Gabriella: An Online System for Real-Time Activity Detection in Untrimmed Surveillance Videos

Mamshad N Rizve*, Ugur Demir*, Praveen Tirupattur*, Aayush J Rana*,
Kevin Duarte*, Ishan Dave*, Yogesh S Rawat† and Mubarak Shah†
Center for Research in Computer Vision
University of Central Florida, Orlando, Florida, USA
Email: *[nayeemrizve, ugur, praveentirupattur, aayushjr, kevin, ishandave]@knights.ucf.edu, †[yogesh, shah]@crcv.ucf.edu

***Abstract*—Activity detection in surveillance videos is a difficult problem due to multiple factors such as large field of view, presence of multiple activities, varying scales and viewpoints, and its untrimmed nature. The existing research in activity detection is mainly focused on datasets, such as UCF-101, JHMDB, THUMOS, and AVA, which partially address these issues. The requirement of processing the surveillance videos in real-time makes this even more challenging. In this work we propose Gabriella, a real-time online system to perform activity detection on untrimmed surveillance videos. The proposed method consists of three stages: tubelet extraction, activity classification, and online tubelet merging. For tubelet extraction, we propose a localization network which takes a video clip as input and spatio-temporally detects potential foreground regions at multiple scales to generate action tubelets. We propose a novel Patch-Dice loss to handle large variations in actor size. Our online processing of videos at a clip level drastically reduces the computation time in detecting activities. The detected tubelets are assigned activity class scores by the classification network and merged together using our proposed Tubelet-Merge Action-Split (TMAS) algorithm to form the final action detections. The TMAS algorithm efficiently connects the tubelets in an online fashion to generate action detections which are robust against varying length activities. We perform our experiments on the VIRAT and MEVA (Multiview Extended Video with Activities) datasets and demonstrate the effectiveness of the proposed approach in terms of speed (∼100 fps) and performance with state-of-the-art results. The code and models will be made publicly available.**

I. Introduction

Deep convolutional neural networks have achieved impressive action classification results in recent years [3], [31], [32]. Similar advancements have been made for the tasks of action detection in *trimmed videos* [6], [15], [29], where the spatial extents of the actions are estimated, and *temporal* action localization in *untrimmed videos* [23], [35], where only the temporal extents of the activities are predicted. However, these improvements have not been transferred to action detection in untrimmed videos, where both the spatial and temporal extents of the activities must be found; current approaches have yet to achieve high performance on this difficult task.

Action detection in untrimmed videos has several major challenges: multiple activity types may occur simultaneously, multiple actors may be present, and the temporal extents of the activities are unknown. Videos in trimmed action detection datasets, like AVA [11], contain multiple actors and activities simultaneously, but each video is trimmed into three second clips which have bounding-box annotations only on the center frame. Untrimmed action detection datasets, like THUMOS'14 [14], are comprised of untrimmed videos, but each video contains only one or two actors performing the same action. Although difficult, these datasets do not contain all the aforementioned challenges, which results in action detection methods that perform poorly when evaluated on videos containing all these challenges. Therefore, in this work we focus on the surveillence video datasets: VIRAT [21] and MEVA (Multiview Extended Video with Activities) [1]. Not only do these two action detection datasets have untrimmed videos with multiple activity types and multiple actors, but also they are comprised of multiple viewpoints and contain several actors performing multiple actions *concurrently*. These actors have varying scales and actor sizes tend to be extremely small relative to the video frame, which makes the detection of activities in these datasets extremely difficult. Figure 1 shows sample frames from the VIRAT dataset and compares them with frames from the THUMOS'14 and AVA datasets.

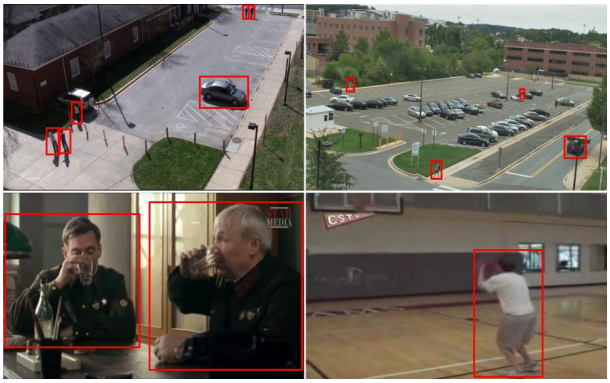

Fig. 1. **Top**: Two Sample frames from different scenes of the VIRAT dataset showing variation in perspective, scale and field-of-view. **Bottom**: Sample frames from the AVA [11] (left) and THUMOS'14 [14] (right) dataset. The VIRAT dataset contains a greater number of concurrent actions as well as a greater variety of action scales (both spatially and temporally).

We focus on untrimmed surveillance videos and propose **Gabriella**, an online real-time system for action detection. Our method is composed of three modules: tubelet localization, tubelet classification, and tubelet merging. Our action localization module generates pixel-level foreground-background

segmentations which localize actions in short video clips. These pixel-level localizations are turned into short spatio-temporal action tubelets, which are passed to a classification network to obtain multi-label predictions. After classification, the tubelets must be linked together to obtain the final detections with varying length; to this end, our novel online Tubelet-Merge Action-Split (TMAS) algorithm merges these short action tubelets into final action detections.

Conventional action detection methods make use of pre-trained, frame-based object detectors to localize all potential actors within the video. Frame-based object detection systems, [10], have two main issues: 1) processing each frame independently requires large amounts of computation, which reduces the overall speed of the system and leads to temporally inconsistent detections between adjacent frames, and 2) all objects within the frame are detected, even those which are not performing actions. Our action localization module processes multiple frames simultaneously and only produces tubelets which correspond to possible actions within the video. This results in temporally consistent localizations as well as a reduction in the number of proposals, which drastically increases the speed of the overall system. To improve the accuracy of our localization network, we propose a novel Patch-Dice loss. The original global Dice loss [30] allows networks to account for large imbalances between foreground and background (which is the case for surveillance videos with very small actors). However, it does not take into account the variation in scale of different foreground objects/actions, which leads networks to focus on only the largest actions. The Patch-Dice loss solves this, allowing our network to localize actions of any scale by computing loss on local neighborhoods of each frame.

Since activities in untrimmed videos can vary in length, it is necessary to handle both short, atomic activities, like *'opening a door'* or *'exiting a vehicle'*, as well as long, repetitive actions like *'walking'* or *'riding'*. To this end, our system processes videos in an online fashion. Once the short tubelets have been localized and classified, our Tubelet-Merge Action-Split (TMAS) algorithm merges them into final action tubes of varying length. By classifying short tubelets and merging them into action tubes, our system successfully detects both atomic and repetitive actions. Also, since each tube can have multiple activities co-occurring simultaneously, the TMAS algorithm splits them to successfully isolate individual activities. Due to the online nature of the TMAS algorithm, as well as the efficiency of the localization network, our system generates action detections at over 100 fps, greatly exceeding the speed of contemporary action detection methods.

Our contributions are as follows:

- We introduce **Gabriella**, a *real-time online* system that performs action detection in *untrimmed* surveillance videos at $\sim$**100** frames per second.
- We propose an action localization network, trained using a novel *Patch-Dice loss*, to detect *activity agnostic tubelets* of varying scale which significantly reduces the processing time of the system.
- The proposed *TMAS* tubelet merging algorithm efficiently connects the tubelets in an *online* fashion and produces detections which are consistent across time as well as robust against varying length activities.

We evaluate the proposed approach on the VIRAT and MEVA datasets, and obtain state-of-the-art results in terms of both speed and performance.

II. Related Works

Convolutional Neural Networks (CNN) have been studied for video analysis and applied successfully for the action recognition problem [3], [31]. Earlier approaches fuse 2D frame features to extract temporal information [16], while recent works mostly apply 3D convolutions to extract spatio-temporal features simultaneously [3], [7], [31]. The works in [7], [28] use two stream network architectures to further exploit temporal dependencies.

Most action classifiers expect short trimmed videos, however, this is unrealistic for action recognition in real world surveillance videos. Predicting the temporal extents of actions is necessary for reliable recognition. In [23], a new layer is proposed to temporally localize activities in videos of MultiThumos dataset [35]. Most works on spatial action detection rely on a region proposal network [25] to detect multiple objects in each frame and combine them temporally to generate action tubelets [22], [34]. In [13], a 3D CNN network efficiently predicts frame-wise background-foreground segmentation map and extrapolates the action tubes. However, such approaches becomes computationally inefficient as the number of proposals grows larger, making it unsuitable for real time performance.

Action detection in untrimmed surveillance videos require to address multiple challenges. In [10], a frame level object detection and optical flow based model is proposed to solve action detection on the VIRAT [21] dataset. They use hierarchical clustering on all detected objects in a video to obtain tube proposals and use optical flow to perform action classification. In [20] authors also perform frame level object detection followed by tracking to generate proposals. These approaches are computationally expensive and not suitable for online processing. Recent work by [9] improves upon [10] to make the system real-time by reducing cluster points per video, reduce subsampling rate, and use GPU accelerated optical flow computation. However, this approach produces too many proposals which ends up affecting the system performance. Our method uses a 3D CNN network for spatio-temporal action segmentation which produces temporally consistent predictions with fewer proposals. Additionally, our system processes videos in an online fashion without using computationally expensive methods (region proposal network, tracking and optical flow) and achieves better performance in real-time.

III. Methodology

A. Overview

An overview of our system can be found in Figure 2. Each untrimmed video is first split into video clips, which

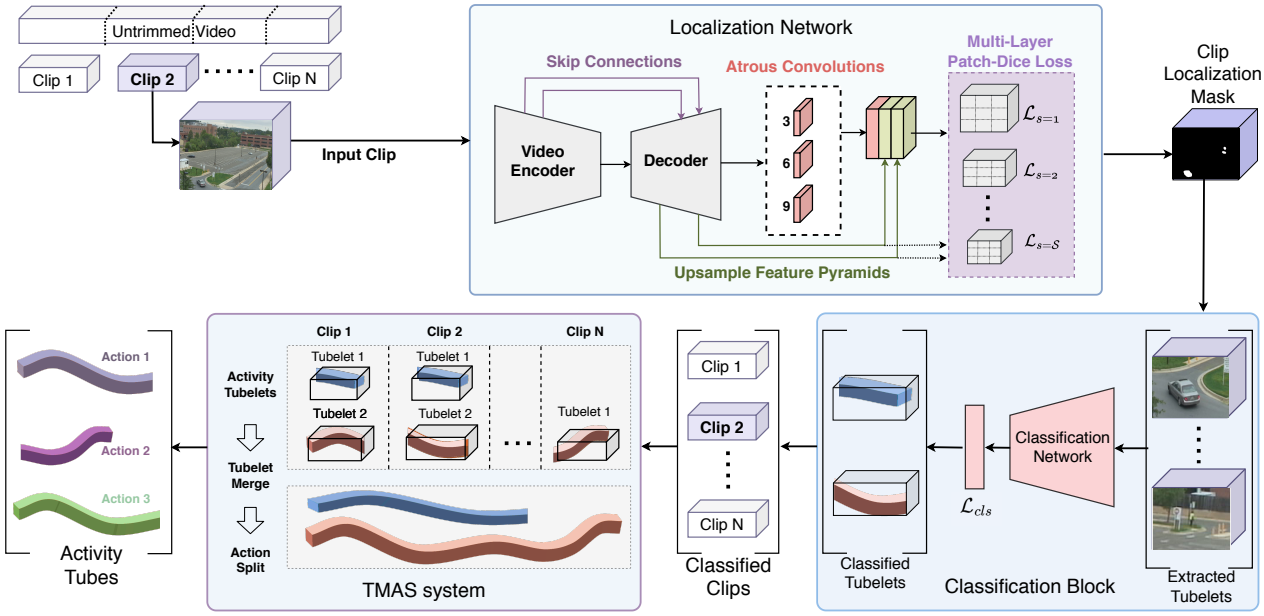

Fig. 2. Overview of the proposed **Gabriella** system for action detection in untrimmed videos. An untrimmed video is processed clip by clip and fed into the localization network, producing localization masks. Tubelet extraction produces tubelets for each clip, which are then classified and passed to our TMAS system. The classified tubelets are merged to create action-agnostic tubes, from which individual action-specific detections are obtained.

are each passed to a localization network to localize potential action tubelets. Then, a network classifies all possible action classes present within each tubelet. Finally, our TMAS system simultaneously filters and combines these short tubelets into longer action tubes. Since our system works on individual video clips in an online fashion, it is able to produce these action detections in real-time. In this section, we will describe the different components of our proposed method.

B. Localization Network

The localization network processes a short video clip and localizes all actions within the clip. The network uses an encoder-decoder structure which extracts class-agnostic action features and generates segmentation masks. We utilize a 3D convolution based encoder, I3D [3], to extract spatio-temporal features required for activity localization. The decoder must use these features to segment regions from the original input which contain activities. Following recent works in image segmentation [4], [5], [19] and video segmentation [6], [13], we use a decoder structure which combines transpose convolutions and upsampling. Stacking many transpose convolution layers is computationally intensive, so we interleave upsampling operations to interpolate features. This results in a shallow decoder network, which prevents over-parameterization and avoids overfitting.

To obtain accurate action localizations, our decoder makes use of skip connections [27], feature pyramids [19], and atrous convolutions [4]. The decoders' input features have been down-sampled by the encoder, so to obtain fine-grained segmentations we utilize skip connections from higher resolution layers of the encoder. Since many of the actors within surveillance videos vary in scale, decoder makes use of feature pyramids: we stack features from various decoder layers (through upsampling) to obtain feature representations at different scales. Furthermore, we apply atrous convolutions to the final feature representation of the decoder so that the decoder may learn the contextual information necessary for action localization.

Patch-Dice Loss: The final output of our localization network is a segmentation mask, $\hat{y}$, where each pixel is assigned a probability of being a part of an action. Given the ground-truth foreground/background mask, y, the network is trained end-to-end using a sum of two losses. The first is the binary cross-entropy (BCE) loss,

$$\mathcal{L}_{\text{BCE}}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i log(\hat{y}_i) + (1-y_i) log(1-\hat{y}_i)\right], \qquad (1)$$

computed over all N pixels. Since the actors tend to be small in surveillance videos, there is a large imbalance between the number of foreground and background pixels, which causes BCE to miss-localize some actoins. A standard approach to remedy this is to use the Dice loss [30]; however, we find that the large variation in scale between different foreground objects (actors) leads the network to focus on the larger actions and ignore smaller actions.

To this end, we propose a Patch-Dice Loss (PDL),

$$\mathcal{L}_{\text{PDL}}(y, \hat{y}) = \sum_{k=1}^{K}\left(1 - \frac{2\sum_{i=1}^{M} p_{k,i} * \hat{p}_{k,i}}{\sum_{i=1}^{M} p_{k,i}^2 + \sum_{i=1}^{M} \hat{p}_{k,i}^2 + \epsilon}\right) \qquad (2)$$

where K is the number of patches, M is the number of pixels per patch, and $p_{k,i}$ denotes the probability value assigned to pixel i in patch k. This loss splits frames into many local

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 tubes, T_{done}
Notation: $\text{Inter}_{\text{temp}}$ calculates temporal overlap between tubelets. $|\mathbf{M}[(\tau_c, *)]|$ returns the cardinality of the set $\{\tau : \mathbf{M}[(\tau_c, \tau)] > 0\}$.

procedure TUBELET-MERGE(**S**)
  $T_{prev}, T_{done} \leftarrow \{\}$ ▷ Initialize candidate and final tubes
  M $\leftarrow$ initialize hash table
  while τ_c in **S do** ▷ Continue until the stream of tubelets ends
    for all τ_p in T_{prev} **do**
      if $\text{Inter}_{\text{temp}}(\tau_p, \tau_c) > 0$ **then**
        $\mathbf{M}[(\tau_p, \tau_c)] \leftarrow IoU(\tau_p, \tau_c)$
      else
        CHECKEND(τ_p, T_{prev}, **M**)
    append τ_c to T_{prev} ▷ Tubelet becomes a candidate tube
  while T_{prev} is not empty **do** ▷ Deals with remaining candidates
    $\tau_p \leftarrow T_{prev}[0]$
    CHECKEND(τ_p, T_{prev}, **M**)
  return T_{done}

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

function MERGE($\tau_1, \tau_2, T_{prev}, \mathbf{M}$) ▷ Merges two candidate tubes
  $\tau_1 \leftarrow (f_1^1, f_2^2, \{\mathbf{b}^1, \mathbf{b}^2\}, \{\mathbf{a}^1, \mathbf{a}^2\})$ ▷ $\{\}$ is concatenation
  remove τ_2 from T_{prev}
  $\mathbf{M}[\tau_1, \tau_i] \leftarrow \mathbf{M}[\tau_2, \tau_i]$ ▷ Done for all τ_i where $\mathbf{M}[\tau_2, \tau_i] \geq 0$

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 action-specific tubes, A_G
Notation: The hyperparameters $\kappa, \alpha, \beta,$ and γ are described in the supplementary materials. $a_c^i[f]$ and $\tau_i[f]$ 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 τ_i in T **do**
    $\tau_{smooth} \leftarrow$ SMOOTH(τ_i)
    for all c in $1 : C$ **do** ▷ Loop through each action class
      $a_L \leftarrow$ EXTRACT(τ_{smooth}, c)
      append a_L to A_G
  return A_G

function SMOOTH(τ_i)
  for all f in $f_1^i : f_2^i$ **do**
    $a_c^i[f] \leftarrow \frac{1}{2\kappa+1} \sum_{k=-\kappa}^{\kappa} a_c^i[f+k]$
  return τ_i

function EXTRACT(τ_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 f in $f_1^i : f_2^i$ **do**
    if $a_c^i[f] > \alpha$ **then** ▷ Continue current action tube
      append $\tau_i[f]$ to a_l
      $count \leftarrow 0$
    else
      $count \leftarrow count + 1$
    if $count > \beta$ **then** ▷ Current action tube is finished
      append a_l to A_L
      $a_l \leftarrow \{\}, count \leftarrow 0$
  remove tubes shorter than γ from A_L
  return A_L

neighborhoods (patches), and computes the dice loss on each patch; this forces the network to segment actions of any size.

Our final loss is a weighted sum of BCE and PDL, calculated over multiple scales:

$$\mathcal{L}_{\text{loc}} = \sum_{s=1}^{S} \lambda_1 \mathcal{L}_{\text{BCE}} \left(y^{(s)}, \hat{y}^{(s)} \right) + \lambda_2 \mathcal{L}_{\text{PDL}} \left(y^{(s)}, \hat{y}^{(s)} \right), \quad (3)$$

where S denote number of layers and for a layer s, $y^{(s)}$ and $\hat{y}^{(s)}$ are the ground-truth and predicted segmentations respectively.

Tubelet Extraction: The segmentation output for each clip is a foreground-background confidence 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 [8], [33] process will generate tubelets for all spatially and temporally linked pixels.

C. Classification Network

Once the tubelets are extracted, our system assigns it action labels. Our classification network is an R(2+1)D network [32], that generates $C + 1$ probability outputs, where C is the number of action classes and the additional output is for the background class (used in the case where no action is present in the tubelet). Since multiple actions could occur simultaneously in one tube, we treat this as a multi-label classification problem. Instead of using a softmax activation for the probability outputs, which is common for single-label action classifiers, a sigmoid activation is used which allows multiple actions classified within a single tubelet. To train this classifier, we use a BCE loss (equation 1) summed over all $C + 1$ probability outputs.

D. TMAS Algorithm

To merge the tubelets and obtain the final action detections (tubes), we propose the Tubelet-Merge Action-Split algorithm (TMAS). Each tubelet, denoted τ_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 $c \in \{0, 1, ...C\}$, where 0 is background. The TMAS algorithm consists of two steps. 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 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 input to this is a temporally sequential stream of tubelets coming from the classification network. The set of candidate tubes is initialized with the first tubelet. For each subsequent tubelet,

we calculate the spatio-temporal overlap with the existing candidate tubes. This results in four possible outcomes: 1) if there is no overlap, the tubelet becomes a new candidate tube, 2) if there is overlap between a single candidate tube and the tubelet, they are merged and become a new candidate tube, 3) if the tubelet 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 output action-agnostic tubes.

Action-Split: From the action-agnostic tubes we obtain action-specific detections 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 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.

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.

IV. Experimental Setup

Datasets: We evalaute our method on two large-scale action detection datasets with untrimmed surveillance videos: VIRAT and MEVA. The first dataset consists of videos from the VIRAT [21] dataset with added action detection annotations. It contains 64 videos (2.47 hours) for training and 54 videos (1.93 hours) for validation, with annotations for 40 different activities involving people and vehicles. There is also a held-out test set containing 246 videos (10.11 hours) whose annotations are not made public. The MEVA dataset [1] consists of 1056 annotated videos, each 5 minutes long, covering both indoor and outdoor scenes. We use 936 of these videos for training and of the remaining 120 we use 50 for validation and 70 for local evaluation. These videos are annotated with 37 different activities, mainly involving humans and vehicles. These annotations follow long-tail distribution, i.e, there are few activities which have many annotated instances as they occur very frequently and there are many activities which have very few instances as they are rare. For the final testing, the system is submitted to an evaluation server where a set of sequestered videos are used for evaluation of the system. More information about the sequestered data and testing protocol for MEVA can be found at ActEv [1] website.

A. Training Details

Network Training The videos for both datasets are high resolution, so we rescale the videos to a lower resolution of 800×448, while maintaining the aspect ratio. The localization network uses a stack of 16 frames to obtain the binary segmentation masks; the ground-truth for these masks is the bounding box annotations for all actions present within the given frames (regions within the bounding boxes are considered foreground and other regions are considered background). The network is trained using SGD [26], with a learning rate of 1e-3 for about 100000 iterations. For our BCE+PDL training we have set both the values of λ_1 and λ_2 to 1. Our classification model is trained with a clip length of 16 frames (with a skip rate of 1 to obtain a second long clip) and a spatial-resolution of 112×112. For the classifier, we use the ADAM optimizer [18] with a learning rate of 1e-4 for 75000 iterations on two NVIDIA GeForce Titan X GPUs.

Data augmentation To increase the diversity of data, we pre-process the videos which are input to the network during training. For the localization network, we apply frame jitter and cropping to simulate shaking of a camera which can happen due to wind. For the classification network, we perform cropping, resizing, and horizontal flipping on the input tubes. Moreover, we use both ground-truth and predicted (outputs of the localization network) tubes for the training of the classifier.

One of the challenging issues with both the VIRAT and MEVA datasets is data imbalance. To balance the data, we first under-sample the classes with largest number of samples. Also, we perform multi-scale cropping and horizontal flipping on classes with the fewest number of samples. Lastly, we perform frame reversal to generate new clips for complementary pairs of classes such as (*Opening*, *Closing*), (*Loading*, *Unloading*), (*Entering*, *Exiting*), and (*Open_Trunk*, *Close_Trunk*) to increase the number of samples for these classes.

Metrics We evaluate the performance of our system using several metrics: probability of missed detection at fixed rate of false alarm per minute ($P_{miss}@R_{FA}$), probability of missed detection at fixed time-based false alarm per minute ($P_{miss}@T_{FA}$), and partial area under the Detection Error Tradeoff curve (AUDC). These measure the quality of action detections for the action detection task. To calculate these metrics, a one-to-one correspondence is found between the ground-truth actions and the detected actions; ground-truth actions without a corresponding detection are missed detections, while detections without corresponding ground-truth actions are false alarms. For more detailed explanations of the evaluation metrics as well as evaluation code, we refer to TRECVID-2019 [2] and MEVA SDL [1]. For ablations of the classification network, we use standard multi-label classification metrics: precision, recall, and F1-score.

[1] https://actev.nist.gov/sdl

Team	$P_{miss@0.15}T_{FA}$	$P_{miss@0.15}R_{FA}$	AUDC
Fraunhofer	0.7747	0.8474	0.8270
vireoJD-MM	0.5482	0.7284	0.6012
NTT_CQUPT	0.5112	0.8725	0.6005
Hitachi	0.5099	0.8240	0.5988
BUPT-MCPRL	0.4328	0.7491	0.5240
MUDSML [20]	0.3915	0.7979	**0.4840**
Ours	**0.3858**	**0.7022**	0.4909

TABLE I
TEMPORAL LOCALIZATION RESULTS ON VIRAT TEST SET FROM TRECVID-2019 LEADERBOARD. ALL THE METRICS RELATE TO THE MISS-RATE, SO LOWER VALUES INDICATE BETTER PERFORMANCE.

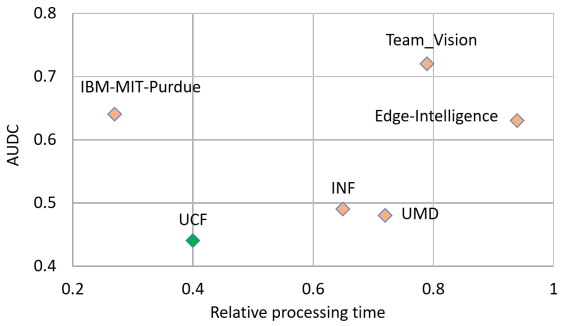

Fig. 3. Runtime vs AUDC Score of different systems on MEVA test set.

V. RESULTS

In this section we present the evaluation of our overall method as well as ablations on its individual components.

A. System Evaluation

Testing Environment: The proposed system is evaluated on the remote test servers provided by NIST. The testing system expects a program that is compatible with the ActEV Command Line Interface (CLI) protocol [2]. CLI protocol helps our system to communicate with the testing environment and to be executed remotely. It runs all of the submitted algorithms and reports the scores on the publicly available leaderboard.

VIRAT: We present our system's performance on the VIRAT test set from TRECVID-2019 leaderboard in Table I for temporal action localization. We compare our results with the teams who submitted to the leaderboard and as you can see from the results we outperform every other team on $P_{miss@0.15}T_{FA}$ and $P_{miss@0.15}R_{FA}$ metrics. Compared to method MUDSML (the best performing comparison) we achieve similar results (within 0.7%) with respect to AUDC, but we achieve over a 9.5% improvement in terms of $P_{miss@0.15}R_{FA}$.

MEVA: We present the results of our system on the MEVA sequestered test set in Table II. Our method achieves state-of-the-art results in both metrics: improving AUDC by over 3.5% and $P_{miss@0.04}T_{FA}$ by over 2%. Notably, our system outperforms others without the need of pre-trained object detectors for localization or optical flow for classification.

[2]https://actev.nist.gov/sdl

Team	AUDC	$P_{miss@0.15}T_{FA}$	Processing Time
Team-Vision	0.717	0.776	0.793
IBM-MIT-Purdue	0.641	0.733	0.272
Edge-Intelligence	0.628	0.754	0.939
INF	0.489	0.559	0.646
UMD [9]	0.475	0.544	0.725
Ours	**0.438**	**0.523**	0.362

TABLE II
TEMPORAL LOCALIZATION RESULTS ON MEVA SEQUESTERED TEST SET. ALL THE METRICS RELATE TO THE MISS-RATE, SO LOWER VALUES INDICATE BETTER PERFORMANCE. THESE RESULTS ARE FROM THE PUBLICLY AVAILABLE LEADERBOARD [2]. $P_{miss}R_{FA}$ IS NOT INCLUDED IN THIS TABLE AS IT IS NOT MADE PUBLIC.

Model	IoU
BCE	62.27%
BCE + Dice Loss	62.35%
BCE + PDL	63.43%

TABLE III
ABLATION EXPERIMENTS TO STUDY THE EFFECT OF THE PATCH-DICE LOSS ON THE LOCALIZATION NETWORK.

B. Run-time analysis

We compare the speed and performance of our system with other systems on MEVA test set in Figure 3. All systems are tested on 4 NVIDIA RTX 2080 Ti GPUs which is the standard configuration for the evaluation system [2]. Our online action detection method outperforms all most all the other systems by a wide margin. Our method also achieves higher than real-time speed, 45fps, on a *single* GPU. This large difference in speed is mainly due to our localization network: it directly generates tubelets instead of relying on per-frame object detections for proposal generation. This greatly reduces the number action proposals and allows us to process videos online very efficiently.

Fig. 4. Qualitative results from the localization network overlaid on the input frame; the three rows demonstrate action masks obtained from the ground-truth and generated using the BCE, and Patch-Dice loss respectively. The first two columns demonstrate that the network trained with Patch-Dice loss can detect small actions that are missed or partially detected if BCE loss was used. The third column demonstrates that the localization masks generated using the Patch-Dice loss have better action boundaries.

C. Ablations

Patch-Dice Loss: We run several ablations to evaluate the effectiveness of the Patch-Dice Loss, and present the results in Table III. Using PDL during training leads to an improvement

Fig. 5. Qualitative results of our system on some sample local evaluation videos. Each row is a sample output from our system, showing spatio-temporal localization and classification of actions in specific frames of the input video. Each action type is shown with different colored bounding box. The example activities shown here are vehicle turns left, vehicle reverses, vehicle starts, person talks to person, and person opens vehicle door. These results demonstrate the ability of our system in handling variation of object scales and detecting multiple action classes.

Architecture	Precision	Recall	F1-Score
I3D [3]	0.36	0.31	0.33
P3D [24]	0.43	0.41	0.41
3D-ResNet [12]	0.46	**0.43**	0.44
R(2+1)D [32]	**0.50**	**0.43**	**0.45**

TABLE IV
ABLATION EXPERIMENTS FOR DIFFERENT CLASSIFICATION NETWORK ARCHITECTURES. PRECISION, RECALL, AND F1-SCORES ARE AVERAGED OVER ALL CLASSES ON VIRAT VALIDATION SET.

in the localization network, mainly due to the increase in number of correct detections. Although the regular dice loss improves localizations when compared to standard BCE, we find that the network does not correctly localize the very small activities. By using PDL during training, the network correctly localizes more of these activities which leads to an overall improvement in the AUDC score.

Classification Network: We experiment with multiple classification models to determine the best network architecture for our system. For a fair comparison, all models are initialized with pre-trained weights on the Kinetics [17] and are trained with the same settings. A comparison of their performance on the VIRAT validation set is shown in Table IV. We use average F1-Score as a metric for comparison and observe that R(2+1)D model [32] outperforms the other models.

TMAS System TMAS is the final step in our system and is crucial for it's success. To show the impact of this step in the overall performance, we compare per-class n-AuDC scores with and without the TMAS algorithm on our local evaluation set of the MEVA dataset. Please refer to Figure 6. With the post-processing step, we observe that the scores improve for the activity classes which occur for a longer temporal span such as 'person reads document'(20) and 'person texts on phone'(24). Please refer to Kitware page [3] for activity name and the corresponding indices.

[3]https://tinyurl.com/rum4ykm

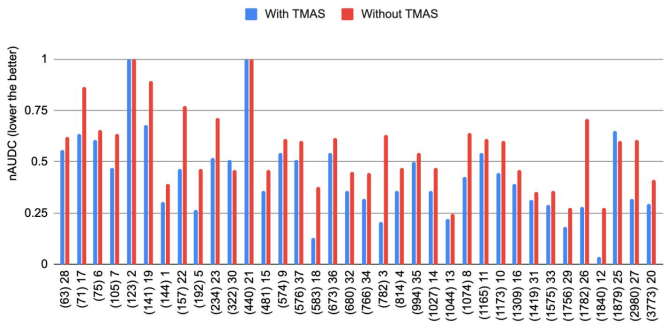

Fig. 6. Per-class n-AuDC scores from our system, with and without the TMAS. The results show the significance of the Tublet-Merge and Action-Split algorithms in the overall performance of our system. The labels on the x-axis are the class indices and the numbers in the brackets indicate average length of the activity in frames. Please refer to Kitware annotations page [3] for the activity names.

D. Qualitative Analysis

We present some qualitative results of our system in Figure 5. Our system performs well on different viewpoints, action scales, and action types. We are able to detect activities involving multiple actors, as well as activities involving interactions between a person and a vehicle. Since we do not rely on frame-based object detection, we produce fewer detections and avoid objects which are not involved in an activity - this results in a drastic reduction in computing power used to classify non-actions, like stationary vehicles.

E. Online action detection:

Online action detection is different from traditional action detection as its goal is to detect an action as it occurs. Our proposed system is an online action detection system as it can process a stream of input frames: it perform localization, classification, and temporal segmentation of activities with little or no delay. Other systems, such as [10] and [20] are restricted to offline detection as they rely on object detection

for every frame in the video, requiring access to future frames to generate tube proposals. While [9] improves computation time, it relies on optical flow computation and produces many proposals, causing trade-off in system performance. This is a major advantage of our system as it can be readily used in real-world surveillance applications.

VI. Conclusion

In this work, we propose Gabriella, a real-time online system to detect activities in untrimmed surveillance videos. The proposed system consists of three main components which includes tubelet extraction, classification, and online tubelet merging. The proposed approach processes short video clips independently which helps in real-time online processing. The efficient merging of tubelets using TMAS algorithm makes the action detections robust to varying length activities. In contrast to existing approaches, it does not require frame level object detection or optical flow extraction which are computationally expensive and need externally trained models. The proposed method provides state-of-the-art results on the VIRAT and MEVA datasets with a processing speed of over 100 fps.

References

[1] Kitware inc, the multiview extended video with activities (meva) dataset.

[2] Trecvid 2019 actev: Activities in extended video.

[3] J. Carreira and A. 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.

[4] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[5] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. 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.

[6] K. Duarte, Y. Rawat, and M. Shah. Videocapsulenet: A simplified network for action detection. In *Advances in Neural Information Processing Systems*, pages 7610–7619, 2018.

[7] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition, 2018.

[8] C. Fiorio and J. Gustedt. Two Linear Time Union-Find Strategies for Image Processing. *Theoretical Computer Science*, 154(2):165–181, 1996.

[9] J. Gleason, C. D. Castillo, and R. Chellappa. Real-time detection of activities in untrimmed videos. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision Workshops*, pages 117–125, 2020.

[10] J. Gleason, R. Ranjan, S. Schwarcz, C. Castillo, J.-C. Chen, and R. Chellappa. A proposal-based solution to spatio-temporal action detection in untrimmed videos. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 141–150. IEEE, 2019.

[11] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6047–6056, 2018.

[12] K. Hara, H. Kataoka, and Y. 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)*, pages 6546–6555, 2018.

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

[14] H. Idrees, A. R. Zamir, Y.-G. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah. The thumos challenge on action recognition for videos in the wild. *Computer Vision and Image Understanding*, 155:1–23, 2017.

[15] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid. Action tubelet detector for spatio-temporal action localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4405–4413, 2017.

[16] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1725–1732, Washington, DC, USA, 2014. IEEE Computer Society.

[17] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] A. Kirillov, R. Girshick, K. He, and P. Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019.

[20] W. Liu, G. Kang, P.-Y. Huang, X. Chang, Y. Qian, J. Liang, L. Gui, J. Wen, and P. Chen. Argus: Efficient activity detection system for extended video analysis. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision Workshops*, pages 126–133, 2020.

[21] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis, et al. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR 2011*, pages 3153–3160. IEEE, 2011.

[22] X. Peng and C. Schmid. Multi-region two-stream r-cnn for action detection. In *European conference on computer vision*, pages 744–759. Springer, 2016.

[23] A. J. Piergiovanni and M. S. Ryoo. Temporal gaussian mixture layer for videos. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 5152–5161, 2019.

[24] Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*, pages 5533–5541, 2017.

[25] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[26] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[27] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015.

[28] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 568–576. Curran Associates, Inc., 2014.

[29] G. Singh, S. Saha, M. Sapienza, P. H. Torr, and F. Cuzzolin. Online real-time multiple spatiotemporal action localisation and prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3637–3646, 2017.

[30] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 240–248. Springer, 2017.

[31] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[32] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[33] K. Wu, E. Otoo, and A. Shoshani. Optimizing connected component labeling algorithms. In J. M. Fitzpatrick and J. M. Reinhardt, editors, *Medical Imaging 2005: Image Processing*, volume 5747, pages 1965 – 1976. International Society for Optics and Photonics, SPIE, 2005.

[34] Z. Yang, J. Gao, and R. Nevatia. Spatio-temporal action detection with cascade proposal and location anticipation. *arXiv preprint arXiv:1708.00042*, 2017.

[35] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision*, 2017.