Learning SpatioTemporal and Motion Features in a Unified 2D Network for Action Recognition

Mengmeng Wang, Jiazheng Xing, Jing Su, Jun Chen, and Yong Liu, *Member, IEEE*

Abstract—Recent methods for action recognition always apply 3D Convolutional Neural Networks (CNNs) to extract spatiotemporal features and introduce optical flows to present motion features. Although achieving state-of-the-art performance, they are expensive in both time and space. In this paper, we propose to represent both two kinds of features in a unified 2D CNN without any 3D convolution or optical flows calculation. In particular, we first design a channel-wise spatiotemporal module to present the spatiotemporal features and a channel-wise motion module to encode feature-level motion features efficiently. Besides, we provide a distinctive illustration of the two modules from the frequency domain by interpreting them as advanced and learnable versions of frequency components. Second, we combine these two modules and an identity mapping path into one united block that can easily replace the original residual block in the ResNet architecture, forming a simple yet effective network dubbed STM network by introducing very limited extra computation cost and parameters. Third, we propose a novel Twins Training framework for action recognition by incorporating a correlation loss to optimize the inter-class and intra-class correlation and a siamese structure to fully stretch the training data. We extensively validate the proposed STM on both temporal-related datasets (i.e., Something-Something v1 & v2) and scene-related datasets (i.e., Kinetics-400, UCF-101, and HMDB-51). It achieves favorable results against state-of-the-art methods in all the datasets.

Index Terms—Action recognition, frequency illustration, motion features, spatiotemporal features, twins training framework

1 Introduction

With the rapid development of the cloud and edge computing, we are used to engaging in social platforms and living under the cameras. A vast amount of videos are recorded every day, attracting more and more researchers to the video understanding field. One of the most essential tasks in video understanding is human action recognition, which aims to recognize the human actions in videos. The most significant features for action recognition are two complementary features, the *spatiotemporal* and *motion* features where the former is responsible for temporal enhanced appearance modeling and the latter captures motion information of the action subject.

Spatiotemporal features are vital for action recognition since they encode the relationship of spatial appearance features from different timestamps. For example, for a "*long jump*" action, the video may start with a running athlete, and the sandpit always shows in the end. It will be easily mistaken as "*running*" if the features of the sandpit are not felicitously integrated into the former frames' appearance features. Existing approaches usually model the spatiotemporal features through 3D CNNs [1], [2], [3], [4], [5], [6] or (2+1)D frameworks [7], [8], [9]. 3D CNNs extend ordinary 2D convolution with an extra temporal dimension so that 3D convolution is able to represent spatiotemporal features intuitively. Nevertheless, expanding the convolution kernel from 2D to 3D will inevitably increase the computational cost by an order of magnitude, and the support of 3D convolutions in different platforms is not good as conventional 2D convolutions, limiting its real applications. The (2+1)D architectures like S3D [8], P3D [9] and R(2+1)D [7] are proposed to extract spatiotemporal features by factorizing a 3D convolution into a 1D temporal convolution and a 2D spatial convolution. They can decompose the 3D CNNs back to 2D CNNs. Lin *et al*. propose TSM [10] to further improve the computational efficiency by shifting part of the channels along the temporal dimension and facilitating information exchange among neighboring frames. Though these methods successfully balance the heavy computation of 3D CNNs, their performance remains unsatisfactory if the motion features are not utilized.

Motion features present motion characteristics between neighboring frames, which are important for recognizing temporal-related actions. For instance, when given a frame like the first image in Fig. 1a, it is hard to tell that the action is "*pulling a plug*" or "*inserting a plug*" without the motion information. The main difference between the spatiotemporal features and the motion features is that the former focuses on the temporal aggregation of neighboring appearance features, while the latter pays more attention to the motion edges of adjacent features and suppresses their same static appearance features. As demonstrated in many works [2], [5], [11], these two kinds of features are complementary to each other. It is a typical way to represent

- *Mengmeng Wang, Jiazheng Xing, Jun Chen, and Yong Liu are with the Laboratory of Advanced Perception on Robotics and Intelligent Learning, College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, Zhejiang, China. E-mail: {mengmengwang, junc}@zju.edu.cn, jzxing83@gmail.com, yongliu@iipc.zju.edu.cn.*
- *Jing Su is with the Department of Optical Science and Engineering, Fudan University, Shanghai 200433, China. E-mail: 16110720001@fudan.edu.cn.*

Manuscript received 30 June 2021; revised 29 Jan. 2022; accepted 30 Apr. 2022. Date of publication 10 May 2022; date of current version 3 Feb. 2023.
This work was supported by the National Natural Science Foundation of China under Grant 61771193.
(Corresponding author: Yong Liu.)
Recommended for acceptance by O. Camps.
Digital Object Identifier no. 10.1109/TPAMI.2022.3173658

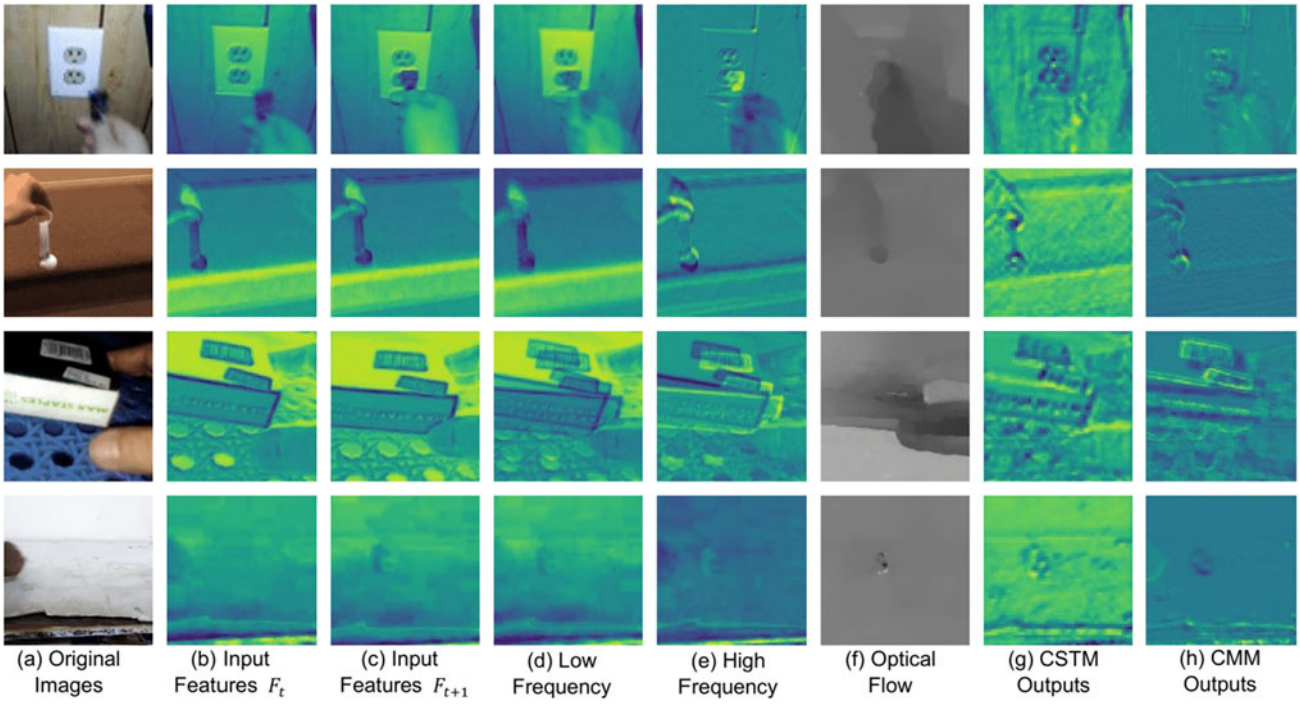

Fig. 1 Representations of STM. (a) is the first frame of all inputs (8 frames). (b) and (c) are the input features maps $\mathbf{F}_t$ and $\mathbf{F}_{t+1}$ (here we show examples of t=0) from Conv_1 block, which is just before the first STM block. (d) and (e) are the temporal low-frequency and high-frequency components of adjacent input features maps $\mathbf{F}_t$ and $\mathbf{F}_{t+1}$ (please refer to Sec. 3.3 for details), which show the appearance information and the motion edges, respectively. (f) represents the optical flows extracted by TV-L1. (g) and (h) are the output spatiotemporal feature maps of CSTM and the output motion feature maps of CMM, extracted from the first CSTM and CMM block. It could be found that the CSTM features share some common characteristics with the temporal low-frequency components (d), i.e., the spatiotemporal features learned by CSTM could capture the significant spatial features enhanced by multiple temporal inputs. Moreover, the CMM features could present the motion features of the action subject, similar to the optical flows (f) and the high-frequency components (e).

motion features by absorbing optical flows within two-stream neural networks [12], [13], [14], [15], [16]. Optical flow is an effective motion representation to describe object/scene movement. To be precise, it is the pattern of apparent motion in a visual scene caused by the relative motion between an observer and a scene. A classic two-stream network [12], [13] consists of a spatial stream with RGB frames as input and a flow stream with optical flows as input. The spatial stream models the appearance features (not spatiotemporal features) without considering the temporal information. The flow stream is designed to model the motion cues with almost the same network structure as the spatial stream. Since the input optical flows are extracted between the original neighboring video frames, it lacks the ability to capture the long-term motion relationship. To conquer this problem, the two-stream networks [13], [14] simply pool the features or average the predicted results of all the input optical flows. However, the extraction of optical flow is expensive in both time and space, and the temporal stream brings extra parameters, limiting vast industrial applications in the real world. In order to avoid the calculation of optical flows, Optical Flow guided Feature (OFF) [17] contains a set of operators including Sobel and element-wise subtraction for OFF generation. MFNet [18] adopts five fixed motion filters as a motion block to find feature-level motion features between two adjacent time steps. However, the performance of these methods is limited since they lack spatiotemporal representations, which benefit a lot to appearance modeling.

Our key insight is that both spatiotemporal features and motion features are indispensable and could be integrated into a unified 2D network so that we do not need any 3D convolution or optical flow calculation. Specifically, we propose a Channel-wise Spatiotemporal Module (CSTM) to present the spatiotemporal features and a Channel-wise Motion Module (CMM) to learn the motion representations. We visualize several examples of these two features in Fig. 1. Compared to the original input features (b) and (c), the CSTM (g) has learned the spatiotemporal representations, which extract appearance features strengthened by multiple timestamps rather than a single frame. As for the CMM (h), it captures the feature-level motion representations of the action subject similar to optical flows (e). We have theoretically explored the relationship between computing optical flows and our motion features and found that the learnable channel-wise kernels in our CMM could learn more task-relevant motion representations. Then, we combine CSTM and CMM as well as an identity mapping path together to form a united block and insert it into the ordinary 2D ResNet by replacing the original residual block with negligible extra parameters (0.8%). It comes out the expected form of our method, a simple yet effective 2D network for video action recognition, termed as *STM* network, which is short for **S**patio**T**emporal and **M**otion features integrated network.

Besides, we find CSTM and CMM could be illustrated from the temporal frequency domain. As shown in Fig. 1, the features encoded by CSTM have a similar denominator

with the low-frequency components (d), which present the scene information of a video. Moreover, CMM learns similar information to the high-frequency representations (e), which depicts the motion edges. Interestingly, we find CSTM and CMM could be interpreted as advanced and learnable versions of low-frequency and high-frequency components, respectively. We demonstrate that simply degenerating CSTM and CMM into frequency components could also achieve a satisfactory performance but not as good as our method.

Furthermore, we explore the training process of STM and propose a novel Twins Training framework, which aims to fully exploit the training data and strengthen our model. We believe that both the architecture and the training framework are significant for the video action recognition task. Nevertheless, most of the previous works in the video action recognition field focus on the architecture design but neglect the importance of the training pipeline. Multigrid [19] has also noticed this problem and proposed a training strategy mainly to speed up the training process while keeping the performance. Differently, our Twins Training framework purses to improve the performance further. Specifically, we introduce a siamese architecture and a correlation loss into the regular training process. When inputting a group of sampled frames, we first generate two groups of distortions with two groups of different augmentations. Then the two distortions will go through a shared siamese network to obtain their representations and the two outputs are used to calculate the normal classification loss and also normalized to compute a correlation loss. The siamese structure could excavate the input training data by applying two groups of different distortions rather than only one time in the regular training process. The correlation loss could help to enlarge the inter-class distance and shrink the intra-class distance. As a result, the performance will be further improved. Note that the framework is only used in training to strengthen the network and will not influence the normal inference process. We experiment with the proposed Twins Training on not just our STM but also representative TSN [12] and TSM [10] to demonstrate its validity. It is proved to be effective on all these methods.

The main contributions of our work can be summarized as follows:

- We propose a simple and effective network named STM for video action recognition, integrating spatiotemporal features and motion features together in a unified 2D CNN.
- We design a channel-wise spatiotemporal module and a channel-wise motion module to encode complementary spatiotemporal and motion representations. Moreover, a distinctive illustration of the two modules is provided from the frequency domain.
- We propose a novel Twins Training framework, which not only makes the utmost of the training data but also decouples the inter-class correlation and reinforces the intra-class correlation, therefore further strengthening our model.
- Extensive experiments demonstrate that our method consistently outperforms or obtains a comparable performance with the state-of-the-art methods on several public benchmark datasets, including Something-Something [20], Kinetics [2], UCF101 [21] and HMDB-51 [22].

This paper builds upon our conference paper [23] and significantly extends it in several aspects. First, we propose a novel training framework that enlarges the inter-class correlation and decreases the intra-class distance, therefore further improving our model. Second, we verify the generalization ability of the proposed framework on not only our STM but also other two representative methods, TSN [14] and TSM [10]. Third, we investigate the relationship of our CMM and optical flow theoretically to illustrate the insight of CMM. Fourth, an in-depth analysis of the proposed method from a different perspective, frequency domain. Fifth, we re-elaborate the essentiality and importance of our insight and provide a more extensive literature survey in deep video action recognition. Last but not least, we enrich the experiments by (1) adding more state-of-the-art approaches for comparison, (2) implementing variants of our preliminary version, and (3) conducting more extensive ablation studies.

2 Related Works

With the great success of deep convolution networks in the computer vision area, a large number of CNN-based methods have been proposed for action recognition and have gradually surpassed the performance of traditional methods [24], [25]. We mainly introduce the CNN-based approaches and classify them into three categories as follows. Besides, we discuss the spatiotemporal modeling of some related tasks beyond our target supervised short-range video action recognition task.

2.1 Two-Stream Networks

A sequence of advances adopt 2D CNNs as the backbone and classify a video by simply aggregating frame-wise prediction [26]. However, these methods only model the appearance feature of each frame independently while ignoring the dynamics between frames, which results in inferior performance when recognizing temporal-related videos.

To handle the mentioned drawback, two-stream-based methods [4], [13], [14], [15], [16] are introduced by modeling appearance and dynamics separately with two networks and fusing two streams through the middle or at last. Among these methods, Simonyan *et al.* [27] first proposed the two-stream ConvNet architecture with both spatial and temporal networks. Temporal Segment Networks (TSN) [12], [14] proposed a sparse temporal sampling strategy for the two-stream structure and fused the two streams by a weighted average at the end. Feichtenhofer *et al.* [16], [28] studied the fusion strategies in the middle of the two streams in order to obtain the spatiotemporal features. However, these types of methods mainly suffer from two limitations. First, these methods need pre-compute optical flow, which is expensive in both time and space. Taking Kinetics-400 dataset as an illustrative example, storing all the optical flow images requires 4.5 TB disk space [29]. Such a massive amount of data would lead to a waste of GPU resources and longer experiment cycles. In addition, pre-computing optical flow is not cheap, which

means all the two-stream methods are not real-time. Second, the learned feature and final prediction from multiple segments are fused simply using weighted or average sum, making it inferior to temporal-relationship modeling.

2.2 3D CNN-Based Networks

It is intuitive to learn spatiotemporal features from RGB frames directly with 3D CNNs [1], [2], [3], [5], [30], [31] which extend the common 2D CNNs with an extra temporal dimension. [32] was a pioneer work to use 3D CNNs for action recognition, but the network was not deep enough to show its potential. C3D [1] extended [32] to a deeper 3D network, VGG16. However, with tremendous parameters to be optimized and a lack of high-quality, large-scale datasets, the performance of C3D remains unsatisfactory. I3D [2] inflated the ImageNet pre-trained 2D kernel into 3D to capture spatiotemporal features and modeled motion features with another flow stream. I3D achieved very competitive performance in benchmark datasets with the help of the large-scale Kinetics dataset and the two-stream setting. Since 3D CNNs try to learn local correlation along the input channels, STCNet [30] inserted its STC block into 3D ResNet to capture both spatial-channel and temporal-channel correlation information throughout network layers. Slowfast [5] involved a slow path to capture spatial semantics and a fast path to capture motion at fine temporal resolution. D3D [3] introduced knowledge distillation to tune the spatial stream to mimic the temporal stream, effectively combining both models into a single stream. More recently, Feichtenhofer proposed X3D [6] to progressively expand a tiny base 2D image architecture into a spatiotemporal one by expanding along multiple possible axes. Although 3D CNN-based methods have achieved state-of-the-art performance, they are still hard to train and challenging to deploy. For training, it requires lots of GPUs and the experiment cycle is very long. For instance, a standard SlowFast network trained on Kinetics400 dataset using a high-end 8-GPU machine takes 10 days [29], or a 64-GPU machine takes 54.3 hours [33] to complete. For deployment, 3D convolution is not as well supported as 2D convolution for different platforms.

2.3 Compute-Efficient Models

To reduce the high computation cost of 3D CNNs, several methods are proposed to find the trade-off between precision and speed [7], [8], [9], [11], [34], [35], [36], [37], [38], [39]. Tran *et al.* [7] and Xie *et al.* [8] discussed several forms of spatiotemporal convolutions, including employing 3D convolution in early layers and 2D convolution in deeper layers (bottom-heavy) or reversed the combinations (top-heavy). P3D [9], S3D [8] and R(2+1)D [7] tried to reduce the cost of 3D convolution by decomposing it into 2D spatial convolution and 1D temporal convolution. TSM [10] further introduced the temporal convolution by shifting part of the channels along the temporal dimension. Our proposed CSTM module is similar to these methods in learning spatiotemporal features, while we employ channel-wise 1D convolution to capture different temporal relationships for different channels.

Though the above methods are successful in balancing the heavy computation of 3D CNNs, they inevitably need the help of two-stream networks with a flow stream to incorporate the motion features to obtain their best performance [10]. Motion information is the key difference between video-based recognition and image-based recognition task. However, calculating optical flow [40] is expensive in both time and space. Recently many approaches have been proposed to estimate optical flow with CNN [41], [42], [43], [44] or explored alternatives of optical flow [14], [17], [18], [45]. TSN frameworks [14] involved RGB difference between two frames to represent motion in videos. Zhao *et al.* [45] used cost volume processing to model apparent motion. Optical Flow guided Feature (OFF) [17] contained a set of operators including Sobel and element-wise subtraction for OFF generation. MFNet [18] adopted five fixed motion filters as a motion block to find feature-level temporal features between two adjacent time steps. CorrNet [18] proposed a learnable correlation operator for motion modeling. Our proposed CMM module is also designed for finding better yet lightweight alternative motion representation in the feature level to avoid the calculation of optical flows. Our motion filters are all learnable and we learn different motion features for different channels between every two adjacent input frames.

A recent work TEA [11] is the most similar work to our method. It uses motion features to recalibrate the spatiotemporal features and enhance the motion pattern. While our STM simply and directly adds the spatiotemporal features and motion encoding together by treating the two kinds of features as independent and complementary representations.

2.4 Temporal Modeling in Related Tasks

Spatiotemporal modeling is hot in several video understanding tasks like complex action recognition and self-supervised action recognition. A comprehensive review of these tasks is beyond the scope of this paper and we only briefly introduce them here. Complex action recognition contains a set of one-actions with a weak temporal pattern that serves a specific purpose. Since complex actions always take much longer to unfold, temporal modeling is key to avoid missing crucial parts. A representative method in this task is Timeception [46], which also introduces channel-wise temporal convolution for temporal modeling. Our CSTM could be interpreted as a special Timeception layer when fixing the temporal convolution kernel size to 3. However, significant motion features are not mentioned in Timeception since they focus on the long, complex actions. Self-supervised action recognition seeks to learn spatiotemporal features from unlabeled videos. Without precise annotations, the methods in this task focus on designing proxy tasks like temporal order verification [47], motion/appearance prediction statistics [48], and relative speed perception [49]. Since they are all designed for unsupervised instance discrimination tasks, they mainly focus on the loss and label design with existing backbones like ResNet [50] and C3D [1] but neglect the architecture improvement, which is our core insight.

3 Method

In this section, we elaborate the technical details of our approach. First, we describe the proposed CSTM and CMM to show how to perform the channel-wise spatiotemporal

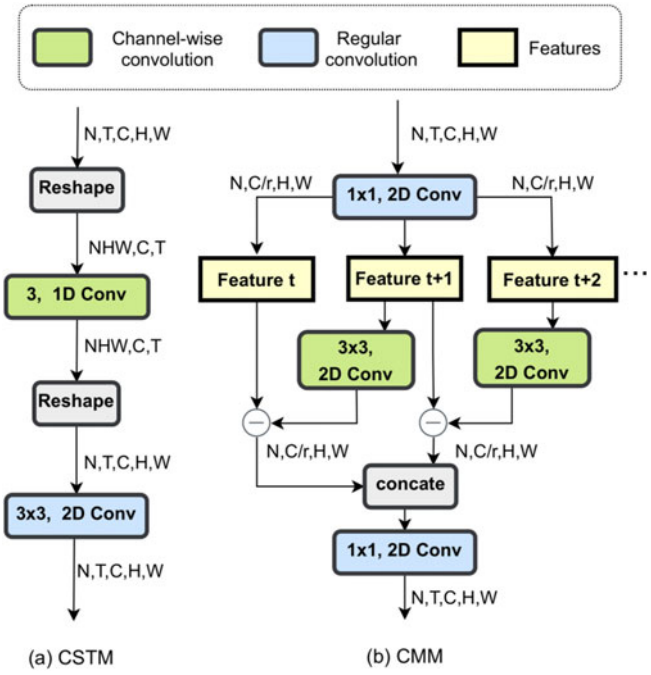

Fig. 2. The architecture of the Channel-wise SpatioTemporal Module and Channel-wise Motion Module. The feature maps are shown as the shape of their tensors. "⊖" denotes element-wise subtraction.

fusion and extract the feature-level motion information, respectively. Then, we give an in-depth illustration of the two modules from the frequency domain. Next, we present how to combine these two modules and how to form our STM network. Afterward, we will introduce the proposed Twins Training framework in detail.

3.1 Channel-Wise SpatioTemporal Module

To represent the spatiotemporal features, previous methods intuitively utilize 3D CNNs [1], [2], [3], [5], [30] or leverage (2+1)D structures [7], [8], [9]. According to the above analyses, the tremendous parameters of 3D CNNs make it hard to train and deploy. Therefore, we choose the (2+1)D structure to formulate our Channel-wise SpatioTemporal Module (CSTM).

CSTM is designed for efficient spatial and temporal modeling. It could extract rich spatiotemporal features, which can significantly boost action recognition performance. As illustrated in Fig. 2a, a 1D channel-wise temporal convolution and a 2D spatial convolution operation are applied sequentially inside CSTM. Specifically, given an input feature map $\mathbf{F} \in \mathbb{R}^{N\times T\times C\times H\times W}$, where N, T, C, H, W are the batch size, time, spatial height and width, respectively, we first reshape $\mathbf{F}$ as: $\mathbf{F} \rightarrow \mathbf{F}^* \in \mathbb{R}^{NHW\times C\times T}$ and then apply a channel-wise 1D convolution on the T dimension to model the temporal relationship. There are mainly two advantages to adopting the channel-wise convolution rather than the ordinary convolution for temporal modeling. First, compared to the ordinary convolution, the computation cost can be reduced by a factor of G, where G is the number of groups. In our settings, G equals the number of input channels. Second, for the feature map $\mathbf{F}^*$, the semantic information of different channels is typically different [51]. We believe that the combination of temporal information for different channels should be independent, which is also validated in the experiments. Thus the channel-wise convolution is adopted to learn independent kernels for each channel. Formally, the channel-wise temporal fusion operation can be formulated as

$$\mathbf{G}_{c,t} = \sum_i \mathbf{K}_{c,i}\mathbf{F}^*_{c,t+i}, \tag{1}$$

where t, c denote temporal, channel dimensions of the feature map respectively. $\mathbf{K}_{c,i}$ are temporal combination kernel weights of cth channel, i is the index of temporal kernel, $\mathbf{F}^*_{c,t+i}$ is the input cth channel features and $\mathbf{G}_{c,t}$ is the output channel-wise temporal fusion features. Here the temporal kernel size of $\mathbf{K}_{c,i}$ is set to 3 thus $i \in [-1, 1]$. Next, $\mathbf{G}$ will be reshaped to the original input shape (i.e., $[N, T, C, H, W]$) and model spatial information via a 2D convolution whose kernel size is 3x3.

We visualize the output feature maps of CSTM to help understand this module in Fig. 1f. We can find that the CSTM has learned the spatiotemporal features which absorb important appearance information from different timestamps.

3.2 Channel-Wise Motion Module

As discovered in [2], [52], the motion features are complementary to spatiotemporal features. Therefore, we propose a lightweight Channel-wise Motion Module (CMM) to extract feature-level motion patterns between adjacent input frames. Note that our goal is to find the motion representation that can help to recognize actions in an efficient way rather than accurate motion information (optical flow) between two frames. Therefore, we will only use the RGB frames and not involve any pre-computed optical flow.

Given the input feature maps $\mathbf{F} \in \mathbb{R}^{N\times T\times C\times H\times W}$, we will first leverage a 1x1 convolution layer to reduce the channels by a factor of r to ease the computing cost and we still use $\mathbf{F}$ to present the features in the following for simplicity. Then we generate feature-level motion information from every two consecutive feature maps. The feature-level motion representations at time step t could be approximately considered as the difference between the two adjacent features $\mathbf{F}_t$ and $\mathbf{F}_{t+1}$. Instead of directly subtracting the original features, we employ learnable channel-wise transformation on $\mathbf{F}_{t+1}$ first and then perform the subtraction operation. Taking $\mathbf{F}_t$ and $\mathbf{F}_{t+1}$ for example, for every channel c, we first apply a 2D channel-wise convolution on $\mathbf{F}_{t+1,c}$ and then subtracts it from $\mathbf{F}_{t,c}$ to obtain the motion representation $\mathbf{H}_{t,c}$

$$\mathbf{H}_{t,c} = \mathbf{F}_{t,c} - \sum_{i,j} \mathbf{K}_{c,i,j}\mathbf{F}_{t+1,c,h+i,w+j}, \tag{2}$$

where c, t, h, w denote channel, temporal and two spatial dimensions of the feature map. $\mathbf{K}_{c,i,j}$ denotes the cth motion filter with the subscripts i, j denote the spatial indices of the kernel. Here $\mathbf{K}^c_{i,j}$ is shared among different timestamps and its kernel size is set to 3×3 thus $i, j \in \{-1, 0, 1\}$. As shown in Fig. 2b, we perform the proposed CMM to every two adjacent feature maps over the temporal dimension, i.e., $\mathbf{F}_t$ and $\mathbf{F}_{t+1}$, $\mathbf{F}_{t+1}$ and $\mathbf{F}_{t+2}$, etc. Therefore, the CMM will produce $T-1$ motion representations. To keep the temporal

size compatible with the input feature maps, we simply use zero to represent the motion information of the last time step and then concatenate them together over the temporal dimension. In the end, another 1x1 2D convolution layer is applied to restore the number of channels to C.

We find that the proposed CMM can boost the performance of the whole model even though the design is quite simple, which proves that the motion features obtained with CMM are complementary to the spatiotemporal features from CSTM. We visualize the motion features learned by CMM in Fig. 1g. From which we can see that compared to the output of CSTM, CMM is able to capture the motion features of the action subjects between neighboring frames.

Relationship With Optical Flow. Comparing (g) and (f) of Fig. 1, we could find that the motion features learned by CMM look similar to the optical flows. This phenomenon makes us think more deeply about the relationship between our motion features and optical flows. Typically, the brightness consistency constraint of optical flow is defined as follows:

$$\mathbf{I}(x, y, t) = \mathbf{I}(x + i, y + j, t + k), \tag{3}$$

where $\mathbf{I}(x, y, t)$ denotes the pixel at the location (x, y) of a frame at time t. For frames t and $(t + k)$, i and j are the spatial pixel displacement in horizontal and vertical axes respectively. It assumes that for any point that moves from (x, y) at frame t to $(x + i, y + j)$ at frame $t + k$, its brightness keeps unchanged over time. Then, the optical flow (i, j) that meets Eq. (3) is calculated between two image frames at time t and $t + k$ at every location of an image, finding the optimal solution (i^*, j^*) through an optimization technique. When we extend Eq. (3) to the feature space by replacing an image $\mathbf{I}(x, y, t)$ with the corresponding feature maps $\mathbf{F}_c(x, y, t)$ (cth channel) and define a residual features $\mathbf{R}_{op}(x, y, t + k)$ as follows:

$$\mathbf{R}_{op}(x, y, t + k) = \mathbf{F}_c(x, y, t) - \mathbf{F}_c(x + i, y + j, t + k). \tag{4}$$

We use $\mathbf{R}_{op}$ to abbreviate $\mathbf{R}_{op}(x, y, t + k)$ in the following. Considering that one pixel in the feature space at a higher hierarchy of a CNN can capture larger movement than the images as it looks at a larger receptive field, we restrict the searching space in a 3×3 local window $\mathbb{P} = \{-1, 0, 1\}$. After that, solving an optical flow problem is to find the optimal solution to minimize the residual $\mathbf{R}_{op}$

$$\begin{aligned} \mathbf{R}_{op} &= \mathbf{F}_c(x, y, t) - \sum_{i,j \in \mathbb{P}} \mathbf{O}_{c,i,j} \mathbf{F}_c(x + i, y + j, t + k), \\ s.t. &\mathbf{O}_{c,i,j} \in \{0, 1\}, \sum_{i,j} \mathbf{O}_{c,i,j} = 1, \end{aligned} \tag{5}$$

where $\mathbf{O}_{c,i,j}$ indicates the corresponding weight inside $\mathbb{P}$ for channel c and it becomes 1 when (i, j) is the most matched displacement otherwise 0. Our motion representation $\mathbf{H}_{t,c}$ in Eq. (2) could be interpreted as the residual features similar to $\mathbf{R}_{op}$. The difference is that the kernels $\mathbf{K}_{c,i,j}$ of CMM is learnable and supervised by the action classification loss, while the $\mathbf{O}_{c,i,j}$ of optical flow is fixed once Eq. (5) is solved by some optimization techniques. That is to say, our CMM could learn more task-relevant motion features than optical flows since the optimization target of our CMM is exactly the final classification task, while the Eq. (5) is to find the most matched displacement and this may not match the final target of the video action recognition task.

3.3 Illustration From Frequency Domain

From the visualization in Fig. 1, we find that the features from CSTM and CMM present similarities with the low-frequency and high-frequency components of the adjacent inputs, respectively. Therefore, we show an in-depth analysis of the proposed method from a different perspective, frequency domain. Mathematically, we derive the frequency spectrum based on the neighboring video features to explain these two modules explicitly. We apply the Discrete Fourier Transform (DFT) to transfer the extracted features from the temporal domain to the frequency domain. For a video feature $\mathbf{F}$, the expansion of DFT function is written as

$$\hat{\mathbf{F}}[n] = \sum_{k=0}^{K-1} \mathbf{F}[k] e^{-j\frac{2\pi}{K}nk}, n = 0, 1, 2, \ldots, K - 1, \tag{6}$$

where $\hat{\mathbf{F}}$ denotes the DFT of $\mathbf{F}$. When considering two adjacent input features $\mathbf{F}$=$\{\mathbf{F}_t, \mathbf{F}_{t+1}\}$, we have K=2. Then $n = 0$ presents the low frequency and $n = 1$ shows the high frequency. Formally

$$\begin{aligned} \hat{\mathbf{F}}[0] &= \mathbf{F}_t + \mathbf{F}_{t+1} \\ \hat{\mathbf{F}}[1] &= \mathbf{F}_t - \mathbf{F}_{t+1}. \end{aligned} \tag{7}$$

It shows that the sum of neighboring features $\mathbf{F}_t$ and $\mathbf{F}_{t+1}$ could represent the low-frequency information, while the difference of them reveals the high-frequency representation. As shown in Fig. 1, the low-frequency representation retains most scene information, while the high-frequency one presents the distinct motion edges. Interestingly, from the visualization, we find the features from our CSTM share common characters with the low-frequency temporal component, and the CMM shows the motion features like the high-frequency temporal component. In fact, our CSTM and CMM could be interpreted as advanced and learnable version of $\hat{\mathbf{F}}[0]$ and $\hat{\mathbf{F}}[1]$, respectively. More specifically, when we set the kernels of Eq. (1) to [0,1,1], CSTM degenerates into $\hat{\mathbf{F}}[0]$. Moreover, CMM becomes $\hat{\mathbf{F}}[1]$ if only the center elements of the weights of Eq. (2) are set to 1 while the others are zeroes. With learnable weights, the proposed CSTM and CMM could continuously learn to obtain better representations rather than directly apply Eq. (7), which is the raw information of different frequency bands. When the kernels are learnable parameters, the weights of CSTM and CMM are supervised and guided by the action classification loss, i.e., our target task, to achieve optimal performance. In contrast, the low-frequency and high-frequency components are special cases of fixed weights and are not adjusted by the target training loss. Therefore, our CSTM and CMM could perform better than the degenerated version. We also demonstrate this in our experiments (Section 5.3). Besides, as the features of CSTM and CMM correspond to low-frequency and high-frequency signals, respectively, it further proves why the two complement each other.

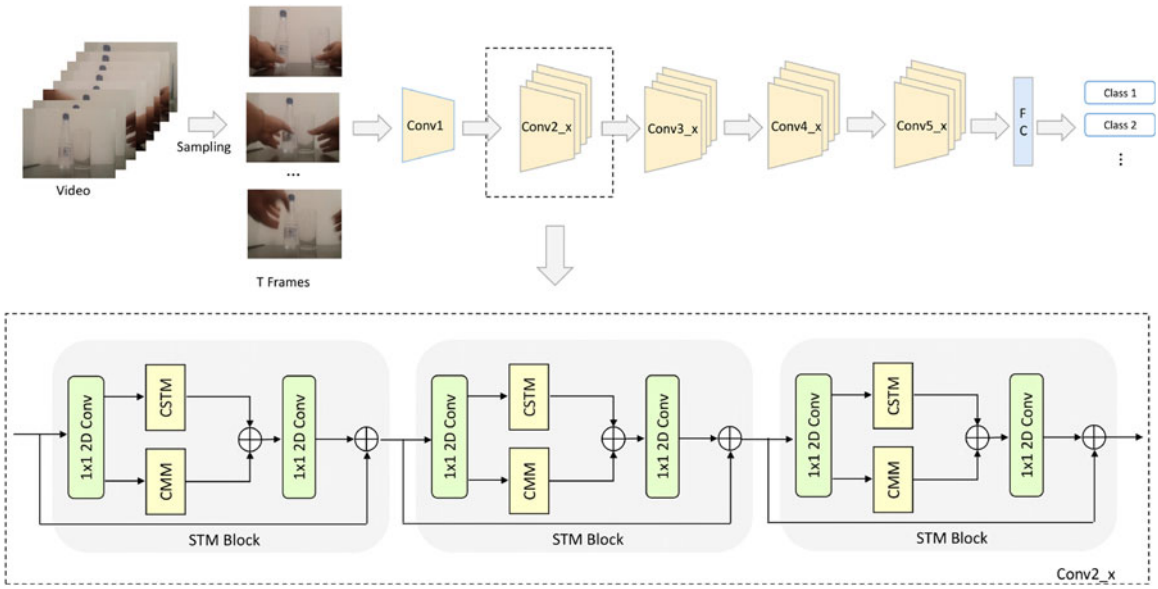

Fig. 3. The overall architecture of the STM network. The input video is first split into N segments equally and then one frame from each segment is sampled. We adopt 2D ResNet-50 as the backbone and replace all residual blocks with STM blocks. The last score fusion stage applies a temporal average pooling operation to reduce the temporal dimension.

3.4 STM Network

In order to keep the framework effective yet lightweight, we combine the proposed CSTM and CMM together to build a united block named STM block that can encode spatiotemporal and motion features together. It can be easily inserted into the existing ResNet architectures. The overall design of the STM block is illustrated in the bottom half of Fig. 3. In this STM block, a 1x1 2D convolution layer is responsible for reducing the channel dimensions first. The compressed feature maps are then passed through the CSTM and CMM to extract spatiotemporal and motion features, respectively. Typically, there are two ways to fuse different types of information: summation and concatenation. We experimentally found that summation works better than concatenation to fuse these two modules. Therefore, an element-wise sum operation is applied after the CSTM and CMM to aggregate the information. Then another 1x1 2D convolution layer is used to restore the channel dimensions. Similar to the ordinary residual block, we also add a parameter-free identity shortcut from the input to the output.

Since the proposed STM block is compatible with the ordinary residual block, we can simply insert it into any existing ResNet architectures to form our STM network. Compared to original ResNet (we consider the 50-layer ResNet-50 here), it brings very limited extra computation cost (1.2%, 32.9 G FLOPs *versus* 33.3 G FLOPs) and parameters (0.8%, 23.8 M *versus* 24 M). We illustrate the overall architecture of the STM network in the top half of Fig. 3. The STM network is a 2D convolutional network that encodes both spatiotemporal features and motion features together without any 3D convolution or pre-computing optical flow. Unless specified, we choose the 2D ResNet-50 [50] as our backbone for its trade-off between accuracy and speed. We replace all residual blocks with the proposed STM blocks. A temporal average pooling operation is applied in the last score fusion stage to reduce the temporal dimension.

3.5 Twins Training

Based on the proposed STM network, we further explore the training framework and propose a novel training framework aiming at further improving the performance, named Twins Training. It fully stretches the training samples and exploits the abundant power of the proposed STM. As shown in Fig. 4, Twins Training employs a siamese network to make the utmost of the training data by jointly encoding two distorted versions $\mathcal{X}^A$ and $\mathcal{X}^B$ of all video frames $\mathcal{X}$, which are sampled from a batch randomly selected videos $\mathcal{V}$. The distortions are generated from a distribution of group data augmentations $\mathcal{A}$. Next, the two batches of $\mathcal{X}^A$ and $\mathcal{X}^B$ are fed into the siamese network, which consists of two weights-shared (Twins) STM f_θ, conducting two batches of classification representations $\mathcal{Y}^A$ and $\mathcal{Y}^B$, where each feature dimension corresponds to one specific category.

We employ two losses in Twins Training to supervise the whole learning process. The first one is the common cross-entropy loss $\mathcal{L}_{cls}$ aiming at classification. $\mathcal{L}_{cls}$ is applied on both $\mathcal{Y}^A$ and $\mathcal{Y}^B$ and calculated with the target classification label $\mathcal{T}$. The second one is a label-free correlation loss $\mathcal{L}_{cor}$ inspired by [53] to decouple the inter-class correlation and reinforce the intra-class correlation. Unlike [53], we add the correlation loss as an auxiliary loss function into the supervised video representation learning framework under a supervised video classification setting and constraine it with the normal cross-entropy loss in the meantime. While [53] applies the correlation loss as the only loss function to the field of instance-aware self-supervised image representation learning for avoiding the collapse of trivial constant representations. Since the correlation loss is only used as an auxiliary function in our Twins Training framework, the feature dimension does not need to be set to a particularly large dimension like 8 k or 16 k in [53], such as Something-Something V1 has only 174 categories and Kinectis-400 has only 400 categories. However, the correlation

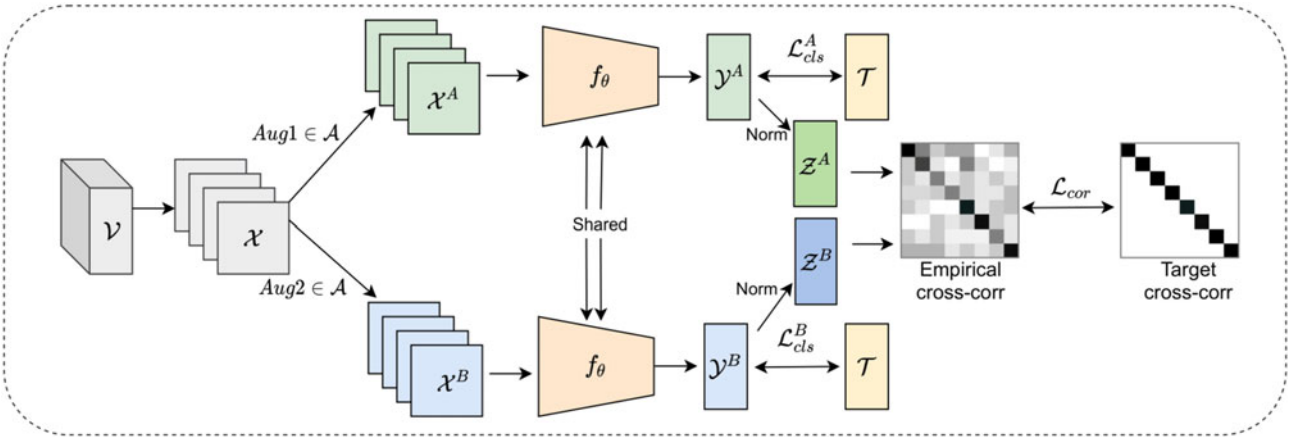

Fig. 4. Twins Training Framework. The input frames are first augmented into two distortions and then go through a shared siamese network to obtain their representations. Next, the two outputs are used to calculate the normal classification loss and also normalized to compute a correlation loss. The framework is only used in training to strengthen the network and will not influence the normal inference process.

loss can still effectively improve our accuracy. We first normalize $\mathcal{Y}^A$ and $\mathcal{Y}^B$ along the batch dimension with L_2 normalization, producing $\mathcal{Z}^A$ and $\mathcal{Z}^B$. Next, the cross-correlation matrix can be calculated between $\mathcal{Z}^A$ and $\mathcal{Z}^B$ along the batch dimension. Formally

$$\mathcal{C}_{ij} = \frac{\sum_n z^A_{n,i} z^B_{n,j}}{\sqrt{\sum_n \left(z^A_{n,i}\right)^2} \sqrt{\sum_n \left(z^B_{n,j}\right)^2}}, \tag{8}$$

where n indexes batch dimension, i, j are the class index. $\mathcal{C}$ is a square matrix with the dimensionality of the network's output (categories), and with values comprised between -1 (i.e., perfect anti-correlation) and 1 (i.e., perfect correlation). Then, $\mathcal{L}_{cor}$ can be formulated as

$$\mathcal{L}_{cor} = \underbrace{\sum_i (1 - \mathcal{C}_{ii})^2}_{intra-class\ term} + \beta \underbrace{\sum_i \sum_{j \neq i} \mathcal{C}^2_{ij}}_{inter-class\ term}, \tag{9}$$

$\mathcal{C}_{ii}$ actually shows the intra-class feature correlation inside ith class and $\mathcal{C}_{ij}$ presents the inter-class feature correlation between ith and jth classes. The first term of $\mathcal{L}_{cor}$ tries to make the diagonal elements of $\mathcal{C}$ to be 1, pulling features from the same category towards or even equal to each other. Thus it could shrink the intra-class distance and reinforce the correlation inside a class. The second term of $\mathcal{L}_{cor}$ tries to equate the off-diagonal elements of $\mathcal{C}$ to 0, pushing features of different categories further away and making them independent to each other, so it could reduce the inter-class redundancy and decouple the correlation among different categories. This is quite different to the similar works of the correlation loss applications like [53], which applies the correlation loss function to an extra three-layer MLP Predictor and uses only the backbone before the Predictor as feature encoding network, i.e., the correlation loss function does not directly affect their final used features. In the Twins Training framework of our STM, the correlation loss function is directly applied to the final classification features, so that it could achieve our goal of reducing the correlation of different categories and aggregating the correlation of the same classes.

The final objective of Twins Training is

$$\mathcal{L} = \frac{1}{2}\left(\mathcal{L}^A_{cls} + \mathcal{L}^B_{cls}\right) + \lambda \mathcal{L}_{cor}, \tag{10}$$

where the parameter λ is used to balance the two kind of losses. $\mathcal{L}^A_{cls}$ and $\mathcal{L}^B_{cls}$ are the cross entropy losses of the two distorted batches.

Besides, it is worth to notice that Twins Training is a training framework used in the training phase and it will bring no changes like extra parameters or computational burdens in the inference process.

4 Experiments

In this section, we first introduce the datasets and the implementation details of our approach. Then we perform extensive experiments to demonstrate that the proposed STM consistently outperforms or obtains a comparable performance with the state-of-the-art methods on both temporal-related datasets (i.e., Something-Something v1 & v2) and scene-related datasets (i.e., Kinetics-400, UCF-101, and HMDB-51). The baseline method in our experiments is Temporal Segment Networks (TSN) [14] where we re-implement it by replacing its backbone to ResNet-50 for fair comparisons. Finally, we give runtime analyses to show the efficiency of STM.

4.1 Datasets

We evaluate the performance of the proposed STM on several public action recognition datasets. We classify these datasets into two categories: (1) temporal-related datasets, including Something-Something v1 & v2 [20]. For these datasets, temporal motion interaction of objects is the key to action understanding. Most of the actions cannot be recognized without considering the temporal relationship; (2) scene-related datasets, including Kinetics-400 [2], UCF-101 [21] and HMDB-51 [22] where the background information contributes a lot for determining the action label in most of the videos. Temporal cues in scene-related datasets are not as important as temporal-related datasets. We give examples in Fig. 5 to show the difference between them. Since our method is designed for effective spatiotemporal

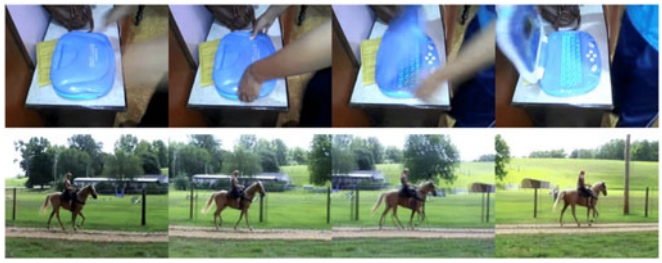

Fig. 5. Examples of temporal-related datasets and scene-related datasets. Top: action for which temporal feature matters. Reversing the order of frames gives the opposite label (opening something versus closing something). Bottom: action for which scene feature matters. Even with only one frame, we can easily predict its label (horse riding).

fusion and motion information extraction, we obtain a large margin performance gain than our baseline TSN in the temporal-related datasets. Nevertheless, for those scene-related datasets, our method also achieves competitive results.

4.2 Implementation Details

Network Details. Given an input video, we first divide it into T segments of equal durations to conduct long-range temporal structure modeling. Then, we randomly sample one frame from each segment to construct the input sequence with T frames. In our experiments, T is set to 8 or 16. r in CMM is set to 16.

Training Details. We train our model with 8 Geforce RTX 3090 GPUs and each GPU processes a mini-batch of 8 video clips (when $T = 8$) or 4 video clips (when $T = 16$). For Kinetics, Something-Something v1 & v2, we start with a learning rate of 0.01 and reduce it by a factor of 10 at 30,40,60 epochs and stop at 70 epochs. For these large-scale datasets, we only use the ImageNet pre-trained model as initialization. For UCF-101 and HMDB-51, we use Kinetics pre-trained model as initialization and start training with a learning rate of 0.001 for 50 epochs. The learning rate is decayed by a factor of 10 every 15 epochs. We use mini-batch SGD as the optimizer with a momentum of 0.9 and a weight decay of 5e-4. The size of the short side of input frames is fixed to 256. Then, we apply augmentation on them and resize the cropped regions to 224×224 for network training. Therefore, the input size of the network is $N \times T \times 3 \times 224 \times 224$, where N is the batch size and T is the number of the sampled frames per video. For the Twins Training framework, β is set to 0.0039 as [53] and λ is set to 0.01. The data argumentation $\mathcal{A}$ consists of corner cropping, scale-jittering, horizontal flipping, color jittering and grayscale.

Inference Details. Following [5], [57], we first scale the shorter spatial side to 256 pixels and take three crops of 256×256 to cover the spatial dimensions and then resize them to 224×224. We randomly sample 10 times from the full-length video for the temporal domain and compute the softmax scores individually. The final prediction is the averaged softmax scores of all clips.

4.3 Results on Temporal-Related Datasets

This section compares our approach with state-of-the-art methods on temporal-related datasets including Something-Something v1 & v2. Something-Something v1 is a large collection of densely labeled video clips that show fundamental human interactions with daily objects. This dataset contains 174 classes with 108,499 videos. Something-Something v2 is an updated version of v1 with more videos (220,847 in total) and greatly reduced label noise.

Table 1 lists the results of our method compared with the state-of-the-art on Something-Something v1 and v2. The results of the baseline method TSN are relatively low compared with other methods, which demonstrates the importance of temporal modeling for these temporal-related datasets. Compared with TSN, our STM network gains 30.7% and 33.4% top-1 accuracy improvement with 8 and 16 frames inputs respectively on Something-Something v1. On Something-Something v2, 8-frame and 16-frame STM also gain 35% and 34.9% improvement compared to TSN, respectively. The methods can be classified into two types as shown in the two parts of Table 1. The upper part presents the 3D CNN-based methods, including S3D-G [8], ECO [34], I3D models [58] 3D ShuffleNet and 3D MobileNet [54]. The lower part is 2D CNN-based methods, including TSN, TRN [55], MFNet [18] TSM [10], TEA [11], STFT [37] and TPN [56]. Even our STM with 8 RGB frames as input surpasses all the 3D CNN-based methods in the upper part, which usually take more frames or optical flow as input. With 16 frames as input, STM achieves the best performance in Something-Something v1. In the Something-Something v2, our results are just slightly lower than the TSM two-stream with a 1.1% top-1 accuracy gap and the STFT with a 0.1% top-5 accuracy gap. However, TSM two-stream relies on optical flow to their best results and its 16 RGB frames version performs worse than our 16-frame STM. STFT gains the best top-5 accuracy with much more frames (64 frames) than ours (16 frames), while our STM surpasses STFT in the top-1 accuracy (64.9% *versus* 64.7%).

4.4 Results on Scene-Related Datasets

We evaluate our STM on three scene-related datasets: Kinetics-400, UCF-101, and HMDB-51. Kinetics-400 is a large-scale human action video dataset with 400 classes. It contains 236,763 clips for training and 19,095 clips for validation. UCF-101 is a relatively small dataset that contains 101 categories and 13,320 clips in total. HMDB-51 is also a small video dataset with 51 classes and 6766 labeled video clips. For UCF-101 and HMDB-51, we followed [14] to adopt the three training/testing splits for evaluation.

Table 2 summarizes the results of STM and other competing methods on the Kinetics-400 dataset. From the evaluation, we can draw the following conclusions: (1) Different from the previous temporal-related datasets, most actions of Kinetics can be recognized by scenes and objects even with one still frame of videos. Therefore the baseline method without any temporal modeling can achieve acceptable accuracy; (2) Though our method mainly focuses on temporal-related actions recognition, STM still achieves very competitive results compared with the state-of-the-art methods. STM outperforms all the 2D CNNs-based methods in the lower part of Table 2, including the most similar work TEA [11] (76.9% *versus* 76.1%). STM even exceeds several 3D CNN-based methods (the upper part of the Table 2) like STC, ARTNet, S3D and ECO. Top-1 accuracy of our 16-frame STM is 0.4% higher than the 32-frame NL (Non-local [57]) I3D, which uses 3D ResNet-50 as the backbone.

TABLE 1
Performance of the STM on the Something-Something v1 and v2 Datasets Compared With the State-of-The-Art Methods. We Report the Inference Cost with a Single "View" (a Temporal Clip with a Spatial Crop) × the Number of Such Views Used (FLOPs × Views). '-' Indicates That the Values are not Available to us.

Method		Backbone	Flow	Pre-training	Frames	FLOPs × views	Something-Something v1		Something-Something v2	
							Top-1	Top-5	Top-1	Top-5
3D CNNs	S3D-G [8]	Inception		ImageNet	64	71.38 G×1	48.2	78.7	-	-
	ECO [34]			Kinetics	8	32 G×1	39.6	-	-	-
	ECO [34]	BNInception+			16	64 G×1	41.4	-	-	-
	$ECO_{EN}Lite$ [34]	3D ResNet-18			92	267 G×1	46.4	-	-	-
	$ECO_{EN}Lite$ Two-Stream [34]		✓		92+92	-	49.5	-	-	-
	ShuffleNetV1 2x [54]	3D ShuffleNetV1		Scratch	32	0.78 G×1	33.9	62.5	-	-
	ShuffleNetV2 2x [54]	3D ShuffleNetV2			32	0.72 G×1	31.9	61.0	-	-
	MobileNetV1 2x [54]	3D MobileNetV1			32	0.92 G×1	29.8	56.9	-	-
	MobileNetV2 1x [54]	3D MobileNetV2			32	0.91 G×1	30.8	59.8	-	-
	I3D [2]	3D ResNet-50		ImageNet	32	153 G×6	41.6	72.2	-	-
	I3D+GCN [2]			+Kinetics	32	303 G×6	43.4	75.1	-	-
2D CNNs	TSN [14]	ResNet-50		Kinetics	8	16 G×1	19.7	46.6	27.8	57.6
					16	33 G×1	19.9	47.3	30.0	60.5
	TRN Multiscale [55]	BNInception		ImageNet	8	16.37 G×1	34.4	-	48.8	77.6
	TRN Two-Stream [55]		✓		8+8	-	42.0	-	55.5	83.1
	MFNet-C101 [18]	ResNet-101		Scratch	10	-	43.9	73.1	-	-
	TSM [10]	ResNet-50		ImageNet	8	32.9 G×1	45.6	74.2	59.1	85.6
	TSM [10]			+	16	65.8 G×1	47.2	77.1	63.4	88.5
	TSM Two-Stream [10]		✓	Kinetics	16+16	-	52.6	**81.9**	**66.0**	90.5
	TEA [11]	ResNet-50		ImageNet	8	35 G×30	51.7	80.5	-	-
					16	70 G×30	52.3	**81.9**	-	-
	STFT [37]	BN-Inception		Kinetics	64	41.21 G×1	52.4	81.8	64.7	**90.8**
	TPN [56]+TSN	ResNet-50		ImageNet	8	-	40.6	-	55.2	-
	TPN [56]+TSM				8	-	49.0	-	62.0	-
	STM	ResNet-50		ImageNet	8	33.3 G×30	50.4	80.6	62.8	89.1
					16	66.5 G×30	**53.3**	**81.9**	64.9	90.7

NL SlowFast [5] yields the state-of-the-art accuracy of 79.8% on this dataset. However, they have a much higher computational cost than STM (234 G×30 *versus* 66.5 G×30). Besides, our 16-frame STM achieves comparable top-1 accuracy against SlowFast with 3D ResNet-50 as backbone and no non-local operation (76.9% *versus* 77.0%). One may notice that X3D [6] achieves a slightly better results than our STM (77.5% *versus* 75.5%) with a bit lower inference cost than ours. However, the training process of X3D is resource-consuming which needs 128 GPUs and trains with 256 epochs. Although its inference cost is slightly lower than ours, the training cost is much more than ours.

We also conduct experiments on the UCF-101 and HMDB-51 to study the generalization ability of our STM. We evaluate our method over three splits and report the averaged results in Table 3. First, compared with the ImageNet pre-trained model, Kinetics pre-train can significantly improve the performance on small datasets. Then, compared with the state-of-the-art methods, STM with 16 frames as inputs outperforms all the methods, including the 3D CNN-based and 2D CNN-based approaches in Table 3. It demonstrates that our STM achieves superior generalization ability on small datasets.

4.5 Runtime Analysis

Our STM achieves the new state-of-the-art or comparable results on several benchmark datasets compared with other methods. More importantly, it is a unified 2D CNN framework without any time-consuming 3D convolution and optical flow calculations. Table 4 shows the accuracy and model complexity of STM and several state-of-the-art methods on the Something-Something v1 dataset. All these tests are conducted on a single Geforce RTX 3090 GPU. For a fair comparison, we evaluate our method by evenly sampling 8 or 16 frames from a video and then applying the center crop. STM-18, STM-34, STM-50 indicate 18-layers, 34-layers and 50-layers versions of STM, which are similar to ResNet-18, ResNet-34 and ResNet-50. TSN_{8F} uses the ordinary ResNet-50 as the backbone, thus it could be regarded as our baseline method. Our STM-50_{8F} brings very limited extra computation cost (1.2%, 32.9 G FLOPs *versus* 33.3 G FLOPs) and parameters (0.8%, 24 M *versus* 23.8 M) against TSN_{8F}, while the performance is improved with a large margin (48.5% *versus* 19.7%). Compared to $I3D_{64F}$ and ECO_{16F}, our STM-50_{8F} achieves 9.2x and 1.9x fewer FLOPs (33.3 G *versus* 306 G, 64 G), 9.7x and 1.3x faster (106.7 V/s *versus* 11.0 V/s, 79.7 V/s), while 6.9% and 7.1% higher top-1 accuracy. Compared to TSM_{16F}, our STM-50_{8F} gains 1.3% higher accuracy with 1.7x faster speed and half FLOPs. As for TEA_{8F} [11], although the accuracy of our STM-50_{8F} is slightly lower than it (48.5% *versus* 48.9%), STM-50_{8F} runs 1.79x faster than TEA (106.7 V/s *versus* 59.5 V/s). Besides, our smaller variants also have stunning performances with faster inference speed and fewer parameters and FLOPs. STM-18_{8F} runs 161.6 videos a second, which is 14.7x faster than $I3D_{64F}$

TABLE 2
Performance of the STM on the Kinetics-400 Dataset Compared With the State-of-The-Art Methods. We Report the Inference Cost with a Single "View" (a Temporal Clip with a Spatial Crop) $\times$ the Number of Such Views Used (FLOPs $\times$ Views). '-' Indicates that the Values are not Available to us.

Method		Backbone	Flow	Frames	FLOPs $\times$ views	Top-1	Top-5
3D CNNs	ECO-RGB$_{en}$ [34]	BNInception +3D ResNet-18		92	267 G$\times$1	70.0	-
	I3D RGB [2]	3D BN-Inception		64	107.89 G$\times$-	72.1	90.3
	I3D Two-Stream [2]		✓	64+64	216 G$\times$-	75.7	92.0
	NL I3D [57]	3D ResNet-50		32	282 G$\times$30	76.5	92.6
	NL I3D [57]	3D ResNet-101		32	359 G$\times$30	77.7	93.3
	STC [30]	3D ResNext101		32	-	68.7	88.5
	ARTNet [59]	3D ResNet-18		32	23.7 G$\times$250	69.2	88.3
	S3D [8]	BN-Inception		64	66.38 G$\times$-	72.2	90.6
	S3D-G [8]			64	71.38 G$\times$-	74.7	93.4
	SlowFast 8x8 [5]	3D ResNet-50		8+8	65.7 G$\times$30	77.0	92.6
	NL SlowFast 8x8 [5]	3D ResNet-101		8+8	116 G$\times$30	78.7	93.5
	NL SlowFast 16x8 [5]			16+8	234 G$\times$30	**79.8**	**93.9**
	X3D-L [6]	3D ResNet-based		16	24.8 G$\times$30	77.5	92.9
	X3D-XL [6]			16	48.4 G$\times$30	79.1	93.9
	TPN [56]	3D ResNet-101		64	-	78.9	93.9
2D CNNs	StNet [60]	ResNet-101		25	310.5 G$\times$-	71.4	-
	Disentangling [45]	BNInception		-	-	71.5	89.9
	R(2+1)D RGB [7]	ResNet-34		32	152 G$\times$115	72.0	90.0
	R(2+1)D Two-Stream [7]		✓	32+32	304 G$\times$115	73.9	90.9
	TSM [10]	ResNet-50		8	65.8 G$\times$10	74.1	91.2
		ResNet-50		16	65.8 G$\times$10	74.7	-
	TSN RGB [14]	BNInception		25	53 G$\times$10	69.1	88.7
	TSN Two-Stream [14]		✓	-	80 G$\times$10	73.9	91.1
	TEA [11]	ResNet-50		8	35 G$\times$30	75.0	91.8
				16	70 G$\times$30	76.1	92.5
	STFT [37]	BN-Inception		64	41.21 G$\times$30	75.0	91.1
	TPN [56]+TSN	ResNet-50		8	-	73.5	-
	STM	ResNet-50		8	33.3 G$\times$30	75.5	92.0
				16	66.5 G$\times$30	76.9	92.7

and 2.0x faster than ECO$_{16F}$ while achieving comparable performance. Such smaller variants will be more suitable for many practical platforms with faster speed and fewer parameters than large models.

4.6 Online Recognition

We have implemented a plain online version of STM for the online application and demonstrated its performance on Kinetics-400 and Something-Something V1. In detail, for an online video, we keep a memory cache to store the historical seven frames. When recognizing a specific frame, we combine it with the stored frames to obtain its recognition result. We use the prediction averaged from all the frames to compare with the offline models. All the STM-online models are tested on a single Geforce RTX 3090 GPU and use 8 RGB frames as inputs.

As shown in Table 5, the STM-online could run in real time with low latency and high accuracy, maintaining similar performance as the offline model. It could be seen that for scene-related datasets like Kinetics-400, the online model achieves comparable performance (75.4% *versus* 75.5%). While for temporal-related datasets like Something-Something V1, online model performs slightly worse than offline model (49.7% *versus* 50.4%). This is intuitive because temporal-related datasets require more temporal cues for better recognition while the scene-related datasets are less dependent on temporal information. Meanwhile, we compare the performance of the STM-online with TSM-online [10]. Although our latency is slightly larger than theirs (5.4ms% *versus* 4.8 ms) since TSM-online directly saves historical features for their shifting operation while we need to do the feature extraction process, the latency of our STM-online and TSM-online are actually all beyond the real-time requirement. Moreover, our STM-online surpasses TSM-online with large gains of 1.1% on Kinetics-400 and 3.4% on Something-Something V1.

5 Ablation Study

In this section, we comprehensively perform ablation studies to analyze each component in our proposed STM on the Something-Something v1 dataset. The proposed Twins Training framework is added and validated in the last part. Unless specified, the ablation experiments in this section use 8 RGB frames as inputs.

5.1 Impact of Two Modules

Our proposed two modules can be inserted into a standard ResNet architecture independently. To validate the contributions of each module in the STM (i.e., CSTM and CMM), we compare the results of the individual module and the combination of both modules in Table 6. We can summarize that each component contributes to the proposed STM

TABLE 3
Performance of the STM on UCF-101 and HMDB-51 Compared With the State-of-The-Art Methods

Method		Backbone	Flow	Pre-training	UCF-101	HMDB-51
3D CNNs	C3D [1]	3D VGG-11		Sports-1 M	82.3	51.6
	STC [30]	3D ResNet101		Kinetics	93.7	66.8
	ARTNet with TSN [59]	3D ResNet-18		Kinetics	94.3	70.9
	ECO [34]	BNInception+3D ResNet-18		Kinetics	94.8	72.4
	I3D RGB [2]	3D Inception-v1		ImageNet+Kinetics	95.1	74.3
2D CNNs	TSN [14]	ResNet-50		ImageNet	86.2	54.7
	TSN RGB [14]	BNInception		ImageNet+Kinetics	91.1	-
	TSN two-Stream [14]		✓	ImageNet+Kinetics	97.0	-
	TSN two-Stream [12]		✓	ImageNet	94.9	71.0
	Four-Stream+IDT [61]	ResNeXt-50	✓	ImageNet	95.4	74.2
		ResNeXt-101	✓	ImageNet	96.0	74.9
	LTC [62]	C3D		-	82.4	-
	LTC two-stream+IDT [62]		✓		92.7	67.2
	TSM [10]	ResNet-50		ImageNet+Kinetics	95.9	73.5
	StNet [60]	ResNet50		ImageNet+Kinetics	93.5	-
	Disentangling [45]	BNInception		ImageNet+Kinetics	95.9	-
	TEA [11]	ResNet-50		ImageNet+Kinetics	96.9	73.3
	STFT [37]	BN-Inception		Kinetics	94.7	71.5
	STM	ResNet-50		ImageNet+Kinetics	**97.1**	**75.2**

block. CSTM learns channel-wise temporal fusion and brings about 28% top-1 accuracy improvement compared to the baseline method TSN while CMM encodes the feature-level motion information and brings 24.4% top-1 accuracy improvement. When combining CSTM and CMM together, we can learn both spatiotemporal and motion features and achieve the best top-1 accuracy, especially, the gain over the baseline is 29.5%.

5.2 Fusion of Two Modules

There are two ways to combine CSTM and CMM: element-wise summation and concatenation. The element-wise summation is parameter-free and easy to implement. For concatenation fusion, we first concatenate outputs of CSTM and CMM over the channel dimension, and the dimension of concatenating features is $2\,C$. Then a 1x1 convolution is applied to reduce the channels to C. We conduct experiments to study the two fusion ways as shown in Table 7, though summation aggregation is simple, it still outperforms concatenation by 7.4% at top-1 accuracy and 6.1% at top-5 accuracy.

TABLE 4
Accuracy and Model Complexity of STM and Other State-of-The-Art Methods on the Something-Something V1 Dataset With a Single Crop. Measured on a Single Geforce RTX 3090 GPU.

Model	Frame	FLOPs	Param	Speed	Acc.
I3D [2]	64	306 G	28.0 M	11.0 V/s	41.6
ECO [34]	16	64 G	47.5 M	79.7 V/s	41.4
TSM [10]	8	32.9 G	24.3 M	120.3 V/s	45.6
	16	65.8 G		63.5 V/s	47.2
TEA [11]	8	35 G	-	59.5 V/s	48.9
TSN [14]	8	32.9 G	23.8 M	121.2 V/s	19.7
STM-18	8	14.6 G	11.0 M	**161.6** V/s	40.5
STM-34	8	29.4 G	20.5 M	155.3 V /s	43.8
STM-50	8	33.3 G	24.0 M	106.7 V/s	48.5
	16	66.5 G		52.5 V/s	**51.6**

5.3 Frequency Degeneration

As illustrated in Section 3.3, the proposed CSTM and CMM can degenerate into low-frequency and high-frequency representations, respectively. We implement a reduced version by setting the kernels in Eq. (1) to [0,1,1] and fixing the center elements of the weights of Eq. (2) to 1 while the others are zeroes. The results are shown in Table 8. Interestingly, the frequency degeneration obtains a 25.5% top-1 accuracy improvement over TSN. It means that even simply representing high-frequency $\hat{\mathbf{F}}_t[1]$ and low-frequency $\hat{\mathbf{F}}_t[0]$ components like Eq. (7) inside every residual block could contribute a lot to the performance. However, when making the kernels in Eq. (1) and Eq. (2) learnable and trainable, i.e., using STM will surpass directly applying the raw frequency representation with by 4% and 5.2% on top-1 and top-5 accuracy, respectively. We visualize an example to compare the frequency degeneration variant and our STM in Fig. 6. It could be found that the learnable CSTM block can pay more attention to the spatiotemporal characteristics of the action than the fixed CSTM block. For example, it can clearly show the hand holding the bottle while the fixed CSTM block focuses more on the background behind the bottle. Compared with the fixed CMM block, the learnable CMM block can pay more attention to the motion edges of action subject and reduce the noise significantly. For instance, the learnable CMM block can present the motion of the fingers, while the fixed CMM block brings lots of noise on the bottle.

TABLE 5
Online Performance on the Kinetics-400 and Something-Something V1 Datasets

Model	Latency	Kinetics	Something
TSM-Online	4.8 ms	74.3	46.3
STM-Offline	-	75.5	50.4
STM-Online	5.4 ms	75.4	49.7

TABLE 6
Impact of Two Modules: Comparison Between CSTM, CMM and STM

Model	Top-1	Top-5
TSN	19.7	46.6
CSTM	47.7	77.9
CMM	44.1	74.8
STM	49.2	79.3

TABLE 7
Fusion of Two Modules: Summation Fusion is Better

Aggregation	Top-1	Top-5
TSN	19.7	46.6
Summation	49.2	79.3
Concatenation	41.8	73.2

TABLE 8
Frequency Degeneration: Frequency Degeneration Could Achieve a Satisfactory Performance, but Not as Good as Our STM

Type	Top-1	Top-5
TSN	19.7	46.6
frequency degeneration	45.2	74.1
STM	49.2	79.3

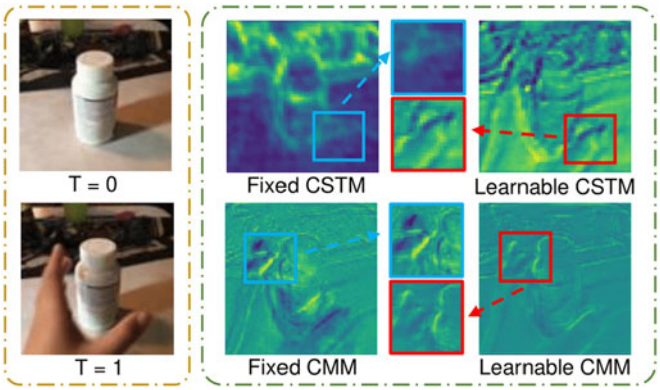

Fig. 6 Feature Visualization of the Frequency Degeneration and our STM. The first column represents the original images of the two moments t=0 and t=1 of an action "holding something". The second column shows the output features of Frequency Degeneration, i.e., fixed CSTM and CMM blocks, which are the temporal low-frequency and high-frequency components of the neighboring input features. The third column presents the output features of our STM, i.e., the learnable CMM and CSTM blocks. Brighter colors indicate larger values in the feature maps.

5.4 Location and Number of STM Block

ResNet-50 architecture can be divided into six stages. We refer the conv2_x to conv5_x as stage 2 to stage 5. The first four rows of Table 9 compare the performance of replacing only the first residual block with STM block on different stages in ResNet-50, from stage 2 to stage 5, respectively. We conclude from the results that replacing only one residual block already yields significant performance improvement compared to the baseline TSN, which demonstrates the effectiveness of the proposed STM block. In detail, we can find that replacing STM block at each stage all promotes the performance, which validates that the proposed temporal modeling strategy could benefit both low-level and high-level temporal features learning. We then replace one block for each stage (i.e., replacing four blocks in all) and obtain better results. Our model achieves the best performance when replacing all original residual blocks with STM blocks (i.e., 16 blocks in all).

TABLE 9
Location and Number of STM Block: Deeper Location and More Blocks Yeild Better Performance

Stage	STM Blocks	Top-1	Top-5
2	1	38.7	70.1
3	1	40.6	71.6
4	1	41.5	72.6
5	1	41.5	71.8
2-5	4	47.9	78.1
2-5	16	49.2	79.3

TABLE 10
Type of Convolution: For CSTM, Channel-Wise Temporal Convolution Yields Better Performance. For CMM, Channel-Wise Spatial Convolution Obtains Better Performance.

Type		Top-1	Params	FLOPs
CSTM	Channel-wise 1D Conv	47.7	23.88 M	32.93 G
	Channel-wise 1D, 2D Conv	42.6	13.28 M	24.05 G
	All Ordinary	46.9	27.64 M	40.59 G
CMM	Channel-wise	44.1	23.95 M	32.95 G
	Ordinary	43.5	24.25 M	33.5 G

TABLE 11
Impact of Twins Training Framework: All the Methods Improve Their Performance in Both Kinetics-400 and Something-Something v1 With the Help of the Twins Training Framework

Model	Twin Training	Kinetics-400@Top-1	Something v1@Top-1
TSN		70.6	19.7
	✓	72.5	20.9
TSM		74.1	47.3
	✓	75.2	48.2
STM		73.7	49.2
	✓	75.5	50.4

5.5 Type of Convolution

We choose channel-wise temporal convolution in CSTM to learn temporal combination individually for each channel and channel-wise spatial convolution in CMM to encode motion weights separately for each channel. We make

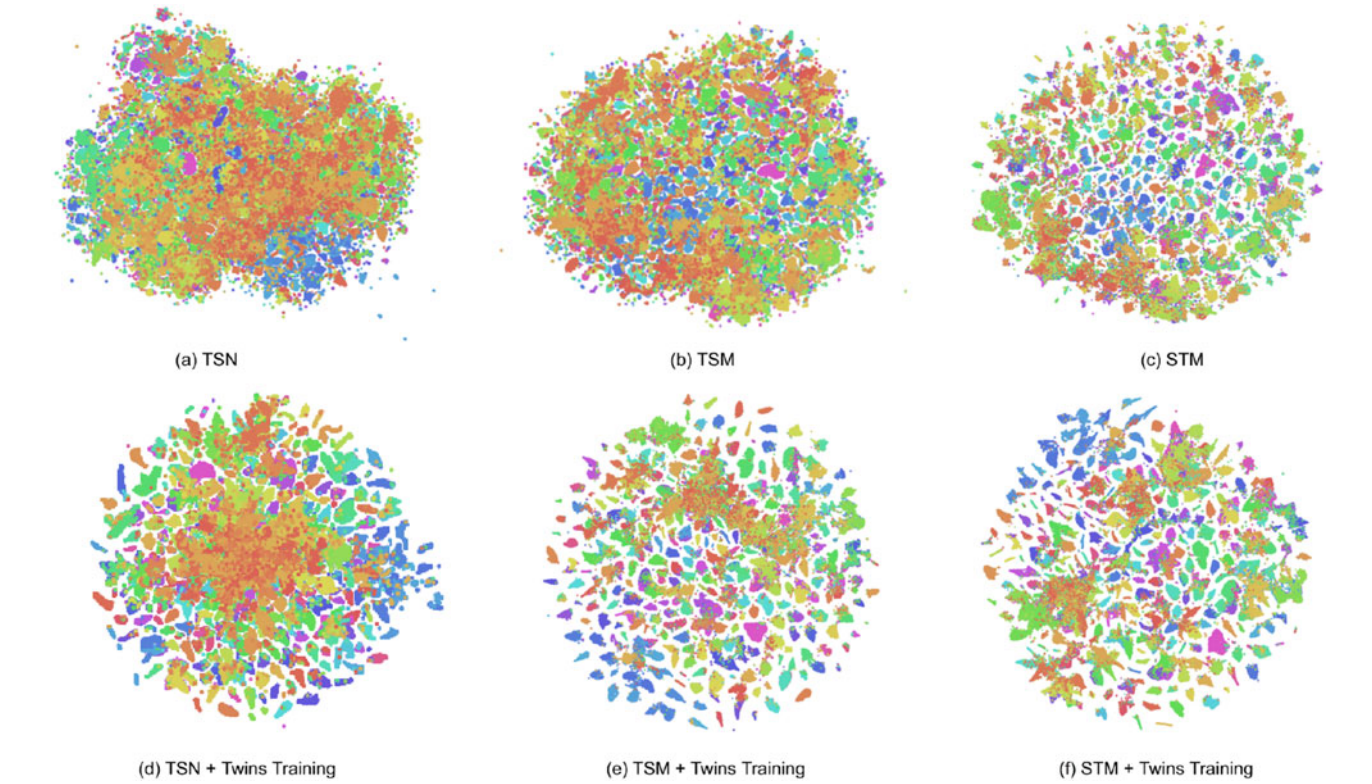

Fig. 7. Feature visualization with t-SNE [63] on Kinetics-400. Each video is visualized as a point. Videos belonging to the same action category have the same color. Without Twins Training (first row), our STM learns semantically more separable features than TSN [14] and TSM [10]. When equipping with the Twins Training framework (second row), all the three methods' inter-class and intra-class distances are dramatically optimized.

comparisons with ordinary convolution in two modules and the results are shown in Table 10. With channel-wise convolutions, we can achieve better performance with few parameters and FLOPs in both two modules, which confirms that channel-wise convolutions for temporal combination and motion modeling are both sufficient and efficient. Moreover, we experiment with channel-wise spatial convolutions in CSTM module as shown in Table 10. The performance drops a lot (5.3%) on top-1 accuracy, indicating that it needs more parameters to learn the semantic spatial features. Thus we adopt ordinary convolution for spatial feature encoding in our CSTM module.

5.6 Impact of Twins Training Framework

We demonstrate the validness of the proposed Twins Training framework on our STM and other two representative methods, TSN and TSM (only 8 RGB frames as inputs). Besides, we also conduct experiments on a larger scene-related benchmark Kinetics-400 to show the generalization ability of Twins Training. As shown in Table. 11, when equipping with Twins Training framework, all the three methods perform superior against original training manner on both two datasets. As visualized in Fig. 7, by projecting the features of the last classification layer into a low 2-dimension space, it is obvious that (1) without Twins Training, our STM learns semantically more separable features than TSN [14] and TSM [10]. Since our STM integrates spatiotemporal and motion features together, while TSN only encodes spatial features and TSM only represents spatiotemporal features; (2) the inter-class distances of all three methods are enlarged and all the intra-class distances are shrunk by employing Twins Training. This success comes from the correlation loss, which could reinforce the correlation inside the same classes and reduce the redundancy between different classes.

6 Conclusion

This paper presents a simple yet effective network for action recognition by encoding spatiotemporal and motion features together in a unified 2D CNN network. We replace the original residual blocks with STM blocks in ResNet architecture to build the STM network. An STM block contains a channel-wise spatiotemporal module to model the spatiotemporal features and a channel-wise motion module to learn motion representations. Moreover, we provide an in-depth illustration of these two modules from the frequency domain and find they could be interpreted as advanced and learnable versions of frequency components. Furthermore, a novel Twins Training framework is proposed to fully exploit the training data and enhance our model by decoupling the inter-class correlation and reinforcing the intra-class correlation. Without any 3D convolution or pre-calculation optical flows, our STM receives state-of-the-art or comparable results on both temporal-related datasets and scene-related datasets. We believe the architectures and ideas discussed in this paper are successful in spatiotemporal and motion features modeling. It could be extended for other video tasks such as video detection and segmentation. We leave this as our future work.

ACKNOWLEDGMENTS

Mengmengwang and Jiazheng Xing are equal contributors.

REFERENCES

[1] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 4489–4497.
[2] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4724–4733.
[3] J. C. Stroud, D. A. Ross, C. Sun, J. Deng, and R. Sukthankar, "D3D: Distilled 3D networks for video action recognition," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2020, pp. 614–623.
[4] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng, "A^2-Nets: Double attention networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 350–359.
[5] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "SlowFast networks for video recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6201–6210.
[6] C. Feichtenhofer, "X3D: Expanding architectures for efficient video recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 200–210.
[7] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6450–6459.
[8] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, "Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 305–321.
[9] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3D residual networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5534–5542.
[10] J. Lin, C. Gan, and S. Han, "TSM: Temporal shift module for efficient video understanding," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 7082–7092.
[11] Y. Li, B. Ji, X. Shi, J. Zhang, B. Kang, and L. Wang, "Tea: Temporal excitation and aggregation for action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 906–915.
[12] L. Wang *et al.*, "Temporal segment networks for action recognition in videos," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 11, pp. 2740–2755, Nov. 2019.
[13] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1933–1941.
[14] L. Wang *et al.*, "Temporal segment networks: Towards good practices for deep action recognition," in *Proc. Eur. Conf. Comp. Vis.*, 2016, pp. 20–36.
[15] Y. Wang, M. Long, J. Wang, and P. S. Yu, "Spatiotemporal pyramid network for video action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2097–2106.
[16] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Spatiotemporal multiplier networks for video action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7445–7454.
[17] S. Sun, Z. Kuang, L. Sheng, W. Ouyang, and W. Zhang, "Optical flow guided feature: A fast and robust motion representation for video action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1390–1399.
[18] M. Lee, S. Lee, S. Son, G. Park, and N. Kwak, "Motion feature network: Fixed motion filter for action recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 387–403.
[19] C.-Y. Wu, R. Girshick, K. He, C. Feichtenhofer, and P. Krahenbuhl, "A multigrid method for efficiently training video models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 150–159.
[20] R. Goyal *et al.*, "The "something something" video database for learning and evaluating visual common sense," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5843–5851.
[21] 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*.
[22] 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.*, 2011, pp. 2556–2563.
[23] B. Jiang, M. Wang, W. Gan, W. Wu, and J. Yan, "STM: Spatiotemporal and motion encoding for action recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 2000–2009.
[24] H. Wang, A. Kläser, C. Schmid, and L. Cheng-Lin, "Action recognition by dense trajectories," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 3169–3176.
[25] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 3551–3558.
[26] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1725–1732.
[27] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 568–576.
[28] C. Feichtenhofer, A. Pinz, and R. Wildes, "Spatiotemporal residual networks for video action recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3468–3476.
[29] Y. Zhu *et al.*, "A comprehensive study of deep video action recognition," 2020, *arXiv:2012.06567*.
[30] A. Diba *et al.*, "Spatio-temporal channel correlation networks for action classification," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 284–299.
[31] H. Wang, D. Tran, L. Torresani, and M. Feiszli, "Video modeling with correlation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 349–358.
[32] 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.
[33] G. Bertasius, H. Wang, and L. Torresani, "Is space-time attention all you need for video understanding?" 2021, *arXiv:2102.05095*.
[34] M. Zolfaghari, K. Singh, and T. Brox, "ECO: Efficient convolutional network for online video understanding," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 695–712.
[35] Y. Zhou, X. Sun, Z.-J. Zha, and W. Zeng, "MiCT: Mixed 3D/2D convolutional tube for human action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 449–458.
[36] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi, "Human action recognition using factorized spatio-temporal convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 4597–4605.
[37] S. Kumawat, M. Verma, Y. Nakashima, and S. Raman, "Depthwise spatio-temporal STFT convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Apr. 29, 2021, doi: 10.1109/TPAMI.2021.3076522.
[38] D. Tran, H. Wang, L. Torresani, and M. Feiszli, "Video classification with channel-separated convolutional networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 5551–5560.
[39] P. Wang, L. Liu, C. Shen, and H. T. Shen, "Order-aware convolutional pooling for video based action recognition," *Pattern Recognit.*, vol. 91, pp. 357–365, 2019.
[40] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime TV-L1 optical flow," in *Proc. Joint Pattern Recognit. Symp.*, 2007, pp. 214–223.
[41] A. Dosovitskiy *et al.*, "FlowNet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2758–2766.
[42] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of optical flow estimation with deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1647–1655.
[43] L. Fan, W. Huang, C. Gan, S. Ermon, B. Gong, and J. Huang, "End-to-end learning of motion representation for video understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6016–6025.
[44] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2720–2729.
[45] Y. Zhao, Y. Xiong, and D. Lin, "Recognize actions by disentangling components of dynamics," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6566–6575.
[46] N. Hussein, E. Gavves, and A. W. Smeulders, "Timeception for complex action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 254–263.
[47] I. Misra, C. L. Zitnick, and M. Hebert, "Shuffle and learn: Unsupervised learning using temporal order verification," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 527–544.
[48] J. Wang, J. Jiao, L. Bao, S. He, Y. Liu, and W. Liu, "Self-supervised spatio-temporal representation learning for videos by predicting motion and appearance statistics," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4001–4010.

[49] P. Chen *et al.*, "RSPNet: Relative speed perception for unsupervised video representation learning," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 1045–1053.
[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[51] L. Chen *et al.*, "SCA-CNN: Spatial and channel-wise attention in convolutional networks for image captioning," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, 2017, pp. 6298–6306.
[52] G. Varol, I. Laptev, and C. Schmid, "Long-term temporal convolutions for action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1510–1517, Jun. 2018.
[53] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," 2021, *arXiv: 2103.03230*.
[54] O. Kopuklu, N. Kose, A. Gunduz, and G. Rigoll, "Resource efficient 3D convolutional neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops*, 2019, pp. 1910–1919.
[55] B. Zhou, A. Andonian, A. Oliva, and A. Torralba, "Temporal relational reasoning in videos," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 803–818.
[56] C. Yang, Y. Xu, J. Shi, B. Dai, and B. Zhou, "Temporal pyramid network for action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 588–597.
[57] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7794–7803.
[58] X. Wang and A. Gupta, "Videos as space-time region graphs," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 399–417.
[59] L. Wang, W. Li, W. Li, and L. Van Gool, "Appearance-and-relation networks for video classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1430–1439.
[60] D. He *et al.*, "StNet: Local and global spatial-temporal modeling for action recognition," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 01, 2019, pp. 8401–8408.
[61] H. Bilen, B. Fernando, E. Gavves, and A. Vedaldi, "Action recognition with dynamic image networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2799–2813, Dec. 2018.
[62] G. Varol, I. Laptev, and C. Schmid, "Long-term temporal convolutions for action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1510–1517, Jun. 2018.
[63] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 2579–2605, 2008.

Mengmeng Wang received the BS degree and the MS degree in control science and engineering from Zhejiang University, Zhejiang, China, in 2015 and 2018, respectively. She is currently working toward the PhD degree with the Laboratory of Advanced Perception on Robotics and Intelligent Learning, College of Control Science and Engineering, Zhejiang University. Her research interests include visual tracking, action recognition, computer vision, and deep learning.

Jiazheng Xing received the BS degree from Chongqing University, Chongqing, China, in 2021. He is currently working toward the MS degree with the Laboratory of Advanced Perception on Robotics and Intelligent Learning, College of Control Science and Engineering. His research interests include action recognition, computer vision, and deep learning.

Jing Su received the PhD degree in optical science and engineering from Fudan University, Shanghai, China, in 2021. She is currently a researcher wit the Department of Smart City, SenseTime. Her research interests include spiking neural network, low-bit computation, Lightweight neural network.

Jun Chen received the MS degree in automation from the Zhejiang University, Hangzhou, China, in 2020. He is currently working toward the PhD degree with the Institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang University, Hangzhou, China. His research interests include neural network quantization and deep learning.

Yong Liu (Member, IEEE) received the BS degree in computer science and engineering and the PhD degree in computer science from Zhejiang University, Zhejiang, China, in 2001 and 2007, respectively. He is currently a professor with the Institute of Cyber-Systems and Control, Zhejiang University. His main research interests include: robot perception and vision, deep learning, Big Data analysis, and multi-sensor fusion. His research interests on machine learning, computer vision, information fusion, and robotics.