GESTURE RECOGNITION USING SPATIOTEMPORAL DEFORMABLE CONVOLUTIONAL REPRESENTATION.

Lei Shi[2,3], *Yifan Zhang*[2,3]*, *Jing Hu*[1], *Jian Cheng*[2,3], *Hanqing Lu*[2,3]

[1]Power Research Institute of State Gride, Jiangxi Electric Power Company, Nanchang, China
[2]National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
[3]University of Chinese Academy of Sciences

ABSTRACT

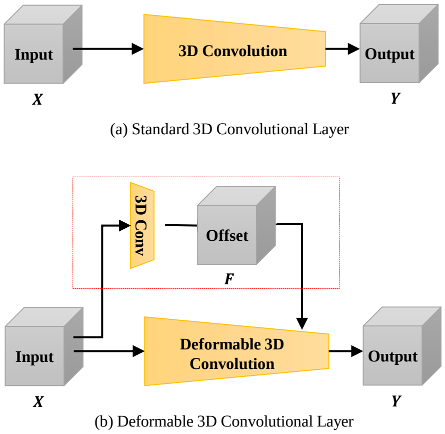
Dynamic gesture recognition, which plays an essential role in human-computer interaction, has been widely investigated but not yet addressed. The interference of the varied and complex background makes the classifier easily be misguided due to the relatively smaller size of the hands and arms compared with the full scenes. In this paper, we address the problem by proposing a novel spatiotemporal deformable convolutional neural network for end-to-end learning. To eliminate the background interference, a light-weight spatiotemporal deformable convolution module is specially designed to augment the spatiotemporal sampling locations of 3D convolution by learning additional offsets according to the preceding feature map. The proposed method is evaluated on two challenging datasets, EgoGesture and Jester, and achieves the state-of-the-art performance on both of the two datasets. The code and trained models will be released for better communication and future work.

Index Terms— Gesture recognition, 3D CNNs

1. INTRODUCTION

Gesture recognition in real-world has drawn significant attention from computer vision community, owing to its broad applications in many areas like VR/AR and human-computer interaction[1, 2, 3, 4, 5, 6]. In the past decades, although many methods have been proposed, dynamic gesture recognition from video sequences is still a challenging problem. Generally speaking, the most discriminative parts in a gesture video clip are hands and arms. However, the region occupied by the hands and arms is relatively small compared to the whole video frame. As a result, the classifier is easily misguided by the varied environments and complex backgrounds in real-world scenes.

To address the issue of the interference from background clutter in gesture recognition, some methods perform hand detection to reduce the effect of the backgrounds [7]. Nevertheless, the additional process for hand detection needs extra computation cost and hand position annotations. Furthermore, the final recognition performance heavily relies on the accuracy of hand detection, which may become the bottleneck of the overall framework. Recently, Cao et al. [6] propose to insert a spatiotemporal transformer module into LSTM to warp the feature map to a canonical view in both spatial and temporal dimensions. It can be trained end-to-end without additional preprocessing. However, based on the learned transform matrix, the transformer can only globally warp the entire feature map which lacks the flexibility for locally geometric transformation. Inspired by Dai et al. [8], a spatiotemporal deformable convolution is proposed in this work to replace the spatiotemporal transformer.



(a) Standard 3D Convolutional Layer

(b) Deformable 3D Convolutional Layer

Fig. 1. Illustration of standard 3D convolutional layer (a) and deformable 3D convolutional layer (b).

The spatiotemporal deformable convolution augments the sampling locations for each convolution step by learning additional offsets in both spatial and temporal dimensions according to the preceding feature map. It enables free-form deformation of a spatiotemporal sampling grid and can generalize various transformations for the shift, scale and rotation. In contrast with Dai et al. [8] which only focus on 2D deformation, our spatiotemporal deformable convolution can not only

[*] Corresponding Author

diversify the sample region and shape to better match the appearance of hands and arms, but also help models pay more attention to the discriminative frames in a video sequence. The spatiotemporal deformable module is light-weight with a small number of parameters for offset learning. It can readily replace the plain 3D convolutional layers and be trained end-to-end with standard back-propagation.

To the best of our knowledge, this is the first work to design spatiotemporal deformable convolution and combine it with 3D CNNs to directly modeling the whole gesture in an end-to-end manner. We demonstrate that our method, which needs only RGB videos without any additional pre-processing such as optical flow extraction, outperforms other methods on two challenge datasets, EgoGesture [9] and Jester [10].

2. RELATED WORK

Gesture recognition has been widely investigated for decades with many works proposed for this issue, ranging from static to dynamic gestures, and from the hand-crafted feature based to convolutional neural network-based methods. Traditional methods focus on designing various hand-crafted features for gesture recognition [1, 2]. However, the performance of these hand-crafted feature based methods is barely satisfactory since it cannot consider all factors at the same time.

Recently, deep leaning based methods have shown the big success [11, 5]. On the 2017 ChaLearn LAP Large-scale Isolated Gesture Recognition Challenge, the C3D [12] based methods have demonstrated the powerful spatiotemporal feature representation ability and achieved remarkable performance [13]. However, compared with the successful models employed in image classification area, e.g. ResNet [14], Inceptions [15], C3D is relatively shallow and its capacity is limited.

3. METHOD

3D convolution can be seen as the weighted sum over a regular 3D sampling grid with weight W. For each location p_i on the input feature map X, the value of corresponding location p_o on the output feature map Y can be calculated as Equation 1.

$$Y(\hat{p_o}) = \sum_{\hat{p_n} \in \mathcal{V}} W(\hat{p_n}) \cdot X(\hat{p_i} + \hat{p_n}) \tag{1}$$

where the hat symbol indicates that the variable is integral. $\hat{p} = (\hat{p_x}, \hat{p_y}, \hat{p_z})$ is the 3D vector representing the 3D points in the feature map. $\hat{p_n}$ enumerates the locations in 3D sampling grid $\mathcal{V}$, which is decided by the kernel size and the dilation value of convolution. For example, if the kernel size is 3 and the dilation value is 1, the $\mathcal{V}$ will be $\{(-1,-1,-1), (-1,-1,0), (-1,-1,1), \cdots, (1,1,1)\}$. A simple 3D convolutional layer is shown in Figure 1(a), where the output size is assumed the same as input.

Rather than using the regular sampling grid, deformable 3D convolution learns 3D offset $\Delta p_{i,n}$ to deform the conventional sampling grid as Equation 2.

$$Y(\hat{p_o}) = \sum_{\hat{p_n} \in \mathcal{V}} W(\hat{p_n}) \cdot X(\hat{p_i} + \hat{p_n} + \Delta p_{i,n}) \tag{2}$$

where $\Delta p_{i,n}$ is individual for each convolution step according to the $\hat{p_i}$ and $\hat{p_n}$. As illustrated in Figure 1(b), the offset map F is obtained in an additional branch inside the dashed box. It is learned by a carefully designed 3D convolutional layer, whose kernel size is set to $3 \times 3 \times 3$ with pad 1 and stride 1. It ensures the spatiotemporal resolution of F the same as X. The number of kernels is designed as $3NC_X$, where 3 indicates three offset directions (one temporal dimension and two spatial dimensions), C_X is the number of input channels and N is the volume of $\mathcal{V}$ (e.g., $N = 27$ for $3 \times 3 \times 3$ kernel). Because the C_X is large sometimes (e.g., the last convolutional layer of ResNeXt has 2048 channels), we apply grouped deformable convolution which divides the C_X into G groups, and each group shares the same offsets. The resulting F has $3NG$ channels. If G is set to a small number, the number of parameters needed to learn can be greatly reduced. Finally, the learned offsets are used in deformable convolution to augment the sampling locations.

Note that the offset learned by convolutional layer is typically fractional. To make the architecture differentiable, trilinear interpolation is applied to get the final output. As shown in Figure 2, trilinear interpolation is the extension of linear interpolation and bilinear interpolation. It calculates the target value according to the surrounding points whose distance to the target is less than 1 as Equation 3

$$\begin{aligned} X(p) = \sum_{\hat{q}} X(\hat{q}) \cdot [(1 - |\hat{q^x} - p^x|)]^+ \\ \cdot [(1 - |\hat{q^y} - p^y|)]^+ [(1 - |\hat{q^z} - p^z|)]^+ \end{aligned} \tag{3}$$

where $[x]^+ = max(0, x)$. X is the input feature map. $p = (p^x, p^y, p^z)$ represents the fractional sampling position after adding the offset and $\hat{q}$ represent the surrounding integral points of p. $X(p)$ is calculated by weighted sum over $X(\hat{q})$, where weights are determined by the distance between p and $\hat{q}$.

During training, both the convolutional kernels for generating the output features and the offsets are learned simultaneously. The gradients can be back-propagated through Equation 2 and Equation 3, which is formulated as Equation 4, Equation 5 and Equation 6.

$$\begin{aligned} \frac{\partial Y(\hat{p_o})}{\partial X(\hat{q})} = \sum_{\hat{p_n} \in \mathcal{V}} W(\hat{p_n}) \cdot [(1 - |\hat{q^x} - p^x|)]^+ \\ \cdot [(1 - |\hat{q^y} - p^y|)]^+ [(1 - |\hat{q^z} - p^z|)]^+ \end{aligned} \tag{4}$$

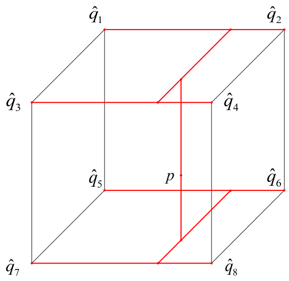


Fig. 2. Illustration of trilinear interpolation. p is the original point whose coordinates are fractional. Its value is calculated by weighted sum of $\hat{q}_i, i = 1, 2, \cdots, 8$, which are the surrounding integral points.

$$\begin{aligned}\frac{\partial Y(\hat{p_o})}{\partial \Delta p^x_{i,n}} =& W(\hat{p_n}) \sum_{\hat{q}} X(\hat{q}) \cdot sign(\hat{q^x} - p^x_i) \\ & \cdot [(1 - |\hat{q^y} - p^y_i|)]^+ [(1 - |\hat{q^z} - p^z_i|)]^+ \end{aligned} \quad (5)$$

$$\frac{\partial Y(\hat{p_o})}{\partial W(\hat{p_n})} = X(\hat{p_i} + \hat{p_n} + \Delta p_{i,n}) \quad (6)$$

where the definitions of symbols are same as Equation 2 and Equation 3. In Equation 5, we only list the partial derivative of the output feature map with respect to the offset along the x dimension, The formulation along the y and z dimensions can be deduced accordingly.

4. EXPERIMENTS

4.1. Datasets

EgoGesture [9] is a large-scale multi-modal dataset for egocentric hand gesture recognition, which designs 83 gestures for interaction with wearable devices. It contains 2081 RGB-D videos, 24161 gesture samples and 2953224 frames from 50 distinct subjects in 6 scenes. The average length of isolated gesture videos is 38 frames.

Jester [10] is a recent video dataset for hand gesture recognition, which contains 27 kinds of predefined hand gestures performed in front of a camera. It has totally 148092 gesture samples extracted from the original videos at 12 frames per second. The samples are officially split into three sets, 118562 samples for training, 14787 samples for validation and 14743 samples for testing without providing labels. The average length of the video is 35 frames.

4.2. Experiments on EgoGesture Dataset

4.2.1. Training Details

Since the average length of video samples in EgoGesture is 38 frames, we use 32 frames with 112 × 112 pixels as a clip to balance the GPU memory and information contained in each clip. For training, we first randomly sample 32 frames and sort them in temporal order, then perform random cropping for each frame to 224 × 224 and finally resize them to 112 × 112 pixels. If the sample is shorter than 32 frames, we expand it by duplicating every frame (e.g., given xy, it will be extended to $xxyy$). This process will be executed recurrently until the sample is longer than 32 frames. Mean-subtraction and std-division are performed for each frame. We use stochastic gradient descent (SGD) with Nesterov momentum (0.9) on 4 GPUs (NVIDIA TITAN XP) for training. Weight decay is set to 0.0005, and initial learning rate is set to 0.001. The learning rate is multiplied by 0.1 at the 20_{th} and 30_{th} epoch. The training process is ended at the 40_{th} epoch. When testing, we uniformly sample 32 frames, then perform central cropping and resizing for each frame.

Table 1. The performance of different 3D CNNs. (D) represents the models embedded with deformable 3D convolution

Model	Depth	Acc	#Params
ResNet3D-18	18	90.68	31.8M
ResNet3D-18 (D)	18	91.66	35.5M
ResNet3D-34	34	91.86	60.8M
ResNet3D-34 (D)	34	92.17	65.8M
ResNet3D-101	101	94.01	81.7M
ResNet3D-101 (D)	101	94.20	85.5M
ResNeXt3D-101	101	94.18	45.8M
ResNeXt3D-101 (D)	101	94.72	52.2M
Inception3D-V1	22	89.61	12.0M
Inception3D-V1 (D)	22	90.92	12.5M
InceptionResNet3D-V2	190	92.27	111.5M
InceptionResNet3D-V2 (D)	190	92.73	118.0M

4.2.2. Spatiotemporal deformable convolution

The C3D used in traditonal methods is relatively shallow compared with the successful models used in image classification. In this work we propose three types of deeper and more powerful models to evaluate the proposed method. In detail, the ResNet3D-18, ResNet3D-34, ResNet3D-101, ResNext3D-101, Inception3D-V1 and InceptionResNet3D-v2 are used for evaluation. All the above models are pretrained on the Kinetics dataset and finetuned on the EgoGesture dataset.

As introduced in Section 3, we plug our proposed spatiotemporal deformable convolution modules in above models to test the effectiveness. Although the plain convolutional layers can be substituted by deformable version easily, it is not sensible to replace them all.

According to the experiments, we replace top 2 layers for ResNet3D-18, top 3 layers for ResNet3D-34 and ResNext3D-101, top 2 layers for Inception3D-V1 and top 11 layers for InceptionResNet3D-V2. The details can be found in the released code. All the results are shown in Table 1, where adding the deformable module brings a consistent increase in accuracy. This fully illustrates the effectiveness of the proposed spatiotemporal deformable convolution modules. Besides, there is only a limited increase in the amount of parameters.

4.2.3. *Compared with the state-of-the-art methods*

The proposed model is compared with previous state-of-the-art methods using RGB videos as input. Table 2 shows the final recognition accuracy of these methods, where our model achieves the best performance compared with other methods with a large margin.

Table 2. Validation Accuracy on EgoGesture.

Methods	Accuracy
iDT-FV [16]	64.30
VGG16+LSTM [17]	74.70
C3D+SVM [12]	86.40
C3D+RSTTM [6]	89.30
Deformable 3D ResNeXt	94.72

4.3. Experiments on Jester Dataset

The Jester dataset is split into training, validation and testing sets according to the official provided .csv files. Training details are similar with the Section 4.2.1.

4.3.1. *Analysis of the spatiotemporal deformable convolution*

To better emphasize the effect of spatiotemporal deformable convolution, we use part of the gestures as input to evaluate the performance of ResNeXt3D-101 with or without embedding deformable convolution module. In particular, we cut out the 32-frame clips from the beginning, middle and end of videos as input, and crop the left-top, central and right-bottom corner of frames respectively. The results are shown in Table 3, from which we can see the performance gain of using deformable convolution increases when the gestures are incomplete or not in the center. It is intuitive because the offsets learned by deformable convolution can help the model find the right things.

Table 3. Accuracy of model using a corner of videos as input. DC stands for the deformable convolution

Position	w/o DC	w/ DC	Gain
Middle,center-crop	95.6	96.1	0.5
Beginning,left-top-crop	91.5	92.3	0.8
End,right-bottom-crop	81.1	82.6	1.5

Two successfully recognized samples in Jester are visualized in Figure 3. Sample locations of one deformable convolutional step are plotted with red points. It can be seen that sample locations are deformed in both spatial and temporal dimensions to match the video content better.

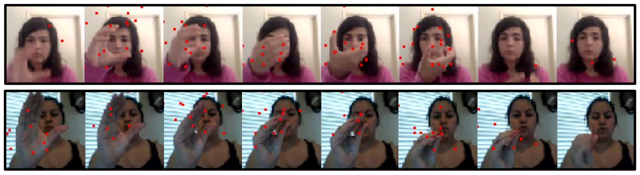


Fig. 3. Visualization of offsets learned by deformable 3D convolution. The labels of two samples are “Turning Hand Clockwise” and “Zooming In With Full Hand”.

4.3.2. *Compared with the state-of-the-art methods*

Our models are evaluated on the test set of Jester. Table 4 shows the final results compared with other methods listed in the leaderboard, where our model achieves the best performance until the submission time. It can be seen that deep 3D CNN shows excellent capacity for video representation, and embedding the deformable convolutional layers brings additional improvement.

Table 4. Test accuracy in Jester dataset.

Methods	Accuracy
20BN’s Jester System [10]	82.34
TRN [18]	94.78
Motion Fused Frames [19]	96.28
ResNeXt3D-101	95.68
Deformable ResNeXt3D-101	96.60

5. CONCLUSION

In this work, a spatiotemporal deformable convolution module is specially designed to augment the sampling locations of 3D convolution for dynamic gesture recognition. It helps models pay attention to discriminative parts of the video sequence in both spatial and temporal dimensions. The final model achieves state-of-the-art performance on two challenging datasets, EgoGesture and Jester.

Acknowledgement This work was supported in part by the National Natural Science Foundation of China under Grant 61572500, 61876182 and 61872364, and in part by the State Grid Corporation Science and Technology Project.

6. REFERENCES

[1] E. Ohn-Bar and M. M. Trivedi, "Hand Gesture Recognition in Real Time for Automotive Interfaces: A Multimodal Vision-Based Approach and Evaluations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 6, pp. 2368–2377, 2014.

[2] Jun Wan, Guodong Guo, and Stan Z. Li, "Explore efficient local features from RGB-D data for one-shot learning gesture recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 8, pp. 1626–1639, 2016.

[3] Lorenzo Baraldi, Francesco Paci, Giuseppe Serra, Luca Benini, and Rita Cucchiara, "Gesture recognition in ego-centric videos using dense trajectories and hand segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 688–693.

[4] Lionel Pigou, Aron Van Den Oord, Sander Dieleman, Mieke Van Herreweghe, and Joni Dambre, "Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video," *International Journal of Computer Vision*, vol. 126, no. 2-4, pp. 430–439, 2018.

[5] G. Zhu, L. Zhang, P. Shen, and J. Song, "Multimodal Gesture Recognition Using 3-D Convolution and Convolutional LSTM," *IEEE Access*, vol. 5, pp. 4517–4524, 2017.

[6] Congqi Cao, Yifan Zhang, Yi Wu, Hanqing Lu, and Jian Cheng, "Egocentric Gesture Recognition Using Recurrent 3d Convolutional Neural Networks With Spatiotemporal Transformer Modules," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[7] Sven Bambach, Stefan Lee, David J. Crandall, and Chen Yu, "Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1949–1957.

[8] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei, "Deformable Convolutional Networks," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[9] Yifan Zhang, Congqi Cao, Jian Cheng, and Hanqing Lu, "EgoGesture: A New Dataset and Benchmark for Egocentric Hand Gesture Recognition," *IEEE Transactions on Multimedia*, pp. 1–1, 2018.

[10] TwentyBn, "The 20bn-JESTER Dataset," *https://www.twentybn.com/datasets/jester*, 2017.

[11] Pichao Wang, Wanqing Li, Song Liu, Zhimin Gao, Chang Tang, and Philip Ogunbona, "Large-scale Isolated Gesture Recognition using Convolutional Neural Networks," *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 7–12, 2016.

[12] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri, "Learning Spatiotemporal Features With 3d Convolutional Networks," in *The IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, bibtex: Tran_2015_ICCV.

[13] Jun Wan, Sergio Escalera, Gholamreza Anbarjafari, Hugo Jair Escalante, Xavier Baro, Isabelle Guyon, Meysam Madadi, Juri Allik, Jelena Gorbova, Chi Lin, and Yiliang Xie, "Results and Analysis of ChaLearn LAP Multi-Modal Isolated and Continuous Gesture Recognition, and Real Versus Fake Expressed Emotions Challenges," in *The IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct. 2017.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going Deeper With Convolutions," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[16] Heng Wang and Cordelia Schmid, "Action Recognition with Improved Trajectories," in *The IEEE International Conference on Computer Vision (ICCV)*, Dec. 2013.

[17] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625–2634.

[18] Bolei Zhou, Alex Andonian, and Antonio Torralba, "Temporal Relational Reasoning in Videos," *arXiv:1711.08496 [cs]*, Nov. 2017.

[19] Okan Kopuklu, Neslihan Kose, and Gerhard Rigoll, "Motion Fused Frames: Data Level Fusion Strategy for Hand Gesture Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 2103–2111.