SURVEY AND STATE OF THE ART

An empirical study on temporal modeling for online action detection

Wen Wang · Xiaojiang Peng · Yu Qiao · Jian Cheng

Received: 10 April 2021 / Accepted: 5 September 2021 / Published online: 13 November 2021 © The Author(s) 2021

Abstract

Online action detection (OAD) is a practical yet challenging task, which has attracted increasing attention in recent years. A typical OAD system mainly consists of three modules: a frame-level feature extractor which is usually based on pre-trained deep Convolutional Neural Networks (CNNs), a temporal modeling module, and an action classifier. Among them, the temporal modeling module is crucial which aggregates discriminative information from historical and current features. Though many temporal modeling methods have been developed for OAD and other topics, their effects are lack of investigation on OAD fairly. This paper aims to provide an empirical study on temporal modeling for OAD including four meta types of temporal modeling methods, i.e. temporal pooling, temporal convolution, recurrent neural networks, and temporal attention, and uncover some good practices to produce a state-of-the-art OAD system. Many of them are explored in OAD for the first time, and extensively evaluated with various hyper parameters. Furthermore, based on our empirical study, we present several hybrid temporal modeling methods. Our best networks, i.e., the hybridization of DCC, LSTM and M-NL, and the hybridization of DCC and M-NL, which outperform previously published results with sizable margins on THUMOS-14 dataset (48.6% vs. 47.2%) and TVSeries dataset (84.3% vs. 83.7%).

Keywords Online action detection · Temporal pooling · Temporal convolution · Recurrent neural network · Temporal attention

Introduction

Online action detection (OAD) is an important problem in computer vision, which has a wide range of applications like visual surveillance, human–computer interaction, and intelligent robot navigation, etc. Different from traditional action recognition and offline action detection that intend to recognize actions from fully observed videos, the goal of online action detection is to detect an action as it happens and ideally even before the action is fully completed. It is a very challenging problem due to the extra restriction on the usage of only historical and current information except for the difficulties of traditional action recognition in untrimmed video streams.

In general, there exist two OAD tasks, i.e. spatial-temporal online action detection (ST OAD) and temporal online action detection. With online setting, the former aims to localize actors and recognize actions in space-time which is introduced in [62], while the latter is to localize and recognize actions temporally only which is systematically introduced in [10]. Our study mainly focuses on the temporal online action detection problem, and we ignore ‘temporal’ for convenience in the rest.
Online action detection aims to predict the ongoing action category from the historical and current frame information. A typical online action detection system is mainly composed of three parts: frame-level feature extraction, temporal modeling, and action classification.

As illustrated in Fig. 1, an online action detection (OAD) system mainly consists of three important parts: a frame-level feature extractor (e.g., deep Convolutional Neural Network, CNN), a temporal modeling module to aggregate frame-level features, and an action classifier. Recent works on online action detection mostly focus on the temporal modeling part, aiming to generate discriminative representations from the historical and current frame features. Inspired by the sequence modeling methods in other areas especially the Long Short-Term Memory recurrent network (LSTM) [28], various temporal modeling methods have been developed for online action detection recently. For example, Geest et al. [10] provided a LSTM-based baseline which shows superiority to the single-frame CNN model. Gao et al. [20] proposed a LSTM-based Reinforced Encoder–Decoder network for both action anticipation and online action detection. Geest et al. [11] proposed a two-stream feedback network, where one stream focuses on the input interpretation and the other models temporal dependencies between actions. Xu et al. [72] utilized LSTM cell to model temporal context aiming to improve online action detection by adding prediction information into observed information.

Although the above LSTM-based temporal modeling methods have significantly boosted the performance on existing OAD datasets (e.g., TVSeries [10], THUMOS-14 [32]), their superiority to other temporal models, e.g., naive temporal pooling, temporal convolution, and attention-based sequence models, is not discussed and remains unknown. Moreover, the fusion of different temporal models is also rarely investigated. To address these problems, we evaluate traditional temporal convolution (TC), pyramid dilated temporal convolution (PDC) [39], and dilated causal convolution (DCC) [49]. For recurrent neural networks, we evaluate LSTM and Gated Recurrent Unit (GRU) with two output choices, i.e., the last hidden state and the average hidden state. For temporal attention, we evaluate naive self-attention (Naive-SA) with a linear fully connected (FC) layer and Softmax function, nonlinear self-attention (Nonlinear-SA) with a FC-tanh-FC-Softmax architecture, Non-local block or standard self-attention with a skip connection, and our Modified Non-local (M-NL) with the current feature as the query (Q) and past information as the key (K) and value (V), which outperforms the traditional Non-local model. It is worth noting that (i) we try to keep the original names of these related methods in other topics though we make adaptations for online action detection, and (ii) many of these methods are introduced into online action detection for the first time to the best of our knowledge, such as TC, PDC, DCC, Non-local, etc. Overall, we extensively explore eleven individual temporal modeling methods with the off-the-shelf two-stream (TS) [58,66] frame features.

The hybridization of temporal modeling methods Generally, these sequence-to-sequence methods, e.g., PDC and LSTM, can be further processed by aggregation methods to create a single representation like temporal pooling and temporal attention. Thus, we present several hybrid temporal modeling methods which combine different temporal modeling methods aiming to uncover the complementarity among them. Interestingly, we find that a simple fusion between dilated causal convolution and our modified non-local or LSTM improves the individual models significantly.

Comparison with state-of-the-art We extensively compare our individual models and hybrid temporal models to existing baselines and recent state-of-the-art methods. Several hybrid temporal models outperform the best existing performance with a sizable margin on both TVSeries and THUMOS-14. Specifically, the fusion of dilated causal convolution and M-NL obtains 84.3% cAP on TVSeries, and the fusion of dilated causal convolution, LSTM, and M-NL achieves 48.6% mAP on THUMOS-14.

In summary, the main contributions of this work can be concluded as follows.

- We provide a fair empirical study on eleven temporal modeling methods for online action detection and many of these methods are introduced into OAD for the first time, such as TC, PDC, DCC, non-local, etc.
- We modify the traditional non-local block with current feature as query and past features as key and value, and term it as M-NL. With this operation, the past information to be weighted according to the dot-product similarity.
between the current feature, which outperforms the traditional non-local operation.

- In Section Hybrid temporal modeling methods, we explore several hybrid temporal modeling methods to uncover the complementarity among them. The two hybrid models (i.e., the hybridization of DCC, LSTM and M-NL, and the hybridization of DCC and M-NL) outperform state-of-the-art in extensive experiments on two benchmark datasets, i.e. TVSeries, THUMOS-14.

### Related work

Our study is related to several other action-related tasks, namely action recognition, early action detection, action anticipation, temporal action detection, spatial-temporal action detection. In this section, we first briefly overview these related tasks separately and then present the recent works on online action detection.

**Action recognition** is an important branch of video-related research areas and has been extensively studied in the past decades. The existing methods are mainly developed for extracting discriminative action features from temporally complete action videos. These methods can be roughly categorized into hand-crafted feature based approaches and deep learning based approaches. Early methods such as Improved Dense Trajectory (IDT) [65] mainly adopted hand-crafted features, such as Histograms of Oriented Optical Flow (HOF) [37], Histogram of Oriented Gradient (HOG) [37] and Motion Boundary Histograms (MBH) [?]. Recent studies demonstrate that action features can be learned by deep learning methods such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). For example, two-stream network [58,66] learned appearance and motion features based on RGB frame and optical flow field separately. RNNs, such as long short-term memory (LSTM) [28] and gated recurrent unit (GRU) [7], have been used to model long-term temporal correlations and motion information in videos, and generate video representation for action classification. Some recent works also tried to model temporal information within a 2D-CNN instead of using 2D-CNN as static feature extractor, e.g. both TSM [41] and TAM [16] proposed an efficient approach to aggregate feature across frames inside the network.

Another type of action recognition approach is based on 3D CNNs, which are widely used for learning large-scale video datasets. C3D [13] was the first successful 3D CNN model for video classification. After that, many works extended C3D to different backbones, e.g. I3D [5] and ResNet3D [25]. In addition, some works aimed to reduce the complexity of 3D CNN by decomposing the 3D convolution into 2D spatial convolution and 1D temporal convolution, e.g. P3D [51], S3D [69], R(2+1)D [14].

**Early action detection** This topic is similar to online action detection but focuses on recognizing actions from the partially observed videos, which usually assume that every video contains exactly one instance of a given action. Hoai et al. [27] proposed a max-margin framework with structured SVM to address this problem. Huang et al. [29] introduced Sequential Max-Margin Event Detectors (SMMED) to efficiently detect an event in the presence of a large number of event classes. Ma et al. [46] addressed the problem by training an LSTM network with ranking loss and merged the detection spans based on the frame-wise prediction scores generated by the LSTM.

**Action anticipation** aiming to predict future unseen actions with historical and current information. Many works have been developed for this task in recent years. For instance, Ryoo et al. [54] developed an early action prediction system by observing some evidences from the temporal accumulated features. Yu et al. [75] formulated the action prediction problem into a probabilistic framework, which aimed to maximize the posterior of activity given observed frames. Aliakbarian et al. [1] developed a multi-stage LSTM architecture that leveraged context-aware and action-aware features, and introduced a novel loss function that encouraged the model to predict the correct class as early as possible. Gao et al. [20] proposed a Reinforced Encoder–Decoder (RED) network for action anticipation, which used reinforcement learning to encourage the model to make the correct anticipations as early as possible. Ke et al. [35] proposed an attended temporal feature, which used multi-scale temporal convolutions to process the time-conditioned observation. The widely used datasets for action anticipation, e.g., UCF-101 [63], JHMDB-21 [31], BIT-Interaction [36], Sports-1M [34], include short trimmed videos, and the task mainly focuses on predicting the class of the current going action timely from only a small ratio of the observed parts. Our task is different from action anticipation, we mainly focus on long and unsegmented video data, e.g. TVSeries, usually with a large variety of irrelevant background.

**Temporal action detection** or localization is another hot topic which aims to temporally localize and recognize actions by observing entire untrimmed videos. The main difference between this topic and OAD is the offline setting, i.e. post-processing is allowed for temporal action localization. In this offline setting, the whole action can be observed first. The problem has recently received increasing attention due to its potential application in video data analysis. Shou et al. [57] localized actions with three stages: action proposal generation, proposal classification and proposal regression. Xu et al. [71] transformed the Faster R-CNN [52] architecture into temporal action localization. Chao et al. [6] improved recep-
Spatial-temporal action detection aims to determine the precise spatial-temporal extents of actions in videos, which has attracted increasing attention recently. Early methods mainly resort to bag-of-words representation and search spatio-temporal path. In deep learning era, many works transform image-based object detection methods into this task, e.g., R-CNN [22], Faster R-CNN [52], SSD [45], etc. These adaptive methods mainly first detect actions in frame level and then link the frame-level bounding boxes into final tubes [23,24,50,61]. Specially, the online setting is used in [61,62].

Online action detection is defined as an online per-frame labelling task given streaming videos, which requires correctly classifying every frame. Geest et al. [10] first introduced the problem by introducing a realistic dataset (i.e., TVSeries) and some baseline results. Their later work [11] introduced a two-stream feedback network, where one stream processed the input and the other one modelled the temporal relations. Li et al. [40] designed a deep LSTM network for 3D skeletons online action detection which also estimated the start and end frame of the current action. Xu et al. [72] proposed the Temporal Recurrent Network (TRN) to model the temporal context by simultaneously performing online action detection and anticipation. Eun et al. [15] designed a novel recurrent unit named Information Discrimination Unit to explicitly discriminate the information relevant to an ongoing action from others. Besides, Shou et al. [56] formulated the online detection of action start (ODAS) as a classification task of sliding windows and introduced a model based on Generative Adversarial Network (GAN) to generate hard negative samples to improve the training of the samples. Gao et al. [21] proposed StartNet to address ODAS which can be decomposed into two stages: action classification and start point localization.

Problem formulation

Given an observed video stream \( V = \{I_0, I_1, \ldots, I_t\} \) containing frames from time 0 to \( t \), the goal of online action detection is to recognize actions of interest occurring in frame \( I_t \) with these observed frames. This is very different from other tasks like action recognition and temporal action detection which assume the entire video sequence is available at once. Formally, online action detection can be defined as the problem of maximizing the posterior probability as follows:

\[
y^*_t = \arg \max_{y_t} P(y_t|I_0, I_1, \ldots, I_t),
\]

where \( y_t \in \mathbb{R}^{K+1} \) is the possible action label vector for frame \( I_t \) with \( K \) action classes and one background class. Thus, conditioned on the observed sequence \( V \), the action label with the maximum probability \( P(y_t|I_0, I_1, \ldots, I_t) \) is chosen to be the detection result of frame \( I_t \). Generally, a pre-trained CNN model \( \Phi \) is first used to extract frame-level features, e.g., the feature of \( t \)th frame \( f_t = \Phi(I_t; \theta) \in \mathbb{R}^d \), where \( \theta \) is the fixed parameter of the model and \( d \) is the dimension of feature embedding. Given the observed frame features \( \{f_0, f_1, \ldots, f_t\} \), a temporal modeling module aims to aggregate discriminative information from them to better estimate the output action scores.

Temporal modeling

For online action detection, considering that faraway frames may be unrelated to the current action state, we usually input frames of a limited sequence length \( L \) to the temporal modeling module, i.e., \( \{f_{t-L+1}, f_{t-L+2}, \ldots, f_t\} \). For convenience, we denote the input features as \( F_{\text{in}} = \{f_1, f_2, \ldots, f_L\} \in \mathbb{R}^{L \times d} \), and assume the output of temporal modeling as \( S_{\text{out}} \in \mathbb{R}^d \), where \( L \) is the length of the input feature sequence and \( d \) is the feature dimension. Note that both feature extraction (see Section 4.3) and action classifier are the same for all the temporal models. We focus on discussing four types of temporal modeling methods as illustrated in Fig. 2 in this section.

Temporal pooling

Temporal feature pooling has been extensively used for video classification [17,33,48,58] which is a simple method to generate video-level representation from frame-level features. As shown in Fig. 2A, given the input feature sequence \( F_{\text{in}} = \{f_i\}_{i=1}^L \in \mathbb{R}^{L \times d} \), we consider two temporal pooling approaches:

**Average pooling** (AvgPool), i.e., \( S_{\text{out}} \approx \frac{1}{L} \sum_{i=1}^L f_i \), the generated feature \( S_t \) is the average representation of the past information \( \{f_1, f_2, ..., f_{L-1}\} \) including the current \( f_L \).
Max pooling (MaxPool) over the temporal dimension, usually the discriminative feature representation is selected, i.e., $S_{out} = \max_t f_t$.

For temporal pooling models, the embedded feature vector $S_{out} \in \mathbb{R}^d$ is then fed into the action classifier to output a probability distribution of the current action over $K$ action classes and one background by default. Note that this operation is the same for all the other temporal models.

**Temporal convolution**

Inspired by the convolutional approaches in the analysis of temporal sequential data [3,8,38,39,49], especially the WaveNet [49], we evaluate: (1) traditional temporal convolution (TC) with various kernel size and dilation rates, (2) pyramid dilated temporal convolution (PDC) originally used in [39], and (3) dilated causal convolution (DCC) developed in [49].

**Traditional temporal convolution** Formally, given input $F_{in} = \{f_1, f_2, \ldots, f_L\} \in \mathbb{R}^{L \times d}$, the dilated temporal convolution models output features as follows:

$$F_0 = \left\{ f_{o1}^{(r)}, f_{o2}^{(r)}, \ldots, f_{oL}^{(r)} \right\},$$

where $f_{oi}^{(r)} = \sum_{i=1}^{s} f_{i+r-i} \times W_{oi}^{(r)}$, $r$ is a dilation rate indicating the temporal stride to sample frames, $W \in \mathbb{R}^{d \times s}$ is a convolutional kernel, and $s$ is the kernel size. It becomes our traditional temporal convolution (i.e., Conv1D without dilation) if $r = 1$.

**Pyramid Dilated Temporal Convolution** The basic block for our PDC and DCC is shown in Fig. 3a, where each TemporalBlock has a dilated Conv1D, a ReLU activation [47], a dropout for regularization and a $1 \times 1$ Conv1D for channel reduction optionally. As shown in Fig. 3b, PDC first separately conducts several TemporalBlock with various dilation rates $\{r_1, r_2, \ldots, r_N\}$, where $N$ is the number of dilation rates, and then concatenates the outputs in temporal dimension. Formally, the concatenated feature at time $t$ is defined as follows:

$$f_t' = \text{concat} \left( f_t^{(r_1)}, f_t^{(r_2)}, \ldots, f_t^{(r_N)} \right) \in \mathbb{R}^{Nd},$$

where $f_t^{(r_N)}$ are the outputs of the $N$th TemporalBlock. Thus, we get the concatenated feature sequence $f_{PDC} = \{ f_t' \}_{t=1}^L \in \mathbb{R}^{L \times Nd \times N}$. To keep the same feature dimension of the inputs,
a Conv1D with kernel size $s = 1$ is added optionally for channel reduction.

Dilated Causal Convolution As shown in Fig. 3(c), our dilated causal convolution (DCC) stacks several TemporalBlock with different dilation rates $r$. For each layer, we increase dilation rate $r$ exponentially with the depth $i$ of the network (i.e., $r = O(2^i)$) and change the feature dimension to $d_i$ at level $i$ of the network. The resulting feature is fused with the input $F_{in}$ by residual connection, leading to the updated features $F'_{DCC}$. The computation of DCC can be denoted as

$$
F'_{DCC} = \text{ReLU} (\tilde{F}^{(rN)}_i + W_{r2}F_{in} + b_{c2}) ,
$$

where $W_{c1}, b_{c1}$ and $W_{c2}, b_{c2}$ are parameters to transform $F^{(rN)}_i$ and $F_{in}$ for the residual connection, respectively.

In our study, we use three dilation rates $\{1, 2, 4\}$ to efficiently enlarge the temporal receptive fields for temporal convolution models (i.e., PDC and DCC), and to map the input sequence to an output sequence of the same length, zero padding of length $(s - 1) \times r$ is added in the TemporalBlock. After the temporal convolutional operation, we use an average pooling layer to generate a single representation $S_{out} \in \mathbb{R}^d$ by default. For online action detection, the past as well as the current information are accumulated by the pooling operation. Thus, it is advantageous to make a decision for the current action based on the accumulated single representation.

Recurrent neural network

Recurrent Neural Network and its variants have recently been transformed from other sequence modeling topics into action classification [12,18,44,48] and detection [21,60,72]. Specifically, we evaluate two popular recurrent cells, namely Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU).

Long short-term memory Given the input feature sequence $F_{in} = \{f_1, f_2, \ldots, f_L\} \in \mathbb{R}^{L \times d}$, the RNN cell can effectively accumulate information to learn a compact representation $H_t = [h_1^t, h_2^t, \ldots]$ with the same length of the inputs. Specifically, at each time $t$, LSTM uses the previous hidden state $h_{t-1}$, the cell $c_{t-1}$, and the feature $f_t$ to update its hidden state $h_t$ and $c_t$. Formally, LSTM is formulated as follows:

$$
i_t = \sigma (W_i f_t + U_i h_{t-1} + V_i c_{t-1} + b_i) ,
g_t = \sigma (W_g f_t + U_g h_{t-1} + V_g c_{t-1} + b_g) ,
c_t = g_t c_{t-1} + i_t \text{tanh} (W_c f_t + U_c h_{t-1} + b_c) ,
o_t = \sigma (W_o f_t + U_o h_t + V_o c_t + b_o) ,
h_t = o_t \text{tanh}(c_t),
$$

where $\sigma$ is the logistic sigmoid function, and $i$, $g$, $c$ and $o$ are, respectively, the input gate, forget gate, memory cell and output gate. $h$ is the hidden state activation vector. $W, U, V$ are the weight metrics, and $b_i, b_g, b_c, b_o$ denote the bias vectors.

Gated recurrent unit Similar to LSTM unit, the GRU has gating units that modulate the flow of information inside the unit. The main difference between LSTM and GRU is that there is no separate memory cell in GRU. Formally, the GRU can be formulated as follows,

$$
r_t = \sigma (W_r f_t + U_r h_{t-1}) ,
\tilde{h}_t = \tanh (W_h f_t + U_h (r_t \odot h_{t-1})) ,
z_t = \sigma (W_z f_t + U_z h_{t-1}) ,
h_t = (1 - z_t) h_{t-1} + z_t \tilde{h}_t,
$$

where $r_t$ is a set of reset gates, $z_t$ is an update gate, and $\odot$ is an element-wise multiplication. $h_{t-1}$ is a candidate hidden state activation at time step $t$. $\tanh$ is a nonlinear activation. $\sigma$ denotes the sigmoid function.

Output Strategy We then consider two methods for the above two recurrent models to generate the single representation $S_{out}$, (i) following the traditional Encoder–Decoder method, we directly take the hidden state $h_L$ at the last time step, i.e. $S_{out} = h_L$. (ii) We average the outputs of all the time steps, i.e. $S_{out} = \frac{1}{T} \sum_{t=1}^{T} h_t$.

Temporal attention

The attention mechanism [2,43,64,73] allows the model to selectively focus on only a subset of frames by increasing the attention weights of the corresponding temporal features, while ignoring irrelevant signals and noise. We evaluate four attention methods, namely (1) naive self-attention (Naive-SA), (2) nonlinear self-attention (Nonlinear-SA) with...
a FC-tanh-FC-Softmax architecture, (3) Non-local block or standard self-attention with a skip connection, and (4) our Modified Non-local with the current information as the query (Q) and past information as the memory (key(K) and value(V)).

**Naive self-attention** Given the feature sequence \( F_{in} = \{f_1, f_2, \ldots, f_L\} \), the Naive-SA can be implemented by a linear fully-connected (FC) layer and Softmax function as follows:

\[
a = \text{Softmax}(WF_{in}^T + b),
\]

where \( W \in \mathbb{R}^{d} \) and \( b \) are weight and bias parameters of the FC, and \( a \in \mathbb{R}^{L} \) is the attention weight vector. \( F_{in}^T \) denotes feature transposition.

**Nonlinear self-attention** Similar to [73], we can also add more nonlinear operation as follows (i.e., the Nonlinear-SA):

\[
a = \text{Softmax}\left(U_2\tanh\left(U_1F_{in}^T + b_1\right) + b_2\right),
\]

where \( U_1 \in \mathbb{R}^{d_1 \times d} \) is a weight matrix, \( b_1 \in \mathbb{R}^{d_1} \) is the bias vector, and \( U_2 \in \mathbb{R}^{d_2 \times d_1} \) and \( b_2 \) are parameters of the second FC layer. With the above two attention weights, the output representation is the weighted average vector \( S_{out} = \sum_{i=1}^{L} a_i f_i \).

**Non-local** is another popular attention based model, which was originally proposed to capture long-range dependencies [67]. The core idea of Non-local is to model correlation between contextual signals by an attention mechanism. Specifically, it aims to encode the input sequence to a higher-level representation by modeling the relationship between queries (Q), keys (K) and values (V) with,

\[
A = \text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_m}}\right)V,
\]

where \( Q \in \mathbb{R}^{L_q \times d_q}, K \in \mathbb{R}^{L_k \times d_k} \) and \( V \in \mathbb{R}^{L_v \times d_v} \). \( d_q - d_k - d_v \) are the feature dimension of the encoded feature representations (usually, \( d_q = d_k = d_v \)). This architecture becomes standard “self-attention” [64] with \( Q=K=V \in \{f_1, f_2, \ldots, f_L\} \), as illustrated in Fig. 2D(c). Normally, we use two convolution layers followed by Batch Normalization [30] and ReLU [47] to generate two new features \( Q \) and \( K \) from \( F_{in} \), and the Non-local method [67] further adds a skip connection between the input and the output as follows:

\[
F' = AF_{in} + F_{in}, \quad F' \in \mathbb{R}^{L \times d}.
\]

The updated temporal feature \( F' \) is processed with average pooling by default to generate the final temporal representation \( S_{out} = \text{Avg}(F') \in \mathbb{R}^{d} \).

**Modified non-local** The query \( Q \) in Eq. (9) can also be a single feature vector, similar to [68] which replaces the self-attention weights by the one between local feature and long-term features. We compute the dot-product attention between current feature \( f_L \) and historical features \( \tilde{F} = \{f_1, f_2, \ldots, f_{L-1}\} \) as illustrated in Fig. 4, and we denote this operation as Modified Non-local (M-NL). Different from traditional non-local operation, this adaption is based on the assumption that the current feature is the most important one for online action detection, and the past information to be weighted according to the dot-product similarity between the current feature. With this operation, an attention weight vector \( a \in \mathbb{R}^{L-1} \) is obtained and used to get the final representation as follows:

\[
S_{out} = a\tilde{F} + f_L, \quad S_{out} \in \mathbb{R}^{d}.
\]

**Training and inference** With the output representation \( S_{out} \) of temporal modeling module, we use a linear fully-connected (FC) layer with Softmax function for action classification, and train the whole network with Cross-Entropy loss. Specifically, we divide the feature sequence of a video into non-overlapped windows with size \( L \) as the input of our temporal modeling module. At test stage, sliding window with size \( L \) and stride 1 is used to formulate the input, and the prediction is made for the last frame.

**Experimental configuration**

In this section, we first introduce two widely-used OAD datasets, i.e. TVSeries and THUMOS-14, and then describe our implementation details, including unit-level feature extraction and hyperparameter settings.
Datasets

TVSeries [10] was originally proposed for online action detection, which consists of 27 episodes of 6 popular TV series, namely Breaking Bad (3 episodes), How I Met Your Mother (8), Mad Men (3), Modern Family (6), Sons of Anarchy (3), and Twenty-four (4). It contains totally 16 hours of videos. The dataset is temporally annotated at the frame level without overlap, each unit contains \( n_u \) continuous frames. A video chunk \( u \) is processed by a visual encoder \( E_v \) to extract the unit-level representation \( f_u = E_v(u) \in \mathbb{R}^d \). In our experiments, we extract frames from all videos at 24 frames per second. The video unit size \( n_u \) is set to 6, i.e. 0.25 s. We use two-stream [70] network as the visual encoder \( E_v \) that is pre-trained on ActivityNet-1.3 [4]. In each unit, the central frame is sampled to calculate the appearance CNN feature, it is the Flatten 673 layer of ResNet-200 [26]. For the motion feature, we sample 6 consecutive frames at the center of a unit and calculate optical flows between them. These flows are then fed into the pretrained BN-Inception model [30], and the output of global pool layer is extracted. The motion features and the appearance features are both 2048-D, and are concatenated into 4096-D vectors (i.e. \( d = 4096 \)), which are used as unit-level features.

Evaluation protocols

For each class on TVSeries, we use the per-frame calibrated average precision (cAP) which is proposed in [10],

\[
cAP = \frac{\sum_k cPrec(k) \times I(k)}{P}, \tag{12}
\]

where calibrated precision \( cPrec = \frac{TP}{TP + FP / \bar{w}} \), \( I(k) \) is an indicator function that is equal to 1 if the cut-off frame \( k \) is a true positive, \( P \) denotes the total number of true positives, and \( w \) is the ratio between negative and positive frames. The mean cAP over all classes is reported for final performance. The advantage of cAP is that it is fair for class imbalance condition. For THUMOS-14, we report per-frame mean Average Precision (mAP) performance.

Implementation details

Unit-level feature extraction Following previous works [19, 20, 42, 72], a long untrimmed video is first cut into video units without overlap, each unit contains \( n_u \) continuous frames. A video chunk \( u \) is processed by a visual encoder \( E_v \) to extract the unit-level representation \( f_u = E_v(u) \in \mathbb{R}^d \). In our experiments, we extract frames from all videos at 24 frames per second. The video unit size \( n_u \) is set to 6, i.e. 0.25 s. We use two-stream [70] network as the visual encoder \( E_v \) that is pre-trained on ActivityNet-1.3 [4]. In each unit, the central frame is sampled to calculate the appearance CNN feature, it is the Flatten 673 layer of ResNet-200 [26]. For the motion feature, we sample 6 consecutive frames at the center of a unit and calculate optical flows between them. These flows are then fed into the pretrained BN-Inception model [30], and the output of global pool layer is extracted. The motion features and the appearance features are both 2048-D, and are concatenated into 4096-D vectors (i.e. \( d = 4096 \)), which are used as unit-level features.

Hyperparameter setting For the PDC model, the concatenate features are fed into an addition \( 1 \times 1 \) convolution to reduce the feature dimensions to 4096. For the DCC model, we use 3 dilated convolution layers, each of which is comprised of one dilated convolution with kernel size \( s = 2 \), stride 1, followed by a ReLU and dropout. The output dimension of the second layer \( d_2 \) is set to 2048. For the Non-local attention model, we set \( d_m \) to 512 and \( d_a \) to 4096. According to the number of action classes, we set \( K + 1 \) to 31 for TVSeries and 21 for THUMOS-14. Our experiments are conducted in Pytorch and we use Stochastic Gradient Descent (SGD) [53] optimizer to train the network from scratch. The leaning rate, momentum and decay rate are set to \( 10^{-3} \), 0.9 and 0.95, respectively. All of our experiments are implemented with 8 GTX TITAN X GPU, Intel i7 CPU, and 128GB memory.

Behavioural study of temporal modeling methods

In this section, we first present a quick comparison among the best settings of the four mentioned temporal modeling methods, and then extensively explore both individual temporal modeling methods and their combinations, and finally compare our results to the state-of-the-art.
A quick comparison of temporal modeling methods

As mentioned in the Section Introduction, we totally explore eleven temporal modeling methods from four meta types, including two temporal pooling methods, three temporal convolution methods, two recurrent neural networks, and four temporal attention methods. For a quick glance, we select the best setting in each model, and compare their performance, the results are shown in Table 1. For a fair comparison, the input sequence length $L$ is fixed to 4. Several observations can be concluded as follows:

(1) For temporal pooling models, AvgPool consistently works better than MaxPool; For temporal convolution models, DCC achieves the best results on both TVSeries and THUMOS-14 than TC and PDC, which indicates that discriminative information can be obtained effectively by stacking dilated causal convolution.

(2) M-NL performs slightly better than AvgPool, which demonstrates the effectiveness of attention mechanism. Third, LSTM outperforms M-NL and AvgPool with sizable margins on both datasets, which shows that the temporal dependencies captured by LSTM is crucial for accurate online action detection.

(3) Overall, an interesting finding is that the temporal-dependent methods, i.e. temporal convolution and RNN, are superior to these temporal-independent methods for online action detection.

Model efficiency analysis Table 1 also compares the number of parameters and FLOPs of each meta type. The FLOPs and parameters indicate the number of operations to process the inputs, which is a vital factor, especially for online action detection. For a fair comparison, the input sequence length $L$ is fixed as 4 by default and we test all the models under the same environment of GTX TITAN X GPU. Number of parameters of AvgPool, DCC, LSTM and M-NL are 0B, 0B, 33.6M, 251.7M respectively. For the comparison of performance and computation, LSTM model would be a compromise choice with similar model accuracy, less number of parameters and computational cost in FLOPs.

Ablation study for individual temporal modeling method

Temporal pooling

We test two temporal pooling methods (i.e. average pooling and max pooling) with different sequence lengths. The results are shown in Fig. 6. We also compare them to the baseline that uses a fully-connected (FC) layer and Softmax to gener-
ate action probabilities frame by frame. The baseline model only relies on the current frame feature which obtains 79.8% (cAP) and 36.3% (mAP) on TVSeries and THUMOS-14, respectively. For temporal pooling, it is clear that average pooling consistently performs better than max pooling on both datasets. Increasing the sequence length improves both pooling methods in the beginning and degrades them dramatically after the saturation length. This can be explained by that appropriate historical information introduces useful context for online action detection while long-term historical information may introduce unrelated information and may also smooth the final representation. Another observation is that increasing the sequence length after $L = 4$ is seriously harmful for TVSeries while not for THUMOS-14. This effect indicates a fact that each video in TVSeries contains multiple actions and numerous varied background frames while each video in THUMOS-14 only contains one action instance. Overall, the simple AvgPool method ($L = 4$), respectively, improves the baselines on TVSeries and THUMOS-14 by 1.4% and 5.2%.

**Recurrent neural network**

We evaluate LSTM and GRU in the following four aspects: input sequence length, output strategy, hidden size, and the number of recurrent layers.

*Input sequence length and output strategy* For these two factors, we vary the sequence length from 2 to 16, and evaluate two alternative output strategies including the last hidden state $S_{out} = h_L$ and the average hidden state $S_{out} = \frac{1}{L} \sum_{t=1}^{L} h_t$. The hidden size is fixed to 4096 and only one recurrent layer is used for this evaluation. Figure 9 illustrates the comparison results for LSTM and GRU. Several conclusions can be drawn as follows.

1. The ‘last hidden state’ strategy performs consistently better than the ‘average hidden state’. It can be explained by that both LSTM and GRU automatically accumulate discriminative information into the last state by their temporal dependency operations while averaging all the hidden states may introduce unrelated or noisy information for online action detection.  

2. LSTM performs better than GRU on THUMOS-14 while similarly or worse on TVSeries. This indicate that the separate memory cell in LSTM is helpful to capture more context information which is crucial for THUMOS-14 while too much context (unrelated actions or background) can degrade performance on TVSeries.

3. The effect of sequence length for both LSTM and GRU is the same as the one for pooling methods, and the best trade-off sequence length is 4 on both datasets.

As shown in Table 2, we compare temporal convolution models with different kernel size $s$ and dilation rate $r$, denoted as $(s, r)$. For PDC and DCC, we use temporal convolutional filters with kernel size $s = 2$ as a building block. The input sequence length is fixed to $L = 4$ for all the comparison experiments. To obtain output with equal length as the input, we add zero padding as it needs. Several observations can be concluded as follows.

1. The comparison between TC(2,1) and TC(3,1) indicates that the kernel size $s = 2$ is slightly better than $s = 3$ on both datasets.

2. The comparison among TC(2,1), TC(2,2), and TC(2,4) shows that different dilation rates perform similarly on both datasets.

3. Both PDC and DCC which combines TC(2,1), TC(2,2), and TC(2,4) in either parallel or cascade manner significantly improve the traditional TC models. This demonstrates that combining multi-dilation temporal convolution layers can capture complementary multi-scale action information.

4. DCC with cascade manner obtains the best results, leading to a performance of 83.1% on TVSeries and 46.8% on THUMOS-14. The success of dilated causal convolution with cascade manner suggests that discriminative and relevant information could be enhanced layer by layer from the smaller temporal receptive field ($r = 1$) to the large one ($r = 4$).

### Temporal convolution

| TC(2,1) | TC(3,1) | TC(2,2) | TC(2,4) | PDC | DCC |
|---------|---------|---------|---------|-----|-----|
| TVSeries | 81.1    | 80.9    | 81.0    | 80.8 | 82.7 | **83.1** |
| THUMOS-14 | 42.4    | 41.9    | 42.6    | 42.7 | 46.1 | **46.8** |

*Table 2* Comparison between different temporal convolutional models

*Best results are shown in bold*
Hidden size

We choose LSTM, and test different hidden sizes $D_h = 128, 256, 512, 1024, 2048, 4096$. The last hidden state output strategy and sequence length $L = 4$ are used for this evaluation. The results are shown in Fig. 7. A clear observation is that increasing hidden size improves the final performance significantly on both datasets. In addition, it gets saturated after hidden size 2048, and the best results are, respectively, 82.9% and 45.9% on TVSeries and THUMOS-14 with hidden size 4096.

The number of recurrent layers

Generally, one can easily stack several recurrent layers to model the complex dependency of sequences. To this end, we evaluate the number of recurrent layers for both LSTM and GRU on TVSeries and THUMOS-14. The results are shown in Fig. 8. Interestingly, adding one more layer does not bring performance gain and even dramatically degrades the performance for LSTM on both datasets. The main problem is that adding one more recurrent layer can double the number of parameters leading to overfitting easily.

Temporal attention

We compare four different attention models mentioned in Section Temporal modeling, i.e. Naive-SA as described in Eq. (7), Nonlinear-SA as described in Eq. (8), Non-local as described in Eq. (10), and M-NL as described in Eq. (11). As shown in Table 1, for temporal attention models, several observations can be concluded as follows.

(1) Nonlinear-SA outperforms Naive-SA by 0.8% on TVSeries and 2.6% on THUMOS-14. Compared to Naive-SA, Nonlinear-SA computes attention weights with one more nonlinear tanh and linear FC which may be more effective for modeling the complex temporal relationships.

(2) Non-local performs equally to Nonlinear-SA on both datasets, indicating that they share the similar attention mechanism more or less.

(3) Our Modified Non-local with current feature as a query and past features as key and value performs better than Non-local by 0.6% on TVSeries and 0.9% on THUMOS-14, showing the effectiveness of our proposed design (i.e., computing the attention between current feature with historical features) for online action detection.

As there is a hyper-parameter $d_1$ in Nonlinear-SA (see Eq. (8)) which can impact the final performance, we also conduct an evaluation in Table 3. We observe $d_1 = 512$ (1024) yield the best performance for TVSeries (THUMOS-14), and the final performance is not very sensitive to it.

| Table 3 | Evaluation of $d_1$ on TVSeries (cAP %) and THUMOS-14 dataset (mAP %) |
|---------|-------------------------------------------------|
| $d_1$   | 256     | 512   | 1024  | 2048  | 4096  |
| TVSeries| 80.6    | **80.9**| 80.6  | 80.7  | 80.3  |
| THUMOS-14| 42.0    | 42.0   | **42.5**| 41.1  | 41.5  |

Best results are shown in bold

| Table 4 | Comparison between different hybrid temporal models on TVSeries (cAP %) and THUMOS-14 (mAP %) |
|---------|-----------------------------------------------------------------------------------------------|
| Hybrid models | TVSeries | THUMOS-14 |
| M1 | LSTM $\oplus$ M-NL | 83.6 | 47.7 |
| M2 | DCC $\oplus$ M-NL | **84.3** | 47.1 |
| M3 | DCC $\oplus$ LSTM $\oplus$ M-NL | 83.7 | **48.6** |
| M4 | LSTM $\oplus$ DCC $\oplus$ M-NL | 83.0 | 48.5 |
| M5 | DCC $\oplus$ LSTM | 83.2 | 47.9 |
| M6 | LSTM $\oplus$ DCC $\oplus$ AvgPool | 81.5 | 47.5 |

Best results are shown in bold
Hybrid temporal modeling methods

Generally, these sequence-to-sequence temporal models, e.g. DCC and LSTM can be further processed by aggregation methods like temporal pooling and temporal attention to generate a single representation. Based on our empirical study of temporal models, we choose the best setting of each temporal modeling method, i.e. LSTM, DCC, AvgPool and M-NL, and propose several novel hybrid temporal modeling methods as presented in Table 4, aiming to uncover the complementarity among them. We mainly combine those temporal-dependent models and temporal-independent models. To clarify the architecture of hybrid models, we illustrate M3 (i.e. DCC ⊕ LSTM ⊕ M-NL) in Fig. 10 as an example. As shown in Fig. 10, the output sequence of DCC is further processed by LSTM aiming to capture strong temporal dependency, and finally, the M-NL module is used to generate the single representation for classification. The other hybrid models are described as follows:

M1 LSTM ⊕ M-NL: The input feature sequence are first fed into LSTM to update the hidden states at all time steps, and these are further fed into our Modified Non-local with the last hidden state as the Q and the other hidden states as K and V to generate a single representation, and the classification is performed on the representation. A diagram is similar to Fig. 10, while not containing the DCC model;

M2 DCC ⊕ M-NL: The output of the DCC network before the average pooling layer is the same as the input sequence length, and our Modified Non-local is performed on the output sequence of DCC to generate a single representation for current action classification. A diagram is similar to Fig. 10, while not containing the LSTM model;

M4 LSTM ⊕ DCC ⊕ M-NL: The hidden states at all time steps of LSTM layer are first fed into DCC and then the M-NL. This model is similar to M3 except for that it swaps the order of DCC and LSTM;

M5 DCC ⊕ LSTM: The output sequence of DCC is processed by LSTM to update the hidden states with the same length of the inputs, and then the last hidden state is used for action classification. A diagram is similar to Fig. 10, while not containing the M-NL model;

M6 LSTM ⊕ DCC ⊕ AvgPool: This model replaces the M-NL of M3 with AvgPool for the output of DCC to generate the final representation for classification.

The results of the hybrid methods on TVSeries and THUMOS-14 are shown in Table 4. Several observations can be concluded as follows.

1. The best results on TVSeries and THUMOS-14 are achieved by M2 and M3, respectively. Specifically, with the combination of DCC and M-NL models, we achieve mean cAP of 84.3% on TVSeries. By further combining LSTM among them, we obtain mAP of 48.6% on THUMOS-14.

2. Combining temporal-dependent models (i.e. LSTM and DCC) with temporal-independent ones (i.e. M-NL) largely improves individual models, which indicates that these temporal models are complementary to each other.

3. Integrating M-NL into DCC ⊕ LSTM (i.e. M5→M3) increases the performance by 0.5% on TVSeries and 0.7% on THUMOS-14, showing the effectiveness of our modified non-local design for online action detection.

4. Integrating LSTM before DCC ⊕ M-NL (i.e. M2→M4) degrades the performance by 1.3% on TVSeries while increases the one by 1.4% on THUMOS-14. This may be explained by that temporal dependencies are important for these long-term action instances of THUMOS-14 while harmful for the dominated short-term action instances of TVSeries. Specifically, DCC uses dilated convolutions to systematically capture multi-scale temporal contextual information with the exponential expansion of the receptive field. By combining LSTM, the long-term temporal dependencies can be further accumulated at the last hidden states, but this operation also lead to the irrelevant information to be accumulated, especially for TVSeries dataset, since different actions can occur at the same time and being performed by the same or multiple actors, as opposed to the setting of THUMOS-14, where actions are separated by a specific non-action.

5. Comparing M3→M4 and M5→M6, the order of LSTM and DCC makes difference on both datasets, the DCC-first order performs better than the LSTM-first order especially for TVSeries, showing the effectiveness of
Fig. 11 The online detection results of ours compared to previous methods in terms of per-frame cAP (%) for each action class on TVSeries

Table 5 Comparison with state-of-the-art methods in terms of per-frame cAP (%) on TVSeries

| Methods      | Inputs | cAP    |
|--------------|--------|--------|
| CNN [10]     | VGG    | 60.8   |
| LSTM [10]    |        | 64.1   |
| RED [20]     |        | 71.2   |
| Stacked LSTM [11] |        | 71.4   |
| 2S-FN [11]   |        | 72.4   |
| TRN [72]     |        | 75.4   |
| SVM [10]     | FV     | 74.3   |
| RED [20]     | TS     | 79.2   |
| TRN [72]     | TS     | 83.7   |
| IDN [15]     |        | 84.7   |
| Ours         | TS     | **84.3** |

The top-2 results are boldly marked and [15] is available online after our submission.

Table 6 Comparison with published state-of-the-art methods in terms of per-frame mAP (%) on THUMOS-14

| Methods            | Setting | mAP |
|--------------------|---------|-----|
| Single-frame CNN   | Offline | 34.7 |
| Two-stream CNN     | Offline | 36.2 |
| C3D+LinearInterp   | Offline | 37.0 |
| Predictive-corrective [9] | Offline | 38.9 |
| LSTM [12]          |        | 39.3 |
| MultiLSTM [74]     |        | 41.3 |
| CDC [55]           |        | 44.4 |
| RED [20]           | Online  | 45.3 |
| TRN [72]           | Online  | 47.2 |
| IDN [15]           |         | **50.0** |
| Ours               | Online  | **48.6** |

The top-2 results are boldly marked and [15] is available online after our submission.

DCC to discriminate relevant and filter out irrelevant information for online action detection.

Comparison with state-of-the-art

In this section, we compare our best results to the state-of-the-art approaches on TVSeries and THUMOS-14 datasets. As shown in Table 5, with two-stream features, we achieve 84.3% in terms of mean cAP on TVSeries which outperforms the recent sophisticated-designed TRN [72] by 0.6%. In addition, our hybrid method yield better performance than TRN with shorter temporal information (i.e., 4 chunks vs. 64 chunks). Besides, we also present the comparison of ours with previous method [10] for each action class on TVSeries in Fig. 11. Our method can always outperform CNN and LSTM by a large margin except for action class Use computer and Write with only subtle change happens in the two scenes.

In Table 6, we compare performances between our proposed hybrid method and the state-of-the-art approaches for online and offline action detection on THUMOS-14 dataset. The compared offline action detection methods perform frame-level prediction. As a result, we achieve mAP of 48.6% which outperforms TRN [72] by 1.4%.

Conclusions

In this paper, we provide an empirical study on temporal modeling for online action detection including four meta types of temporal modeling methods, i.e. temporal pooling, temporal convolution, recurrent neural networks, and temporal attention. We extensively explore eleven individual temporal modeling methods and explore several hybrid temporal models which combine temporal-dependent models with temporal-independent models to uncover the complementarity among them. Based on our empirical study, we find that a simple hybridization of dilated causal convolution and M-NL or LSTM improves the individual models significantly and also outperforms the best existing performance with a sizable margin on both TVSeries and THUMOS-14 datasets.

Declarations

Conflict of interest On behalf of all authors, the corresponding authors state that there is no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copy-
right holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Aliakbarian MS, Saleh FS, Salzmann M, Fernando B, Petersson L, Andersson L (2017) Encouraging lstms to anticipate actions very early. ICCV 1:280–289
2. Bahdanau D, Cho K, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473
3. Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271
4. Caba F, Escorca V, Ghanem B, Niebles JC (2015) Activitynet: A large-scale video benchmark for human activity understanding. CVPR 1:961–970
5. Carreira J, Zisserman A (2017) Quo vadis, action recognition? a new model and the kinetics dataset. CVPR 1:6299–6308
6. Chao YW, Vijayanarasimhan S, Seybold B, Ross DA, Deng J, Sukthankar R (2018) Rethinking the faster r-cnn architecture for temporal action localization. CVPR 1:1130–1139
7. Cho K, van Merrienboer B, Bahdanau D, Bengio Y (2014) On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259
8. Dauphin YN, Fan A, Auli M, Grangier D (2017) Language modeling with gated convolutional networks. ICML 70:933–941
9. Dave P, Russakovsky O, Ramanan D (2017) Predictive-corrective networks for action detection. CVPR 1:2067–2076
10. De Geest R, Gavves E, Ghodrat A, Li Z, Snoek C, Tuytelaars T (2016) Online action detection. ECCV 9909:269–284
11. De Geest R, Tuytelaars T (2018) Modeling temporal structure with lstm for online action detection. WACV 1:1549–1557
12. Donahue J, Hendricks LA, Rohrbach M, Venugopalan S, Guadarrama S, Saenko K, Darrell T (2015) Long-term recurrent convolutional networks for visual recognition and description. CVPR 1:2625–2634
13. Du T, Bourdev L, Fergus R, Torresani L, Paluri M (2015) Learning spatiotemporal features with 3d convolutional networks. ICCV 1:4489–4497
14. Du T, Wang H, Torresani L, Ray J, Lecun Y (2018) A closer look at spatiotemporal convolutions for action recognition. CVPR 1:6450–6459
15. Eun H, Moon J, Park J, Jung C, Kim C (2020) Learning to discriminate information for online action detection. CVPR 1:806–815
16. Fan Q, Chen CFR, Kuehne H, Pistola M, Cox D (2019) More is less: Learning efficient video representations by big-little network and depthwise temporal aggregation. NIPS 32:2261–2270
17. Feichtenhofer C, Pinz A, Zisserman A (2016) Convolutional two-stream network fusion for video action recognition. CVPR 1:1933–1941
18. Gammulle H, Denman S, Sridharan S, Fookes C (2017) Two stream lstm: A deep fusion framework for human action recognition. WACV 1:177–186
19. Gao J, Chen K, Nevatia R (2018) Ctap: Complementary temporal action proposal generation. ECCV 11206:70–85
20. Gao J, Yang Z, Nevatia R (2017) RED: reinforced encoder-decoder networks for action anticipation. BMVC 92:1–11
21. Gao M, Xu M, Davis LS, Socher R, Xiong C (2019) Startrnet: Online detection of action start in untrimmed videos. ICCV 1:5541–5550
22. Girshick RB, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR 1:580–587
23. Gkioxari G, Malik J (2014) Finding action tubes. arXiv preprint arXiv:1411.6031
24. Gu C, Sun C, Ross DA, Vondrick C, Pantofaru C, Li Y, Vijayanarasimhan S, Toderici G, Ricco S, Sukthankar R, Schmid C, Malik J (2018) AVA: A video dataset of spatio-temporally localized atomic visual actions. CVPR 1:6047–6056
25. Hara K, Kataoka H, Satoh Y (2017) Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? CVPR 1:6546–6555
26. He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385
27. Hoai M, Torre FD (2012) Max-margin early event detectors. CVPR 1:2863–2870
28. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Computation 9:1735–1780
29. Huang D, Yao S, Wang Y, De La Torre F (2014) Sequential max-margin event detectors. ECCV 8691:410–424
30. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167
31. Jhuang H, Gall J, Zuffi S, Schmid C, Black MJ (2013) Towards understanding action recognition. ICCV 1:3192–3199
32. Jiang YG, Liu J, Roshan Zamir A, Toderici G, Laptev I, Shah M, Sukthankar R (2014) THUMOS challenge: Action recognition with a large number of classes. http://crcv.ucf.edu/THUMOS14/
33. Kar A, Rai N, Sikka K, Sharma G (2017) Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos. CVPR 1:5699–5708
34. Karpathy A, Toderici G, Shetty S, Leung T, Li FF (2014) Large-scale video classification with convolutional neural networks. CVPR 1:1725–1732
35. Ke Q, Fritz M, Schiele B (2019) Time-conditioned action anticipation in one shot. CVPR 1:9917–9926
36. Kong Y, Jia Y (2014) Fu Y (2014) Interactive phrases: Semantic descriptions for human interaction recognition. TPAMI 36:1775–1788
37. Laptev I, Marszalek M, Schmid C, Rozenfeld B (2008) Learning realistic human actions from movies. CVPR 1:1–8
38. Lea C, Flynn MD, Vidal R, Reiter A, Hager GD (2016) Temporal convolutional networks for action segmentation and detection. arXiv preprint arXiv:1611.05267
39. Li J, Wang J, Tian Q, Gao W, Zhang S (2019) Global-local temporal representations for video person re-identification. ICCV 1:3957–3966
40. Li Y, Lan C, Xing J, Zeng W, Yuan C, Liu J (2016) Online human action detection using joint classification-regression recurrent neural networks. ECCV 9911:203–220
41. Lin J, Gan C, Han S (2018) Temporal shift module for efficient video understanding. arXiv preprint arXiv:1811.08383
42. Lin T, Zhao X, Su H, Wang C, Yang M (2018) Bsn: Boundary sensitive network for temporal action proposal generation. ECCV 11208:3–21
43. Lin Z, Feng M, Santos CNd, Yu M, Xiang B, Zhou B, Bengio Y (2017) A structured self-attentive sentence embedding. arXiv preprint arXiv:1703.03130
44. Liu J, Wang G, Hu P, Duan LY, Kot AC (2017) Global context-aware attention lstm networks for 3d action recognition. CVPR 1:3671–3680
45. Liu W, Anguelov D, Erhan D, Szegedy C, Reed SE, Fu C, Berg AC (2016) SSD: single shot multibox detector. ECCV 9905:21–37
46. Ma, S., Sigal, L., Sclaroff, S.: Learning activity progression in boltzmann machines. In: ICML, pp. 807–814
48. Ng YH, Hausknecht M, Vijayanarasimhan S, Vinyals O, Monga R, Toderici G (2015) Beyond short snippets: Deep networks for video classification. CVPR 1:4694–4702

49. Oord AVD, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior A, Kavukcuoglu K (2016) Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499

50. Peng X, Schmid C (2016) Multi-region two-stream R-CNN for action detection. ECCV 9908:744–759

51. Qiu Z, Yao T, Tao M (2017) Learning spatio-temporal representation with pseudo-3d residual networks. ICCV 1:5534–5542

52. Ren S, He K, Girshick R, Sun J (2017) Faster r-cnn: Towards real-time object detection with region proposal networks. TPAMI 39:1137–1149

53. Robbins H, Monro S (1951) A stochastic approximation method. The annals of mathematical statistics 22:400–407

54. Ryoo MS (2012) Human activity prediction: Early recognition of ongoing activities from streaming videos. CVPR 1:1036–1043

55. Shou Z, Chan J, Zareian A, Miyazawa K, Chang SF (2017) Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. ICCV 1:1417–1426

56. Shou Z, Pan J, Chan J, Miyazawa K, Mansour H, Vetro A, Giro-I-Nieto X, Chang SF (2018) Online detection of action start in untrimmed, streaming videos. CVPR 11207:534–551

57. Shou Z, Wang D, Chang SF (2016) Temporal action localization in untrimmed videos via multi-stage cnns. CVPR 1:1049–1058

58. Simonyan K, Zisserman A (2014) Two-stream convolutional networks for action recognition in videos. NIPS 1:568–576

59. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556

60. Singh B, Marks TK, Jones M, Tuzel O, Ming S (2016) A multi-stream bi-directional recurrent neural network for fine-grained action detection. CVPR 1:1961–1970

61. Singh G, Saha S, Sapienza M, Torr PH, Cuzzolin F. Online real-time multiple spatiotemporal action localisation and prediction. In: ICCV, vol. 1

62. Soomro K, Idrees H, Shah M (2016) Predicting the where and what of actors and actions through online action localization. CVPR 1:2648–2657

63. Soomro K, Zamir AR, Shah M (2012) Ucf101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402

64. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. arXiv preprint arXiv:1706.03762

65. Wang H, Schmid C (2013) Action recognition with improved trajectories. ICCV 1:3551–3558

66. Wang L, Xiong Y, Wang Z, Qiao Y (2015) Towards good practices for very deep two-stream convnets. arXiv preprint arXiv:1507.02159

67. Wang X, Girshick R, Gupta A, He K (2018) Non-local neural networks. CVPR 1:7794–7803

68. Wu CY, Feichtenhofer C, Fan H, He K, Krahenbuhl P, Girshick R (2019) Long-term feature banks for detailed video understanding. CVPR 1:284–293

69. Xie S, Sun C, Huang J, Tu Z, Murphy K (2017) Rethinking spatiotemporal feature learning for video understanding. arXiv preprint arXiv:1712.04851

70. Xiong Y, Wang L, Wang Z, Zhang B, Song H, Li W, Lin D, Qiao Y, Van Gool L, Tang X (2016) Cuhk ethz siat submission to activitynet challenge 2016. arXiv preprint arXiv:1608.00797

71. Xu H, Das A, Saenko K (2017) R-c3d: Region convolutional 3d network for temporal activity detection. ICCV 1:5794–5803

72. Xu M, Gao M, Chen YT, Davis LS, Crandall DJ (2019) Temporal recurrent networks for online action detection. ICCV 1:5531–5540

73. Yang Z, Yang D, Dyer C, He X, Smola A, Hovy E (2016) Hierarchical attention networks for document classification. NAACL 1:1480–1489

74. Yeung S, Russakovsky O, Jin N, Andriluka M, Mori G, Fei-Fei L (2018) Every moment counts: Dense detailed labeling of actions in complex videos. IJCV 126:375–389

75. Yu C, Barrett D, Barbu A, Narayanaswamy S, Song W (2013) Recognize human activities from partially observed videos. CVPR 1:2658–2665

76. Zeng R, Huang W, Tan M, Rong Y, Zhao P, Huang J, Gan C (2019) Graph convolutional networks for temporal action localization. ICCV 1:7093–7102

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.