IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 16, 2019, accepted August 12, 2019, date of publication August 21, 2019, date of current version September 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2936604

Temporal Spiking Recurrent Neural Network for Action Recognition

WEI WANG[1], SIYUAN HAO[2], (Member, IEEE), YUNCHAO WEI[3], SHENGTAO XIAO[4], JIASHI FENG[4], AND NICU SEBE[5], (Senior Member, IEEE)

[1]Computer Vision Laboratory, École polytechnique fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland
[2]Information and Control Engineering College, Qingdao University of Technology, Qingdao 266520, China
[3]Beckman Institute, University of Illinois at Urbana–Champaign, Urbana, IL 61820, USA
[4]Department of Electrical and Computer Engineering, National University of Singapore, Singapore 259776
[5]Department of Information Engineering and Computer Science, University of Trento, 38123 Trento, Italy

Corresponding author: Siyuan Hao (lemonbananan@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61701272.

ABSTRACT In this paper, we propose a novel temporal spiking recurrent neural network (TSRNN) to perform robust action recognition in videos. The proposed TSRNN employs a novel spiking architecture which utilizes the local discriminative features from high-confidence reliable frames as spiking signals. The conventional CNN-RNNs typically used for this problem treat all the frames equally important such that they are error-prone to noisy frames. The TSRNN solves this problem by employing a temporal pooling architecture which can help RNN select sparse and reliable frames and enhances its capability in modelling long-range temporal information. Besides, a message passing bridge is added between the spiking signals and the recurrent unit. In this way, the spiking signals can guide RNN to correct its long-term memory across multiple frames from contamination caused by noisy frames with distracting factors (*e.g.*, occlusion, rapid scene transition). With these two novel components, TSRNN achieves competitive performance compared with the state-of-the-art CNN-RNN architectures on two large scale public benchmarks, UCF101 and HMDB51.

INDEX TERMS Action recognition, temporal spiking, recurrent neural network.

I. INTRODUCTION

Human action recognition in videos has drawn growing attention in computer vision, owing to its broad practical applications in many areas such as visual surveillance, behavior analysis, and virtual reality [1]–[5]. Different from image-based recognition tasks which only considers the visual apperance, for action recognition tasks, the visual appearance of each frame and the temporal motion information are equally important and should be considered simultaneously for action recognition. One common practice thus is to extract representations of videos by simultaneously making use of both video frames and optical flows [6], as widely adopted by many previous works [7]–[9].

Many researchers focus on learning better hand-crafted [10] or deep features [11] to boost the performance of classification and detection tasks [12], [13]. For action recognition, traditional methods [3], [4] rely on hand-crafted features (*e.g.* Motion Boundary Histograms, HOGHOF and SIFT [14]. Recently, some works [2], [15]–[17] propose to adopt Convolutional Neural Networks (CNNs) for video-based human action recognition, in which CNNs are taken as effective representation learners. However, the improvement brought by CNNs upon those traditional approaches is very marginal compared with the remarkable benefit CNNs have brought to other tasks like image classification. The reason is that the CNN-based methods only learn the local visual appearance of each frame and are limited in modeling the long-term cross-frame motion and other dynamics from a global view, leading to inferior performance. To address this issue, some works [8], [9], [18], [19] propose to build recurrent neural networks (RNNs) upon CNNs for capturing the long-term information. For instance, in [19], the standard gated recurrent unit (GRU) is extended to a Convolutional GRU by replacing the product operations with convolutional operations. Similar to [19], we also employ Convolutional GRU as the backbone of our network. These CNN-RNN-based approaches indeed perform better for the

The associate editor coordinating the review of this article and approving it for publication was Qichun Zhang.

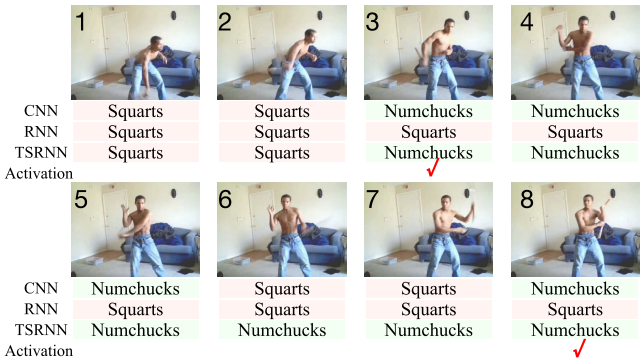


FIGURE 1. Illustration of the importance of spiking signals for correcting the contaminated memory of RNN (row 2), and activating the most reliable frames (marked with red check mark in row 4). The first two frames are wrongly predicted owing to the blurry frames as "numchucks" are invisible. Without spiking signals, all following frames (3-8) in RNN are erroneously predicted. In contrast, those frames are successfully corrected by introducing spiking signals in TSRNN (row 3). Besides, the reliable frames (*i.e.* , frame 3 & frame 8) can be found and activated.

video-based action recognition. However, these methods treat the information from all the frames equally important and this inevitably introduces noise from some "bad" frames caused by occlusion, fast moving or rapid scene transition. Such noise will contaminate the representations learned by the RNN in an accumulative way, which may bring irreparable damage to the final action recognition result. As shown in Figure 1, due to the missing of the key object (*i.e.*, numchucks), the first two frames are falsely predicted as "Squarts" by RNN, leading to wrong predictions on the following frames even when the numchucks appear. This behavior is cause by the wrong memory in RNN. In this paper, we focus on designing a robust scheme to correct the wrong memory of RNN to make better predictions.

To recognize actions more robustly even in the presence of noisy frames, we propose a novel temporal spiking recurrent neural network (TSRNN). As shown in Figure 2, the TSRNN consists of two branches, *i.e.* the key-frame branch which is used to learn the spiking signals from RGB frames and the temporal context branch which is used to learn the global semantics by taking advantage of RNN. The spiking signals provided by the key-frame branch have two functions. Firstly, the spiking signals learned from the local view (*i.e.*, single frame), which are independent of RNN, can identify the positive frame at any position of the target video. Thus, they can help RNN identify the discriminative frames, and activate the corresponding recurrent module with the help of the temporal pooling operation. Secondly, the message passing bridge between the spiking signals and the recurrent unit integrates information from both temporal local view (*i.e.*, single frame) and global view (*i.e.*, long-duration frame sequence). Therefore, the spiking signals can help identify and correct wrong long-term "memory" contaminated by those noisy frames. Besides, the recurrent connection in the temporal context branch is established on the top of convolution layer. Therefore, the inputs for the RNN are feature tensors which can better preserve the structure information than the flattened feature vectors.

Our contribution can be summarized as follows:

(i) We propose a novel temporal spiking recurrent neural network (TSRNN) where the pooling operation is implemented at the frame-level instead of the pixel-level. Therefore, the reliable discriminative frames can be identified for robustly recognizing actions even in challenging conditions.

(ii) A novel message passing bridge is introduced to the convolutional recurrent unit. This bridge allows the spiking signals to help RNN correct its contaminated memory. TSRNN not only makes a prediction from the long-term view of the target video but also accepts modulation from the spiking signals about each frame provided by the key-frame branch.

The rest of the paper is organized as follows. Section II briefly reviews the related works on action recognition.

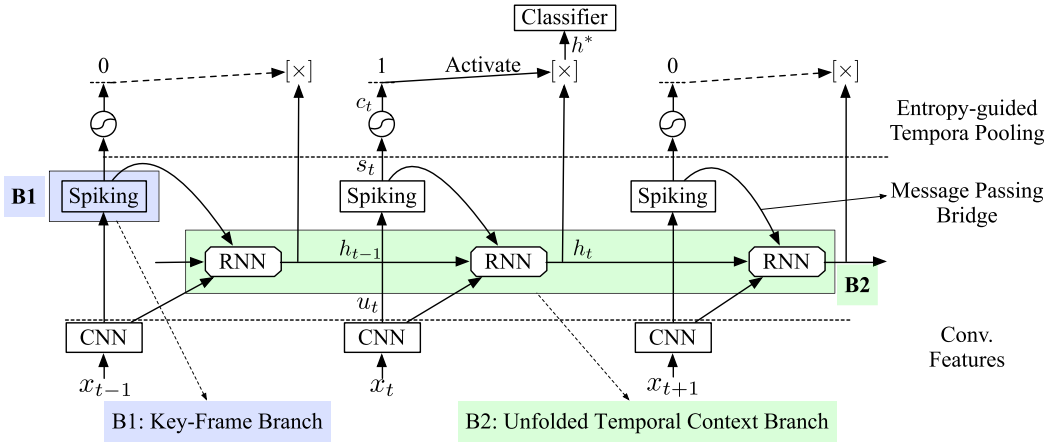


FIGURE 2. Overview of the proposed TSRNN. The TSRNN contains two branches: the key-frame branch (B1) and the temporal context branch (B2). B1 has a local view of the video and produces spiking signals. B2 is a RNN, which has a global view of the videos since RNN can memorize the previous frames. A message passing bridge is added between the spiking signals and the RNN. The memory of RNN can be corrected by the spiking signals. After mining out the most discriminative spiking signal by B1, it will activate the corresponding recurrent unit in B2.

Section III introduces the architecture of the proposed TSRNN. Section IV shows the results of our extensive evaluation on two publicly available benchmarks, UCF101 and HMDB51, and achieve comparable performance with the state-of-the-art results. Finally, in Section V conclusions are drawn.

II. RELATED WORK

In this section we review the most relevant works on action recognition using hand-crafted features, deep features, and recurrent neural networks.

A. HAND-CRAFTED AND CONVOLUTIONAL FEATURES

The previous works on feature extraction can be mainly divided into two categories: the hand-crafted features and the convolution features learned from CNNs. Traditional methods rely on the hand-crafted features for action recognition, such as dense trajectories [3], improved dense trajectories [4], and space-time interest points [20]. Dense trajectory employs a lot of hand-crafted features, such as Motion Boundary Histograms (MBH) [21], Histograms of Oriented Gradient (HOG), Histograms of Optical Flow (HOF) [22], and HOGHOF [22]. Recently, CNNs were also employed for feature extractions. Typically, previous works [8], [9], [16] follow the two-stream ConvNet architecture [2] with the spatial stream which takes as inputs the RGB video frames and the temporal stream that takes as inputs the stacked optical flow images. The optical flow images are obtained by linearly scaling the optical flows [6] to the range of [0, 255].

Simonyan and Zisserman [2] compared two optical flow stacking methods, which are (1) stacking the optical flows directly, and (2) stacking the optical flows by aligning them along the motion trajectories across frames. The optical flow stacking along the trajectory is analogous to the trajectory feature in [3] which is computed by stacking the displacement vectors along the trajectory. However, the trajectory stacking by alignment does not improve the performance. The main possible reason is that ConvNet cannot mine the motion patterns even though it is good at describing appearances. Some recent works reveal that RNN is better at modeling the temporal structure, as reviewed below.

B. RNN ARCHITECTURES

Because of RNN's good capability of modeling sequences, some recent works use RNN to model the temporal structures [23]. Yue-Hei Ng *et. al* [8] revealed that LSTM is better at encoding the temporal information than the pooling architectures over the stacked feature maps of the frames or the optical flows. Specifically, attention LSTM is employed [18], [24] where the attention mask of the next frame is predicted based on the memory of LSTM. Only static RGB frames are utlized to generate the attention mask. Therefore, the generated attention masks are unstable. For example, in some cases the attention is put to the background instead of the moving objects. In order to make the LSTM focus on the foreground, Li *et al.* [9] inferred attention from optical flows instead of video frames. They presented a two layer LSTM for action recognition. The bottom LSTM layer takes as inputs the optical flow images and outputs the *attention* masks. An element-wise product is performed between the feature maps of the frame and the feature maps of the attention masks. Then the weighted feature map is passed to the top LSTM layer. The optical flow reflects the salient motion of the foreground objects, and better performance was achieved. Actually, the optical flows can also be added beside the frames as another input and work in parallel with the frames [16]. Differently, in this paper, we employ the Gated Recurrent Unit (GRU), another widely used recurrent unit [25], as the recurrent unit instead of LSTM. A message passing bridge is added externally from the spiking signals which are independent from the recurrent unit. It's worth mentioning that the spiking signals in this paper share a similar name with [26]. However, the motivations and operations behind the spiking signals are quite different. In [26], recurrent networks of spiking neurons are employed to learn probabilistic planning such that the generated spike sequences realize mental plans. However, in out method, the spiking signals are employed to guide RNN to correct its long-term memory.

III. THE TSRNN MODEL

Following the common practice [27], we train two TSRNNs, one for the RGB frames (RGB-TSRNN) and one for the optical flow images (OF-TSRNN). In the following, we detail the TSRNN using the RGB-TSRNN. The OF-TSRNN is a replicate of RGB-TSRNN whose inputs are the stacked optical flow images instead of RGB images. In this section, we will illustrate the architecture using the RGB-TSRNN. From Figure 2 we can observe that the TSRNN includes two branches, *i.e.*, the key-frame branch (B1) and the temporal context branch (B2). B1 learns to generate spiking signals, and B2 learns global features over the whole video by taking advantage of RNN. The details of each branch are detailed as follows.

A. KEY-FRAME BRANCH

The key-frame branch is in charge of extracting spiking signals from each input frame x_t, (t=1, 2, ...). It has a **local** view of the video. Throughout the paper, we use the following notations. For an input video V, we evenly sample N frames. The input video is represented by $V=\{x_t\}_{t=1}^{N}$ where x_t represents the t-th RGB frame. Given the input frame x_t, the output spiking signal of the key-frame branch (B1) is denoted as $\mathbf{s_t}$. The key-frame branch consists of a spiking layer which is in charge of producing spiking signals, and it is built on the top of a CNN model which is used to extract convolutional feature maps. There are many CNN models available. A thorough comparison is performed in [16] revealing that the GoogleNet with batch normalization [28] has better performance than the other deep CNN models. Therefore, we employ the GoogleNet as the CNN model. We take the output of the last convolutional layer in the CNN model as the representation

of the input frames. Given the frame x_t to the CNN model, the corresponding output is denoted as $\mathbf{u_t}$. Then we feed the feature tensor $\mathbf{u_t}$ to the spiking layer to produce the spiking signals $\mathbf{s_t}$ using the following spiking function,

$$\mathbf{s_t} = f^{(s)}(\mathbf{u_t}). \tag{1}$$

where $\mathbf{s_t}$=$[s_{t,1}, s_{t,2}, ..., s_{t,C}]$. C denotes the number of action labels. The spiking layer $f^{(s)}(\cdot)$ is a composite of multiple layers including a fully connected layer and a softmax layer. The supervisions are the action labels, and the cross-entropy loss is employed to train the network.

$$\mathcal{L}_{B1} = -\sum_{t=1}^{N}\sum_{c=1}^{C} y_{t,c} \cdot log(s_{t,c}) \tag{2}$$

where $y_{t,c}$ is 1 if frame x_t belongs to action c, and $s_{t,c}$ is the output probability that frame x_t belongs to action c. Thus, the output spiking signals are the semantic representations (*i.e.* probabilities over all action candidates). Next, the spiking signals are passed through the message passing bridge to the RNN to correct its memory, and these signals are also employed to pool out the reliable frames.

1) MESSAGE PASSING BRIDGE

The message passing bridge between the spiking signals and the RNN allows the spiking signals to check the internal state of the RNN and modify it. Specifically, the semantic spiking signals for local view is independent of the long-term temporal motion, which can provide a useful spiking signal (*i.e.* semantics of key objects) to eliminate the negative effects caused by potentially noisy frames. The RNN can better learn the video semantics from a global view. In this way, both the semantic representations from the local and the global view of the video are leveraged to make a reliable prediction. From Figure 1, we can observe that the frames (3-8) are successfully corrected by introducing local semantic-level spiking signals.

2) TEMPORAL POOLING USING SPIKING SIGNAL ENTROPY

The aim of the temporal pooling is to select the most reliable and discriminative output of B2 based on the spiking signals generated by B1. As shown in Figure 2, we first mine out the index of the most discriminative frame of B1. Then the corresponding output of B2 with the same index will be activated to predict the action label since the output in B2 has a large temporal receptive filed as it contains the memory of all the previous frames. If frame x_t is good enough to predict the action label, it should have a strong confidence that this frame belongs to a certain action class and result in a **peaky** label distribution i.e. $\mathbf{s}_t$ should have **low entropy**. Given the spiking signal $\mathbf{s_t}$ from B1, we employ its entropy $\mathbf{c_t}$ as the activation score,

$$\mathbf{c_t} = -\sum_{i=1}^{C} s_{t,i} \log s_{t,i}, \tag{3}$$

Then we pool out the most discriminative frame via temporal max-pooling on the opposite number of the entropy $\mathbf{c_t}$. Let γ denote the size of the temporal pooling window. In the window $[1, \gamma]$, we have

$$k = \arg_t \max\{-\mathbf{c_t} | t \in [1, \gamma]\}, \tag{4}$$

where k is the index of the maximum score. Then the corresponding output $\mathbf{h_k}$ is activated for the subsequent action classification task.

B. TEMPORAL CONTEXT BRANCH

The temporal context branch is a convolutional RNN. It is in charge of modeling the long-term temporal content of videos. Different from the conventional RNN which flattens the input as a vector, the convolutional RNN takes as inputs the tensors directly. In this way, the detailed spatial structure can be preserved. In general, Long Short Term Memory (LSTM) [29] and Gated Recurrent Unit (GRU) [25] are two most widely used recurrent modules for RNN. Greff *et al.* [30] conducted a thorough analysis and revealed that usually GRU has competitive performance compared with LSTM. Besides, GRU has simpler structures and much fewer parameters than LSTM. Therefore, following [19], we choose to extend the standard GRU to convolutional GRU as the recurrent module.

In addition, a **message passing bridge** is built within the convolutional GRU. The message passing bridge can allow the spiking signals to interfere the memory of the recurrent module. With the help of spiking signals, the memory could be corrected. A good example is shown in Figure 1, from which we can observe that the wrong memory of RNN is corrected in TSRNN.

The memory of video context is carried by the hidden state $\mathbf{h_t}$ in GRU. The hidden state $\mathbf{h_t}, (t = 1, 2, \ldots, N)$ connects the GRU along the time axis as a chain. In the t-th recurrence, the GRU takes the memory, $\mathbf{h_{t-1}}$ (*i.e.*, hidden state), from the previous recurrence as the input to itself. The hidden states are initialized with zeros. The detailed structure is available in Section III-B1.

As shown in Figure 2, the recurrent module takes as inputs $[\mathbf{h_{t-1}}, \mathbf{s_t}, \mathbf{u_t}]$ which are the memory from the previous recurrence, the spiking signals from the key-frame branch, and the feature tensors from the CNN model respectively. The output of RNN is calculated by

$$\mathbf{h_t} = RNN_{bridge}(\mathbf{h_{t-1}}, [\mathbf{s_t}, \mathbf{u_t}]), \tag{5}$$

where $\mathbf{h_t}$ is the updated memory and it will be passed to the next recurrence. In this way, the current state is informed of what has happened before and a global view is provided. If this recurrent module is activated, its updated memory will be fed forward to the subsequent classification layers in order to make the final predictions. With the help of the message passing bridge, the input spiking vector $\mathbf{s_t}$ can be put aside together with the feature tensor $\mathbf{u_t}$ as the input to RNN. To make $\mathbf{s_t}$ compatible with the convolutional operation in RNN, we replicate the spiking vectors to tensors. The message passing bridge enables the spiking signals to help

update the memory. The spiking signals can help RNN to decide what to remember and what to forget by influencing the internal gates in the recurrent module.

1) RNN WITH MESSAGE PASSING BRIDGE

Different from the conventional RNN which flattens the input as a vector, the convolutional RNN replaces all the product with convolutional layers and takes as inputs the tensors directly such that the detailed spatial structure can be preserved in the memory.

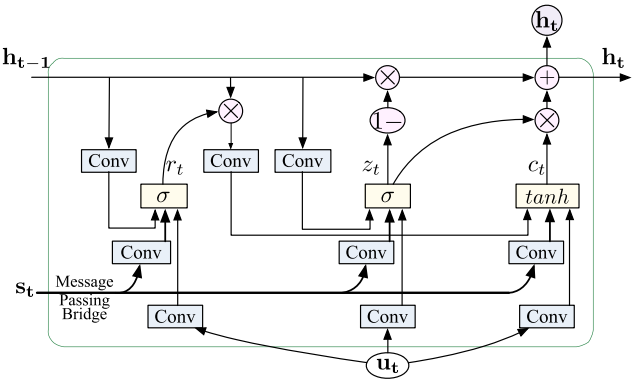


FIGURE 3. The message passing bridge allows the spiking signals $\mathbf{s_t}$ to look at and modify the gates in the Conv-GRU.

Here we employ GRU as the recurrent module. The structure of the convolutional GRU with message passing bridge is shown in Figure 3. In accordance with Figure 3, the model is shown as follows:

$$\begin{aligned} z_t &= \sigma(W_{zh} * h_{t-1} + [W_{zu}, W_{zs}] * [u_t, s_t]), \\ r_t &= \sigma(W_{rh} * h_{t-1} + [W_{ru}, W_{rs}] * [u_t, s_t]), \\ c_t &= tanh(W_{ch} * (r_t \odot h_{t-1}) + [W_{cu}, W_{cs}] * [u_t, s_t]), \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot c_t. \end{aligned} \tag{6}$$

where σ is the logistic sigmoid function and *tanh* is the hyperbolic function. W represents the filter of the convolution layer and the bias term is absorbed in W. Each GRU has two gates (*i.e.*, reset gate r_t and update gate z_t), one hidden state h_t, and one new variable c_t to be added to the hidden state.

The *reset* gate r_t decides whether the memory of previous frames should be ignored. If r_t is close to 0, the memory of the previous frames will be forced to be discarded, and the unit will focus on the current input frame only. This gate allows the unit to remember or drop the irrelevant frames.

The *update* gate z_t updates the memory via the coupled gates, z_t and $1-z_t$. This setting means that the unit only accepts a new frame when it forgets something correspondingly in the memory. $1-z_t$ controls what to remember from the memory and z_t controls what to be accepted.

c_t is the new information created by a *tanh* layer. This new information will first be weighted by the reset gate z_t and then be added into the memory stream to form a new video memory (*hidden state* h_t).

The system will have short-term memory and ignore the previous frames if the reset gate is always activated. On the other hand, if the update gate is always activated, the system will have long-term memory and all the previous frames will be memorized.

We would like to explain why we build RNN on the top of the last convolutional layer: First, the last convolutional layer of the CNN model has the largest receptive field among all the layers such that its output feature tensor contains the richest context information. Using the output from other layers will introduce redundant information and it may actually hurt the final performance. Secondly, using more CNN layers will introduce more recurrent layers and this will significantly increase computation cost. Thus, we only build RNN on the top of the last convolutional layer as it balances the performance and efficiency well.

C. ACCUMULATIVE LOSS FUNCTION

After unfolding the RNNs along the time, we can observe that the RNN in the rear of the sequence have longer-term memory compared with the RNN in the front. Thus, the rear RNNs should play more important roles since they have broader views of the video, and better predictions may be made by focusing more on these RNNs. In order to force the model to focus more on the rear RNNs, we set a bigger gain for the loss of the rear RNNs.

The classifier in Figure 2 consists of a fully connected layer and soft-max layer. We rely on the cross-entropy to calculate the classification loss. Let M represent the number of temporal pooling window. Given the activated output $\mathbf{h}_l^*$ of the l-th pooling window, its corresponding output from the soft-max layer is denoted as $S_{l,c}$. $y_{l,c}$ is a boolean variable which indicates that the video has the c-th action label. We have the following loss function,

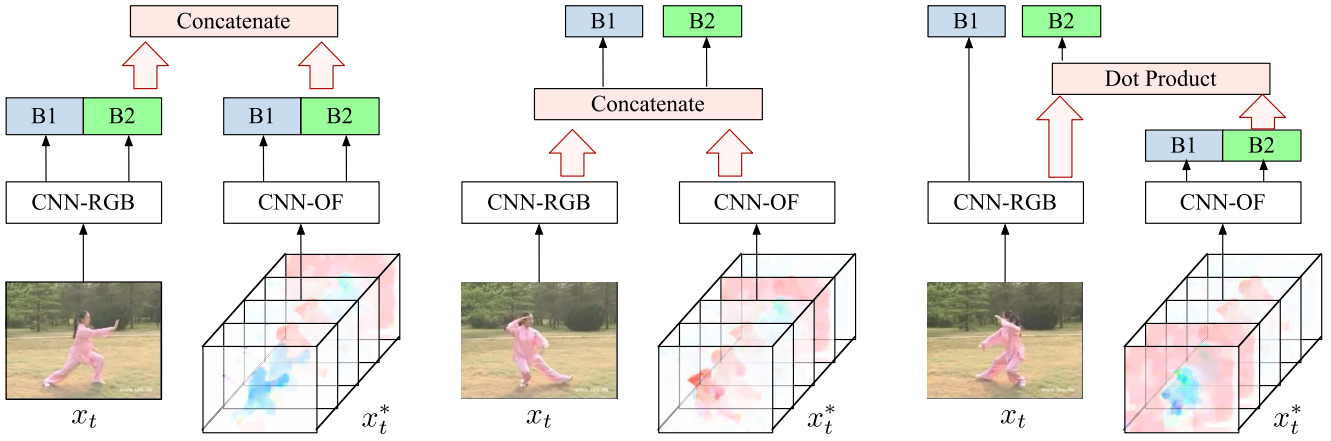
$$\mathcal{L} = \sum_{l=1}^{M} \frac{l}{M} \Big(- \sum_{i=1}^{C} y_{l,c} \log S_{l,c} \Big), \tag{7}$$

where $\frac{l}{M}$ is the gain for the loss of the l-th pooling window. Therefore, as the gain increases monotonically, the predictions at the rear RNNs will draw more attention.

In accordance with the setting in the training phase, we utilize the same weights for prediction in the testing stage. Let $\mathbf{p}_l$ represent the probability vector from the l-th triggered RNN output, whose j-th element $p_{l,j}$ denotes the probability that the video has the j-th action label. Given a series of predictions $[\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_l, \ldots, \mathbf{p}_M]$, we employ the following function to calculate the label of the video,

$$y = arg \max_j \frac{l}{M} \sum_{l=1}^{M} p_{l,j}. \tag{8}$$

As shown in Eqn. (8), the weight for the predicted probability from the l-th pooling window is also set as $\frac{l}{M}$, which is the same as the gain for the l-th pooling window in the training phase.



(a) late fusion (b) early fusion (c) attention fusion

FIGURE 4. Illustration of different fusion strategies. x_t denotes RGB frame at the time t, and x_t^* is a stack of optical flow images. We stack 5 pairs of optical flow frames starting from t. Each optical flow pair consists of a horizontal optical flow and a vertical optical flow. The CNN-RGB and CNN-OF blocks in the figure are the CNN models which process RGB and optical flow images.

D. FUSION OF RGB-TSRNN AND OF-TSRNN

The RGB-TSRNN and OF-TSRNN are complementary to each other since they view the video from different perspectives. The RGB-TSRNN may help correct the OF-TSRNN and vice versa. For example, the optical flow images of *Brushing Teeth* are visually similar to the ones of *Shaving Beard* since the hands in both actions move in a similar pattern. Therefore, the OF-TSRNN could hardly distinguish between them. However, these two actions are visually quite different. As a result, the OF-TSRNN has an accuracy of 38.9% for the action of *Brushing Teeth* while the RGB-TSRNN has an accuracy of 83.3%.

On the contrary, the OF-TSRNN may also help correct the RGB-TSRNN. For instance, the appearances of the action *Handstand Walking* and *Handstand Pushups* are very similar since the actors have the same poses. However, their movements are quite different which can be reflected by the optical flow images. The flows of the *Handstand Walking* are backwards and forwards while the flows of the *Handstand Pushups* are up and down. Hence, the OF-TSRNN has higher classification accuracy compared with that of the RGB-TSRNN (73.5% vs. 35.3%). Therefore, we expect to get better performance by fusing the two TSRNNs.

Figure 4 shows three different fusion strategies. The details of the fusion strategies are as follows,

(a) Late Fusion: First, the two TSRNNs are trained independently. Then the prediction scores from the temporal context branch from both TSRNNs are fused using the SVM for each recurrence as in [2], [8].

(b) Early Fusion: The two TSRNNs are fused by stacking the feature maps from the bottom CNN models. The stacked feature maps are passed to the two branches on top of the CNN model. The rest of the structure remains the same.

(c) Attention Fusion: The memory of the temporal context branch of the OF-TSRNN works as an attention mask. The products of the weight masks from the CNN-OF and feature tensors from the CNN-RGB are taken as input to temporal context branch of RGB-TSRNN. The output of the RGB-SRNN will be used for the final prediction.

IV. EXPERIMENTAL VALIDATION

A. DATASET DESCRIPTION

We evaluate our framework using two large public action datasets which are UCF101 [31] and HMDB51 [32]. We give a brief description of the two datasets as follows.

1) UCF101

The UCF101 dataset consists of 101 action classes, 13,320 video clips and 27 hours of video data. These user-uploaded videos are recorded in unconstrained environments containing camera motion, cluttered background, various illumination conditions, *etc*. These videos cover a wide range of action classes, such as playing musical instruments, sports, body-motion, human-object interaction, and human-human interaction. All video clips have a fixed frame rate of 25 *fps* and the resolution of 320×240. The length of each video clip is 7.21s on average from the minimum 1.06s to the maximum 71.04s. There are 3 splits for training and testing.

2) HMDB51

The HMDB51 dataset contains 6,766 video clips from 51 actions with each action containing more than 132 samples on average. In our experiments, we follow the standard evaluation scheme which divides the clips into three different training splits. In each split, each action has 70 clips for training and 30 clips for testing.

B. EXPERIMENT SETTINGS

To explore the best fusion strategies and architectures, we follow the same evaluation scheme as [2], [9] where the results of the first split of the UCF101 dataset are employed to evaluate the performance of each architecture. To compare the

performance of our method and the state-of-the-art baselines, the average accuracy is employed as the evaluation metric.

C. IMPLEMENTATION DETAILS

1) VIDEO REPRESENTATION

Following the common practice [2], [16], for each video, we sample 25 frames and 25 groups of optical flow images evenly as the inputs to the RGB-TSRNN and OF-TSRNN respectively. The optical flow images are obtained by linearly scaling the optical flows to the range of [0, 255]. The optical flows are extracted using the algorithm [33] implemented in OpenCV. Each optical flow image group consists of five consecutive pairs of optical flow images and they are stacked together as the inputs to the OF-TSRNN.

2) DATA AUGMENTATION

Data augmentation is always applied to reduce the risk of over-fitting and boost the generalization performance of the CNNs. To make a fair comparison with the state-of-the-art methods, we implement data augmentation by horizontal flipping and RGB jittering for the randomly cropped images. The sub-images with the size of 224×224 are randomly cropped from the original images.

3) TRAINING RGB-TSRNN

To explore the contribution of each branch, we first train the key-frame branch independently, and then we train the temporal context branch and fine-tune the whole network. We initialize the key frame branch together with the CNN model using the GoogleNet with Batch Normalization [28] whose parameters are pretrained using ImageNet [34]. Then we fine-tune the model using video frames with the action labels using the stochastic gradient descent algorithm with momentum. Next, we fix the learned parameters of the key frame branch together with the CNN model, and learn the parameters of the temporal context branch. The parameters are fixed by setting their learning rate to 0. Finally, we fine-tune the overall architecture. When training the key-frame branch, we follow the same parameter setting from [16]. When training the temporal context branch, the learning rate is initialized at 0.001 and it is decreased by the ratio of 0.5 every 5 epochs. We report the results in 20 epochs.

4) TRAINING OF-TSRNN

The training strategy of the OF-TSRNN is similar to the one of RGB-TSRNN. The difference is that their inputs have different number of channels and they require different CNN models, as shown in Figure 4, to extract feature tensors. The inputs to the OF-TSRNN in each time step are the stacked 5 pairs of optical flow images (each pair consists of one horizontal image and one vertical flow image). Therefore, the input to the network has 10 channels. In order to take advantage of the GoogleNet which is pre-trained using RGB images, we take the average of the weights of the first convolution layer and replicate it 10 times for the 10 channels. Then the model is fine-tuned using the stacked optical flow images.

D. RESULTS

In this section, we first analyze the contribution of each branch by implementing an ablation study. Next, we test different fusion strategies of the RGB-TSRNN and the OF-TSRNN. Finally, we report the performance comparison between our TSRNN and other state-of-the-art methods.

1) ABLATION STUDY OF THE TSRNN BRANCHES

To explore the contribution of each branch, we implement the ablation study. For the temporal context branch, we test its performance under different conditions:

(1) standard CNN: only key-frame branch is available;
(2) standard Conv. RNN: only temporal branch is available (using all frames);
(3) standard Conv. RNN + pooling: only temporal branch is available (using selected frames);
(4) both branches are available (with spiking signals);
(5) both branches are available (with spiking signals and temporal pooling);
(6) both branches are available (with average loss).
(7) both branches are available (with accumulative loss).

TABLE 1. Performance of each branch within TSRNN on the first split of the UCF101 dataset.

Branches	RGB(%)	OF(%)
Two-Stream NIPS 2014 [2]	72.7	81.0
Attention-LSTM CVIU 2017 [9]	79.6	82.1
Trajectory CNN CVPR 2015 [15]	82.8	82.2
TSN ECCV 2016 [16]	85.7	87.9
(1) standard CNN: Key-Frame Branch	84.9	83.5
(2) standard Conv. RNN: Temporal Branch (using all frames)	85.3	84.4
(3) standard Conv. RNN + pooling: Temporal Branch (using selected frames)	85.4	84.4
(4) Both Branches (spiking + avg.)	85.8	86.6
(5) Both Branches (spiking + acc.)	86.4	87.3
(6) Both Branches (spiking + pooling + avg.)	86.0	86.9
(7) Both Branches (spiking + pooling + acc.)	**86.8**	**88.1**

The size of the temporal pooling window is set to 5. Following previous literatures [2], [9], [15], we make comparisons using the first split of the UCF101 dataset. Table 1 shows the advantages of using the spiking signals via the message passing bridge and the temporal pooling architecture. In the following paragraphs, we first compare the performance of CNN and RNN. Next, we study the contribution of the spiking signals and the message passing bridge. Finally, we study the contribution of the accumulative loss.

a: KEY-FRAME BRANCH VS TEMPORAL BRANCH

From Table 1, we can observe that the standard Conv. RNN (*i.e.* the temporal branches in options 2 & 3) has superior performance than the key-frame branch (*i.e.*, option 1). This observation indicates that the standard Conv. RNN has better performance than CNN as it uses temporal information.

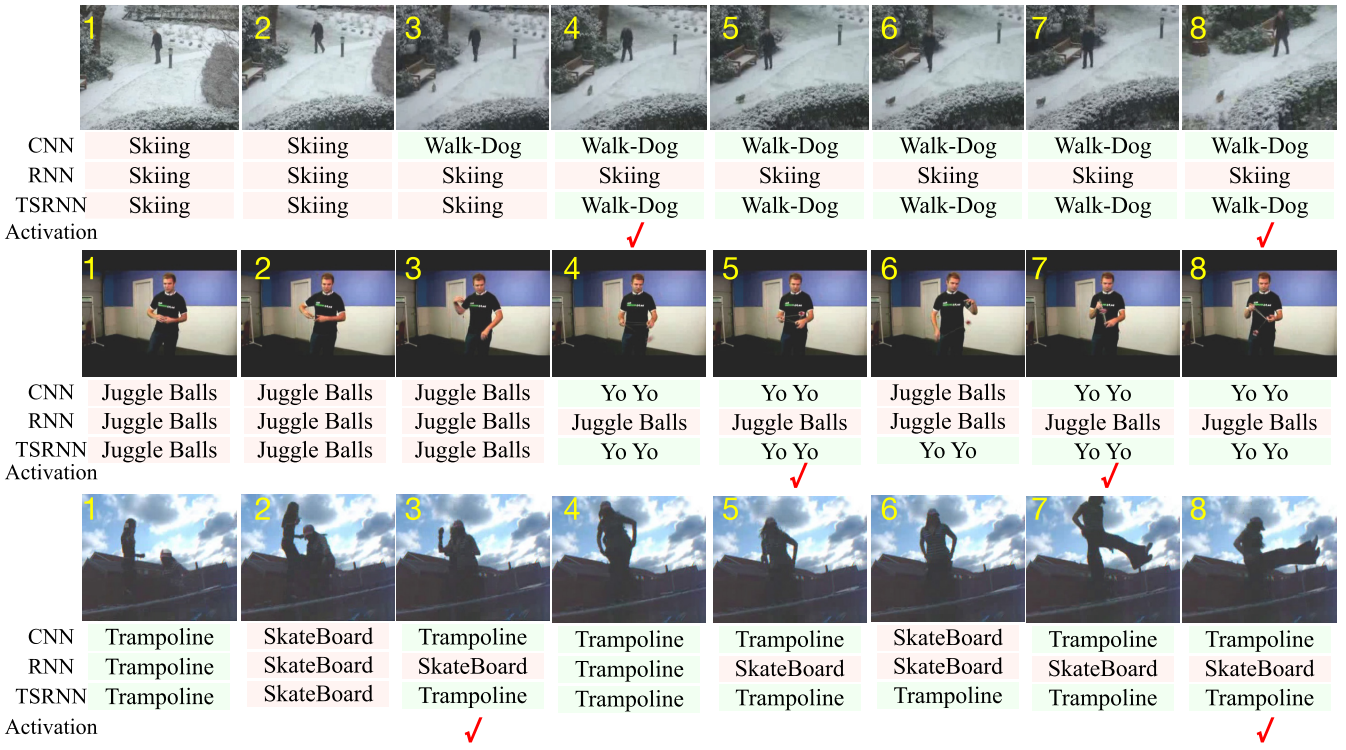


FIGURE 5. Visualization of the Predictions. The ground truth labels of the three videos are Walk-wth-Dog, YoYo and Trampoline. Below each video frame we show the prediction results with three different methods where the 1st row corresponds to the key frame branch which is a CNN model; 2nd row corresponds to the temporal context branch which is based on a CNN-RNN model; and the 3rd row corresponds to our TSRNN. The 4-th row shows the activated frames in the pooling windows based on the entropy of the spiking signals.

Besides, if we select some representative frames (option 3) instead of using all the frames (option 2), the performance of RNN could be further boosted, but the performance gain is very limited. Through the comparison between temporal branch (using all frames) and temporal branch (using selected frames), we can observe that we have 0.1% performance gain for the RGB images while the performance on the OF images remains the same.

b: THE CONTRIBUTION OF SPIKING SIGNALS

Option 4 is built on the top of Option 2 by introducing spiking signals while the rest of the network remains the same. Option 2 is the case in which only temporal branch is available (using all frames). This baseline is a standard Conv. GRU baseline which contains no spiking mechanism. The accuracy is 85.3% for RGB images and 84.4% for stacked optical flow images. Option 4 is our model which involves spiking signals. We can observe that the performance of standard Conv. GRU is inferior compared with the Conv. GRU with spiking signals which are 85.8% and 86.6%. The magnitude of the impact of the spiking signal with the message passing bridge on the Conv-GRU performance is 0.5% and 2.2%. This observation validates the spiking signal does help to boost the performance.

We have also visualized the effect of using spiking signals. Figure 5 shows the qualitative results. From Figure 5, we can observe that in the first video, without the spiking signals, the contaminated memory in RNN (second row) will continuously predicted the label as ski while the right label should be walk-with-dog. With the help of spiking signals, the memory could be corrected as shown in the TSRNN (third row). Besides, the entropy of the spiking signals could also be employed to select reliable frames. For instance, in the first video, frame 4 and frame 8 in which the dogs are clearly visible are selected. In the second video, frame 5 and frame 7 in which the strings of YoYo are clearly visible are selected. Similar results could be obtained in the third video. We can observe that by eliminating noisy frames, not only the memories could be corrected, but also the final predictions of the RNN could be corrected. We have also compared the predictions of RNN and TSRNN at all time steps and the percentage of the corrections brought by the spiking signals is 4.4% for RNN of RGB and 5.2% for RNN of OF.

c: CONTRIBUTION OF THE ACCUMULATIVE LOSS & TEMPORAL POOLING

In addition to the weighted accumulative average of the predictions from each pooling window, we also take the average without weights. However, this will lead to inferior performance as shown in Table 1 (options 4 & 5). This observation shows that the recurrent unit at the end of the sequence can make more accurate prediction. Actually, the recurrent unit at the end in more informative as it has a memory of all the previous frames. Therefore, the weighted temporal branch performs better than the others as it puts larger weight on the predictions of the recurrent unit at the end. We also tested

a special case where only the weight of the last recurrent step is set to 1 while the weights of all the other steps are set to 0. In this case, the accuracy is 85.1% and 83.9% for RGB and OF which is inferior than the case in which the weighted predictions from all the steps are employed. Apart from the accumulative loss, we can also observe that the temporal pooling operation could further boost the performance by selecting good signals while ignoring the bad ones. By comparing options 4 & 6, 5 & 7, we can observe clearly that the pooling operation can benefit the final prediction.

d: RGB-TSRNN VS OF-TSRNN

Table 1 shows the performance of each branch of both the RGB-TSRNN and OF-TSRNN. For the RGB-TSRNN, we can observe that the temporal context branch has better performance compared with the key-frame branch. This observation verifies that the sequence information does matter and RNN can further boost the performance. For the OF-SRNN, similar results can be observed. The improvement of the OF-TSRNN is more significant compared with that of the RGB-TSRNN. This demonstrates that the aggregation of the motion images along the time axis is more effective than the aggregation of the static RGB images.

2) TSRNN FUSION

We explore different fusion strategies, as shown in Figure 4, for the two TSRNNs. We make the predictions by taking the weighted average of the predicted probabilities from each pooling window in order to keep in accordance with the loss in the training phase. To implement attention fusion for the tempral context branch, we use the same setting of [9] where the optical flow images are employed to generate the masks for the RGB frames. For the SVM fusion, we follow the same setting of [2].

TABLE 2. Accuracy(%) of different fusion strategies on the UCF101 dataset (split 1) with/without temporal pooling operations.

Fusion	Early	Late(SVM)	Late(DotProduct)	Attention
Bridge	87.8	93.2	93.5	88.9
Bridge+Pooling	87.9	93.5	93.8	89.1

Table 4 shows the performance of each fusion strategy. We can observe that the late fusions always give better performance compared with other fusion strategies, and the late fusion with simple dot-product gives the best performance. This means the high-level semantic representations from the top layers of the CNN model are more representative than the low-level features from the bottom layers of the CNN model. Similarly, the temporal pooling operation could further boost the performance.

3) COMPARISON WITH THE STATE-OF-THE-ART

After exploring the fusion strategies and the contribution of each branch, we test our model on the UCF101 and HMDB51 datasets by comparing it with other baselines.

TABLE 3. Comparison between our TSRNN with the state-of-the-art methods.

Methods	HMDB51(%)	UCF101(%)
Traditional methods:		
DT (MVSV) CVPR 2014 [35]	55.9	83.5
iDTs (FV) ICCV 2013 [4]	57.2	85.9
iDTs (HSV) CVIU 2016 [36]	61.1	87.9
CNN-based methods:		
Attention-LSTM ECCV 2016 [18]	41.3	77.0
Two-Stream NIPS 2014 [2]	59.4	88.0
LSTM CVPR 2015 [8]	–	88.3
VideoLSTM CVIU 2017 [9]	56.4	89.2
TDD (FV) CVPR 2015 [15]	63.2	90.3
LTC (LSTM) CVPR 2015 [37]	64.8	91.7
Action VLAD CVPR 2017 [38]	66.9	92.7
Spatio-Temp. Network CVPR 2017 [39]	68.9	**94.6**
TSN ECCV 2016 [16]	68.5	94.0
TSN ECCV 2016 [16]+ Warped Flow	69.4	94.2
Spatio-Temp. Multiplier CVPR 2017 [40]	68.9	94.2
Spatio-Temp. Vector CVPR 2017 [41]	**69.5**	93.6
TSRNN + bridge	69.7	94.2
TSRNN + bridge + pooling	**69.9**	**94.4**

The 3-fold mean accuracy of each dataset is reported as the evaluation metric. The results are summarized in Table 3.

We first compare our model with traditional approaches, such as dense trajectories (DT) and the improved dense trajectories (iDT) [35], [36]). We also compare it with the CNN approaches [2], [15], [16] and RNN approaches [8], [9], [37].

From Table 3, we can observe that deep learning methods have better performance compared with the traditional methods since the deep learning methods are better at extracting semantic features from images. Moreover, by encoding the temporal information, even better performance could be achieved [16]. Since RNNs perform better at encoding the long-term temporal sequences, the RNN based methods are also employed [8], [9], [37]. We can also observe that our TSRNN has competitive performance as Spatio-Temporal Vector [41] on the HMDB51 dataset as the spiking mechanism can make the RNN more robust to the noisy frames using the spiking signals to select reliable frames and correct the contaminated memories. However, the UCF101 dataset has already been saturated. Therefore, it is very difficult to improve the performance.

So far, the best performance on the UCF101 dataset (without using extra training data) is 94.9% from [40]. However, one should note that their big performance gain comes from multi-model fusion which boosts their performance from 94.2% to 94.9%. In particular, [40] augments deep features with hand-crafted features (the improved dense trajectory-iDT), which requires another complex framework to extract points of interests and implement clustering. In such a case, the fusion result (*i.e.* 94.9%) comes from an inelegant solution and the computation cost is also very high. In the scenario where only deep models are employed to process the RGB images and optical flows, our accuracy is 94.4% which is comparable to 94.2% reported in [40]. Similarly, the best performance on HMDB51 dataset is 73.1% [41]

TABLE 4. Accuracy(%) on the UCF101 & HMDB51 datasets by combining hand-crafted features.

Methods	HMDB51(%)	UCF101(%)
[40]+iDT	72.2	94.9
[41]+iDT	73.1	94.6
Ours + iDT	72.8	94.7

in which they employ two types of hand-crafted features which are iDT and Histograms of Motion Gradients which boost their performance from 69.5% to 73.1%. Based on a fair comparison, our performance 69.9% is better compared with [41] whose performance is 69.5%.

To have a more fair comparison with [40], [41], we have also implemented extra experiments by augmenting iDT features. Similar to [40], [41], we fuse confidence scores from iDT and deep features to obtain the final prediction, and the corresponding results are shown in the table above. The accuracies on HMDB51 and UCF101 are further boosted to 72.8% and 94.7% respectively. Reference [40] has the best performance on UCF101, and worst performance on HMDB51 while [41] has the best performance on HMDB51, and worst performance on UCF101. Compared with [40], [41], we can observe that although our method does not achieve the best performance, our method seems more stable as it always achieves an intermediate rank. Besides, although the accuracy improvement is not very significant compared with other works, the ablation study shows that the performance gain brought by the spiking mechanism is considerably significant compared with the standard Conv. RNN. Except for combining hand-crafted features and deep features, another way to boost the performance is data augmentation. For instance, I3D [42] employs a huge extra dataset to boost the action recognition performance such that it is not directly comparable with our method.

V. CONCLUSION

In this paper, we presented a novel Temporal Spiking-RNN for action recognition which achieves competitive performance compared with the state-of-the-art methods using less features. The TSRNN consists of two branches (*i.e.*, the key frame branch and temporal context branch). The key frame branch has a local view of the video and it is in charge of generating spiking signals of each individual input. The temporal context branch is an RNN which has a global view of the video. The recurrent module is a convolutional Gated Recurrent Unit with a message passing bridge. The temporal pooling architecture over the two branches can select the most reliable and descriminative frames based on the entropy of the spiking signals and activate the corresponding recurrent unit in the temporal context branch. The spiking signals could also help correct the contaminated long-term memory of the temporal context branch via the message passing bridge. Moreover, the inputs to the convolutional GRU are feature tensors instead of flattened vectors such that the spatial structure can be better preserved in the memory of the convolutional GRU. The ablation study shows the advantage of using spiking signals and the message passing bridge over the standard RNN.

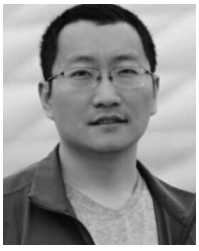
REFERENCES

[1] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.
[2] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 568–576.
[3] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Action recognition by dense trajectories," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 3169–3176.
[4] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 3551–3558.
[5] G. Zhu, C. Xu, Q. Huang, W. Gao, and L. Xing, "Player action recognition in broadcast tennis video with applications to semantic analysis of sports game," in *Proc. 14th ACM Int. Conf. Multimedia*, 2006, pp. 431–440.
[6] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Proc. Scand. Conf. Image Anal.* Berlin, Germany: Springer, 2003, pp. 363–370.
[7] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," 2016, *arXiv:1604.06573*. [Online]. Available: https://arxiv.org/abs/1604.06573
[8] 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 Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 4694–4702.
[9] Z. Li, K. Gavrilyuk, E. Gavves, M. Jain, and C. G. M. Snoek, "VideoLSTM convolves, attends and flows for action recognition," *Comput. Vis. Image Understand.*, vol. 166, pp. 41–50, Jan. 2018.
[10] L. Liu, P. Fieguth, Y. Guo, X. Wang, and M. Pietikäinen, "Local binary features for texture classification: Taxonomy and experimental study," *Pattern Recognit.*, vol. 62, pp. 135–160, Feb. 2017.
[11] L. Liu, J. Chen, G. Zhao, P. Fieguth, X. Chen, and M. Pietikäinen, "Texture classification in extreme scale variations using GANet," 2018, *arXiv:1802.04441*. [Online]. Available: https://arxiv.org/abs/1802.04441
[12] L. Liu, J. Chen, P. Fieguth, G. Zhao, R. Chellappa, and M. Pietikäinen, "From BoW to CNN: Two decades of texture representation for texture classification," *Int. J. Comput. Vis.*, vol. 127, no. 1, pp. 74–109, 2019.
[13] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," 2018, *arXiv:1809.02165*. [Online]. Available: https://arxiv.org/abs/1809.02165
[14] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *Proc. 15th ACM Int. Conf. Multimedia*, 2007, pp. 357–360.
[15] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 4305–4314.
[16] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 20–36.
[17] P. Wang, Z. Li, Y. Hou, and W. Li, "Action recognition based on joint trajectory maps using convolutional neural networks," in *Proc. 24th ACM Int. Conf. Multimedia*, 2016, pp. 102–106.
[18] S. Sharma, R. Kiros, and R. Salakhutdinov, "Action recognition using visual attention," in *Proc. Int. Conf. Learn. Represent. Workshop*, 2016, pp. 1–11.
[19] B. Huang, H. Huang, and H. Lu, "Convolutional gated recurrent units fusion for video action recognition," in *Proc. Int. Conf. Neural Inf. Process.*, 2017, pp. 114–123.
[20] I. Laptev, "On space-time interest points," *Int. J. Comput. Vis.*, vol. 64, nos. 2–3, pp. 107–123, 2005.
[21] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 428–441.
[22] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, pp. 886–893.

[23] W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, and X. Xie, "Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks," in *Proc. AAAI*, 2016, pp. 3697–3703.
[24] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data," in *Proc. AAAI*, 2017, pp. 4263–4270.
[25] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1–15.
[26] E. Rueckert, D. Kappel, D. Tanneberg, D. Pecevski, and J. Peters, "Recurrent spiking networks solve planning tasks," *Sci. Rep.*, vol. 6, Feb. 2016, Art. no. 21142.
[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Sep. 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556
[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1–11.
[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[30] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," 2015, *arXiv:1503.04069*. [Online]. Available: https://arxiv.org/abs/1503.04069
[31] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," 2012, *arXiv:1212.0402*. [Online]. Available: https://arxiv.org/abs/1212.0402
[32] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2556–2563. doi: 10.1109/ICCV.2011.6126543.
[33] C. Zach, T. Pock, and H. Bischof, "A duality based approach for real-time TV-L^1 optical flow," in *Proc. Joint Pattern Recognit. Symp.* Berlin, Germany: Springer, 2007, pp. 214–223.
[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[35] Z. Cai, L. Wang, X. Peng, and Y. Qiao, "Multi-view super vector for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 596–603.
[36] X. Peng, L. Wang, X. Wang, and Y. Qiao, "Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice," *Comput. Vis. Image Understand.*, vol. 150, pp. 109–125, Sep. 2016.
[37] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 2625–2634.
[38] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, "ActionVLAD: Learning spatio-temporal aggregation for action classification," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2017, pp. 971–980.
[39] Y. Wang, M. Long, J. Wang, and P. S. Yu, "Spatiotemporal pyramid network for video action recognition," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 2097–2106.
[40] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Spatiotemporal multiplier networks for video action recognition," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 7445–7454.
[41] I. C. Duta, B. Ionescu, K. Aizawa, and N. Sebe, "Spatio-temporal vector of locally max pooled features for action recognition in videos," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 3205–3214.
[42] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," in *Proc. CVPR*, Jul. 2017, pp. 4724–4733.

WEI WANG received the Ph.D. degree from the University of Trento, Italy, in 2018. He currently holds a Postdoctoral position with the Ecole Polytechnique Fédérale de Lausanne (EPFL). His research interests include computer vision, deep learning, and augmented reality, particularly human centered perception, including face and hand analysis.

SIYUAN HAO (M'17) received the Ph.D. degree from the College of Information and Communications Engineering, Harbin Engineering University, Harbin, China, in 2015. She is currently a Researcher with the Qingdao University of Technology, Qingdao, China, where she teaches remote sensing and electrical communication. Her research interests include hyperspectral imagery processing and machine learning.



YUNCHAO WEI received the Ph.D. degree from Beijing Jiaotong University, Beijing, China, in 2016, advised by Prof. Y. Zhao. He is currently a Postdoctoral Researcher with the University of Illinois at Urbana–Champaign. He has published more than 30 papers in top-tier conferences/journals, with over 1000 citations in Google Scholar. His current research interest includes computer vision techniques for large-scale data analysis; specifically, he has been involved in weakly supervised and semi-supervised object recognition, multi-label image classification, image/video object detection, and multi-modal analysis. He received the Excellent Doctoral Dissertation Awards of the Chinese Institute of Electronics (CIE), in 2016, the Winner Prize of the Object Detection Task (1a) in ILSVRC 2014, and the Runner-Up Prizes of all the video object detection tasks in ILSVRC 2017.

SHENGTAO XIAO received the B.Eng. degree from the National University of Singapore, Singapore, in 2013, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering.

His research interests include computer vision, deep learning, facial landmark detection, 3D face reconstruction, and human–computer interaction.

JIASHI FENG received the Ph.D. degree from the National University of Singapore (NUS), in 2014. He was a Postdoctoral Research Fellow with the University of California at Berkeley, Berkeley. He joined NUS as a Faculty Member, where he is currently an Assistant Professor with the Department of Electrical and Computer Engineering. His research areas include computer vision, machine learning, object recognition, detection, segmentation, robust learning, and deep learning.

NICU SEBE (SM'96) received the Ph.D. degree from Leiden University, The Netherlands, in 2001. He is currently with the Department of Information Engineering and Computer Science, University of Trento, Italy, where he is leading the research in the areas of multimedia information retrieval and human behavior understanding. He is a Senior Member of the ACM and a Fellow of IAPR. He was the General Co-Chair of ACM Multimedia 2013 and the Program Chair of ACM Multimedia 2011, ECCV 2016, and ICCV 2017.