MS-TCT: Multi-Scale Temporal ConvTransformer for Action Detection

Rui Dai\textsuperscript{1,2}, Srijan Das\textsuperscript{3}, Kumara Kahatapitiya\textsuperscript{3}, Michael S. Ryoo\textsuperscript{3}, François Brémont\textsuperscript{1,2}
\textsuperscript{1}Inria \textsuperscript{2}Université Côte d’Azur \textsuperscript{3}Stony Brook University
\textsuperscript{1,2}\{name.surname\}@inria.fr

Abstract

Action detection is a significant and challenging task, especially in densely-labelled datasets of untrimmed videos. Such data consist of complex temporal relations including composite or co-occurring actions. To detect actions in these complex settings, it is critical to capture both short-term and long-term temporal information efficiently. To this end, we propose a novel ‘ConvTransformer’ network for action detection: MS-TCT\textsuperscript{1}. This network comprises of three main components: (1) a Temporal Encoder module which explores global and local temporal relations at multiple temporal resolutions, (2) a Temporal Scale Mixer module which effectively fuses multi-scale features, creating a unified feature representation, and (3) a Classification module which learns a center-relative position of each action instance in time, and predicts frame-level classification scores. Our experimental results on multiple challenging datasets such as Charades, TSU and MultiTHUMOS, validate the effectiveness of the proposed method, which outperforms the state-of-the-art methods on all three datasets.

1. Introduction

Action detection is a well-known problem in computer vision, which is aimed towards finding precise temporal boundaries among actions occurring in untrimmed videos. It aligns well with real-world settings, because every minute of a video is potentially filled with multiple actions to be detected and labelled. There are public datasets \cite{charades, tsu, multi-thumos} which provide dense annotations to tackle this problem, having an action distribution similar to the real-world. However, such data can be challenging, with multiple actions occurring concurrently over different time spans, and having limited background information. Therefore, understanding both short-term and long-term temporal dependencies among actions is critical for making good predictions. For instance, the action of ‘taking food’ (see Fig. 1) can get context information from ‘opening fridge’ and ‘making sandwich’, which correspond to the short-term and long-term action dependencies, respectively. Also, the occurrence of ‘putting something on the table’ and ‘making sandwich’ provide contextual information to detect the composite action ‘cooking’. This example shows the need for an effective temporal modeling technique for detecting actions in a densely-labelled videos.

Towards modeling temporal relations in untrimmed videos, multiple previous methods \cite{9, 10, 12, 13, 31, 39} use 1D temporal convolutions \cite{31}. However, limited by their kernel size, convolution-based methods can directly access local information only, not learning direct relations between temporally-distant segments in a video (here, we consider a set of consecutive frames as a segment). Thus, such methods fail to model long-range interactions between segments which may be important for action detection. With the success of Transformers \cite{17, 35, 45, 57} in natural language processing and more recently in computer vision, recent methods \cite{43, 44} have leveraged multi-head self-attention (MHSA) to model long-term relations in videos for action detection. Such attention mechanisms can build direct one-to-one global relationships between each temporal segment (i.e., temporal token) of a video to detect highly-correlated and composite actions. However, existing methods rely on modeling such long-term relationships on input frames themselves. Here, a temporal token covers only a

\textsuperscript{1}Code/Models: https://github.com/dairui01/MS-TCT

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure.png}
\caption{Complex temporal relations in untrimmed videos: Here, we show a typical distribution of actions in a densely-labelled video, which consists of both long-term and short-term dependencies among actions.}
\end{figure}
few frames, which is often too short w.r.t. to the duration of action instances. Also, in this setting, transformers need to explicitly learn strong relationships between adjacent tokens which arise due to temporal consistency, whereas it comes naturally for temporal convolutions (i.e., local inductive bias). Therefore, a pure transformer architecture may not be sufficient to model complex temporal dependencies for action detection.

To this end, we propose Multi-Scale Temporal ConvTransformer (MS-TCT), a model which benefits from both convolutions and self-attention. We use convolutions in a token-based architecture to promote multiple temporal scales of tokens, and to blend neighboring tokens imposing a temporal consistency with ease. In fact, MS-TCT is built on top of temporal segments encoded using a 3D convolutional backbone [5]. Each temporal segment is considered as a single input token to MS-TCT, to be processed in multiple stages with different temporal scales. These scales are determined by the size of the temporal segment, which is considered as a single token at the input of each stage. Having different scales allows MS-TCT to learn both fine-grained relations between atomic actions (e.g., ‘open fridge’) in the early stages, and coarse relations between composite actions (e.g., ‘cooking’) in the latter stages. To be more specific, each stage consists of a temporal convolution layer for merging tokens, followed by a set of multi-head self-attention layers and temporal convolution layers, which model global temporal relations and infuse local information among tokens, respectively. As convolution introduces an inductive bias [16], the use of temporal convolution layers in MS-TCT can infuse positional information related to tokens [22, 24], even without having any positional embeddings, unlike pure transformers [17]. Followed by the modeling of temporal relations at different scales, a mixer module is used to fuse the features from each stages to get a unified feature representation. Finally, to predict densely-distributed actions, we introduce a heat-map branch in MS-TCT in addition to the usual multi-label classification branch. This heat-map encourages the network to predict the relative temporal position of instances of each action class. Fig. 2 shows the relative temporal position, which is computed based on a Gaussian filter parameterized by the instance center and its duration. It represents the relative temporal position w.r.t. to the action instance center at any given time. With this new branch, MS-TCT can embed a class-wise relative temporal position in token representations, encouraging discriminative token classification in complex videos.

To summarize, the main contributions of this work are to (1) propose an effective and efficient ConvTransformer for modeling complex temporal relations in untrimmed videos, (2) introduce a new branch to learn the position relative to instance-center, which promotes action detection in densely-labelled videos, and (3) improve the state-of-the-art on three challenging densely-labelled action datasets.

2. Related Work

Action detection has received a lot of interest in recent years [8, 13, 15, 21, 32, 51, 54]. In this work, we focus on action detection in densely-labelled videos [11, 42, 52]. The early attempts on modeling complex temporal relations tend to use anchor-based methods [6, 50]. However, dense action distributions require large amount of such anchors. Super-event [38] utilizes a set of Gaussian filters to learn video glimpses, which are later summed up with a soft attention mechanism to form a global representation. However, as these Gaussians are independent of the input videos, it can not handle videos with minor frequencies of composite actions effectively. Similarly, TGM [39] is also a temporal filter based on Gaussian distributions, which enables the learning of longer temporal structures with a limited number of parameters. PDAN [10] is a temporal convolutional network, with temporal kernels which are adaptive to the input data. Although TGM and PDAN achieve state-of-the-art performance in modeling complex temporal relations, these relations are constrained to local regions, thus preventing them to learn long-range relationships. Coarse-Fine Networks [27] leverage two X3D [18] networks in a Slow-Fast [19] fashion. This network can jointly model spatio-temporal relations. However, it is limited by the number of input frames in X3D backbone, and a large stride is required to process long videos efficiently. This prevents Coarse-Fine Networks from considering the fine-grained details in long videos for detecting action boundaries. A concurrent work [26] looks into detection pretraining with only classification labels, to improve downstream action detection. Recently, some attempts have been proposed to model long-term relationships explicitly: MTCN [29] benefits from the temporal context of action and labels, whereas TQN [53] factorizes categories into pre-defined attribute queries to predict fine-grained actions. However, it is not trivial to extend both approaches to action detection in untrimmed videos.

Recent Transformer models have been successful in both
image and video domain [2, 3, 7, 17, 35, 36, 40, 46, 47, 49, 55, 57]. Although Vision Transformers such as TimeSformer [45] can consider frame-level input tokens to model temporal relations, it is limited to short video clips which is insufficient to model fine-grained details in longer real-world videos. As a compromise, recent action detection methods use multi-head self-attention layers on top of the visual segments encoded by 3D convolutional backbones [5]. RTD-Net [43], an extension of DETR [57], uses a transformer decoder to model the relations between the proposal and the tokens. However, this network is designed only for sparsely-annotated videos [4, 25], where only a single action exists per video. In dense action distributions, the module that detects the boundaries in RTD-Net fails to separate foreground and background regions. MLAD [44] learns class-specific features and uses a transformer encoder to model class relations at each time-step and temporal relations for each class. However, MLAD struggles with datasets that has complex labels [42], since it is hard to extract class-specific features in such videos. In contrast to these transformers introduced for action detection, we propose a ConvTransformer: MS-TCT, which inherits a transformer encoder architecture, while also gaining benefits from temporal convolution. Our method can model temporal tokens both globally and locally at different temporal scales. Although other ConvTransformers [16, 22, 28, 48] exist for image classification, our network is designed and rooted for densely-labelled action detection.

3. Multi-Scale Temporal ConvTransformer

First, we define the problem statement of action detection in densely-labelled settings. Formally, for a video sequence of length $T$, each time-step $t$ contains a ground-truth action label $y_{t,c} \in \{0, 1\}$, where $c \in \{1, ..., C\}$ indicates an action class. For each time-step, an action detection model needs to predict class probabilities $\tilde{y}_{t,c} \in [0, 1]$. Here, we describe our proposed action detection network: MS-TCT. As depicted in Fig. 3, it consists of four main components: (1) a Visual Encoder which encodes a preliminary video representation, (2) a Temporal Encoder which structurally models the temporal relations at different temporal scales (i.e., resolution), (3) a Temporal Scale Mixer, dubbed as TS Mixer, which combines multi-scale temporal representations, and (4) a Classification Module which predicts class probabilities. In the following sections, we present the details of each these components of MS-TCT.

3.1. Visual Encoder

The input to our action detection network: MS-TCT, is an untrimmed video which may span for a long duration [11] (e.g. multiple minutes). However, processing long videos in both spatial and temporal dimensions can be challenging, mainly due to computational burden. As a compromise, similar to previous action detection models [10, 39, 44], we consider features of video segments extracted by a 3D CNN as inputs to MS-TCT, which embed spatial information latently as channels. Specifically, we use an I3D backbone [5] to encode videos. Each video is divided into $T$ non-overlapping segments (during training), each of which consists of 8 frames. Such RGB frames are fed as an input segment to the I3D network. Each video token is seen as a transformer token of a time-step (i.e., temporal token). We stack the tokens along the temporal axis to form a $T \times D_0$ video token representation, to be fed in to the Temporal Encoder.

3.2. Temporal Encoder

As previously highlighted in Section 1, efficient temporal modeling is critical for understanding long-term temporal relations in a video, especially for complex action compositions. Given a set of video tokens, there are two
main ways to model temporal information: using (1) a 1D Temporal Convolutional layer [31], which focuses on the neighboring tokens but overlooks the direct long-term temporal dependencies in a video, or (2) a Transformer [45] layer that globally encodes one-to-one interactions of all tokens, while neglecting the local semantics, which has proven beneficial in modeling the highly-correlated visual signals [20, 23]. Our Temporal Encoder benefits from the best of both worlds, by exploring both local and global contextual information in an alternating fashion.

As shown in Fig. 3, Temporal Encoder follows a hierarchical structure with $N$ stages: Earlier stages learn a fine-grained action representation with more temporal tokens, whereas the latter stages learn a coarse representation with fewer tokens. Each stage corresponds to a semantic level (i.e., temporal resolution) and consists of one Temporal Merging block and $\times B$ Global-Local Relational Blocks (see Fig. 4):

**Temporal Merging Block** is the key component for introducing network hierarchy, which shrinks the number of tokens (i.e., temporal resolution) while increasing the feature dimension. This step can be seen as a weighted pooling operation among the neighboring tokens. In practice, we use a single temporal convolutional layer (with a kernel size of $k$, and a stride of 2, in general) to halve the number of tokens and extend the channel size by $\times \gamma$. In the first stage, we keep a stride of 1 to maintain the same number of tokens as the 1D output, and project the feature size from $D_0$ to $D$ (see Fig. 3). This is simply a design choice.

**Global-Local Relational Block** is further decomposed in to a Global Relational Block and a Local Relational Block (see Fig. 4). In Global Relational Block, we use the standard multi-head self-attention layer [45] to model long-term action dependencies, i.e., global contextual relations. In Local Relational Block, we use a temporal convolutional layer (with a kernel size of $k$) to enhance the token representation by infusing the contextual information from the neighboring tokens, i.e., local inductive bias. This enhances the temporal consistency of each token while modeling the short-term temporal information corresponding to an action instance.

In the following, we formulate the computation flow inside the Global-Local Relational Block. For brevity, here, we drop the stage index $n$. For a block $j \in \{1, \ldots, B\}$, we represent the input tokens as $X_j \in \mathbb{R}^{D \times D'}$. First, the tokens go through multi-head attention layer in Global Relational Block, which consists of $H$ attention heads. For each head $i \in \{1, \ldots, H\}$, an input $X_j$ is projected in to $Q_{ij} = W^Q_{ij}X_j$, $K_{ij} = W^K_{ij}X_j$ and $V_{ij} = W^V_{ij}X_j$, where $W^Q_{ij}, W^K_{ij}, W^V_{ij} \in \mathbb{R}^{D_h \times D'}$ represent the weights of linear layers and $D_h = \frac{D}{H}$ represents the feature dimension of each head. Consequently, the self-attention for head $i$ is computed as,

$$
\text{Att}_{ij} = \text{Softmax}\left(\frac{Q_{ij}K_{ij}^T}{\sqrt{D_h}}\right)V_{ij}.
$$

Then, the output of different attention heads are mixed with an additional linear layer as,

$$
M_j = W^O_j \text{Concat}(\text{Att}_{1j}, \ldots, \text{Att}_{Hj}) + X_j,
$$

where $W^O_j \in \mathbb{R}^{D' \times D'}$ represents the weight of the linear layer. The output feature size of multi-head attention layer is the same as the input feature size.

Next, the output tokens of multi-head attention are fed in to the Local Relational Block, which consists of two linear layers and a temporal convolutional layer. As shown in Fig. 4, the tokens first go through a linear layer to increase the feature dimension from $D'$ to $\theta D'$, followed by a temporal convolutional layer with a kernel size of $k$, which blends the neighboring tokens to provide local positional information to the temporal tokens [24]. Finally, another linear layer projects the feature dimension back to $D'$. The two linear layers in this block enable the transition between the multi-head attention layer and temporal convolutional layer. The output feature dimension remains the same as the input feature for the Local Relational Block. This output is fed to the next Global Relational Block if block $j < B$.

The output tokens from the last Global-Local Relational Block from each stage are combined and fed to the following Temporal Scale Mixer.

### 3.3. Temporal Scale Mixer

After obtaining the tokens at different temporal scales, the question that remains is, how to aggregate such multi-scale tokens to have a unified video representation? To predict the action probabilities, our classification module needs to make predictions at the original temporal length as the network input. Thus, we require to interpolate the tokens across the temporal dimension, which is achieved by performing an up-sampling and a linear projection step.
Figure 5. **Temporal Scale Mixer Module:** The output tokens $F_n$ of stage $n$ is resized and up-sampled to $T \times D_v$, then summed with the tokens from the last stage $N$.

As shown in Fig. 5, for the output $F_n$ from stage $n \in \{1, ..., N\}$, this operation can be formulated as,

$$g_n(F_n) = \text{UpSampling}_n(F_n W^n),$$

(3)

where $W^n \in \mathbb{R}^{D_v \times \gamma^{n-1} D}$ with an upsampling rate of $n$. In our hierarchical architecture, earlier stages (with lower semantics) have higher temporal resolution, whereas the latter stages (with high semantics) have lower temporal resolution. To balance the resolution and semantics, upsampled tokens from the last stage $N$ is processed through a linear layer and summed with the upsampled tokens from each stage ($n < N$). This operation can be formulated as,

$$F'_n = g_n(F_n) \oplus g_N(F_N) W_n,$$

(4)

where $F'_n$ is the refined tokens of stage $n$, $\oplus$ indicates the element-wise addition and $W_n \in \mathbb{R}^{D_v \times D_v}$. Here, all the refined token representations have the same temporal length. Finally, we concatenate them to get the final multi-scale video representation $F_v \in \mathbb{R}^{T \times N D_v}$.

$$F_v = \text{Concat}(F'_1, ..., F'_{N-1}, F_N).$$

(5)

Note that more complicated fusion methods [14, 34] can be built on top of these multi-scale tokens. However, we see that the simple version described above performs the best.

The multi-scale video representation $F_v$ is then sent to the classification module for making predictions.

### 3.4. Classification Module

Training MS-TCT is achieved by jointly learning two classification tasks. As mentioned in Section 1, in this work, we introduce a new classification branch to learn a heat-map of the action instances. This heat-map is different from the ground truth label as it varies across time, based on the action center and duration. The objective of using such heat-map representation is to encode temporal relative positioning in the learned tokens of MS-TCT.

In order to train the heat-map branch, we first need to build the class-wise ground-truth heat-map response $G^* \in [0, 1]^{T \times C}$, where $C$ indicates the number of action classes.

In this work, we construct $G^*$ by considering the maximum response of a set of one-dimensional Gaussian filters. Each Gaussian filter corresponds to an instance of action class in a video, centered at the specific action instance, in time. More precisely, for every temporal location $t$ the ground-truth heat-map response is formulated as,

$$G^*_c(t) = \max_{a=1, ..., A_c} \text{Gaussian}(t, t_{a,c}; \sigma),$$

(6)

$$\text{Gaussian}(t, t_{a,c}; \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(t-t_{a,c})^2}{2\sigma^2}\right).$$

(7)

Here, $\text{Gaussian}(\cdot, \cdot; \sigma)$ provides an instance-specific Gaussian activation according to the center and instance duration. Moreover, $\sigma$ is equal to $\frac{1}{2}$ of each instance duration and $t_{a,c}$ represents the center for class $c$ and instance $a$. $A_c$ is the total number of instances for class $c$ in the video. As shown in Fig. 3, heat-map $G$ is computed using a temporal convolutional layer with a kernel size of $k$ and a non-linear activation, followed by another linear layer with a sigmoid activation. Given the ground-truth $G^*$ and the predicted heat-map $G$, we compute the action focal loss [33, 56] which is formulated as,

$$L_{\text{Focal}} = \frac{1}{A} \sum_{t,c} \begin{cases} \left(1 - G^*_{t,c}\right)^2 \log(G^*_{t,c}) & \text{if } G^*_{t,c} = 1, \\ \left(1 - G^*_{t,c}\right)^2 \log(1 - G^*_{t,c}) & \text{otherwise}, \end{cases}$$

(8)

where $A$ is the total number of action instances in a video.

Similar to the previous work [10, 44], we leverage another branch to perform the usual multi-label classification. With video features $F_v$, the predictions are computed using two linear layers with a sigmoid activation, and Binary Cross Entropy (BCE) loss [37] is computed against the ground-truth labels. Only the scores predicted from this branch are used in evaluation. Input to both the branches are the same output tokens $F_v$. The heat-map branch encourages the model to embed the relative position w.r.t. the instance center in to video tokens $F_v$. Consequently, the classification branch can also benefit from such positional information to make better predictions.

The overall loss is formulated as a weighted sum of the two losses mentioned above, with the weight $\alpha$ is chosen according to the numerical scale of losses.

$$L_{\text{Total}} = L_{\text{BCE}} + \alpha L_{\text{Focal}}.$$  

(9)

### 4. Experiments

**Datasets:** We evaluate our framework on three challenging multi-label action detection datasets: Charades [42], TSU [11] and MultiTHUMOS [52]. Charades [42] is a large dataset with 9848 videos of daily indoor actions. The dataset contains 66K+ temporal annotations for 157 action classes, with a high overlap among action instances of different classes. This is in contrast to other action detection...
datasets such as ActivityNet [4], which only have one action per time-step. We evaluate on the localization setting of the dataset [41]. Similar to the Charades, TSU [11] is also recorded in indoor environment with dense annotations. Up to 5 actions can happen at the same time in a given frame. However, different from Charades, TSU has many long-term composite actions. MultiTHUMOS [52] is an extended version of THUMOS’14 [25], containing dense, multi-label action annotations for 65 classes across 413 sports videos. By default, we evaluate the per-frame mAP on these densely-labelled datasets following [41, 52].

Implementation Details: In the proposed network, we use number of stage $N = 4$ the number of Global-Local Relational Blocks $B = 3$ for each stage. Note that for small dataset as MultiTHUMOS, $B = 2$ is sufficient. The number of attention heads for the Global Relational Block is set to 8. We use the same output feature dimension of I3D (after Global Average Pooling) as input to MS-TCT, and thus $D_0 = 1024$. Input features are then projected in to $D = 256$ dimensional feature using the temporal merging block in the first stage. We consider feature expansion rate $\gamma = 1.5$ and $\theta = 8$. Kernel size $k$ of temporal convolutional layer is set to be 3, with zero padding to maintain the resolution. The loss balance factor $\alpha = 0.05$. The number of tokens is fixed to $T = 256$ as input to MS-TCT. During training, we randomly sample consecutive $T$ tokens from a given I3D feature representation. At inference, we follow [44] to use a sliding window approach to make predictions. Our model is trained on two GTX 1080 Ti GPUs with a batch-size of 32. We use Adam optimizer [30] with an initial learning rate of 0.0001, which is scaled by a factor of 0.5 with a patience of 8 epochs.

4.1. Ablation Study

In this section, we study the effectiveness of each component in the proposed network on Charades dataset.

<table>
<thead>
<tr>
<th>Component</th>
<th>Temporal Encoder</th>
<th>Global Encoder</th>
<th>Local Encoder</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS Mixer</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>15.6</td>
</tr>
<tr>
<td>Heat-Map Branch</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>23.6</td>
</tr>
<tr>
<td>Classification Branch</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>24.1</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>10.7</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>25.4</td>
</tr>
</tbody>
</table>

Importance of Each Component in MS-TCT: As shown in Table 1, I3D features with the classification branch only, is considered as the representative baseline. This baseline consists in a classifier that discriminates the I3D features at each time-step without any further temporal modeling. On top of that, adding our Temporal Encoder significantly improves the performance (+ 7.0%) w.r.t. I3D feature baseline. This improvement reflects the effectiveness of the Temporal Encoder in modeling the temporal relations within the videos. In addition, if we introduce a Temporal Scale Mixer to blend the features from different temporal scales, it gives a + 0.5% improvement, with minimal increase in computations. Finally, we study the utility of our heat-map branch in the classification module. We find that the heat-map branch is effective when optimized along with the classification branch, but fails to learn discriminative representations when optimized without it (25.4% vs 10.7%). The heat-map branch encourages the tokens to predict the action center while down-playing the tokens towards action boundaries. In comparison, the classification branch improves the token representations equally for all tokens, despite action boundaries. Thus, when optimized together, both branches enable the model to learn a better action representation. While having all the components, the proposed network achieves a significant + 9.8% improvement w.r.t. I3D feature baseline validating that each component in MS-TCT is instrumental for the task of action detection.

Design Choice for a Stage: In Table 2, we present the ablation related to the design choices of a stage in the Temporal Encoder. Each row in Table 2 indicates the result of removing a component in each stage. Note that, removing the Temporal Merge block indicates replacing this block with a temporal convolutional layer of stride 1, i.e., only the channel dimension is modified across stages. In Table 2, we find that removing any component can drop the performance with a significant margin. This observation shows the importance of jointly modeling both global and local relations in our method, and the effectiveness of the multi-scale structure. These properties in MS-TCT make it easier to learn complex temporal relationships which span across both (1) neighboring temporal segments, and (2) distant temporal segments.

Analysis of the Local Relational Block: We also dig deeper into the Local Relational Block in each stage. As shown in Fig. 4, there are two linear layers and one temporal convolutional layer in a Local Relational Block. In Table 3, we further perform ablations of these components. First, we find that without the temporal convolutional layer, the detection performance drops. This observation shows the importance of mixing the transformer tokens with a temporal locality. Second, we study the importance of the transition
Table 3. Ablation on the design of Local Relational Block: Per-frame mAP on Charades using only RGB input. ✓ indicates we remove the linear or temporal convolutional layer. Feature expansion rate 1 indicates that the feature-size is not changed in the Local Relational Block.

<table>
<thead>
<tr>
<th>Feature Expansion Rate (θ)</th>
<th>Temporal Convolution</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 ✓</td>
<td></td>
<td>22.3</td>
</tr>
<tr>
<td>1 ✓</td>
<td>✓</td>
<td>22.4</td>
</tr>
<tr>
<td>4 ✓</td>
<td>✓</td>
<td>24.2</td>
</tr>
<tr>
<td>8 ✓</td>
<td>✓</td>
<td>25.4</td>
</tr>
</tbody>
</table>

Table 4. Comparison with the state-of-the-art methods on three densely labelled datasets. Backbone indicates the visual encoder. Note that the evaluation for the methods is based on per-frame mAP (%) using only RGB videos.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>GFLOPs</th>
<th>Charades mAP (%)</th>
<th>MultiTHUMOS mAP (%)</th>
<th>TSU mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-C3D [50]</td>
<td>C3D</td>
<td>8.7</td>
<td>12.7</td>
<td>36.4</td>
<td>17.2</td>
</tr>
<tr>
<td>Super-event [38]</td>
<td>I3D</td>
<td>0.8</td>
<td>18.6</td>
<td>36.4</td>
<td>17.2</td>
</tr>
<tr>
<td>TGM [39]</td>
<td>I3D</td>
<td>1.2</td>
<td>20.6</td>
<td>37.2</td>
<td>26.7</td>
</tr>
<tr>
<td>PDAN [10]</td>
<td>I3D</td>
<td>3.2</td>
<td>23.7</td>
<td>40.2</td>
<td>32.7</td>
</tr>
<tr>
<td>Coarse-Fine [27]</td>
<td>X3D</td>
<td>-</td>
<td>25.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MLAD [44]</td>
<td>I3D</td>
<td>44.8</td>
<td>18.4</td>
<td>42.2</td>
<td>-</td>
</tr>
<tr>
<td>MS-TCT</td>
<td>I3D</td>
<td>6.6</td>
<td>25.4</td>
<td>43.1</td>
<td>33.7</td>
</tr>
</tbody>
</table>

layer (i.e., linear layer). When the feature size remains constant, having the transition layer can boost the performance by +1.8%, which shows the importance of such transition layers. Finally, we study how the expansion rate affects the network performance. While setting different feature expansion rates, we find that temporal convolution can better model the local temporal relations when the input feature is in a higher dimensional space.

4.2. Comparison to the State-of-the-Art

In this section, we compare MS-TCT with the state-of-the-art action detection methods (see Table 4). Proposal-based methods, such as R-C3D [50] fail in multi-label datasets due to the highly-overlapping action instances, which challenge the proposal and NMS-based methods. Super-event [38] superimposes a global representation to each local feature based on a series of learnable temporal filters. However, the distribution of actions varies from one video to the other. As super-event learns a fixed filter location for all the videos in the training distribution, this location is suitable to mainly actions with high frequency. TGM [39] and PDAN [10] are methods based on temporal convolution of video segments. Nevertheless, those methods only process videos locally at a single temporal scale. Thus, they are not effective in modeling long-term dependencies and high-level semantics. Coarse-Fine Network [27] achieves 25.1% on Charades. However, this method is built on top of the video encoder X3D [18], which prevents the usage of higher number of input frames. Moreover, it relies on a large stride between the frames. Therefore, it fails to model fine-grained action relations, and can not process long videos in MultiTHUMOS and TSU. MLAD [44] jointly models action class relations for every time-step and temporal relations for every class. This design leads to a huge computational cost, while under-performing on datasets with a large number of action classes (e.g., Charades). Thanks to the combination of transformer and convolution in a multi-scale hierarchy, the proposed MS-TCT consistently outperforms previous state-of-the-art methods in all three challenging multi-label action detection datasets that we considered. We also compare the computational requirement (FLOPs) for the methods built on top of the same Visual Encoder (i.e., I3D features), taking as input the same batch of data. We observe that the FLOPs of MS-TCT is higher with a reasonable margin than pure convolutional methods (i.e., PDAN, TGM, super-event). However, compared to a transformer based action detection method MLAD, MS-TCT uses only ¼th of the FLOPs.

We also evaluate our network with the action-conditional metrics introduced in [44] on Charades dataset in Table 5. These metrics are used to measure a method’s ability to model both co-occurrence dependencies and temporal dependencies of action classes. Although our network is not specifically designed to model cross-class relations as in MLAD, it still achieves higher performance on all action-conditional metrics with a large margin, showing that MS-TCT effectively models action dependencies both within a time-step (i.e., co-occurring action, \( \tau = 0 \)) and throughout the temporal dimension (\( \tau > 0 \)).

Finally, we present a qualitative evaluation for PDAN and MS-TCT on the Charades dataset in Fig. 6. As the prediction of the Coarse-Fine Network is similar to the X3D network which is limited to dozens of frames, thus we can not compare with the Coarse-Fine network on the whole video. Here, we observe that MS-TCT can predict action instances more precisely compared to PDAN. This comparison reflects the effectiveness of the transformer architecture and multi-scale temporal modeling.

4.3. Discussion and Analysis

Transformer, Convolution or ConvTransformer? To confirm the effectiveness of our ConvTransformer, we com-
Table 5. Evaluation on the Charades dataset using the action-conditional metrics [44]: Similar to MLAD, both RGB and Optical flow are used for the evaluation. \(P_{AC}\) - Action-Conditional Precision, \(R_{AC}\) - Action-Conditional Recall, \(F_{1,AC}\) - Action-Conditional F1-Score, \(mAP_{AC}\) - Action-Conditional Mean Average Precision. \(\tau\) indicates the temporal window size.

<table>
<thead>
<tr>
<th>Stage-Type</th>
<th>(\tau = 0)</th>
<th>(\tau = 20)</th>
<th>(\tau = 40)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(P_{AC})</td>
<td>(R_{AC})</td>
<td>(F_{1,AC})</td>
</tr>
<tr>
<td>I3D</td>
<td>14.3</td>
<td>1.3</td>
<td>2.1</td>
</tr>
<tr>
<td>CF</td>
<td>10.3</td>
<td>1.0</td>
<td>1.6</td>
</tr>
<tr>
<td>MLAD [44]</td>
<td>19.3</td>
<td>7.2</td>
<td>8.9</td>
</tr>
<tr>
<td>MS-TCT</td>
<td>26.3</td>
<td>15.5</td>
<td>19.5</td>
</tr>
</tbody>
</table>

Table 6. Study on stage type showing the effect of having both convolutions and self-attention.

<table>
<thead>
<tr>
<th>Stage-Type</th>
<th>mAP</th>
<th>Variance: (\sigma)</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pure Transformer</td>
<td>22.3</td>
<td>1/8 duration</td>
<td>24.6</td>
</tr>
<tr>
<td>Pure Convolution</td>
<td>21.4</td>
<td>1/4 duration</td>
<td>24.8</td>
</tr>
<tr>
<td>ConvTransformer</td>
<td><strong>25.4</strong></td>
<td>1/2 duration</td>
<td><strong>25.4</strong></td>
</tr>
</tbody>
</table>

Table 7. Study on \(\sigma\) showing the effect of scale of Gaussians in heat-maps.

Figure 7. Heat-map visualization along time axis: On the top, we show the ground truth heat-map \((G^*)\) of the example video. On the bottom is the corresponding learned heat-map \((G)\) of MS-TCT. As the heat-map is generated by a Gaussian function, the lighter region indicates closer to the center of the instance.

Heat-map Analysis: We visualize the ground truth heat-map \((G^*)\) and the corresponding predicted heat-map \((G)\) in Fig. 7. We observe that with the heat-map branch, MS-TCT predicts the center location of the action instances, showing that MS-TCT embeds the center-relative information in to the tokens. However, as we optimize with the focal loss to highlight the center, the boundaries of the action instance in this heat-map are less visible. We then study the impact of \(\sigma\) on performance. As shown in Table 7, we set \(\sigma\) to be either \(\frac{1}{8}\), \(\frac{1}{4}\) or \(\frac{1}{2}\) of the instance duration while generating the ground-truth heat-map \((G^*)\). MS-TCT improves by +0.5\%, +0.7\%, +1.3\% respectively w.r.t. the MS-TCT without the heat-map branch, when \(G^*\) set to different \(\sigma\). This result reflects that a larger \(\sigma\) can better provide the center-relative position. We investigate further by adding a heat-map branch to another action detection model: PDAN [10]. Although the heat-map branch also improves PDAN (+0.4\%), the relative improvement is lower compared to MS-TCT (+1.3\%). Our method features a multi-stage hierarchy along with a TS Mixer. As the heat-map branch takes input from all the stages, the center-relative position is embedded even in an early stage. Such tokens with the relative position information, when fed through the following stages, benefits the multi-head attention to better model temporal relations among the tokens. This design makes MS-TCT to better leverage the heat-map branch compared to PDAN.

Temporal Positional Embedding: We further study whether the Temporal Encoder of MS-TCT benefits from positional embedding. We find that the performance drops by 0.2\% on Charades when a learnable positional embedding [17] is added to the input tokens before processing them with the Temporal Encoder. This shows that the current design can implicitly provide a temporal positioning for the tokens. Adding further positional information to the tokens makes it redundant, leading to lower detection performance.
5. Conclusion

In this work, we proposed a novel ConvTransformer network: MS-TCT for action detection. It benefits from both convolutions and self-attention to model local and global temporal relations respectively, at multiple temporal scales. Also, we introduced a new branch to learn class-wise relative positions of the action instance center. MS-TCT is evaluated on three challenging densely-labelled action detection benchmarks, on which it achieves new state-of-the-art results.

Acknowledgements: This work was supported by the French government, through the 3IA Côte d’Azur Investments in the Future project managed by the National Research Agency with the reference number ANR-19-P3IA-0002. This work was also supported in part by the National Science Foundation (IIS-2104404 and CNS-2104416). The authors are grateful to the OPAL infrastructure from Université Côte d’Azur for providing resources and support.

References


