Pattern Recognition 107 (2020) 107416

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

# Gesture recognition based on deep deformable 3D convolutional neural networks

Yifan Zhang[a,b,*], Lei Shi[a,b], Yi Wu[c], Ke Cheng[a,b], Jian Cheng[a,b,d], Hanqing Lu[a,b]

[a] NLPR & AIRIA, Institute of Automation, Chinese Academy of Sciences, China
[b] School of Artificial Intelligence, University of Chinese Academy of Sciences, China
[c] Wormpex AI Research
[d] CAS Center for Excellence in Brain Science and Intelligence Technology, China

ARTICLE INFO

*Article history:*
Received 31 May 2019
Revised 24 April 2020
Accepted 29 April 2020
Available online 24 June 2020

*Keywords:*
Gesture recognition
Spatiotemporal deformable convolution
Spatiotemporal convolutional neural network

ABSTRACT

Dynamic gesture recognition, which plays an essential role in human-computer interaction, has been widely investigated but not yet fully addressed. The challenge mainly lies in three folders: 1) to model both of the spatial appearance and the temporal evolution simultaneously; 2) to address the interference from the varied and complex background; 3) the requirement of real-time processing. In this paper, we address the above challenges by proposing a novel deep deformable 3D convolutional neural network for end-to-end learning, which not only gains impressive accuracy in challenging datasets but also can meet the requirement of the real-time processing. We propose three types of very deep 3D CNNs for gesture recognition, which can directly model the spatiotemporal information with their inherent hierarchical structure. To eliminate the background interference, a light-weight spatiotemporal deformable convolutional module is specially designed to augment the spatiotemporal sampling locations of the 3D convolution by learning additional offsets according to the preceding feature map. It can not only diversify the shape of the convolution kernel to better fit the appearance of the hands and arms, but also help the models pay more attention to the discriminative frames in the video sequence. The proposed method is evaluated on three challenging datasets, EgoGesture, Jester and Chalearn-IsoGD, and achieves the state-of-the-art performance on all of them. Our model ranked first on Jester's official leader-board until the submission time. The code and the trained models are released for better communication and future works[1].

© 2020 Elsevier Ltd. All rights reserved.

## 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]. In the past decades, although many methods have been proposed, dynamic gesture recognition from video sequence is still a challenging problem. The difficulties mainly lie in three folders:

1) The most discriminative parts in a gesture video clip are the hands and arms. The area of the region they occupied 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. 2) Different from action recognition, background context can be hardly employed to facilitate gesture recognition. Motion information plays a more important role in gesture recognition than action recognition. The model needs to distinguish the fine-grained difference in the movement of hands. In a widely used action dataset UCF-101 [3], action categories can usually be identified from a still image as illustrated in Fig. 1-(a), because the background context, such as surrounding scenes and interacted objects, can provide enough cues for recognition. However, it is difficult to distinguish dynamic gestures using a still image. For example, “moving hand left” versus “moving hand right” cannot be classified with only one frame, because both of them can be turned into each other by just reversing the temporal order of the image sequence as illustrated in Fig. 1-(b). 3) An applicable gesture recognition system requires to real-timely process the video stream. However, the most popular deep neural network methods for video classification, i.e., two-stream-based deep neural network [4], cannot be executed in real-time. In detail, the optical flow, which is responsible for extracting temporal information in

[*] Corresponding author.
*E-mail address:* yfzhang@nlpr.ia.ac.cn (Y. Zhang).

[1] https://github.com/lshiwjx/deform_conv3d_pytorch_op

https://doi.org/10.1016/j.patcog.2020.107416
0031-3203/© 2020 Elsevier Ltd. All rights reserved.

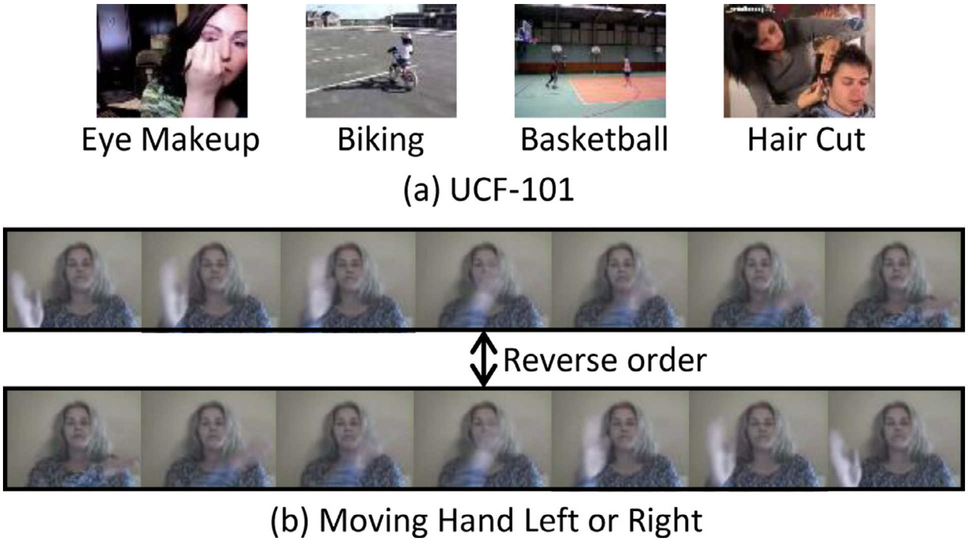


**Fig. 1.** (a) Images randomly selected in the UCF-101 [3] action dataset. (b) A gesture image sequence in the Jester dataset and its duplicate by reversing the order of the sequence.

the two-stream framework, has to be obtained off-line due to its intensive computation.

To address the issue of the interference from background clutter in gesture recognition, certain methods perform hand detection to reduce the effect of the backgrounds [5]. 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 the LSTM to warp the feature map to a canonical view in both the 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. [7], a spatiotemporal deformable convolution is proposed in this work to replace the spatiotemporal transformer. Conventional convolution can be seen as a weighted sum over a sampling grid in the input feature map which is fixed to be a rectangle. The spatiotemporal deformable convolution augments the sampling locations for each convolutional 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.[7] which only focus on the 2D deformation, our spatiotemporal deformable convolution can not only 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 the standard back-propagation.

As for the requirement of modeling both the spatial and temporal information simultaneously and running gesture-recognition system real-timely, the 3D convolutional neural networks (CNNs) is a suitable choice. The hierarchical architecture of the 3D CNN is intuitively suitable for spatiotemporal modeling, which can capture the appearance and motion simultaneously from the low-level details to the high-level semantics. Besides, due to the ability to be processed in parallel, the 3D CNNs are faster during training and inference compared with the two-stream-based methods and can be executed in real-time.

However, the traditional 3D CNN models are mainly based on the C3D [8] structure, which has only eight convolutional layers. It is shallower than most of the successful 2D models used in image classification domain, resulting in limited representation capacity. A valid question is why not build deeper 3D CNN models for gesture recognition. In this work, we propose three types of deep 3D CNN models for gesture recognition based on three 2D CNN models succeeded in image classification domain, namely, ResNet [9], ResNeXt [10] and Inceptions [11]. Since the model becomes deeper, it becomes harder to train due to a large number of parameters. In this work, certain practical skills, which are proved to be important for training very deep 3D CNNs, are also proposed and validated, including using the models pre-trained on the large-scale action recognition datasets, performing data augmentation in both the spatial and temporal dimensions. Using these skills, our 3D deep models exhibit evident improvement compared with the traditional C3D.

To the best of our knowledge, This is the first work to propose the spatiotemporal deformable convolution and combine it with the very deep 3D CNNs to directly model the whole gestures in an end-to-end manner. We demonstrate that our method, which can perform inference in real-time and needs only RGB videos without any additional pre-processing such as optical flow extraction, outperforms other methods on three challenge datasets, EgoGesture [12], Jester [13] and Chalearn-IsoGD [14]. All of the proposed three types of very deep 3D CNN models can be improved by inserting our spatiotemporal deformable convolution module.

The main contributions of our work include:

- We articulate the differences between gesture recognition and general action recognition in three aspects: 1) Background context can provide useful knowledge for action recognition. However, it is useless and even harmful in gesture recognition. 2) The motion of hands and arms are more crucial part in gesture recognition than in action recognition. 3) Gesture recognition is more sensitive to the computational complexity than action recognition as it is mainly used in a real-time human computer interaction system. Therefore, we think that the gesture recognition is a fine-grained classification task. The model needs to focus on the spatial appearance and temporal motion of the hands and arms.
- We design a light-weight spatiotemporal deformable convolution module which enables free-form deformation of the sam-

pling grid for a convolutional kernel on both spatial and temporal dimensions. The proposed method achieves state-of-the-art performance on three challenging datasets, EgoGesture, Jester and Chalearn-IsoGD.

- We provide an insight that the benefit of plugging the spatiotemporal deformable convolution module to the higher level layer is larger than to the lower level layer.
- We propose a data spatiotemporal augmentation method to randomly generate diverse data samples in a spatiotemporal cube, which is proved to be effective for model training.

## 2. Related work

### 2.1. Gesture recognition

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 methods to CNN-based methods. Traditional methods focus on designing various hand-crafted features for gesture recognition. Ohn-Bar et al. [15] evaluate a set of common spatiotemporal descriptors and employ them for an in-vehicle vision-based gesture recognition system. Wan et al. [16] propose a method named mixed features around sparse keypoints (MFSK) to extract spatiotemporal features and perform one-shot learning gesture recognition from RGB-D data. Tand et al. [17] propose to combine templates and velocity information to spot the beginning and ending points in hand gesture trajectories. Different weights are assigned to feature sequences based on the positions of corner points in the arbitrary trajectories. However, the expression capacity of these hand-crafted features is still limited. Some of the features are computational expensive.

Recently, deep-learning-based methods have achieve tremendous success in many computer vision tasks such as image detection [18], pose estimation [19] and action recognition [20]. It also exhibit excellent performance in the field of gesture recognition. Wang et al. [21] propose three simple representations of depth sequences, i.e., Dynamic Depth Image (DDI), Dynamic Depth Normal Image (DDNI) and Dynamic Depth Motion Normal Images (DDMNI), and fine-tune the existing ConvNets models trained on image data for classification of depth sequences. To better utilize the temporal information, C3D [8] is employed for dynamic gesture recognition, since it can directly model the spatial and temporal information with the 3D Convolutional kernel. On the 2017 ChaLearn LAP Large-scale Isolated Gesture Recognition Challenge, the C3D-based methods have demonstrated the powerful spatiotemporal feature representation ability and achieved remarkable performance. Li et al. [22] enhance the traditional C3D model with the help of the saliency theory, which is employed to alleviate the interference of gesture-irrelevant factors. However, compared with the successful models employed in image classification area, e.g. ResNet [9], Inceptions [11], C3D is relatively shallow and its capacity is limited.

Many works leverage the LSTM for long-term sequence modeling. Zhu et al. [23] combine the C3D and LSTM to model the gesture. The Spatial Pyramid Pooling (SPP) is employed to normalize the spatiotemporal features for final classification. Pigou et al. [1] demonstrate that the temporal information is more important for gesture recognition compared with other tasks such as video classification. They further show that temporal pooling and LSTM is crucial for this task and lead to significant improvements. However, RNN-based method is hard to train due to the vanishing and the exploding gradient problems [24]. Besides, when using LSTM for gesture recognition, the input of LSTM is high-level representation extracted by CNN, which may cause the neglect of low-level temporal information. In this sense, directly model the whole sequence with 3D CNNs is more suitable.

Employing optical flow is another rewarding method for motion encoding. It is widely used for action recognition [4,20] and is leveraged as a modal input for gesture recognition [25]. However, there is a great computational preprocessing complexity for calculating the optical flow, which cannot meet the requirement of real-time execution. Thus, we do not consider this kind of methods in this work.

Recent works [26,27] focus on the multi-modal fusion. Duan et al. [26] fuse the RGB stream, the depth steam and the optical flow field of the RGB/depth videos in an unified framework, where the SoftMax scores are added to get the final prediction. They additionally propose to use the saliency information to help modeling the human motions. Chang [27] proposes to employ the skeletal joint-based features and the appearance information near the active hand in an RGB image to capture the detailed motion of fingers. In this work, since our motivation is to investigate the effectiveness of the spatiotemporal deformable module, we only consider the RGB modality.

Compared with action recognition or scene recognition where the background can provide useful context knowledge, it becomes an obstacle for gesture recognition. To avoid the interference of the background clutter, many methods perform the spatial-temporal hand segment process [15]. Using additional detection process brings additional computational cost. Since the recognition performance heavily relies on the accuracy of detection, it always becomes the bottleneck of the system. Recently, Cao et al. [6] propose a recurrent spatiotemporal transformer module, which can learn a 3D homography transformation matrix in the training process and actively transform the 3D feature maps into a canonical view. Instead of transforming all of the pixels with a uniform manner, we propose a spatiotemporal deformable convolution module in this paper to augment each sampling location individually for spatiotemporal convolution, which brings more flexibility while maintaining the end-to-end learning capability.

### 2.2. 3D CNNs

3D CNNs have been widely used in the action recognition field. The 3D CNNs [28] use 3D convolutional kernels which can directly extract the spatiotemporal features form the low levels to the high levels. Because the 3D CNNs have much more parameters than the 2D CNNs, it is more difficult to train and the performance is also limited. Recently, due to the emergence of the large-scale video dataset [29] and using the pre-trained models, the 3D-CNN-based methods have shown better performance than the 2D-CNN-based methods. The earliest 3D convolutional network is C3D [8], which is designed based on the VGG ConvNet and has only 8 convolutional layers. After that, many deeper 3D convolutional networks are designed based on the 2D CNNs that are successful used in the image classification field. For example, I3D [29] is designed based on the Inception [30] model and Res3D [31] is designed based on the ResNet [32] model. Besides, S3D [33] proposes to replace some of the 3D convolutional layers to the 2D convolutional layers to save the computation while keeping the accuracy unchanged. In [34], the 3 × 3 × 3 convolutions are replaced with one 1 × 3 × 3 convolutional filters on the spatial domain and one 3 × 1 × 1 convolutions on the temporal domain.

### 2.3. Deformable convolution

Conventional convolution samples the input feature map at fixed locations, which lacks internal mechanisms to handle the geometric transformations. Many works have been proposed to solve this problem. Some methods focus on the modification of input

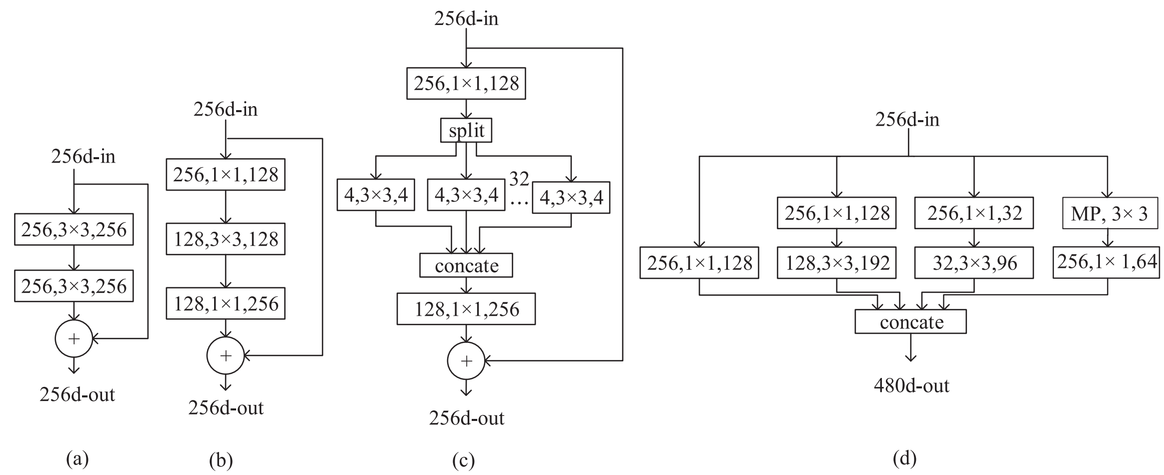


**Fig. 2.** Illustrations of the basic blocks for three types of models explored in this work: (a). ResNet Basic Block. (b). ResNet Bottleneck Block. (c). ResNeXt Bottle Block. (d). Inception Block. MP denotes the max-pooling layer.

feature map. Residual Attention Network [35] learns an attention mask in a residual branch, which is multiplied to the input feature map to emphasize the important contents of the feature map. Spatial Transformer Network [36] learns a transform matrix to warp the feature map to the desired form.

Other methods focus on augmenting the sampling locations of convolution. Dilated convolution [37] moving the sampling location father apart by increasing kernel's stride to be larger, which can expand the receptive field of convolution while retaining the same computing complexity. Deformable convolution [7] also augments the sampling grid by learning 2D offsets for each of the sampling locations according to the preceding feature map but has more diversity and flexibility compared with the above methods.

## 3. Method

In this section, we propose three types of deep 3D CNN models for gesture recognition based on three 2D models succeeded in image classification domain.

### 3.1. Going deeper with 3D CNNs

Conventional C3D [8] is a VGG-like model, which has 8 convolutional layers, 5 max-pooling layers, and 2 fully-connected layers, followed by a SoftMax layer. It is relatively shallow but has 79M parameters. More than half of the parameters come from the fully-connected layers, which have been proved redundancy and unfriendly for training [9,11]. We modify the architecture of C3D by removing the fully-connected layers and adding a 3D global-average-pooling layer, which directly performs the spatiotemporal average for feature maps of last convolutional layer. We also apply batch normalization after each of the convolutional layers, which has been proved to be practical [30].

However, the modified C3D is still shallow, and the deeper models are needed for better feature extraction. As there are already various successful models employed in image classification domain, we empirically select three types of typical 2D CNNs and build the deeper 3D CNNs based on these models. In particular, ResNet, ResNeXt and Inception are explored in the experiment. Among these models, ResNet [9] uses the residual block (Fig. 2(a)) to build network. Each residual block provides shortcut connections that skip one or more layers, whose outputs are finally added with the output of skipped layers. To build the deeper model, the residual bottleneck block is applied, which add 1 × 1 blocks to reduce the channel dimension (Fig. 2(b)). The final network is built by heaping numbers of residual blocks. ResNeXt [10] widens the ResNet along channel dimension to increase the capacity of the model. Meanwhile, the convolutional kernels are divided into groups, and each group is corresponded to a number of channels (Fig. 2(c)). Inception-based networks [11,30,38] exploit the split-transform-merge strategy to design the network. An example of a basic inception block is shown in (Fig. 2(d)), which employs convolutions with different kernel size to capture details at various scales. The outputs of each of the convolutional branches are concatenated in the end, and the overall network is the stack of these blocks. In this work, the ResNet-18, ResNet-34, ResNet-101 [9], ResNeXt-101 [10], Inception-V1 [11] and Inception-ResNet-V2 [38] are selected for evaluation.

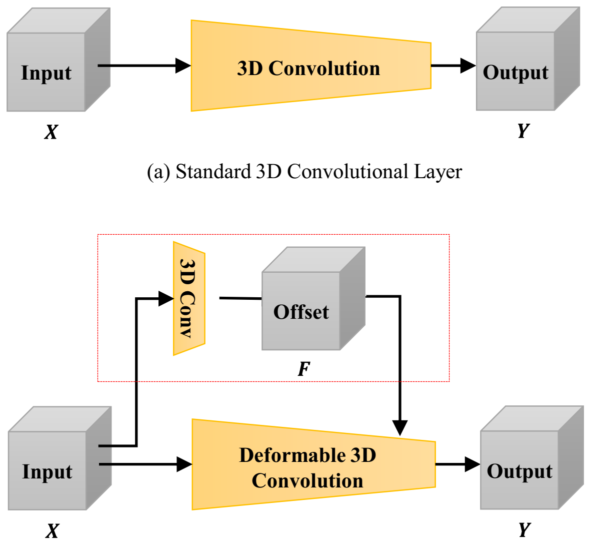
To extend the selected architectures to 3D versions, we expand the kernel size of all the convolutional and pooling layers from $k \times k$ to $k \times k \times k$. Because the input length along the temporal dimension is shorter than those along spatial dimensions, some details of the model are modified to avoid down-sampling in temporal dimension too early. For ResNet3D-18, ResNet3D-34, ResNet3D-101, ResNeXt3D-101 and Inception-ResNet3D-V2, stride of the first convolutional layer and max-pooling layer is changed from (2, 2, 2) to (1, 2, 2) which corresponds to the $(z, y, x)$ dimensions of the feature map, where $z$ is the temporal dimension. Similarly, the stride of the first two max-pooling layers in Inception3D-V1 is also modified to (1,2,2). The depth and number of parameters of these extended models are listed in Table 3, where the deepest model (Inception-ResNet3D-V2) has 190 layers.

### 3.2. Spatiotemporal deformable convolution

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 Eq. (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



(a) Standard 3D Convolutional Layer

(b) Deformable 3D Convolutional Layer

**Fig. 3.** Illustrations of the standard 3D convolutional layer and the deformable 3D convolutional layer.

convolutional layer is shown in Fig. 3(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 Eq. (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 $\hat{p}_i$ and $\hat{p}_n$. As illustrated in Fig. 3(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 × 3 × 3 with pad 1 and stride 1. It keeps 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 the sampling grid $\mathcal{V}$ (e.g., $N = 27$ for 3 × 3 × 3 kernel). Because $C_X$ may be quite large in certain cases (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 Fig. 4, 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 Eq. (3)

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

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 $\hat{q}$.

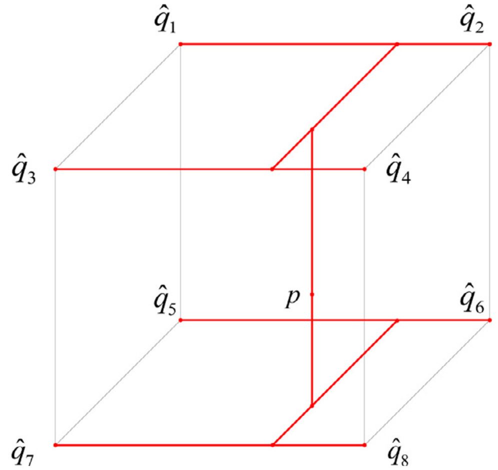


**Fig. 4.** Illustration of the trilinear interpolation. $p$ is the original point whose coordinates are fractional. Its value is calculated by weighted sum of $\hat{q}_i, i = 1, 2, \cdots, 8$, which are the surrounding integral points.

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

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

$$\frac{\partial Y(\hat{p}_o)}{\partial \Delta p^x_{i,n}} = W(\hat{p}_n) \sum_{\hat{q}} X(\hat{q}) \cdot sign(\hat{q}^x - p^x_i) \cdot [(1 - |\hat{q}^y - p^y_i|)]^+ [(1 - |\hat{q}^z - p^z_i|)]^+ \tag{5}$$

$$\frac{\partial Y(\hat{p}_o)}{\partial W(\hat{p}_n)} = X(\hat{p}_i + \hat{p}_n + \Delta p_{i,n}) \tag{6}$$

Where the definitions of symbols are same as Eq. (2) and Eq. (3). In Eq. (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

The experiments are conducted on three publicly available datasets: EgoGesture [12], Jester [13] and Chalearn-IsoGD [14].

Most of the gestures designed in these datasets are challenging to distinguish, which highly depend on temporal relations between frames rather than appearance, such as “Swiping Left” versus “Swiping Right”, “Turning Hand Clockwise” versus “Turning Hand Counterclockwise”. They are well adapted for testing our methods due to their higher requirements for spatiotemporal feature expression abilities.

#### 4.1.1. EgoGesture

EgoGesture 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. Each video has more than one gestures, and most of the gesture samples are less than 3 seconds. The average length of isolated gesture videos is 38 frames.

### 4.1.2. Jester

Jester 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 148,092 gesture samples extracted from the original videos at 12 frames per second. The samples are officially split into three sets, 118,562 samples for training, 14,787 samples for validation and 14,743 samples for testing without providing labels. The average length of the video is 35 frames.

### 4.1.3. Chalearn

Chalearn-IsoGD is a large dataset which contains 249 kinds of gestures performed by 21 different individuals. There are totally 35,787 samples for training and 5784 samples for validation. It has both the RGB and the Depth modality. In this work, we only use the RGB videos.

## 4.2. Training details

We use 32 frames whose size is 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 the temporal order. 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. Then we perform random cropping for each frame with the cropping ratio randomly selected from 0.7 − 1. The cropped frames are finally resized to 112 × 112 pixels. Mean-subtraction and std-division are performed for each frame. When testing, we uniformly sample 32 frames and center-crop them to 224 × 224 pixels, which is finally resized to 112 × 112 pixels.

We use the stochastic gradient descent (SGD) with Nesterov momentum (0.9) as the optimizer. We use 4 GPUs (NVIDIA TITAN XP) for training. Batch size is 64 for ResNet3D-18, ResNet3D-34, ResNeXt3D-101 and Inception3D-V1, and is 32 for ResNet3D-101 and Inception-ResNet3D-V2. Cross-entropy is selected as the loss function to back-propagate gradients. 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 plugging the deformable convolutional module, the weights and the biases of the convolution, which produce the offset field, is initialized to 0. The model is first pretrained in the target dataset while the deformable convolutional module is freezed. Then both the basic model and the deformable convolutional module are finetuned with the learning rate set as 0.0001.

## 4.3. Experiments on EgoGesture Dataset

We randomly split the EgoGesture dataset into training (80%) and testing (20%) sets according to their subjects. Each video sequence is segmented into isolated gesture samples based on the manual annotations of the beginning and the ending frames. The learning task is to predict the class labels for each gesture sample, and the classification accuracy is used as the evaluation metric. Although the dataset has both RGB and depth videos, we only use the RGB videos as input.

### 4.3.1. Input strategy

Because the convolutional neural networks require fixed size inputs, the gestures need to be preprocessed to a fixed length. Traditional methods exploit the segment-based strategy used in action recognition. It first splits the gestures into short fixed-length clips, then processes each clip with 3D CNNs separately and finally fuses the results by average pooling or LSTM-based methods. We argue that there is no need to segment the gesture because it will neglect the temporal relationship between segments. Besides, these methods are ineffective due to the segmenting step.

**Table 1**
Results for different input strategies.

| Model | Accuracy |
|---|---|
| C3D (Average Pooling) | 85.1 |
| C3D (LSTM) | 88.9 |
| C3D (Uniform Sampling) | **89.8** |

**Table 2**
Results of the modified C3D and ResNet3D-18 with and without using models pre-trained on the Kinetics dataset.

| Model | Depth | Pre-train | Accuracy | #Params |
|---|---|---|---|---|
| Standard C3D | 11 | No | 89.8 | 79.0M |
| Modified C3D | 9 | No | 90.4 | 36.2M |
| Modified C3D | 9 | Yes | **91.3** | 36.2M |

**Table 3**
Results for different 3D CNNs.

| Model | Depth | Acc | #Params | FPS |
|---|---|---|---|---|
| ResNet3D-18 | 18 | 90.7 | 31.8M | 1950 |
| Deformable ResNet3D-18 | 18 | 91.7 | 35.5M | 1900 |
| ResNet3D-34 | 34 | 91.9 | 60.8M | 1600 |
| Deformable ResNet3D-34 | 34 | 92.2 | 65.8M | 1540 |
| ResNet3D-101 | 101 | 94.0 | 81.7M | 1040 |
| Deformable ResNet3D-101 | 101 | 94.2 | 85.5M | 907 |
| ResNeXt3D-101 | 101 | 94.2 | 45.8M | 700 |
| Deformable ResNeXt3D-101 | 101 | **94.7** | 52.2M | 600 |
| Inception3D-V1 | 22 | 89.6 | 12.0M | 1910 |
| Deformable Inception3D-V1 | 22 | 90.9 | 12.5M | 1870 |
| InceptionResNet3D-V2 | 190 | 92.3 | 111.5M | 640 |
| Deformable InceptionResNet3D-V2 | 190 | 92.7 | 118.0M | 416 |

Instead of using segment-based methods, we uniformly sample the whole gesture into an uniform length and feed them into the 3D CNN. The results employing different strategies are shown in Table 1, from which we can see that directly sampling the whole gesture outperforms other methods.

### 4.3.2. Model pre-training

The three fully connected layers at the end of the standard C3D bring more than half of the model parameters, which makes the model difficult to train. By removing the three fully connected layers and adding Batch Normalization after convolution, the modified C3D shows the superiority with higher accuracy yet with fewer parameters (Table 2).

To test the importance of pre-training, we evaluate the modified C3D with and without the pre-trained model. As there is no large dataset for gesture recognition, an action dataset, Kinetics [29], is used in this paper for model pre-training. It has 300,000 videos around 10 seconds for 400 action classes. The results are shown in Table 2. It can be found that the pre-training is vital for the training of 3D CNNs as the performance gap with or without pre-training is large. We argue that it is because the 3D CNN has a vast number of parameters due to the additional dimension of the kernel, which makes it more data-hungry.

### 4.3.3. Going deeper

The C3D is relatively shallow compared with the successful models used in image classification. We further test the deeper and more powerful models described in Section 3.1 to see whether these models can perform better. The ResNet3D-18, ResNet3D-34, ResNet3D-101, ResNext3D-101, Inception3D-V1 and InceptionResNet-v2 are evaluated. All the above models are pre-trained on the Kinetics dataset and finetuned on the EgoGesture

**Table 4**
Effect of different positions for deformable 3D convolution. Res5c represents the third convolutional layer in Conv5_x of ResNeXt3D-101 shown in Table 6. Others can be inferred accordingly.

| Position | Accuracy | #params |
|---|---|---|
| None | 94.2 | 45.8M |
| res5c | 94.5 | 47.9M |
| res5bc | 94.6 | 50.0M |
| res5abc | **94.7** | 52.2M |
| res5abc&res4abc | 94.7 | 58.0M |

**Table 5**
Results for different data augmentation strategies. (D) stands for the spatiotemporal data augmentation strategy described in Section 4.3.5.

| Methods | Accuracy |
|---|---|
| ResNext3D-101 | 94.2 |
| ResNext3D-101(D) | 94.5 |
| ResNext3D-101(D) + left-right flip | 93.0 |
| ResNext3D-101(D) + dropout | 94.7 |
| Deformable ResNext3D-101(D) + dropout | **95.1** |

dataset. The FPS is calculated by averaging the inference time of validation set with batch size 1 on a single Titan-XP GPU. The loading and preprocessing time are took in to account in the calculation of FPS.

Final results are listed in Table 3, which shows that the recognition performance rises when using deeper models. In detail, the Inception3D get the lowest accuracy as it has the minimum number of parameters compared with other models. The performance of the ResNet family improves when they going deeper. By using the group convolution, ResNeXt3D-101 achieves the best performance among these models while keeping the number of parameters moderate. Hence, we use this architecture as the backbone model in our work. The InceptionResNet3D-V2 is the deepest model with the largest number of parameters, but its performance is not as well as ResNeXt3D-101. Moreover, all of these models can be executed in real-time since we only use the RGB modality in an end-to-end framework without additional steps such as detection and segmentation. There is a negative correlation between the FPS rate and the parameter amount of the model. It can be seen that the extra computational cost of adding the deformable convolutional module is limited for all of the models because the module only needs to be added in a small number of top layers. The speed decrease of the deeper models (e.g. InceptionResNet3D-V2) is larger than the shallower models (e.g. ResNet3D-18) by adding the module. The reason is that the channel number of the top layers of the deeper model is larger than that of the shallower model, which results in more parameters for this layer.

### 4.3.4. Embedding the spatiotemporal deformable convolution

As introduced in Section 3.2, 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. Table 4 evaluates the effect of deformable convolution positions in the network, where we make the experiments based on ResNext3D-101 (Table 6). We gradually replace the convolutional layer with the deformable version from the top layer to the bottom layer. It is observed that the accuracy is steadily improved when more deformable convolution layers are used, and the best accuracy is obtained when the final three convolutional layers ($conv5_x$ in Table 6) are modified to deformable version. We believe that the learning of offsets needs the high-level semantic information, which the lower layers cannot provide. Extra deformable convolutional layers will bring additional parameters to learn, which may harm the training process. According to the result, three deformable convolutional layers are enough for this architecture.

As for other architecture, we replace the top 2 layers of ResNet3D-18, top 3 layers of ResNet3D-34, top 2 layers of Inception3D-V1 and top 11 layers of InceptionResNet3D-V2. The details can be found in the released code. All the results are shown in Table 3, where adding the deformable module brings consistent improvement in accuracy. This well demonstrates the effectiveness of the proposed spatiotemporal deformable convolution modules. Besides, there is only a limited increase in the amount of parameters and computation time.

### 4.3.5. Spatiotemporal data augmentation

Data augmentation has been proved an essential skill in the training process for image classification. It should be effective for training of 3D CNNs because the 3D CNNs have a large number of parameters and suffer more from the overfitting problem. We design a spatiotemporal data augmentation method. It first crops the video in both spatial and temporal dimensions, then resizes the cropped video to the final resolution. The crop-shape is randomly generated, and the resize-shape is decided by the network architecture.

In detail, we use ResNext3D-101 to test the data augmentation skill in EgoGesture. The input-shape of the network we used in this work is 32 × 112 × 112 which corresponds to length, height and width. The original size of the video is $l \times 240 \times 320$, where $l$ represents the temporal length of the video and is varied for different samples. The crop-shape is randomly generated between 40 × 224 × 224 and $l \times 240 \times 320$. The original video is first cropped to the generated crop-shape and then resized to 32 × 112 × 112. The operation of temporal resizing is achieved by uniform sampling. When testing the model, we choose three crop-shapes, i.e. 32 × 208 × 208, 40 × 224 × 224 and 48 × 240 × 240. For each crop-shape, 8 corners and 1 center of each sample are cropped. There are totally 27 clips generated for one gesture video, and prediction scores are averaged to predict the final label. All experiments are fine-tuned on Kinetics pre-trained models, and other details are the same as Section 4.3.2. Besides, left-right flipping for image and adding the dropout layer (drop rate is 0.5) before fully connected layer are also tested.

Table 5 shows the results of experiments introduced above. It shows consistent improvement using spatiotemporal data augmentation skills and dropout layer. An interesting phenomenon is that using left-right flipping augmentation harm the performance of recognition. We argue that flipping operation confuses the model when distinguishing gestures like “swiping left” versus “swiping right.” Combing the data augmentation skills with deformable convolution module, our Deformable ResNeXt3D-101 achieves the highest accuracy.

### 4.3.6. Compared with other methods

Table 6 describes the architecture of Deformable ResNeXt3D-101. The proposed model is compared with previous state-of-the-art methods using RGB videos as input. iDT-FV [39] is the most widely used hand-crafted features for video analyses. VGG16+LSTM [40] extracts the frame feature with VGG and feed them into LSTM for video-level modeling. C3D+SVM [8] uses C3D to model short clips and fuses the features with average pooling. The final label is predicted by SVM. C3D+LSTM [6] is similar to VGG16+LSTM, but it replaces the VGG16 with C3D. C3D+RSTTM [6] further plugs a spatiotemporal transformer into LSTM to better augment the model. Table 7 shows the final recognition accuracy of these methods, where our model achieves the

**Table 6**
Illustration of the architecture for Deformable ResNeXt3D-101. Each convolutional layer is followed with Batch Normalization and ReLU. (T) represents the temporal dimension and (XY) represents the spatial dimension. Downsampling is performed on conv3_1, conv4_1 and conv5_1 with the stride of 2. Every convolutional layer with 3 × 3 × 3 kernel size is grouped along the channel dimension with $g = 32$.

| Layers | Deformable 3D ResNeXt-101 |
|---|---|
| conv1 | $7 \times 7 \times 7, 64, stride1(T), 2(XY)$<br>$3 \times 3 \times 3, max - pooling, stride1(T), 2(XY)$ |
| conv2_x | $\begin{Bmatrix} 1 \times 1 \times 1, 128 \\ 3 \times 3 \times 3, 128 \\ 1 \times 1 \times 1, 256 \end{Bmatrix} \times 3$ |
| conv3_x | $\begin{Bmatrix} 1 \times 1 \times 1, 256 \\ 3 \times 3 \times 3, 256 \\ 1 \times 1 \times 1, 512 \end{Bmatrix} \times 4$ |
| conv4_x | $\begin{Bmatrix} 1 \times 1 \times 1, 512 \\ 3 \times 3 \times 3, 512 \\ 1 \times 1 \times 1, 1024 \end{Bmatrix} \times 23$ |
| deformable conv5_x | $\begin{Bmatrix} 1 \times 1 \times 1, 1024 \\ 3 \times 3 \times 3, 2048 \\ 1 \times 1 \times 1, 2048 \end{Bmatrix} \times 3$ |
| average pooling, dropout, fc, SoftMax | |

**Table 7**
Validation accuracies on EgoGesture.

| Methods | Accuracy |
|---|---|
| iDT-FV [39] | 64.3 |
| VGG16+LSTM [40] | 74.7 |
| C3D+SVM [8] | 86.4 |
| C3D+LSTM [6] | 88.9 |
| C3D+RSTTM [6] | 89.3 |
| Inception3D-V1 [29] | 90.9 |
| ResNet3D-18 [31] | 91.7 |
| S3D [33] | 93.4 |
| Deformable 3D ResNeXt | **95.1** |

**Table 8**
Validation accuracies on Jester.

| Methods | Accuracy |
|---|---|
| Modified C3D | 92.2 |
| ResNet3D-18 | 93.2 |
| ResNet3D-34 | 93.9 |
| ResNet3D-101 | 94.4 |
| Inception3D-V1 | 92.6 |
| InceptionResNet-v2 | 95.4 |
| S3D | 96.6 |
| ResNeXt3D-101 | 96.4 |
| Deformable ResNeXt3D-101 | **97.1** |

best performance compared with other methods with a large margin.

## 4.4. Results on jester

### 4.4.1. Accuracy on validation set

The Jester dataset is split into training, validation and testing sets according to the official provided.csv files. Training details are similar with Section 4.2 and the final accuracy of different models on validation set are listed in Table 8. Consistent with the results in EgoGesture, the ResNeXt3D-101 achieves the best performance among tested models and embedding deformable convolution further improves the recognition accuracy. Different from the results in EgoGesture, the difference between the competing models in Jester is more significant. We think that the number of the samples in Jester is larger than EgoGesture, which is beneficial to train deeper and larger models.

**Table 9**
Validation accuracies of classifying "Swiping Left" versus "Swiping Right" using ResNet3D-18, which is trained with original videos and evaluated with videos and labels that have different orders.

| Input | Accuracy |
|---|---|
| Original video / original label | 98.3 |
| Order-reversed video / original label | 1.2 |
| Order-reversed video / modified label | 94.3 |

**Table 10**
Validation accuracies on Jester using videos and labels in different orders.

| Input | Accuracy |
|---|---|
| Original video / original label | 97.1 |
| Order-reversed video / original label | 45.5 |
| Single image | 58.5 |

### 4.4.2. Do 3D CNNs really learn the temporal information?

We pick up a subset of Jester which contains only two classes, i.e., "Swiping Left" and "Swiping Right." These two classes are strongly temporal dependent. We train a ResNet3D-18 on this subset, which achieves 98.26% on validation set (Table 9). Then, we reverse the order of the testing videos without modifying the label and evaluate the original model again using the reversed data. The performance drops dramatically to 1.24%. Finally, we use the order-reversed testing videos again but modify the labels of the two classes accordingly. For example, if we reverse the order of a video with label "Swiping Left", we will modify the label to "Swiping Right." When evaluating the original model again, the performance returns back to the original level (94.28%). It means that 3D CNNs distinguish the two classes mainly based on the temporal information. In other words, 3D CNNs do learn the temporal information to distinguish the two classes.

We further make the experiment on all of the classes on Jester. In detail, we train a Deformable ResNeXt3D-101 with original videos and test it using the order-reversed videos. Table 10 shows that there is a significant drop on performance when using order-reversed videos for testing. We plot the accuracy values of all classes as well as the difference between accuracy values using different inputs in Fig 5. It shows that the performance of using order-reversed videos as input drops more in temporal dependent classes such as "Swiping Left" versus "Swiping Right" and drops less in temporally independent classes such as "Thumb Up." It illustrates that the 3D CNNs can well capture the appearance information for the temporally independent classes as well as the temporal information for the temporal dependent classes.

We also test the performance of applying 2D CNNs, which cannot model temporal information at all. We train a 2D ResNeXt-101 using the randomly selected frame from videos as input. For evaluating, the results of all the frames of the video are averaged to get the final prediction. Table 10 shows that the accuracy of applying 2D CNNs is much lower than applying 3D CNNs, which illustrates the importance of modeling temporal information. Fig. 5 shows that 2D CNNs cannot distinguish the classes that are strongly temporal dependent. For example, it can successfully recognize the class of "Thumb Up" ( $>$ 90%), but it fails to distinguish the samples of "Swiping Left" versus "Swiping Right" ( $<$ 40%).

The confusion matrix of final result using Deformable ResNeXt3D-101 is plotted in Fig. 6. It shows that most of the confusing gesture pairs are successfully distinguished, such as "Swiping Left" versus "Swiping Right," "Swiping Up" versus "Swiping Down." The pair "Turning Hand Clockwise" and "Turning Hand Counterclockwise" are a little confused. Based our obervation, we

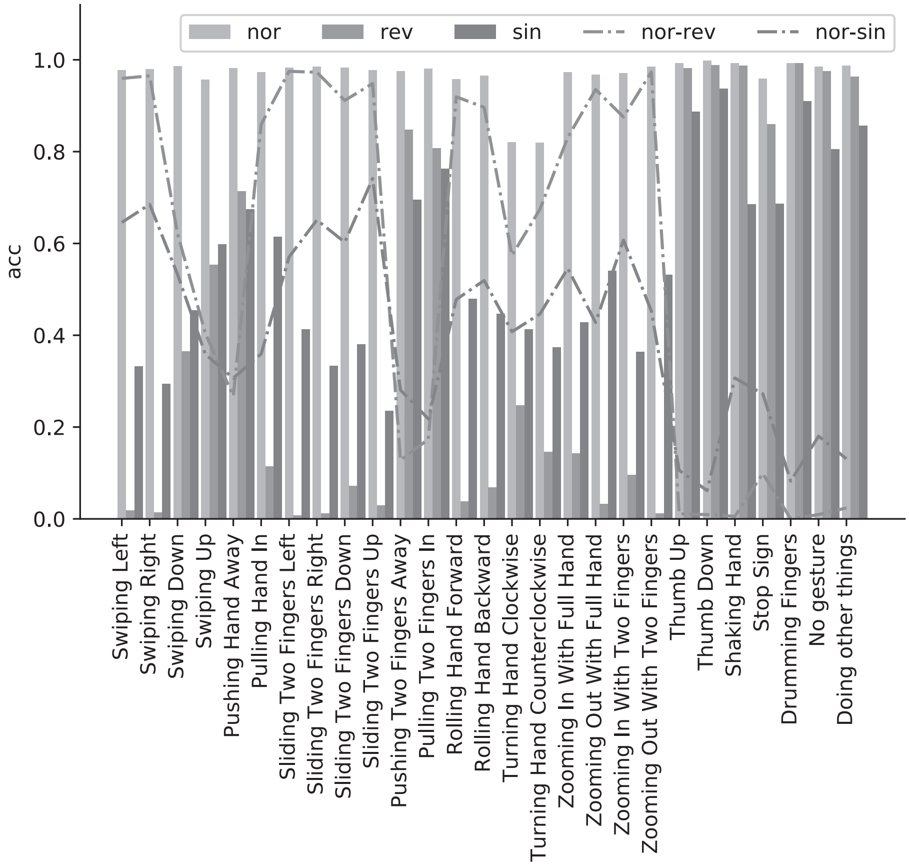


**Fig. 5.** The accuracies for each of the classes with different inputs. Nor, rev and sin stand for the normal-order video, reverse-order video and single image respectively. The symbols nor-rev and nor-sin stand for the difference between the two items.

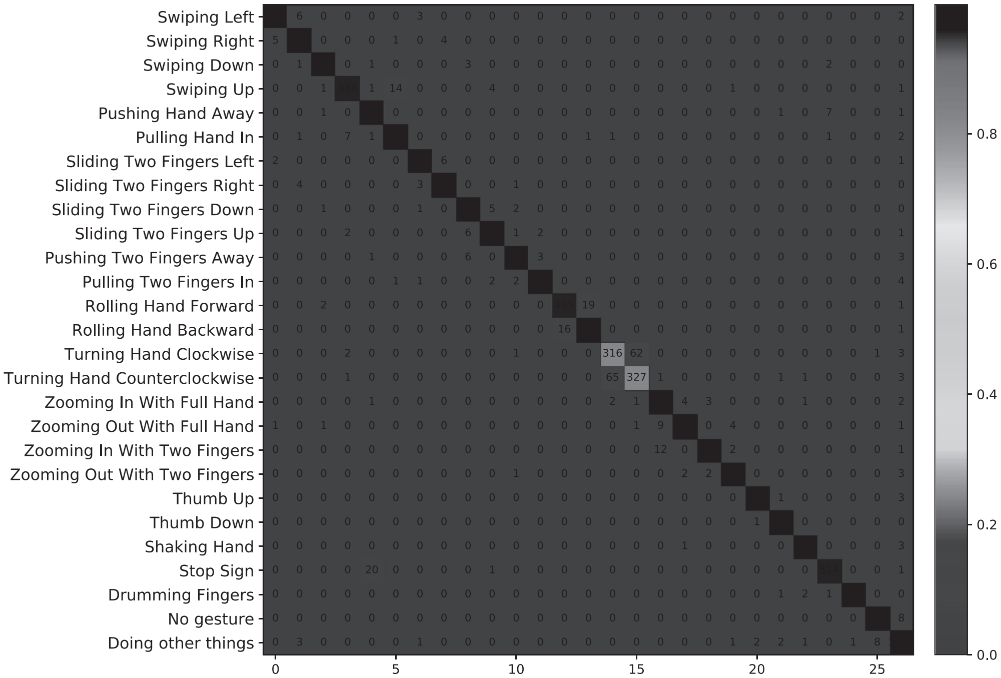


**Fig. 6.** Confusion matrix of recognition result in Jester.

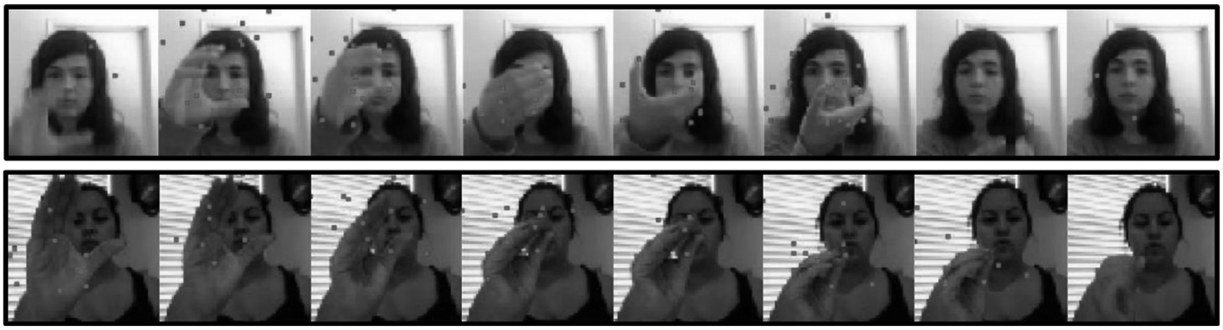


**Fig. 7.** Visualization of the offsets learned by deformable 3D convolution. The labels of two samples are “Turning Hand Clockwise” and “Zooming In With Full Hand”.

**Table 11**
Accuracies of 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 |

**Table 12**
Test accuracy in Jester dataset.

| Methods | Accuracy |
|---|---|
| 20BN’s Jester System | 82.3 |
| Ford’s Gesture Recognition System | 94.1 |
| Besnet [41] | 94.2 |
| TRN [42] | 94.8 |
| DIN | 95.31 |
| Spatiotemporal Two Streams Network | 96.3 |
| Motion Fused Frames [43] | 96.3 |
| ResNeXt3D-101 | 95.7 |
| Deformable ResNeXt3D-101 | **96.6** |

found that there are a lot of wrong labels for these two classes in the original dataset, which causes the low performance.

### 4.4.3. Deformable convolution

To better show the effectiveness of 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 11, from which we can see the performance gain of using deformable convolution increases when the gestures are incomplete or not in the center. It can be explained as the offsets learned by deformable convolution can help the model extract useful features.

Two successfully recognized samples in Jester are visualized in Fig. 7. Sample locations of one deformable convolutional step are plotted with red points. It can be seen that sample locations are deformed in both spatial (the sample locations are not in a grid) and temporal (the number of sample locations in different frames are different) dimensions to match the video content better.

### 4.4.4. Accuracy on test data

Our models are further evaluated on the test set of Jester, and the result is submitted to the official leaderboard. Table 12 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. We cannot find the papers of several submissions so we only list the methods name they provided in the website

**Table 13**
Validation accuracies in Chalearn-IsoGD dataset.

| Model | Chalearn |
|---|---|
| Li et al [22] | 37.3 |
| Zhu et al [23] | 43.9 |
| ResC3D [44] | 45.1 |
| Wang et al [21] | 43.7 |
| 2SCVN-RGB [26] | 45.7 |
| Roitberg et al [45] | 52.3 |
| ResNeXt3D-101 | 53.0 |
| Deformable ResNeXt3D-101 | 54.3 |
| InceptionResNet3D-V2 | 55.1 |
| Deformable InceptionResNet3D-V2 | **55.8** |

### 4.5. Results on chalearn-IsoGD

We also conduct experiments on the Chalearn-IsoGD dataset. ResNeXt3D-101 and InceptionResNet3D-V2 are compared with their deformable version in Table 13. It shows that the InceptionResNet3D-V2 performs better than ResNeXt3D-101. Adding the deformable convolutional module brings consistent improvements for both of the two models. Besides, our models are compared with other methods that use only the RGB videos as input on the Chalearn-IsoGD dataset, where our model achieves the state-of-the-art performance.

## 5. Conclusion

In this work, three types of very deep 3D CNNs, which are extended from the models succeeded in the image classification domain, are proposed for dynamic gesture recognition. A spatiotemporal deformable convolutional module is specially designed to augment the sampling locations of the 3D convolution, which helps models paying more attention to discriminative parts of the video sequence in both spatial and temporal dimensions. We investigate the effect of the hyper-parameters of the deformable convolutional module and get the best configuration based on the ablation studies. We observed that the benefit of plugging the spatiotemporal deformable convolution module to the higher level layer is larger than that to the lower level layer. This can be explained as the offset learning needs high-level semantic information under larger receptive field, which lower layers cannot provide. We further propose some practical skills for the training of the deep 3D CNNs, such as the spatiotemporal data augmentation and using pretrained models. The final model is evaluated on three challenging datasets, EgoGesture, Jester and Chalearn, which achieves the state-of-the-art performance on all of them. We also conduct two experiments to confirm the ability of the 3D CNNs for modeling the temporal information and the ability of the spatiotemporal deformable convolutional module for paying attention to discriminative contents in videos. Future works can focus on how to better model the temporal relations between frames and reduce the parameters in the 3D CNNs without the dropping of the performance. It is also worth to investigate how to combine the RGB videos with other data modalities such as the depth information and the pose information.

## Declaration of Competing Interest

None.

## Acknowledgement

This work was supported in part by the NSFC 61876182, 61872364, 61876086, and in part by Jiangsu Frontier Technology Basic Research Project BK20192004. Lei Shi is the co-first author. Yi Wu is the corresponding author.

## References

[1] L. Pigou, A. Van Den Oord, S. Dieleman, M. Van Herreweghe, J. Dambre, Beyond temporal pooling: recurrence and temporal convolutions for gesture recognition in video, Int J Comput Vis 126 (2–4) (2018) 430–439.

[2] D. Wu, L. Pigou, P.J. Kindermans, L.E. Nam, L. Shao, J. Dambre, J.M. Odobez, Deep dynamic neural networks for multimodal gesture segmentation and recognition, IEEE Transactions on Pattern Analysis Machine Intelligence 38 (8) (2016) 1583–1597.

[3] K. Soomro, A.R. Zamir, M. Shah, UCF101: A dataset of 101 human actions classes from videos in the wild, CoRR Abs/1212.0402 (2012).

[4] K. Simonyan, A. Zisserman, Two-Stream Convolutional Networks for aAction Recognition in Videos, in: Advances in Neural Information Processing Systems, 2014, pp. 568–576.

[5] S. Anwar, S.K. Sinha, S. Vivek, V. Ashank, Hand Gesture Recognition: A Survey, in: Nanoelectronics, Circuits and Communication Systems, Springer, 2019, pp. 365–371.

[6] C. Cao, Y. Zhang, Y. Wu, H. Lu, J. Cheng, Egocentric Gesture Recognition Using Recurrent 3D Convolutional Neural Networks With Spatiotemporal Transformer Modules, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 3783–3791.

[7] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, Y. Wei, Deformable Convolutional Networks, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 41, 2017, pp. 40–54.

[8] D. Tran, L. Bourdev, R. Fergus, L. Torresani, M. Paluri, Learning Spatiotemporal Features With 3d Convolutional Networks, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 4489–4497.

[9] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[10] S. Xie, R. Girshick, P. Dollar, Z. Tu, K. He, Aggregated Residual Transformations for Deep Neural Networks, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1492–1500.

[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going Deeper With Convolutions, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1–9.

[12] Y. Zhang, C. Cao, J. Cheng, H. Lu, Egogesture: a new dataset and benchmark for egocentric hand gesture recognition, IEEE Trans Multimedia 20 (2018) 1038–1050.

[13] J. Materzynska, G. Berger, I. Bax, R. Memisevic, The Jester Dataset: A Large-Scale Video Dataset of Human Gestures, in: Proc. IEEE International Conference on Computer Vision Workshops (ICCVW), 2019.

[14] I. Guyon, V. Athitsos, P. Jangyodsuk, H.J. Escalante, The chalearn gesture dataset (cgd 2011), Mach Vis Appl 25 (8) (2014) 1929–1951.

[15] E. Ohn-Bar, M.M. Trivedi, Hand gesture recognition in real time for automotive interfaces: a multimodal vision-based approach and evaluations, IEEE Trans. Intell. Transp. Syst. 15 (6) (2014) 2368–2377.

[16] J. Wan, G. Guo, S.Z. Li, Explore efficient local features from RGB-ddata for one-shot learning gesture recognition, IEEE Trans Pattern Anal Mach Intell 38 (8) (2016) 1626–1639.

[17] J. Tang, H. Cheng, Y. Zhao, H. Guo, Structured dynamic time warping for continuous hand trajectory gesture recognition, Pattern Recognit 80 (2018) 21–31.

[18] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, IEEE Trans Pattern Anal Mach Intell 37 (9) (2015) 1904–1916.

[19] C. Ionescu, D. Papava, V. Olaru, C. Sminchisescu, Human3.6m: large scale datasets and predictive methods for 3d human sensing in natural environments, IEEE Trans Pattern Anal Mach Intell 36 (7) (2014) 1325–1339.

[20] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, L. Van Gool, Temporal segment networks for action recognition in videos, IEEE Trans Pattern Anal Mach Intell 41 (11) (2019) 2740–2755.

[21] P. Wang, W. Li, Z. Gao, C. Tang, P.O. Ogunbona, Depth pooling based large-scale 3-daction recognition with convolutional neural networks, IEEE Trans Multimedia 20 (5) (2018) 1051–1061.

[22] Y. Li, Q. Miao, K. Tian, Y. Fan, X. Xu, R. Li, J. Song, Large-scale gesture recognition with a fusion of rgb-d data based on saliency theory and c3d model, IEEE Trans. Circuits Syst. Video Technol. 28 (10) (2017) 2956–2964.

[23] G. Zhu, L. Zhang, P. Shen, J. Song, Multimodal gesture recognition using 3-D-convolution and convolutional LSTM, IEEE Access 5 (2017) 4517–4524.

[24] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: Proceedings of the International Conference on Machine Learning (ICML), 2013, pp. 1310–1318.

[25] Y. Li, Q. Miao, K. Tian, Y. Fan, X. Xu, Z. Ma, J. Song, Large-scale gesture recognition with a fusion of RGB-D data based on optical flow and the C3D model, Pattern Recognit Lett 119 (2019) 187–194.

[26] J. Duan, J. Wan, S. Zhou, X. Guo, S.Z. Li, A unified framework for multimodal isolated gesture recognition, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 14 (1s) (2018) 21.

[27] J.Y. Chang, Nonparametric feature matching based conditional random fields for gesture recognition from multi-modal video, IEEE Trans Pattern Anal Mach Intell 38 (8) (2016) 1612–1625.

[28] S. Ji, W. Xu, M. Yang, K. Yu, 3D convolutional neural networks for human action recognition, IEEE Trans Pattern Anal Mach Intell 35 (1) (2013) 221–231.

[29] J. Carreira, A. Zisserman, Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6299–6308.

[30] S. Ioffe, C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, in: Proceedings of the International Conference on Machine Learning (ICML), 2015, pp. 448–456.

[31] K. Hara, H. Kataoka, Y. Satoh, Can Spatiotemporal 3d CNNs Retrace the History of 2d CNNs and ImageNet? in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 6546–6555.

[32] K. He, X. Zhang, S. Ren, J. Sun, Identity Mappings in Deep Residual Networks, in: Proceedings of the European Conference on Computer Vision (ECCV), 2016, pp. 630–645.

[33] S. Xie, C. Sun, J. Huang, Z. Tu, K. Murphy, Rethinking Spatiotemporal Feature Learning: Speed-accuracy Trade-offs in Video Classification, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 305–321.

[34] Z. Qiu, T. Yao, T. Mei, Learning deep spatio-temporal dependence for semantic video segmentation, IEEE Trans Multimedia 20 (4) (2017) 939–949.

[35] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, X. Tang, Residual Attention Network for Image Classification, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. 6156–3164

[36] M. Jaderberg, K. Simonyan, A. Zisserman, K. Kavukcuoglu, Spatial Transformer Networks, in: Advances in Neural Information Processing Systems, 2015, pp. 2017–2025.

[37] F. Yu, V. Koltun, Multi-Scale Context Aggregation by Dilated Convolutions, in: Proceedings of the International Conference on Learning Representations (ICLR), 2016.

[38] C. Szegedy, S. Ioffe, V. Vanhoucke, A.A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning., in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI), 2017, pp. 4278–4282.

[39] H. Wang, A. Kläser, C. Schmid, C.-L. Liu, Dense trajectories and motion boundary descriptors for action recognition, Int J Comput Vis 103 (1) (2013) 60–79.

[40] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, T. Darrell, Long-Term Recurrent Convolutional Networks for Visual Recognition and Description, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 2625–2634.

[41] B. Ghanem, J.C. Niebles, C. Snoek, F.C. Heilbron, H. Alwassel, R. Khrisna, V. Escorcia, K. Hata, S. Buch, Activitynet challenge 2017 summary, CoRR Abs/1710.08011 (2017).

[42] B. Zhou, A. Andonian, A. Torralba, Temporal Relational Reasoning in Videos, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018.

[43] O. Kopuklu, N. Kose, G. 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 (CVPRW), 2018, pp. 2103–2111.

[44] Y. Li, Q. Miao, X. Qi, Z. Ma, W. Ouyang, A spatiotemporal attention-based resc3d model for large-scale gesture recognition, Mach Vis Appl 30 (5) (2019) 875–888.

[45] A. Roitberg, T. Pollert, M. Haurilet, M. Martin, R. Stiefelhagen, Analysis of Deep Fusion Strategies for Multi-Modal Gesture Recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2019.

**Yifan Zhang** received the B.E. degree in Automation from Southeast University in 2004, and the Ph.D. degree in Pattern Recognition and Intelligent Systems from Institute of Automation, Chinese Academy of Sciences in 2010. Then he has joined National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences, Beijing, China, where he is currently an Associate Professor. From 2011 to 2012, he was a postdoctor with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute (RPI), Troy, New York. His research interests include machine learning, computer vision, probabilistic graphical models, and their applications, especially on video content analysis, gesture recognition, action recognition etc.

**Lei Shi** received the B.E. degree in Automation from Central South University in 2016. He is currently a Ph.D. candidate of Image and Video Analysis at the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His current research interests include machine learning, pattern recognition and relative applications, especially on video-based action recognition and gesture recognition.

**Yi Wu** is a Principal Researcher at Wormpex AI Research. He received the Ph.D. degree in pattern recognition and intelligent systems from Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2009. From 2010 to 2012, he was a Post-Doctoral Fellow with Temple University, Philadelphia, PA, USA. From 2012 to 2014, he was a Post-Doctoral Fellow with the University of California, Merced, CA, USA. From 2017 to 2018, he was a Research Assistant Professor at Indiana University School of Medicine. His research interests include computer vision, medical image analysis, and deep learning.

**Ke Cheng** received the B.E. degree in Automation from Huazhong University of Science and Technology in 2015. He is currently a Ph.D. candidate of Image and Video Analysis at the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His current research interests include machine learning, pattern recognition and relative applications.

**Jian Cheng** received the B.S. and M.S. degrees from Wuhan University, in 1998 and 2001, respectively, and the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, in 2004. From 2004 to 2006, he was a Post-Doctoral Fellow with the Nokia Research Center, China. Currently, he is a Professor in the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His current research interests include machine learning, pattern recognition, computing architecture and chips and data mining.

**Hanqing Lu** received the B.E. and M.E. degrees from the Harbin Institute of Technology, in 1982 and 1985, rescpetively, and the Ph.D. degree in Huazhong University of Sciences and Technology, Wuhan, China in 1992. Currently he is a Professor of Institute of Automation, Chinese Academy of Sciences. His research interests include image and video analysis, pattern recognition and object recognition. He published more than 100 papers in these areas.