Temporal Transformer Networks With Self-Supervision for Action Recognition

Yongkang Zhang, Jun Li, Na Jiang, Guoming Wu, Han Zhang, Zhiping Shi, *Member, IEEE*, Zhaoxun Liu, Zizhang Wu, and Xianglong Liu, *Member, IEEE*

***Abstract*—In recent years, Internet of Things (IoT) has made rapid development, and IoT devices are developing toward intelligence. IoT terminal devices represented by surveillance cameras play an irreplaceable role in modern society, most of them are integrated with video action recognition and other intelligent functions. However, their performance is somewhat affected by the limitation of computing resources of IoT terminal devices and the lack of long-range nonlinear temporal relation modeling and reverse motion information modeling. To address this urgent problem, we introduce a startling temporal transformer network with self-supervision (TTSN). Our high-performance TTSN mainly consists of a temporal transformer module and a temporal sequence self-supervision (TSS) module. Concisely speaking, we utilize the efficient temporal transformer module to model the nonlinear temporal dependencies among nonlocal frames, which significantly enhances complex motion feature representations. The TSS module we employ unprecedentedly adopts the streamlined strategy of "random batch random channel" to reverse the sequence of video frames, allowing robust extractions of motion information representation from inversed temporal dimensions and improving the generalization capability of the model. Extensive experiments on three widely used data sets (HMDB51, UCF101, and Something–Something V1) have conclusively demonstrated that our proposed TTSN is promising as it successfully achieves state-of-the-art performance for video action recognition. Our TTSN provides the possibility for its application in IoT scenarios due to its computational complexity and high performance.**

***Index Terms*—Temporal modeling, temporal sequence self-supervision (TSS), temporal transformer, video action recognition, video analysis in Internet of Things (IoT).**

I. Introduction

IN RECENT years, Internet of Things (IoT) has been developed rapidly, and IoT devices, such as surveillance cameras, play an important role in modern society. And most of them are equipped with intelligent applications, such as action recognition, anomaly detection, etc. This poses a new requirement for IoT, i.e., a lightweight and efficient video recognition model is needed for real-time action recognition in video, and in this article, we explore the above problem. As we all know, deep neural network (DNN) has made great progress [2], [3], and DNN-based video action recognition methods have undergone considerable progress [4], [5], [6], [7], [8]; notwithstanding, with extra temporal dimension in videos, it is still challenging to devise a high-performance recognition approach [9], [10], [11]. An essential key to the breakthrough of this issue lies in the understanding of motion information derived from given videos. This ability requires the neural network to possess a solid modeling ability to thoroughly seize the motion information representation in the temporal dimension of a given video. To contrive to meet this requirement, the video action recognition algorithm [12] principally incorporates optical flow-based methods [6], 3-D CNNs-based methods [13], [14], and 2-D CNNs with temporal modeling methods [1], [5]. Regarding the optical flow-based methods applied, the temporal sequence of motion information is modeled by optical flows. Nevertheless, a lapse of these methods alone is that the optical flow calculation demands numerous overheads, making it not so hospitable to apply. Subsequently, the 3-D CNNs-based methods extract motion information directly from RGB frames, end-to-end, by 3-D convolution; however, it is unwantedly costly to compute and deploy such a network. Last but not least, the 2-D CNNs-based methods, such as RNN [15], [16], model a given video into an ordered sequence of frames; yet they focus solely on capturing crude temporal structure relationship among frames [10], making the 2-D CNNs-based methods alone not so ideal to model motion information effectively and robustly.

Delightfully, attempts have been made [1], [6], [17], [18], [19] to, via 2-D CNNs, directly extract motion information representations from RGB frames, having assisted in lifting the above issues concerning recognition accuracy and computation complexity. However, most of these methods model temporal dimensions-based singularly on first-order temporal differences, overlooking long-distance nonlinear temporal relations and reversed motions among frames. Such deficiencies undeviatingly result in a nonnegligible performance loss for video action recognition: even a simple action, such as "sit" shown in Fig. 1, cannot be correctly recognized. There are also some other works have employed using Transformer to process video data, Girdhar et al. [20] employed Transformer

Manuscript received 4 January 2023; revised 13 February 2023; accepted 4 March 2023. Date of publication 16 March 2023; date of current version 7 July 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62106158; in part by the State Key Laboratory of Software Development Environment under Grant SKLSDE-2020KF-08; and in part by the Research and Development Program of Beijing Municipal Education Commission under Grant KM202210028007. *(Corresponding author: Jun Li.)*

Yongkang Zhang, Jun Li, Na Jiang, Guoming Wu, Han Zhang, and Zhiping Shi are with the Information Engineering College, Capital Normal University, Beijing 100048, China (e-mail: junmuzi@gmail.com).

Zhaoxun Liu and Xianglong Liu are with the State Key Laboratory of Software Development Environment, School of Computer Science and Engineering, Beihang University, Beijing 100191, China.

Zizhang Wu is with the Computer Vision Perception Department, ZongMu Technology, Shanghai 201203, China.

Digital Object Identifier 10.1109/JIOT.2023.3257992

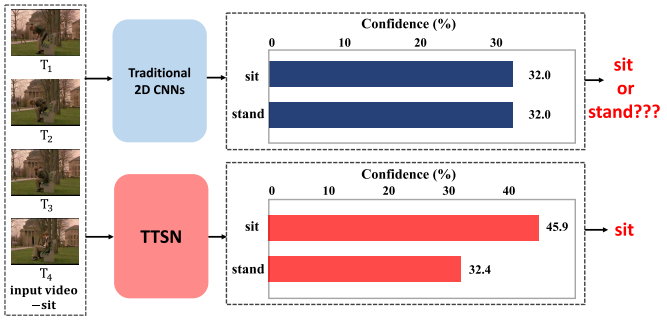

Fig. 1. Example comparison of recognition results between traditional 2-D CNNs (e.g., TDN [1]) and our TTSN model in the inference stage on the HMDB51 data set. We notice that the actions (e.g., sit and stand) that are reverse-ordered in the temporal dimension but similar in the spatial dimension are very confusing to the traditional 2-D CNNs. When inputting a video clip with the sit category, traditional 2-D CNNs, lacking effective modeling of the temporal dimension, recognize it as sit with 32.0% confidence and stand with a veritably same-weighted 32.0% confidence, making the model fail to make the correct prediction. However, our TTSN model, adaptively combining the temporal transformer module and the TSS module and modeling the complex and nonlinear temporal motion features over long-range frames, recognizes it as sit with 45.9% confidence and stand with 32.4% confidence, making the two actions significantly distinguishable.

to encode information of 3-s spanning the given clip, which limits the model's ability to model the long-range temporal dimension, and the model relies too much on context information. Jiang et al. [21] proposed a two-pathway transformer network that uses memory-based attention to explore motion information, which has expensive computational overhead and the model exceedingly depends on the quality of frame differences. Plizzari et al. [22] treated each joint independently and looks for information on the variability of the same joint along a given temporal dimension; however, it requires additional skeleton auxiliary data to complete the video activity recognition.

To address the above issues, we introduce a temporal transformer network with self-supervision (TTSN) to efficiently and accurately perform complex temporal motion information modeling. Our TTSN consists of an efficient temporal transformer (ETT) module combined with a temporal sequence self-supervision (TSS) module. The temporal transformer module mainly includes a temporal position embedding submodule and a temporal transformer coding layer submodule, which is used to model complex long-range nonlinear motion information. Furthermore, the ETT module can: 1) perform efficient pixel-level modeling on the temporal dimension of a given video; 2) excavate motion information representations held in nonlocal frames; and 3) recognize and transmit motion-sensitive pixels in the given spatial dimension.

Naturally, human beings, after iterated learning, possess the spontaneous adeptness to define complex, even befuddling, actions, such as "sit" and "stand," which hold high spatial similarity yet reverse to one another in temporal sequence order. Spurred by this rationale, we harness self-supervision learning to model reverse motion learning and devise a TSS module. The ETT module learns an action from front to back, while the TSS module learns the very one over from back to front. Thus, the network can learn and recognize the variability of motions of complex actions in any given temporal dimension, thanks to which the network is capable of distinguishing these actions correctly and efficiently. In addition, the TSS module utilizes unlabeled data to let a neural network robustly learn global reverse motion information representations in a given temporal dimension, which is manageable, ingenious, versatile, and smooth to integrate into a training phase of a network with only a thin volume of supplementarily added computational overheads.

Foremostly, our contributions can be summarized as follows.

1) In this article, we introduce a novel 2-D CNN-based action recognition network TTSN. Our TTSN abandons unnecessarily copious amounts of optical flow computation and thanks to which it can outmatch the performance of 3-D CNNs while still managing to preserve the complexity of 2-D CNNs.
2) We introduce a high-performance temporal transformer module to excavate long-range and complex nonlinear motion information representations carried in a temporal dimension and perform pixel-level dependency modeling.
3) We devise a TSS module to let the network learn actions from back to front along a temporal dimension, applied by which the network can extract a thoroughness of high-level motion direction perception information to enhance the robustness of the network and recognize complex actions.
4) Our proposed TTSN model, compared to the existing state-of-the-art methods, achieves the most high-grade performance on several mainstream data sets, such as HMDB51, UCF101, and Something–Something V1, demonstrated by a large number of extensive experiments conducted.

II. Related Work

In this section, we review several of the most related methods for 2-D CNNs-based Action Recognition, Transformer, and Self-Supervised Learning. We further discuss the differences between them and this work.

A. 2-D CNNs-Based Action Recognition

Recently, video action recognition is a prominent research hotspot. With the development of DNNs, 2-D CNN-based methods are prevalent among video action recognition solutions. The work [17] proposes a time shift module to move part of the channel along the temporal dimension to model the motion information among adjacent frames. And the work [18] proposes a spatiotemporal coding module to encode and model spatiotemporal features. In considering computational performance, a temporal information enhancement module [23] is introduced to decouple the correlation among channels and interactively model the temporal information. Weng et al. [24] proposed a frame-level feature filtering mechanism to diminish information redundancy. The above methods are limited to modeling motion information in the local frame range, and the parameters and modeling capabilities of the models are very limited. Li et al. [19] designed a time excitation and aggregation module to calculate motion-sensitive channels. But

channel dimension modeling does not seem to be particularly effective in the video domain. Huang et al. [25] proposed a time pyramid network to perform multiscale modeling of spatio-temporal features. Liu et al. [26] proposed an efficient temporal dynamic convolution kernel. In addition, Wang et al. [1] used the difference information among frames to model motion information. Wang et al. [27] performed adaptive calculations on spatiotemporal characteristic and channel response. These methods all work pretty well, but introduce an expensive computational overhead that is not cost effective. At the same time, they cannot be applied to IoT scenarios with limited computing resources.

B. Transformer

The author of the self-attention mechanism first proposes this method in natural language processing. Gradually, this method permeates the research of computer vision. The self-attention mechanism [28], [29], [30], [31], [32], [33] allows the neural network to selectively focus on a particular part of the input and extract useful neural network perception information. Vaswani's groundbreaking work [34] proposes the self-attention mechanism for the first time and successfully applies it to natural language processing solutions. Since then, researchers of computer vision have also adapted this method. Ma et al. [35] proposed a video saliency forecasting transformer and achieves superior performance on both VSF and VSP tasks. The focus is on enhancing spatiotemporal modeling capabilities by allowing the model to predict the salience of future video frames. Dosovitskiy et al. [36] united the self-attention mechanism with the transformer network structure and applies it to a solution of image classification. This method is only applicable to image field. Du et al. [37] proposed an interactive self-attention model to model pixels in a local space. However, it is not enough to perform such self-attention in local space region. Fu et al. [38] and Yuan et al. [39] employed cross-sparse self-attention to model the feature map in a spatial perspective. Wang et al. [40] performed simple pixel-level modeling of nonlocal feature maps. Girdhar et al. [20] and Jiang et al. [21] employed Transformer to encode and model motion information and fuse context information for video action recognition and detection. These methods neglect the importance of modeling the temporal dimension, which contains rich motion information. Plizzari et al. [22] explored the difference information of the same joint along the spatial and temporal dimensions to conduct effective skeleton-based action recognition. Ma et al. [41] fused heatmaps from different views to improve the accuracy of 2-D keypoint prediction. Chen et al. [42] proposed a bi-directional GRU deep-learning method, called TRANS-BiGRU, to efficiently learn and recognize different types of activities performed by multiple residents. Skeleton and different view information will introduce expensive computational overhead.

C. Self-Supervised Learning

Self-supervised learning is a novel neural network learning model. It practices an auxiliary pretext task to provide supervision signals for feature learning using unlabeled training data, thanks to which the network can learn the transferable visual feature representation and apply the representation to downstream tasks. Traditional self-supervised learning excavates transferable visual information representation from the spatial dimension. Zhu et al. [43] presented a novel two-stage framework that combines Instance-CL and unsupervised clustering to progressively learn desirable temporal representations with high intraclass compactness. Wang et al. [44] proposed a co-occurrence action parsing algorithm that uses unlabeled data to learn correlations between actions. This method brings less performance gain and does not introduce temporal information. Kim et al. [45] proposed an idea of clipping, chunking, and rearranging the obtained spatio-temporal blocks in spatio-temporal dimensions; this approach generates exponential species possibilities and makes the network most likely to learn low-level clues. This method is highly stochastic and leads to a significant impact on the model's modeling ability. Gidaris et al. [46] practiced a rotation pretext task, which randomly rotates the input image. The classifier is employed to control the rotation angle of the image. Zhang et al. [47] practiced the pretext task of predicting the color of a specific channel. Pathak et al. [48] proposed the pretext task of image restoration from the perspective of spatial context. After developing rapidly, self-supervised learning has also been frequently adopted in other particular fields of computer vision, such as medical image segmentation. These methods are simple to implement but have little success in video recognition tasks. At the same time, self-supervised learning is also applied in many other fields, such as unsupervised visual representation learning [49], [50], [51], [52], knowledge distillation [53], reidentification [54], unsupervised optical flow algorithm [55], Visual Question Answering [56], dense face alignment [57], [58], [59], and vision-based industrial smoke emissions recognition [60]. Considering and weighing the advantages and disadvantages of the various methods mentioned above, we propose the temporal self-supervised algorithm in this article.

III. Temporal Transformer Network With Self-Supervision

In this section, we first present an overview of our method, including the crucial components of our proposed TTSN and a concise description of each essential module. We then describe each module specifically, including the precise principle we utilize, implementation details, formalized definitions, etc. Eventually, we present a formal definition of the loss function and elucidate its rationale.

A. Overview

In this article, we introduce a novel method TTSN, a 2-D CNNs-based video action recognition network, for video action recognition. Our TTSN principally consists of an ETT module, a TSS module, and a ResNet-based backbone. The ETT module is chiefly engaged in modeling, both nonlocal and nonlinear, of a given temporal dimension and thoroughly excavating the motion information carried in the temporal

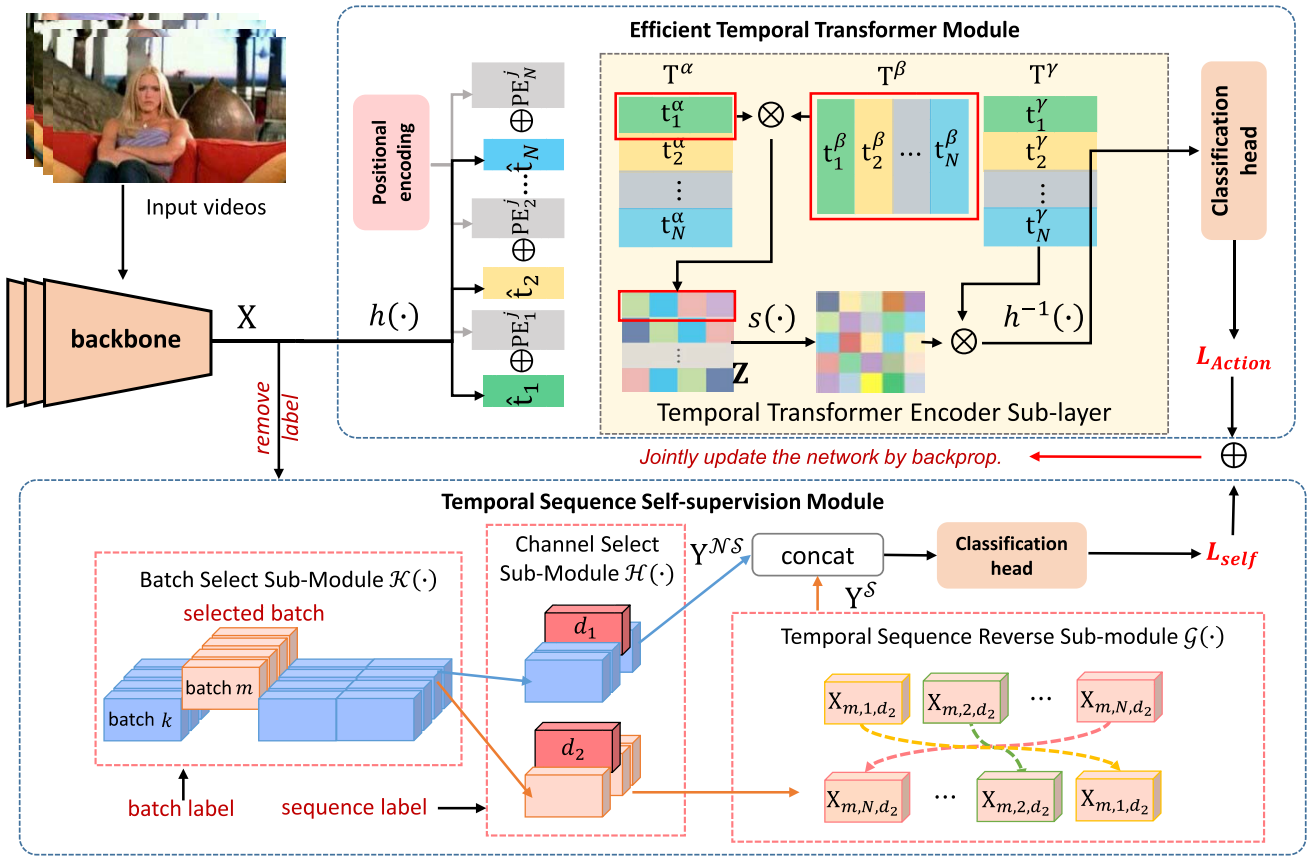

Fig. 2. Overall architecture of our TTSN. Our TTSN consists of a Backbone $\mathcal{F}$, an ETT module, and a TSS module. In the training phase, we employ both the ETT module and the TSS module. In the testing phase, we employ only the ETT module.

dimension of a given video. The TSS module is for reverse-ordered processing of unlabeled, randomly sampled videos from a data set. When executing the reverse-ordered processing, the TSS module randomly selects a channel to process. Ultimately, the TSS module makes the backbone to perceive and derive rich and high-level motion direction information; it also enhances the robustness of the network, making it more sophisticated in distinguishing complex actions.

As shown in Fig. 2, the input videos contain multiple variable-length videos. Using the sparse temporal sampling strategy proposed in [1] and [6], a large set of video frames (termed as $N f$) is obtained for each video.

In the phase of feature extraction, we assign B videos as the input, where B is the size of a batch. Input videos initially pass through the backbone and collect a feature map $\mathbf{X} \in \mathbb{R}^{B\times N\times C\times H\times W}$. Distinctively, in the phase of training, $\mathbf{X}$ passes through the ETT module and the TSS module, and we utilize the corresponding loss function to constrain the output of ETT and TSS to reach a total loss, after which a robust nonlinear motion modeling in both directions, forward and backward, can be realized. Eventually, in the phase of testing, $\mathbf{X}$ passes exclusively through the ETT module and completes an efficient inference.

B. Efficient Temporal Transformer Module

A temporal dimension of a given video contains rich motion information. Therefore, the ability of a network to model motion information would be essential to the model and determine the performance of the final classification. Inspired by [34] and [36], we propose a simple temporal transformer structure, namely, an ETT module, to efficiently model the complex motion information of a temporal dimension with a nonlinear and nonlocal method.

According to Fig. 2, the ETT module accepts feature map $\mathbf{X}$ as input and principally comprises a positional encoding submodule and a temporal transformer encoder submodule. To demonstrate explicitly, here, we take only one batch (batch k) of $\mathbf{X}$ as an instance and reasonably redefine $\mathbf{X}_k$ as $\hat{\mathbf{X}} = \{\hat{\mathbf{X}}_i \in \mathbb{R}^{C\times H\times W} | i = \{1, 2, \ldots, N\}\}$ to elucidate the principle of the ETT module. In enhancing the modeling ability of the ETT module and avoiding long-winded computation, a linear transformation function $h(\cdot)$ is employed to accomplish frame-level feature embedding and is tied for the transformation of the feature map $\hat{\mathbf{X}}_i$, altering the dimension of each video frame from $C \times H \times W$ to $1 \times l$ ($l \ll CHW$), and obtaining the 1-D frameset $\hat{\mathbf{T}} = \{\hat{\mathbf{t}}_i \in \mathbb{R}^{1\times l} | i = \{1, 2, \ldots, N\}\}$. Subsequently, we add a learnable and randomly initialized positional encoding $\mathbf{PE}_i^j \in \mathbb{R}^{1\times l}$ ($j \in \{\alpha, \beta, \gamma\}$) to $\hat{\mathbf{t}}_i$, representing the raw 1-D frame tensor forsaking the positional encoding obtained after each frame $\hat{\mathbf{X}}_i$ passes through the frame embedding module $h(\cdot)$. The above process can be formalized as follows:

$$\mathbf{t}_i^j = \hat{\mathbf{t}}_i + \mathbf{PE}_i^j \tag{1}$$

$$\hat{\mathbf{t}}_i = h\left(\hat{\mathbf{X}}_i\right) \tag{2}$$

where $\mathbf{t}_i^j$ is the 1-D frame tensor with positional encoding.

To facilitate the temporal transformer encoder submodule to learn temporal nonlinear motion features more versatilely and convertibly, we add three sets of independent positional encodings to each 1-D frame tensor and collect the calculation result of each frame $\mathbf{t}_i^j$, as shown in (1). Subsequently, we sequentially concatenate all 1-D frame tensors $\mathbf{t}_i^j$ to, respectively, construct three independent temporal encoding feature maps $\mathbf{T}^j$, where $j \in \{\alpha, \beta, \gamma\}$.

As for the temporal transformer encoder, we use $\mathbf{T}^\alpha$, $\mathbf{T}^\beta$, and $\mathbf{T}^\gamma$ to process nonlinear and nonlocal modeling on a temporal dimension of a given video. We devise a simple and ETT encoder submodule, containing no MLP layer and other unnecessary structures such as the decoder structure. It not only lessens the computational cost and reasoning time consumption but also promotes the modeling ability of the model in the temporal dimension. The rationale of the transformer encoder can be formalized as follows:

$$\mathbf{A}^* = h^{-1}\left(s(\mathbf{Z})\mathbf{T}^\gamma\right) \tag{3}$$

$$\mathbf{Z} = \lambda \mathbf{T}^\alpha \left(\mathbf{T}^\beta\right)^T \tag{4}$$

where $s(\cdot)$ is the normalization function. $\mathbf{A}^*$ represents the auxiliary attention map with dimension $BN \times C \times H \times W$, containing motion sensitive pixel information in the temporal dimension. These motion-sensitive information can guide the network to complete the classification of action categories. $h^{-1}(\cdot)$ represents the operation of inverse transformation. $\mathbf{Z}$ represents the relation matrix, portraying the long-range nonlinear dependencies of frames along the given temporal dimension. λ is a learnable parameter and is used to enhance the learning ability and improve the expressiveness of the network.

We consider that $\mathbf{A}^*$ is capable of guiding the network to complete the classification of action categories since it concentrates on the region of motion-sensitive pixels on the spatial scale of video frames, supported by the notion that regions where motion-sensitive pixels are exactly where the network's attention should be. The attention is calculated by the temporal transformer encoder sublayer in the ETT module through nonlocal and nonlinear modeling on the given temporal dimension. In the experiment section, we conduct a visualization operation on $\mathbf{A}^*$.

Here, we succinctly elucidate the implementation of each submodule of the ETT module. First, $h(\cdot)$ represents the frame-level feature embedding submodule and can be realized by a convolution and a linear projection operation. The positional encoding submodule can generate a learnable and randomly initialized positional encoding, which is superimposed directly on 1-D frame tensor $\hat{\mathbf{t}}_i$. Next, $h^{-1}(\cdot)$ represents the inverse transformation submodule, which can be realized by a convolution operation. And $s(\cdot)$ is a normalization function implemented with Softmax.

C. Temporal Sequence Self-Supervision Module

Human beings learn a new action by repeatedly observing; regularly, this process acquires a large set of instances, especially, when the one is complex and befuddling. These actions, yet reversed along the temporal sequence, are similar in the spatial dimension. When a temporal sequence is modified or changed, actions on the sequence may be mistreated as different ones. Some of these actions are too complicated for 2-D CNNs-based methods [1] to process. Therefore, when the model can distinguish this temporal difference, the classification performance, as well as the robustness of the model, could be enhanced. Inspired by the learning behavior in human beings, we examine self-supervised learning to model reversed motion learning with a temporal sequence self-supervised (TSS) module, which uses unlabeled video data. TSS can incorporate with the ETT module to achieve bidirectional motion information modeling. Concretely, we consider processing the actions in reversed order along the temporal dimension and contrive to make the neural network learn to distinguish whether the current action is in the reversed order along the temporal dimension. We endeavor to enable the model to learn the same action from back to front along the temporal dimension. In this way, the model can distinguish the complicated actions stated above; also, the robustness of the network is strengthened.

In the pretext task design for TSS module, concerning to the stochasticity strategy for processing the time dimension, we devise a myriad of algorithms, namely, "All batch all channel-Reverse ($\mathcal{AA}$)," "Random batch all channel-Reverse ($\mathcal{RA}$)," "All batch random channel-Reverse ($\mathcal{AR}$)," and "Random batch random channel-Reverse ($\mathcal{RR}$)."

The $\mathcal{RR}$ algorithm possesses the highest randomness and leading performance among the others; so in TTSN, we use the algorithm of $\mathcal{RR}$ by default. As shown in Fig. 2, after inputting the feature map into the TSS module, the $\mathcal{RR}$ algorithm first generates batch-level pseudo-labels and sequence-level pseudo-labels for the input feature maps. For batch-level pseudo-labels, there are two categories: 1) Selected Batch (termed as $\mathcal{S}$) and 2) Not Selected Batches (termed as $\mathcal{NS}$). For sequence-level pseudo-labels, there are also two categories: 1) Temporal Norm Sequence (termed as *NOR*) and 2) Temporal Reverse Sequence (termed as *REV*).

For batches (e.g., batch k) whose label is $\mathcal{NS}$, we randomly select one channel (e.g., channel d_1) from each frame in the batch and maintain the original temporal sequence unchanged, then the feature maps $\mathbf{X}_{k,d_1} \in \mathbb{R}^{N\times 1\times H\times W}$ can be obtained. Finally, we concatenate them to construct the result feature maps $\mathbf{Y}^{\mathcal{NS}} \in \mathbb{R}^{B^{\mathcal{NS}}\times N\times 1\times H\times W}$, where $B^{\mathcal{NS}}$ is the number of unselected batches; in addition, the sequence-level pseudo-label of the video frames in the batch is *NOR* to form the label set $\mathbf{G}^{\mathcal{NS}} \in \mathbb{R}^{B^{\mathcal{NS}}}$. For batches (e.g., batch m) with a pseudo-label of $\mathcal{S}$, we also randomly select one channel (e.g., channel d_2) from the N frames in the batch to obtain the feature maps $\mathbf{X}_{m,d_2}$. Then, we reverse the order of the channels in the temporal dimension to obtain the feature maps $\mathbf{Y}^{\mathcal{S}} \in \mathbb{R}^{B^{\mathcal{S}}\times 1\times H\times W}$, where $B^{\mathcal{S}}$ is the number of selected batches. Eventually, the sequence-level pseudo-label of the video frame is *REV*, forming the label set $\mathbf{G}^{\mathcal{S}} \in \mathbb{R}^{B^{\mathcal{S}}}$.

$\mathcal{AA}$ selects all batches, reverses all frames, and uses all channels in the reversing operation; $\mathcal{RA}$ randomly selects one batch to reverse and uses all channels; $\mathcal{AR}$ selects all batches to reverse, and randomly selects one channel from each batch

TABLE I
RELATION BETWEEN THE SELF-SUPERVISION ALGORITHM AND THE DIFFERENT VALUES OF HYPERPARAMETERS ρ, η

Algorithm	Γ	ρ	η
$\mathcal{AA}$	$\mathcal{G}$	1	1
$\mathcal{RA}$	$\mathcal{G} \circ \mathcal{K}$	1	0
$\mathcal{AR}$	$\mathcal{G} \circ \mathcal{H}$	0	1
$\mathcal{RR}$	$\mathcal{G} \circ \mathcal{H} \circ \mathcal{K}$	0	0

to reverse. The rationale of the TSS module can be formalized by the following:

$$\Gamma(\varepsilon, \rho, \eta) = (1-\varepsilon)\mathcal{G} \circ (1-\rho)\mathcal{H} \circ (1-\eta)\mathcal{K} \tag{5}$$

where "$\circ$" represents the compound operation of the function; $\mathcal{G}(\cdot)$ represents the temporal sequence reversion; $\mathcal{H}(\cdot)$ represents the random channel selection; $\mathcal{K}(\cdot)$ represents the random batch selection; ε, ρ, and η represent the hyperparameters to control the final self-supervision algorithm Γ, where $\varepsilon, \rho, \eta \in \{0, 1\}$ and $\Gamma \in \{\mathcal{AA}, \mathcal{AR}, \mathcal{RA}, \mathcal{RR}\}$. Notice that the operations of $\mathcal{G}, \mathcal{H}$, and $\mathcal{K}$ do not comply with the commutative law.

Based on different values of ε, ρ, and η in (5), Γ can be transformed into different self-supervision algorithms. The coefficients of the function [e.g., $\mathcal{G}(\cdot)$, $\mathcal{H}(\cdot)$, and $\mathcal{K}(\cdot)$] remain 0 if the function does not participate in the compound operation of the function; otherwise, 1. Notice the reversion can only execute for the TTSN model, so we assign ε to 0. The relation between the self-supervision algorithm and different values of hyperparameters ρ and η is manifested in Table I.

Since the algorithm of $\mathcal{RR}$ demands the most moderate GPU configuration notwithstanding still performs the best (this will be demonstrated more evidently in the experiment section), we select $\mathcal{RR}$ as the auxiliary task for the TSS module in this article. We take the $\mathcal{RR}$ algorithm as an instance to illustrate the fundamental flow, which is detailed in Algorithm 1.

Algorithm 1 $\mathcal{RR}$ Algorithm of the TTSN Module

Input: Training video set $\mathbf{X} \in \mathbb{R}^{B\times N\times C\times H\times W}$ and corresponding hyper-parameters $\varepsilon = 0, \rho = 0, \eta = 0$.
Output: The output feature maps $\mathbf{Y}$ and corresponding labels $\mathbf{G}$.
Parameters:
$\mathbf{Y}$: Total unlabeled data after processing.
$\mathbf{G}$: Pseudo-labels for classification.
$\mathbf{B}$: batchsize.
$\mathbf{Y}^{\mathcal{NS}}$ / $\mathbf{Y}^{\mathcal{S}}$: Unlabeled data after processing, whose batch-label is $\mathcal{NS}$ / $\mathcal{S}$.
$\mathbf{G}^{\mathcal{NS}}$ / $\mathbf{G}^{\mathcal{S}}$: Pseudo-labels set for classification, whose value is *NOR* / *REV*.

Init $\mathbf{Y}$,$\mathbf{G}$={}
for b in B **do**
  Init $\mathbf{Y}^{\mathcal{NS}}$,$\mathbf{Y}^{\mathcal{S}}$,$\mathbf{G}^{\mathcal{NS}}$,$\mathbf{G}^{\mathcal{S}}$={}
  $s_b = \mathcal{K}(\mathbf{X}_b)$
  if $s_b = \mathcal{NS}$ **then**
    $\mathbf{G}_{b,\forall N} = NOR$
    $d_1 = \mathcal{H}(\mathbf{X}_b)$
    $\mathbf{Y}_b = \mathbf{X}_{b,d_1}$
    $\Leftrightarrow [\mathbf{x}_{b,1,d_1}, \mathbf{x}_{b,2,d_1}, \ldots, \mathbf{x}_{b,N,d_1}]$
    $\mathbf{Y}^{\mathcal{NS}} = [\mathbf{Y}^{\mathcal{NS}},\mathbf{Y}_b]$,$\mathbf{G}^{\mathcal{NS}} = [\mathbf{G}^{\mathcal{NS}},\mathbf{G}_{b,\forall N}]$
  else if $s_b = \mathcal{S}$ **then**
    $\mathbf{G}_{b,\forall N} = REV$
    $d_2 = \mathcal{H}(\mathbf{X}_b)$
    $\mathbf{Y}_b = \mathcal{G}(\mathbf{X}_{b,d_2})$
    $\Leftrightarrow [\mathbf{x}_{b,N,d_2}, \mathbf{x}_{b,N-1,d_2}, ..., \mathbf{x}_{b,1,d_2}]$
    $\mathbf{Y}^{\mathcal{S}} = [\mathbf{Y}^{\mathcal{S}},\mathbf{Y}_b]$,$\mathbf{G}^{\mathcal{S}} = [\mathbf{G}^{\mathcal{S}},\mathbf{G}_{b,\forall N}]$
  end if
  $\mathbf{Y} = [\mathbf{Y},\mathbf{Y}^{\mathcal{NS}},\mathbf{Y}^{\mathcal{S}}]$, $\mathbf{G} = [\mathbf{G},\mathbf{G}^{\mathcal{NS}}, \mathbf{G}^{\mathcal{S}}]$
end for

D. Loss Function

In this section, we elucidate the loss function of our TTSN. It consists of two components: 1) an ETT module loss L_{Action} and 2) a TSS loss L_{self}. The L_{Action} is for constraining the ETT module to model complex nonlinear motion information in a given temporal dimension and ultimately perform accurate classification. Likewise, the L_{self} is for constraining the backbone and deriving reversed motion features.

1) Loss Function of ETT: The ETT module we proposed is very adaptable and can be added to veritably any locality of the network. Supported by our experience, with an added locality getting imminent to the very front of the network, the demand for extra computational overheads induced about would expand exponentially, causing a considerable loss in performance; such an impairment is uninvited. Therefore, to ensure efficiency and accuracy in computation, we append the ETT module to the end of our network to excavate the underutilized motion information in the given temporal dimension of a high-level abstract feature map. In the process of training, the loss function of the ETT is defined as follows:

$$L_{\text{Action}}(\omega_1; \mathbf{A}^*, \mathbf{X}) = -\sum_{p=1}^{c} t_p \log\left(\mathcal{R}^p_{\omega_1}(\mathbf{A}^* + \mathbf{X})\right) \tag{6}$$

where c represents the total number of categories; t_p represents the category label; $\mathcal{R}^p_\rho(\cdot)$ represents the standardized prediction score of action category p; and ω_1 represents the parameters of the action classifier. We apply this loss function to the whole training set of the corresponding data set.

2) Loss Function of TSS: During the training process, we implement an auxiliary self-supervised loss function to constrain the training process of TTSN in the temporal dimensions. In this work, the training set is from the parental training set, modifications are that we remove its original labels and introduce new pseudo-labels to grant supervision signals. The loss function of the TSS module is defined as follows:

$$L_{\text{self}}\left(\omega_2; Y^{\mathcal{NS}}, \mathbf{Y}^{\mathcal{S}}\right) = -\sum_{t_q\in\{0,1\}} \left(t_q \log\left(\mathcal{H}^q_{\omega_2}\left(\left[\mathbf{Y}^{\mathcal{NS}}, \mathbf{Y}^{\mathcal{S}}\right]\right)\right)\right) \tag{7}$$

where $t_q \in \{0, 1\}$ represents the value of pseudo-labels of selected and unselected batches; $[\cdot, \cdot]$ denotes the concating

operation. We apply this loss function on an unlabeled data set randomly sampled from the training set, whose original labels have been deprecated; $\mathcal{H}^{q}_{\omega_2}(\cdot)$ represents, concerning to the prediction of the temporal order of frames, the normalized prediction scores of the selected and unselected batches; and ω_2 represents the parameters of the self-supervised classifier.

3) Total Loss of TTSN: Combining Formulation (6) and (7), we define the total loss function as follows:

$$L = \theta_1 L_{\text{Action}}\left(\omega_1; \mathbf{A}^*, \mathrm{X}\right) + \theta_2 L_{\text{self}}\left(\omega_2; Y^{\mathcal{NS}}, \mathrm{Y}^{\mathcal{S}}\right) \quad (8)$$

where θ_1 and θ_2 are hyperparameters used to balance the weights of the ETT module and the respective TSS module.

According to the experiments we conducted, being the value of θ_2 grows, concurrently, the confinement $\mathcal{E}$ of the TSS module strengthens, which satisfies $\mathcal{E} \propto \theta_2$. We assign $\theta_1 = 1.0$ and restrict θ_2 to qualify the relation $\theta_1/\theta_2 \in [10, 100]$ empirically; the value of θ_2 depends on the scale of the data set.

IV. Experiment

In this section, we first establish the context and settings of our experiments; subsequently, we present the results of our proposed TTSN model being experimented on three frequently used data sets: 1) HMDB51; 2) UCF101; and 3) Something–Something V1 and compare them with several state-of-the-art methods. This section culminates in the exhibition of our detailed ablation experiments and visualized results for each.

A. Data Sets

We verify our proposed TTSN on three frequently used data sets: 1) HMDB51; 2) UCF101; and 3) Something–Something V1. Among them, HMDB51 [61] contains 51 categories and 6849 videos. Each category comprises at least 51 videos, most of which are from YouTube and Google, with a resolution of 320×240. UCF101 [62] is a data set, particularly for action recognition, comprising numerous real action videos which are from YouTube, providing 13 320 videos from 101 categories. Something–Something V1 [63] is a relatively massive data set of 174 categories of actions and 108 499 videos, containing various daily actions between humans and mundane objects. It would be unlikely to determine what the individual in a video is doing depending solely on one frame extracted. The above three data sets comprehensively evaluate our proposed TTSN model from different regards and viewpoints.

B. Implementation Details

We employ ResNet50 as the backbone to realize our proposed TTSN. We apply 8*f* (8 frames) and 16*f* (16 frames) settings which signifies we extract N frames from each video. TTSN uses these N frames in each video for modeling; we assign $N = 8$ and $N = 16$. In the pretraining stage of TTSN, conventionally, we adopt two different pretrain strategies that ImageNet solitarily or ImageNet + Kinetics combined are employed. As for the ETT module, we employ only one temporal transformer encoder submodule to lessen reasoning time and improve performance. In the TSS module, we employ "Random batch random channel-Reverse" as this algorithm performs the best and holds the strongest randomness. In succeeding ablation experiments, we also trial on other algorithms we propose, including ($\mathcal{AA}$, $\mathcal{RA}$, $\mathcal{AR}$), for the TSS module.

Since the three data sets contain distinct video quantities, we apply different training strategies and hyperparameter settings to each data set. For HMDB51 and UCF101 data sets, we assign $lr = 1.5 \times 10^{-4}$ and train our TTSN for 50 epochs with $B = 4$. For the Something–Something V1 data set, we assign $lr = 1.3 \times 10^{-3}$ and train our TTSN for 65 epochs with $B = 8$. If the verification set performance saturates, the learning rate would be divided by 10. In this article, we set *lr_steps* = [25, 35] for HMDB51 and UCF101, and *lr_steps* = [30, 45, 55] for Something–Something V1. Training and testing employ a center crop of 224 × 224 from a single clip. From HMDB51, UCF101 to Something–Something V1, the scale of the data set is increasing. The TSS module demands a stronger constraint ability to deal with a large-scale data set so that the network can learn motion information representations robustly. Therefore, for HMDB51, we empirically specify ($\theta_1 = 1.0, \theta_2 = 0.01$) in loss function Formulation (8). For UCF101 and Something–Something V1, we specify ($\theta_1 = 1.0, \theta_2 = 0.1$).

TABLE II
Comparison Between Our TTSN and Several of the State-of-the-Art Methods on HMDB51. "*" Denotes the Result Collected From Running the Code Open-Sourced by the Method's Author

Method	Pretrain	Backbone	Top1.(%)
C3D [13]	Sports-1M	ResNet18	54.9
TSN [6]	ImageNet	Inception V2	53.7
TDN* [1]	ImageNet	ResNet50	56.4
TTSN	ImageNet	ResNet50	**60.2**
ARTNet [12]	Kinetics	ResNet18	70.9
TSM [17]	Kinetics	ResNet50	73.2
R(2+1)D [64]	Kinetics	ResNet34	74.5
STM [18]	ImageNet+Kinetics	ResNet50	72.2
TEA [19]	ImageNet+Kinetics	ResNet50	73.3
I3D [65]	ImageNet+Kinetics	Inception V2	74.8
S3D [66]	ImageNet+Kinetics	Inception V2	75.9
TDN* [1]	ImageNet+Kinetics	ResNet50	79.1
TTSN	ImageNet+Kinetics	ResNet50	**80.2**

C. Comparison With the State-of-the-Art

In this section, we, respectively, report the results of our TTSN model on HMDB51, UCF101, and Something–Something V1 data sets. When experimenting, we instantiate our proposed TTSN with the backbone of ResNet50 and compare it to other state-of-the-art methods with a similar backbone. First, we thoroughly verify our proposed TTSN on the HMDB51 data set, as is shown in Table II. From the table, we can see that when our TTSN loads only the pretrained model on ImageNet, *Top*1. reaches 60.2%. To the best of our knowledge, compared with the latest 2-D CNN-based action recognition method TDN, the result improves by 3.8%. More importantly, it should be emphasized that our ImageNet pretrained model outperforms some Sport-1M pretrained 3-D

TABLE III
COMPARISON BETWEEN OUR TTSN AND SEVERAL OF THE STATE-OF-THE-ART METHODS ON UCF101. "*" DENOTES THE RESULT IS COLLECTED FROM RUNNING THE CODE OPEN-SOURCED BY THE METHOD'S AUTHOR

Method	Pretrain	Backbone	Top1.(%)
C3D [13]	ImageNet	ResNet18 V2	85.8
TSN [6]	ImageNet	Inception V2	86.4
TDN* [1]	ImageNet	ResNet50	85.8
TTSN	ImageNet	ResNet50	**86.4**
STC [67]	Kinetics	ResNet101	93.7
TSM [17]	Kinetics	ResNet50	96.0
ARTNet [12]	Kinetics	ResNet18	94.3
R(2+1)D [64]	Kinetics	ResNet34	96.8
StNet [68]	ImageNet + Kinetics	ResNet50	93.5
I3D [65]	ImageNet + Kinetics	Inception V2	95.6
S3D [66]	ImageNet + Kinetics	Inception V2	96.8
STM [18]	ImageNet + Kinetics	ResNet50	96.2
TDN* [1]	ImageNet + Kinetics	ResNet50	96.4
TTSN	ImageNet + Kinetics	ResNet50	**96.8**

CNN models (e.g., C3D [13]) in recognition accuracy. When our TTSN loads the pretrained model on Image + Kinetics combined, *Top*1. reaches 80.2%, the best performance of the model based on 2-D CNNs on HMDB-51, even if compared with the latest TDN model. Our TTSN is a 2-D CNN-based method with simple and efficient spatio-temporal modeling capability, and we introduce a temporal self-supervised algorithm that allows TTSN to maintain the computational cost of 2-D CNN-based methods (e.g., TDN [1]) and achieve the performance of 3-D CNN-based methods (e.g., I3D [65]). This further illustrates the validity of our proposed TTSN. And it is very suitable for IoT scenarios, i.e., low computational overhead and high performance.

To evaluate our proposed TTSN model more, We employ ResNet50 as the backbone and make a comparison with other existing state-of-the-art models on UCF101 data set. When TTSN only loads the pretrained model on the ImageNet data set, *Top*1. reaches 86.4%, as is shown in Table III. Furthermore, when TTSN loads the pretrained model on ImageNet + Kinetics combined, *Top*1. reaches 96.8%. From the table, we can see that the TTSN model achieves the best recognition results, outmatching some classical 3-D CNN models (e.g., C3D [13], P3D [76], I3D [65], and S3D [66]) and even several of those with the more powerful ResNet101 as their backbone (e.g., STC [67]).

To verify our TTSN thoroughly, we also conduct experiments on a larger action recognition data set Something–Something V1. As shown in Table IV, we also present the results of our TTSN model under the settings of 8*f* and 16*f*. As we can see from the table, under the 8*f* setting, *Top*1. and *Top*5. reach 52.4% and 80.6%, respectively; while under the 16*f* setting, *Top*1. and *Top*5. reach 53.3% and 81.5%. The Something–Something V1 data set holds a strong temporal correlation, making it unlikely to accurately determine categories of actions solely based on a single frame. The comparison of our TTSN with the state-of-the-art methods on the Something–Something V1 data set proves that our TTSN has a powerful temporal modeling capability and performs the best in action recognition.

TABLE IV
COMPARISON BETWEEN OUR TTSN AND SEVERAL OF THE STATE-OF-THE-ART METHODS ON SOMETHING–SOMETHING V1. "*" DENOTES THE RESULT IS COLLECTED FROM RUNNING THE CODE OPEN-SOURCED BY THE METHOD'S AUTHOR, WHILE "-" DENOTES UNAVAILABLE DATA

Method	Backbone	Frames	Top1.(%)	Top5.(%)
TSN-RGB [6]	BNInception	8	19.5	-
TRN-Multiscale [69]	BNInception	8	34.4	-
ECO_{EN}Lite [70]	BN+R18	92	46.4	-
S3D-G [66]	Inception	64	48.2	78.7
I3D [65]	ResNet50	32×2	41.6	72.2
NL I3D+GCN [71]	ResNet50+GCN	32×2	46.1	76.8
TAM [72]	bLResNet50	16×2	48.4	78.8
GST [73]	ResNet50	16	48.6	77.9
CorrNet [74]	ResNet50	32×10	49.3	-
TSM [17]	ResNet50	8+16	49.7	78.5
SmallBigNet [75]	ResNet50	8+16	50.4	80.5
TANet [26]	ResNet50	8+16	50.6	79.3
STM [18]	ResNet50	16×30	50.7	80.4
TEA [19]	ResNet50	16	51.9	80.3
TEINet [23]	ResNet50	8+16	52.5	-
TDN* [1]	ResNet50	8	49.7	78.7
TDN* [1]	ResNet50	16	52.0	80.6
TTSN	ResNet50	8	52.4	80.6
TTSN	ResNet50	16	**53.5**	**81.8**

TABLE V
COMPARISON OF MULTIPLICATION CALCULATION (GFLOPS) AMOUNT AND PARAMETER AMOUNT (PARAMS)

Module	GFLOPs	Params
ETT	6.58	2.20×10^6
TSS	1.62	12.59×10^6
TTSN (Ours)	80.2	68.79×10^6
TDN [1]	72	54×10^6
I3D [65]	306	79×10^6
NL I3D+GCN [71]	606	-

D. Ablation Studies

In this section, we present various ablation studies on the HMDB51 data set and elucidate the properties of each component of our TTSN, using input settings of $B = 4$ and 16*f*, and the pretrained model on the combined ImageNet + Kinetics data set as initialization.

Complexity Analysis: We first examine and analyze the time complexity and the number of parameters of the single module proposed in our TTSN. As for the time complexity, we examine the GFLOPs of the ETT module and the TSS module. We have also conducted a parameter analysis of these two modules. According to Table V, the ETT module and the TSS module only induce a tiny number of computational overheads. The GFLOPs of the ETT module is 6.58 under the settings of 16*f*. We recognize that the additional computational overheads of the ETT module mainly come from the temporal transformer encoder submodule. However, it has been alleviated by the frame embedding function $h(\cdot)$. The TSS module, on the other hand, produces GFLOPs 1.62 with the settings of 16*f*. The parameter quantities of the ETT module and the TSS modules are very lightweight, 2.20M and 12.59M, respectively. Furthermore, the module sizes of ETT and TSS are 18.33 and 96.12 MB. At the same time, it should

TABLE VI
RESULT OF THE ABLATION EXPERIMENT ON EFFECTIVENESS. "✓" DENOTES EMPLOYED, WHILE "-" DENOTES UNEMPLOYED

ETT	TSS	Top1.(%)
✓	-	79.3
-	✓	79.3
✓	✓	80.2

be emphasized that the TSS module is only used in the training phase of the model; and in the inference phase, there is only the ETT module. Therefore, the proposed TTSN model is very lightweight and efficient. TTSN is a CNN-based method, and we compared its GFLOPs with the classical 3-D CNN-based methods [65], [71], the results are shown in Table V, showing that the GFLOPs of TTSN (80.2) is significantly lower than [65] (306) and [71] (606), and according to the results in Tables III and IV, the performance of TTSN is significantly better than [65] and [71].

Single Module Analysis: To verify the effectiveness of the single module proposed in our TTSN, we have designed and conducted multiple sets of ablation experiments. Respectively, we conduct various ablation experiments on the ETT module and the TSS module in Table VI. We pretrain our TTSN with ImageNet + Kinetics combined data sets and employ the setting of 16f with ResNet50 as the backbone. As shown in the Table, when we employ only one of the two modules or both of them simultaneously in our TTSN, the performance of TTSN would improve under all circumstances; particularly, the network performs most desirable when employing both modules in our TTSN.

Hidden Dimension l Analysis: We conduct the ablation experiments on the hidden dimension l of temporal transformer encoder submodule to examine its effects on performance. l is the dimension of 1-D frame tensor $\mathbf{t}_i^j$, $j \in \{\alpha, \beta, \gamma\}, i \in \{1, 2, \ldots, N\}$ and it also represents the hidden dimension of temporal transformer encoder submodule. According to Fig. 3, we notice the overall performance of the network exhibit an upward tendency with the index of the hidden dimension l keeps on climbing. Our TTSN obtains the best accuracy when the hidden layer dimension l is 12 544. The hidden dimension l cannot be too small (e.g., $l = 49$) to avoid the lack of effective temporal modeling or too large to avoid overfitting, so we take $l = 12\,544$ for all experiments in this article.

Universal Analysis: We add the proposed method (ETT + TSS modules) to other baseline networks (e.g., TSM [17]) to verify the effectiveness and universality of the proposed method. We embed ETT and TSS modules into TSM [17]; as shown in Table VII, the *Top*1. performance increased by 0.5% and 0.3%, respectively. More notably, by adding both ETT and TSS modules, the TSM model is able to boost the best performance improvement, achieving a 0.8% improvement in *Top*1. recognition accuracy compared to the original model.

Self-Supervision Algorithms Analysis: To determine the final algorithm for the TSS module, we devise four distinct algorithms in total, namely, $\mathcal{AA}$, $\mathcal{AR}$, $\mathcal{RA}$, and $\mathcal{RR}$. To examine and analyze the impacts of these algorithms proposed for our TTSN, we conduct a myriad of ablation experiments, reporting

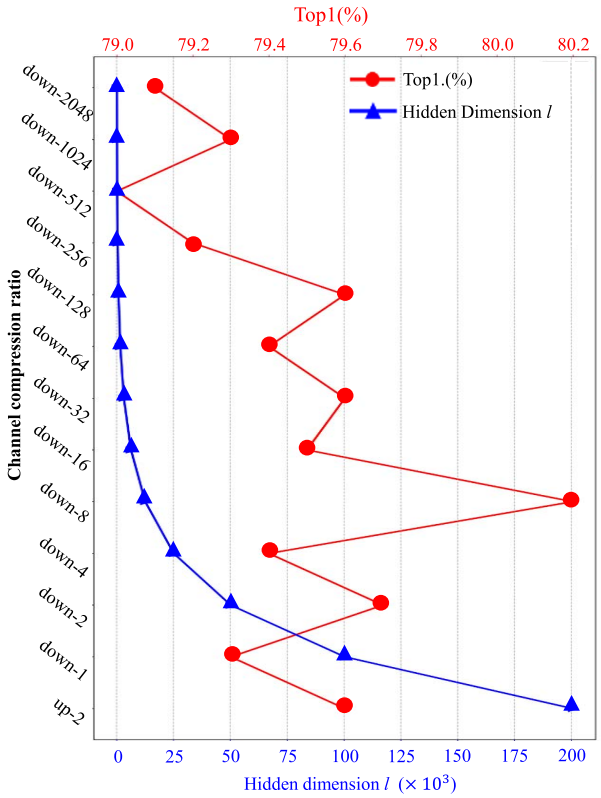

Fig. 3. Result of the ablation experiment on the hidden dimension l of the temporal transformer encoder submodule in the ETT Module. The vertical axis on the left side is **Top1.**(%), and the vertical axis on the right side is **Hidden dimension** l. The horizontal axis is the channel compression ratio of $h(\cdot)$ to $\mathbf{X}$. As the channel compression ratio increases, the hidden dimension l decreases. **down-**u and **up-**u denote compressing and expanding the number of channels of $\mathbf{X}$ by u times, respectively, and $l \propto (1/u)C$, $\mathbf{X} \in \mathbb{R}^{B\times N\times C\times H\times W}$.

TABLE VII
UNIVERSALITY ANALYSIS OF THE PROPOSED METHOD. WE USE **TSM** [17] AS OUR BASELINE. "*" DENOTES THE RESULT COLLECTED FROM RUNNING THE CODE OPEN-SOURCED BY THE METHOD'S AUTHOR

Method	Top1.(%)
TSM*	71.9
TSM + ETT	72.4
TSM + TSS	72.2
TSM + ETT + TSS	**72.7**

"TOP1. (%)" and "GPU usage," on the HMDB51 data set, which is shown in Table VIII. According to table, we notice that with the gradual increment of randomness, the TOP1. (%) performance keeps escalating; concurrently, there is a continuous drop in the GPU usage. We assume the **Randomness level** of the above four algorithms is $\widetilde{R}$, whose relationship is as follows: $\widetilde{R}_{\mathcal{AA}} < \widetilde{R}_{\mathcal{RA}} < \widetilde{R}_{\mathcal{AR}} < \widetilde{R}_{\mathcal{RR}}$. This phenomenon shows that random selection can increase the diversity of samples, thereby further improving the modeling ability of the model in the temporal dimension.

Inference Time Analysis: We report the inference time of our TTSN on RTX 3090, and specify $B = 4$, using the setting of 8f and 16f, respectively. The measuring of the inference time takes into account all evaluations, including the consumption of loading data and network inference. From Table IX, we

TABLE VIII
RESULT OF THE ABLATION EXPERIMENTS OF OUR PROPOSED SELF-SUPERVISION ALGORITHMS. THE + SYMBOL, FOR EXAMPLE, $+\mathcal{AA}$ INDICATES WE EMPLOY THE $\mathcal{AA}$ ALGORITHM IN THE TSS MODULE OF OUR TTSN

Method	Top1. (%)	GPU (MB)
ETT+$\mathcal{AA}$	79.1	13967
ETT+$\mathcal{RA}$	79.4	13967
ETT+$\mathcal{AR}$	79.5	13675
ETT+$\mathcal{RR}$	**80.2**	**13539**

TABLE IX
INFERENCE TIME ANALYSIS ON A NVIDIA RTX 3090 GPU

Method	Frames	Time (ms/video)
TSN[6]	8	7.9
TSM[17]	16	16.7
STM[18]	8	11.1
I3D[65]	32	2095
TDN[1]	8	22.1
TTSN	8	16.1
	16	29.38

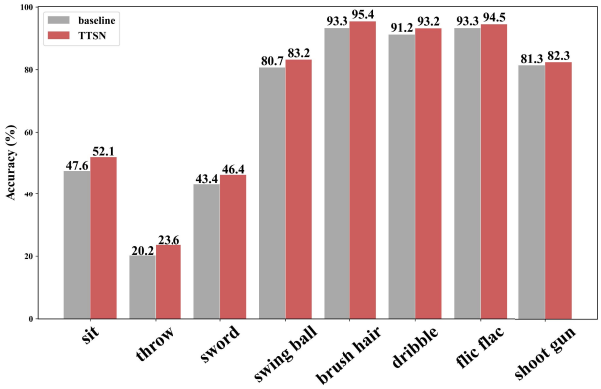

Fig. 4. Accuracy analysis of some categories on the HMDB51 data set.

can see that whether it is 8 or 16 frames, our TTSN model is able to perform very efficient real-time inference, although it is somewhat slower than some previous models.

Accuracy Analysis of Some Categories: We analyze the accuracy of each category on HMDB51 and UCF101 to find some potential regularities. We would start illustrating by comparing the accuracy of some categories on HMDB51 and UCF101, respectively. From Figs. 4 and 5, we notice that the accuracy of some of the confusing actions we have described earlier, such as "sit," "stand," and "throw," has been successfully improved. We also noticed that the accuracy of distinguishing some complex actions in the time dimension has been improved, such as "sword," "dribble," "sumo wrestling," "surfing," and "push ups." These phenomena further corroborate our proposed TTSN model, which can enhances the recognition ability of the model for complex and confusing actions by simultaneously modeling complex nonlinear relation and inverse motion information in the temporal dimension.

Confusion Matrix Analysis: We present the confusion matrix of our TTSN on HMDB51 and UCF101, respectively. For

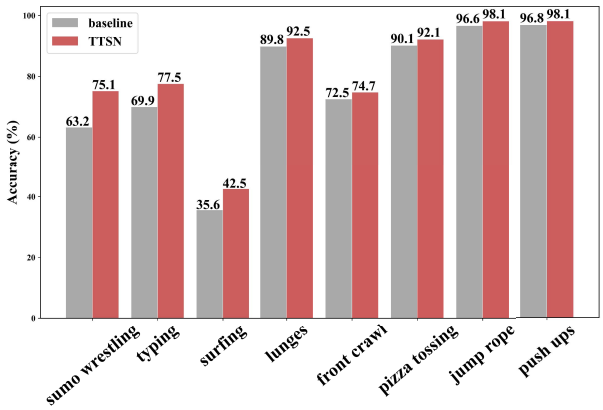

Fig. 5. Accuracy analysis of some categories on the UCF101 data set.

better visualization, we select some categories randomly here. As shown in Fig. 6, our TTSN achieves promising recognition accuracy on both data sets, especially, on the UCF101 data set where it obtains very robust classification accuracy for each category, thanks to the ability of our model to model the temporal relation of the video in a nonlinear and bidirectional manner.

Visualization Analysis: We visualize the obtained attention maps generated by the ETT module, as shown in Fig. 7. As we expected, the regions that the network pays attention to are those that undergo significant motion changes occur along the temporal dimension, that is, the motion-sensitive pixel regions that we obtained from the nonlinear temporal relation modeling. This further illustrates, from another perspective, the effectiveness of our proposed TTSN for video action recognition modeling.

Long-Term Modeling Analysis: We conducted experiments on HMDB51 and UCF101 data sets for a long-term modeling analysis to demonstrate the effectiveness of our proposed TTSN for modeling nonlocal and long-range actions. We first counted the average duration of videos corresponding to each category of action in the HMDB51 and UCF101 data sets, and analyzed the performance improvement of each category, which is shown in Figs. 8 and 9, respectively. Regardless of the length of the video, similar to TDN [1] and TSM [17] networks, our proposed TTSN network extracts 8 or 16 frames of the video as the input, and then the network models the long-terms and nonlinear relationships for that input. In spite of this, there is a correlation between the average duration of the video and the performance improvement. From the figure, we can see that it is basically consistent with the fact that longer video duration makes larger performance improvement. This fact further demonstrates that our ETT module is not only capable of modeling long-term nonlinear relationships for network input features but also has a better recognition accuracy for action categories with longer durations.

V. CONCLUSION

With the development of the IoT, more and more data is being disseminated in the form of video, which places a higher demand on video analysis and understanding. In this article,

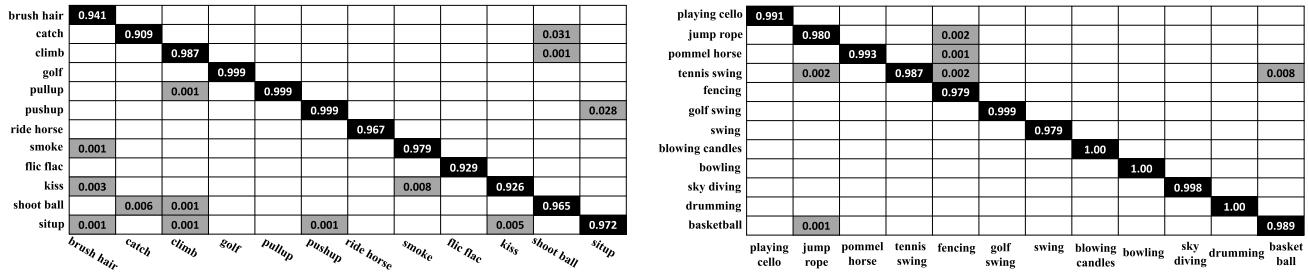

Fig. 6. Confusion matrix of our TTSN on some categories of HMDB51 and UCF101 data sets, where the number in the box indicates the confidence that the prediction is the correct result. (a) HMDB51. (b) UCF101.

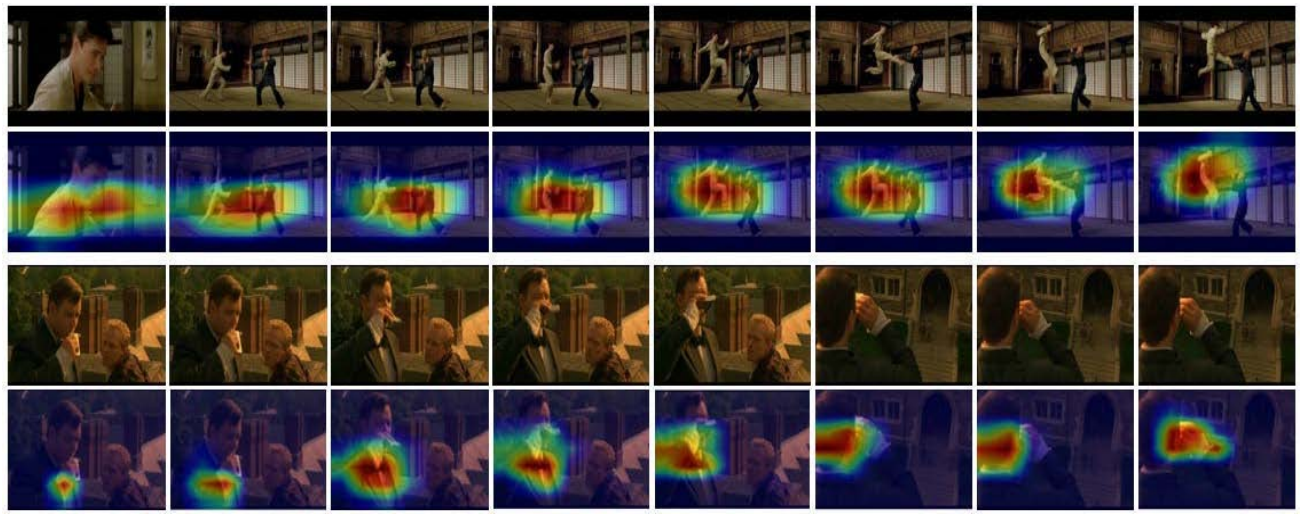

Fig. 7. Attention maps generated by the ETT Module. We employ 16*f* to visualize on the HMDB51 data set. *Odd rows: Raw RGB frames. Even rows: Raw RGB frames with our attention maps.*

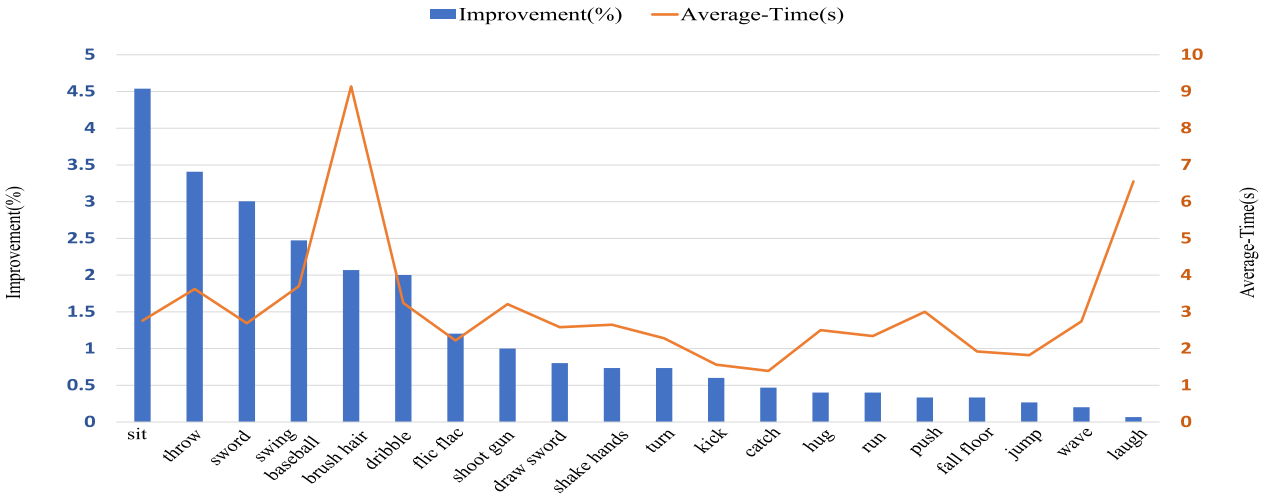

Fig. 8. Performance improvement of our TTSN model compared to the TDN [1] model in each category on HMDB51 data set (e.g., the 20 categories with the most performance improvement).

we introduce TTSN, a novel 2-D CNNs-based action recognition network. Our TTSN mainly comprises two modules: 1) an ETT module and 2) a TSS module. The ETT module is employed to efficiently model the nonlinear and nonlocal complex relation of a given temporal dimension and help the network complete the focusing process. The TSS module, on the other hand, constrains the TTSN network to learn robustly and reverse motion information representation from a given temporal dimension. Our TTSN is particularly excellent in learning actions that are inverse to one another in the temporal dimension yet similar in the spatial dimension, such as "sit" and "stand," or holding a complex nonlinear repetitive

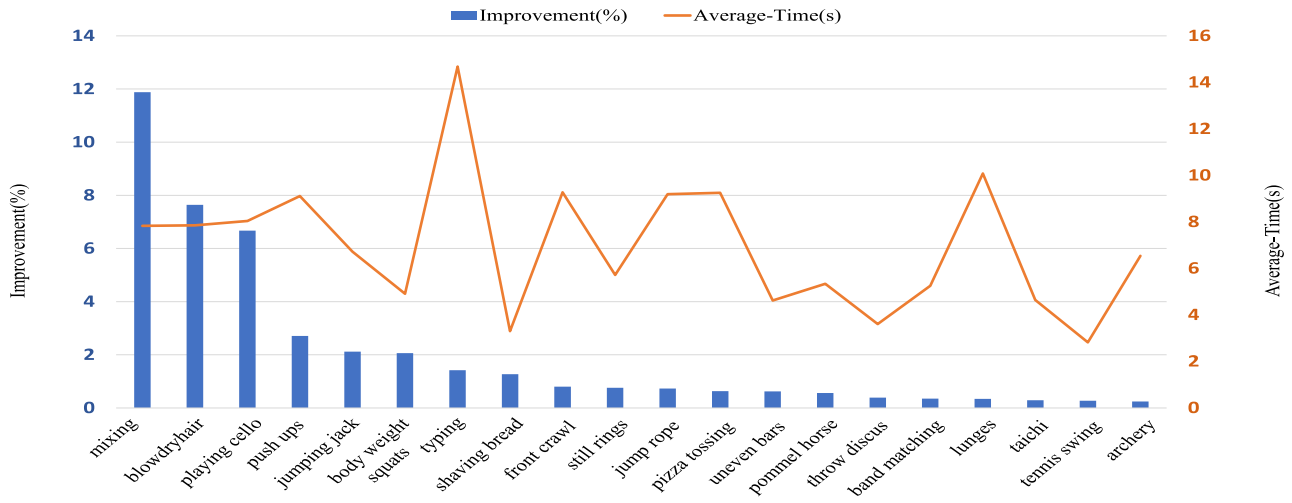

Fig. 9. Performance improvement of our TTSN model compared to the TDN [1] model in each category on the HMDB51 data set (e.g., the 20 categories with the most performance improvement).

nature, such as "brushing hair" and "flic-flac." Principally, by avoiding massive computation of optical flow-based methods and maintaining the complexity of 2-D CNNs-based methods, our TTSN is promising because it still contrives to achieve the performance of 3-D CNNs-based methods, confirmed by outmatching the state-of-the-art action recognition performance on three mainstream data sets (HMDB51, UCF101, and Something–Something V1). In summary, our work provides a lightweight and efficient video action recognition model, which can achieve almost real-time speed and can be easily deployed to video sensors in the IoT.

REFERENCES

[1] L. Wang, Z. Tong, B. Ji, and G. Wu, "TDN: Temporal difference networks for efficient action recognition," in *Proc. CVPR*, 2021, pp. 1895–1904.
[2] L. Chamain, S. Qi, and Z. Ding, "End-to-end image classification and compression with variational autoencoders," *IEEE Internet Things J.*, vol. 9, no. 21, pp. 21916–21931, Nov. 2022.
[3] B.-Y. Wu and C.-H. Lu, "Pure frequency-domain deep neural network for IoT-enabled smart cameras," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 19049–19061, Oct. 2022.
[4] A. Karpathy, G. Toderici, S. Shetty, T. Leung, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. CVPR*, 2014, pp. 1725–1732.
[5] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," 2014, *arXiv:1406.2199*.
[6] L. Wang et al., "Temporal segment networks: Towards good practices for deep action recognition," in *Proc. ECCV*, 2016, pp. 20–36.
[7] H. Wu, X. Ma, and Y. Li, "Convolutional networks with channel and STIPs attention model for action recognition in videos," *IEEE Trans. Multimedia*, vol. 22, no. 9, pp. 2293–2306, Sep. 2020.
[8] P. Zhao, L. Xie, Y. Zhang, and Q. Tian, "Universal-to-specific framework for complex action recognition," *IEEE Trans. Multimedia*, vol. 23, pp. 3441–3453, 2020.
[9] J. Wang, Y. Lin, A. J. Ma, and P. C. Yuen, "Self-supervised temporal discriminative learning for video representation learning," 2020, *arXiv:2008.02129*.
[10] J. Li, X. Liu, M. Zhang, and D. Wang, "Spatio-temporal deformable 3D ConvNets with attention for action recognition," *Pattern Recognit.*, vol. 98, Feb. 2020, Art. no. 107037.
[11] L. Kong, D. Pei, R. He, D. Huang, and Y. Wang, "Spatio-temporal player relation modeling for tactic recognition in sports videos," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 9, pp. 6086–6099, Sep. 2022.
[12] L. Wang, W. Li, W. Li, and L. Van Gool, "Appearance-and-relation networks for video classification," in *Proc. CVPR*, 2018, pp. 1430–1439.
[13] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proc. ICCV*, 2015, pp. 4489–4497.
[14] 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.
[15] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proc. CVPR*, 2015, pp. 4694–4702.
[16] H. Zhu, R. Vial, and S. Lu, "TORNADO: A spatio-temporal convolutional regression network for video action proposal," in *Proc. ICCV*, 2017, pp. 5813–5821.
[17] J. Lin, C. Gan, and S. Han, "TSM: Temporal shift module for efficient video understanding," in *Proc. ICCV*, 2019, pp. 7083–7093.
[18] B. Jiang, M. Wang, W. Gan, W. Wu, and J. Yan, "STM: Spatiotemporal and motion encoding for action recognition," in *Proc. ICCV*, 2019, pp. 2000–2009.
[19] Y. Li, B. Ji, X. Shi, J. Zhang, B. Kang, and L. Wang, "TEA: Temporal excitation and aggregation for action recognition," in *Proc. CVPR*, 2020, pp. 909–918.
[20] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman, "Video action transformer network," in *Proc. CVPR*, 2019, pp. 244–253.
[21] B. Jiang, J. Yu, L. Zhou, K. Wu, and Y. Yang, "Two-pathway transformer network for video action recognition," in *Proc. ICIP*, 2021, pp. 1089–1093.
[22] C. Plizzari, M. Cannici, and M. Matteucci, "Skeleton-based action recognition via spatial and temporal transformer networks," *Comput. Vis. Image Understand.*, vols. 208–209, Jul. 2021, Art. no. 103219.
[23] Z. Liu, D. Luo, Y. Wang, L. Wang, and T. Lu, "TEINet: Towards an efficient architecture for video recognition," in *Proc. AAAI*, vol. 34, 2020, pp. 11669–11676.
[24] J. Weng et al., "Temporal distinct representation learning for action recognition," in *Proc. ECCV*, 2020, pp. 363–378.
[25] L. Huang, Y. Yuan, J. Guo, C. Zhang, X. Chen, and J. Wang, "Interlaced sparse self-attention for semantic segmentation," 2019, *arXiv:1907.12273*.
[26] Z. Liu, L. Wang, W. Wu, C. Qian, and T. Lu, "TAM: Temporal adaptive module for video recognition," in *Proc. ICCV*, 2021, pp. 13708–13718.
[27] Z. Wang, Q. She, and A. Smolic, "Action-Net: Multipath excitation for action recognition," in *Proc. CVPR*, 2021, pp. 13214–13223.
[28] J. Hou, X. Wu, Y. Sun, and Y. Jia, "Content-attention representation by factorized action-scene network for action recognition," *IEEE Trans. Multimedia*, vol. 20, no. 6, pp. 1537–1547, Jun. 2018.
[29] K. Zhu, R. Wang, Q. Zhao, J. Cheng, and D. Tao, "A cuboid CNN model with an attention mechanism for skeleton-based action recognition," *IEEE Trans. Multimedia*, vol. 22, no. 11, pp. 2977–2989, Nov. 2020.
[30] J. Li, X. Liu, W. Zhang, M. Zhang, J. Song, and N. Sebe, "Spatio-temporal attention networks for action recognition and detection," *IEEE Trans. Multimedia*, vol. 22, no. 11, pp. 2990–3001, Nov. 2020.

[31] Y. Hao et al., "Attention in attention: Modeling context correlation for efficient video classification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 10, pp. 7120–7132, Oct. 2022.
[32] K. Zhang, Y. Li, J. Wang, E. Cambria, and X. Li, "Real-time video emotion recognition based on reinforcement learning and domain knowledge," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 3, pp. 1034–1047, Mar. 2022.
[33] W. Zhang, W. Zhao, X. Tan, L. Shao, and C. Ran, "Adaptive RF fingerprints fusion via dual attention convolutions," *IEEE Internet Things J.*, vol. 9, no. 24, pp. 25181–25195, Dec. 2022.
[34] A. Vaswani et al., "Attention is all you need," in *Proc. NeurIPS*, vol. 30, 2017, pp. 6000–6010.
[35] C. Ma, H. Sun, Y. Rao, J. Zhou, and J. Lu, "Video saliency forecasting transformer," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 10, pp. 6850–6862, Oct. 2022.
[36] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.
[37] Y. Du, C. Yuan, B. Li, L. Zhao, Y. Li, and W. Hu, "Interaction-aware spatio-temporal pyramid attention networks for action classification," in *Proc. ECCV*, 2018, pp. 373–389.
[38] J. Fu et al., "Dual attention network for scene segmentation," in *Proc. CVPR*, 2019, pp. 3146–3154.
[39] Y. Yuan, L. Huang, J. Guo, C. Zhang, X. Chen, and J. Wang, "OCNet: Object context network for scene parsing," 2018, *arXiv:1809.00916*.
[40] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. CVPR*, 2018, pp. 7794–7803.
[41] H. Ma et al., "Transfusion: Cross-view fusion with transformer for 3D human pose estimation," 2021, *arXiv:2110.09554*.
[42] D. Chen, S. Yongchareon, E. Lai, J. Yu, Q. Sheng, and Y. Li, "Transformer with bidirectional GRU for non intrusive, sensor based activity recognition in a multi resident environment," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 23716–23727, Dec. 2022.
[43] Y. Zhu, H. Shuai, G. Liu, and Q. Liu, "Self-supervised video representation learning using improved instance-wise contrastive learning and deep clustering," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 10, pp. 6741–6752, Oct. 2022.
[44] Z. Wang et al., "SSCAP: Self-supervised co-occurrence action parsing for unsupervised temporal action segmentation," in *Proc. WACV*, 2022, pp. 1819–1828.
[45] D. Kim, D. Cho, and I. Kweon, "Self-supervised video representation learning with space-time cubic puzzles," in *Proc. AAAI*, vol. 33, 2019, pp. 8545–8552.
[46] S. Gidaris, A. Bursuc, N. Komodakis, P. Pérez, and M. Cord, "Boosting few-shot visual learning with self-supervision," in *Proc. ICCV*, 2019, pp. 8059–8068.
[47] R. Zhang, P. Isola, and A. Efros, "Colorful image colorization," in *Proc. ECCV*, 2016, pp. 649–666.
[48] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. Efros, "Context encoders: Feature learning by inpainting," in *Proc. CVPR*, 2016, pp. 2536–2544.
[49] C. Ouyang, C. Biffi, C. Chen, T. Kart, H. Qiu, and D. Rueckert, "Self-supervision with superpixels: Training few-shot medical image segmentation without annotation," in *Proc. ECCV*, 2020, pp. 762–780.
[50] C. Doersch, A. Gupta, and A. Efros, "Unsupervised visual representation learning by context prediction," in *Proc. ICCV*, 2015, pp. 1422–1430.
[51] L. Dong, W. Hung, J. Huang, S. Wang, and M. Yang, "Unsupervised visual representation learning by graph-based consistent constraints," in *Proc. ECCV*, 2016, pp. 678–694.
[52] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. CVPR*, 2020, pp. 9729–9738.
[53] G. Xu, Z. Liu, X. Li, and C. Chen, "Knowledge distillation meets self-supervision," in *Proc. ECCV*, 2020, pp. 588–604.
[54] Z. Wang, J. Zhang, L. Zheng, Y. Liu, and S. Wang, "Cycas: Self-supervised cycle association for learning re-identifiable descriptions," in *Proc. ECCV*, 2020, pp. 72–88.
[55] R. Jonschkowski, A. Stone, J. Barron, A. Gordon, K. Konolige, and A. Angelova, "What matters in unsupervised optical flow," in *Proc. ECCV*, 2020, pp. 557–572.
[56] Z. Wang, X. Liu, L. Wang, Y. Qiao, X. Xie, and C. Fowlkes, "Structured triplet learning with pos-tag guided attention for visual question answering," in *Proc. WACV*, 2018, pp. 1888–1896.
[57] D. Zhao and Y. Qi, "Generative landmarks guided eyeglasses removal 3D face reconstruction," in *Proc. MMM*, 2022, pp. 109–120.
[58] T. Koizumi and W. Smith, "Look ma, no landmarks!'—Unsupervised, model-based dense face alignment," in *Proc. ECCV*, 2020, pp. 690–706.
[59] D. Zhao and Y. Qi, "Generative face parsing map guided 3D face reconstruction under occluded scenes," in *Proc. CGI*, 2021, pp. 252–263.
[60] H. Tao, C. Xie, J. Wang, and Z. Xin, "CENet: A channel-enhanced spatiotemporal network with sufficient supervision information for recognizing industrial smoke emissions," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 18749–18759, Oct. 2022.
[61] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition," in *Proc. ICCV*, 2011, pp. 2556–2563.
[62] K. Soomro, A. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," 2012, *arXiv:1212.0402*.
[63] R. Goyal et al., "The 'something something' video database for learning and evaluating visual common sense," in *Proc. ICCV*, 2017, pp. 5842–5850.
[64] T. Du, H. Wang, L. Torresani, J. Ray, and Y. Lecun, "A closer look at spatiotemporal convolutions for action recognition," in *Proc. CVPR*, 2018, pp. 6450–6459.
[65] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," in *Proc. CVPR*, 2017, pp. 6299–6308.
[66] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, "Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification," in *Proc. ECCV*, 2018, pp. 305–321.
[67] A. Diba, M. Fayyaz, V. Sharma, M. Arzani, and L. Van Gool, "Spatio-temporal channel correlation networks for action classification," in *Proc. ECCV*, 2018, pp. 284–299.
[68] D. He, "StNet: Local and global spatial-temporal modeling for action recognition," in *Proc. AAAI*, vol. 33, 2019, pp. 8401–8408.
[69] B. Zhou, A. Andonian, and A. Torralba, "Temporal relational reasoning in videos," in *Proc. ECCV*, 2018, pp. 803–818.
[70] M. Zolfaghari, K. Singh, and T. Brox, "ECO: Efficient convolutional network for online video understanding," in *Proc. ECCV*, 2018, pp. 695–712.
[71] X. Wang and A. Gupta, "Videos as space-time region graphs," in *Proc. ECCV*, 2018, pp. 399–417.
[72] Q. Fan, C. Chen, H. Kuehne, M. Pistoia, and D. Cox, "More is less: Learning efficient video representations by big-little network and depthwise temporal aggregation," in *Proc. NeurIPS*, vol. 32, 2019, pp. 1–11.
[73] C. Luo and A. Yuille, "Grouped spatial-temporal aggregation for efficient action recognition," in *Proc. ICCV*, 2019, pp. 5512–5521.
[74] H. Wang, D. Tran, L. Torresani, and M. Feiszli, "Video modeling with correlation networks," in *Proc. CVPR*, 2020, pp. 352–361.
[75] X. Li, Y. Wang, Z. Zhou, and Y. Qiao, "SmallbigNet: Integrating core and contextual views for video classification," in *Proc. CVPR*, 2020, pp. 1092–1101.
[76] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3D residual networks," in *Proc. ICCV*, 2017, pp. 5533–5541.