Dynamic Hand Gesture Recognition Based on Short-Term Sampling Neural Networks

Wenjin Zhang, Jiacun Wang, *Senior Member, IEEE*, and Fangping Lan

***Abstract*—Hand gestures are a natural way for human-robot interaction. Vision based dynamic hand gesture recognition has become a hot research topic due to its various applications. This paper presents a novel deep learning network for hand gesture recognition. The network integrates several well-proved modules together to learn both short-term and long-term features from video inputs and meanwhile avoid intensive computation. To learn short-term features, each video input is segmented into a fixed number of frame groups. A frame is randomly selected from each group and represented as an RGB image as well as an optical flow snapshot. These two entities are fused and fed into a convolutional neural network (ConvNet) for feature extraction. The ConvNets for all groups share parameters. To learn long-term features, outputs from all ConvNets are fed into a long short-term memory (LSTM) network, by which a final classification result is predicted. The new model has been tested with two popular hand gesture datasets, namely the Jester dataset and Nvidia dataset. Comparing with other models, our model produced very competitive results. The robustness of the new model has also been proved with an augmented dataset with enhanced diversity of hand gestures.**

***Index Terms*—Convolutional neural network (ConvNet), hand gesture recognition, long short-term memory (LSTM) network, short-term sampling, transfer learning.**

I. Introduction

A human gesture is a state or sequence of states of a special part of a human body, especially the face and hands. Among them, hand gestures are widely used to replace the spoken language for communication. For example, American sign language is a well-developed body language based on hand gestures [1], [2]. With the rapid advances in information technology and computer science, touchless human-computer interaction has been a hot research topic and so is hand gesture recognition [3]. Based on input devices, existing hand gesture recognition techniques can be classified into two categories: non-vision-based and vision-based [4], [5]. Non-vision-based approaches usually use wearable equipment with certain types of optical or mechanical sensors, such as accelerometers and gyroscopes, which translates hand motion into electrical signals. Although these sensors can provide high recognition accuracy, users have to wear them all the time. Moreover, they cannot produce any signal on static hand gestures. On the other hand, vision-based approaches use single or multiple cameras or precise body motion sensors such as Leap Motion controllers and Microsoft Kinect [6]–[9]. Hand gestures can be recognized as long as they are visible to the cameras or sensors, regardless of whether they are dynamic or static. Vision-based approaches are thus more practical than non-vision based ones. Nowadays, most researchers focus on vision-based techniques.

Hand gesture recognition is a branch of action recognition that is hindered by two major obstacles. One is how to efficiently gain the spatial and temporal information from video inputs. In the past years, the most popular network architecture is the two-stream convolutional neural network, proposed by K. Simonyan and A. Zisserman in 2014. In this architecture, spatial features and temporal features are learned by two separate networks and are then fused using support vector machines (SVM) [10]. A lot of attempts in improving the spatial and temporal learning performance were evolved from this design [11]–[15]. However, the use of two separate learning networks means at least doubled training parameters and computational cost compared to using a single network. Moreover, fusing sampled frames with SVM does not catch long-range temporal information across consecutive sampled frames.

Wang *et al.* developed temporal segment networks (TSN) in [15], [16] for action recognition. In the TSN model, each input video sample is divided into a number of segments and a short snippet is randomly selected from each segment. The snippets are represented by modalities such as RGB frames, optical flow and RGB differences. Convolutional neural networks (ConvNets) that are used to learn these snippets all share parameters. The class scores of different snippets are fused by the segmental consensus function to yield segmental consensus, which is a video-level prediction. Predictions from all modalities are fused to produce the final prediction. Experiment showed that the model not only achieved very good action recognition accuracy but also maintained reasonable computation cost.

Motivated and inspired by the above observations and achievements, this study proposed a new network architecture for hand gesture recognition, called *short-term sampling neural network (STSNN)*. The STSNN architecture, like TSNs, uses a single ConvNet component to capture both short-term spatial information and temporal information from color and optical flow, respectively. It then uses a long-short term

Manuscript received April 8, 2020; revised July 20, 2020; accepted August 13, 2020. Recommended by Associate Editor MengChu Zhou. *(Corresponding author: Wenjin Zhang.)*

Citation: W. J. Zhang, J. C. Wang, and F. P. Lan, "Dynamic hand gesture recognition based on short-term sampling neural networks," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 1, pp. 110–120, Jan. 2021.

The authors are with the Department of Computer Science and Software Engineering, Monmouth University, New Jersey 07740 USA (e-mail: s1260807@monmouth.edu; jwang@monmouth.edu; fangpinglan0116@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2020.1003465

memory network (LSTM) [17] to capture long-term temporal information. Similar to TSNs, the STSSN framework first divides the RGB video and its optical flow each into a fixed number of video frame groups. One representative sample is randomly selected from each group. The two samples from the RGB frame group and the corresponding optical flow group are fused into one sample. Then this architecture captures features from these representative samples through ConvNets. After that, features learned from each ConvNet are fed into an LSTM, where the long-range features are learned. In the end, the LSTM produces hand gesture classification from features learned through each LSTM component.

To deal with dataset limitation, we proposed a novel dataset enhancement approach, which is different from normal ones, such as horizontal flip and vertical flip that may affect hand gesture type. In our approach, we zoom out the original video with random distances to generate the "zoomed-out" dataset. It not only provides more training data samples but also helps verify the robustness of a trained model. In addition, random sampling in each group and stacking optical flow frames on RGB frames could also introduce the data diversity because different sampling representatives from different groups can form many different combinations of inputs.

Experiments have been performed on Jester dataset, "zoomed-out" Jester dataset and Nvidia dataset to evaluate the effectiveness of our STSSN model for hand gesture recognition. To reduce the training cost and ensure the ConvNet architecture capture the features from the input effectively, we introduce the pre-trained Inception V2 as our feature extraction module. STSNN with Inception V2 obtained the best performance in the experiment on Jester and Jester "zoomed-out" dataset with the testing accuracies of 95.73% and 95.69%, respectively. On the Nvidia dataset, it achieved a great performance with the classification accuracy of 85.13%, which outperforms the most popular models.

The main contribution of the paper is two-fold:

1) It designed a new deep learning neural network model that integrates several state-of-the-art techniques for action recognition to tackle the complexity and performance issues in dynamic hand gesture recognition. Short-term video sampling, feature fusion, ConvNets with transfer learning and LSTMs are the key components of the new model.

2) It developed a novel approach to "zoom-out" the existing dataset to increase the diversity of the dataset and thus ensure the robustness of a trained model.

The rest of the paper is organized as follows: Section II briefly reviews related work in gesture recognition, including ConvNets, LSTM networks, and action recognition techniques. Section III presents the proposed STSNN architecture. Section IV discusses the experiments and results. A summary of the paper and future work is presented in Section V.

II. Related Work

A. Convolutional Neural Networks

Recent years have seen more and more applications of ConvNets in the domain of computer vision. ConvNets have achieved great success in object recognition, motion analysis, scene reconstruction, and image restoration. Lecun *et al.* proposed the standard convolutional neural networks called "LeNet-5" and applied this network in the recognition of handwritten digits, achieving an average accuracy of 99.3% on the MNIST Dataset in 1998 [18]. Simard *et al.*, Schölkopf *et al.*, and Ciregan *et al.* optimized Lecun's approach and improved the accuracy to 99.6%, 99.61%, and 99.64%, respectively [19]–[21]. Bong *et al.* proposed a low-power neural network and developed a ConvNets-based face recognition system [22]. He *et al.* presented a residual learning framework to improve the ConvNets' performance of image recognition and won the first price in the area of detecting images, locating images, detecting COCO and segmenting COCO in the ILSVRC&COCO 2015 Competitions 1 [23]. According to the literature review, it is obvious that ConvNets are the best choice for computer vision at present.

B. Long Short-Term Memory Networks

Long short-term memory neural networks, LSTMs, are a variation of recurrent neural networks, RNNs. RNNs are designed to process sequences of inputs where all the inputs are related to each other. The key component of RNN architecture is the loop that could persist the previous information with the internal state memory as shown in Fig. 1. x and o are input and output, respectively. h represents the hidden state of each time step and t is the index of time sequence. The state of the current step could be passed to the next step by using this loop. However, this design always encounters the exploding and vanishing gradient problem when training a traditional RNN. LSTMs are designed to solve this problem. A common architecture of an LSTM block is composed of three gates: input gate, output gate, and forget gate, plus a memory cell as shown in Fig. 2. Among them, the forget gate could control the value remaining in the cell [24].

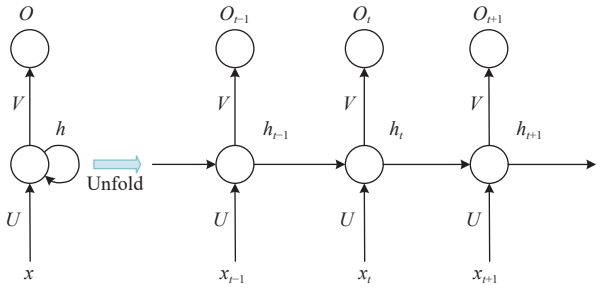

Fig. 1. The architecture of recurrent neural network.

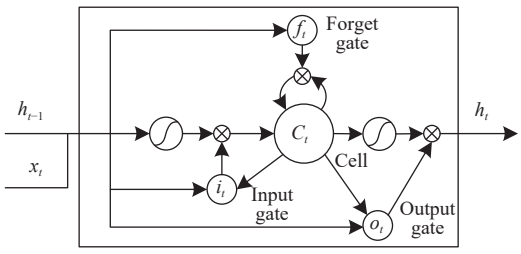

Fig. 2. The architecture of LSTM block.

LSTMs have achieved great performance in some areas of sequence data processing in the past years. For example, Microsoft used "dialog session-based long-short-term memory" approach to reach 95.1% recognition accuracy on switchboard corpus with 165 000 words vocabulary [25]. Google recognizes the speech by using LSTM on smartphone for Google Translate and smart assistant [26], [27]. Apple also used it for Sifi [28]. Amazon applied this approach to the product Alexa [29]. In 2019, a computer program AlphaStar based on a deep LSTM core, developed by DeepMind, excelled at the video game Starcraft II [30].

C. Action Recognition

Dynamic hand gesture recognition can be viewed as a branch of action recognition, whose task is to identify different actions by acquiring the context from a whole video of action rather than from each frame [31]. This means that action recognition needs to identify the different state of action from each 2D frame where the action may or may not be performed and then synthesize the obtained state to determine what the action is.

Although deep learning has achieved great success in the area of image classification in the past years, the progress of video classification research is slow. The major issues are with the standard yet defect benchmark and huge computational cost. UCF101 and Sports1M datasets have been recognized as the standard benchmark by researchers for a long time, but there are some problems with this benchmark. For UCF101, although the quantity of data could meet the requirement, the actual diversity of this dataset is not sufficient because of the high spatial correlation. Besides, both of them have the same theme, which is sports. It is difficult to prove that an approach that works well in this theme could be generalized into other tasks. Another problem is the huge computational cost. Video data can be considered as an image sequence, which means that video analysis is equal to process a large number of images. Therefore, video processing takes n times of the computational cost of a single image processing, where n is the number of video frames. According to Tran *et al.*, they spent three to four days to train a deep learning model on UCF101 and about two months to train on the Sports-1M dataset [32].

There are mainly two categories of deep learning networks for action recognition: single-stream networks and two-stream networks. The major difference between these two categories is the design choice around combining spatiotemporal information. In 2014, Karpathy *et al.* explored multiple ways to use a single-frame baseline architecture with pre-trained 2D ConvNets to fuse the temporal information from consecutive frames [33]. The architecture is illustrated in Fig. 3.

Considering the difficulty of deep neural networks in learning movement features, Simmoyan and Zisserman suggested using stacked optical flow to learn temporal features [10]. This architecture is shown in Fig. 4 and it has two separate neural networks. Among them, one is designed for spatial context from RGB frames and another one is used to learn motion context from optical flow. The author trained the two networks separately and fused them using a support vector machine.

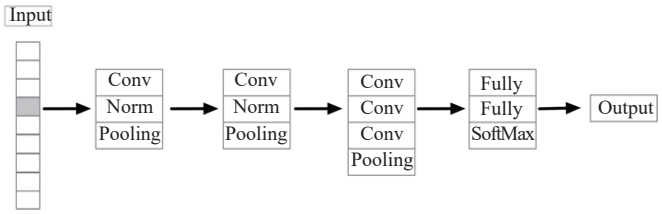

Fig. 3. Single-frame baseline architecture for action recognition. Conv, norm, pooling, and fully stand for convolution, normalization, pooling, and fully connected layers, respectively.

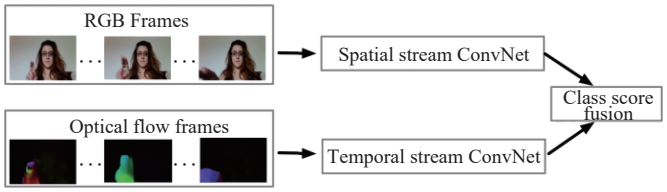

Fig. 4. Two stream architecture for action recognition.

D. Hand Gesture Recognition

A lot of vision-based approaches for recognizing hand gestures with normal cameras or depth cameras, such as Leap Motion and Kinect controllers, were reported. Among them with normal cameras, Tsai *et al.* proposed a synthetically-trained neural network for 3D hand gestures [34]. An end-to-end deep neural network, LPSNet, was proposed by Li *et al.* to recognize hand gestures [35]. Köpüklü *et al.* presented a new data level fusion method, motion fused frames (MFFs), to increase the accuracy of deep learning in [36]. In 2019, Köpüklü *et al.* attempted to use a lightweight CNN and a deep CNN to detect and classify hand gestures, respectively [37]. There are also some other works based on depth cameras [38]. Hu *et al.* trained deep learning neural networks with skeleton data collected from a Leap Motion controller and then used it to control a drone [8]. Marin *et al.* used an SVM with a Gaussian radial basis function kernel to recognize gestures with the combination of skeletal data and the depth feature from Kinect and Leap Motion controllers [9].

III. Short-Term Sampling Neural Networks

In this section, we give a detailed description of the short-term sampling neural networks, STSNN. Firstly, we explain the rationale for this design. Then, we present the architecture of STSNN in detail. After that, we introduce the transfer learning technique into STSNN to increase model training efficiency.

A. Overview of STSNN

To perform hand gesture recognition effectively, we need to capture and utilize both spatial and temporal information in a video input. Both 3D ConvNets and two-stream ConvNet architectures try to fetch the information from a stack of RGB frames and optical flow [10], [39]. However, in our previous study for hand gesture recognition, we trained a 3D ConvNet [40] model with more than 146 million parameters. Such a

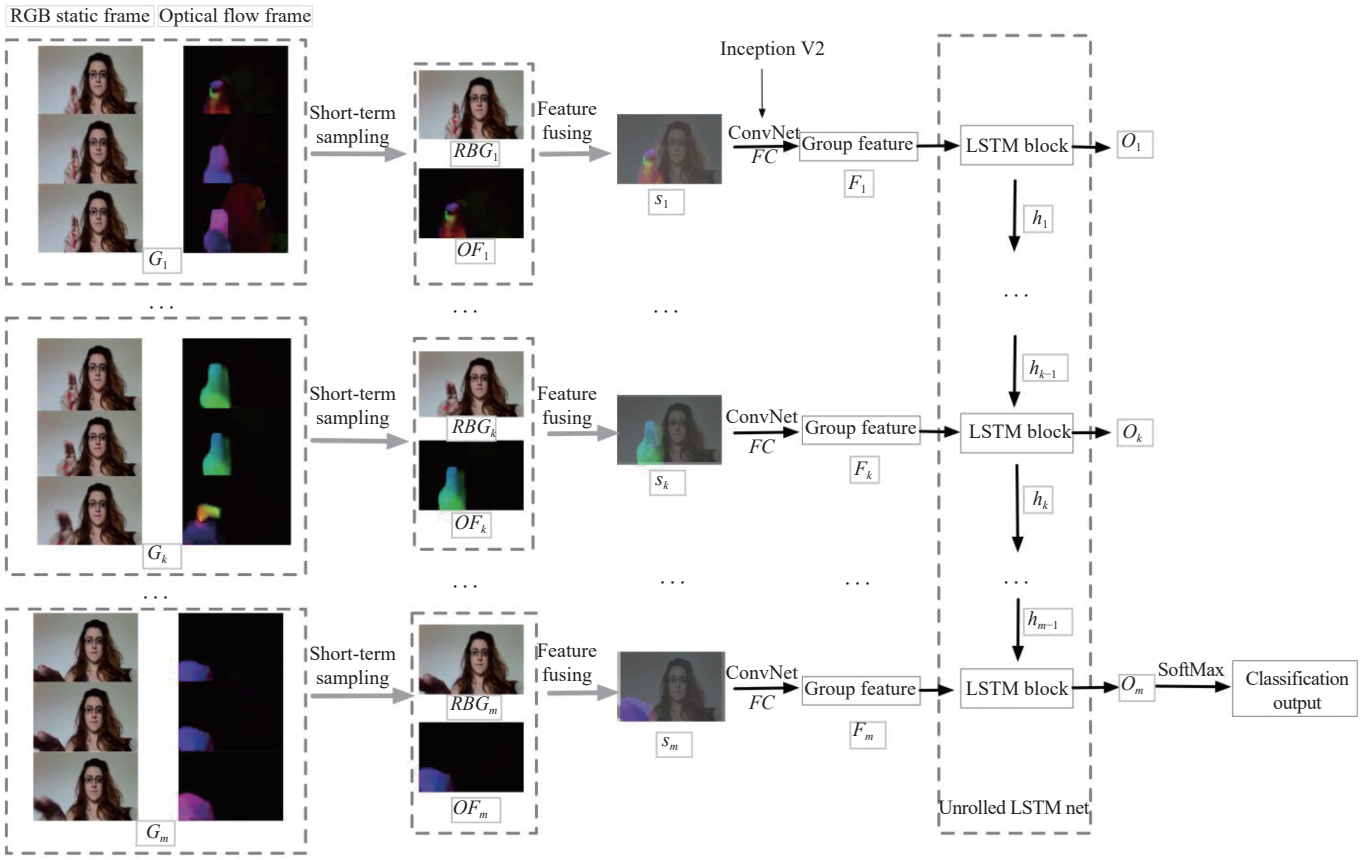

Fig. 5. The architecture of short-term sampling neural networks: It divides the input video frames and optical frames to *m* groups and the representative color and optical flow samples are randomly selected from each group. And then, the group sample is generated by combining the color and optical flow frame. ConvNets module is used to fetch the group features. The different group features collected by the FC module flow into the LSTM module for classification.

complicated model is hard to deploy into an embedded system. Therefore, the goal of this design is to reduce the neural network computing cost and meanwhile ensure recognition accuracy and performance.

Because in an action video frames are densely recorded but the content changes slowly across the frames, sampling frames at a fixed rate has been proposed to reduce computing cost [41]. We adopted this approach in our previous study [40]. Although it is helpful to some extent, this approach ignores the diversity of video length: different hand gesture video samples have different durations. For example, the "swiping left" hand gesture lasts much longer than the "stop sign" gesture. In this case, if the two gesture video clips are sampled at the same rate, we will get more frames from the "swiping left" gesture than from the "stop sign" gesture, and then either padding or trimming will have to be used.

We use a group based short-term sampling approach, in which we divide consecutive video frames into a *fixed* number of groups and then take a sample from each group randomly. This way, we can get a fixed number of samples for each video sample, regardless of its duration. In addition, this group sampling approach can make sure that the features are learned fairly because all the samples are distributed uniformly along with the time. In order to ensure that the model can capture the spatial and short-term temporal information together, the representative sample of each group is the combination of RGB frame and optical flow frame, inspired by the early-fusion idea, introduced by Zhang *et al.* [40]. After that, a ConvNet is used to capture the features from each representative sample. All ConvNets share the same parameters to reduce training parameters. Next, an LSTM module is used to learn the long-range temporal features from the feature sequence obtained from ConvNets. In the end, the SoftMax function is chosen to calculate the predicted probability of classes.

B. The STSNN Architecture

In this section, we give a detailed description of the STSNN architecture. This architecture consists of four modules: a module for group based short-term sampling, a ConvNet for feature extraction, an LSTM module for the long-range temporal features learning, and an output layer for hand gesture classification. The overall architecture is illustrated in Fig. 5.

Denote by N the total number of frames in a video input. We divide them into m groups, which are denoted by G_1, G_2, …, G_m. The number of frames in each group is

$$N_g = \left\lfloor \frac{N}{m} \right\rfloor.$$

As illustrated in Fig. 6, from each group G_k, a sample $s_k \in R^{w \times h \times 6}$ is produced by fusing an RGB static frame sample $RBG_k \in R^{w \times h \times 3}$ and an optical flow frame sample

$OF_k \in R^{w\times h\times 3}$ using the following formula:

$$s_k \in R^{w\times h\times 6} = cat(3, RGB_k, OF_k)$$

where *cat*(*dim, A, B*) concatenates *B* to the end of *A* along the *dim*-th dimension, *k* is the index of a group, *w* and *h* are the width and height of a frame, respectively.

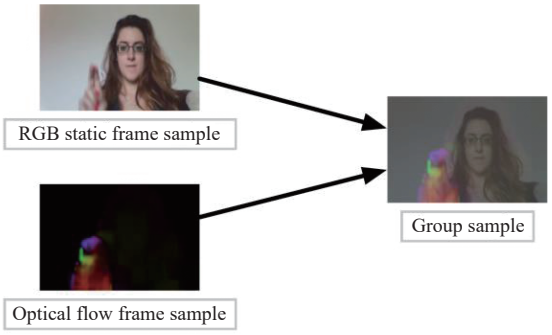

Fig. 6. The process of fusing group sample.

The group feature F_k, fetched from the sample s_k by the ConvNet filter, is described as

$$F_k = f_{FC}(f_{\text{ConvNet}}(s_k, W))$$

where *W* is the parameters of ConvNet and they are shared by all groups' filters. That saves computation costs significantly. With a standard 3D-CNN neural network that performs 3D convolutions on the frame sequence, if its filter size and the number of output feature maps are same as that are used in STSNN, the number of training parameters in the first layer of the 3D CNN will be $FS \times FS \times FM$ times of that of STSNN [41], where *FS* is the filter size and *FM* denotes the number of feature maps. One fully connected layer, f_{FC}, is used to transform the feature maps from ConvNet to a fixed dimension of vector and then we could get *m* feature vectors, $R^{m\times d}$, where *d* is the dimension of a feature. The LSTM input is generated by stacking these *m* feature vectors together. The LSTM module is used to fuse the group features and learn the long-range temporal information. It is a recurrent process that the hidden state h_k and output O_k of the LSTM block is calculated based on previous hidden state h_{k-1} and current LSTM block input F_k, that is

$$(h_k, O_k) = f_{\text{LSTMblock}}(h_{k-1}, F_k), 1 < k \le m$$

When $k = 1$,

$$(h_1, O_1) = f_{\text{LSTMblock}}(F_1).$$

Thus, the output O_m of the last LSTM block represents the consensus result from all the group features. The final predicted class probability is

$$\hat{y} = f_h(O_m).$$

The consensus data are passed to f_h, a hypothesis function unit, which calculates the final probability of each classification. In this study, we choose the SoftMax function as the hypothesis function. The benefit of using SoftMax long-term temporal features from the sequence of short-term group features. However, its corresponding consensus function f_{LSTM} must be differentiable or have sub-gradient, because it should support gradient descent optimization algorithm for backpropagation. The work of Feichtenhofer *et al.* showed that the gradient-based optimization approach could update the parameters from the loss of the whole video rather than just one group [13]. Assuming that the consensus function output is *P*,

$$P = O_m = f_{\text{LSTM}}(f_{FC}(f_{\text{ConvNet}}(s_1, W)), \ldots, f_{FC}(f_{\text{ConvNet}}(s_k, W)), \ldots, f_{FC}(f_{\text{ConvNet}}(s_m, W)))$$

and a standard categorical cross-entropy loss is used, then the final loss function, $\mathcal{J}$, is defined as

$$\mathcal{J}(y, P) = -\sum_{i=1}^{C} y_i (p_i - \log \sum_{j=1}^{C} e^{p_j})$$

where *y* is the actual probability distribution, y_i is the probability of the *i*-th class, *C* is the number of hand gesture classes, and p_i is the *i*-th dimension of *P*. According to the gradient descent parameter updating formula, the change of the parameters in one training iteration is related to the gradients of the loss $\mathcal{J}$ concerning the parameters to be trained, as follows:

$$\frac{\partial \mathcal{J}(y, P)}{\partial W} = \frac{\partial \mathcal{J}}{\partial P} \sum_{k=1}^{m} \frac{\partial P}{\partial f_{\text{ConvNet}}(s_k, W)} \frac{\partial f_{ConvNet}(s_k, W)}{\partial W}$$

where *m* is the number of groups.

C. Transfer Learning With Inception V2

Transfer learning plays a very important role in improving training performance and saving training costs. According to Pan *et al.*, an image classification model designed for one domain may apply to another domain because their data may have the same feature and distribution [42]. A good practice of transfer learning is to extract features by using ConvNets. Thus, we choose the pre-trained model of Inception architecture with Batch Normalization, Inception V2. Inception V2 not only has a good balance between efficiency and accuracy, its batch normalization can also reduce the risk of over-fitting. These features lead to better testing performance through better generalization [43]. The dropout layer is another technique to reduce the overfitting problem. One extra dropout layer was added after the pooling layer in Inception and the dropout rate was set to 0.5 [44].

IV. Model Training and Testing

This section introduces the details of model training and testing results. We first discuss the hand gesture datasets and how we enhanced the Jester dataset to improve the robustness of the trained model. Then, we go over the train details, such as optimization function, learning rate, and so on. In the end, we compare the performance of our model with other popular models.

A. Experiment System Overview

As Fig. 7 shows, the architecture of the system includes three components: the input module, the preprocessing module and the recognition module. The input module starts to capture the image sequence of a gesture at the rate of 12 frames per second when a user presses the "space" key. This capturing is stopped when the user presses the key again. The

user should finish a hand gesture between the two consecutive keystrokes. It then uploads the image sequence into the preprocessing module asynchronously during, rather than after, the video input process. The responsibility of the preprocessing module is to calculate the optical flow and normalize the data. When the data flow into the recognition module, a prediction result will be produced. A web service is developed to share the recognition result with other applications to control engineering targets, such as robots and drones.

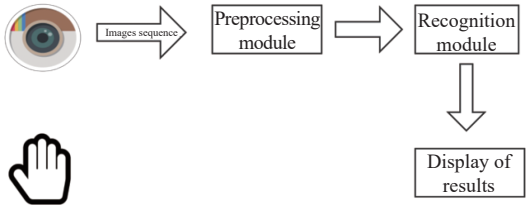

Fig. 7. Experiment system architecture.

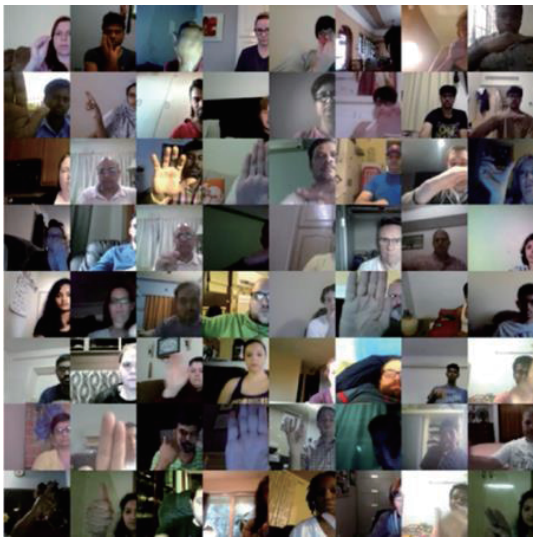

Fig. 8. Some examples of hand gesture video clips from the Jester dataset.

B. Datasets

1) The Jester dataset

In this study, the 20BN-Jester dataset, which was collected and organized by Twenty BN, was chosen to train and test our model [45]. This dataset is a large collection of labeled video clips that show humans hand gestures collected by using the laptop camera or webcam. All the hand gestures were created by a lot of crowd workers rather than very few people. Besides, the gestures were performed in a very complex background such as a rotating fan, bright bulb, moving cat and so on. As Fig. 8 shows, there is a large variance of the appearance of people, and complex ambient occlusion and background scenes. Because of that, this dataset is a good choice to train machine learning models for hand gesture recognition. There are 148 092 videos in this dataset: 118 562 for training, 14 787 for validation and 14 743 for testing, which is provided as TGZ archive. Each video was converted into JPG images at the rate of 12 frames per second. So, the archive fold has 148 092 directories that contain these images. There are 27 types of hand gestures listed in Table I. Among them, 25 are gesture classes. The "No gesture" and "Doing other things" classes are special classes that could not be recognized as any hand gestures. "No gesture" means that a user sits or stands still without any movement and "Doing other things" is a collection of various activities exception 25 hands gesture included in the dataset. Thus, based on this dataset, a well-trained model should be able to handle these two exceptions.

2) The Zoomed-Out Jester Dataset

In order to test whether this design has the ability to handle situations that there is a bigger distance between video cameras and actors who perform hand gestures than usual. This distance of the samples in the Jester dataset is about 50 cm, meaning actors sat right in front of the cameras. That is the limitation of this dataset for a specific application. When we want to apply our approach to a real application, we cannot assume that users always do the hand gesture close to the webcam or camera. This motivates us to enhance this dataset by "zooming out" video frames in the dataset to create "distant" video samples, in which the distance reaches to 100 cm.

TABLE I
CLASS OVERVIEW IN JESTER DATASET

Class no.	Gesture description	Number of samples
1	Swiping down	5303
2	Swiping left	5160
3	Swiping right	5066
4	Swiping up	5240
5	Thumb down	5460
6	Thumb up	5457
7	Turning hand clockwise	3980
8	Turning hand counterclockwise	4181
9	Zooming in with full hand	5307
10	Zooming in with two fingers	5355
11	Zooming our with full hand	5330
12	Zooming out with two fingers	5379
13	Stop sign	5413
14	Sliding two figs. up	5262
15	Sliding two figs. right	5244
16	Sliding two figs. left	5345
17	Sliding two figs. down	5410
18	Shaking hand	5314
19	Rolling hand forward	5165
20	Rolling hand backward	5031
21	Pushing two figs. away	5031
22	Pushing hand away	5434
23	Pulling two figs. in	5315
24	Pulling hand in	5379
25	No gesture	5344
26	Drumming fingers	5444
27	Doing other things	12 416

In order to zoom out the Jester dataset, we take into consideration the fact that adjacent pixels in a picture tend to have strong similarities, especially the background pixels, like white walls or sofa. Thus, our approach is to duplicate boundary pixels of a frame. Of course, such zooming-out processing should not affect the gesture in a video clip. In other words, the boundary pixels to be duplicated in a frame are not any pixels that will change in the following frames. As mentioned before, the optical flow algorithm could detect the movement between consecutive frames. Our experiment shows that 31.4% of moving pixels are located at the boundaries of video frames. One solution is to find a frame in which all boundary pixels are safe to duplicate and then use these pixels to extend the background of all frames in the video. To modify the dataset so that the distance between actors and cameras varies within a certain range, we set a maximum number of pixels to duplicated, $N_{\max}$, and then we randomly generate a number of duplicated pixels N_{extend} for each video, $N_{\text{extend}}[0, N_{\max}]$. The whole process of zooming out video frames is shown in Fig. 9. The frame in Fig. 9(b) is generated by filling the surrounding area of the original frame in Fig. 9(a) with its boundary pixels. Then the "zoomed-out" frames are generated by resizing it to the same size. Comparing the original frames with "zoomed-out" frames, this approach effectively "increases" the distance between the actors and cameras in Fig. 10.

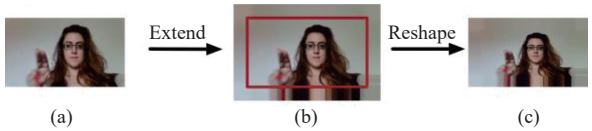

Fig. 9. An example of zooming out a frame.

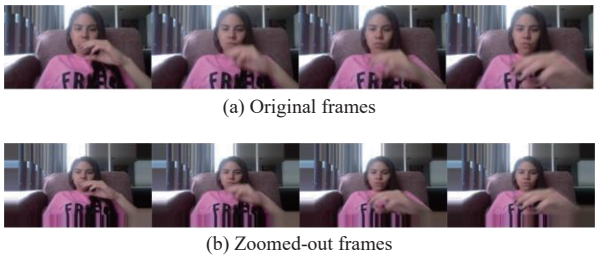

Fig. 10. The comparison between the original dataset and the "zoom out" Jester dataset (a) is the original frame and (b) is the "zoomed-out" frame.

3) The Nvidia Dataset

The Nvidia hand gesture dataset (Nvidia dataset) is collected by using the SoftKinetic DS325 sensor, including 25 hand gesture types. The samples of this dataset are front-video and depth videos and randomly split into training (70%, 1050 videos) and test (30%, 482 videos) sets by subject. In this study, we focus on using the normal camera as the sensor to recognize the hand gesture. Thus, we only take the RGB frames from the videos for the experiment.

C. Training Details

This section introduces the experimental hardware and software environment and hyper-parameters about the learning phase.

1) Environment

The training and recognition system is implemented with Python 3.5, PyTorch 1.0 and Cuda 10.0. OpenCV 3 and pillow 5.3 libraries are chosen to preprocess video inputs because these libraries are friendly to Ubuntu Operation System, especially Cuda 10.0 and OpenCV 3.0.

Hardware for the experiment is as follows: Processor: Intel (R) Core (TM) i7-8700 3.20 GHz; System memory: 16GB (2400 MHz); GPU: NVIDIA Corporation GTX 1070 7 GB.

2) Training Configuration

We use the cross-entropy loss to measure the performance of our classification model and the mini-batch stochastic gradient descent algorithm, mini-batch SGD, to optimize the parameters of the network. The batch size is set to 16 and momentum 0.9. The dropout rate is set to 0.5. The network parameters are initialized with a pre-trained model, Inception V2, from ImageNet. The learning rate is set to 0.001 at the beginning of the experiment and decreases to its 0.1 when the loss fails to decrease compared with previous epoch training. The maximum optimization number of epochs is set to 50.

The number of groups in a video input may affect the model performance. Fewer groups could reduce the computation cost but also impact the accuracy of the model. On the other hand, more groups could increase the computation cost and meanwhile improve the accuracy of classification. Thus, we should find a balance between model accuracy and computation cost by choosing a suitable number of groups. For this purpose, we set the number of groups to 3, 5, 7, 9, 11, 13, and 15 and then compare the classification accuracy with the different number of groups.

3) Data Augmentation

In order to deal with the problem of dataset bias, in addition to zoom out images, two online data augmentation approaches, rotation and crop, are applied to prevent model overfitting and improve the robustness of the trained model. They are applied to each training batch before it is fed to the STSNN model. The parameters of each augmentation approach are randomly selected from a fixed range. The range of rotation degree is from 0 to 45 and the range of crop rate is from 75% to 100%, but the two approaches do not apply to input data where the rotation degree is 0 and crop rate is 100%.

D. Results on the Jester Dataset

1) Change of Loss

During the training process, the network parameters are updated based on the gradient descend algorithm to minimize the cross-entropy loss. As Fig. 11 shows the loss of the model has been on a downward trend, which indicates that the training process is effective.

2) Accuracy

To find the optimal number of groups that every input video should be divided into, we did experiments with numbers 3, 5, 7, 9, 11, 13, and 15 and checked the classification accuracy for each option. The experiment results are shown in Fig. 12. When the group number increased from 3 to 9, the accuracy of the model continued to increase. However, when the group number

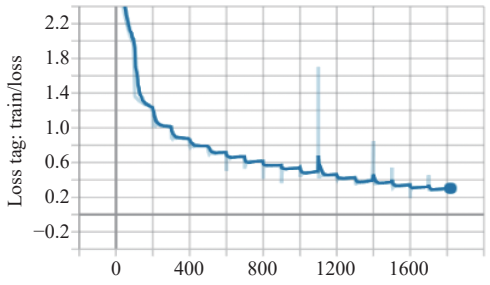

Fig. 11. The change of loss.

increased further, the accuracy pretty much remained unchanged. As we mentioned earlier, the more groups in a video, the higher the computational cost. Thus, our model achieves the best performance with an accuracy of 95.73% when every input video is divided into 9 groups. The best result on the "zoomed-out" Jester dataset is 95.39% in Fig. 13. Comparing with the result of Jester dataset, our model achieves approximately the same accuracy when the group number is 9. However, the model trained by the "zoomed-out" Jester dataset should have better robustness and practicability.

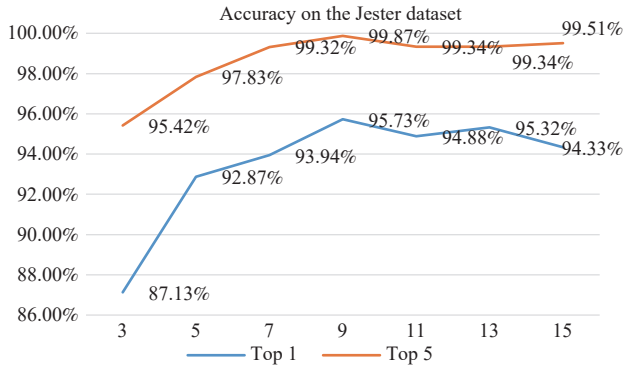

Fig. 12. Result on the validation set of Jester dataset with different group numbers. *x*-axis: number of groups; *y*-axis: accuracy.

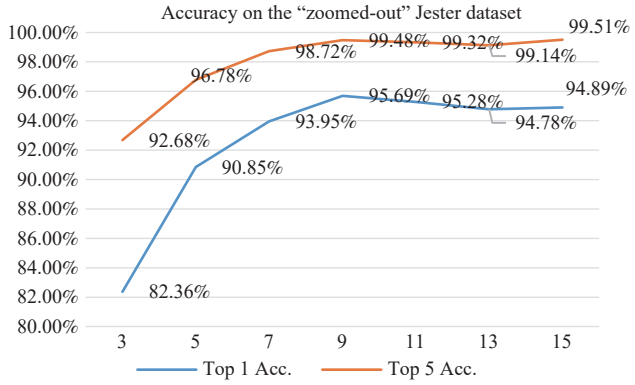

Fig. 13. Result on the validation set of the "zoomed-out" Jester dataset with different group numbers. *x*-axis: number of groups; *y*-axis: accuracy.

Fig. 14 shows the performance confusion matrix of the STSNN model on the Jester validation dataset. The row indices represent the actual labels and the column indices denote the predicted labels. According to this confusion matrix, this model tends to make mistakes in identifying "turning hand clockwise" and "turning hand counterclockwise", "swiping right", and "slide two fingers right", but it still has great classification performance on the whole testing dataset.

3) Comparison With Other Approaches

The Jester website provides a learning board for researchers to test and rank their approaches. So far, about 90 research teams tested their models in this platform, including ours. The recognition accuracy of the top 60 models ranges from 91.22% to 97.26%. The accuracy of our model stands at 95.73%, 1.52% shy from the one ranked the first [46]. Table II shows part of the list. The website does not provide details of each approach and thus it is hard for us to have a full-scale comparison of these approaches. Comparing with the 3D CNN architecture that achieved a 94.85% accuracy, our approach only needs to train a ConvNet architecture that has been pre-trained through ImageNet dataset. Moreover, our approach a higher recognition accuracy than the 3D CNN.

TABLE II
COMPARISON WITH OTHER APPROACHES ON JESTER DATASET

Model	Acc. (%)
MobileNet + NL + SlowFast	97.26
Fusion_TSN_LSTM	97.09
MFFs	96.28
ResNet 101	95.87
3D CNN	94.85
Temporal pyramid relation network	95.34
STSNN	**95.73**

E. Results on the Nvidia Dataset

The Nvidia dataset is relatively small and can lead to the over-fitting problem. To mitigate the risk, we used the transfer learning and dropout layer technique as well. Before training the STSNN in Nvidia dataset, we reused the parameters of the pre-trained model on Jester dataset to initialize the network parameters and then froze all the training parameters of Inception V2 except the first two convolutional and batch normalization layers during the training phase. The number of groups is 9. The initial learning rate is set to 0.001 and decreased to 0.0005 at epoch 25 and 0.00025 at epoch 55, respectively. There were two dropout layers in the model: one is added after Inception V2 component with a drop rate of 0.5 and the other locates after the LSTM component with a drop rate of 0.8.

We also did a comparative experiment that trained the model from scratch. In Table III, we compare our models with state-of-the-art approaches in terms of recognition accuracy, which uses color and optical flow to recognize the hand gesture. Although the accuracy of our model trained from scratch is a little worse than R3DCNN and MFFs, when applied with transfer learning, it achieved great performance on this benchmark with an average accuracy of 85.13%. Nvidia also evaluated human classification accuracy by asking six subjects to label gesture videos in the test set after seeing the color video. Our result is close to human performance, 88.4%.

Actual Label \ Predicted Label	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1	479	11	0	0	0	1	8	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	4
2	12	449	0	1	1	1	0	14	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	5	2
3	2	1	500	3	2	1	0	0	5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	2	2	0
4	0	0	5	476	0	25	0	0	2	6	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0
5	1	0	2	0	532	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	11	0	0	1
6	2	1	0	4	0	500	0	0	0	0	0	4	0	1	0	1	0	0	0	0	1	0	0	1	0	0	1
7	13	1	0	0	0	0	488	3	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
8	3	31	0	0	0	0	3	475	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
9	0	0	14	2	0	0	1	0	503	4	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
10	0	0	1	7	0	0	0	0	17	484	5	4	0	0	0	0	0	0	0	0	0	0	0	0	1	1	2
11	0	1	0	0	3	0	0	2	9	2	506	3	0	0	0	0	0	0	1	1	0	0	0	0	0	1	2
12	0	0	0	2	0	3	3	0	0	3	2	502	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
13	0	0	3	0	0	0	0	0	0	0	0	0	500	18	0	0	0	0	0	0	0	0	0	0	0	0	0
14	1	0	2	0	0	0	0	0	0	1	0	0	15	474	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	1	1	0	3	0	0	0	0	0	1	0	0	322	46	1	0	0	0	0	0	6	0	0	2	2
16	1	1	0	2	0	1	0	0	0	0	0	0	0	0	41	344	0	1	0	0	0	0	6	0	0	0	2
17	0	0	0	0	0	0	0	0	0	1	0	0	0	0	2	1	505	6	9	0	0	0	0	0	0	0	2
18	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	1	6	492	1	20	0	0	1	0	1	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	10	0	506	4	0	0	0	1	0	0	0
20	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	6	5	512	1	0	1	1	0	0	2
21	1	0	0	0	0	0	1	2	0	0	1	0	0	0	0	0	0	0	0	0	532	0	0	0	0	0	2
22	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	531	0	0	0	1	1
23	0	0	0	0	0	1	0	0	0	0	1	0	0	0	2	2	1	0	1	1	1	0	513	2	1	0	2
24	0	1	2	0	5	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	0	0	1	513	0	0	0
25	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0	0	0	0	528	0	1
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	529	4
27	3	4	4	0	1	1	0	0	2	1	2	0	0	0	4	0	1	0	0	2	6	6	7	1	4	10	1409

Fig. 14. Confusion matrix of the STSNN model for hand gesture recognition on the Jester validation dataset.

TABLE III
COMPARISON WITH OTHER METHODS ON NVIDIA DATASET

Model	Modality	Acc. (%)
Two stream CNN [10]	RGB + Optical flow	65.60
IDT [47]		73.40
STSNN without transfer learning		78.94
R3DCNN [48]		79.30
MFFs [36]		84.70
STSNN with transfer learning		85.13
Human [48]	RGB	88.40

F. Feature Map Visualization

In addition to the model accuracy, we also want to know more about the network model itself. Several model visualization approaches have been used by researchers, such as activation maximization, network dissection based visualization network inversion, and deconvolutional neural networks, to interpret ConvNets. In this study, we adopt the deconvolutional neural networks based visualization approach, projecting feature maps into images directly by using DeconvNet architecture. Fig. 15 shows the feature map of our ConvNet model when it processes the "pulling two fig.", "swiping right", and "stop sign" hand gesture videos. It shows that the ConvNets can successfully track the hand movement.

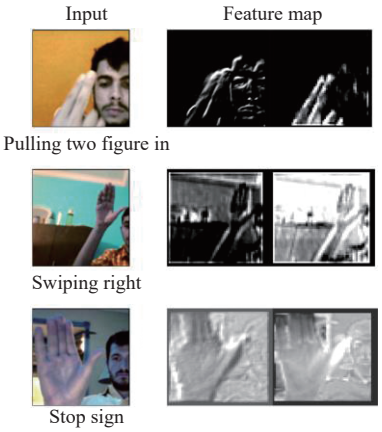

Fig. 15. Visualization of ConvNets model for hand gesture recognition.

V. CONCLUSIONS

This study proposed a short-term sampling neural network for dynamic hand gesture recognition. Each hand gesture is captured as a video input. Each video input is divided into a fixed number of groups of frames. A sample frame is taken randomly from each group of color and optical flow frames. The samples are fed into ConvNets for feature extraction and the features are fused and passed to an LSTM for prediction of the class of a hand gesture input.

The developed system based on the new model was trained and evaluated on the Jester dataset. To test the robustness of the new approach, we zoomed out the Jester dataset by coping with the boundary of original images. We achieved an average accuracy of 95.73%, 95.69% on the Jester dataset and the "zoomed-out" Jester dataset, respectively. This model was

also tested on Nvidia dataset and achieved a great performance with a classification accuracy of 85.13%. The results of these experiments show that the short-term sampling neural network model is effective for hand gesture recognition.

Future work of this study is as follows:

1) We will apply our approach to some other challenging video analysis tasks. In this way, we can test our approach in other application domains.

2) The optical flow approach plays a critical role in this study. However, the current optical flow algorithm is still costly. The deep learning methods could be used to calculate the optical flow features.

3) This study assumes that the input video has already been trimmed and each video contains only one of the hand gestures. This proposed model can be applied to untrimmed videos by sampling at a reasonable rate [49], [50].

REFERENCES

[1] T. Starner, J. Weaver, and A. Pentland, "Real-time American sign language recognition using desk and wearable computer based video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 12, pp. 1371–1375, Dec. 1998.

[2] H. Cooper, B. Holt, and R. Bowden, "Sign language recognition," in *Visual Analysis of Humans: Looking at People*, T. B. Moeslund, A. Hilton, V. Krüger, and L. Sigal, Eds. London, UK: Springer, 2011, pp. 539–562.

[3] J. S. Sonkusare, N. B. Chopade, R. Sor, and S. L. Tade, "A review on hand gesture recognition system," in *Proc. Int. Conf. Computing Communication Control and Automation*, Pune, India, 2015, pp. 790–794.

[4] L. Dipietro, A. M. Sabatini, and P. Dario, "A survey of glove-based systems and their applications," *IEEE Trans. Syst.*, *Man*, *Cybern.*, *Part C*, vol. 38, no. 4, pp. 461–482, Jul. 2008.

[5] B. K. Chakraborty, D. Sarma, M. K. Bhuyan, and K. F. MacDorman, "Review of constraints on vision-based gesture recognition for human-computer interaction," *in IET Comput. Vis.*, vol. 12, no. 1, pp. 3–15, Feb. 2018.

[6] C. Zhu, J. Y. Yang, Z. P. Shao, and C. P. Liu, "Vision based hand gesture recognition using 3D shape context," *IEEE/CAA J. Autom. Sinica*, DOI: 10.1109/JAS.2019.1911534.

[7] X. H. Yuan, L. B. Kong, D. C. Feng, and Z. C. Wei, "Automatic feature point detection and tracking of human actions in time-of-flight videos," *IEEE/CAA J. Autom. Sinica*, vol. 4, no. 4, pp. 677–685, Sept. 2017.

[8] B. Hu and J. C. Wang, "Deep learning based hand gesture recognition and UAV flight controls," in *Proc. 24th Int. Conf. Automation and Computing*, Newcastle upon Tyne, United Kingdom, 2018, pp. 1–6.

[9] G. Marin, F. Dominio, and P. Zanuttigh, "Hand gesture recognition with leap motion and kinect devices," in *Proc. IEEE Int. Conf. Image Processing*, Paris, France, 2014, pp. 1565–1569.

[10] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. 27th Int. Conf. Neural Information Processing Systems*, Lake Tahoe, USA, 2014, pp. 568–576.

[11] M. Asadi-Aghbolaghi, A. Clapés, M. Bellantonio, H. J. Escalante, V. Ponce-López, X. Baró, I. Guyon, S. Kasaei, and S. Escalera, "Deep learning for action and gesture recognition in image sequences: A survey," in *Gesture Recognition*, S. Escalera, I. Guyon, and V. Athitsos, Eds. Cham, Germany: Springer, 2017.

[12] Y. Zhu, Z. Z. Lan, S. Newsam, and A. Hauptmann, "Hidden two-stream convolutional networks for action recognition", in *Proc. 14th Asian Conf. Computer Vision*, Perth, Australia, 2018.

[13] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Las Vegas, USA, 2016, pp. 1933–1941.

[14] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, "ActionVLAD: Learning spatio-temporal aggregation for action classification," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Honolulu, USA, 2017, pp. 3165–3174.

[15] L. M. Wang, Y. J. Xiong, Z. Wang, Y. Qiao, D. H. Lin, X. O. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *Proc. 14th European Conf. Computer Vision*, Amsterdam, The Netherlands, 2016.

[16] L. M. Wang, Y. J. Xiong, Z. Wang, Y. Qiao, D. H. Lin, X. O. Tang, and L. Van Gool, "Temporal segment networks for action recognition in videos," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 11, pp. 2740–2755, Nov. 2019.

[17] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. 15th Annual Conf. Int. Speech Communication Association: Celebrating the Diversity of Spoken Languages*, Singapore, 2014.

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[19] M. Shilman, Z. L. Wei, S. Raghupathy, P. Simard, and D. Jones, "Discerning structure from freeform handwritten notes," in *Proc. 7th Int. Conf. Document Analysis and Recognition*, Edinburgh, UK, 2003, pp. 60–65.

[20] B. Schölkopf, J. Platt, and T. Hofmann, "Efficient learning of sparse representations with an energy-based model," in *Advances in Neural Information Processing Systems 19: Proc. the 2006 Conf.*, MIT Press, 2007.

[21] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Providence, USA, 2012, pp. 3642–3649.

[22] K. Bong, S. Choi, C. Kim, and H. J. Yoo, "Low-power convolutional neural network processor for a face-recognition system," *IEEE Micro*, vol. 37, no. 6, pp. 30–38, Nov.-Dec. 2017.

[23] K. M. He, X. Y. Zhang, S. Q. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Las Vegas, USA, 2016, pp. 770–778.

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

[25] R. Haridy. (2017, Aug. 22). Microsoft's speech recognition system is no. as good as a human. Microsoft, Redmond, Washington. [Online] Available: https://newatlas.com/microsoft-speech-recognition-equals-humans/50999/

[26] F. Beaufays. (2015, Aug.). The neural networks behind Google Voice transcription. Google, Mountain View, CA. [Online]. Available: https://ai.googleblog.com/2015/08/the-neural-networks-behind-google-voice.html

[27] H. Sak, A. Senior, K. Rao, F. Beaufays, and J. Schalkwyk. (2015, Sept.). Google voice search: Faster and more accurate. Google, Mountain View, CA. [Online]. Available: https://ai.googleblog.com/2015/09/google-voice-search-faster-and-more.html

[28] C. Smith. (2016, Jun. 13). iOS 10: Siri no. works in third-party apps, comes with extra AI features. Apple Inc., Cupertino, CA. [Online]. Available: https://bgr.com/2016/06/13/ios-10-siri-third-party-apps/

[29] W. Vogels. (2016, Nov. 30). Bringing the Magic of Amazon AI and Alexa to App. on AWS. Amazon, Seattle, Washington, [Online]. Available: https://www.allthingsdistributed.com/2016/11/amazon-ai-and-alexa-for-all-aws-apps.html

[30] AlphaStar team: Mastering the Real-Time Strategy Game StarCraft II. DeepMind, London, UK. [Online]. Available: https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

[31] C. K. Li, Y. H. Hou, P. C. Wang, and W. Q. Li, "Multiview-based 3-D action recognition using deep networks," *IEEE Trans. Human-Machine Systems*, vol. 49, no. 1, pp. 95–104, Feb. 2019.

[32] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning

spatiotemporal features with 3D convolutional networks," in *Proc. IEEE Int. Conf. Computer Vision*, Santiago, Chile, 2015, pp. 4489–4497.

[33] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Columbus, USA, 2014, pp. 1725–1732.

[34] C. J. Tsai, Y. W. Tsai, S. L. Hsu, and Y. C. Wu, "Synthetic training of deep CNN for 3D hand gesture identification," in *Proc. Int. Conf. Control, Artificial Intelligence, Robotics & Optimization*, Prague, Czech Republic, 2017, pp. 165–170.

[35] C. Y. Li, X. Zhang, and L. W. Jin, "LPSNet: A novel log path signature feature based hand gesture recognition framework," in *Proc. IEEE Int. Conf. Computer Vision Workshops*, Venice, Italy, 2017, pp. 631–639.

[36] O. Köpüklü, N. Köse, and G. Rigoll, "Motion fused frames: Data level fusion strategy for hand gesture recognition," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition Workshops*, Salt Lake City, USA, 2018, pp. 2184–21848.

[37] O. Köpüklü, A. Gunduz, N. Kose, and G. Rigoll, "Real-time hand gesture detection and classification using convolutional neural networks," in *Proc. 14th IEEE Int. Conf. Automatic Face & Gesture Recognition*, Lille, France, 2019, pp. 1–8.

[38] P. C. Wang, W. Q. Li, P. Ogunbona, J. Wan, and S. Escalera, Sergio. (2019)., "RGB-D-based human motion recognition with deep learning: A survey," *Comput. Vis. Image Understanding*, vol. 171, pp. 118–139, Jun. 2018.

[39] O. Köpüklü, N. Kose, A. Gunduz, and G. Rigoll, "Resource Efficient 3D Convolutional Neural Networks," in *Proc. IEEE/CVF Int. Conf. Computer Vision Workshop*, Seoul, Korea (South), 2019, pp. 1910–1919.

[40] W. J. Zhang and J. C. Wang, "Dynamic hand gesture recognition based on 3D convolutional neural network models," in *Proc. IEEE 16th Int. Conf. Networking, Sensing and Control*, Banff, Canada, 2019, pp. 224–229.

[41] J. Y. H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Boston, USA, 2015, pp. 4694–4702.

[42] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[43] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Las Vegas, USA, 2016, pp. 2818–2826.

[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[45] J. Materzynska, G. Berger, I. Bax, and R. Memisevic, "The jester dataset: A large-scale video dataset of human gestures," in *Proc. IEEE/CVF Int. Conf. Computer Vision Workshop*, Seoul, Korea (South), 2019, pp. 2874–2882.

[46] Twentybn, Twenty Billion Neurons Inc. Toronto, Canada. (2017) [Online]. Available: https://20bn.com/datasets/jester.

[47] H. Wang, D. Oneata, J. Verbeek, and C. Schmid, "A robust and efficient video representation for action recognition," *Int. J. Comput. Vis.*, vol. 119, no. 3, pp. 219–238, Oct.–Dec. 2016.

[48] P. Molchanov, X. D. Yang, 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. Computer Vision and Pattern Recognition*, Las Vegas, USA, 2016, pp. 4207–4215.

[49] S. C. Gao, M. C. Zhou, Y. R. Wang, J. J. Cheng, H. Yachi, and J. H. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 2, pp. 601–614, Feb. 2019.

[50] J. J. Wang and T. Kumbasar, "Parameter optimization of interval Type-2 fuzzy neural networks based on PSO and BBBC methods," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 1, pp. 247–257, Jan. 2019.

Wenjin Zhang received the B.S. degree in software engineering from Chuangshu Institute of Technology in 2018, and the M.S. degree in software engineering from Monmouth Unversity, USA, in 2020. Now, he is an Adjunct Professor at Monmouth University in the Department of Computer Science and Software Engineering. His reseach interests include machine learning, deep learning, and computer vision.

Jiacun Wang (M'00–SM'00) received the Ph.D. degree in computer engineering from Nanjing University of Science and Technology (NUST) in 1991. He is currently a Professor of software engineering at Monmouth University, USA. From 2001 to 2004, he was a Member of Scientific Staff with Nortel Networks in Richardson, Texas. Prior to joining Nortel, he was a Research Associate of the School of Computer Science, Florida International University (FIU) at Miami. Prior to joining FIU, he was an Associate Professor at NUST. His research interests include software engineering, discrete event systems, formal methods, wireless networking, and real-time distributed systems. He authored Timed Petri Nets: Theory and Application Kluwer, 1998), Real-time Embedded Systems (Wiley, 2018), and Formal Methods in Computer Science (CRC, 2019), edited Handbook of Finite Stat Based Models and Applications (CRC, 2012), and published about 90 research papers in journals and conferences. Dr. Wang was an Associate Editor of *IEEE Transactions on Systems, Man and Cybernetics, Part C*. He has served as general chair, program chair, and special sessions chair or program committee member for many international conferences.

Fangping Lan received the B.S. degree in computer engineering from Changshu Institute of Technology (CIT) in 2019. She is currently a graduate student of software engineering at Monmouth University, USA. From Sept. 2019 to May 2020, she was a Graduate Research Assistant with Monmouth University. Her research interests include machine learning and deep learning.