Noname manuscript No.
(will be inserted by the editor)

Temporal Pyramid Networks with Enhanced Relation Mechanism for Online Action Detection

Jui-Ting Shen · Ti-Huai Song · Li-Chen Fu

Received: date / Accepted: date

Abstract Online action detection aims to identify actions as soon as each video frame arrives from a streaming video. An input video sequence contains not only the action of interest frames but also background (non-action) and other irrelevant frames. Those frames will cause the network to learn less discriminative features. This paper explores an Enhanced Relation Layer (ERL) embedded in the Temporal Convolution Network (TCN), which updates the features according to their relevance to the action of interest. Because the relevant features should be considered essential. ERL gives each time-step a relevance and an actioness score. Those scores imply the relevance to the action of interest and the probability of the action occurrence, respectively. They guide the network to focus on those more essential features and learn a more discriminative representation for identifying the action happening in the current time-step. The temporal information of an input sequence is learned from TCN. The output feature of each layer in TCN has different receptive fields, focusing on different temporal scales. However, lower-level features are semantically weak. Therefore, we design a Temporal Pyramid Network with a top-down architecture to transform the strong semantic ability from higher-levels to lower-levels, a multi-temporal-scale feature sequence has been built to further identify actions with different temporal lengths. In the experiment part, our method is tested on two benchmark datasets, THUMOS-14 and TVSerires. We achieve superior performance as compared with baseline networks and promising results as compared with the state-of-the-art works.

Keywords Deep learning · Video content analysis · Online action detection · Temporal modeling · Feature pyramid

Jui-Ting Shen
Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan (R.O.C.)
E-mail: r07921069@ntu.edu.tw

Ti-Huai Song
Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan (R.O.C.)
E-mail: r09921135@ntu.edu.tw

Li-Chen Fu
Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan (R.O.C.)

1 Introduction

With the flourishing of the internet and digital camera development, the number of videos increases rapidly. Video content analysis has raised lots of attention from the industry and academic fields. There are majorly two branches in video content analysis. One is video action recognition, and the other is video action detection. Action detection aims to detect human action instances with an action category and its time interval, which indicates the starting and ending time of an action. However, most of the works defines action detection as an off-line problem, which means we need to observe the whole video to determine action instances. But in the real-world applications, such as surveillance systems, autonomous driving, assistance robots and abnormal behaviors, the action should be detected immediately. The detection needs to be executed every moment from a streaming video without accessing any future information.

Online Action Detection (Geest et al., 2016) is still a very challenging task due to several reasons. First, we can only observe part of the video that has been played

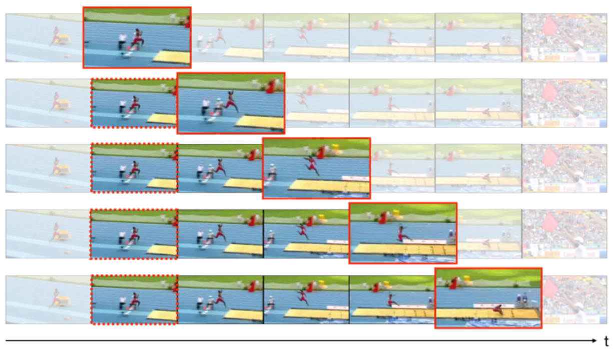

Fig. 1 An example of an identical action instance, the length of the action we observed could be different due to the observation time difference. Every row is the same example of a video sequence. The highlighted frames are the frames currently observed. The frames in the second column are where the action start to take place. The later the observation time (upper row to lower row), the longer the action information we can obtain (non-transparent frames in each row).

for a streaming video. Since the action needs to be detected as soon as every frame arrives, we cannot make the decision after watching the entire video. Thus, the contents in the future are not available. Second, the start of the action is unknown. Videos in action detection tasks have a large amount of background (non-action) frames. We don't know when and how many actions take place in the video. Third, large intra-class variability exists in the data. The variability exists in two aspects, spatial and temporal. In the temporal aspect, the same action performed by different actors may have different durations. On the other hand, due to the different observation time points, the durations of the action we observed will be different, as depicted in Fig.1

Given a video sequence, we are curious about the actions conducted in the current frame. The sequence comprises not only the ongoing actions but other irrelevant actions and backgrounds. If we directly take the entire sequence into account, the model will accumulate the information in the sequence without explicitly considering its relevance to the current action, affecting the detection results. In paper, we propose an Enhanced Relation Layer (ERL), which neglects the irrelevant information while emphasizing relevant one based on its relation with the frames from the action of interest. ERL also filters out the background frames based on their actioness score. By such a mechanism, the representation of the learned features will be more discriminative.

Besides, we also try to handle the temporal variability problem of actions. In object detection tasks, leveraging multi-scale features to detect objects of different sizes is a commonly used strategy. We adopt this idea from the spatial domain to the temporal domain and utilize the Temporal Convolution Network (TCN) (Bai et al., 2018) as our backbone model. Through the designed pyramid architecture of the convolution model, the output features will be able to obtain semantic *multi-temporal-scale* information. Then, we can use these features to detect actions with different temporal lengths. As a result, we name the network as Temporal Pyramid Network.

2 Related Work

2.1 Action Recognition and Sequential Modeling

Videos contain not only the spatial domain, but the temporal dimension is also involved. Simonyan and Zisserman (2014) trained two 2D Convolutional Neural Networks (CNNs), one for RGB image and the other for optical flow (Brox et al., 2004) to depict human motion, and then merge two learned information through late fusion. Ji et al. (2012) extended the traditional 2D CNNs to 3D CNNs where the temporal information is involved in convolutions as the additional dimension. Recently, Feichtenhofer et al. (2019) proposed a Slow-Fast Network, an architecture with two pathways. A Slow path with a low frame rate captures spatial semantics, and a Fast path operates with a high frame rate to capture motion semantics. A lateral connection then fuses two pathways.

On the other hand, instead of extracting features with a single network, Donahue et al. (2015) proposed a Recurrent Neural Network (RNN). RNN is capable of modeling the temporal information between video frames. Temporal Convolutional Network (TCN) (Bai et al., 2018) is another architecture to model sequential details. TCN models the temporal information between adjacent features via a one-dimensional convolution and outputs a feature sequence with each time step corresponding to the input sequence.

2.2 Action Detection

The main drawback of action recognition is that the video is manually trimmed with only one action. Thus, action detection has gained more attention lately. Action detection can be decomposed into two tasks: localization and recognition.

Action detection is mainly grouped into two frameworks: one-stage detection and two-stage detection. Two-stage detection follows the pattern of "proposal then classification." Xiong et al. (2017) proposed a Temporal Actioness Grouping (TAG) framework to determine action boundaries. TAG utilizes the extracted feature at each timestep to decide an action confidence score implying whether the timestep contains actions, and then group those features with high action confidence score as proposals. Gao et al. (2017b) proposed a Temporal Unit Regression Network (TURN). This method decomposes long untrimmed video into video units. Adjacent units form a clip and let each unit construct a clip pyramid with multiple temporal scales as an anchor unit. They jointly predict action proposals and refine the temporal boundaries by temporal coordinate regression.

Later, Lin et al. (2018) designed a Boundary Sensitive Network (BSN) in three stages. Briefly, BSN first locates the boundaries, and then combine the boundaries points directly into a temporal proposal. Then, it extracts a proposal-level feature based on the sequence of action confidence scores for each candidate proposal. Finally, based on the proposal-level extracted feature, it evaluates the confidence of the temporal proposals. One year later, Boundary Matching Network (BMN) (Lin et al., 2019) was also proposed by Lin *et al.* as an improved BSN version. BMN can generate a one-dimensional probability of boundary and the two-dimensional confidence map for boundary matching simultaneously to evaluate proposals' confidence.

On the other hand, one-stage detection tackles the problem of proposal and classification simultaneously. Lin et al. (2017a) proposed a single shot temporal action detection (SSAD) based on one-dimensional temporal convolution layer to skip proposal generation step and directly detects action instance in the video. Feifei's group proposed a Single-Stream Temporal Action Detection (SS-TAD) network that effectively integrates two sub-tasks to accomplish action detection in a single unified end-to-end framework. More specifically, a single pathway consists of two parallel recurrent modules, one learns proposal features, and the other learns classification features. Two features are then merged for final detection. However, these one-stage and two-stage methods obtained so far are not designed for online action detection as they need to look over the entire video before they can make decisions.

2.3 Online Action Detection

In online action detection, the decision is made based only on the knowledge from the current and the past frames. Geest et al. (2016) analyzed and compared three baseline methods in this problem. Besides the baseline methods, Gao et al. (2017a) proposed an encoder-decoder network with a reinforcement loss (RED) for action anticipation, aiming to detect action in the future. They view online action detection as a particular case for action anticipation, where the anticipation timestep is zero. Afterward, Xu et al. (2019) also introduce an LSTM-based encoder-decoder network named Temporal Recurrent Network (TRN). TRN sequentially processes video frames via TRN cells. But while RNN only models the dependency between the current and past information, decoders in the TRN cell anticipate the information in the next few time steps. Then they merge the predicted future information with the current and past information for detecting action.

Recently, Eun et al. (2020) try to learn a discriminative representation for identifying the current action. They designed a recurrent unit extended from Gated Recurrent Unit (GRU) (Cho et al., 2014), named Information Discriminative Unit (IDU). They added two main components in IDU. First, the reset and update modules in IDU additionally take the current information into account, considering whether the past information is relevant to an ongoing action. Second, they introduced the early embedding module to encourage the feature to be more high-level to effectively model the relevance. In summary, IDU decides whether to accumulate input information based on its relevance to the current action. It enables the network to learn a more discriminative representation.

3 Enhanced Relation Layer

The previous online action detection approaches (Gao et al., 2017a; Xu et al., 2019) take the entire feature sequence as input to learn the representation of current action. They ignore the fact that the whole input sequence usually contains not only the action of interest features but also some background features and features representing other actions, which is irrelevant to prediction of the current action. If we directly take the entire sequence as input, those irrelevant features will hamper the learning of features representing the current action and make the feature less discriminative. It is necessary to design a module that guides the network to focus on those features related to the action of interest.

The attention mechanism (Vaswani et al., 2017) plays an essential role in guiding the model to focus on the crucial areas and features. The network for processing sequential inputs, i.e. recurrent networks like LSTM (Donahue et al., 2015) and GRU (Cho et al., 2014), 1D CovNet, cannot extract the interaction between two particular features. On the contrary, the attention mechanism can learn the dependencies between any two arbitrary features in a sequence, even in the long temporal distance. Therefore, we design an Enhance Relation Layer (ERL) with an

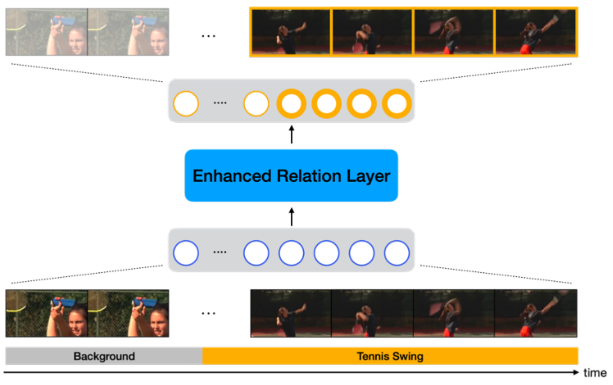

Fig. 2 The proposed Enhanced Relation Layer updates the feature sequence by the relevance between each feature and the action of interest. The relevant features are highlighted whereas the irrelevance ones are ignored.

attention mechanism to update the feature sequence by the relevance between each feature and the action of interest, as shown in Fig. 2.

3.1 Relevance Module

The objective of Relevance Module is to infer the relevance of each time-step to the current time-step in the feature sequence. The action of interest is, therefore, the feature x_0 at the current time-step. To learn the dependency between the feature at time-step x_t and the feature at the current time-step x_0, first of all, we embed x_0 into query, key, and value by:

$$q_0 = W^q x_0 \tag{1}$$

$$k_0 = W^q x_0 \tag{2}$$

$$v_0 = W^q x_0 \tag{3}$$

where W^q, $W^k \in R^{d^k \times d^x}$, $W^v \in R^{d^v \times d^x}$ are matrices with learnable parameters that transform feature x_0 into q_0, k_0, and v_0, representing query, key and value vector, respectively. d^x is the dimension of the features in the feature sequence, d^k is the dimension of query and key vectors, and d^v is the dimension of value vector. By the same embedding matrices, we can also embed x_t into q_t, k_t and v_t with $t \in [-T, -1]$ from the previous time-steps. The query vectors are designed for matching other vectors; the key vectors are for being matched; the value vectors contain the information to be extracted.

For time-step t in the sequence, we compute the relevance score r_t, representing the relevance of the feature x_t to the feature at the current time-step x_0 as follows:

$$r_t = \sigma((k_t \cdot q_0)/\sqrt{d^k}) \tag{4}$$

where σ is the logistic sigmoid function. We match the query of the current time-step and the key of time-step t by computing the dot product of the two vectors, divided by $\sqrt{d^k}$, and apply the sigmoid function to constrain the output within the range from 0 to 1. The purpose of scaling the dot product by $1/\sqrt{d^k}$ is to prevent the enormous value of the dot product from letting the output value of the sigmoid function mostly stay in the saturation regions.

The relevance score can indicate the correlation between the feature of the time-step t and the feature of the current time-step, which contains the action of interest. If the feature is more relevant to the current time-step feature, the relevance score should be closer to 1; otherwise, the relevance score should be closer to 0. After calculating the relevance score, the feature of time-step t is updated as follows:

$$x_t^{updated} = r_t \cdot v_t \tag{5}$$

by multiplying the relevance score with the value vector, which contains the information of feature x_t to be extracted. We can effectively pay attention to those relevant features and filter out the irrelevant ones (background and other actions).

In practical implementation, we do not compute the relevance scores and update the features one by one, which is time-consuming and ineffective. Instead, we update the entire feature sequence at once. As shown in Fig. 3, we stack all the input sequence features together as $X = [x_t]_{t=-T}^{0} \in R^{T \times d^x}$, where T is the temporal length of the feature sequence. Then we transform the input sequence into the key, and value space simultaneously as follows:

$$K = W^k X \tag{6}$$

$$V = W^v X \tag{7}$$

K, and V represent the key and value vectors of every time-step in the sequence, respectively, where $K \in R^{T \times d^k}$ and $V \in R^{T \times d^v}$. The relevance score vector is calculated as:

$$R = \sigma((K q_0^\top)/\sqrt{d^k}) \tag{8}$$

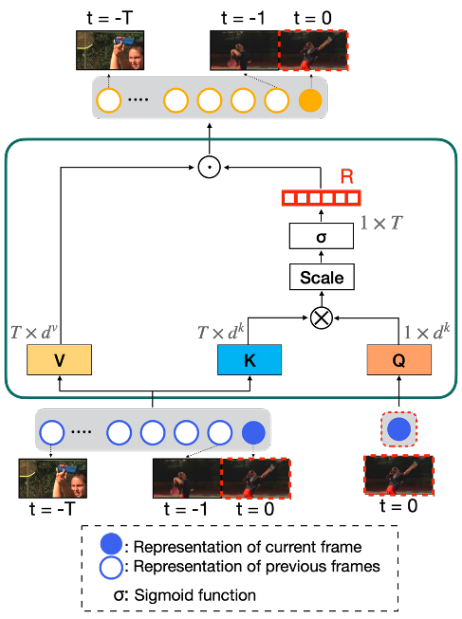

Fig. 3 The architecture of Relevance Module in Enhance Relation Layer.

where σ is the sigmoid function and $1/\sqrt{d^k}$) is the scaling factor. $R = \{r_t\}_{t=-T}^{0}$ is the set of the relevance scores, where r_t is the relevance score of time-step t . The relevance scores are further supervised to learn a more precise relation with the current time-step. To encourage the relevance score to be reliable, we define a relevance loss L_r based on $L1$ distance between r_t and the ground truth relevance for supervision as:

$$L_r = \frac{1}{T}\sum_{t=-T}^{0} |r_t - [\![y_t = y_0]\!]| \tag{9}$$

where y_t and y_0 are the ground truth action label for the tth time-step and the current time-step, respectively. If the label of tth time-step is the same as the current time-step, the ground truth relevance is defined as 1; otherwise, 0.

Finally, we calculate the output of the Relevance Module as:

$$X^{relevance} = [r_t V_t]^\top, \quad t \in [-T, 0] \tag{10}$$

where $V_t \in R^{d^v}$ is the value vector of time-step t.The output feature sequence $X^{relevance} \in R^{T\times d^v}$ is calculated by multiplying the relevance score and the value feature of every time-step in the sequence. In summary, the Relevance Module updates the feature sequence according to each time-step's relevance to the current time-step.

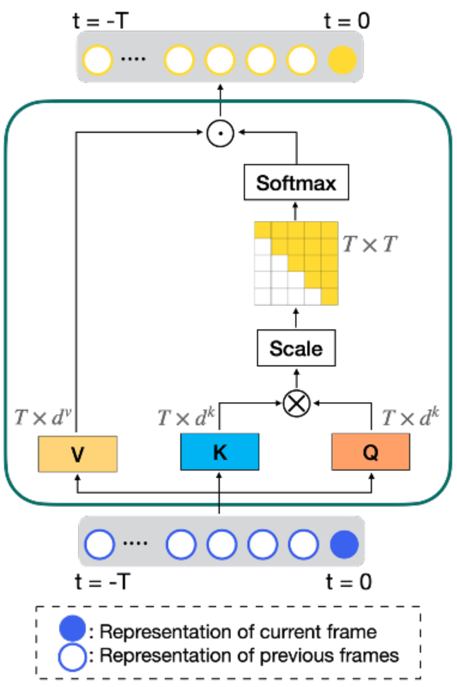

Fig. 4 The architecture of Enhanced Relevance Module in Enhance Relation Layer.

3.2 Enhanced Relevance Module

However, there will be a problem if we only take the current time-step into account to define the relevance score. The relevance score is not trustworthy when one of the two features (time-step t and the current time-step) is not representative enough. An intuitive idea to tackle this problem is to consider the set of consecutive time-steps between the current time-step, where the action of interest is now taking place, and time-step t. The reason we choose the frames between them is that actions are normally continuous. There is a high probability that these frames may also contain the action of interest. If one of the two frames (time-step t and current) is not representative enough, the frames between them can reduce the impact caused by the incorrect relevance score.

We design attention with enhanced relevance to realize the aforementioned idea. As depicted in Fig. 4, we project the feature sequence into query, key, and value space, the same as what we do to compute the relevance score. Then, we take the dot product of the query Q and key K sequence to obtain an $T \times T$ score map W as follows:

$$W = KQ^\top/\sqrt{d^k} \tag{11}$$

where the score $W_{i,j}$ in the map indicates the dependency between two time-steps i and j. However, for each time-step, we are focusing on the time-steps from it to the current time-step. Therefore, we extract the upper triangular part of W to meet the following:

$$W_{upper} = \begin{cases} W_{i,j}, & i \leq j \\ 0, & i > j \end{cases} \tag{12}$$

where $i, j = 1, 2, \ldots, T$. This operation can shield the weights of the time-steps that we are not interested in. Finally, we apply a softmax function in each row to normalize W_{upper} to get the final score $W_{upper'}$. Note that the total contribution of the frames is equal to one by normalizing the row. Given the final score $W_{upper'}$, we can get the enhanced relevance feature by:

$$\begin{aligned} X^{enhance} &= W_{upper'}V \\ &= softmax(triu(KQ^{\top}/\sqrt{d^k})) \end{aligned} \tag{13}$$

where $triu()$ is the operation of extracting only the upper triangular part of a matrix, and V is the value sequence.

3.3 Actioness Module

Besides computing the relevance of a frame at certain time-step to the frame at the current time-step, we also want to filter out the background frames more thoroughly. Therefore, we design an additional attention mechanism, named Actioness Module, to estimate the actioness score, which indicates the probability of a frame from an occuring action. The sequence is then filtered based on the actioness score. Fig. 5 shows the architecture of the Actioness Module. Like the Enhanced Relevance Module, we first acquire query Q, key K, and value sequence V from the input feature sequence. Then, we compute the weight matrix W by dot product and scaling Q and V. Given the weight matrix, we take the sum of the weights of each column of W to indicate the importance level of each time-step, so we can get the actioness scores vector as follows:

$$A = \sigma(\sum_{i=-T}^{0} W_i) \tag{14}$$

where the sigmoid function constrains the value between 0 to 1, and $A = \{a_t\}_{t=-T}^{0}$ is the set of the actioness scores, where a_t denotes the probability of action which takes place at time step t. If an action takes place in the frame at time-step t, a_t should be closer to 1; otherwise, a_t should be closer to 0.

Similar to how we specify the relevance score, we define an actioness loss L_a through which the actioness score can be made more accurate. The actioness loss L_a is based on $L1$ distance between a_t and the ground truth actioness score:

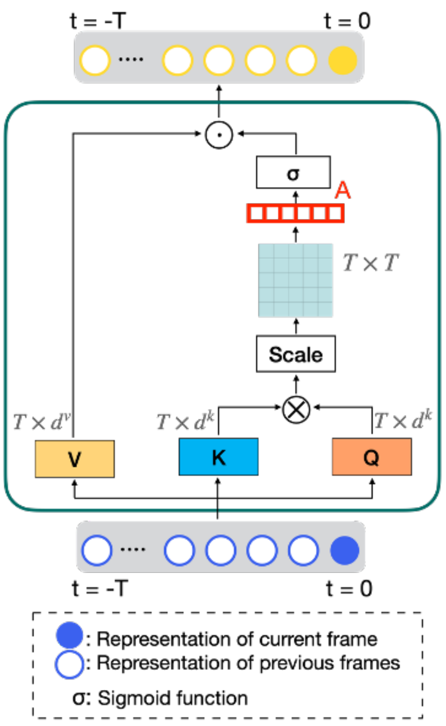

Fig. 5 The architecture of Actioness Module in Enhance Relation Layer.

$$L_a = \frac{1}{T} \sum_{t=-T}^{0} |a_t - [\![y_t = 0]\!]| \tag{15}$$

where y_t is the ground truth action label for the tth time-step. If the label of tth time-step is zero, which denotes the background, the ground truth actioness score is defined as 0; if the label is other than zero, the ground truth actioness score is 1. Then, the feature sequence is updated by the actioness score as follow:

$$X^{actioness} = [a_t V_t]^{\top}, \quad t \in [-T, 0] \tag{16}$$

where $V_t \in R^{d^v}$ is the value vector of time-step t.The output feature sequence $X^{actioness} \in R^{T \times d^v}$ is calculated by multiplying the actioness score and the value feature of every time-step in the sequence. In summary, the Actioness Module updates the feature sequence according to the learned actioness score.

3.4 Combination with Convolution Networks

After we have the three attention modules: Relevance, Enhance Relevance, and Actioness. The next question

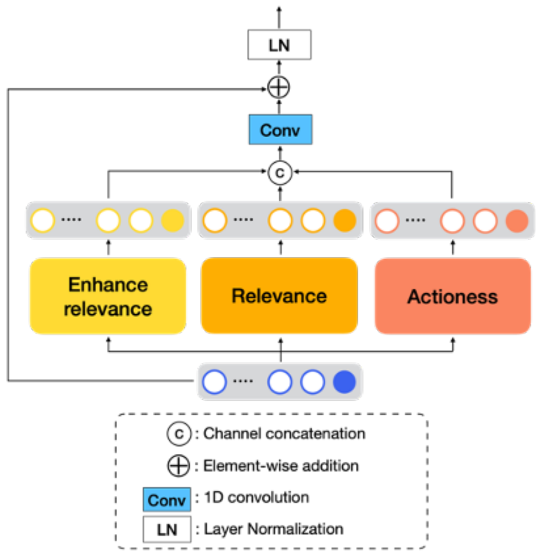

Fig. 6 The network architecture of Enhanced Relation Layer. Multi-head attention combines the three designed attention modules.

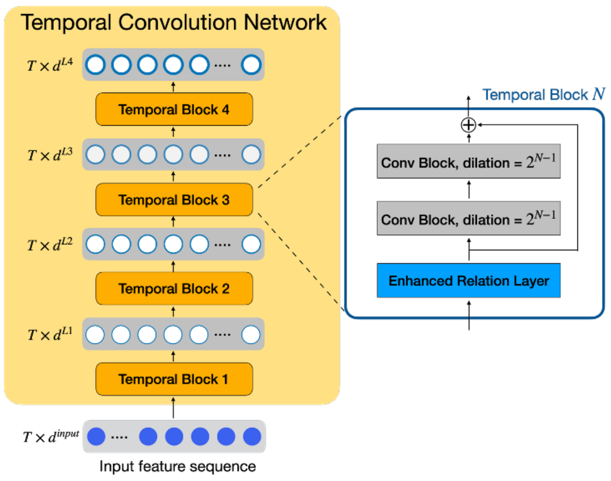

Fig. 7 We adopt TCN as our backbone model. d^{input} is the dimension of the input features, and d^{L_n} denotes the dimension of the output features of the nth Temporal Block. We design a TCN with four blocks. Each block consists of our proposed Enhanced Relation Layer following by two casual convolutions with dilation equal to 2^{n-1}.

is how we combine these three modules. Multi-head attention (Vaswani et al., 2017) allows the model to jointly attend to information from different representation subspaces at different positions. We perform the three attention mechanisms in parallel, yielding three $T \times d^v$ dimensional output feature sequences. With the three feature sequences, we concatenate them along the channel dimension. The concatenated feature sequence then undergoes a one-dimensional convolutional layer with kernel size equal to 1 to reduce the channel dimensions, as depicted in Fig. 6 Besides, we add the input sequence to the output of the multi-head attention via a residual connection, and then apply layer normalization to the sum. We name this multi-head attention as the complete Enhanced Relation Layer. The equation of Enhanced Relation Layer can be written as follows:

$$ERL(X) = LN(X + Multihead(X)) \tag{17}$$

$$\begin{aligned} Multihead(X) = \\ & conv1d([[X^{relevance}, X^{enhance}, X^{actioness}]]) \end{aligned} \tag{18}$$

where $conv1d$ is one-dimensional convolution with kernel size equals to 1 to reduce the dimension of the features. With the Enhanced Relation Layer, we can filter out the irrelevant information and emphasize the relevant information contained in a sequence. The next important issue is finding out how to incorporate the Temporal Relation Layer into the backbone model to improve the performance.

We select the Temporal Convolutional Network (TCN) as our backbone model. The convolutions in TCN are one-dimensional, shifting along the sequence's temporal dimension to capture the temporal dependencies. Besides, the convolution is casual convolution. The output of the process of features up to time-step t only involves features from features at time-step t and earlier, not later. The causal convolution fits well with the setting of online action detection, where the information comes only from the past and present, without any future information.

We design a Temporal Convolution Network with four blocks (as shown in Fig. 7). Each block consists of two one-dimensional causal convolutions, with dilation 2^{n-1}, where n denotes the nth block in TCN. We increase the dilation with the depth of the network to effectively expand the receptive field. In the output of the first block, the receptive field is one. As the sequence keeps passing through the network's depth, the receptive field will ultimately equal the temporal length of the input sequence. By design, the final output feature at time-step t will be based on the information from time-step $t-T$ to t, which covers the entire length of the input sequence. We insert the Enhanced Relation Layer (ERL) in front of the convolution layers. The input sequence of each block first goes through the Enhanced Relation Layer to filter out irrelevance information, and then captures the temporal pattern from the attentive feature sequence by the convolutional layers.

4 Temporal Pyramid Network

In the convolutional networks, feature maps (two-dimensional) and feature sequence (one-dimensional) in different layers inherently have different receptive fields. Features in the deeper layers have larger receptive fields as compared to those in shallower layers. Utilizing feature maps from multiple layers to detect objects in various scales is a commonly used concept in the object detection task. However, this idea has not yet been applied to action detection. Especially, in online action detection, we do not know when the action starts in a sequence. As depicted in Fig. 1, for an identical action instance, the length of the action we observed could be different, considering the observation time point. For the above reason, we attempt to use multiple feature sequences from the Temporal Convolution Network to detect actions with different temporal lengths.

However, the semantic representation capability of the feature sequences from the shallower layers is weak. In other words, the features from the shallower layers are not discriminative enough to distinguish an action. Inspired by Feature Pyramid Network (Lin et al., 2017b), we create a top-down pathway to transfer the strong semantic ability of the higher-level sequences also to the lower-level sequences. Accordingly, we can acquire multiple levels of feature sequence, all with rich semantics. It allows us to detect actions with various temporal lengths effectively. We name this structure the Temporal Pyramid Network (TPN).

We illustrate the architecture of the Temporal Pyramid Network in Fig. 8. Our goal is to leverage the feature sequence hierarchy of the designed Temporal Convolution Network (Section 3.3.4), where semantic ability goes from low to high as the layer goes deeper. We aim to obtain high-level semantic features throughout each layer by building a pyramid structure. The Temporal Pyramid Network structure is composed of a bottom-up pathway (left half of TPN) and a top-down pathway (right half of TPN).

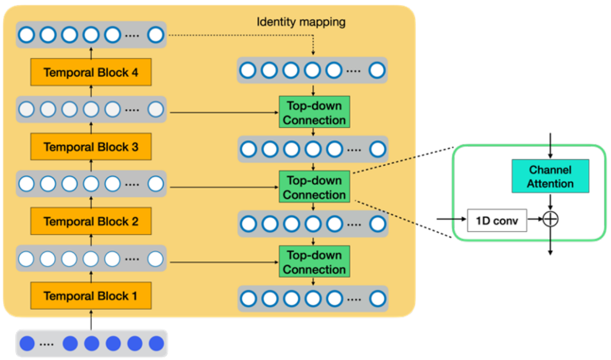

Fig. 8 The architecture of Temporal Pyramid Network. The strong semantic ability of the higher-level feature sequences is transferred to the lower-level by top-down connections.

4.1 Bottom-up Pathway

The bottom-up pathway is the backbone Temporal Convolution Network (TCN) with relation layers (see Section 3.3). TCN inherently consists of a feature sequence of several temporal scales. Every sequence has the same length since TCN keeps the output of each layer with the same size. We define a temporal block as one pyramid stage and choose the output of each block's output as the feature sequence to construct the overall pyramid. The sequence will be updated through a block with the Enhanced Relation Layer and two one-dimensional convolutional layers. We denote the output of each temporal blocks as $\{C_1, C_2, C_3, C_4\}$, where $C_i, 1 \leq i \leq 4$, are with dilations of 1, 2, 4, and 8, receptively. Note that the receptive field is more enlarged, and the semantic ability is getting stronger from C_1 to C_4, as shown in Fig. 8, where circles with thicker edges indicate the richer semantic features.

4.2 Top-down Pathway

The top-down pathway fuses the temporally coarser, but semantically stronger, feature sequence from the higher pyramid level with the temporally finer, but semantically weaker, feature sequence from the lower level. Top-down connections do the fusion process. The entire procedure of constructing the top-down pathway starts with C_4, the output of the last Temporal Block in the bottom-up pathway.

We take C_4 as the initial sequence of the top-down pathway. With the initial sequence with coarse temporal resolution but strong semantic features, we first apply channel-wise attention to the higher-level sequence instead of directly adding the lower-level sequence to it. Channel-wise attention is proven to have the ability to lead the model to focus on more discriminative features. The channel-wise attention is illustrated in Fig. 9. After receiving the feature sequence, we pass it to Temporal Average Pooling and Temporal Max Pooling, and both of them are followed by two fully-connected layers. We add the average attention feature and the max attention feature, and then apply the sigmoid function to obtain the final channel attention feature. The values in the attention feature are made between 0 to 1 because of the sigmoid function. If one channel matches the better feature, the corresponding position in the attention feature will return a value closer to 1, otherwise 0. Finally,

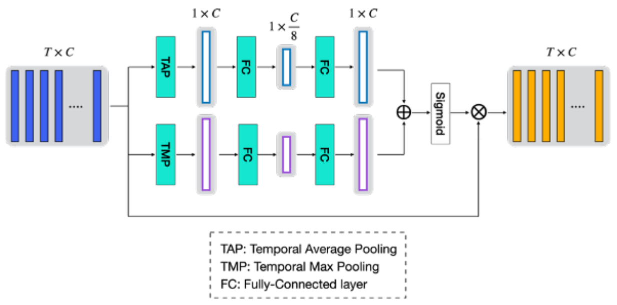
T × C
1 × C
1 × C/8
1 × C
TAP
TMP
FC
FC
FC
FC
Sigmoid
T × C
TAP: Temporal Average Pooling
TMP: Temporal Max Pooling
FC: Fully-Connected layer

Fig. 9 The illustration of the channel-wise attention.

we multiply the attention feature to the original input sequence.

The attentive feature sequence is then merged with the lower-level sequence by element-wise addition. It is worth noting that the lower-level sequence additionally undergoes a one-dimensional convolution with kernel size equal to one, where the dimension of the resulting lower-level features is not the same as the attentive features. The above-mentioned process is iterated until the most subtle resolution sequence is generated. This final set of feature sequences is denoted as $\{P_1, P_2, P_3, P_4\}$ corresponding to $\{C_1, C_2, C_3, C_4\}$, i.e., P_i and C_i are of the same level, $i = 1, \ldots, 4$.

4.3 Multi-layer Output

As mentioned before, action instances in a video usually have various temporal scales. We build the Temporal Pyramid Network with Enhanced Relation Layers to learn the inherent temporal patterns of the sequence. Output feature sequences in different depths in convolution networks have different receptive fields. Furthermore, we construct a pyramid architecture passing the rich semantic information from the higher level to the lower level. Thus, we can obtain sequences with multiple receptive fields, all with rich semantic ability, to facilitate learning of different scale patterns.

Inspired by multi-scale object detection networks (Hariharan et al., 2017), it would be better to combine outputs of multiple layers to make the detection rather than taking only one output feature sequence. Since the output sequences in the top-down layer of the Temporal Pyramid Network have the same length, as shown in Fig. 10, it is convenient to directly concatenate $\{P_1, P_2, P_3, P_4\}$ to form a *multi-temporal-scale* feature f_t^{multi} to enhance the prediction further. The *multi-temporal-scale* feature can distinguish actions of various lengths. We feed the *multi-temporal-scale* feature into a fully connected layer to obtain the final probability distribution $p_t \in R^{K+1}$ overall K action classes and

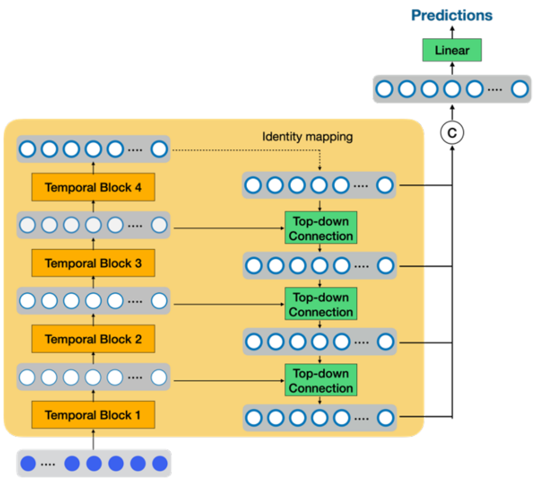

Fig. 10 Temporal Pyramid Network using *multi-temporal-scale* feature sequences to predict action categories.

background of an ongoing action as explained below. Here, *multi* is an abbreviation for *multi-temporal-scale.*

$$p_t = softmax(W^p f_t^{multi}) \tag{19}$$

where W^p is a trainable matrix that converts f_t^{multi} to $K+1$ dimension. We define a classification loss L_c for the *multi-temporal-scale* feature sequence by employing the standard cross-entropy loss as

$$L_c = -\sum_{t=-T}^{0} \sum_{k-0}^{K} [\![y_t = k]\!] \log p_t^k \tag{20}$$

where y_t is the ground truth action label for the tth time-step, and p_t^k is the kth element of p_t, which indicates the probability of the frame of time-step t belonging to kth action class.

5 Experiments

In this chapter, we describe the implementation details in this paper. Before the experimental results, we give the introductions of two public action detection datasets and the two evaluation metrics. In order to proof that every part of proposed method is useful, we design a series of the ablation studies. In the end, we compare our method with various state-of-the-art ones on two public datasets, including THUMOS-14 (Jiang et al., 2014) and TVSeries (Geest et al., 2016). The quantitative experimental results confirm the advantages of the proposed Enhanced Relation Layers and Temporal Pyramid structure.

5.1 Training Details

We follow the same setting as the state-of-the-art methods (Gao et al., 2017a; Xu et al., 2019; Eun et al., 2020) to formulate the online action detection problem. On both THUMOS-14 and TVSerires datasets, we extract video frames at 24 fps and set the number of frames in each chunk N to 6. We take the current chunk and 63 past chunks (i.e., $T = 63$) to form an input sequence for TPN, which is 16 seconds long.

Among several two-stream architectures, we select Temporal Segment Network (TSN) (Wang et al., 2016) pretrained on the Kinetics dataset (Carreira and Zisserman, 2017) as our feature extractor. TSN consists of Inception with Batch Normalization (BN-Inception) (Ioffe and Szegedy, 2015) for both appearance and motion networks. We use the output of the *global_pool* layer in BN-Inception as the appearance feature x_t^a and a motion feature x_t^m, both with 1024 dimensions. The concatenated feature representation x_t is then a 2048-dimension feature.

To train our TPN, the total loss is the sum of relevance losses L_r, actioness losses L_a in each layer of the bottom-up pathway of TPN, and the final classification loss. The equation is listed as:

$$L_{total} = L_c + \alpha \sum_{l=1}^{L} L_{r,l} + \beta \sum_{l=1}^{L} L_{a,l} \tag{21}$$

where L is the number of layers in the backbone TCN. α and β are balance parameters, which both are set to 0.3 during training.

We update TPN by using stochastic gradient descent with a mini-batch of 64 samples to minimize the L_{total} in Eq. 21. Through backpropagation, the network is updated by Adam optimizer (Kingma and Ba, 2015) with an initial learning rate equals to 1x10-4, and the training process is monitored with early stopping.

In the inference process, TPN is going to output a $multi-temporal-feature$ sequence, which represents the frames from timestep $-T$ to the current timestep. We only take the feature of the current timestep into account, since the goal is to detect the action that takes place in the current chunk. An action probability distribution of the current chunk is then generated by the fully-connected layer.

5.2 Action Detection Dataset

To verify the performance of the proposed method, we evaluate two benchmark datasets: THUMOS-14 (Jiang et al., 2014) and TVSeries (Geest et al., 2016). Both of them are challenging datasets and are commonly used in the online action detection task. Besides, we introduce two evaluation protocols suitable for the online action detection task.

5.2.1 THUMOS-14 Dataset

THUMOS-14 is a large and realistic dataset introduced by the THUMOS detection challenge (Jiang et al., 2014). This challenge contains two parts, action recognition and temporal action detection. The temporal action detection part of THUMOS-14 contains a total of over 20 hours of videos from 20 sport classes.

The training set of THUMOS-14 contains only trimmed videos, which are not suitable for action detection. However, there are 200 and 213 untrimmed videos in the validation set and the test set, respectively. Thus, we use the validation set for training and the test set for evaluation. Every video has, on average, 15.8 action instances, each with temporal annotations of starting time and ending time of the action. Several reasons cause challenges. Each video contains 71% of background data, which are dominant in each video. Different actions may take place at the same time. Moreover, there are many shot-cuts in the video. Actions possibly extend over multiple shots, where the viewpoint of one action instance can suddenly change.

5.2.2 TVSeries Dataset

TVSeries dataset is introduced in (Geest et al., 2016). The videos in this dataset depict realistic actions as they take place in real life. The videos are from the first episodes of six TV series. Each video is around 150 minutes, with a total of about 16 hours, and is divided into training, validation, and testing set. Every set contains at least one episode of every series. Having different series in training and testing sets would introduce a domain shift, which makes the dataset challenging.

The 30 action classes defined in this dataset and the number of instances of each class are shown in Table 4 2. The total number of action instances is 6231. The annotation contains start and end frames of all instances, and action instances can be overlapping in time. There is a large variability in this dataset. First, there are multiple actors, and everyone does an action in different ways. Second, different actions can take place at the same time, performed by the same or multiple actors. Third, the way the action is recorded can be very different from one action to another, where the viewpoint is not fixed. Fourth, this dataset also contains a highly varying background, and background data dominates the video.

Table 1 The results of Enhanced Relation Layer (ERL) with different composition.

Methods	THUMOS-14	TVSeries
	mAP(%)	cmAP(%)
w/o ERL	48.7	82.1
Relevance	49.9	82.9
Actioness	50.2	83.0
E-Relevance	51.9	83.5
E-Relevance + Actioness	52.8	84.2

5.2.3 Evaluation Metrics

In (Geest et al., 2016), they also propose an evaluation protocol for online action detection, in which a decision needs to be made at every frame. For every action, we want to know how likely is it that the action is going on in that frame, based on the available information (current and past frames). Therefore, it is logical to use the average precision over all frames as a metric for the performance of the online action detector.

For each action class, all frames are sorted in descending order according to their probability of this action class. Then, we calculate the precision of the class at a cut-off value according to the following equation:

$$\text{Prec(k)} = \frac{TP(k)}{TP(k) + FP(k)} \tag{22}$$

where $TP(k)$ and $FP(k)$ are the numbers of true positive and false positive at a cut-off k, respectively. The ground-truth label of the frame defines true positive and false positive. If the ground-truth label is the same as the action class, it is a true positive, otherwise, it is a false positive. The average precision (AP) of a class is defined as:

$$\text{AP} = \frac{1}{N} \sum_{k} \text{Prec(k)}\, I(k) \tag{23}$$

where $I(k)$ as an indicator function equal to 1 if the frame at cut-off k is a true positive, and equal to 0 otherwise. N_P is the total number of positive frames. Finally, mean average precision (mAP) is the mean of the AP value over all classes, which is formulated by:

$$\begin{aligned} \text{mAP} &= \frac{1}{N_c} \sum_{n=1}^{N_c} \text{AP}_n\, I(k) \\ &= \frac{1}{N_c} \sum_{n=1}^{N_c} \frac{1}{N_P} \sum_{k} \text{Prec(k)}\, I(k) \end{aligned} \tag{24}$$

where N_c is the number of action classes, and AP_n is the average precision for action class n. Mean average precision is then the final performance metric of an online action detection method, which quantifies how well the model performs at every frame. Besides, mean average precision is very sensitive to the sorting result in the very first step. In other words, frames with a higher probability score contribute more to the average than those with a lower score.

However, mAP is sensitive to changes in the ratio of positive (action) frames versus negative (background) frames due to differences in contents of different videos. If a video has relatively more background frames, the probability of the model falsely detects some background frames with higher confidence than some true positives will increase. Thus, AP will decrease, making it hard to compare two different classes' average precision if they do not have the same ratio of positive versus negative. To enable a fair comparison, Geest et al. (Geest et al., 2016) proposed the calibrated precision at a cut-off k as:

$$\text{cPrec(k)} = \frac{\omega TP(k)}{\omega TP(k) + FP(k)} \tag{25}$$

where ω is a ratio between negative frames and positive frames, such that the total weight of the negatives becomes equal to the total weight of the positives. Similar to the average precision, the calibrated average precision (cAP) of a class can be computed as :

$$\text{cAP} = \frac{1}{N_P} \sum_{k} \text{cPrec(k)}\, I(k) \tag{26}$$

where $I(k)$ is the indicator function the same as it in AP, and N_P is the total number of positive frames. Then, the mean calibrated average precision (cmAP) is obtained by averaging the cAP values over all action classes:

$$\begin{aligned} \text{cmAP} &= \frac{1}{N_c} \sum_{n=1}^{N_c} \text{cAP}_n\, I(k) \\ &= \frac{1}{N_c} \sum_{n=1}^{N_c} \frac{1}{N_P} \sum_{k} \text{cPrec(k)}\, I(k) \end{aligned} \tag{27}$$

where N_c is the number of action classes, and cAP_n is the calibrated average precision for action class n.

5.3 Ablation Study

To validate the effectiveness of each component in our proposed method for online action detection, we conduct a series of comprehensive ablation studies of them in the following paragraph.

5.3.1 The Results of Enhanced Relation Layer

To verify that the Enhanced Relation Layers (ERL) we design are effective, we demonstrate some experiments, as shown in Table 1. First, we train a baseline model, a general Temporal Convolution Network without any attention mechanism, along with two models with relevance attention and actioness attention, respectively. We can see improvements of two attentions on both datasets.

We then substitute the relevance attention to enhance relevance by designing an additional enhanced-relevance (e-relevance) attention alongside. As we can see, comparing with the relevance attention alone, adding e-relevance improves the results in mAP by 2.0 in THUMOS-14 and cmAP by 0.6 in TVSeries. These results verify the effectiveness of enhanced relevance.

Lastly, we add the three attention mechanisms (relevance, e-relevance, and actioness) to consider. This version of the model can discriminate the importance of a frame by relevance to the current frame and the probability of action occurs. The results show that this design can achieve the best performance with mAP of 52.8 in THUMOS-14 and 84.2 of cmAP in TVSeries, comparing only to consider relevance or actioness. From then on, we use this design as the default structure of the Temporal Relation Layer.

5.3.2 The Results of Temporal Pyramid Network

As we showed the Temporal Relation Layer's utility, we will analyze the effectiveness of the other proposed method, the temporal pyramid architecture, in this subsection. The primary purpose of the temporal pyramid is to pass the strong semantic ability of the feature sequence from the higher-level to the lower-level, then concatenate those feature sequences to detect action with different lengths. For comparison, we train a general Temporal Convolution Network with multi-layer output.

Table 2 shows the experimental results. To be mentioned, every network is without the ERLs in backbone TCN. TPN basic represent Temporal Pyramid Network (TPN) without channel-wise attention in the top-down connection. Comparing it with TCN, the result slightly improves from 48.7 to 49.8 in THUMOS-14 and from 82.1 to 83.0 in TVSeries. This result verifies the effectiveness of *multi-temporal-scale* features output by TPN, which we use these features to detect action with different lengths.

Table 2 The effectiveness of the proposed Temporal Pyramid Network (TPN), comparing with the Temporal Convolution Network (TCN).

Methods	THUMOS-14	TVSeries
	mAP(%)	cmAP(%)
TCN	48.7	82.1
TPN basic	49.8	83.0
TPN w/ channel attention	51.2	83.3

Table 3 The effectiveness of the proposed methods. TCN refers to Temporal Convolution Network; ERL refers to Enhanced Relation Layer; TPN refers to Temporal Pyramid Network.

Methods	THUMOS-14	TVSeries
	mAP(%)	cmAP(%)
TCN (baseline)	48.7	82.1
TPN w/ ERL	52.8	84.2
TPN	52.0	83.3
TPN w/ ERL (ours)	53.8	85.4

5.3.3 The effect of all proposed method

We perform the following experiments to validate the effectiveness of the two parts of our proposed method, namely, Temporal Relation Layer (TRL) and Temporal Pyramid Network (TPN), as shown in Table 3.

First of all, we define the general Temporal Convolution Network (TCN), without any additional module, as the baseline model. The results in mAP / cmAP are 48.7 / 82.1 in THUMOS-14 and TVSeries, respectively.

Next, we train two models with each element separately. One is TCN with the Enhanced Relation Layer (ERL), and the other is Temporal Pyramid Network (TPN), i.e., TCN with the pyramid structure. The experimental results show that both ERL and TPN can effectively improve the baseline, validating the effectiveness of the proposed ERL and TPN. Comparing with the baseline, adding ERL can gain 4.1 mAP in THUMOS-14 and 2.1 cmAP in TVSeries, verifies ERL's ability to filter irrelevance frames and focus on the relevant ones. In TPN, the results improve the baseline with 3.3 mAP in THUMOS-14 and 1.2 cmAP in TVSeries. This improvement confirms the idea of using $multi-temporal-scale$ features to detect actions with different lengths.

Finally, when applying two designs simultaneously, we reach the best results, which means adding TRL into TPN is complementary and can tackle different problems. With the combination of TRL and TPN, the results achieve the highest on both THUMOS-14 and TVSeries dataset with 53.8 mAP and 85.4 cmAP, re-

Table 4 Comparison of the state-of-the-art methods on the THUMOS-14

Methods	Setting	mAP(%)
CNN (Geest et al., 2016)	off-line	34.7
LSTM(Geest et al., 2016)	off-line	39.3
CDC (Shou et al., 2017)	off-line	44.4
RED (Gao et al., 2017a)	online	45.3
TRN (Xu et al., 2019)	online	47.2
IDN (Eun et al., 2020)	online	50.0
ours	online	**53.8**

Table 5 Comparison of the state-of-the-art methods on the TVSeries

Methods	Setting	mAP(%)
CNN (Geest et al., 2016)	off-line	60.8
LSTM(Geest et al., 2016)	off-line	64.1
CDC (Shou et al., 2017)	off-line	-
RED (Gao et al., 2017a)	online	79.2
TRN (Xu et al., 2019)	online	83.7
IDN (Eun et al., 2020)	online	84.7
ours	online	**85.4**

spectively. We assign TPN with TRL as our final proposed method.

5.4 Online Action Detection Result

We compares the performance of our approach with those of several state-of-the-art methods published in the literature. The following are the comparison results on THUMOS-14 dataset and TVSeries dataset.

5.4.1 The Result of THUMOS-14 dataset

To evaluate our proposed method, we follow the evaluation metrics used in previous works (Gao et al., 2017a; Xu et al., 2019; Eun et al., 2020) and report the mAP in percentage terms. In the experiment, we train on the validation set with 200 untrimmed videos and evaluate on the test set with 213 untrimmed videos.

Table 4 shows the performance of several methods, which are all deep learning-based. Except for CNN based approach, all of the methods utilize a sequential learning module, C3D for (Shou et al., 2017), LSTM for (Gao et al., 2017a; Xu et al., 2019), and GRU for (Eun et al., 2020). We can see that using a sequential learning module achieve higher performances because an action is composed of a series of frames. Sequential information plays a crucial role when identifying actions. RED (Gao et al., 2017a) and TRN (Xu et al., 2019) achieve high performance by anticipating the unknown future information, and IDN (Eun et al., 2020) update the gates in GRU by considering the relevance to the current timestep, achieving the recently best performance.

5.4.2 The Result of TVSeries Dataset

We follow the evaluation rules on the TVSeries. Instead of mAP, we use cmAP as the evaluation metric. The quantitative results are shown in Table 5. Our method, Temporal Pyramid Network with Temporal Relation Layers, outperforms the state-of-the-art methods. Similar to the results in THUMOS-14, the CNN-based method has the lowest performance on the TVSeries. Those methods that specifically target online settings have outstanding performances since the learned features are more discriminative.

Another observation from the experimental results is that, as compared to our results on the dataset THUMOS-14, the performance boost is not greater than that for the dataset TVSerires. A logical reason is that, on TVSeries dataset, the average length of the action in the video is about 2.0 seconds, which is relatively shorter as compared with the average length of the action in THUMOS-14 dataset, which is about 4.0 seconds on average. Thus, the impact of the $multi-temporal-scale$ generated by TPN will not be as obvious as before.

6 Conclusion

In this paper, we proposed a novel online action detection system that takes a video sequence as input and aims to identify actions that take place in the current frame. A deep Convolutional Neural Network (CNN) is utilized to not only extract *multi-temporal-scale* features, but also model the relations between the frame of action of interest and the frame at each timestep.

To learn a more discriminative feature, we propose an Enhanced Relation Layer (ERL), which is composed of three attention mechanisms. ERL can effectively filter out irrelevant and background frames, guiding the detection model to focus on frames relevant to the action of interest. We integrate ERLs into the backbone Temporal Convolution Network (TCN) to mine the temporal dependencies. Besides, we define TCN with ERLs as the bottom-up pathway of our Temporal Pyramid Network (TPN). The top-down pathway of TPN can transfer the strong semantic ability from the higher-level feature to the lower-level. We then fuse features from every level to obtain the *multi-temporal-scale* feature, which helps detect action with different lengths.

The comprehensive ablation studies validate the effectiveness of our proposed methods, ERL and TPN. The experimental results of our method achieve 53.2 in mAP on the THUMOS-14 dataset and 83.4 in cmAP on TVSeries, which outperforms most state-of-the-art methods. Our method takes a feature sequence as input and model the temporal relations between frames at different time-steps, which can support any kind of feature extractor. The results show that our complete action detection system might be able to be utilized for those applications needs detecting humans on the spatial domain to identify their actions.

References

Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint* arXiv:1803.01271

Brox T, Bruhn A, Papenberg N, Weickert J (2004) High accuracy optical flow estimation based on a theory for warping. In: *European conference on computer vision (ECCV)*, Springer, pp 25–36

Carreira J, Zisserman A (2017) Quo vadis, action recognition? a new model and the kinetics dataset. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp 6299–6308

Cho K, Merrienboer BV, G¸lÁehre a, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint* arXiv:1406.1078

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. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp 2625–2634

Eun H, Moon J, Park J, Jung C, Kim C (2020) Learning to discriminate information for online action detection. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp 809–818

Feichtenhofer C, Fan H, Malik J, He K (2019) Slowfast networks for video recognition. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp 6201–6211

Gao J, Yang Z, Nevatia R (2017a) Red: Reinforced encoder-decoder networks for action anticipation. *arXiv preprint* arXiv:1707.04818

Gao J, Yang Z, Sun C, Chen K, Nevatia R (2017b) Turn tap: Temporal unit regression network for temporal action proposals. In: *Proceedings of the IEEE international conference on computer vision (ICCV)*, pp 3648–3656

Geest RD, Gavves E, Ghodrati A, Li Z, Snoek CGM, Tuytelaars T (2016) Online action detection. *arXiv preprint* arXiv:0902.0885

Hariharan B, Arbeláez P, Girshick R, Malik J (2017) Object instance segmentation and fine-grained localization using hypercolumns. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39(4):627–639

Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *International conference on machine learning (ICML)*, PMLR, pp 448–456

Ji S, Xu W, Yang M, Yu K (2012) 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(1):221–231

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/

Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. *arXiv preprint* arXiv:1412.6980

Lin T, Zhao X, Shou Z (2017a) Single shot temporal action detection. In: *Proceedings of the 25th ACM international conference on Multimedia*, pp 988–996

Lin T, Zhao X, Su H, Wang C, Yang M (2018) Bsn: Boundary sensitive network for temporal action proposal generation. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp 3–19

Lin T, Liu X, Li X, Ding E, Wen S (2019) Bmn: Boundary-matching network for temporal action proposal generation. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp 3889–3898

Lin TY, Doll·r P, Girshick RB, He K, Hariharan B, Belongie SJ (2017b) Feature pyramid networks for object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp 2117–2125

Shou Z, Chan J, Zareian A, Miyazawa K, Chang S (2017) Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp 5734–5743

Simonyan K, Zisserman A (2014) Two-stream convolutional networks for action recognition in videos. *arXiv preprint* arXiv:1406.2199

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

Wang L, Xiong Y, Wang Z, Qiao Y, Lin D, Tang X, Gool L (2016) Temporal segment networks: Towards good practices for deep action recognition. *arXiv preprint* arXiv:1608.00859

Xiong Y, Zhao Y, Wang L, Lin D, Tang X (2017) A pursuit of temporal accuracy in general activity detection. *arXiv preprint* arXiv:1703.02716

Xu M, Gao M, Chen Y, Davis L, Crandall DJ (2019) Temporal recurrent networks for online action detection. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp 5532–5541