International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2019): 7.583

Review on CNN based Sign Language Recognition Methods

Akash Vijayan[1], Sajeena .A[2]

[1]Electronics and Communication, TKM College of Engineering, Kollam, Kerala, India
akashvijayan2013[at]gmail.com

[2]Associate Professor, Electronics and Communication, TKM College of Engineering, Kollam, Kerala, India
sajina.nizam[at]gmail.com

Abstract: ***Fingerspelled sign learning is the initial stage of sign learning and moreover, are used when no corresponding sign exists or signer is not aware of it. Most of the existing tools for sign language learning use external sensors which are costly and unreliable. . The ASL, short for American Sign Language is the most widely used sign language across the globe with certain variations according to the country.Sign language recognition can improve the problem that the number of people who need to use sign language is large but the popularity of sign language is poor, and provide a more convenient way of study, work and life for people with hearing and language impairment. Hand locating and sign language recognition methods can generally be divided into traditional methods and deep learning methods. In recent years, with the brilliant achievements of deep learning in the field of computer vision, it has been proved that the method based on deep learning has many advantages, such as rich feature extraction, strong modeling ability and intuitive training. Therefore, this paper studies hand locating and sign language recognition of common sign language based on neural network.we proposed an efficient deep convolutional neural networks approach for hand gesture recognition. The proposed approach employed transfer learning to beat the scarcity of a large labeled hand gesture dataset***

Keywords: Fading, Interference, Wideband, Throughput, Received signal

1. Introduction

The hand gesture is a nonverbal form of communication. It consists of linguistic content that carries a large amount of information in sign language. It also plays a pivotal role in human-computer interaction (HCI) systems. Therefore, automatic hand gesture recognition is in high demand. Since the end of the last century, this field has attracted the attention of many researchers. The importance of automatic hand gesture recognition has increased for the following reasons (1) the growth of the deaf and hard-of-hearing populations, and (2) the extended use of vision-based and touchless applications and devices such as video games, smart TV control, and virtual reality applications. Robust hand gesture recognition is required as a part of sign language interpretation to help hearing-impaired people. There is a significant communication gap between people who can hear and hearing-impaired people. A translation system between gestural language and verbal language will bridge this communication gap. This translation system will facilitate the lives of hearing-impaired people and help them to integrate with society. Unlike sign language translation, hand gesture recognition techniques involve HCI to a great degree.

Today, HCI has a wide range of applications from video games to telesurgery. As with all time-varying signals, hand gestures cannot be directly compared in Euclidean space because of their temporal dependency. This dependency indicates important discriminative features. Temporal misalignment, in addition to massive irrelevant regions in every frame, makes it very hard to extract representative hand-engineered features for hand gestures. For conventional classifiers to perform well, the extracted features should implicate vigorous descriptors. These descriptors code enough information for the inter-frames temporal dependency, as well as the hand position, shape and orientation in each frame. The computed features should be able to minimize the effect of different circumstances like background clutter and occlusions. Therefore, we employed deep learning in this paper as a promising solution.

In recent years, many researchers have efficiently exploited convolutional neural networks (CNNs) deep architectures for feature engineering. CNNs have shown excellent performance in fields such as object and speech recognition, image classification, and edge distribution and human activity recognition. The existence of large datasets that comprise millions of annotated samples is the main reason behind such excellent performance. Unfortunately, the requirement for a large labeled dataset is not met in the case of hand gestures. To beat the scarcity of labeled dataset for fitting deep architectures, the transfer learning is investigated in this study. We propose a well-adapted deep architecture for automatic hand gesture recognition. The main contributions of this study are as follows:

1) A method to normalize the spatial dimensions of gesture videos based on the facial position, facial length, and human body part ratios. The signer does not need to be in the center of the frame or be a fixed distance from the camera.
2) A 3DCNN model to learn region-based spatiotemporal features for hand gestures. The input of this model is a sequence of RGB frames captured by a basic camera. It does not require other input channels, colored gloves, or a complex setup.
3) Developing different fusion techniques to globalize the local features learned by the 3DCNN model and comparing their performance.

1.1 3D Convolutional Neural Network

In 2D CNNs, 2D convolution is performed at the convolutional layers to extract features from local neighborhood on feature maps in the previous layer. Then an additive bias is applied and the result is passed through a sigmoid function. Formally, the value of unit at position (x, y) in the jthfeature map in the ith layer, denoted as v_{yj}^{xi}, is given by

$$v_{ij}^{xy} = \tanh\left(b_{ij} + \sum_{m}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1} w_{ijm}^{pq} v_{(i-1)m}^{(x+p)(y+q)}\right),$$

where tanh(·) is the hyperbolic tangent function b_{ij}, is the bias for this feature map, m indexes over the set of feature maps in the $(i-1)^{th}$ layer connected to the current feature map, w_{ijk}^{pq} is the value at the position (p, q) of the kernel connected to the kth feature map, and P_i and Q_i are the height and width of the kernel, respectively. In the subsampling layers, the resolution of the feature maps is reduced by pooling over local neighbourhood on the feature maps in the previous layer, thereby increasing invariance to distortions on the inputs. A CNN architecture can be constructed by stacking multiple layers of convolution and subsampling in an alternating fashion. The parameters of CNN, such as the bias b_{ij} and the kernel weight w_{ijk}^{pq} are usually trained using either supervised or unsupervised approaches.

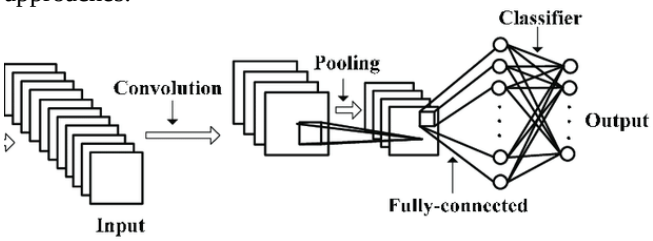


Figure 2.1: Basic steps in 3D CNN

1.2 3D Convolution

In 2D CNNs, convolutions are applied on the 2D feature maps to compute features from the spatial dimensions only. When applied to video analysis problems, it is desirable to capture the motion information encoded in multiple contiguous frames. To this end, we propose to perform 3D convolutions in the convolution stages of CNNs to compute features from both spatial and temporal dimensions. The 3D convolution is achieved by convolving a 3D kernel to the cube formed by stacking multiple contiguous frames together. By this construction, the feature maps in the convolution layer is connected to multiple contiguous frames in the previous layer, thereby capturing motion information. Formally, the value at position (x, y, z) on the jth feature map in the ithlayer is given by

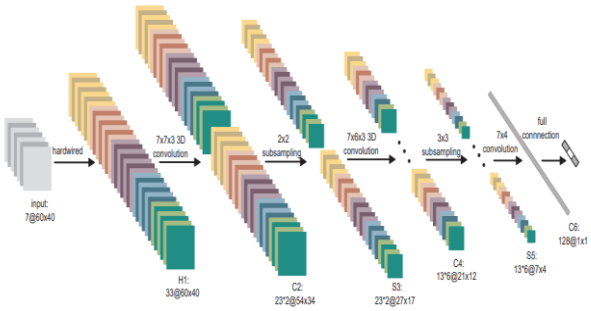
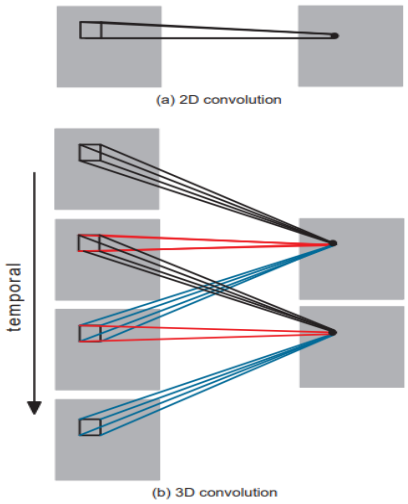


Figure 4.1: A 3D CNN architecture for human action recognition. This architecture consists of 1 hardwired layer, 3 convolution layers, 2 subsampling layers, and 1 full connection layer

where R_i is the size of the 3D kernel along the temporal dimension, w_{ijk}^{pq} is the (p, q, r)th value of the kernel connected to the m*th* feature map in the previous layer. A comparison of 2D and 3D convolutions is given in Figure 1. Note that a 3D convolutional kernel can only extract one type of features from the frame cube, since the kernel weights are replicated across the entire cube. A general design principle of CNNs is that the number of feature maps should be increased in late layers by generating multiple types of features from the same set of lower-level feature maps. Similar to the case of 2D convolution, this can be achieved by applying multiple 3D convolutions with distinct kernels to the same location in the previous layer



1.3 A 3D CNN Architecture

Based on the 3D convolution described above, a variety of CNN architectures can be devised. In the following, we describe a 3D CNN architecture that we have developed for human action recognition on the TRECVID data set. In this architecture shown in Figure 3, we consider 7 frames of size 60×40 centered on the current frame as inputs to the 3D CNN model. We first apply a set of hardwired kernels to generate multiple channels of information from the input frames. This results in 33 feature maps in the second layer in 5 different channels known as gray, gradient-x, gradient-y, optflow-x, and optflow-y. The gray channel contains the gray pixel values of the 7 input frames. The feature maps in the gradient-x and gradient-y channels are obtained by computing gradients along the horizontal and vertical

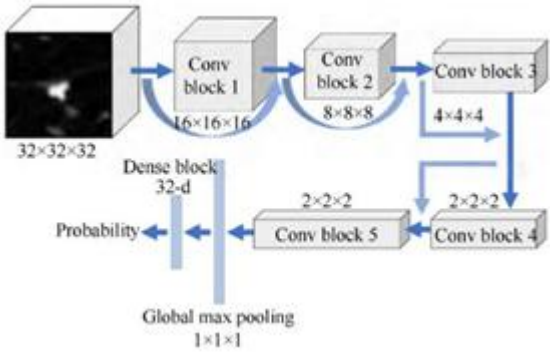
directions, respectively, on each of the 7 input frames, and the optflow-x and outflow-y channels contain the optical flow fields, along the horizontal and vertical directions, respectively, computed from adjacent input frames. This hardwired layer is used to encode our prior knowledge on features, and this scheme usually leads to better performance as compared to random initialization

$$v_{ij}^{xyz}=\tanh\left(b_{ij}+\sum_{m}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1}\sum_{r=0}^{R_i-1}w_{ijm}^{pqr}v_{(i-1)m}^{(x+p)(y+q)(z+r)}\right)$$

2. Proposed System

In this study, we utilized a 3DCNN architecture for spatiotemporal feature learning using two approaches. In the first approach, 3DCNN was used to extract the features from the entire video sample, while a SoftMax layer was used for classification. In the second approach, we aimed to enhance the temporal dependency of the video frames. To achieve this, the same 3DCNN architecture was trained to extract the features from different regions in the video sample. We then investigated different techniques for feature fusion.

2.1 Single 3DCNN Structure



The system proposed in the first approach is illustrated in Fig. 2. It consists of three main phases: video preprocessing, feature learning, and classification.

1) Video Preprocessing

The first step in the preprocessing phase was converting the input video into RGB frames sequence. Because the video sequences had different durations, linear sampling was applied to normalize all the sequences to a fixed length of 16 frames, as the original model was fit on video sequences of 16 frames each. The corresponding indices of 16 frames are calculated asin (1).

$$index_i = \text{round}\left(\frac{\text{len}(input)}{16}\times i\right), i \in \{1, 16\} \quad (1)$$

where, len(input) is the length of the input sequence. Other techniques such as the Bag of Visual Words have been used in the literature to normalize the temporal dimension of the input videos. Linear sampling was preferred in this work to preserve the order of the selected frames. The order of the selected frames indicates essential discriminative features in gesture recognition.

However, spatial dimension.

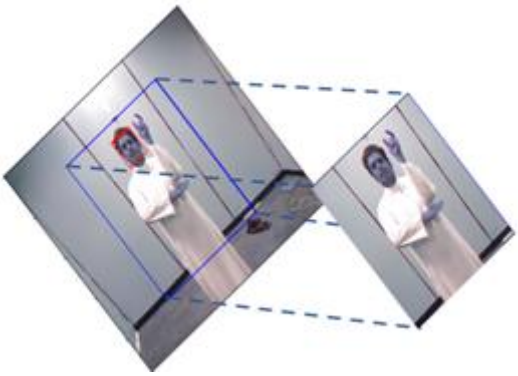


Figure 5.1: Cropping the signing space

normalization was also required to overcome variations in the heights and distances of the signers from the camera. We achieved this normalization in two steps: - First, we employed the face detection algorithm proposed by Viola and Jones to detect the face of the signer in the first frame of the sequence. - Then, based on the position and height of the detected face, we used the human body part ratios to estimate the height and width of the gesture space to be cropped in all frames, as illustrated in Fig. 3. The final step in the preprocessing phase was to resize the cropped square frames in all input videos to a fixed size of 112 × 112 pixels while maintaining the same aspect ratio. The RGB channels of each gesture sample were also normalized separately such that each channel had a zero mean and unit variance. This resizing and normalization reduced the computation cost and training convergence of the model in the next phase. The final inputs to the feature learning phase were 112 × 112 × 16 × 3 volumes.

2) Feature Learning

A deep 3DCNN is proposed for feature learning, to extract the local spatiotemporal features of gesture sequences. Transfer learning was employed here to beat the scarcity of a large labeled dataset of gestures. We started with a pre-trained version of the 3DCNN structure. This had already been trained using millions of samples of human action recognition. After excluding the output layer, the structure consisted of six consecutive blocks. The first two blocks have a single 3DCNN layer each. The first layer comprises 64 kernels and the second comprises 128 kernels. The third block contains two 3DCNN layers, each of 256 kernels. The fourth block contains two 3DCNN layers, each of 512 kernels. The fifth block contains two 3DCNN layers, each of 512 kernels and a zero-padding layer. The sixth block consists of two dense layers with 4096 neurons each. These two layers globalize the feature modeling. The output of each of the first four blocks transits to the successive block through a max-pooling layer. All 3DCNN kernels are (3 × 3 × 3) in size with a stride of (1 × 1 × 1). All max-pooling kernels are of size (2 × 2 × 2) with stride (2 × 2 × 2), except for the max-pooling kernel that follows the first block, which is of size (1 × 2 × 2) with stride (1 × 2 × 2) to preserve the temporal information in the early stage. A simple nonlinear function rectified linear unit (ReLU) was used for activation, as shown in (2). This function was preferred as it has a simple derivative to speed up large-network training [32]. ReLU (x) = (x, x ≥ 0 0, x < 0 (2) Each 3D kernel in the first layer is convolved to a volume of the 16 input stacking frames to produce a spatiotemporal

feature map. The 3D kernels in the successive layers are similarly convolved to a volume of stacking feature maps produced by the predecessor layers. In general, the value at any position (x, y, z) on the Kth feature map in the Lth layer is calculated as

$$V_{LK}^{xyz} = ReLU\left(b_{LK} + \sum_m \sum_{p=0}^{P_L-1} \sum_{q=0}^{Q_L-1} \sum_{r=0}^{R_L-1} W_{LKm}^{pqr} V_{(L-1)m}^{(x+p)(y+q)(z+r)}\right)$$

where PL, QL, and RL are the dimensions of the 3D kernel and W pqr LKm is the (p, q, r) th value of the kernel connected to the mth feature map in the preceding layer. The two fully connected layers globalize the feature modeling where the last layer outputs a feature vector length of 4096 to represent each sample

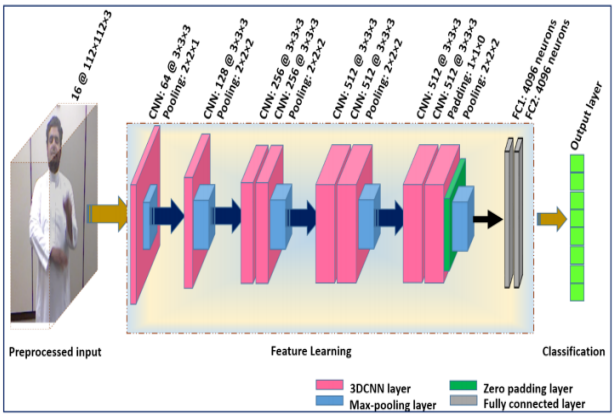


Figure 5.2: Fusion of parallel 3DCNN structure

3) Classification

The features extracted in the previous phase are fed into a SoftMax layer for classification, as illustrated in Fig. 2. The SoftMax activation function as defined in (4) outputs the probability of each class. The predicted output is the class with the maximum probability.

$$SoftMax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}}$$

2.2 Fusion of Parallel 3DCNN Structure

In the second approach, the proposed system enhanced the temporal contribution of the extracted features. To achieve this, in the preprocessing phase, linear sampling is applied to select 32 frames instead of 16 frames. Thereafter, three instances of the deep 3DCNN structure described in the previous approach are utilized to learn the spatiotemporal features in the beginning, middle, and end of the video sequence. The selected frames are divided into three short clips of 16 frames each with a 50% overlap. Each deep 3DCNN instance is trained to extract the features from one of the three clips. Various techniques are then utilized to fuse the features extracted from different parts of the video

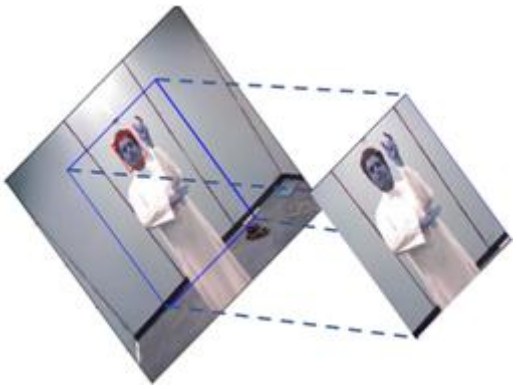


Figure 6.1: In depth cropping

Finally, the fused features are forwarded to the SoftMax layer for classification. Fig. 5.2 illustrates a general diagram of the second approach. Feature fusion Three techniques for feature fusion, multilayer perceptron (MLP) neural network, long short-term memory (LSTM) network, and stacked autoencoder were investigated.

MLP Fusion

MLP processes the input features through a series of computational nodes called neurons. These neurons are grouped into consecutive layers and interconnected with one another via weighted connections. These neurons transform the features by performing nonlinear operations. The features are then projected into a space where the input becomes linearly separable. MLP architectures with different numbers of layers were investigated in this research.

LSTM Fusion

An LSTM is a recurrent neural network (RNN) adopted to learn long-term contextual dependencies from learned local feature sequences. Fig. 6.3 illustrates the basic building block of LSTM networks. The behavior of this LSTM unit is controlled by three gates: the input gate, the forget gate, and the output gate. The input is fed into these gates to control which operations are to be performed by the unit. The memory state and output of an LSTM unit are updated at each time step. The LSTM transition equations at time step t can be formulated as

$$i_t = \sigma(W_i.[c_{t-1}, h_{t-1}, x_t] + b_i)$$
$$f_t = \sigma(W_f.[c_{t-1}, h_{t-1}, x_t] + b_f)$$
$$c_t = f_t.c_{t-1} + i_t.\tilde{c}_t$$
$$o_t = \sigma(W_o.[c_t, h_{t-1}, x_t] + b_o)$$
$$h_t = o_t.tanh(c_t)$$

where x_t, h_t, and ct are the input vector, output vector, and memory state, respectively, at time t. Terms i, f, o, and $\tilde{c}$ represent the input gate, forget gate, output gate, and cell activation, respectively, all of which are the same size as the input vector. Term σ represents nonlinear sigmoid functions.

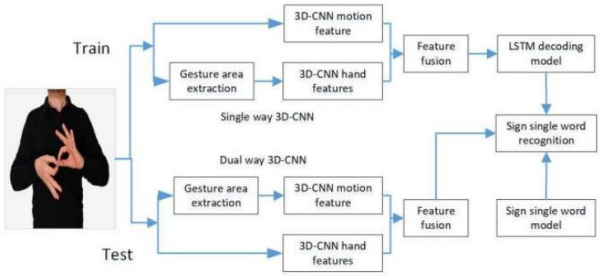


Figure 6.3: LSTM building block

We utilized recurrent LSTM architectures of different numbers of layers to enhance the automatic feature representation and model the temporal dependencies.

3. Auto Encoder Fusion

The simplest form of an autoencoder is a single hidden fully connected layer with input and output layers, as shown in Fig. 6.2. The number of nodes in the output layer must be the same as in the input layer. The autoencoder creates a new representation for the input data through a pair of maps $yf \rightarrow zg \rightarrow y'$. The first one is the encoder map $z = (y)$, and the second one is the decoder map $y' = g(z)$. The input data dimension is reduced by the encoder. The encoder transforms the input data of dimension d to a smaller dimension m [35]. The decoder reconstructs the input data from the reduced dimension m back to the original dimension d. During training, the autoencoder is usually forced to prioritize which aspects of the input should be kept [36]. The autoencoder maps an input point y to a code z via a sigmoid activation function as

$$z = f(y) = \sigma(Wy^T + b)$$

where W is a weight matrix, and b is a bias vector. The z code is also termed the latent representation of point y. The sigmoid function transforms the input values to the activation values, which are mostly either close to zero or 1. The decoder then maps this activation to the reconstructed y′ to the same dimensional space of y such that

$$y' = g(z) = \sigma(W'z + \bar{b})^T$$

where the weight matrix of the decoder is often the transpose of the weight matrix of the encoder, $W' = WT$. Training the autoencoder means finding the optimal values for W, b, and $\bar{b}$ that minimize the cost function [35]. A deeper stacked autoencoder, as used in our experiments, can be built by adding more paired layers to the encoder and decoder sides.

3.1 Hand Locating Based On Faster R-CNN

Sign language is mainly the movement of the hand, and has nothing to do with the complex background and movement of the human body. The interference of external factors will affect the result of sign language recognition. Therefore, the hand locating of sign language words is not only a crucial pretreatment step in sign language recognition, but also an essential step to extract gesture features and further gesture simulation. In the image sequence of sign language, the gesture is accompanied by the interference of a large number of skinlike region, which has the characteristics of complex and variable hand shape, much special hand shape and fuzzy hand movement. Therefore, the traditional methods of skin color detection and template classification cannot locate and classify gestures well. As one of the most advanced target detection networks, Faster R-CNN integrates the RPN module and the Faster R-CNN measurement module into an end-to-end network, so as to obtain better performance in terms of speed and accuracy. Compared with the single-stage target detection algorithm such as YOLO, Faster R-CNN can better meet the requirements of accurate detection and location of gestures.

Compared with Fast R-CNN, Faster R-CNN designs the RPN module. RPN is a convolutional neural network, which uses the shared CNN feature to generate candidate regions, and replaces the traditional selective search candidate region extraction algorithm, significantly improving the accuracy of candidate regions. At the same time, because the RPN module shares convolutional network with Fast R-CNN detection module, the detection speed of the Faster R-CNN module is also improved significantly. We train the Faster R-CNN hand locating network by using a data set consisting of 40 common words and 10,000 sign language images, among which the optimizer uses Stochastic Batch Gradient Descent (SGD). It can be found from the figure that the loss value of network decreases gradually with the increase of training times, and at 55000 iterations, the network tends to converge without overfitting. It compares the accuracy of hand locating of Faster R-CNN and the other two methods

3.2 Temporal Structure Modeling

Many research works have been devoted to modeling the temporal structure of video for action recognition [19], [20], [21], [22], [59], [60]. Gaidon et al. [20] annotated each atomic action for each video and proposed Actom Sequence Model (ASM) for action detection. Niebles et al. [19] proposed to use latent variables to model the temporal decomposition of complex actions, and resorted to the Latent SVM [61] to learn the model parameters in an iterative approach. Wang et al. [21] and Pirsiavash et al. [59] extended the temporal decomposition of complex action into a hierarchical manner using Latent Hierarchical Model (LHM) and Segmental Grammar Model (SGM), respectively. Wang et al. [60] designed a sequential skeleton model (SSM) to capture the relations among dynamic-poselets, and performed spatio-temporal action detection. Fernando et al. [22] modeled the temporal evolution of BoVW representations for action recognition. Several recent works focused on modeling long-range temporal structure with ConvNets [4], [24], [25], [58]. In general, these methods directly operated on a continuous video frame sequence with recurrent neural networks [4], [25], [55] or 3D ConvNets [24], [58]. Although these methods0162-8828 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information. This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ TPAMI.2018.2868668, IEEE Transactions on Pattern Analysis and Machine Intelligence 4 aim to deal with longer

video duration, they usually process sequences of fixed lengths ranging from 5 to 120 frames due to the limit of computational cost and GPU memory. It is still non-trivial for these methods to learn from the entire video due to their limited temporal coverage. Our method differs from these end-to-end deep ConvNets by its novel adoption of a sparse temporal sampling strategy, which enables efficient learning using the entire videos without the limitation of sequence length. Therefore, our temporal segment network is a video-level and end-to-end framework for temporal structure modeling on the entire video. 3

3.3 Temporal Segment Networks

In this section, we give a detailed description of our temporal segment network framework. Specifically, we first discuss the motivation of segment based sampling. Then, we introduce the architecture of temporal segment network framework. After this, we present several aggregating functions of temporal segment network and provide analysis on these functions. Finally, we investigate several practical issues for the instantiation of temporal segment network framework. 3.1 Segment Based Sampling As discussed in Sec. 1, long-range temporal modeling is important for action understanding in videos. The existing deep architectures such as two-stream ConvNets [1] and 3D convolutional networks [16] are designed to operate on a single frame or a stack of frames (e.g., 16 frames) with limited temporal durations. Therefore, these structures lack capacity of incorporating long-range temporal information of videos into the learning of action models. In order to model long-range temporal structures, several approaches have been proposed to stack more consecutive frames at a fixed sampling rate [4], [24], [58]. Although this dense and local sampling could help to relieve the problem of the original short-term CovNets [1], [16], it still suffers in both computational and modeling aspects. From the computational perspective, it would greatly increase the cost of ConvNet training, as this dense sampling usually requires a large number of frames to capture long-range structures. For example, it totally samples 100 frames in the work of [24] and 120 frames for the method of [4]. From the modeling perspective, its temporal coverage is still local and limited by its fixed sampling interval, failing to capture the visual content over the entire video. For instance, the sampled 100 frames [24] only occupy a small portion of a 10-second video (around 300 frames). We observe that although the frames are densely recorded in the videos, the content changes relatively slowly. This motivates us to explore a new paradigm for temporal structure modeling, called segment based sampling. This strategy is essentially a kind of sparse and global sampling method. Concerning the property of spareness, only a small number of sparsely sampled snippets would be used to model the temporal structures in a human action. Normally, the number of sampled frames for one training iteration is fixed to a predefined value independent of the durations of the videos. This guarantees that the computational cost will be constant, regardless of the temporal range we are dealing with. On the global property, our segment based sampling ensures these sampled snippets would distribute uniformly along the temporal dimension. Therefore, no matter how long the action videos will last for, our sampled snippets would always roughly cover the visual content of whole video, enabling us to model the long-range temporal structure throughout the entire video. Based on this paradigm for temporal structure modeling, we propose temporal segment network, a video-level training framework as shown in Figure 1, which would be explained in the next subsection.

4. Conclusion

1) In the preprocessing phase, linear sampling was used to normalize the temporal dimension of hand gesture samples. For spatial dimension normalization,
2) The detected face and human body part ratios. Thenused 3DCNN for feature learning in two approaches
3) In the first approach, a single 3DCNN instance was trained to extract the hand gesture features from the entire video.
4) In the second approach, three instances of the 3DCNN structure were trained to extract the hand gesture features from the beginning, middle, and end of the video sample
5) Temporal Segment Network (TSN), a video-level framework that aims to model long range temporal structure
6) The former provides an effective and efficient way to capture long-range temporal structure, while the latter makes it possible to train very deep networks on a limited training set without severe over-fitting

References

[1] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in NIPS, 2014, pp. 568–576.
[2] H. Wang and C. Schmid, “Action recognition with improvedtrajectories,” in ICCV, 2013, pp. 3551–3558.
[3] L. Wang, Y. Qiao, and X. Tang, “Motionlets: Mid-level 3D parts forhuman motion recognition,” in CVPR, 2013, pp. 2674–2681.
[4] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals,R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification,” in CVPR, 2015, pp. 4694–4702.
[5] L. Wang, Y. Qiao, and X. Tang, “Action recognition with trajectorypooled deep-convolutional descriptors,” in CVPR, 2015, pp. 4305–4314.
[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-basedlearning applied to document recognition,” Proceedings of the IEEE,vol. 86, no. 11, pp. 2278–2324, 1998.
[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in NIPS, 2012,pp. 1106–1114.
[8] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in ICLR, 2015, pp. 1–14.
[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper withconvolutions,” in CVPR, 2015, pp. 1–9.

[10] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learn-ing deep features for scene recognition using places database," inNIPS, 2014, pp. 487–495.
[11] D. Bragg, O. Koller, M. Bellard, L. Berke, P. Boudrealt, A. Braffort, N. Caselli, M. Huenerfauth, H. Kacorri, T. Verhoef, and C. Vogler, ''Sign language recognition, generation, and translation: An interdisciplinary perspective,'' 2019, arXiv:1908.08597. [Online]. Available: https://arxiv.org/ abs/1908.08597
[12] G. O. Young, ''Synthetic structure of industrial plastics,'' in Plastics, vol. 3, J. Peters, Ed., 2nd ed. New York, NY, USA: McGraw-Hill, 1964, pp. 15–64.
[13] L.-C. Wang, R. Wang, D.-H. Kong, and B.-C. Yin, ''Similarity assessment model for Chinese sign language videos,'' IEEE Trans. Multimedia, vol. 16, no. 3, pp. 751–761, Apr. 2014.
[14] L. Pigou, A. van den Oord, S. Dieleman, M. Van Herreweghe, and J. Dambre, ''Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video,'' 2015, arXiv:1506.01911. [Online]. Available: https://arxiv.org/abs/1506.01911
[15] Y. X. Molchanov, S. Gupta, K. Kim, S. Tyree, and J. Kautz, ''Online detection and classification of dynamic hand gestures with recurrent 3D convolutional neural network,'' in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, USA, Jun. 2016, pp. 4207–4215.
[16] L. Pigou, S. Dieleman, P. J. Kindermans, and B. Schrauwen, ''Sign language recognition using convolutional neural networks,'' in Proc. Workshop Eur. Conf. Comput. Vis., Zurich, Switzerland, 2014, pp. 572–578.
[17] T. Liu, W. Zhou, and H. Li, ''Sign language recognition with long shortterm memory,'' in Proc. IEEE Int. Conf. Image Process. (ICIP), Phoenix, AZ, USA, Sep. 2016, pp. 2871–2875.
[18] X. Li, C. Mao, S. Huang, and Z. Ye, ''Chinese sign language recognition based on SHS descriptor and encoder-decoder LSTM model,'' in Proc. Chin. Conf. Biometric Recognition Cham, Switzerland: Springer, 2017, pp. 719–728.
[19] S. Huang, C. Mao, J. Tao, and Z. Ye, ''A novel Chinese sign language recognition method based on keyframe-centered clips,'' IEEE Signal Process. Lett., vol. 25, no. 3, pp. 442–446, Mar. 201
[20] H. Brashear, ''Improving the efficacy of automated sign language practice tools,'' ACM SIGACCESS Accessibility Comput., vol. 89, no. 1, pp. 11–17,Sep. 2007.
[21] Y. Bouzid, M. A. Khenissi, F. Essalmi, and M. Jemni, ''Using educational games for sign language learning—A signwriting learning game: Case study,'' Educ. Technol. Soc., vol. 19, no. 1, pp. 129–141, Jan. 2016.
[22] C.-H. Chuan and C. A. Guardino, ''Designing smartsignplay: An interactive and intelligent american sign language app for children who are deaf or hard of hearing and their families,'' in Proc. 21st Int. Conf.Intell. User Interfaces. New York, NY, USA: ACM, 2016, pp. 45–48.doi: 10.1145/2876456.2879483.
[23] J. Gameiro, T. Cardoso, and Y. Rybarczyk, ''Kinect-sign: Teaching sign language to 'listeners' through a game,'' in Proc. Int. Summer Workshop Multimodal Inter. New York, NY, USA, Springer, 2013, pp. 141–159.
[24] N. Adamo-Villani, E. Carpenter, and L. Arns, ''An immersive virtual environment for learning sign language mathematics,'' in Proc. ACMSIGGRAPH Educators Program, Jul. 2006, p. 20.
[25] E. Efthimiou et al., ''Sign language recognition, generation, and modelling: A research effort with applications in deaf communication,'' in Proc. 4th Workshop Represent. Process. Sign Lang. Corpora Sign Lang.Technol., 2010, pp. 80–83.
[26] P. Dreuw et al., The signspeak project—Bridging the gap between signers and speakers,'' in Proc. LREC, 2010, pp. 476–481. [Online].Available https://repository.ubn.ru.nl/handle/2066/85929
[27] J. A. Bangham et al., ''Virtual signing: Capture, animation, storage and transmission-an overview of the visicast project,'' Tech. Rep., 2000.
[28] M. J. Cheok, Z. Omar, and M. H. Jaward, ''A review of hand gesture and sign language recognition techniques,'' Int. J. Mach. Learn. Cybern.,vol. 10, no. 1, pp. 131–153, Jan. 2017.
[29] L. Quesada and G. Lòpez, and L. Guerrero, ''Automatic recognition of the american sign language fingerspelling alphabet to assist people living with speech or hearing impairments,'' J. Ambient Intell. Humanized Computer.,vol. 8, no. 4, pp. 625–635, Aug. 2017.
[30] P. Kumar, H. Gauba, P. P. Roy, and D. P. Dogra, ''A multimodal framework for sensor based sign language recognition,'' Neurocomputing, vol. 259,pp. 21–38, Oct. 2017