Pattern Recognition Letters 158 (2022) 55–62

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Recognizing micro actions in videos by learning multi-layer local features

Yang Mi[a,*], Zhihao Liu[b], Kai Zhao[c], Song Wang[d]

[a] Department of Data Science and Engineering, College of Information and Electrical Engineering, China Agriculture University, Beijing 100083, China
[b] Beijing Key Lab of Traffic Data Analysis and Mining, Beijing Jiaotong University, Beijing 100044, China
[c] School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, Jiangsu, 221116, China
[d] Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29201, USA

ARTICLE INFO

Article history:
Received 28 April 2021
Revised 17 January 2022
Accepted 2 April 2022
Available online 9 April 2022

Edited by Jiwen Lu

Keywords:
Action recognition
Micro action
Lower-level layers
Local features

ABSTRACT

Recognizing micro actions such as slight head shaking or hand clapping from videos can be challenging since they only involve small movements of local body parts. In this paper, we propose to fuse features from both higher-level and lower-level layers of convolutional neural networks for improving the accuracy of micro-action recognition. Deep features in higher-level layers have been shown to be effective in recognizing general actions, such as biking and jumping, that involve relatively large movements. Different from features in higher-level layers, features in lower-level layers are usually of higher resolution and can help capture small motions in micro actions. In this paper, we employ class-discriminative information as a guidance in lower-level layers to learn local features that are highly associated with micro-action regions. In the experiments, we evaluate the proposed method on two micro-action video datasets and achieve new state-of-the-art performance. We also test the proposed method on two general-action video datasets with promising performance.

1. Introduction

Video-based action recognition has been widely studied in the pattern-recognition community [1–3]. Most of the existing works focused on recognizing general actions that involve relatively large movements, such as lifting and jumping, while few works [4–6] concentrated on recognizing micro actions that only involve small or even subtle movements within local regions such as slight head shaking and shoulder shrugging. Sample micro-action and general-action video frames are shown in Fig. 1. However, accurate micro-action recognition is very important to video surveillance and processing since these actions reflect the behavior and intention of the subjects, e.g., a slight head shaking may indicate a disagreement while a slight head nodding may indicate an agreement.

Clearly, with small movements, micro-action recognition can be a very challenging problem, especially when the micro actions of interest are mixed with other unconcerned general actions. Current state-of-the-art approaches for general-action recognition are based on deep learning, where convolutional neural networks (CNNs) are usually used to learn deep spatial-temporal features in higher-level layers for classification. With low resolution and semantic meanings, such deep features have been shown to be effective for general-action recognition [7–9]. However, they may be insufficient to capture small movements for accurate micro-action recognition [5].

Compared with higher-level layer CNN features, lower-level layer CNN features can also be general and discriminative [10], and usually provide higher spatial resolution information [11], which help capture small motion for micro actions. In object detection, the lower-level layer CNN features can improve semantic segmentation quality since they capture more fine details of the input objects [12]. Inspired by these findings, in this paper we propose to enhance micro-action recognition by learning the local features from multiple lower-level layers. The discriminative and fine information contained in these lower-level-layer local features can benefit the learning of small motion within local regions. Using the temporal shift module (TSM) video network [7] on ResNet-50 [13] as the baseline, we first calculate the class activation map (CAM) [14] and then employ CAM as the class discriminative information to locate the regions associated with the small motion of interest. After that, we extract the local feature maps from the lower-level convolutional layers and further learn the

* Corresponding author.
E-mail addresses: miy@cau.edu.cn (Y. Mi), 16120394@bjtu.edu.cn (Z. Liu), kaizhao@cumt.edu.cn (K. Zhao), songwang@cec.sc.edu (S. Wang).

https://doi.org/10.1016/j.patrec.2022.04.002
0167-8655/

Fig. 1. Two sample micro-action videos: (a) shake head and (b) shrug shoulders, and two sample general-action videos: (c) lift and (d) jump. Green boxes show the actions of interest.

extracted local features via residual convolutional block [13], followed by the dropout and fully-connected layers to compute the class scores. The scores from multiple lower-level layer features are fused with those from higher-level layers in the original TSM for the final classification. We name the above learning progress for lower-level layers as Multi-layer Local Feature Extractor. In the experiments, we evaluate the proposed method on two micro-action video datasets [5,6], resulting in new state-of-the-art performance. We also evaluate the proposed method on two general-action video datasets [15,16] to show its wide usefulness.

The main contributions of this paper are: 1) We propose to enhance micro-action recognition by learning the local features from multiple lower-level layers. 2) We combine class discriminative information and residual convolutional block to extract and further learn lower-level layer local features. 3) The proposed method achieves state-of-the-art performance on two micro-action datasets as well as promising results on two general-action datasets.

The remainder of this paper is organized as follows. Section 2 briefly discusses the related work. Section 3 elaborates on the proposed method in detail. Section 4 reports the experimental results, followed by a brief conclusion in Section 5.

2. Related work

Video-based action recognition is one of the prime tasks in computer vision. It has drawn much attention in the multimedia community for decades. In this section, we briefly review related work on general-action recognition and micro-action recognition.

2.1. General-action recognition

For video-based general-action recognition, many methods have been developed to obtain effective spatial-temporal features from videos. These methods can be classified into two categories based on the way of extracting features: hand-crafted feature based and deep-learning-feature based approaches.

Conventional action-recognition approaches usually extract hand-crafted features from videos, including histograms of oriented gradients (HOG) [17], spatio-temporal interest points (STIP) [18], and improved dense trajectories (iDT) [19]. Among them, iDT is one of the state-of-the-art hand-crafted representations, due to its strong performance on video action benchmarks [15,16].

Recently, many action-recognition methods have concentrated on learning spatial-temporal features from videos via deep neural networks. Two-stream CNNs [20] was developed to learn the appearance and motion features from RGB frames and the optical-flow sequence, respectively for action recognition. Later, Jeffrey et al. [21] used the recurrent neural networks (RNNs) with long short-term memory (LSTM) unit on the top of CNNs to learn temporal dynamics across video frames. Wang et al. [9] developed a temporal segment network by averaging video-clip representations for video-level prediction. Tran et al. [8] proposed $R(2+1)D$ Net by decomposing 3D convolutions into 2D spatial convolutions and 1D temporal convolutions to reduce the number of learnable parameters. Martinez et al. [22] treated action recognition as fine-grained video classification and learned discriminative filter banks from the second last convolutional layer, by using a 1×1 convolutional layer and a max-pooling layer. However, almost all the above works aimed at general-action recognition without considering motion subtleness of micro actions. Different from them, this paper further learns local features from multiple lower-level layers and combine them with higher-level layer deep features for classifying micro actions.

2.2. Micro-action recognition

As far as we know, only a few previous works focused on micro-action recognition. Yonetani et al. [6] extracted hand-crafted features from both first-person (actor's view) and third-person (observer's view) videos for recognizing micro actions. One of its limitations is its requirement of both first-person and third-person videos and manual synchronization between them. In addition, existing general-action recognition methods are used for feature extraction. In this paper, we study micro-action recognition by only using the third-person videos and propose to learn multi-layer local features to address the motion subtleness in micro-action recognition. Mi et al. [4] developed a segment-level temporal pyramid for micro-action videos by combining multi-scale temporal information upon the features extracted from the last convolutional layers. However, it does not consider the local features from lower-level CNN layers, which is the main contribution of the method proposed in this paper. Mi et al. [5] proposed a dual-branch network to utilize both high-layer and mid-layer CNN features for micro-action recognition. The mid-layer CNN features were further explored via a combination of $1 \times 1 \times 1$ convolutional layer, non-local operation and pooling to learn motion representation. However, the features from much lower-level CNN layers are still not studied. Different from their work, we thoughtfully consider the features from much lower CNN layers by extracting the local features based on class discriminative information. In Sections 4, we will show that our method significantly outperforms these three existing micro-action recognition approaches on two micro-action datasets [5,6].

2.3. Learning features from multiple layers

Learning features from multiple CNN layers have been studied in the community. Li et al. [23] proposed the deeply-supervised CNN model (DS-CNN) with trainable feature aggregation. DS-CNN first aggregated features from the same level of CNN layer for multiple frames. Then, aggregated features from different levels of CNN layers were used to compute classification scores. Finally, these scores were combined via weighted sum to predict video-level action classes. Different from this work, we use CAM as a guidance to extract local features from different lower-level CNN layers for capturing fine details of micro actions. In Section 4.3, we also show that our method significantly outperforms this work on two general-action datasets UCF101 and HMDB51 [15,16]. Tang et al. [24] proposed the GoogLeNet based multi-stage feature fusion to recognize scenes in images. However, the features are directly pooled from the corresponding CNN layers in these works. In this paper, we further explore the local features in different lower-level CNN layers for micro-action recognition. In [12,25], multiple

levels of CNN features are used to predict the positions for objects with multiple scales in input images. In this paper, we enhance micro-action recognition by learning local features from multiple lower-level CNN layers.

3. Proposed method

In this section, we describe the proposed method for micro-action recognition in detail. Our model utilizes the peak response of class activation maps [14] as a guidance along with residual blocks, to extract and further learn the micro-motion related local features for multiple lower-level convolutional layers. We first introduce the base architecture, then describe the proposed network, and finally elaborate on the loss function of our model.

3.1. Base architecture

Our model is based on a general-action recognition framework by simply inserting temporal shift module (TSM) [7] into 2D CNNs, which requires the same computation and parameters as conventional 2D CNNs and can achieve the performance of 3D CNNs. Same as the temporal segment network (TSN) [9], TSM video network extracts averaged features from strided sampled frames and outputs the recognition result with a single forward pass. In this paper, we use TSM video network with backbone ResNet-50 [13] as the baseline architecture to build the proposed network. For simplicity, we refer TSM video network as TSM in the remaining paper.

3.2. Proposed network

The overall architecture of the proposed network is shown in Fig. 2. It starts with the baseline model TSM, in which the input is the video-frame sequence and the final outputs are scores for all the action classes. The parts within the red dotted box at the end of TSM are used to calculate the class activation maps (CAM) by combining the weights of the fully-connected layer and the feature maps of the last convolutional layer *Conv5_x* following the way in Zhou et al. [14]. Although CAM is computed by using the high-level layer CNN features, the peak response of CAM could indicate the specific region of the image where the network focuses on to recognize the corresponding class [26,27]. Inspired by this, we use (CAM) as a guidance to locate the regions associated with the small motion of interest in this paper. Later in the visualization experiment, we will show that the proposed method can effectively locates the micro actions of interest with the use of CAM. For an input video frame, $f_k(x, y)$ denotes the response of kth output feature map of *Conv5_x* at spatial location (x, y); w_k^c denotes the weight that connects the kth input unit to the cth output unit in the fully-connected layer, which indicates the importance of kth feature map of *Conv5_x* for class c. Then, the class activation map M_c for class c is calculated by:

$$M_c(x, y) = \sum_{k=1}^{K_5} w_k^c f_k(x, y), \tag{1}$$

where K_5 is the number of output feature maps (i.e., the number of channels) of *Conv5_x*.

Taking CAM as input, we propose the Multi-layer Local Feature Extractor (MLFE, as shown in the lower part of Fig. 2) to locate micro-action related regions in lower-level layers, and the resulting regions are used to learn local features for better recognizing micro actions. As discussed in Garcia-Gasulla et al. [10], Zhang et al. [11], lower-level layer features usually have higher resolutions than higher-level layer features, and can also be general and discriminative. We expect that the local features learned from MLFE can be class-discriminative for micro actions by involving higher resolution information. As shown in Fig. 2, we finally combine the higher-level layer features from TSM and lower-level layer features from MLFE for action recognition.

In MLFE, given CAM and the feature maps of lower-level layers as input, the Local Feature Extractor (LFE) learns local features that relate to small motion for each lower-level layer. Then, the outputs from all LFEs are fused by an aggregation function which returns an evenly averaging of class scores. Let O_2, O_3, O_4 represent the outputs of LFEs which are applied on lower-level layers *Conv2_x*, *Conv3_x*, and *Conv4_x*, respectively. We compute the output of MLFE for nth frame of the input sequence as:

$$O_{MLFE}^n = \mathcal{A}(O_2, O_3, O_4), \tag{2}$$

where $\mathcal{A}$ is a simple averaging operation. Then, the final output of MLFE is obtained by applying a consensus function $\mathcal{G}$ in [9] to obtain the video-level output of MLFE by

$$O_{MLFE} = \mathcal{G}(O_{MLFE}^1, O_{MLFE}^2, \ldots, O_{MLFE}^N), \tag{3}$$

where N is the total number of input frames. We follow [9] to select $\mathcal{G}$ as temporal average pooling.

In the end, for an input video-frame sequence, we sum up O_{MLFE} and the TSM output O_{TSM} by a coefficient λ to get the final output O, i.e., the predicted class scores in video-level as

$$O = O_{TSM} + \lambda \cdot O_{MLFE}. \tag{4}$$

3.3. Local feature extractor

Local Feature Extractor (LFE) is applied to certain lower-level layers to learn local features that are highly related to the micro actions of interest, by using the peak response from CAM as a guidance to locate the small motion. Although only one single peak location is selected by LFE, the area besides the peak location are

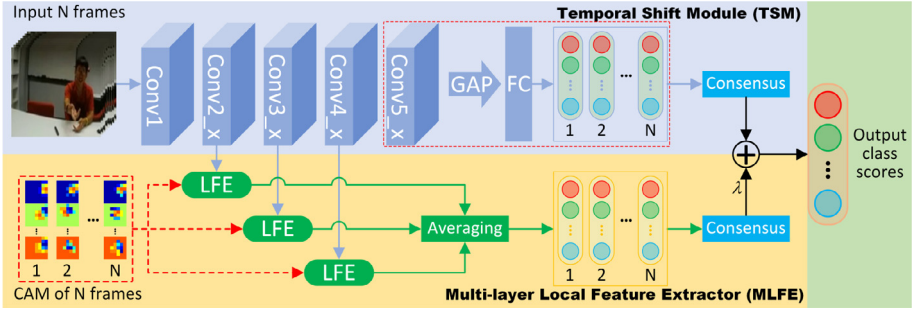

Fig. 2. The overall architecture of our model. TSM with backbone ResNet-50 is on the top. The output sizes of *Conv1*, *Conv2_x*, *Conv3_x*, *Conv4_x*, and *Conv5_x* are 112×112, 56×56, 28×28, 14×14, and 7×7, respectively. The proposed MLFE is on the bottom. The input of the model is the video frame sequence. The output of the model is the weighted sum of the output of TSM and MLFE. GAP and FC denote global average pooling and fully-connected layer, respectively.

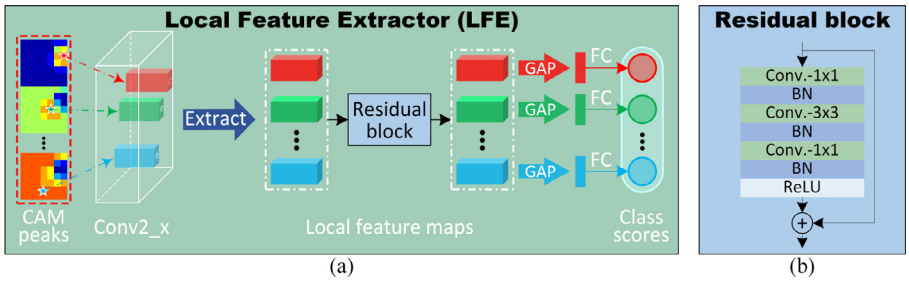

Fig. 3. An illustration of the network architecture of Local Feature Extracter (LFE) and its residual block. In LFE, the response peak in CAM is marked by a star. The corresponding regions in *Conv2_x* mapped through the response peaks are shown in colored cubes. These cubes are used to further learn local features. Each cube corresponds to a local region that is highly associated with an action class. In the residual block, BN and ReLU represent the batch normalization [28] and the non-linearity activation [29], respectively. GAP and FC denote global average pooling and fully-connected layer, respectively.

still covered by the features from TSM, since we also use the high-level layer features from TSM which are computed upon the whole image frame to compensate with the low-level layer features for the final classification. Taking the lower-level layer *Conv2_x* as an example, the detailed architecture of LFE is shown in Fig. 3. Let $\mathcal{F}_2(x, y)$ represent the output feature maps of *Conv2_x*. (P_x^c, P_y^c) represents the location of the response peak of M_c for class c, and it is computed by

$$(P_x^c, P_y^c) = \text{argmax}(M_c), \tag{5}$$

where argmax$(\cdot)$ returns the indices of the maximum value. Each 1×1 peak is mapped to an 8×8 square region in $\mathcal{F}_2(x, y)$ according to the spatial correspondence, where the width and height of $\mathcal{F}_2(x, y)$ are 8 times that of the output feature maps from *Conv_5*. The local feature map $\mathcal{F}_2^c$ for class c can be extracted by

$$\begin{aligned} \mathcal{F}_2^c &= \mathcal{F}_2(x, y), \\ P_x^c \cdot scale \le x &\le (P_x^c + 1) \cdot scale - 1, \\ P_y^c \cdot scale \le y &\le (P_y^c + 1) \cdot scale - 1, \end{aligned} \tag{6}$$

where *scale* is calculated by

$$scale = \frac{W_2}{W_5} = \frac{H_2}{H_5}, \tag{7}$$

and W_2, H_2 and W_5, H_5 are the width and height of the output feature maps of *Conv2_x* and *Conv5_x*, respectively.

Next, we feed $\mathcal{F}_2^c$ to a standard residual block with 3 convolutional layers, whose kernel sizes are 1×1, 3×3, and 1×1, respectively. It aims to further learn motion features for micro actions. The resulting learned features are fed into a GAP and an FC layer with an output size of 1 to get the score O_2^c for the class c. We denote the residual block as $\mathcal{R}(\cdot)$, then O_2^c is computed by

$$O_2^c = FC(GAP(\mathcal{R}(\mathcal{F}_2^c))). \tag{8}$$

Note that the residual block and FC layer share weights for all the action classes. The FC layer outputs the score of class c, which is a single value within [0,1] (ideally 0 or 1). The final output O_2 of LFE on *Conv2_x* is the concatenation of the scores O_2^c of all classes. And, we normalize the different scores across classes by using the Softmax function.

For the lower-level layers *Conv3_x* and *Conv4_x*, we follow the same procedure of computing O_2^c to calculate the scores O_3^c and O_4^c for class c. Specifically, the corresponding local feature maps $\mathcal{F}_3^c$ and $\mathcal{F}_4^c$ extracted from these layers are obtained by mapping each 1×1 peak to a 4×4 square region and a 2×2 square region, respectively. Then, a standard residual block with 3 convolutional layers is used to learn local features from $\mathcal{F}_3^c$, followed by a GAP and an FC to compute O_3^c. Notice that, we do not use residual block for $\mathcal{F}_4^c$, due to its smaller spatial size. Only a GAP and an FC are used to calculate O_4^c. The final outputs O_3 and O_4 of LFEs on these layers are the concatenation of the scores O_3^c and O_4^c for all classes, respectively.

3.4. Loss function

The loss function of our model consists of two parts: the loss of TSM and the loss of MLFE. This is inspired by the existing micro-action recognition work [5] and fine-grained image classification work [30], where the final loss is the weighted sum of the loss from different branches in their CNN models. This design may cover the trade-off between low-level layer features and high-level layer features by training TSM and MLFE with different coefficients in the loss of each. We define the loss function as:

$$\begin{aligned} \mathcal{L}(b, O) &= L(b, O_{TSM}) + \lambda \cdot L(b, O_{MLFE}) \\ &= -\sum_{c=1}^{C} b_c \left(O_{TSM}^c - \log \sum_{j=1}^{C} \exp O_{TSM}^j \right) \\ &\quad - \lambda \cdot \sum_{c=1}^{C} b_c \left(O_{MLFE}^c - \log \sum_{j=1}^{C} \exp O_{MLFE}^j \right), \end{aligned} \tag{9}$$

where $L(\cdot)$ represents the standard cross-entropy loss. The coefficient λ which is the same as the one in Eq. (4) is also the weight for the loss of MLFE, and b_c is the ground-truth label of class c. The model is trained in an end-to-end manner and λ is set to 0.1 empirically.

4. Experiments

In this section, we first describe the evaluation datasets, and then introduce the training & testing setup. Finally, we report the experimental results.

4.1. Datasets

Micro-action video datasets. Paired Egocentric Videos (PEV) [6] is an existing micro-action video dataset, which contains 911 pairs of first-person (actor's view) and third-person (observer's view) videos, each pair containing a micro action performed by person A and watched by another person B. Notice that, we only use the third-person videos for evaluation. There are seven micro actions: *Pointing*, *Attention*, *Positive*, *Negative*, *Passing*, *Receiving* and *Gesture*. The action performer is always sitting or standing when performing the micro actions. Micro Action 10 (MA10) dataset [5] is another micro-action video dataset with 6000 videos, where the action performer is always walking when performing the micro actions. It has 10 micro actions: *Applaud*, *HandSalute*, *Nod*, *ShakeHead*, *PutPalmsTogether*, *HeadScratch*, *ShrugShoulders*, *Stop*, *ThrowUpHands*, and *Waving*. Each video contains only one micro action, and each action has 600 videos. For both PEV and MA10, the evaluation scheme is to conduct a three-fold cross validation on all 911 and 6000 videos, respectively. Specifically, one dataset is split into three subsets of similar size: two subsets are used for training and the remaining subset is used for testing in

Table 1
Performance of the proposed method with different variations on PEV and MA10 datasets. #Params. denotes the total number of network parameters. FLOPs/Video denotes the number of floating-point operations per video.

Variation	PEV	MA10	#Params.	FLOPs/Video
TSM	67.9%	74.1%	24.3 M	33.00 G
TSM + LFE (*Conv2_x*)	70.1%	75.7%	24.4 M	33.35 G
TSM + LFE (*Conv3_x*)	69.5%	75.1%	24.6 M	33.36 G
TSM + LFE (*Conv4_x*)	69.7%	75.2%	24.4 M	33.01 G
TSM + MLFE	**70.8**%	**76.3**%	24.8 M	33.72 G
TSM* + MLFE	69.3%	75.0%	24.8 M	33.72 G

each split. We report the average accuracy over all actions as the recognition performance for each dataset.

General-action video datasets. HMDB51 [15] is a general-action recognition benchmark. It contains 6766 videos from 51 action categories. We follow the original evaluation protocol, using 3 training/testing splits, and report the average accuracy of these splits. UCF101 [16] is another popular general-action recognition benchmark, which contains 13,320 videos from 101 action categories. We adopt the original three training/testing splits for evaluation and report the average accuracy over these splits.

Training and testing setup. The network is fine-tuned from Kinetics [31] pre-trained weights. The training parameters are 50 training epochs, initial learning rate 0.001 (decays by 0.1 at epoch 20 and 40), weight decay 0.0001, batch size 16. For the network input, we follow the normal setting in TSM to sparsely sample 8 RGB frames from an input video as one sample for training and sample two clips per video with the full resolution image where the shorter side is 256 for testing.

4.2. Experimental results on micro-action datasets

Impact of LFE and MLFE. We conduct a series of ablation experiments on both PEV and MA10 datasets, to show the effectiveness of each component in Multi-Layer Local Feature Extractor (MLFE). Here, the temporal shift module video network (TSM) [7] is used as the baseline method for comparison. We apply several variations on TSM: TSM+LFE (*Conv2_x*), TSM+LFE (*Conv3_x*) and TSM+LFE (*Conv4_x*), where these variations only employ one Local Feature Extractor (LFE) on the output feature maps from layer *Conv2_x*, *Conv3_x* and *Conv4_x* in ResNet-50 [13], respectively; TSM+MLFE, in which the proposed method applies LFE on the output feature maps on all these layers.

As shown in Table 1, the performance of the baseline method is improved by adding different variations of the proposed LFE model, where the accuracy of TSM is increased by 2.2%, 1.6%, 1.8%, 2.9% by embedding LFE (*Conv2_x*), LFE (*Conv3_x*), LFE (*Conv4_x*) and MLFE, respectively. These results verify the usefulness of LFE, which can effectively learn local features from lower-level layers for better recognizing micro actions. Among the three variations of LFE, LFE (*Conv2_x*) obtains relatively larger accuracies while the accuracies of the other two variations are very close. This is because, the features extracted from LFE on *Conv2_x* contain higher resolution information compared with the ones in other two layers, and this information is useful for representing small motion. The differences between the results of using LFE on different CNN layers are not large. This may be because, the different levels of features extracted by LFEs are all from the same local region associated with the small motion of interest and these features are not only important but also complementary for representing such motion. By using MLFE, the proposed method achieves the best results, which verifies the effectiveness of combining multi-layer local features for micro-action recognition.

We also conduct an experiment by only using MLFE loss without TSM loss, which is denoted as TSM* + MLFE. We can see from the table that, the performance degrades when using TSM loss without MLFE loss. This may be because that the high-level layer features involves highly semantic information and can be compensated with detailed information of the low-level layer features for better micro-action recognition.

Table 2
Performance of the proposed method with different strategies for fusing the output from multiple LFE on PEV and MA10 datasets.

Fusion	PEV	MA10
Averaging	**70.8**%	**76.3**%
Sum	70.1%	75.8%
Concat.	70.3%	76.0%

We also perform the complexity analysis by comparing the numbers of network parameters and the number of floating-point operations per video for these four variations. By involving LFE (*Conv2_x*), LFE (*Conv3_x*), LFE (*Conv4_x*) and MLFE, the number of parameters are increased by 0.1 M, 0.3 M, 0.1 M and 0.5 M, respectively, while the number of floating-point operations per video are raised by 0.35 G, 0.36 G, 0.01 G and 0.72 G, respectively. These results show that learning the local features from the lower-level layers only modestly increases the model size and computational cost, which further verify the effectiveness of the proposed LFE and MLFE. Note that, the numbers of parameters and FLOPs brought by applying LFE to different CNN layers show no trend. We explain this as follows. The dimensions of the local feature maps extracted by LFEs on different CNN layers do not show certain trend. Specifically, the local feature maps extracted by LFE (*Conv2_x*), LFE (*Conv3_x*) and LFE (*Conv4_x*) have the spatial size of 8×8, 4×4 and 2×2, respectively. Meanwhile, these feature maps have the following numbers of channels: 256, 512 and 1024, respectively. Thus, the model parameters and computational cost brought additionally by the corresponding layers upon these feature maps in different LFEs also show no trend.

We can also see from table that, the proposed method (TSM + MLFE) achieves a 2.0% (0.5 M) gain on the number of parameters and 2.2% (0.72 G) gain on the number of flops. Meanwhile, the proposed method has a 4.3% (2.9 actual percent) gain on the recognition accuracies on PEV dataset, and a 3.0% (2.2 actual percent) gain on the recognition accuracies on MA10 dataset, respectively. These results also show that the proposed method enjoys a reasonable accuracy-cost trade-off.

Visualization on LFE. In Fig. 4, we visually interpret LFE on the lower-level layer *Conv2_x*. By using the class-discriminative information, LFE effectively locates the micro actions of interest (e.g., human head motion and hand motion for *Positive* and *Pointing* actions, respectively) on lower-level layer features. The extracted local feature maps with higher resolution information are discriminative for capturing the small movements of the corresponding body parts.

Impact of fusion strategies. We also study the impact of using different strategies for fusing the outputs from multiple LFEs. Besides using the default Averaging fusion as described in Section 3.2, we also test the other two fusion operations: 1) Sum: taking the sum of class scores from all LFEs; 2) Concat: concatenating the output from GAP operation in each LFE, followed by a fully-connected layer to calculate class scores. Table 2 shows the recognition accuracy of all these three ways of fusion. Averaging leads to better results than Sum or Concat, which justifies the choice of using averaging for fusing the output from multiple LFEs.

Impact of the coefficient. We study the impact of the coefficient (λ), which is used for fusing the output class scores from MLFE with the ones from the TSM, in an enumeration way. Specifically, we select several different values with the even interval of 0.1 for the coefficient. Note that, the coefficient value of 0 means using

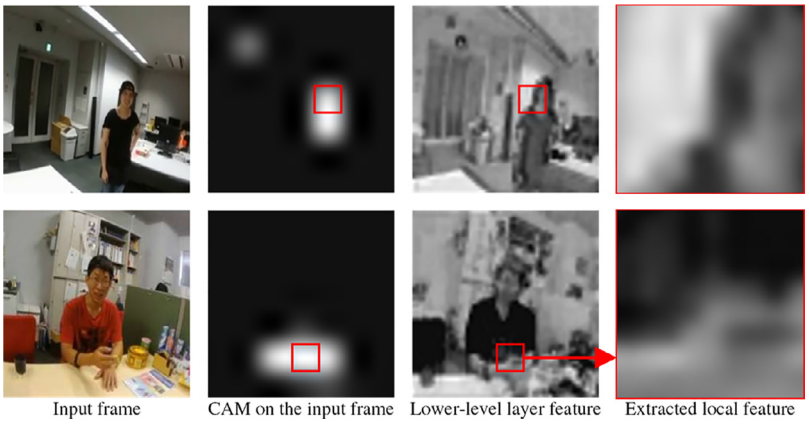

Fig. 4. Visualization of the proposed LFE. The lower-level layer features and extracted local features are enlarged to the same size as the input frames. The second, third and fourth columns are shown in terms of saliency maps.

Table 3
Comparison results on PEV and MA10 datasets. The top part of the table refers to general-action-recognition methods; the middle part of the table refers to micro-action-recognition methods. #Params. denotes the total number of network parameters. FLOPs/Video denotes the number of floating-point operations per video. Flex. denotes the flexible number of temporal clips with spatial crop which are used in C3D. Ms/V denotes milliseconds per video.

Method	Pretraining dataset	Backbone	#Params.	FLOPs/Video	Ms/V	PEV	MA10
iDT [19]	None	–	–	–	–	43.0%	40.1%
C3D [32]	Sports-1M	3D CNNs	79.0 M	296.7 G × Flex.	114.1 × Flex	53.8%	45.8%
TSN [9]	ImageNet	BN-Inception	10.3 M	8.8 G × 3	10.2	77.7%	62.4%
R(2+1)D-Two Stream [8]	Kinetics	3D ResNets	63.8 M	304.0 G × 115	1344.2	65.4%	64.7%
TSM [7]	Kinetics	ResNet-50	24.3 M	4.1 G × 8	12.6	67.9%	74.1%
MPOV [6]	None	–	–	–	–	69.0%	–
STP-RGB [4]	ImageNet	BN-Inception	10.3 M	16.4 G × 3	18.9	60.0%	65.5%
STP-TwoStream [4]	ImageNet	BN-Inception	10.3 M	35.0 G × 3	40.4	80.6%	68.5%
DBN-RGB [5]	ImageNet	BN-Inception	10.8 M	17.2 G × 3	19.8	63.9%	73.5%
DBN-TwoStream [5]	ImageNet	BN-Inception	10.8 M	42.6 G × 3	49.1	81.3%	76.8%
Proposed-RGB	ImageNet	BN-Inception	10.5 M	2.0 G × 8	6.2	69.2%	75.1%
Proposed-RGB	Kinetics	ResNet-50	24.8 M	4.2 G × 8	12.9	70.8%	76.3%
Proposed-TwoStream	ImageNet	BN-Inception	10.5 M	4.0 G × 8	12.4	82.0%	78.8%
Proposed-TwoStream	Kinetics	ResNet-50	24.8 M	8.6 G × 8	26.5	**82.4**%	**79.1**%

only TSM without MLFE, and the coefficient value of 1 indicates that the class scores from TSM and MLFE share the same weight. During the experiment, we use the same experimental setting to train the network several times and report the best testing results for each value of λ, to alleviate the potential effect of the experimental bias during the parameter optimization in training. The results of recognition accuracy are shown in Fig. 5. We can see that $\lambda = 0.1$ leads to the best performance, while the change of the value in certain range (0.1 to 0.5) does not change the performance much. These result indicates that the local features from the low-level layers make a small but essential adjustment upon the final output, and enable the network to capture small motion in a finer manner than just using higher-level layer features as in TSM.

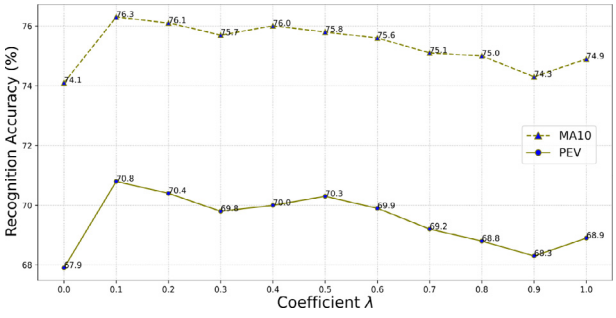

Fig. 5. Performance of the proposed method with different values for coefficient λ on PEV and MA10 datasets.

Comparison results. We choose five general-action-recognition approaches [7–9,19,32] and three micro-action-recognition approaches [4–6] for comparison. The comprehensive comparison results are presented in Table 3. Specifically, for the comparison approaches – iDT, MPOV and STP on PEV, we use the results reported in their works. For all the other results of the comparison methods, we use their released codes with default settings to conduct experiments and ensure the loss converges during the training process. Note that, MPOV cannot be tested on MA10, since it needs the point-of-view features from first-person videos which are not contained in MA10. "RGB" and "TwoStream" after STP, DBN and proposed method denote the input modalities of only RGB frame and both RGB frame & optical flow, respectively. For "TwoStream", we follow TSN to fuse the class scores from RGB-based and flow-based streams.

The proposed method obtains the recognition accuracies of 70.8% and 76.3% on the PEV and MA10 dataset, respectively. The performance of the proposed method further improves with the additional optical-flow input, and achieves 82.4% and 79.1% on the PEV and MA10 datasets. The proposed method outperforms all the comparison methods on both datasets, by using only RGB frame as the input modality. When using both RGB frame and optical flow as input, it still achieves the state-of-the-art recognition accuracy on both datasets. When pre-trained on ImageNet, the proposed method still outperforms all the comparison methods on both datasets, by using only RGB frame or both RGB frame and

Table 4
Comparison results against several existing action recognition methods on UCF101 and HMDB51 datasets. The top part of the table refers to handcrafted-feature-based methods; the rest part of the table refers to deep-learning-based methods.

Method	Pretraining dataset	Backbone	UCF101	HMDB51
iDT [19]	None	–	85.9%	57.2%
MoFAP [33]	None	–	88.3%	61.7%
C3D [32]	Sports-1M	3D CNNs	85.2%	51.6%
Two Stream CNNs [20]	ImageNet	2D CNNs	88.0%	59.4%
LRCN [21]	ImageNet	2D CNNs + LSTM	82.7%	–
Two-Stream Fusion [34]	ImageNet	VGG-16	92.5%	65.4%
Spatiotemp. MultiNet [35]	ImageNet	ResNet-152	94.2%	68.9%
TSN [9]	ImageNet	BN-Inception	94.2%	69.4%
DS-CNN [23]	ImageNet	BN-Inception	95.8%	73.9%
TSM [7]	Kinetics	ResNet-50	95.9%	73.5%
STCC [36]	ImageNet + Kinetics	ResNet-34	96.9%	75.2%
CFST [37]	ImageNet + Kinetics	DenseNet-121	96.9%	75.7%
$R(2+1)D$-TwoStream [8]	Kinetics	3D ResNets	**97.3**%	78.7%
Proposed	Kinetics	ResNet-50	97.2%	**79.4**%

optical flow as the input modality. These results verify the effectiveness of the proposed method which learns local features from lower-level CNN layers for enhancing micro-action recognition. We use both input modalities for the remaining experiments since they leads to the best results.

Besides the recognition accuracies, we also compare the complexity for all deep-learning-based methods in terms of the number of model parameters and the number of floating-point operations per videos. Specifically for counting the floating-point operations per videos, we follow DBN to report the number of floating-point operations per temporal clip with spatial crop. The proposed method improves the recognition accuracies of the baseline TSM by 14.5 percentage points and 5 percentage points on two datasets respectively, while only introduces a very small amount of additional parameters (0.5 M) and roughly doubled the amount of floating-point operations (4.5 G $\times$ 8). The proposed method enjoys the smallest level of computational cost (FLOPs) when compared with other two deep-learning based micro-action-recognition methods. We also compare the time complexity in terms of milliseconds per video (Ms/V). The comparison results about the time complexity is shown in the sixth column in the table. We can see that the proposed method also enjoys the smallest level of time complexity (Ms/V) when compared with other two deep-learning based micro-action-recognition methods. These results further verify the efficiency of the proposed method.

4.3. Experimental results on general-action datasets

We compare the proposed method with several state-of-the-art action-recognition methods on two popular general-action benchmarks – UCF101 and HMDB51. The results reported in their respective papers are used for comparison. We can see from Table 4, the proposed method improve the recognition accuracies by 1.3 percentage points and 5.9 percentage points on UCF101 and HMDB51 respectively, when compared with the baseline TSM. The proposed method outperforms most of the comparison methods on both UCF101 and HMDB51. Meanwhile, the performance of the proposed method is comparable with the recent state-of-the-art methods $R(2+1)D$-TwoStream on UCF101. These results show the wide usefulness of the proposed method on general-action videos.

5. Conclusion

In this paper, we propose a new method to recognize micro actions in videos, by learning the local features from multiple lower-level layers of CNNs. In this method, we utilized class-discriminative information as a guidance to locate micro-motion related regions in lower-level convolutional layers. The resulting local feature maps are further learned via residual convolutional blocks, followed by global averaging pooling and fully-connected layers to compute class scores. Finally, these scores from multiple lower-layers are fused via averaging, and further combined with the class scores from higher-level layers via weighted sum. Experimental analysis on two micro-action video datasets verified the effectiveness of the proposed method. The proposed method also achieved promising results on two general-action video datasets.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Y. Han, P. Zhang, T. Zhuo, W. Huang, Y. Zhang, Going deeper with two-stream convnets for action recognition in video surveillance, Pattern Recognit. Lett. 107 (2018) 83–90.
[2] W. Hao, Z. Zhang, Spatiotemporal distilled dense-connectivity network for video action recognition, Pattern Recognit. 92 (2019) 13–24.
[3] S.R. Mishra, T.K. Mishra, G. Sanyal, A. Sarkar, S.C. Satapathy, Real time human action recognition using triggered frame extraction and a typical CNN heuristic, Pattern Recognit. Lett. 135 (2020) 329–336.
[4] Y. Mi, S. Wang, Recognizing micro actions in videos: learning motion details via segment-level temporal pyramid, ICME, 2019.
[5] Y. Mi, X. Zhang, Z. Li, S. Wang, Dual-branch network with a subtle motion detector for microaction recognition in videos, IEEE TIP 29 (2020) 6194–6208.
[6] R. Yonetani, K.M. Kitani, Y. Sato, Recognizing micro-actions and reactions from paired egocentric videos, CVPR, 2016.
[7] J. Lin, C. Gan, S. Han, TSM: temporal shift module for efficient video understanding, ICCV, 2019.
[8] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, M. Paluri, A closer look at spatiotemporal convolutions for action recognition, CVPR, 2018.
[9] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, L. Van Gool, Temporal segment networks: towards good practices for deep action recognition, ECCV, 2016.
[10] D. Garcia-Gasulla, F. Parés, A. Vilalta, J. Moreno, E. Ayguadé, J. Labarta, U. Cortés, T. Suzumura, On the behavior of convolutional nets for feature extraction, JAIR 61 (2018) 563–592.
[11] H. Zhang, K. Wang, Y. Tian, C. Gou, F.-Y. Wang, MFR-CNN: incorporating multi-scale features and global information for traffic object detection, IEEE TVT 9 (2018) 8019–8030.
[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, SSD: single shot multibox detector, ECCV, 2016.
[13] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, CVPR, 2016.
[14] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba, Learning deep features for discriminative localization, CVPR, 2016.
[15] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, T. Serre, HMDB: a large video database for human motion recognition, ICCV, 2011.
[16] K. Soomro, A.R. Zamir, M. Shah, Ucf101: a dataset of 101 human actions classes from videos in the wild, arXiv (2012).
[17] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, CVPR, 2005.

[18] P. Scovanner, S. Ali, M. Shah, A 3-dimensional sift descriptor and its application to action recognition, ACMMM, 2007.
[19] H. Wang, C. Schmid, Action recognition with improved trajectories, ICCV, 2013.
[20] K. Simonyan, A. Zisserman, Two-stream convolutional networks for action recognition in videos, NIPS, 2014.
[21] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, T. Darrell, Long-term recurrent convolutional networks for visual recognition and description, CVPR, 2015.
[22] B. Martinez, D. Modolo, Y. Xiong, J. Tighe, Action recognition with spatial-temporal discriminative filter banks, ICCV, 2019.
[23] Y. Li, K. Li, X. Wang, Deeply-supervised CNN model for action recognition with trainable feature aggregation, IJCAI, 2018.
[24] P. Tang, H. Wang, S. Kwong, G-ms2f: googlenet based multi-stage feature fusion of deep CNN for scene recognition, Neurocomputing 225 (2017) 188–197.
[25] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, CVPR, 2017.
[26] X. Lin, Q. Zou, X. Xu, Action-guided attention mining and relation reasoning network for human-object interaction detection, IJCAI, 2021.
[27] W. Yang, H. Huang, Z. Zhang, X. Chen, K. Huang, S. Zhang, Towards rich feature discovery with class activation maps augmentation for person re-identification, CVPR, 2019.
[28] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, ICML, 2015.
[29] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, ICML, 2010.
[30] Y. Wang, V.I. Morariu, L.S. Davis, Learning a discriminative filter bank within a CNNfor fine-grained recognition, CVPR, 2018.
[31] J. Carreira, A. Zisserman, Quo vadis, action recognition? A new model and the kinetics dataset, CVPR, 2017.
[32] D. Tran, L. Bourdev, R. Fergus, L. Torresani, M. Paluri, Learning spatiotemporal features with 3Dconvolutional networks, ICCV, 2015.
[33] L. Wang, Y. Qiao, X. Tang, MoFAP: a multi-level representation for action recognition, IJCV 3 (2016) 254–271.
[34] C. Feichtenhofer, A. Pinz, A. Zisserman, Convolutional two-stream network fusion for video action recognition, CVPR, 2016.
[35] C. Feichtenhofer, A. Pinz, R.P. Wildes, Spatiotemporal multiplier networks for video action recognition, CVPR, 2017.
[36] C. Cheng, C. Zhang, Y. Wei, Y.-G. Jiang, Sparse temporal causal convolution for efficient action modeling, ACMMM, 2019.
[37] H. Wu, Z.-J. Zha, X. Wen, Z. Chen, D. Liu, X. Chen, Cross-fiber spatial-temporal co-enhanced networks for video action recognition, ACMMM, 2019.