi-person 2d pose estimation using part affinity fields." Proc IEEE Conf Comput Vis Pattern Recognit
4. Carrara F et al (2019) LSTM-based real-time action detection and prediction in human motion streams. Multimed Tools Appl 78(19):27309–27331
5. Chu J, Guo Z, Leng L (2018) Object Detection Based on Multi-Layer Convolution Feature Fusion and Online Hard Example Mining. IEEE Access 6:19959–19967. https://doi.org/10.1109/ACCESS.2018.2815149
6. Donahue J et al (2015) "Long-term recurrent convolutional networks for visual recognition and description." Proc IEEE Conf Comput Vis Pattern Recognit
7. He K et al (2016) "Deep residual learning for image recognition." Proc IEEE Conf Comput Vis Pattern Recognit
8. Howard AG et al (2017) "MobileNets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861
9. Iandola FN et al (2016) "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size." arXiv preprint arXiv:1602.07360
10. Ji S et al (2012) 3D convolutional neural networks for human action recognition. IEEE Trans Pattern Anal Mach Intell 35(1):221–231
11. Krizhevsky A, Sutskever I, Hinton GE (2012) "Imagenet classification with deep convolutional neural networks." Adv Neural Inf Proces Syst
12. LeCun Y et al (1998) "Gradient-based learning applied to document recognition." Proc IEEE 86(11):2278–2324
13. Li B et al (2018) 3D skeleton based action recognition by video-domain translation-scale invariant mapping and multi-scale dilated CNN. Multimed Tools Appl 77(17):22901–22921

14. Lin TY et al (2014) “Microsoft coco: Common objects in context.” European conference on computer vision. Springer, Cham
15. Russakovsky O et al (2015) Imagenet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252
16. Sandler M et al (2018) “MobileNetv2: Inverted residuals and linear bottlenecks.” Proc IEEE Conf Comput Vis Pattern Recognit
17. Simonyan K, Zisserman A (2014) “Very deep convolutional networks for large-scale image recognition.” arXiv preprint arXiv:1409.1556
18. Sun K et al (2019) “Deep high-resolution representation learning for human pose estimation.” Proc IEEE Conf Comput Vis Pattern Recognit
19. Szegedy C et al (2015) “Going deeper with convolutions.” Proc IEEE Conf Comput Vis Pattern Recognit
20. Szegedy C et al (2016) “Rethinking the inception architecture for computer vision.” Proc IEEE Conf Comput Vis Pattern Recognit
21. Szegedy C et al (2017) “Inception-v4, inception-resnet and the impact of residual connections on learning.” Thirty-first AAAI conference on artificial intelligence
22. Wei SE et al (2016) “Convolutional pose machines.” Proc IEEE Conf Comput Vis Pattern Recognit