UCF-System:Activity Detection in Untrimmed Videos

Ishan Dave*, **Zacchaeus Scheffer***, **Praveen Tirupattur***, **Yogesh Rawat**†, **Mubarak Shah**†
Center for Research in Computer Vision
University of Central Florida, Orlando, Florida
*{ishandave,zaccy,praveentirupattur}@knights.ucf.edu,†{yogesh, shah}@crcv.ucf.edu

1 Method

1.1 Overview

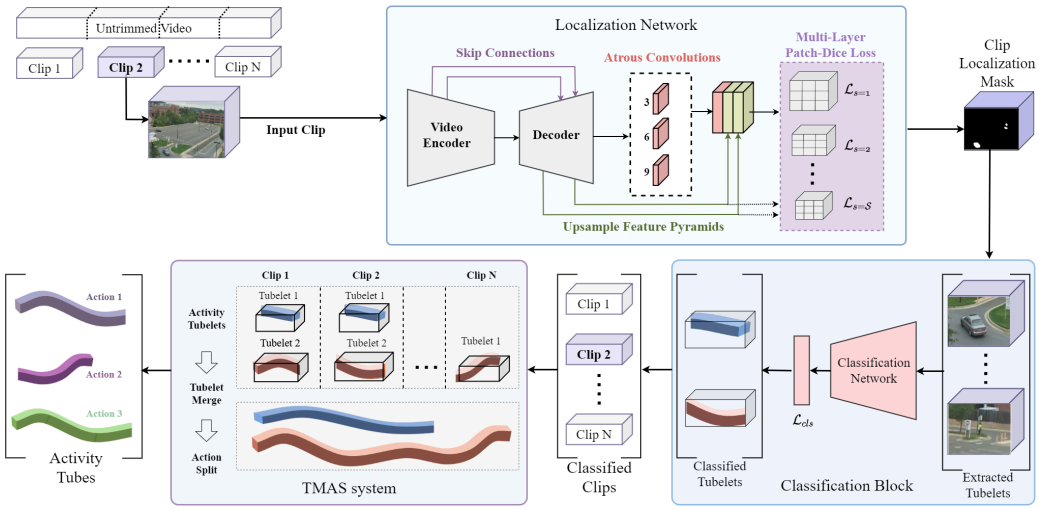

Figure 1: Overview of the proposed system for action detection in untrimmed videos. An untrimmed video is processed clip by clip and fed into the localization network, producing localization masks. Tubelet extraction produces tubelets for each clip, which are then classified and passed to our TMAS system. The classified tubelets are merged to create action-agnostic tubes, from which individual action-specific detections are obtained.

The proposed system takes in a video clip as input and detects all activities in the form of tubelets. The system first operates on entire clip to spatio-temporally localize action tubelets. Once we extract potential tubelets, our classification system identifies all possible activities occurring within each tubelet. These action predictions are then fed into our TMAS system, which simultaneously filters and combines them into accurate and consistent action tubes. As an end result, we obtain spatio-temporal action detections over long untrimmed videos in an online real-time process. The following sections describe the different components of our system.

1.2 Localization Network

The tube extraction process is the first step in the pipeline, responsible for extracting all action tubes from the untrimmed video input. Localizing action regions both temporally and spatially is vital for the classification task as the length and location of action is unknown beforehand. Furthermore, each action tube could comprise of multiple actors performing multiple actions concurrently. This requires actor and action agnostic tube extraction.

As shown in Figure 1, first we divide the untrimmed videos into smaller clips, which are then forwarded into the localization network. Following an encoder-decoder approach, the network produces segmentation masks for action regions, each of which represents an action tubelet. These tubelets are then individually processed by subsequent components of our system. Since this bottom-up segmentation is performed for multiple frames simultaneously, we reduce the time taken to localize multiple actions within a video clip.

The proposed localization network uses an encoder-decoder structure, and extracts class-agnostic action features which can be used to generate segmentation masks; this requires an effective feature extractor as an encoder. To this end, we utilize a 3D convolution based encoder, I3D [2], to learn spatio-temporal features required for activity localization.

In the decoder, we use extracted action features to segment regions from the original input which contain activities. Depending on the task, this segmentation can be coarse (patches of regions) to fine grained (pixel level). We opt to produce relatively fine grained segmentations, keeping a balance between separable action tubes and a reasonable memory utilization. Following recent works in image segmentation [3, 4, 9] and video segmentation [5, 7], we use a decoder structure which combines transpose convolutions and upsampling. Stacking many transpose convolution layers is memory intensive and adds many parameters to the decoder, so we interleave upsampling operations to interpolate features. This results in a shallow decoder network, which prevents over-parameterization and avoids overfitting. Further architectural details of the localization network are presented in the supplementary materials.

Skip Connections: It is difficult to produce fine grained pixel level segmentations using features which have been spatially down-sampled by the encoder. To obtain these fine grained segmentations one would need a deep, memory intensive decoder. To circumvent this we pass low level features from various layers of encoder directly to corresponding layers of the decoder network, which has been found effective for image segmentation [9, 4]. This allows the decoder network to reincorporate essential features that were lost during encoding process, which results in improved pixel level segmentations.

Feature Pyramids: The objects present in surveillance videos have varying scale. Therefore, using information from layers with different feature resolutions helps in segmenting objects of different sizes. The authors in [4, 9] have recently shown that building feature pyramids from various layers, at different scales, aids in learning contextual information. Motivated by this, we stack features from various decoder layers (through upsampling) to obtain feature representations at different scales.

Atrous Convolution: Having contextual information for each pixel aids in the process of learning richer features for fine grained segmentation. Since conventional convolutions have a limited receptive field, they are unable to learn this contextual information without incurring a heavy memory overhead. To address this, we apply atrous convolution, with different receptive windows, on our final feature representation. Atrous convolution applies a convolution operation with dilated kernels centered around a pixel, which effectively increases the receptive field. We apply dilation at multiple rates to infer contextual relations among pixels with varying distances. The authors in [4] utilize atrous convolution for encoder-decoder based segmentation and observed improved performance.

Multi Layer Loss: The final output of our localization network is a segmentation mask where each pixel is assigned a probability of being a part of an action. To train the network, we compute a Binary Cross Entropy (BCE) loss, where each pixel can have a probability between zero (no activity) and one (activity). As our network is deep, only calculating the loss at the final layer affects the convergence time and the gradient update values for earlier layers. Hence, we apply the segmentation loss at multiple decoder layers, which improves the features learned in multiple layers and allows for more distributed backpropagation.

For a layer m with N total pixels, the loss is given as:

$$\mathcal{L}_m(\hat{y}, y) = -1N \sum_{i=0}^{N} [y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)] \tag{1}$$

where $\hat{y}_i$ are predicted probability score of the pixel belonging to an activity and y_i are the ground truth labels of the pixels.

The combined loss for all M layers in the multi layer loss is given by:

$$\mathcal{L}_{loc} = \sum_{m=1}^{M} \mathcal{L}_m \tag{2}$$

Tubelet Extraction: The segmentation output for each clip is a probability mask which isolates potential action tubes. To obtain individual tubelets from this segmentation output, we threshold the output to create a binary mask followed by spatio-temporal connected component extraction. The connected component process will generate tubelets for all spatially and temporally linked pixels.

1.3 Tubelet Classification

The next step in our proposed system is tubelet classification. Our action classification network is a multi-label prediction network, which classifies the actions present within each tubelet. We treat this as a multi-label classification problem because actors can perform multiple activities simultaneously. For example, an actor can perform the actions *Riding* and *activity_carrying* at the same time. We use a 3D-Convolution based deep learning model [6] initialized with pre-trained weights on Kinetics [8] dataset for action classification. We modify the final layer of the model to have a $C + 1$ dimensional output, where C is the number of action classes and the additional output is for the background class. A sigmoid activation is used in the final layer in place of a softmax as this is a multi-label classifier. We use BCE loss to train the classifier which is defined as,

$$\mathcal{L}_{cls}(\hat{y}, y) = -1C + 1 \sum_{i=0}^{C} [y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)] \tag{3}$$

where $\hat{y}_i$ is the prediction and y_i is the ground truth label.

1.4 TMAS Algorithm

To merge the tubelets and obtain the final action tubes, we propose the Tubelet-Merge Action-Split algorithm (TMAS). Each tubelet t_i is described as follows: $\left(f_1^i, f_2^i, \mathbf{b}^i, \mathbf{a}_c^i\right)$ where f_1^i is the start time, f_2^i is the end time, $\mathbf{b}^i$ are the bounding boxes for each frame of the tubelet, and $\mathbf{a}_c^i$ are the frame-level action probability scores for each action class $c \in \{0, 1, ...C\}$, where 0 is background. First, we merge the tubelets into action-agnostic tubes of varying length; then we split these action-agnostic tubes into a set of action-specific tubes which contain the spatio-temporal localizations for the various activities in the video.

Tubelet-Merge The procedure to merge tubelets into action-agnostic tubes is described in Algorithm 1. The temporally sequential stream of tubelets coming from the classification network are passed to the Tubelet-Merge procedure as input. The set of candidate tubes is initialized with the first tubelet. For each subsequent tubelet, we look for spatio-temporal overlap with the existing candidate tubes. This results in four possible outcomes: 1) If there is no overlap, the tubelet itself becomes a new candidate tube, 2) If there is a unique match found between a candidate tube and the tubelet, they are merged and become a new candidate tube, 3) if the tublet has an overlap with multiple candidates, then the tubelet becomes a new candidate, 4) if multiple tublets have an overlap with a single candidate tube, then the tubelet with the highest overlap is merged with that candidate and the other tubelets become separate candidate tubes. Once all tubelets are checked, the candidate tubes become the final action-agnostic tubes.

Action-Split From the action-agnostic tubes we obtain action-specific spatio-temporal localizations using the Action-Split procedure described in Algorithm 2. We start by smoothing out per-frame action confidence scores; which accounts for fragmentation caused by action miss-classifications. Then we build the action-specific tubes by checking for continuous occurrences of each action class; this allows several occurrences of the same activity to occur within a single tube. For instance, a person *walking* might stop and *stand* for several seconds and start walking again; this entire sequence will be contained in a single spatio-temporal tube, but the Action-Split procedure will correctly generate two separate instances of *activity_walking* and one instance of *activity_standing*. To be robust to classification errors, action tubes with the same action label that are within a limited temporal neighborhood are combined together to form a single continuous action prediction.

Algorithm 1 The Tubelet-Merge algorithm which merges tubelets into action-agnostic tubes. The CHECKEND function determines if a candidate tube becomes a final tube or is merged with another candidate.

Input: A stream of tubelets, **S**, from the classifier
Output: A set of action-agnostic spatio-temporal tubes, T_{done}
Notation: $Inter_{temp}$ calculates temporal overlap between tubelets.
$|\mathbf{M}[(t_c, *)]|$ returns the cardinality of the set $\{t : \mathbf{M}[(t_c, t)] > 0\}$.

procedure TUBELET-MERGE(**S**)
  $T_{prev}, T_{done} \leftarrow \{\}$ ▷ Initialize candidate and final tubes
  $\mathbf{M} \leftarrow initializehashtable$
  while $t_c in$ **S do** ▷ Continue until the stream of tubelets ends
    for all $t_p in T_{prev}$ **do**
      if $Inter_{temp}(t_p, t_c) > 0$ **then**
        $\mathbf{M}[(t_p, t_c)] \leftarrow IoU(t_p, t_c)$
      else
        CHECKEND(t_p, T_{prev}, **M**)
    append t_c to T_{prev} ▷ Tubelet becomes a candidate tube
  while $T_{prev} isnotempty$ **do** ▷ Deals with remaining candidates
    $t_p \leftarrow T_{prev}[0]$
    CHECKEND(t_p, T_{prev}, **M**)
  return T_{done}

function CHECKEND($t_p, T_{prev}, \mathbf{M}$)
  if $|\mathbf{M}[(t_p, *)]| == 0$ **then**
    MOVE(t_p, T_{prev}, T_{done}) ▷ Moves t_p from T_{prev} to T_{done}
  else if $|\mathbf{M}[(t_p, *)]| == 1$ **then**
    $t_i \leftarrow \max_{t_i} \mathbf{M}[(t_p, t_i)]$
    if $|\mathbf{M}[(*, t_i)]| == 1$ **then**
      MERGE(t_p, t_i, T_{prev}, **M**)
    else
      MOVE(t_p, T_{prev}, T_{done})
  else
    $t_i \leftarrow \max_{t_i} \mathbf{M}[(t_p, t_i)]$
    MERGE(t_p, t_i, T_{prev}, **M**)

function MERGE($t_1, t_2, T_{prev}, \mathbf{M}$) ▷ Merges two candidate tubes
  $t_1 \leftarrow (f_1^1, f_2^2, \{\mathbf{b}^1, \mathbf{b}^2\}, \{\mathbf{a}^1, \mathbf{a}^2\})$ ▷ $\{\}$ is concatenation
  $removet_2 fromT_{prev}$
  $\mathbf{M}[t_1, t_i] \leftarrow \mathbf{M}[t_2, t_i]$ ▷ $Donefor all t_i where \mathbf{M}[t_2, t_i] \geq 0$

Runtime Complexity The worst-case runtime of our TMAS algorithm is $\mathcal{O}\left(n^2\right)$, where n is the total number of candidate tubes at any given time. However, we sequentially process our tubelets and constantly shift the candidate tubes which can not have any possible future match to the set of final tubes. Therefore, the set of candidate tubes at any particular time is reasonably small and our TMAS algorithm contributes negligible overhead to our system's overall computation time.

2 Experiments

2.1 Implementation Details

We use I3D backbone for the action localization and Wide-ResNet-50 model for the tubelet classification. For the training networks, we use Adam Optimizer with a base learning rate of 1e-3 and a learning rate scheduler which drops the learning rate to 1/10th of its value at the loss plateau.

2.2 Results

We train our pipeline on the VIRAT dataset with training set of TRECVID 2020 [1] split. The comparison with the other methods are shown in Table 1, which is reported on the TRECVID ActEV 2020 Evaluation leaderboard.

Algorithm 2 The Action-Split algorithm which converts the action-agnostic tubes into action-specific predictions.

Input: A set of action-agnostic tubes, T, and a set of actions, C
Output: A set of spatio-temporal action-specific tubes, A_G
Notation: The hyperparameters $\tau, \alpha, \beta, and \gamma$ are described in the supplementary materials. $a_c^i[f]$ and $t_i[f]$ contain the action prediction scores and tube information at frame f, respectively.

procedure ACTION-SPLIT(T)
    $A_G \leftarrow \{\}$ ▷ Initializes the action-specific tubes
    for all $t_i in T$ **do**
        $t_{smooth} \leftarrow SMOOTH(t_i)$
        for all $cin1 : C$ **do** ▷ Loop through each action class
            $a_L \leftarrow EXTRACT(t_{smooth}, c)$
            $appenda_L toA_G$
    return A_G

function SMOOTH(t_i)
    for all $finf_1^i : f_2^i$ **do**
        $a_c^i[f] \leftarrow \frac{1}{2\tau+1} \sum_{k=-\tau}^{\tau} a_c^i[f+k]$
    return t_i

function EXTRACT(t_i, c) ▷ Extracts tubes of a specific class
    $A_L, a_l \leftarrow \{\}$ ▷ Initialize extracted action tubes and a placeholder
    $count \leftarrow 0$
    for all $finf_1^i : f_2^i$ **do**
        if $a_c^i[f] > \alpha$ **then** ▷ Continue current action tube
            $appendt_i[f] toa_l$
            $count \leftarrow 0$
        else
            $count \leftarrow count + 1$
        if $count > \beta$ **then** ▷ Current action tube is finished
            $appenda_l toA_L$
            $a_l \leftarrow \{\}, count \leftarrow 0$
    $removetubesshorterthan\gamma fromA_L$
    return A_L

Rank	*Team name*	*System name*	*Partial AUDC**	*mean-p_ miss@0.15tfa*	*mean-w_p_ miss@0.15rfa*
1	INF	INF	0.42307	0.33241	0.80965
2	BUPT-MCPRL	MCPRL_S1	0.55515	0.48779	0.84519
3	**UCF**	**UCF-P**	**0.58485**	**0.5473**	**0.8354**
4	TokyoTech_AIST	TTA-SF2	0.79753	0.75502	0.87889
5	CERTH-ITI	P	0.86576	0.84454	0.88237
6	Team UEC	UEC	0.95168	0.95329	0.983
7	kindai_kobe	kindai_ogu _baseline	0.9682	0.96443	0.95665

Table 1: Comparison of methods tested on NIST TRECVID-2020 evaluation set

Acknowledgements

This research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA R&D Contract No. D17PC00345. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

References

[1] George Awad, Asad A. Butt, Keith Curtis, Yooyoung Lee, Jonathan Fiscus, Afzal Godil, Andrew Delgado, Jesse Zhang, Eliot Godard, Lukas Diduch, Jeffrey Liu, Alan F. Smeaton, Yvette Graham, Gareth J. F. Jones, Wessel Kraaij, and Georges Quénot. Trecvid 2020: comprehensive campaign for evaluating video retrieval tasks across multiple application domains. In *Proceedings of TRECVID 2020*. NIST, USA, 2020.

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[5] Kevin Duarte, Yogesh Rawat, and Mubarak Shah. Videocapsulenet: A simplified network for action detection. In *Advances in Neural Information Processing Systems*, pages 7610–7619, 2018.

[6] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[7] Rui Hou, Chen Chen, and Mubarak Shah. An end-to-end 3d convolutional neural network for action detection and segmentation in videos. *arXiv preprint arXiv:1712.01111*, 2017.

[8] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[9] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019.