CTRN: CLASS TEMPORAL RELATIONAL NETWORK FOR ACTION DETECTION

Rui Dai, Srijan Das, Francois F Bremond

CTRN: CLASS TEMPORAL RELATIONAL NETWORK FOR ACTION DETECTION

Rui Dai,
Inria, Université Côte d'Azur
rui.dai@inria.fr

Srijan Das
Stony Brook University
srijan.das@stonybrook.edu

François Brémond
Inria, Université Côte d'Azur
francois.bremond@inria.fr

ABSTRACT

Action detection is an essential and challenging task, especially for densely labelled datasets of untrimmed videos. There are many real-world challenges in those datasets, such as composite action, co-occurring action, and high temporal variation of instance duration. For handling these challenges, we propose to explore both the class and temporal relations of detected actions. In this work, we introduce an end-to-end network: Class-Temporal Relational Network (CTRN). It contains three key components: (1) The Representation Transform Module filters the class-specific features from the mixed representations to build a graph structured data. (2) The Class-Temporal Module models the class and temporal relations in a sequential manner. (3) G-classifier leverages the privileged knowledge of the snippet-wise co-occurring action pairs to further improve the co-occurring action detection. We evaluate CTRN on three challenging densely labelled datasets and achieve state-of-the-art performance, reflecting the effectiveness and robustness of our method.

Action detection is a challenging computer vision problem which targets at finding precise temporal boundaries of actions occurring in an untrimmed video. Many studies on action detection focus on videos with sparse and well-separated instances of action Xu et al. (2020); Zeng et al. (2019); Chen et al. (2020). For instance, action detection algorithms on popular datasets like THUMOS Jiang et al. (2014) and ActivityNet Caba Heilbron et al. (2015) generally learn representations for single actions in a video. However, in daily life, human actions are continuous and can be very dense. Every minute is filled with potential actions to be detected and labelled. The methods designed for sparsely labelled datasets are hard to generalize to such real-world scenarios.

Towards this research direction, several methods Piergiovanni & Ryoo (2019); Dai et al. (2021b); Piergiovanni & Ryoo (2018) have been proposed to model complex temporal relationships and to process datasets like Charades Sigurdsson et al. (2016), TSU Dai et al. (2020) and MultiTHUMOS Yeung et al. (2018). Those datasets encompassing real-world challenges share the following characteristics: Firstly, the actions are densely labelled and background instances are rare in these videos compared to sparsely labelled datasets. Secondly, the video has rich temporal structure and a set of actions occurring together often follows a well defined temporal pattern. For example, *drinking from bottle* always happens after *taking a bottle* and *reading a book* also related to *opening a book* in Fig 1. Finally, humans are great at multitasking, multiple actions can co-occur at the same time. For example, *reading book* while *drinking water*.

Existing methods have mostly focused on modelling the variation of visual cues across time locally Lea et al. (2017) or globally Piergiovanni & Ryoo (2018) within a video. However, these methods take into account the temporal information without any further semantics. Real-world videos contain many complex actions with inherent relationships between action classes at the same time steps or across distant time steps (see Fig. 1). Modelling such class-temporal relationships can be extremely useful for locating actions in those videos.

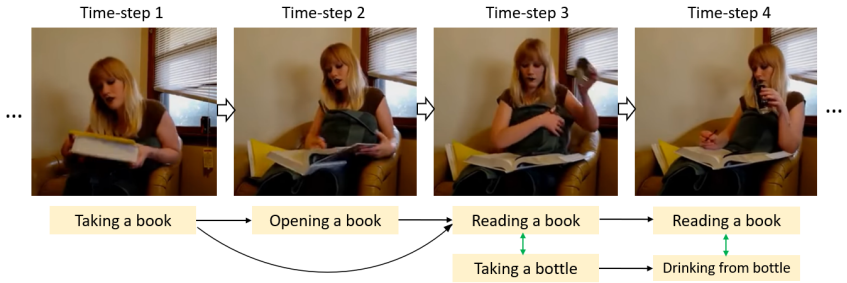

Figure 1: Class-Temporal Relation. In a dense labelled video, there are dependencies between action classes (1) across **different** time steps in black arrows and (2) at **same** time step (i.e. co-occurring actions) in green arrows.

To this end, we introduce Class-Temporal Relational Network (CTRN) to harness the relationships among the action classes in a video. To explore such relations, CTRN first filters the class-specific representation from the input features at each time step in a video. Then the transformed per-class representation is utilized for modelling the inter-class relations. (1) Across **different** time-steps, a graph-based layer is proposed to learn the dependencies between different action classes of the video. This learned relation map is shared among all the time-steps to refine the action features from the related actions (e.g. *open the book* and *read book*). Then a temporal layer is used to aggregate features from the same class over time to enable the graph-based layer to explore both short-term and long-term class dependencies. (2) At the **same** time step, a graph-based classifier is proposed to leverage the privileged co-occurring action probabilities to improve co-occurring action detection.

To summarize, the main contributions of this work are: 1) A graph-based module to explore action class relations across different time-steps. 2) A graph-based classifier to tackle the co-occurring action challenge. 3) We evaluate our model on three challenging densely labelled datasets for the action detection task. Our method outperforms state-of-the-art results using fewer parameters and FLOPs.

1 Related Work

Action detection has received a lot of interest in recent years Dai et al. (2021a); Huang et al. (2020); Dai et al. (2019a); Piergiovanni & Ryoo (2018). In this work, we focus on densely labelled action detection for handling videos with additional temporal relationships between different action classes Yeung et al. (2018); Sigurdsson et al. (2016); Dai et al. (2020). Different from the sparsely labelled detection methods which output a sparse set of action snippets Caba Heilbron et al. (2015); Jiang et al. (2014), densely labelled detection methods need to predict what action is occurring at every snippet in a video. There are two principle frameworks for densely labelled action detection: the anchor-based Xu et al. (2017); Chen et al. (2020) and the Seq2Seq-based Lea et al. (2017); Dai et al. (2019a) frameworks. The anchor-based frameworks are often slow and suffer from over-generated proposals and rigid boundaries. Seq2Seq-based Piergiovanni & Ryoo (2019) frameworks apply temporal filters over snippet-wise features with a snippet-level classification, therefore, it interprets the sequence of images to a sequence of predictions. Compared to anchor-based methods, the Seq2Seq methods achieve better performance for action detection on datasets with densely distributed actions. This is mainly because of the combinatorial explosion of action proposals generated by the former methods. Thus, the recent methods are following the Seq2Seq framework: Lea et al. Lea et al. (2017) introduced temporal convolutional network (TCN) for the action detection task. This method increases the temporal reception field by using dilated convolutions to model long temporal patterns. Similarly, Piergiovanni et al. Piergiovanni & Ryoo (2019) introduced Temporal Gaussian Mixture (TGM) layers. In contrast to standard convolution layer, TGM computes the filter weights based on Gaussian distributions, which enables TGM to learn longer temporal structures with a limited number of parameters. However, both methods focus only on the temporal modelling while overlooking the action class relations in the untrimmed videos. Although Super-event Piergiovanni & Ryoo (2018) models the latent contextual representation among actions, the modelled Super-event represents only the latent correlation among the time steps and not among the action

classes. Most related to our research direction, Tirupattur et al. Tirupattur et al. (2021) introduced MLAD that can explore the class-temporal relations with a set of self-attention layers: an inter-class attention map for every time-step and an inter-time attention map for every action class. However, the large number of attention maps leads to huge computational costs for long untrimmed videos and hence, limits the model to learn the discriminative relations among the action classes. To tackle this, we propose CTRN, which is a graph-based model. Different from MLAD, CTRN explores the action class relation shared by all the time steps but in different temporal scales. This design enables CTRN to effectively handle both short-term and long-term action relations simultaneously. Moreover, we propose a G-Classifier that focuses on the co-occurring actions taking into account the inter-dependencies of the actions in the training distribution. The proposed method is introduced in details in the following section.

2 PROPOSED METHOD

In this section, we formulate the proposed end-to-end model Class-Temporal Relational Network (CTRN) for action detection. As shown in Fig. 2, our model is composed of four major components. The **Visual Encoder** encodes the video into a sequence of snippet-level spatio-temporal representation. This representation is fed to a Class-temporal Relational Network (CTRN) that predicts the action labels at each time instant. The sub-components in CTRN consist of the following: Firstly, a **Representation Transform Module**, which transforms the mixed visual representation into a class-wise representation. Secondly, a **Class-Temporal Module** explores the action class relations across different time-steps and at different temporal resolutions. Finally, a **G-Classifier** which classifies the class-temporal features into action categories. Unlike previous binary classifiers Piergiovanni & Ryoo (2018; 2019) that overlook the dependencies between the action classes, G-Classifier leverages the privilege class dependencies within the training data, thus improving the co-occurring action detection performance. In the following, we introduce these modules in details.

2.1 VISUAL ENCODER

Similar to most action detection models Lea et al. (2017); Piergiovanni & Ryoo (2018; 2019), our model processes the features on top of video snippet representations extracted from 2D/3D CNNs. In this work, we use spatio-temporal features extracted from RGB and Optical Flow (OF) I3D networks Carreira & Zisserman (2017) to encode appearance and motion information respectively. Then, a video is divided into T non-overlapping snippets, each snippet consisting of 16 frames. The inputs to the RGB and Flow deep networks are either the color images or the corresponding OF frames of a snippet. We stack the snippet-level features along the temporal axis to form a $T \times D_1$ dimensional video representation, denoted as X. The action instances in X are always longer than a snippet and their visual representation mixes information of all action classes. As a result, X is not discriminative enough, and needs both temporal and class modelling. To this end, we develop the class-temporal relationship from the input representation X within CTRN which is described in the following. Note that the model architecture remains the same for both RGB and OF streams.

2.2 REPRESENTATION TRANSFORM MODULE

The input X is first fed into the Representation Transform Module (RTM). The goal of this module is to transform the input to a class-specific representation and to lightweight the channel size to facilitate the following computation. In practice, RTM duplicates the input features C times into a new dimension representing the action classes followed by a channel-mixer MLP with non-linear activation and dropout. MLP is the linear transformation layer Almeida (1997) to do the linear projection. The equation can be formulated as:

$$X'_i = ReLU(MLP(X)) \tag{1}$$

$$X' = DropOut([X'_1, X'_2, ...X'_C])) \tag{2}$$

where $X' \in \mathbb{R}^{T \times C \times D_2}$ is the output representation of RTM. $D_2 = \frac{D_1}{\beta}$ in which β is larger than 1 to shallow the channel size. In order to learn class-specific representation, we embed an auxiliary branch with a G-classifier that maps X' to the action labels (see Fig. 2). This transformed feature representation is further exploited to explore the class and temporal relations in the subsequent modules of the network.

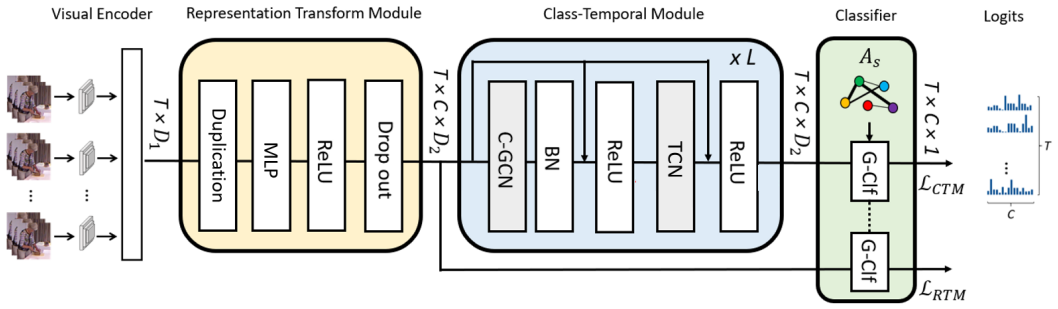

Figure 2: Overall structure. The model composed of a Visual Encoder, a Representation Transform Module, a Class-Temporal Module (with C-GCN and TCN) and a G-Classifier (i.e. G-Clf).

2.3 Class-Temporal Modeling

The Class-Temporal Module (CTM) is the key component of CTRN that exploits the class-temporal relations of its input feature. Inspired by the recent success of Graph Convolutional Network (GCN) in relational reasoning Kipf & Welling (2016); Zeng et al. (2019); Xu et al. (2020); Huang et al. (2020), we build this module with GCN. The objective of this component is to update the feature representations by propagating the information across different classes and across different time steps. For modelling the action class relations, we introduce a Class-GCN (C-GCN) layer while the traditional Temporal Convolutional Network (TCN) layer Lea et al. (2017) is utilized to aggregate the temporal information. The combination of C-GCN and TCN enables CTM to capture the class semantic information along different temporal hierarchies. Thanks to the learnable graph structure, C-GCN is adaptive with the temporal scale set by TCN.

In the following, we first introduce how we map the feature representation to the graph structure and then we introduce the CTM components.

2.3.1 Representation-to-Graph Mapping

For GCN to process the action relations, the data is to be converted into a graphical structure. As we have transformed the representation into the class-specific format, thus each vertex of the graph represents an action class at a time step with an embedding vector belonging to $\mathbb{R}^{D_2}$. In total, the graph consists of $C \times T$ vertices whose topology is defined by an adjacency matrix ($A_\mathcal{C}$). This matrix determines whether there are connections (i.e. relations) and its weights determine the intensity of the connections.

2.3.2 Class-GCN (C-GCN)

Class-GCN aims at performing the cross-class reasoning over the constructed graph representation. The relations between the many action instances are complex and are different across videos. Besides, multiple C-GCNs are stacked in CTM through which C-GCNs capture different levels of semantic information. Consequently, the graph adjacency $A_\mathcal{C}$ learns from the data itself for it to be adaptive across different temporal scales.

In practice, $A_\mathcal{C} \in \mathbb{R}^{C \times C}$ is parameterized and is optimized together with other parameters in the training process. Moreover, to differentiate the class relations owing to different videos, the adjacency matrix $A_\mathcal{C}$ learns the inter-dependencies among the classes using a self-attention mechanism. For this, the input feature $X_{C_{in}} \in \mathbb{R}^{D_2 \times T \times C}$ is first embedded using bottleneck convolutional layer (i.e. 1×1). After that, the output feature maps are rearranged into $\mathbb{R}^{D_2 T \times C}$ and $\mathbb{R}^{C \times D_2 T}$ followed by a matrix multiplication. The value of the resultant matrix is then normalized by a `softmax` activation. Now, the superimposed adjacency matrix $A'_\mathcal{C}$ can be formulated as:

$$A'_\mathcal{C} = A_\mathcal{C} + softmax(W_1^\top X_{C_{in}}^\top W_2 X_{C_{in}}) \tag{3}$$

where $X_{C_{in}}$ is the input of the C-GCN, and W_1 and W_2 are the weights of the bottleneck convolutions. Each value in this matrix can be seen as a soft edge between two vertices. The learned graph is shared across different time-steps but unique for different layers and videos. This design choice

can capture the inter-class dependencies in a video and makes C-GCN scalable across different temporal scales. Finally, we perform the graph convolutional operation with the formulation in Kipf & Welling (2016):

$$X_{C_{out}} = A'_C X_{C_{in}} W_3 \tag{4}$$

where $W_3 \in \mathbb{R}^{D_2 \times D_2}$ is the learnable weight matrix. The operation with A'_C and with W_3 represents the message passing and vertex feature updating, respectively. Finally, $X_{C_{out}}$ is rearranged to $\mathbb{R}^{D_2 \times T \times C}$.

2.3.3 CTM BLOCK

As shown in Fig. 2, there are L blocks in CTM, each block is composed of a C-GCN and a TCN layer along with batch normalization and non-linear activations. To stabilize the training, two residual connections are added in each block.

As mentioned earlier, TCN Lea et al. (2017) aggregates the features across the temporal dimension while increasing the size of the temporal receptive field. In this work, we set a fixed kernel size K for all the TCNs. Thanks to the hierarchical structure of CTM, C-GCN can focus on short-term action-dependencies in lower blocks and long-term action dependencies in higher blocks. The refined feature representation from the last block is fed into G-Classifier for the snippet-level classification.

2.4 G-CLASSIFIER (G-CLF)

Finally, we introduce a graph-based G-Classifier to perform the final snippet-level classification. In action detection, multiple actions could happen simultaneously; thus, prior knowledge of inter-dependencies among different action classes can benefit in making precise predictions. Inspired from Chen et al. (2019) in image recognition, we introduce a GCN-based classifier in our task, which has different working mechanisms. In the previous work Chen et al. (2019), the input of GCN is the word embedding of the label and the output is the classifier weights between the feature representation and the prediction. In contrast to that, our input and output to the GCN are the video representation and the prediction scores respectively. As our input is directly the video representation, thus the action occurrence information within each snippet are captured, which benefits the information propagation between relevant classes through graph connections. Compared to the standard binary classifier Piergiovanni & Ryoo (2018), G-Classifier has an additional message passing step between the potential co-occurring action pairs, thus improving the co-occurring action detection performance. Different to C-GCN, G-Classifier models only the snippet-level features and focuses only on the actions that occur simultaneously.

In practice, firstly, we compute the co-occurrence probabilities of all the action pairs in the training snippets. M_{ij} indicates the concurring times for action class C_i and C_j. Then, the conditional probability matrix $P_{ij} = P(C_j|C_i)$ is given by:

$$P_{ij} = M_{ij}/N_i \tag{5}$$

where N_i indicates the occurrence times of C_i in training set, and $P_{ij} \in \mathbb{R}^{C \times C}$ indicates the probability of class C_j given that C_i occurs at the same time. In fine-grained action datasets, some rare co-occurrences may add noise for detecting other common actions, and the number of co-occurrences from training and test set may not be completely consistent. In this work, we perform a thresholding operation to binarize the conditional probability matrix to filter the noisy edges and make the classifier more robust to inconsistent action classes. If $P_{ij} \geq \theta$, $A_{S_{ij}}$ is assigned 1, otherwise 0, where θ is the threshold. The computed co-occurrence matrix A_S is a binary correlation matrix which in turn defines the adjacency matrix of the graph for G-Classifier. The feature of a node is computed by the weighted sum of its own features and the adjacent nodes' features. However, the binary correlation matrix may change the feature scale Kipf & Welling (2016) and make the node feature over-smoothed Li et al. (2018). To alleviate this problem, we normalized the A_S following the re-weighted scheme in Chen et al. (2019). Different to the learnable adjacency matrices in C-GCN, A_S is fixed during training. The formulation of this G-Classifier is given below:

$$S = \sigma(A_S X^L W_S) \tag{6}$$

where S is the prediction score, σ is the `sigmoid` activation. X^L is the output feature from the last block of the Class-Temporal Module, and $W_S \in \mathbb{R}^{1 \times D_2}$ are the learnable weights of the G-Classifier. To learn the parameters, we optimize the multi-label binary cross-entropy loss with the

Table 1: Comparison with the State-of-the-art on three densely labelled datasets. The results are given in per-frame mAP (%). RGB +OF indicates the late fusion performance.

Model	Modality	Charades	TSU	MultiTHUMOS
R-C3D Xu et al. (2017)	RGB	12.7	8.7	-
I3D + TAN Dai et al. (2019b)	RGB+OF	17.6	-	33.3
I3D + Superevent Piergiovanni & Ryoo (2018)	RGB	18.6	17.2	36.4
I3D + TGM Piergiovanni & Ryoo (2019)	RGB	20.6	26.7	37.2
I3D + TGM Piergiovanni & Ryoo (2019)	RGB+OF	21.5	-	44.3
I3D + TGM + Superevent Piergiovanni & Ryoo (2019)	RGB+OF	22.3	-	46.4
I3D + MLAD Tirupattur et al. (2021)	RGB	18.4	-	42.2
I3D + MLAD Tirupattur et al. (2021)	RGB+OF	22.9	-	49.6
I3D + CTRN	RGB	25.3	**33.5**	44.0
I3D + CTRN	RGB+OF	**27.8**	-	**51.2**

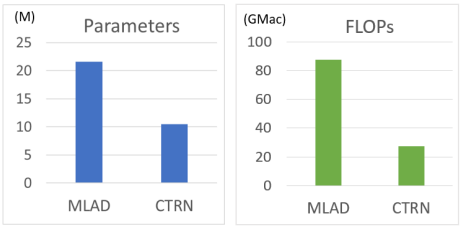

Figure 3: Computation efficiency.

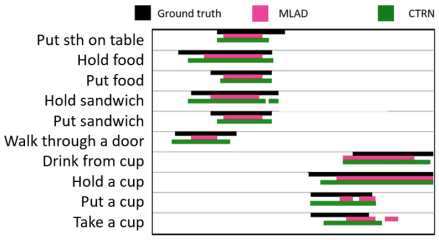

Figure 4: Model prediction visualization.

prediction results from the RTM and CTM. The total objective is formulate as:

$$\mathcal{L}_{total} = \mathcal{L}_{CTM} + \alpha \mathcal{L}_{RTM} \tag{7}$$

where α is a weighting factor. Thus, by jointly optimizing both the entropy losses, the model learns the relevant action labels per segment along with learning the class-specific semantics across the Representation transform module.

3 Experiment

3.1 Experimental Settings

Datasets. We evaluate our method on three densely labelled action detection datasets: Charades Sigurdsson et al. (2017), TSU Dai et al. (2020) and MultiTHUMOS Yeung et al. (2018). These datasets contain videos of different types: (1) sports and daily living videos, (2) short and long videos. We follow the original settings of these datasets for action detection. All these datasets are evaluated by the per-frame mAP.

Implementation. To have a fair comparison with previous works Piergiovanni & Ryoo (2019); Tirupattur et al. (2021); Piergiovanni & Ryoo (2018), our network is built on top of I3D, where D_1 is 1024 and D_2 is 64. Dropout probability is 0.3. For CTM, we choose a 5-block (L) structure. For C-GCN, the adjacency matrix is initialized by 1 and normalized by columns. In TCN, the kernel size K is 9 and padding rate is 4. For G-Classifier, θ is set to 0.05. While learning the parameters, the weighting factor α is 1.2 and the random seed is fixed. We use Adam optimizer Kingma & Ba (2014) with an initial learning rate of 0.001, and we scale it by a factor of 0.3 with a patience of 10 epochs. The network is trained on a 4-GPU machine for 300 epochs. For two-stream network, a mean pooling is performed between the prediction logits of the RGB and Flow streams.

3.2 Comparison with State-of-the-Art Methods

The proposed CTRN is compared with previous state-of-the-art methods on the Charades, TSU and MultiTHUMOS datasets in Table 1. Our proposed method outperforms current state-of-the-art methods on all three datasets. For example, +6.9% (relatively +37.5%) w.r.t. MLAD Tirupattur et al. (2021) on Charades while using only RGB. We then show the ability of CTRN capturing action co-occurrence, we evaluate with the action-conditional metric Tirupattur et al. (2021) in Table 2.

Table 2: Evaluation on the Charades dataset using the action-conditional metric Tirupattur et al. (2021). P_{AC} - Action-Conditional Precision, R_{AC} - Action-Conditional Recall, $F1_{AC}$ - Action-Conditional F1-Score, mAP_{AC} - Action-Conditional Mean Average Precision. τ indicates the temporal window size. $\tau = 0$ corresponds to the actions occuring at the same time.

	τ = 0				τ = 20				τ = 40			
	P_{AC}	R_{AC}	$F1_{AC}$	mAP_{AC}	P_{AC}	R_{AC}	$F1_{AC}$	mAP_{AC}	P_{AC}	R_{AC}	$F1_{AC}$	mAP_{AC}
I3D	14.3	1.3	2.1	15.2	12.7	1.9	2.9	21.4	14.9	2.0	3.1	20.3
CF	10.3	1.0	1.6	15.8	9.0	1.5	2.2	22.2	10.7	1.6	2.4	21.0
MLAD Tirupattur et al. (2021)	19.3	7.2	8.9	28.9	18.9	8.9	10.5	35.7	19.6	9.0	10.8	34.8
CTRN	**23.9**	**8.0**	**11.9**	**29.7**	**21.7**	**9.1**	**12.9**	**36.8**	**23.0**	**9.3**	**13.2**	**35.5**

Table 3: Ablation study on Charades dataset using only RGB.

CTRN Components				Charades
RTM	C-GCN	TCN	G-Classifier	Per-frame mAP
×	×	×	×	15.6
✓	×	×	×	16.1
✓	✓	×	×	19.9
✓	×	✓	×	21.4
✓	✓	✓	×	24.7
✓	×	×	✓	18.4
✓	✓	✓	✓	**25.3**

Compared with state-of-the-art methods, our method achieves higher performance on all action-conditional metrics showing that CTRN effectively models action dependencies both within a time-step (i.e. co-occurring action, τ = 0) and throughout time ($\tau > 0$).

To confirm the advancement of our method, we present further comparisons with MLAD. Firstly, we compare the model efficiency and complexity in Fig. 3. MLAD is about 2 times larger in parameters and 3.5 times larger in FLOPs than CTRN while processing the same batch of videos. Hence, our method is more lightweight and computationally efficient than MLAD. Secondly, we visualize the detection results of an example video in Fig. 4 for both MLAD and CTRN. Qualitatively, we find that CTRN can detect the actions more precisely than MLAD.

3.3 Ablation studies

Firstly, in Table 3, we study the complementation of the components in the proposed network on the Charades dataset. We first discuss how RTM leads to a better feature representation of the input spatio-temporal feature map from I3D. RTM is an essential pre-step before class-temporal modelling. Thanks to RTM that filters the class-specific feature, the model can slightly improve the detection performance (+0.5%). We then explore how the different components in CTM affects the action detection performance. We find that both C-GCN and TCN improve the performance w.r.t. a model with only RTM (+23.6, and 32.9% relatively). The action detection performance is further improved by the combination of both C-GCN and TCN, thus reflecting the complementary nature of both the operations. Finally, we study the performance with/without G-Classifier. With the proposed classifier, RTM and RTM+CTM further improve the action detection performance by +2.3% and +0.6% respectively. Note that for the baseline without G-Classifier, similar to the previous work Piergiovanni & Ryoo (2018), we utilize a 1×1 convolution as the classifier. These results show that the different components of CTRN contribute to the overall performance of our network.

Secondly, we analyse if G-Classifier can better detect the co-occurring actions. We compared our method with the standard binary classifier Piergiovanni & Ryoo (2018), Chen et al. Chen et al. (2019) by computing the mAP with the snippets containing more than one action. All three classifiers are build upon the same backbone (CTRN). We find that the standard binary classifier, Chen et al. and G-Classifier achieve 24.1%, 25.3% and 27.6% respectively on Charades, reflecting that G-Classifier can better detect the co-occurring actions.

3.4 Qualitative Analysis

In Fig. 5, we show the adjacency matrix of G-Classifier in Charades (157 classes), which provides the information of all the co-occurring action pairs with high probabilities. For example, *holding a vacuum* & *tiding something on the floor* and *fixing hair* & *watching in a mirror* are the actions

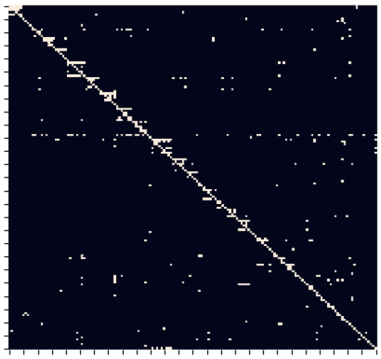

Figure 5: The adjacency matrix of the G-Classifier A_S.

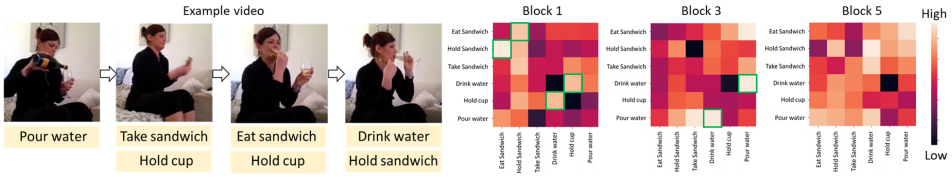

Figure 6: Visualization of the learned C-GCN adjacency matrix A'_C for different layers. Here, we visualize the 1^{st}, 3^{rd} and 5^{th} block's adjacency matrices. For simplicity, we provide only the relevant action classes in the example video.

always occur at the same time. Prior access to such privilege knowledge is crucial for detecting the co-occurring actions in the densely labelled videos.

In CTM, TCN is used to aggregate the temporal information which enables C-GCN to explore action relations at different temporal scales. To validate the usage of these layers, in Fig. 6, we visualize the learned adjacency matrix of C-GCN from three different blocks. We find that in Block 1, C-GCN focuses on capturing the contextual information pertaining to locally related action classes. For example, *eat sandwich* & *hold sandwich* and *drink water* & *hold cup* are always occurring closely in the video. Then we find that, Block 3 has increased the temporal receptive field, thus, C-GCN can capture the long-term dependencies between distant action classes. For example, *Pour water* and *Drink water*. Finally, Block 5 possess the largest receptive field where each local snippet feature contains the whole video information. Therefore, C-GCN in this block models all the potential action relations in the video, resulting in many activated links in the adjacency matrix.

4 Conclusion

In this work, we propose a novel class-temporal relation network for action detection. This network can handle both action class relations at the same time-step, and across different time-steps by using its three key components, namely Representation Transform Module, Class-Temporal Module, and G-Classifier. The network is evaluated on three challenging densely labelled datasets and achieves state-of-the-art performance on all of them. Furthermore, this network has less computation cost and parameters than the representative baseline method. For future perspectives, we will investigate combining C-GCN and TCN into a single layer.

Acknowledgement

This work has been supported by the French government, through the 3IA Cote d'Azur Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002. The authors are also grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support.

REFERENCES

Luis B Almeida. C1. 2 multilayer perceptrons. *Handbook of Neural Computation C*, 1, 1997.

Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 961–970, 2015.

Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4724–4733. IEEE, 2017.

Guang Chen, Can Zhang, and Yuexian Zou. Afnet: Temporal locality-aware network with dual structure for accurate and fast action detection. *IEEE Transactions on Multimedia*, 2020.

Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. Multi-label image recognition with graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5177–5186, 2019.

Rui Dai, Luca Minciullo, Lorenzo Garattoni, Gianpiero Francesca, and François Bremond. Self-attention temporal convolutional network for long-term daily living activity detection. In *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–7. IEEE, 2019a.

Rui Dai, Srijan Das, Saurav Sharma, Luca Minciullo, Lorenzo Garattoni, Francois Bremond, and Gianpiero Francesca. Toyota smarthome untrimmed: Real-world untrimmed videos for activity detection. *arXiv preprint arXiv:2010.14982*, 2020.

Rui Dai, Srijan Das, and Francois Bremond. Learning an augmented rgb representation with cross-modal knowledge distillation for action detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13053–13064, October 2021a.

Rui Dai, Srijan Das, Luca Minciullo, Lorenzo Garattoni, Gianpiero Francesca, and Francois Bremond. Pdan: Pyramid dilated attention network for action detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2970–2979, January 2021b.

Xiyang Dai, Bharat Singh, Joe Yue-Hei Ng, and Larry Davis. Tan: Temporal aggregation network for dense multi-label action recognition. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 151–160. IEEE, 2019b.

Yifei Huang, Yusuke Sugano, and Yoichi Sato. Improving action segmentation via graph-based temporal reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14024–14034, 2020.

Yu-Gang. Jiang, Jingen Liu, Amir Roshan Zamir, George Toderici, Ivan Laptev, Mubarak Shah, and Rahul Sukthankar. THUMOS challenge: Action recognition with a large number of classes. `http://crcv.ucf.edu/THUMOS14/`, 2014.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 156–165, 2017.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

AJ Piergiovanni and Michael S Ryoo. Learning latent super-events to detect multiple activities in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

AJ Piergiovanni and Michael S Ryoo. Temporal gaussian mixture layer for videos. *International Conference on Machine Learning (ICML)*, 2019.

Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision(ECCV)*, 2016.

Gunnar A Sigurdsson, Santosh Divvala, Ali Farhadi, and Abhinav Gupta. Asynchronous temporal fields for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 585–594, 2017.

Praveen Tirupattur, Kevin Duarte, Yogesh Rawat, and Mubarak Shah. Modeling multi-label action dependencies for temporal action localization. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

Huijuan Xu, Abir Das, and Kate Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 5783–5792, 2017.

Mengmeng Xu, Chen Zhao, David S Rojas, Ali Thabet, and Bernard Ghanem. G-tad: Sub-graph localization for temporal action detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10156–10165, 2020.

Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision*, 126(2-4):375–389, 2018.

Runhao Zeng, Wenbing Huang, Mingkui Tan, Yu Rong, Peilin Zhao, Junzhou Huang, and Chuang Gan. Graph convolutional networks for temporal action localization. In *ICCV*, 2019.