16th International Conference on Machine Vision Applications (MVA)
National Olympics Memorial Youth Center, Tokyo, Japan, May 27-31, 2019.

End-to-End Feature Pyramid Network for Real-Time Multi-Person Pose Estimation

Dingli Luo[1,2], Songlin Du[1], and Takeshi Ikenaga[1]

[1]Graduate School of Information, Production and Systems, Waseda University,
Kitakyushu 808-0135, Japan
[2]University of Electronic Science and Technology of China,
Chengdu 611731, China
luodingli@toki.waseda.jp

Abstract

In computer vision, pose estimation system is widely used to construct human body transformation. However, it is hard to achieve these targets together: stable real-time speed, variance human number and high accuracy. This paper proposes an end-to-end pose estimation network. It contains a neural network friendly representation of human pose. Then it proposes a correspond real-time end-to-end pose estimation network based on feature pyramid network structure with attention-based detection modules. This network can detect multiple humans in more than 60 fps with 384×384 *resolution on GTX 1070 with affordable accuracy. This work shows the potential of this network structure can perform both faster and better compared with state-of-the-art results.*

1 Introduction

The task of pose estimation system is taking an RGB image as input then determining the transformations for every body part of each person in the image. This kind of system has been widely used in the motion capture systems [1], sports analysis systems [2] and augmented reality applications [3].

Many of them have real-time constraints to make the whole system have good performance. It is a problem to both have good accuracy and stable high speed with variance human number in inputs. There are two kinds of deep learning based methods have been proposed: two-stages and one-stage. Two-stages methods like Mask-RCNN [4] first recognize each person in the image then use single person pose estimation network to detect human pose one-by-one. This can give high accuracy but can hardly achieve stable real-time speed with variance human number. One-stage methods like OpenPose [5] directly produce middle information of all humans' pose and then use the traditional method to construct each human's pose. These methods don't need the recognition for each human and the speed of one-stage methods is not affected by the number of humans and can archive stable high speed. So these method is more popular in real-time applications.

Although the one-stage solution is fast, the connection term causes accuracy loss. The network structure limited the middle information used by connection term. Moreover, the less information causes the lower accuracy. However, the more information the network produces, the slower the network performs. It is not easy to find the balance manually. Also, to provide high-quality middle information, many researchers add many redundancy structures. The redundancy structures also slow down the whole process. Finally, although this network is designed as one stage, users process these network with the image in different scales to adapt the different size of humans in the image. Processing more than one time is much slower.

Unlike the other one-stage solution, this paper proposes an end-to-end pose estimation neural network for archiving real-time speed in affordable accuracy loss. Our network directly produces the connection information instead of middle information. The only calculation after network processing is linking the joints based on network output which is easy and fast. By using the context information, the network produces higher quality connection predict results. To fit the different size of humans, a feature pyramid network [6] structure is added to adapt different scale of human in one time. Moreover, attention based mechanism is also added to mask out the best suitable area for different scale of feature maps. These three methods help us to produce a smaller and faster network.

Our contributions are summarized as follows:

- We propose a new neural network friendly representation of human pose to make the network directly produce a human pose.
- We propose a feature pyramid and attention based network structure to do multiple human pose estimation faster.
- We give the performance report of our network by the evaluation on Microsoft COCO 2014 [7] dataset and prove our network achieves better performance compared with current state-of-the-art solution.

2 End-to-End Feature Pyramid Network

2.1 Human pose representation

To make the neural network directly produces the pose transform, a data structure have to be designed in order to represent each joint of the body.

A joint contains three kinds of information: the center position, the class of current joint and the parent joint. a feature map is used to represent every joint in the input image. Each body part is encoded at center position $\{x, y\}$ in feature map with a vector

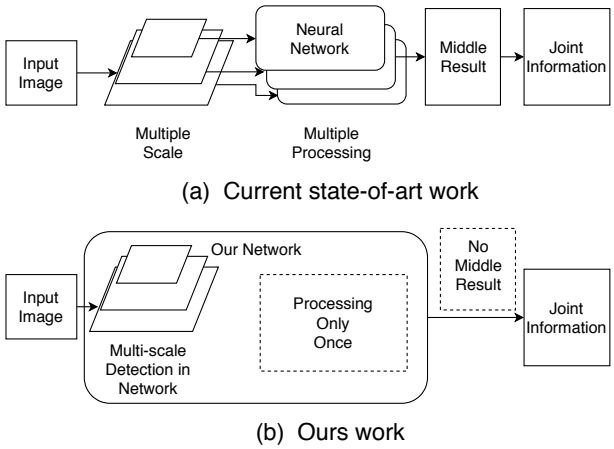

Figure 1. The overall structure of our system and the comparison between current state-of-the-art work. Current state-of-the-art work adapt different human sizes by processing the network multiple times in different image scale. And the network produces one kind of middle result and then use another step to get the final result. Our work uses a build-in multiple-scale detection module in network. And our network directly produces the joint information without the middle results. This makes our network processes faster.

$\{P_c, TargetOffset\}$. P_c represents the probability of body part class.

$$TargetOffset = (TargetX - X, TargetY - Y) \quad (1)$$

represents the offset between joint's target location and the center.

For given input image in size $(width, height)$, the network will produce a $(width/8, height/8, 9)$ feature map called heat map and a $(width/8, height/8, 2)$ feature map called connection map. The vector at $\{x, y\}$ in heatmap represent the probability P_c of current position $\{x, y\}$ is the center of the following body parts: $\{Nose, Head, Shoulder, Elbow, Hand, Pelvis, Knee, Foot\}$. The left or right of each body part is not identified by network because it is tough to request the neural network to classify left or right. Classifying the side class based on the local area is impossible. So, the network needs much more calculation to identify the side information with global information. Then the vector at $\{x, y\}$ in connection map represents the target position offset $TargetOffset$ if current position is any center of body part. If it is not any body part center, the output is directly removed which means any result is affordable.

Current joint's target location offset is used as output instead of directly output the target joint's memory address because currently there is no good solution to achieve this. A long short-term memory neural network (LSTM) [8] may be used to generate a directed graph of joints one by one, but the speed cannot achieve our need. So the network outputs the target location offset directly. Then use a simple search to find the nearest joint with correspond class near the target location and connect them.

We split the pose estimation problem into two different problems:

Classification Problem: Classify the class of a local area is easy. All tasks need global information are removed like side classify.

Regression Problem: Regression problem means that the network predicts the target location offset in current output pixel position $\{x, y\}$ but do not consider the class of this pixel. Different from [5], the target location can be directly calculated by this offset instead of searching PAF maps.

This design significantly improves the performance of small size network and improve the speed of the current network.

2.2 Network architecture

The network architecture is shown in Figure 2. The whole network is constructed by 4 parts. They are as follows:

Convolution Part: We use ResNet34 [9] as a backbone. The input image is scaled down by a set of convolutional neural networks. Using residual connections between convolutional layers has been reported to show good performance compared with building large size network. It also solves the gradient vanishing problem in deep neural networks.

Deconvolution Part: We take the last layers output from ResNet34 and processed by a deconvolution layer. Then concatenate the deconvolution layer's output with the correspond convolution layer's output. We repeat this Num_deconv times. We decided to make $Num_deconv = 3$ based on our test results.

Map Generation Part: Each output of deconvolution layers are processed by a "Detect Module". Then the results are used by two different "Map Generation Module" to generate one heat map and one connection map individually. We design this because we think part of the calculation of heat map and connection map is shared to speed up.

Output part: In this part, the different output from deconvolution layer with same map class are concatenated together. Then use one 1×1 convolution layer to generate the final output.

The convolution part and deconvolution part construct a feature pyramid network structure which has been widely used in object detection tasks [6]. Both of them has been proved to detect targets in different scales. This structure is used to avoid scale image into many different sizes and process the network more than one time.

The map generation part is made by two kinds of modules. The details of each kind of module are as follows.

Detection Module: Although the classification and the regression task are different, they still share part of calculations. This part of calculation is combined into one module called "Detection Module". It contains 3 blocks with the same structure.

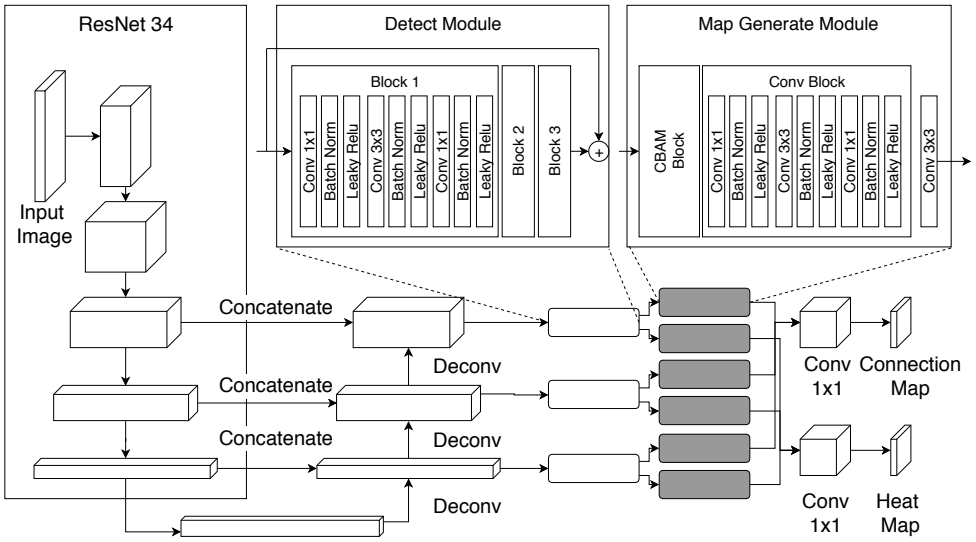

Figure 2. Brief network architecture. It is divided into 4 parts: convolution part, deconvolution part, map generation part and output part. Section 2.2 explains the details of the network

This block is inspired by [9]. Instead of directly doing a 3x3 convolution with full filters, a 1x1 convolution is used to reduce the filter number into a half, then do 3x3 convolution, and use a 1x1 convolution to recover the original filter number. This convolution design has been proved to be faster because of less calculation.

Map Generate Module: This is the module responsible for generating heat map or connection map. this module contains a Convolutional Block Attention Module(CBAM) [10], a convolution block which is the same as the detection module and an output 3x3 convolution layer. Since the different level of feature map is good at detecting different scale of objects, CBAM modules are used to mask out the area which this level is responsible for.

2.3 Loss function and training

Our network's loss function is as follow:

$$Loss = Loss_{classification} + Loss_{target} \quad (2)$$

We do not use cross-entropy as classification loss, but use an L2 loss between network output classes C and target classes $\hat{C}$:

$$Loss_{classification} = \sum_{y=0}^{height} \sum_{x=0}^{width} \sum_{c=0}^{9} (C - \hat{C})^2 \quad (3)$$

For target location loss, we use a masked L2 loss:

$$Loss_{target} = \sum_{j=0}^{height} \sum_{i=0}^{width} 1_{i,j}^{obj} ((x - \hat{x})^2 + (y - \hat{y})^2) \quad (4)$$

This loss function design is inspired by Yolov2 [11] network.

There are 2 kinds of $Loss_{target}$, with a mask or without. Without version

$$Loss'_{target} = \sum_{j=0}^{height} \sum_{i=0}^{width} ((x - \hat{x})^2 + (y - \hat{y})^2) \quad (5)$$

shows much worse accuracy compared with the masked version. The masked version still produces target position info even the area is not a center of any body part. However, it can directly ignore this and focus on the accuracy where have body parts. In our result, applying mask reduce the error into 22%.

We use Adam RMSprop[12] with Nesterov momentum optimizer (Nadam). The learning rate is set at 0.002. Whole training takes about 4 days in one Geforce GTX 1080 Ti. We train this network on the COCO2014 [7] data set.

2.4 Connection fix

After getting the result feature maps, each joint is connected with its parent based on the target position info. This connection progress is made through the following steps:

1. **Joint Detection.** Detect every single joint based on the heat map. It iterates all pixel on the heat map, find out if the background probability is less than a given threshold. If so, process this pixel as follow.
   (a) Find out the maximum probability class, if the probability is more than that threshold, continue.
   (b) Apply a non-maximum suppression to this pixel to find the center of this body part.
   (c) If find the center, read the target offset and calculate the target's position.
   (d) Add this joint into the joint list.
2. **Joint Link.** Link joints based on target position.
   (a) Search potential joints in the target position's local area.
   (b) Check the joint class is link-able.
   (c) If the joint is in $\{Hand, Foot, Elbow, Knee\}$, put this two joints as a potential edge into edge list. Otherwise, find the closest joint and connect them.

Table 1. Processing FPS compared with state-of-art

GPU	Ours	OpenPose
GTX 1080Ti	83	10
GTX 1060	55	5
GTX 1050	38	2.4

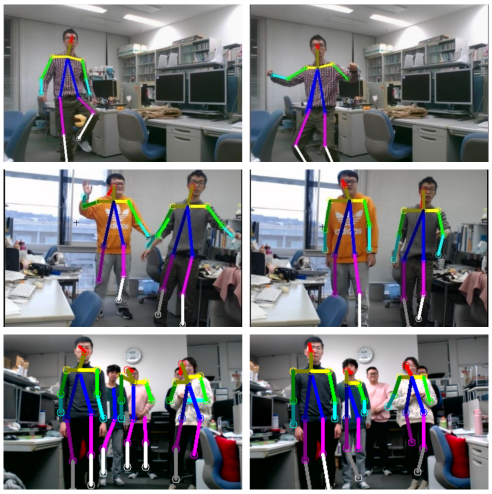

Figure 3. Results of single person and multi-person. These are running on a laptop with GTX 1070 gpu with more than 60 fps.

(d) Each time when it finds a possible edge with minimal error, connected them, and remove edges which contains the same target joint. Repeatedly do so until edge list is empty.

There is a problem if current joint is directily connected to the joint which is nearest to target position. Some joints only have one child. However, sometimes two joints' target position close to the same joint. So firstly this method stores all potential connection pair with errors. Then it connects each joint pair once with the same error. Then it removes all pair with same target joint and force the joint have same target joint to find another lowest error target. Then the "same target joint problem" is solved.

3 Experimental results

3.1 Pose estimation accuracy

This network gets an L2 loss of connection map at 57.6 and an L2 loss of classification at 45.4. It means that there is about 7% loss compared with current state-of-the-art. Our network can detect humans without large collision. If the human size is too large or too small, it do not perform really well. However, for Kinect-like usage, our network performs well. Many detection results can be found at Figure 3.

3.2 Processing time

Network's performance is tested on a set of devices. Our network shows 9 to 10 times faster than current state-of-the-art. The complex report is in Table 1. We compare our result with current state-of-the-art OpenPose solution. The software environment is CUDA 9.2 and Tensorflow-gpu 1.10. This network is implemented with tensorflow and Keras. The number of humans does not affect the speed.

4 Conclusion and future work

In this study, we developed a real-time end-to-end pose estimation network. We proposed the encoding solution, the network architecture and the performance report of our network. The experimental results show our network can archive real-time speed on a lap-top level computer with an affordable accuracy.

For future work, we proposed the solution to increase the position accuracy by adding an inner offset. Using inner offset to calculate the center position in high resolution can archieve higher accuracy without change output resolution. Also, we are considering about rebuild left and right information in "Connection fix" phrase. We just easily decide the side of shoulder and pelvis based on body and head direction which can be easily done by a simple classifier. Then we decide other body parts' side depending on shoulder and pelvis side by connection information.

References

[1] Moeslund, Thomas B., and Erik Granum.: "A survey of computer vision-based human motion capture," *Computer vision and image understanding*, vol.81, no.3, pp.231–268, 2001.

[2] Barris, Sian, and Chris Button.: "A review of vision-based motion analysis in sport," *Sports Medicine*, vol.38, no.12, pp.1025–1043, 2008.

[3] Schall, Gerhard, et al.: "Global pose estimation using multi-sensor fusion for outdoor augmented reality," *2009 8th ieee international symposium on mixed and augmented reality*, pp.153–162, 2009.

[4] He, Kaiming, et al.: "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017.

[5] Cao Z, Simon T, Wei S E, et al.: "Realtime multi-person 2d pose estimation using part affinity fields," *arXiv preprint*, arXiv:1611.08050, 2016.

[6] Lin, Tsung-Yi, et al.: "Feature Pyramid Networks for Object Detection," in *CVPRc*, 2017.

[7] Lin T Y, Maire M, Belongie S, et al.: "Microsoft coco: Common objects in context," *European conference on computer vision*, Springer, Cham, pp.740–755, 2014.

[8] S. Hochreiter and J. Schmidhuber.: "Long short-term memory," *Neural Computation*, vol.9, no.8, pp.1735-1780, 1997.

[9] He, K., Zhang, X., Ren, S., and Sun, J.: "Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp.770–778, 2016.

[10] Woo, Sanghyun, et al.: "Cbam: Convolutional block attention module," in *Proc. of European Conf. on Computer Vision (ECCV)*, 2018.

[11] Redmon J, Farhadi A. "YOLO9000: better, faster, stronger," *arXiv preprint*, 2017.

[12] Kingma, Diederik P., and Jimmy Ba.: "Adam: A method for stochastic optimization.", *arXiv preprint*, arXiv:1412.6980, 2014.