Pattern Recognition Letters 141 (2021) 1–7

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Learn to cycle: Time-consistent feature discovery for action recognition

Alexandros Stergiou*, Ronald Poppe

Utrecht University, Princetonplein 5, 3584 CC Utrecht, the Netherlands

ARTICLE INFO

Article history:
Received 21 June 2020
Revised 10 November 2020
Accepted 17 November 2020
Available online 18 November 2020

MSC:
68T45

Keywords:
Squeeze and recursion
Temporal gates
Temporal cyclic error
3D-CNNs
Spatio-temporal CNNs
Action recognition

ABSTRACT

Generalizing over temporal variations is a prerequisite for effective action recognition in videos. Despite significant advances in deep neural networks, it remains a challenge to focus on short-term discriminative motions in relation to the overall performance of an action. We address this challenge by allowing some flexibility in discovering relevant spatio-temporal features. We introduce Squeeze and Recursion Temporal Gates (SRTG), an approach that favors inputs with similar activations with potential temporal variations. We implement this idea with a novel CNN block that uses an LSTM to encapsulate feature dynamics, in conjunction with a temporal gate that is responsible for evaluating the consistency of the discovered dynamics and the modeled features. We show consistent improvement when using SRTG blocks, with only a minimal increase in the number of GFLOPs. On Kinetics-700, we perform on par with current state-of-the-art models, and outperform these on HACS, Moments in Time, UCF-101 and HMDB-51.[1]

1. Introduction

Action recognition in videos is an active field of research. A major challenge comes from dealing with the vast variation in the temporal display of the action [1,2]. In deep neural networks, temporal motion has primarily been modeled either with optical flow as a separate input stream [3] or using 3D convolutions [4]. The latter have shown consistent improvements in state-of-the-art models [5–7].

3D convolution kernels in convolutional neural networks (3D-CNNs) take into account fixed-sized temporal regions. Kernels in early layers have small receptive fields that primarily focus on simple patterns such as texture and linear movement. Later layers have significantly larger receptive fields that are capable of modeling complex spatio-temporal patterns. Through this hierarchical dependency, the relations between discriminative short-term motions within the larger motion patterns are only established in the very last network layers. Consequently, when training a 3D-CNN, the learned features might include incidental correlations instead of consistent temporal patterns. Thus, there appears to be room for improvement in the discovery of discriminative spatio-temporal features.

To improve this discovery process, we propose *Squeeze and Recursion Temporal Gates* (SRTG): a method that aims at extracting features that are temporally consistent. Instead of relying on a fixed-size window, our approach relates specific short-term activations to the overall motion in the video, as shown in Fig. 1. We introduce a novel block that uses an LSTM [8] to encapsulate feature dynamics, and a temporal gate to decide whether these discovered dynamics are consistent with the modeled features. The novel block can be used as at various places in a wide range of CNN architectures, with minimal computational overhead.

Our contributions are as follows:

- We implement a novel block, Squeeze and Recursion Temporal Gates (SRTG), that favors inputs that are temporally consistent with the modeled features.
- The SRTG block can be used in a wide range of 3D-CNNs, including those with residual connections, with minimal computational overhead (∼0.15% of model GFLOPs).
- We demonstrate state-of-the-art performance on five action recognition datasets when SRTG blocks are used. Networks with SRTG consistently outperform their vanilla counterparts, independent of the network depth, the convolution block type and dataset.

We discuss the advancements in the modeling of time for action recognition in Section 2. A detailed description of the main

* Corresponding author.
E-mail address: a.g.stergiou@uu.nl (A. Stergiou).
[1] The code for this project can be found at: https://git.io/JfuPi.

https://doi.org/10.1016/j.patrec.2020.11.012

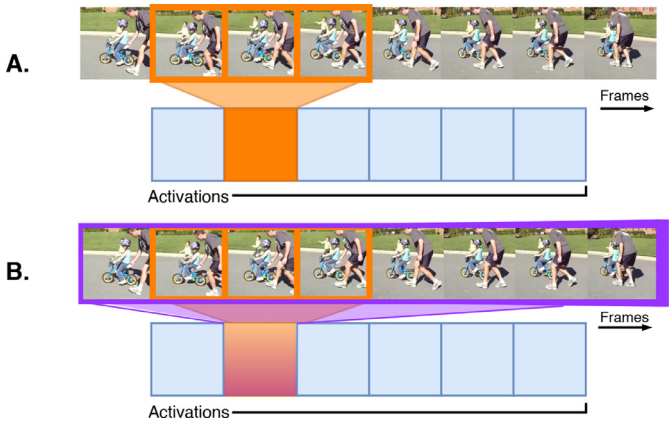

Fig. 1. A. Original 3D convolution block. Activation maps consider a fixed-size temporal window. Features are specific to the local neighborhood. B. SRTG convolution block. Activation maps take global time information into account.

methodology is provided in Section 3. Experimental setup and results are presented in Section 4 and we conclude in Section 5.

2. Related work

We discuss how temporal information is represented in CNNs, in particular using 3D convolutions.

Time representation in CNNs. Apart from the hand-coded calculation of optical flow [3], the predominant method for representing spatio-temporal information in CNNs is the use of 3D convolutions. These convolutions process motion information jointly with spatial information [4]. Because the spatial and temporal dimensions of videos are strongly connected, this has led to great improvements especially for deeper 3D-CNN models [5,9]. Recent work targets the efficient incorporation of temporal information at different time scales through the use of separate pathways [6,7].

3D convolution variants. A large body of work has focused on reducing the computational requirements of 3D convolutions. Most of these methods are targeted towards the decoupling of temporal information, for example as *pseudo* and (2+1)D 3D convolutions [10,11]. Others have proposed a decoupling of horizontal and vertical motions [12].

Information fusion of spatio-temporal activations. *Squeeze and Excitation* [13], *Gather and Excite* [14] and *Point-wise Spatial Attention* [15] consider self-attention in convolutional blocks for image-based input. In the video domain, self-attention has been implemented by [16] using clustering, to integrate local patterns with different attention units. Others have studied the use of non-local operations that capture long-range temporal dependencies through different distances [17]. Wang et al. [18] proposed to filter feature responses with activations decoupled to branches for appearance and spatial relations. Qiu et al. [19] have extended the idea of creating separate pathways for general features that can be updated through network block activations.

While these methods have shown increased generalization performance, they do not address the discovery of local spatio-temporal features across large time sequences. As activations are constrained by the spatio-temporal locality of their receptive fields, they are not allowed to effectively consider temporal variations of actions based on their general motion and time of execution. Instead of mapping the locality of features to each of the frame-wise activations, our work combines the locally-learned spatio-temporal features with their temporal variations across the duration of the video sequence.

3. Squeeze and recursion temporal gates

In this section, we introduce Squeeze and Recursion Temporal Gates (SRTG) blocks, and the possible configurations for their use in CNNs. We will denote layer input a as a stack of T frames $a_{(C\times T\times H\times W)}$ with C the number of channels, T the number of frames, and H and W the spatial dimensions of the video. The backbone blocks that SRTG are applied to also include residual connections where the final accumulated activations are the sum of the previous block activations $a^{[l-1]}$ and the current computed features $z^{[l]}$ denoted as $a^{[l]} = z^{[l]} + a^{[l-1]}$, with block index l.

3.1. Squeeze and recursion

Squeeze and Recursion blocks can be built on top of any spatio-temporal activation map $a^{[l]} = g(z^{[l]})$ for any activation function $g()$ applied to a volume of features $z^{[l]}$, shown in Fig. 2(a). This process is similar to Squeeze and Excitation [13]. For each block, the activation maps are pooled in both spatial dimensions to create a vectorized representation of the volume's features across time. Each element in the vector contains the intensity values of a frame squeezed, so to say, in a single average value. This process encapsulates the average temporal attention through the discovered features.

Recurrent cells. The importance of each feature in the temporal attention feature vector is decided by an LSTM sub-network. Through the sequential chain structure of recurrent cells, the features that are generally informative for entire video sequences can be discovered. We briefly describe the inner workings of the LSTM sub-network [8] and how the importance of each feature for the entire video is learned, as depicted in Fig. 3.

To focus on salient patterns, low intensity activations are discarded in the first operation of the recurrent cell at the *forget gate layer*. A decision $f_{(t)}$ is made given the input $pool(a^{[l]})_{(t)}$ and informative features from the previous frame $h_{(t-1)}$. The features that are to be stored are decided by the product of the sigmoidal (σ) *input gate layer* $i_{(t)}$, and the vector of candidate values $\widetilde{C}_{(t)}$ as computed as:

$$\begin{aligned} i_{(t)} &= \{\sigma(w_i * [h_{(t-1)}, pool(a^{[l]})_{(t)}] + b_i)\} \\ \widetilde{C}_{(t)} &= \{tanh(w_C * [h_{(t-1)}, pool(a^{[l]})_{(t)}] + b_C)\} \end{aligned} \quad (1)$$

The previous cell state $C_{(t-1)}$ is then updated based on the forget and input gates in order to ignore features that are not consistent across time and to determine the update weight. The new cell state $C_{(t)}$ is calculated as:

$$C_{(t)} = f_{(t)} * C_{(t-1)} + i_{(t)} * \widetilde{C}_{(t)} \quad (2)$$

The output of the recurrent cell $h_{(t)}$ is given by the current cell state $C_{(t)}$, the previous hidden state $h_{(t-1)}$ and current input $pool(a^{[l]})_{(t)}$ as:

$$\begin{aligned} h_{(t)} &= a_{(t)} * tanh(C_{(t)}), \ \ where \\ a_{(t)} &= \{\sigma(w_a * [h_{(t-1)}, pool(a^{[l]})_{(t)}] + b_a)\} \end{aligned} \quad (3)$$

The hidden states are again squeezed together to re-create a coherent sequence of filtered spatio-temporal feature intensities $a^{\star[l]}$. This new attention vector considers previous cell states, thus creating a generalized vector based on the feature intensity across time.

3.2. Temporal gates for cyclic consistency

Cyclic consistency. To evaluate the similarity between two temporal volumes, cyclic consistency has been widely used (e.g., [20,21]). The technique is based on the one-to-one mapping of frames from two time sequences, schematically summarized in

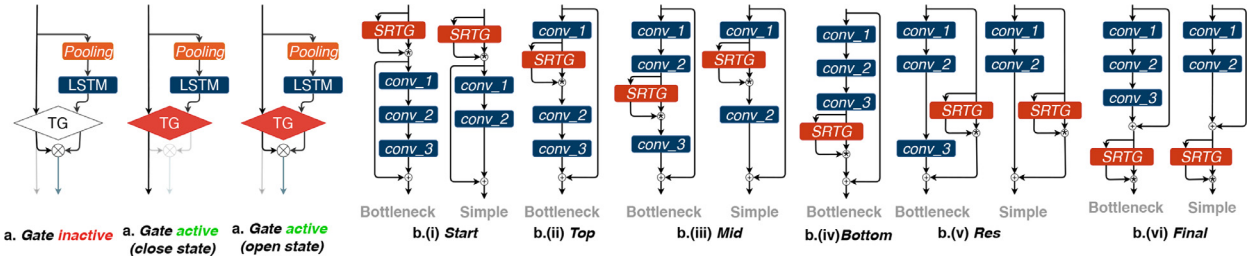

Fig. 2. (a) SRTG gate states. The gates can be inactive or active. When inactive, main stream and LSTM stream are fused. When active, the output is determined by the Temporal Gate and is either the fused result (open gate) or only the main stream (close state). (b) SRTG configuration options described in Section 3.3. Similar to Residual Networks, we distinguish between *Simple* blocks with two conv operations and *Bottleneck* blocks with three conv operations.

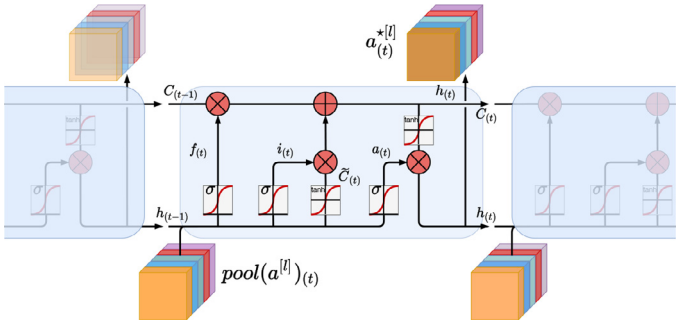

Fig. 3. Overview of the LSTM-chained cells used for the discovery of globally informative local features. Each input corresponds to a temporal activation map and produces a feature vector of the same size as the input.

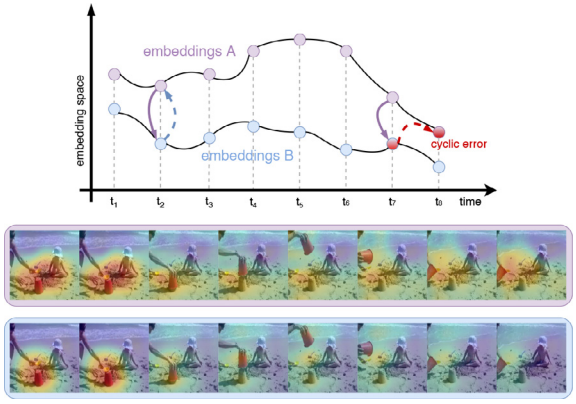

Fig. 4. Temporal Cyclic Error. Soft nearest neighbor is used to match points between two embeddings. Cycle-consistent points cycle back to original points (visualized for t_2). Otherwise, a temporal cyclic error occurs (e.g. at t_7). Corresponding salient areas below are visualized with *CFP* [22].

Fig. 4. Each of the two feature spaces can be considered an *embedding space*. Two embedding spaces are cycle-consistent if and only if, each point at time t in the embedding space $\boldsymbol{A}$, has a minimum distance point in embedding space $\boldsymbol{B}$ that is also at time t. Equivalently, each point at time t in embedding space $\boldsymbol{B}$ should also have a minimum distance point in embedding space $\boldsymbol{A}$ at time t. As shown in Fig. 4, when points do not cycle back to the same temporal location, they do not exhibit cyclic consistency. In this case, a temporal cyclic error occurs.

By having points that can cycle back to themselves, a similarity baseline between embedding spaces can be established. Although individual features of the two spaces may be different, they should demonstrate an overall similarity when their alignment in terms of cyclic consistency is the same. Therefore, comparing volumes by their cyclic consistency is a suitable measure to account for (temporal) variations.

Soft nearest neighbor distance. The main challenge in creating a coherent similarity measure between two embeddings is to deal with the vast embedding spaces, as well as to discover the "nearest" point in an adjacent embedding. The idea of *soft matches* for projected points in embeddings [23] is based on finding the closest point in an embedding space through the weighted sum of all possible matches and then selecting the closest actual observation.

To find the soft nearest neighbor of an activation $a^A_{(t)}$ in embedding space $\boldsymbol{B}$, the euclidean distances between observation $a^B_{(t)}$ and all points in $\boldsymbol{B}$ are calculated (see Fig. 5). Each frame is considered a separate instance for which we want to find the minimum point in the adjacent embedding space. We weight the similarity of each frame in embedding space $\boldsymbol{B}$ to activation $a^A_{(t)}$ using a *softmax* activation and by exploiting the exponential difference between activation pairs:

$$\tilde{a}^{(B\to A)}_{(t)} = \sum_{i}^{T} z_{(i)} * a^B_{(i)}, \ \ where\ z_{(i)} = \frac{e^{-||a^A_{(t)} - a^B_{(i)}||^2}}{\sum_i^T e^{-||a^A_{(t)} - a^B_{(i)}||^2}} \tag{4}$$

The *softmax* activation produces a normal distribution of similarities $\mathcal{N}(\mu, \sigma^2)$, centered on the frame with the minimum distance from activation $a^A_{(t)}$. Based on the discovery of the nearest neighbor $\tilde{a}^{(B\to A)}_{(t)}$, the distance to nearest frames in $\boldsymbol{B}$ can then be computed. This enables the discovery of frames that are closely related to the initially considered frame $a^A_{(t)}$, achieved by minimizing the L2 distance from the found soft match:

$$a^{(B\to A)}_{(t)} = \underset{i}{\text{argmin}}(||\tilde{a}^{(B\to A)}_{(t)} - a^B_{(i)}||^2) \tag{5}$$

We define a point as *consistent* if and only if the initial temporal location t matches precisely the temporal location of the computed point in embedding space $\boldsymbol{B}$, $a^{(B\to A)}_{(t)} = a^B_{(t)}\ \forall t \in \{1, \ldots, T\}$. To establish a consistency check for frames in embedding space $\boldsymbol{A}$, the same procedure is repeated in reverse for every frame in embedding space $\boldsymbol{B}$, calculating the soft nearest neighbor in embedding space $\boldsymbol{A}$. The two embeddings are considered *cycle-consistent* if and only if all points on both embedding spaces map back to themselves through the other embedding space: $a^{(B\to A)}_{(t)} = a^B_{(t)}$ *and* $a^{(A\to B)}_{(t)} = a^A_{(t)}\ \forall t \in \{1, \ldots, T\}$.

Temporal gates. The temporal activation vector encapsulates average feature attention over time. However, it does not enforce a precise similarity to the local spatio-temporal activations. Thus, we compute cyclic consistency between the pooled activations $pool(a^{[l]})$ and the outputted recurrent cells $a^{\star[l]}$. In this context, cyclic consistency is used as a gating mechanism to fuse the recurrent cell hidden states with unpooled versions of the activations when the two volumes are temporally cycle-consistent. This ensures that only time-consistent information is added back to the network, as shown in Fig. 2(a).

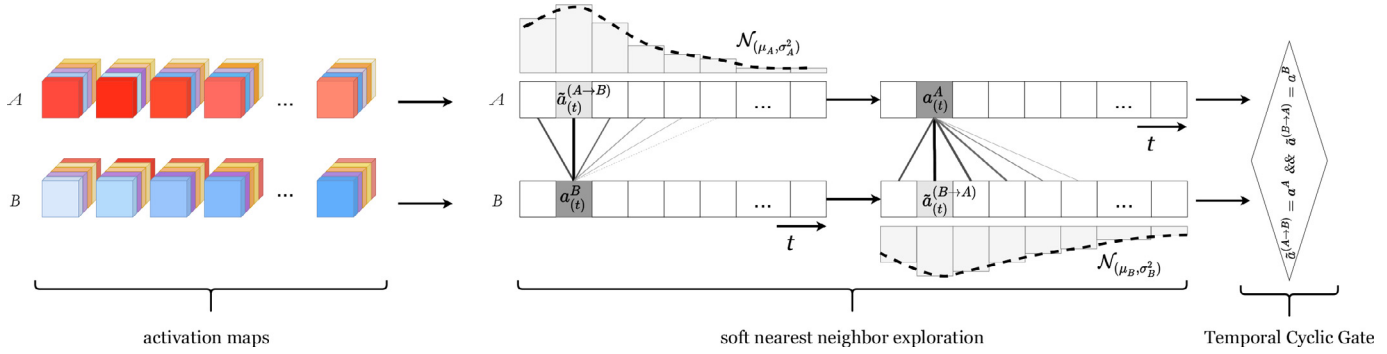

Fig. 5. Temporal Gates. Activations of each frame ($a^B_{(t_i)}$) in embedding space **B** are compared to the activations of every frame ($a^A_{(t_j)}$) in embedding space **A**. We calculate for each frame-wise activation map ($a^B_{(t)}$) the corresponding soft nearest neighbor ($\tilde{a}^{A\to B}_{(t)}$) in encoding space **A**. We then equivalently obtain $\tilde{a}^{B\to A}_{(t)}$ in encoding space **B**. The gate is open when $\tilde{a}^{A\to B}$ and $\tilde{a}^{B\to A}$ are exactly and sequentially equal to a^A and a^B.

3.3. SRTG block variants

Cyclic consistency can be considered in different parts of a convolution block, and we investigate six different approaches in terms of constructing a SRTG block. In each case, the principle of global and local information fusion remains. The block configurations only differ in the relative locations of the SRTG and the LSTM input. All configurations are shown in Fig. 2(b). Similar to networks with residual connections, we consider Simple blocks with two conv operations and Bottleneck blocks with three conv operations. Not all SRTG configurations apply to the Simple blocks.

Start. SRTG is the first operation ensuring that operations will be based on both global and local information. This is used in both Simple and Bottleneck residual blocks.

Top. Activations of the first convolution are used by the LSTM, with fused features being used by the final convolution. This is specific to Bottleneck blocks.

Mid. SRTG is added at the middle of Simple blocks and after the second convolution at Bottleneck blocks.

End. Local and global features are fused at the end of the final convolution, before the concatenation of the residual connection. This is only used in Bottleneck blocks.

Res. The SRTG block is applied to the residual connection. This transforms the residual connection to further include global spatio-temporal features combining them with convolutional activations from either Simple or Bottleneck blocks.

Final. SRTG is added at the end of the residual block, which allows for the activations to be calculated jointly with their representations across time on the entire video. This can be used in both Simple and Bottleneck blocks.

4. Experiments and results

We evaluate our approach on five action recognition benchmark datasets (Section 4.1). We perform experiments with several ResNet backbones with various depths. Each network uses either 3D convolutions (*r3d*) or (2+1)D convolutions (*r(2+1)d*).

4.1. Datasets

We use five action recognition datasets for our experiments:

Human Action Clips and Segments (HACS, [24]) includes approximately 500K clips of 200 classes. Clips are 60-frame segments extracted from 50k unique videos.

Kinetics-700 (K-700, [25]) is the extension of Kinetics-400/600 to 700 classes. It contains approximately 600k clips of varying duration.

Moments in Time (MiT, [26]) is one of the largest video datasets of human actions and activities. It includes 339 classes with approximately 800K, 3-second clips.

UCF-101 [27] includes 101 classes and 13k clips that vary between 2 and 14 s in duration.

HMDB-51 [28] contains 7K clips divided over 51 classes with at least 101 clips per class.

4.2. Experimental settings

Training was performed with a random sub-sampling of 16 frames, resized to 224×224. We adopted a multigrid training scheme [29] with an initial learning rate of 0.1, halved at each cycle. We used a SGD optimizer with $1e^{-6}$ weight decay and a step-wise learning rate reduction. All tested SRTG blocks incorporate stacked dual LSTMs (2 layers). For HACS, K-700 and MiT, we use the train/test splits suggested by the authors, and report on split1 for UCF-101 and HMDB-51.

4.3. Comparison of SRTG block configurations

We compare the different SRTG block configurations with a 34-layer r3d and r(2+1)d. ResNets-34 contain Simple blocks with two conv layers instead of the Bottleneck blocks with three conv layers. We therefore only evaluate the Start, Mid, Res and Final configurations. Results, summarized in Table 2, are obtained on HACS by training from scratch. All SRTG blocks perform better than their vanilla counterparts. This demonstrates the merits of our more flexible treatment of the temporal dimension. This effect appears to be stronger when the filtering is applied later. Indeed, the best performing SRTG configuration *Final* achieves a top-1 accuracy improvement of 3.781% for 3D and 4.686% for (2+1)D convolution blocks.

4.4. Comparison of network architectures

To better understand the merits of our method, we compare a number of network architectures with and without SRTG (Final configuration). We summarize the performance on all five benchmark datasets in Table 1. The top part of the table contains the results for state-of-the-art networks including I3D [5] which is based on an Inception-v1 network. The remaining evaluated architectures use Resnet backbones. Temporal Shift Module (TSM, [30]) and Multi-Fiber networks (MF, [6]) use a r3d-50 backbone and Channel-Separated Convolutions (ir-CSN, [31]) and SlowFast networks (SF, [7]) are based on r3d-101 backbones. We further include a 50-layer SlowFast network for an additional comparison of lower-capacity models. We have used the trained networks from the respective authors' repositories. These trained models are typically

Table 1
Action recognition accuracy for all five benchmark datasets with the model's average inference times per clip. Top part of the table includes state-of-the-art models, evaluated from trained models provided by the respective authors. Middle and bottom parts summarize the results for r3/(2+1)d with/without SRTG.

Model	Inf. (msec.)	HACS		Kinetics-700		Moments in Time		UCF-101		HMDB-51	
	(↓F / ↑B)	top-1(%)	top-5(%)	top-1(%)	top-5(%)	top-1(%)	top-5(%)	top-1(%)	top-5(%)	top-1(%)	top-5(%)
I3D	21.3/80.0	79.948	94.482	53.015	69.193	28.143	54.570	92.453	97.619	71.768	94.128
TSM	38.6/143.4	N/A	N/A	54.032	72.216	N/A	N/A	92.336	97.961	72.391	94.158
ir-CSN-101	51.4/461.2	N/A	N/A	54.665	73.784	N/A	N/A	94.708	98.681	73.554	95.394
MF-Net	32.8/236.0	N/A	N/A	54.249	73.378	27.286	48.237	93.863	98.372	72.654	94.896
SF r3d-50	26.9/84.0	N/A	N/A	56.167	75.569	N/A	N/A	94.619	98.756	73.291	95.410
SF r3d-101	39.3/125.1	N/A	N/A	**57.326**	77.194	N/A	N/A	95.756	99.138	74.205	95.974
r3d-34	32.7/74.1	74.818	92.839	46.138	67.108	24.876	50.104	89.405	96.883	69.583	91.833
r3d-50	28.2/87.7	78.361	93.763	49.083	72.541	28.165	53.492	93.126	96.293	72.192	94.562
r3d-101	41.6/110.2	80.492	95.179	52.583	74.631	31.466	57.382	95.756	98.423	75.650	95.917
r(2+1)d-34	40.8/152.0	75.703	93.571	46.625	68.229	25.614	52.731	88.956	96.972	69.205	90.750
r(2+1)d-50	33.2/128.7	81.340	94.514	49.927	73.396	29.359	55.241	93.923	97.843	73.056	94.381
r(2+1)d-101	49.9/163.6	82.957	95.683	52.536	75.177	32.213	57.748	95.503	98.705	75.837	95.512
SRTG r3d-34	35.2/80.6	78.599	93.569	49.153	72.682	28.549	52.347	94.799	98.064	74.319	94.784
SRTG r3d-50	31.8/96.9	80.362	95.548	53.522	74.171	30.717	55.650	95.756	98.550	75.650	95.674
SRTG r3d-101	49.2/131.6	81.659	96.326	56.462	76.819	33.564	58.491	**97.325**	**99.557**	**77.536**	**96.253**
SRTG r(2+1)d-34	46.3/157.0	80.389	94.267	49.427	73.233	28.972	54.176	94.149	97.814	72.861	92.667
SRTG r(2+1)d-50	37.6/141.5	83.774	96.560	54.174	74.620	31.603	56.796	95.675	98.842	75.297	95.141
SRTG r(2+1)d-101	58.9/172.2	**84.326**	**96.852**	56.826	**77.439**	**33.723**	**59.114**	97.281	99.160	77.036	95.985

Table 2
Comparison of r3d-34 with SRTG configurations on HACS.

Config	Gates	top-1 (%)		top-5 (%)	
		3D	(2+1)D	3D	(2+1)D
No SRTG	✗	74.818	75.703	92.839	93.571
Start	√	75.705	76.438	93.230	93.781
Mid	√	75.489	76.685	93.224	93.746
Res	√	76.703	77.094	93.307	93.856
Final	√	**78.599**	**80.389**	**93.569**	**94.267**

pre-trained on other datasets. Missing values are due to the lack of a trained model. Any deviations from previously reported performances are due to the use of multigrid [29] with a base cycle batch size of 32.

The second and third parts of Table 1 summarize the performances of ResNets with various depths and 3D or (2+1)D convolutions, with and without SRTG, respectively. Models for HACS are trained from scratch. The weights of models for K-700 and MiT are initialized based on those from the pre-trained HACS model. For UCF-101 and HMDB-51, we fine-tune the pre-trained models on HACS and K700.

For the state-of-the-art architectures, larger and deeper models improve accuracy. This is in line with the general trend. Models implemented with (2+1)D convolution blocks perform slightly better than their counterparts with 3D convolutions but these differences are modest and not consistent across datasets.

As shown in Table 1, adding SRTG blocks to any architecture consistently improves performance. Table 3 shows pairwise comparisons of the performance on the three largest benchmark datasets for networks with and without SRTG. When using SRTG blocks, the improvements are in the range of 1.2–4.7% for HACS, 2.8–4.4% for K-700 and 2.1–3.7% for MiT. For smaller networks, we observe larger gains. The use of time-consistent features obtained through our method appears to improve the generalization ability of 3D-CNNs.

The r3d and r(2+1)d networks with SRTG perform at least on-par with the current state-of-the-art architectures. The r3d-101 outperforms current state-of-the-art in HACS, MiT, UCF-101 and HMDB-51. For MiT, our top-1 accuracy of 33.564% largely surpasses other tested architectures. The (2+1)D variant further outperforms current architectures on HACS with 84.326% top-1 accuracy. Its performance on Kinectics-700 is comparable to the best performing SlowFast r3d-101 model. While SlowFast achieves better top-1 accuracy, a r(2+1)d-101 network with SRTG blocks has higher top-5 accuracy. This similar performance is remarkable given the relatively low complexity of the SRTG r3d-101 and r(2+1)d-101 models. SlowFast is built on a dual-network configuration with two subparts responsible for long-term and short-term movements. SlowFast therefore includes a significantly larger number of operations than our tested networks with SRTG blocks. We analyze the computation cost of the SRTG block in Section 4.5.

Finally, we observe that the performance gain with SRTG is substantial for the two smaller datasets, UCF-101 and HMDB-51. The already almost saturated performance of the ResNet-101 models on UCF-101 increases with 1.569% and 1.778% for the 3D and (2+1)D convolution variants, respectively. This further demonstrates that SRTG can improve the selection of features that contain less noise and generalize better, even when there is fewer training data available.

4.5. Analysis of computational overhead

The SRTG block can be added to a large range of 3D-CNN architectures. It leverages the small computational costs of LSTMs compared to 3D convolutions. That enables us to increase the number of parameters without a significant increase in the number of GFLOPs. This also corresponds to the small additional memory usage compared to baseline models on both forward and backward passes. We present the number of multi-accumulative operations (MACs)[2] used for the r3/(2+1)d architectures with and without SRTG in Fig. 6, with respect to the corresponding accuracies. The additional computation overhead, for models that include the proposed block, is approximately 0.15% of the total number of operations in the vanilla networks. This constitutes a negligible increase, compared to the performance gains, making SRTG a lightweight block that can be easily used on top of existing networks. The low computational impact is further evident in Table 1, where SRTG models have maximum added latency times over the baseline counterparts of +9.0 milliseconds for forward passes and +21.4 milliseconds for gradient calculations. Average added latency

[2] Multi-accumulative operations [32] are based on the product of two numbers increased by an accumulator. They relate to the accumulated sum of convolutions between the dot product of the weights and input region.

Table 3
Pairwise comparisons of r3d and r(2+1)d networks with and without SRTG on HACS, K-700 and MiT.

Dataset	r3d-50		r(2+1)d-50		r3d-101	
	None	SRTG	None	SRTG	None	SRTG
HACS	78.361	80.362 (**+2.0**)	81.340	83.474 (**+2.1**)	80.492	81.659 (**+1.1**)
K-700	49.083	53.522 (**+4.4**)	49.927	54.174 (**+4.2**)	52.583	56.462 (**+3.8**)
MiT	28.165	30.717 (**+2.5**)	29.359	31.603 (**+3.3**)	31.466	33.564 (**+2.0**)

(a) HACS (r3d) (b) HACS (r(2+1)d) (c) K-700 (r3d) (d) K-700 (r(2+1)d) (e) MiT (r3d)

Fig. 6. Accuracy in relation to computation cost. Top-1 accuracy and operations (in GMACs) of r3/r(2+1)d with/without SRTG on HACS, K-700 and MiT.

Table 4
Results on UCF-101 and HMDB-51 based on transfer learning.

Model	Pre-training	GFLOPs	UCF-101 top-1 (%)	HMDB-51 top-1 (%)
SRTG r3d-34	HACS	110.48	94.799	74.319
	HACS+K-700		95.842	74.183
	HACS+MiT		95.166	74.235
SRTG	HACS	110.8	94.149	72.861
r(2+1)d-34	HACS+K-700		94.569	73.217
	HACS+MiT		95.648	74.473
SRTG r3d-50	HACS	150.98	95.756	75.650
	HACS+K-700		96.853	75.972
	HACS+MiT		96.533	76.014
SRTG	HACS	151.6	95.675	75.297
r(2+1)d-50	HACS+K-700		95.993	75.743
	HACS+MiT		96.278	75.988

times for r3d and r(2+1)d networks are +5.4 ms and +10.5 ms for forward and backward respectively.

4.6. Evaluating feature transferability

A common practice to train CNNs is to use transfer learning on a pre-trained network. To evaluate the performance of the SRTG block after transfer learning, we pre-train on several datasets and fine-tune on smaller datasets UCF-101 and HMDB-51. Through this experiment, we can further eliminate biases relating to the pre-training datsets and compare the accuracies achieved with respect to the SRTG blocks.

As shown in Table 4, the accuracy rates remain fairly consistent for the pre-training datasets. This consistency is due to the large sizes of these datasets, as well as the overall robustness of the proposed method. The average offset between each of the pre-trained models is 0.71% for UCF-101 and 0.47% for HMDB-51. These are only minor changes in accuracy, which further demonstrates that the improvements observed are due to the inclusion of SRTG blocks in the network.

5. Conclusions

We have introduced a novel Squeeze and Recursion Temporal Gates (SRTG) block that can be added to a large range of CNN architectures to create time-consistent features. The SRTG block uses an LSTM to capture multi-frame feature dynamics, and a temporal gate to evaluate the cyclic consistency between the discovered dynamics and the modeled features. SRTG blocks add a negligible computational overhead (0.03–0.4 GFLOPs), which makes both forward and backward passes efficient. Adding our proposed SRTG blocks in ResNet backbones with 3D or (2+1)D convolutions consistently leads to performance gains. Our results are on par with, and in most cases outperform, the current state-of-the-art on action recognition datasets including Kinetics-700 and Moments in Time. For HACS, we obtain a state-of-the-art-performance of 84.3%. Our combined experiments demonstrate the generalization ability of the discovered time-consistent features. Using sophisticated sampling techniques [33,34], we plan to overcome the limitation of requiring segmented clips. This would allow for our work to be used for the detection of actions in long videos involving human behavior.

Declaration of Competing Interest

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

Acknowledgments

This publication is supported by the Netherlands Organization for Scientific Research (NWO) with a TOP-C2 grant for “Automatic recognition of bodily interactions” (ARBITER).

References

[1] S. Herath, M. Harandi, F. Porikli, Going deeper into action recognition: a survey, Image Vis. Comput. 60 (2017) 4–21.
[2] A. Stergiou, R. Poppe, Analyzing human-human interactions: a survey, Comput. Vis. Image Underst. 188 (2019) 102799.

[3] K. Simonyan, A. Zisserman, Two-stream convolutional networks for action recognition in videos, in: Advances in Neural Information Processing Systems (NIPS), 2014, pp. 568–576.
[4] S. Ji, W. Xu, M. Yang, K. Yu, 3D Convolutional neural networks for human action recognition, Trans. Pattern Anal. Mach.Intell. 35 (1) (2013) 221–231.
[5] J. Carreira, A. Zisserman, Quo vadis, action recognition? A new model and the Kinetics dataset, in: Computer Vision and Pattern Recognition (CVPR), IEEE, 2017, pp. 4724–4733.
[6] Y. Chen, Y. Kalantidis, J. Li, S. Yan, J. Feng, Multi-fiber networks for video recognition, in: European Conference on Computer Vision (ECCV), 2018, pp. 352–367.
[7] C. Feichtenhofer, H. Fan, J. Malik, K. He, SlowFast networks for video recognition, in: International Conference on Computer Vision (ICCV), IEEE, 2019, pp. 6202–6211.
[8] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[9] K. Hara, H. Kataoka, Y. Satoh, Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet? in: Computer Vision and Pattern Recognition (CVPR), IEEE, 2018, pp. 18–22.
[10] Z. Qiu, T. Yao, T. Mei, Learning spatio-temporal representation with pseudo-3D residual networks, in: International Conference on Computer Vision (ICCV), IEEE, 2017, pp. 5534–5542.
[11] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, M. Paluri, A closer look at spatiotemporal convolutions for action recognition, in: Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2018, pp. 6450–6459.
[12] A. Stergiou, R. Poppe, Spatio-temporal FAST 3D convolutions for human action recognition, in: International Conference on Machine Learning Applications (ICMLA), IEEE, 2019, pp. 1830–1834.
[13] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2018, pp. 7132–7141.
[14] J. Hu, L. Shen, S. Albanie, G. Sun, A. Vedaldi, Gather-excite: exploiting feature context in convolutional neural networks, in: Advances in Neural Information Processing Systems (NIPS), 2018, pp. 9401–9411.
[15] H. Zhao, Y. Zhang, S. Liu, J. Shi, C. Change Loy, D. Lin, J. Jia, PSANet: Point–wise spatial attention network for scene parsing, in: European Conference on Computer Vision (ECCV), 2018, pp. 267–283.
[16] X. Long, C. Gan, G. De Melo, J. Wu, X. Liu, S. Wen, Attention clusters: purely attention based local feature integration for video classification, in: Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2018, pp. 7834–7843.
[17] X. Wang, R. Girshick, A. Gupta, K. He, Non-local neural networks, in: Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2018, pp. 7794–7803.
[18] L. Wang, W. Li, W. Li, L. Van Gool, Appearance-and-relation networks for video classification, in: Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2018, pp. 1430–1439.
[19] Z. Qiu, T. Yao, C.-W. Ngo, X. Tian, T. Mei, Learning spatio-temporal representation with local and global diffusion, in: Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2019, pp. 12056–12065.
[20] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, A. Zisserman, Temporal cycle–consistency learning, in: Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2019, pp. 1801–1810.
[21] X. Wang, A. Jabri, A.A. Efros, Learning correspondence from the cycle-consistency of time, in: Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2019, pp. 2566–2576.
[22] A. Stergiou, G. Kapidis, G. Kalliatakis, C. Chrysoulas, R. Poppe, R. Veltkamp, Class feature pyramids for video explanation, in: International Conference on Computer Vision Workshop (ICCVW), IEEE, 2019, pp. 4255–4264.
[23] J. Goldberger, G.E. Hinton, S.T. Roweis, R.R. Salakhutdinov, Neighbourhood components analysis, in: Advances in Neural Information Processing Systems (NIPS), 2005, pp. 513–520.
[24] H. Zhao, A. Torralba, L. Torresani, Z. Yan, HACS: Human action clips and segments dataset for recognition and temporal localization, in: International Conference on Computer Vision (ICCV), IEEE, 2019, pp. 8668–8678.
[25] J. Carreira, E. Noland, C. Hillier, A. Zisserman, A short note on the Kinetics-700 human action dataset, arXiv:1907.06987 (2019).
[26] M. Monfort, A. Andonian, B. Zhou, K. Ramakrishnan, S.A. Bargal, T. Yan, L. Brown, Q. Fan, D. Gutfreund, C. Vondrick, et al., Moments in time dataset: one million videos for event understanding, IEEE Trans. Pattern Anal. Mach. Intell. 42 (2) (2019) 502–508.
[27] K. Soomro, A.R. Zamir, M. Shah, UCF101: A dataset of 101 human actions classes from videos in the wild, arXiv:1212.0402 (2012).
[28] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, T. Serre, HMDB: A large video database for human motion recognition, in: International Conference on Computer Vision (ICCV), IEEE, 2011, pp. 2556–2563.
[29] C.-Y. Wu, R. Girshick, K. He, C. Feichtenhofer, P. Krähenbühl, A multigrid method for efficiently training video models, in: Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2020, pp. 153–162.
[30] J. Lin, C. Gan, S. Han, TSM: Temporal shift module for efficient video understanding, in: International Conference on Computer Vision (ICCV), IEEE, 2019, pp. 7083–7093.
[31] D. Tran, H. Wang, L. Torresani, M. Feiszli, Video classification with channel-separated convolutional networks, in: International Conference on Computer Vision (ICCV), IEEE, 2019, pp. 5552–5561.
[32] P.E. Ludgate, On a proposed analytical machine, in: The Origins of Digital Computers, Springer, 1982, pp. 73–87.
[33] B. Korbar, D. Tran, L. Torresani, SCSampler: sampling salient clips from video for efficient action recognition, in: International Conference on Computer Vision (ICCV), IEEE, 2019, pp. 6231–6241.
[34] Z. Wu, C. Xiong, C.-Y. Ma, R. Socher, L.S. Davis, AdaFrame: adaptive frame selection for fast video recognition, in: Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2019, pp. 1278–1287.