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

Abstract

Activity detection in surveillance videos is a challenging problem due to multiple factors such as large field of view, presence of multiple activities, varying scales and viewpoints, and its untrimmed nature. The requirement of processing the surveillance videos in real-time makes this more challenging. In this work, we propose a real-time online system to perform activity detection on untrimmed surveillance videos. The proposed system consists of three stages: first we detect tubelets with activities, then classify them, and finally merge them to generate spatio-temporal activity detections. We propose a localization network which takes a video clip as input and makes use of feature pyramid, multi-layer loss, and atrous convolutions to address the issue of multiple scales and detect small activities in terms of tubelets. The online processing of videos at a clip level drastically reduces the computation time in detecting activities. The detected tubelets are assigned activity class scores and merged together using our proposed Tubelet-Merge Action-Split (TMAS) algorithm to form action tubes. The TMAS algorithm efficiently connects the tubelets in an online fashion to generate spatio-temporal detections which are robust against varying length activities. We perform our experiments on the DIVA (Deep Intermodal Video Analytics) dataset and demonstrate the effectiveness of the proposed approach in terms of speed ($\sim$100 fps) and performance with state-of-the-art results. The code and models will be made publicly available.

1 Introduction

Deep convolutional neural networks have achieved impressive action classification results in recent years [25, 2, 26]. Similar advancements have been made for the tasks of action detection in trimmed videos [12, 24, 5] and temporal action localization in untrimmed videos [29, 19]. However, these improvements have not been transferred to spatio-temporal action detection in untrimmed videos; current computer vision systems have yet to achieve high performance on this difficult task.

Action detection in untrimmed security videos poses multiple challenges. Surveillance videos comprise of multiple viewpoints and contain several actors performing multiple actions concurrently. These actors have varying scales and tend to be extremely small relative to the video frame, which makes detection of small activities extremely challenging. These challenges make it difficult to extend existing methods to detect actions in the untrimmed security videos found in the DIVA (Deep Intermodal Video Analytics) dataset [17]. Current methods are trained and evaluated on datasets which contain some, but not all of these challenges. For example, THUMOS'14 [11] is comprised of untrimmed videos, but each video contains only one or two actors performing the same action. The AVA dataset [8] contains multiple actors and actions, but each video is trimmed. Figure 1 shows sample frame from the DIVA dataset and compares them with frames from other action detection datasets.

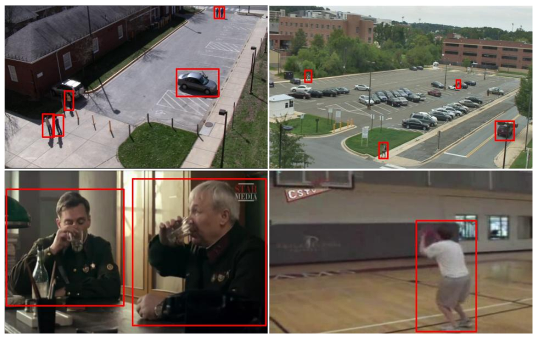

Figure 1: **Top**: Two Sample frames from different scenes of the DIVA dataset showing variation in perspective, scale and field-of-view. **Bottom**: Sample frames from the AVA dataset [8] (left) and from the THUMOS'14 dataset [11] (right). The DIVA dataset contains a greater number of concurrent actions as well as a greater variety of action scales (both spatially and temporally).

In this work, we focus on untrimmed surveillance videos and propose an online real-time system for spatio-temporal action detection. 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 pipeline processes videos in an online fashion: it localizes and classifies short action tubelets. Our Tubelet-Merge Action-Split (TMAS) algorithm, merges these tubelets into action-agnostic tubes of varying length and splits the tubes into a set of final spatio-temporal action predictions. By classifying short tubelets and merging them into action tubes, our system is able to detect both atomic and repetitive actions.

Our action localization module proposes pixel-level action localizations for a short video clip. This allows our system to generate action tubelet proposals without the use of frame-based object detectors. Frame-based object detection systems [7] 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. Due to our localization network and the overall system's online processing of videos, inference is performed at $\sim$100 frames per second, greatly exceeding the speed of contemporary action detection systems.

Our contributions are as follows:

- We propose a real-time online system that performs spatio-temporal action detection in untrimmed surveillance videos at $\sim$100 frames per second.
- We propose a novel action localization network to detect activity agnostic tubelets 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 spatio-temporal detections which are consistent across time as well as robust against varying length activities.

We evaluate the proposed approach on the DIVA (Deep Intermodal Video Analytics) dataset and obtain state-of-the-art results in terms of both speed and performance.

2 Literature Review

Convolutional Neural Networks (CNN) have been studied for video analysis and applied successfully for the action recognition problem [25, 2]. Earlier approaches fuse 2D frame features to extract temporal information[13], while recent works mostly apply 3D convolution to extract spatio-temporal features simultaneously [25, 2, 6, 27]. The work in [6, 23] use two stream network architectures to further exploit temporal dependencies.

Most action classifiers expect short trimmed videos but this is unrealistic for action recognition in real world surveillance videos. Predicting the temporal extents of actions is necessary for reliable recognition. In [19], a new layer is proposed to temporally localize activities in videos of MultiThumos dataset [29]. They represent actions as combinations of Gaussian distributions, which are predicted by their temporal convolution layer. Another major property of real world videos is that multiple actions can occur within the same scene. Most works on spatial action detection rely on a region proposal network [21] to detect multiple objects in each frame and combine them temporally to generate action tubelets [28, 18]. However, this approach becomes computationally inefficient as the number of proposals grows larger, making it unsuitable for real time methods. In [10], a 3D CNN network efficiently predicts frame-wise background-foreground segmentation map and extrapolates the action tubes. Duarte et al.[5] proposes a capsule based action detection network which segments and recognize actions jointly.

Spatio-temporal action detection in untrimmed videos requires more complex systems than aforementioned approaches. In [7], a frame level object detection and optical flow based model is proposed to solve spatio-temporal action detection for DIVA [17] dataset. They apply hierarchical clustering on all detected object regions in a video to obtain tube proposals and use optical flow to perform action classification. This approach is computationally expensive and not suitable for online processing. Our framework uses a 3D CNN network for spatio-temporal action segmentation which produces temporally consistent predictions and a fewer number of proposals. Additionally, our system processes videos in an online fashion without using computationally expensive methods (region proposal network, optical flow) and achieves better performance in real-time.

3 Methodology

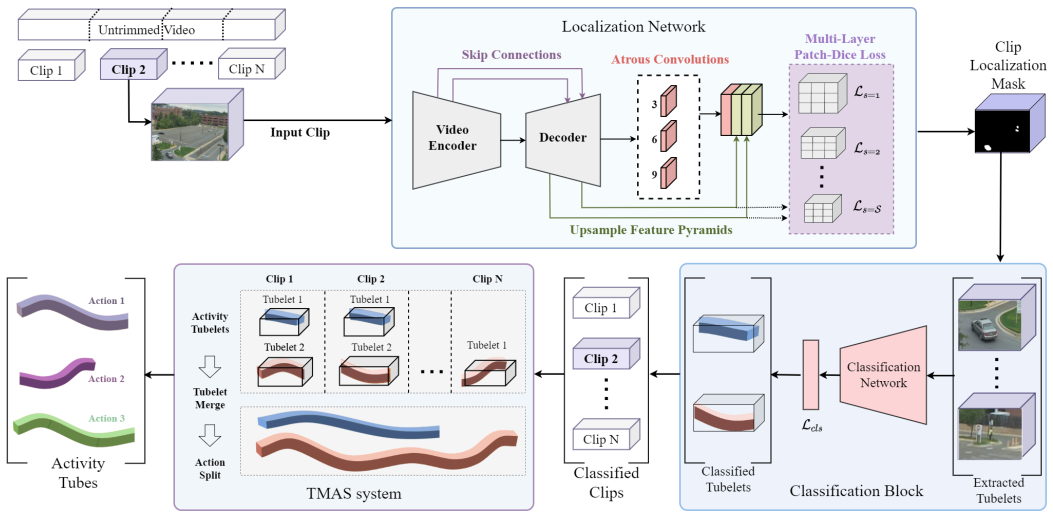

Figure 2: Overview of the proposed system for action detection in untrimmed videos. An untrimmed video is processed clip by clip and fed into the localization network, producing localization masks. Tubelet extraction produces tubelets for each clip, which are then classified and passed to our TMAS system.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

3.1 Overview

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.

3.2 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 2, 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, 16] and video segmentation [5, 10], 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.

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 [16, 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, 16] 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.

3.3 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 [9] initialized with pre-trained weights on Kinetics [14] 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.

3.4 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

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 $\{t : \mathbf{M}[(t_c,t)] > 0\}$.
 1: procedure TUBELET-MERGE(S)
 2:     $T_{prev}, T_{done} \leftarrow \{\}$                    ▷ Initialize candidate and final tubes
 3:     $\mathbf{M} \leftarrow initializehashtable$
 4:     while $t_c in \mathbf{S}$ do                     ▷ Continue until the stream of tubelets ends
 5:         for all $t_p in T_{prev}$ do
 6:             if $Inter_{temp}(t_p, t_c) > 0$ then
 7:                 $\mathbf{M}[(t_p, t_c)] \leftarrow IoU(t_p, t_c)$
 8:             else
 9:                 CHECKEND($t_p$, $T_{prev}$, $\mathbf{M}$)
10:         append $t_c$ to $T_{prev}$                   ▷ Tubelet becomes a candidate tube
11:     while $T_{prev} isnotempty$ do                 ▷ Deals with remaining candidates
12:         $t_p \leftarrow T_{prev}[0]$
13:         CHECKEND($t_p$, $T_{prev}$, $\mathbf{M}$)
14:     return $T_{done}$
 1: function CHECKEND($t_p, T_{prev}, \mathbf{M}$)
 2:     if $|\mathbf{M}[(t_p,*)]| == 0$ then
 3:         MOVE($t_p, T_{prev}, T_{done}$)              ▷ Moves $t_p$ from $T_{prev}$ to $T_{done}$
 4:     else if $|\mathbf{M}[(t_p,*)]| == 1$ then
 5:         $t_i \leftarrow \max_{t_i} \mathbf{M}[(t_p, t_i)]$
 6:         if $|\mathbf{M}[(*,t_i)]| == 1$ then
 7:             MERGE($t_p, t_i, T_{prev}, \mathbf{M}$)
 8:         else
 9:             MOVE($t_p, T_{prev}, T_{done}$)
10:     else
11:         $t_i \leftarrow \max_{t_i} \mathbf{M}[(t_p, t_i)]$
12:         MERGE($t_p, t_i, T_{prev}, \mathbf{M}$)
 1: function MERGE($t_1, t_2, T_{prev}, \mathbf{M}$)          ▷ Merges two candidate tubes
 2:     $t_1 \leftarrow (f_1^1, f_2^2, \{\mathbf{b}^1, \mathbf{b}^2\}, \{\mathbf{a}^1, \mathbf{a}^2\})$      ▷ {} is concatenation
 3:     $removet_2 fromT_{prev}$
 4:     $\mathbf{M}[t_1, t_i] \leftarrow \mathbf{M}[t_2, t_i]$        ▷ $Doneforallt_i where\mathbf{M}[t_2, t_i] \geq 0$
```

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.

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.

4 Experimental Setup

4.1 DIVA Dataset

The DIVA dataset is a large-scale spatio-temporal action detection dataset with untrimmed surveillance videos. It consists of videos from the VIRAT [17] dataset with added annotations for action

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: $a_c^i[f]$ and $t_i[f]$ contain the action prediction scores and tube information at frame $f$, respectively.
 1: procedure ACTION-SPLIT($T$)
 2:     $A_G \leftarrow \{\}$                                  ▷ Initializes the action-specific tubes
 3:     for all $t_i inT$ do
 4:         $t_{smooth} \leftarrow SMOOTH(t_i)$
 5:         for all $cin1 : C$ do                              ▷ Loop through each action class
 6:             $a_L \leftarrow EXTRACT(t_{smooth}, c)$
 7:             $append a_L to A_G$
 8:     return $A_G$
 1: function SMOOTH($t_i$)
 2:     for all $f in f_1^i : f_2^i$ do
 3:         $a_c^i\,[f] \leftarrow \frac{1}{2\tau+1} \sum_{k=-\tau}^{\tau} a_c^i\,[f+k]$
 4:     return $t_i$
 1: function EXTRACT($t_i, c$)                                 ▷ Extracts tubes of a specific class
 2:     $A_L, a_l \leftarrow \{\}$                  ▷ Initialize extracted action tubes and a placeholder
 3:     $count \leftarrow 0$
 4:     for all $f in f_1^i : f_2^i$ do
 5:         if $a_c^i[f] > \alpha$ then                        ▷ Continue current action tube
 6:             $append t_i[f] to a_l$
 7:             $count \leftarrow 0$
 8:         else
 9:             $count \leftarrow count + 1$
10:         if $count > \beta$ then                            ▷ Current action tube is finished
11:             $append a_l to A_L$
12:             $a_l \leftarrow \{\}, count \leftarrow 0$
13:     $remove tubes shorter than \gamma from A_L$
14:     return $A_L$
```

detection. There are 64 videos (2.47 hours) in training and 54 videos (1.93 hours) in the validation set, with annotations for all the activities. There are 246 videos (10.11 hours) in the held out test set for which the annotations are not made public. There are 40 different types of activities in the dataset which can be broadly categorized into activities involving only people, activities involving only vehicles, and activities involving both people and vehicles. Of these 40 classes only 18 classes are used for evaluation of the system: *activity_carrying*, *vehicle_turning_right*, *vehicle_turning_left*, *Closing*, *Opening*, *Exiting*, *Entering*, *Talking*, *Transport_HeavyCarry*, *Unloading*, *Pull*, *Loading*, *Open_Trunk*, *Closing_Trunk*, *Riding*, *specialized_texting_phone*, *specialized_talking_phone*, and *vehicle_u_turn*. Remaining classes are considered *hard_negative*, or background. Videos in this dataset are captured from surveillance cameras mounted on buildings, mostly overlooking parking lots or streets, where the activities take place. All videos are high resolution, either 1920×1080 or 1280×720, with variation in scale, orientation of objects, and camera viewpoints.

4.2 Training Details

Network Training DIVA videos are high resolution and it is not feasible to train a localization model at that scale. Therefore, we rescale the videos to a lower resolution of 800×448, while maintaining the aspect ratio. A stack of 16 frames is fed to the localization network to obtain the binary segmentation masks. The network is trained using SGD [22], with a learning rate of $1e-3$ for about 100000 iterations. 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 x 112. We use the ADAM optimizer [15] with a learning rate of 1e-4 to train the classifier for 75000 iterations on a single NVIDIA GeForce Titan X GPU.

Data Pre-processing For the data used in training the localization network, we apply random frame jitter and crop to simulate shaking cameras and changes in scale. These augmentations add variation to the training data, and greatly reduces overfitting.

Model	**IoU**
Baseline	0.540
Baseline+FPN	0.572
Baseline+FPN+MLL	0.613
Baseline+FPN+MLL+Atrous	0.628

Table 1: Ablation experiments to study the effect of different components of the localization network.

The spatial resolution of activity tubelets can be of any size, but the input to the classification model is fixed. To address this issue we crop a square region encompassing the activity within the tubelet and rescale it to the size expected by the classification model. This approach helps us in maintaining the aspect ratio of the objects in the input clips. In temporal domain, the activity tubelets can be of arbitrary length, so we extract multiple clips by applying a sliding window over the length of the entire tubelet. To train the classification model, clips are generated using both the ground-truth annotations as well as the tubelets from the localization network. This increases the amount of training data seen by the classifier, allowing for improved performance on the test set. Training clips from the localization model that do not contain any of the 18 activity classes used for evaluation are labelled as *hard_negative*.

One of the challenging issues with the DIVA dataset is data imbalance. Refer to Figure 3 for distribution of per-class action instances/frames from DIVA ground-truth training set. Since clips are extracted using a sliding window, the class imbalance is further exasperated for longer, repetitive activities like *activity_carrying* and *Riding* as well as shorter, atomic activities like *Loading*, *Unloading*, *Entering*, and *Exiting*. To overcome this issue, we use augmentation techniques such as multi-scale cropping, frame reversal, spatial-jittering, and flipping of frames along the vertical axis. In multi-scale cropping we randomly crop a region around the center of a clip at different scales to simulate zoom-in and zoom-out effect to have actors/objects at different scales. We reverse the order of frames in a clip to generate new clips for specific 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. With spatial-jittering we simulate camera shake due to wind, which occasionally occurs in test videos.

5 Results

In this section we present evaluation results of individual components in the proposed architecture and discuss the overall system performance.

5.1 Localization

We run several ablations to evaluate the effect of the different components - feature pyramid, atrous convolution, and multi-layer loss - of our localization network. Table 1 shows the performance of different variations of our action localization network on the DIVA validation set in terms of IoU scores. The baseline network is an I3D based encoder-decoder architecture with skip connections. The addition of these components improves our IoU scores considerably.

5.2 Classification

We experiment with multiple classification models to determine the optimal network architecture for our system. A comparison of their performance on the validation set is shown in Table 2. We use average F1-Score as a metric to compare the different models. For a fair comparison, all models are initialized with pre-trained weights on the Kinetics dataset [14] and are trained with the same settings. We observe that the 3D-ResNet based model, WideResNet-50 [9], outperforms the other architectures.

We also analyze the class level performance of the classification model on the validation set. The results using our final classification model are shown in Figure 3. We can observe that our network performs better on longer activity classes such as *Riding* and *activity_carrying* with higher number of training samples when compared to shorter activity classes like *Unloading* and *Loading* which have fewer samples. Activity classes *specialized_texting_phone*, *specialized_talking_phone* have the

Architecture	Precision	Recall	F1-Score
I3D [2]	0.36	0.31	0.33
R(2+1)D [26]	0.43	0.39	0.40
P3D [20]	0.43	0.41	0.41
3D-ResNet [9]	**0.46**	**0.43**	**0.44**

Table 2: Average Precision, Recall and F1-score on DIVA validation set with different network architectures for classification. We use pre-trained weights on Kinetics dataset [14] to initialize all these models before training.

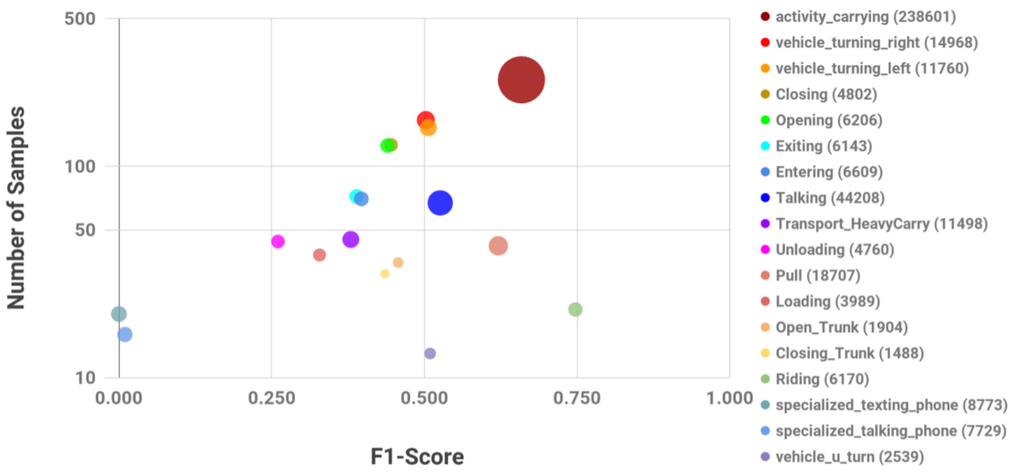

Figure 3: Number of samples per class and F1-Score achieved by the trained classifier on the DIVA validation set. The size of each bubble indicate the number of frames for an activity class. The exact numbers are indicated next to the activity name in the legend on the right.

lowest F1-Score as these have considerably fewer samples and appear very similar to *activity_standing* which is a background activity. Qualitative results of our system are included in Figure 4.

5.3 System Evaluation

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 both temporally, for the action detection (AD) task, and spatio-temporally, for the action-object detection (AOD) 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 different evaluation metrics as well as evaluation code, we refer to TRECVID 2020 [1].

Comparison 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 3, which is reported on the TRECVID ActEV 2020 Evaluation leaderboard.

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

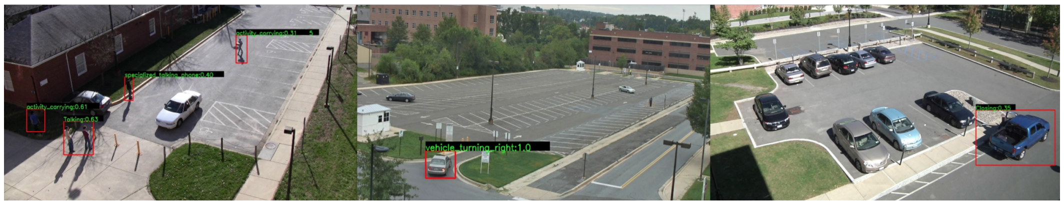

Figure 4: Qualitative results of our system on some sample validation videos. This demonstrates the ability of our system to handle varying scenes, object scales, and action types.

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

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] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition, 2018.

[7] Joshua Gleason, Rajeev Ranjan, Steven Schwarcz, Carlos Castillo, Jun-Cheng Chen, and Rama Chellappa. A proposal-based solution to spatio-temporal action detection in untrimmed videos. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019.

[8] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul 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.

[9] 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.

[10] 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.

[11] Haroon Idrees, Amir R Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The thumos challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 155:1–23, 2017.

[12] Vicky Kalogeiton, Philippe Weinzaepfel, Vittorio Ferrari, and Cordelia Schmid. Action tubelet detector for spatio-temporal action localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4405–4413, 2017.

[13] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li 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.

[14] 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.

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] 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.

[17] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, JK Aggarwal, Hyungtae Lee, Larry Davis, et al. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR 2011*, pages 3153–3160. IEEE, 2011.

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

[19] A. J. Piergiovanni and Michael 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.

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

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[22] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[23] Karen Simonyan and Andrew 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.

[24] Gurkirt Singh, Suman Saha, Michael Sapienza, Philip HS Torr, and Fabio Cuzzolin. Online real-time multiple spatiotemporal action localisation and prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3637–3646, 2017.

[25] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[26] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar 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.

[27] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[28] Zhenheng Yang, Jiyang Gao, and Ram Nevatia. Spatio-temporal action detection with cascade proposal and location anticipation. *arXiv preprint arXiv:1708.00042*, 2017.

[29] 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*, 2017.