Activity Detection in Untrimmed Videos Using Chunk-based Classifiers

Joshua Gleason[*], Steven Schwarcz[*], Rajeev Ranjan, Carlos D. Castillo, Jun-Cheng Chen, Rama Chellappa
University of Maryland, College Park
gleason@umiacs.umd.edu, schwarcz@umiacs.umd.edu
[*]These authors contributed equally to this work

Abstract

Activity detection in untrimmed videos - the process of detecting and localizing human activities in potentially long videos - is a challenging problem in computer vision. We propose an algorithm which is based on the proposition that despite the differences between activity classification and detection, a strong classifier can still be used to achieve state-of-the-art performance in detection by breaking the video into multiple overlapping chunks and classifying each individually. We further introduce two new auxiliary tasks which we call chunk inclusion and localization. The outputs of these tasks, when carefully applied, can be used to dramatically improve performance. We call our method Chunk Aggregation. It is straight-forward to implement and use, and is agnostic to the backbone activity classification architecture used. We also demonstrate the effectiveness of chunk association by presenting results and a series of ablation experiments on the THUMOS'14 and ActEV datasets.

1. Introduction

Activity classification in a long standing problem within computer vision. Great progress has been made in activity detection in the trimmed case [7, 11], where the activity to be classified can reliably be expected to take up most of the video. More challenging, however, is the untrimmed case, wherein one or more activities may occur anywhere in the video, often nothing of interest occurring in between the activities. This setting, where activities must be detected in addition to classified, generally calls for additional or alternate techniques in order to achieve reasonable results.

With the advent of deep learning, many modern approaches to activity detection and localization have used deep neural architectures to achieve state of the art performance. Many such approaches draw on very sophisticated architectures, making use of LSTMs [53, 54] or adaptations of object detection architectures such as Faster R-CNN [9, 50] to spatio-temporal data.

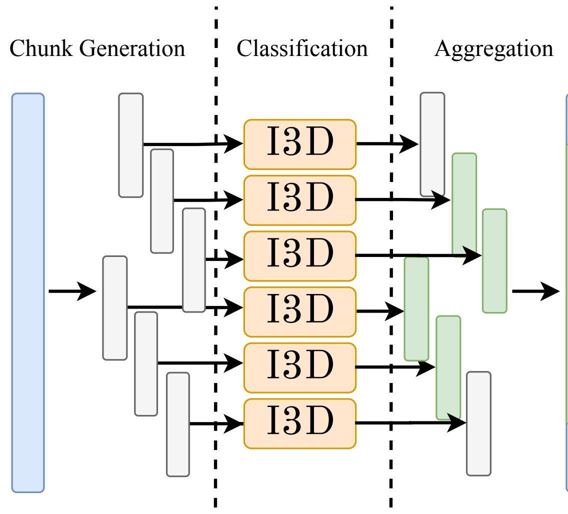

Figure 1. An overview of the system we present. Our procedure has three stages: we first generate a set of chunks by breaking up the video into overlapping pieces, then we classify each chunk individually. In the last stage of our system, we aggregate the classified chunks in order to create our final activity detections.

We propose a comparatively simpler approach, leveraging the strong progress that has been made in activity classification for trimmed videos. Our main contribution is to show that a strong, carefully-trained activity classifier can achieve state-of-the-performance in activity localization by breaking the activity into many small overlapping chunks and classifying each chunk individually. Further, by learning two auxiliary tasks and applying the learned output in a clever way, it is possible to achieve even more significant performance improvements beyond the state of the art.

The resulting technique, which we call Chunk Association, is straight-forward to implement and use, and is agnostic to the chosen backbone activity classification architecture. We therefore believe that it presents a very general technique that can continue to be useful even as the state-of-the-art in trimmed activity or video classification advances, with the backbone architecture replaced as newer, better classification methods become available.

Our method also presents some very simple trade-offs depending on the use case and hardware available to run our algorithm. It can, for instance, be tuned to run faster or slower by varying the input modality used, while still maintaining a high level of performance.

We demonstrate the effectiveness of chunk association by presenting results on the THUMOS'14 [25] dataset for temporal activity localization. We also show that our method can be extended to perform spatial localization as well by performing experiments on the ActEV dataset for sparse spatio-temporal localization [36]. Finally, we present the results of a series of ablation experiments to better understand the components of our algorithm and validate the design choices that we have made.

2. Related Work

In recent years, a significant amount of research has been done on the problem of activity recognition. In this section we distinguish between two types of activity recognition: 1) activity classification, which refers to the problem of classifying videos containing only one activity, where each video is trimmed to the beginning and end of the activity, and 2) activity detection which refers to the problem of determining where, if anywhere, activities are occurring in a video. Despite significant progress made in activity classification, robust systems capable of activity detection for general use have remained elusive.

Activity Classification Following the success of deep CNNs in image classification by AlexNet [31], a number of early attempts were made to adapt CNN-based image classification methods to activity classification [30, 42, 56, 35]. One notable work is the two-stream CNN framework [42] which was able to obtain state-of-the-art performance by combining optical-flow and RGB using two parallel 2D CNNs. Following the two-stream work, a number of activity classification techniques were proposed which used similar strategies [12, 13, 14, 47, 48, 44].

The I3D model was introduced along with the Kinetics dataset [7]. I3D is a two-stream architecture which uses 3D convolutions. While not unique for its use of 3D convolutions [24, 44, 43], I3D was able to significantly outperform other activity classification systems like C3D [43] by taking advantage of the carefully curated, large-scale Kinetics dataset [19] and building a system that was well suited for transfer-learning on other datasets.

Activity Detection Activity detection approaches can broadly be categorized into proposal-based or end-to-end systems. In proposal-based activity detection, a collection of subsets of a video are generated to be considered as potential activities. Proposals can be viewed as high-recall, low-precision activity detections. They may be either dependent or independent of the activity class. Once the proposals are collected, classification is applied to distinguish between true and false positives, and, in the case of class-independent proposals, to determine the specific activity.

In 2014, many state of the art systems for activity detection in unconstrained videos utilized Fisher vector representation with dense trajectories evaluated over dense sliding windows [45, 29, 37, 46]. Caba et al. [6] presented an efficient proposal-based method using sparse dictionary learning. The trend of using sparse methods was relatively short-lived; however, as deep learning based approaches became more prevalent. Particularly relevant to our method, Gao et al. [16], motivated by advancements in object detection, introduced the use of cascaded boundary regression for sliding window based proposals. This was found to perform much better than other contemporary methods and provides motivation for the temporal refinement component of our system. Gleason et al. [18] propose a temporal refinement I3D system which utilizes off the shelf classification with additional regression based temporal refinement. Other approaches inspired by object-detection methods are Dai et al. [9] and Xu et al. [50] which both use Faster R-CNN[38]-based systems adapted for temporal activity detection.

Research into proposal-based methods often focus on improving proposal generation and using off-the-shelf CNN classifiers [10, 5, 17]. Other works focus more on alterations to the classification architecture for use in activity classification and detection [40].

To get an idea of where our algorithm fits into this discussion, in this work we present an algorithm which uses short sliding window-based proposals, and focuses primarily on modifications to off-the-shelf CNN classifiers, along with improvements to proposal aggregation. By proposal aggregation, we are referring to a post-processing step which is used to combine multiple short proposals into a single activity which is discussed in detail in Section 3.

An alternative to proposal-based activity detection is all-in-one activity detection, which simultaneously classifies and localizes activities. Richard et al. [39] introduced a probabilistic model for temporal activity detection that jointly models segmentation and classification. Yuan et al. [55] introduce a method for temporal localization by aggregating frame-wise features using an efficient algorithm for computing structured maximal sums. Hou et al. [21] proposed the tube convolutional neural network (T-CNN) which generates tube proposals from video-clips and performs activity classification and localization using an end-to-end 3D-CNN. Kalogeiton et al. [28] extract convolutional features from each frame of a video, and stack them to learn spatial locations and activity scores.

3. Method

Our method is composed of three stages: chunk generation, classification, and aggregation. In the chunk generation stage, the video is broken down into overlapping chunks of 64 frames each. These chunks are then fed into a state-of-the-art video classification algorithm and classified according to their activity class. Critically, we also equip our video classifier with two auxiliary tasks: "temporal refinement" and a flag that determines if the activity end points fall within the chunk, which we call "chunk inclusion". These two tasks are easy to learn, but their output plays a critical role in the aggregation stage, where chunks are recombined into activity detections.

3.1. Chunk Generation

For each video, we construct a series of chunks X_i consisting of $n = 64$ frames each. Chunk generation begins by designating the first 64 frames of a given video to be the first chunk, and then, moving in strides of $s = 16$ frames at a times, progressively creating chunks out of every set of 64 consecutive frames. Thus, each frame of the video is included in at most 4 different chunks, and neighboring chunks overlap by 48 frames. Algorithm 1 illustrates this procedure.

Algorithm 1 Chunk generation for a single video.

```
1: $t \leftarrow 0$
2: $s \leftarrow 16$
3: $chunks = empty\_list()$
4: while $t + 64 \leq video.length$ do
5:     $new\_chunk \leftarrow$ Chunk$(t, t + 64)$
6:     $chunks.add(new\_chunk)$
7:     $t \leftarrow t + s$
8: return $chunks$
```

3.2. Classification

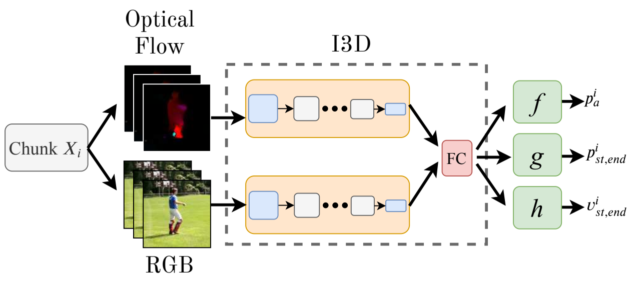

Figure 2. An illustration of chunk classification stage of our algorithm. We extract 64 consecutive 224 × 224 RGB and optical flow frames per chunk X_i. These are fed into I3D, and three outputs are produced: activity class scores p^i_a, chunk inclusion scores (p^i_{st}, p^i_{end}), and temporal localization scores (v^i_{st}, v^i_{end})

The classification stage of our algorithm is built on existing video-classification methods. Since the video has already been divided into small, manageable chunks, each chunk can be treated as an individual video and subsequently classified.

At the time of this writing, the most powerful open source video classification algorithm is the Inflated 3D Convolution (I3D) architecture [7], we take the 64 frames that compose it and compute optical flow for each one using the TV-L1 optical flow algorithm [57]. We then randomly crop the 256 × 256 frames into 224 × 224 and feed them into I3D in a 2 stream configuration, similar to the original authors in [7]. We also perform random horizontal flips for additional augmentation. However, unlike [7], we train I3D on three different tasks using 3 different branches: 1) an activity classification branch f, 2) a chunk inclusion branch g, and 3) a localization branch h. See Figure 2 for an outline of our network architecture. In addition, we also make a minor modification to the I3D architecture, and instead of averaging the logits from the two streams to get final results, we concatenate the features in the penultimate fully connected layer and produce a single set of logits.

Activity classification is the standard task for which I3D was designed. We designate a chunk X_i as belonging to an activity class a if at least 55% percent of the chunk contains an instance of that activity class, with ties going to the activity that occurs on the most frames. We thus train our classifier using standard cross-entropy loss:

$$\mathcal{L}^i_{cls} = \sum_{a=0}^{n_{cls}} -y^i_a \cdot \log(p_a), \tag{1}$$

where n_{cls} is the number of classes, and y^i_a is an indicator that is 1 when chunk X_i is assigned class a and 0 otherwise. p^i_a is the softmax output of video classification branch f which corresponds to activity class a, with $a = 0$ designating that no activity has occurred.

The chunk-inclusion task on a chunk X_i simply performs two classification tasks: one to determine if the activity began within X_i, and another to determine if the activity ended within X_i. The inclusion ground truth labels, which we denote y^i_{st} and y^i_{end} for start and end inclusion respectively, are trivial to compute during chunk generation. We learn these with a cross entropy loss as well:

$$\mathcal{L}^i_{inc} = \frac{1}{2}\left(\text{bce}(p^i_{st}, y^i_{st})\right) + \left(\text{bce}(p^i_{end}, y^i_{end})\right) \tag{2}$$

where p^i_{st} and p^i_{end} correspond to the two separate sigmoid outputs of the chunk inclusion branch g. bce corresponds to the standard binary-cross entropy loss function.

The final task we learn is temporal refinement, which is performed similarly to [18]. Whenever an activity starts or ends within a chunk, we learn either of two temporal regression parameters r_{st} or r_{end}, which we compute in the ground truth as

$$(r^i_{st}, r^i_{end}) = \left(\frac{\hat{t}_{st} - t^i_c}{n'}, \frac{\hat{t}_{end} - t^i_c}{n'} \right). \tag{3}$$

where t^i_c is the center of chunk X_i, ($\hat{t}_{st}$,$\hat{t}_{end}$) are the start and end frames of the ground truth activity assigned to chunk i, and $n' = n/2$, where $n = 64$ is the chunk length. As an example, if the start and end of the chunk were perfectly aligned with the ground truth, we would have $(r^i_{st}, r^i_{end}) = (-1, 1)$.

We train these values using a smooth_{L1} loss [38]. The temporal localization branch h produces two regression outputs $v^i = (v^i_{st}, v^i_{end})$ according to

$$\begin{aligned} \mathcal{L}_{loc} = y^i_{st} \cdot \text{smooth}_{L1}(r_{st} - v^i_{st}) \\ + y^i_{end} \cdot \text{smooth}_{L1}(r_{end} - v^i_{end}). \end{aligned} \tag{4}$$

We note that the y^i_{st} and y^i_{end} terms prevent the learning of regression when an activity does not begin or end within the chunk.

Our final loss can thus be written:

$$\mathcal{L}_{full} = \mathcal{L}_{cls} + \lambda \mathbb{I}_{a \geq 1}(\mathcal{L}_{inc} + \mathcal{L}_{loc}), \tag{5}$$

where $\mathbb{I}_{a \geq 1}$ is an indicator function that is equal to 1 when the ground truth action has an index greater than 0, and is equal to 0 otherwise (*i.e.* is equal to 1 when the ground truth does not designate ”no activity”). λ is a weight parameter that we experimentally set to $\lambda = 0.25$.

3.3. Chunk Aggregation

Breaking the video into chunks and classifying the pieces is a straight-forward procedure, but after the chunks have been classified, it is not immediately obvious how to recombine them in a way that generates temporally accurate and consistently correct activity detections. As it turns out, however, it is still possible to achieve excellent performance simply by merging adjacent chunks of the same class.

Specifically, our algorithm starts by assigning all chunks X_i with assigned activity classes other than ”no activity” to their own detection d_i. If X_i and X_j are 2 adjacent chunks assigned to the same activity class, we merge d_i and d_j into a new chunk d'.

Inclusion Thesholding Simply merging all neighboring chunks in this manner would be sufficient to achieve activity detection performance that is comparable to the state-of-the-art (see Section 4.2.2); however, it is only when we make use of the learned chunk inclusion and localization parameters during aggregation that we achieve the best performance. Thus, we make use of the learned inclusion information at this stage by not connecting chunks when at least one of the chunks directly predicts that it should not have a neighbor. Specifically, we do not merge adjacent chunks X_i and X_j if X_j immediately precedes X_i and $p^i_{st} > T_{inc}$, which indicates that the algorithm has determined that the activity begins in X_i. Similarly, we do not merge X_i and X_j if X_i immediately precedes X_j and $p^i_{end} > T_{inc}$. In our experiments, we use the value $T_{inc} = 0.8$ and perform non-maxima suppression on the p^i_{st} and p^i_{end} before merging, wherein we set all p^i_{st} and p^i_{end} to 0 if they are not higher than their immediate neighbors. Except when prohibited by the inclusion values, we then merge all chunks to create our activity detections.

Temporal Refinement As a final measure, we take the earliest and latest chunks in each detection (which we denote X_i and X_j respectively) and apply the learned temporal refinement values v^i_{st} and v^j_{end} to X_i and X_j by reversing the computation in Equation 3. More specifically, our new start time is $t^i_c + n'v^i_{st}$ and our new end time is $t^j_c + n'v^j_{end}$.

3.4. Spatio-Temporal Activity Detection

Our algorithm can also be extended to work in the context of spatio-temporal activity detection, in the situation where activities are sparse both temporally and spatially. In this context, an action may take up only a small portion of the screen as shown in Figure 5, and many actions may be happening at the same time in disparate parts of the frame. In this context, merely breaking the video into temporal chunks would not be sufficient to perform proper detection, since we would not be performing spatial localization.

Thus, to extend our method to spatio-temporal localization, we employ an additional proposal-generation stage similar to [18]. We begin by performing 2D object detection on a per-frame basis to identify the people and vehicles within the video, in this case using Mask-RCNN [20] . We represent these detections as a 3-dimensional feature (x, y, t) where x and y represent the center of the detection's bounding box in pixel coordinates and t is the frame number. We then use Divisive Hierarchical Clustering [34, 26] to generate a linkage tree of detections from these 3-dimensional features. We break this tree at various levels to produce clusters of various sizes. For complete details on cluster generation see [18].

We treat each of these clusters the same as we treat entire videos in the temporal localization task, breaking them into chunks and classifying the chunks individually. When we define the bounds for a chunk, we use the bounding box of all the 2D detections that fall within the chunk. When aggregating, we merge chunks that are within 16 frames of each other and have spatial IoU of at least 0.1.

4. Experiments

In this section, we present a series of experiments designed to help understand the impact of various components of our system. The majority of experiments presented in this section were performed on the THUMOS'14 action detection dataset [25]. To show that this approach generalizes to other scenarios we provide additional evaluation results on the ActEV [36] dataset for spatio-temporal activity detection.

4.1. Training

We train the backbone I3D network on THUMOS'14 using SGD with learning rate 0.01, momentum 0.9, and weight decay 0.0005. We train on 8 Nvidia 1080Ti GPUs for a total of 2 epochs, lowering the learning rate to 0.001 in the second epoch. We use a batch size of 24 for two-stream training experiments (RGB and optical flow) and 48 for single stream experiments (optical flow or RGB individually). We initialize I3D with weights pre-trained on the Kinetics dataset [7].

For most of the time, no activities are happening and if we trained on all the chunks of our system, we would inevitably bias heavily towards the "no activity" class. Thus, when creating chunks for training, we carefully control the number of chunks we expose the network to, enforcing that only 10% of the samples seen by the network are negatives.

4.2. THUMOS'14

Figure 3. Sample frames from videos within the THUMOS'14 dataset.

4.2.1 Dataset

THUMOS'14 is a temporal action detection dataset which consists of 2765 trimmed training videos, 200 untrimmed validation videos, and 213 untrimmed test videos. Since the training data is trimmed, we follow the common practice and perform the majority of our experiments by training on the validation set and evaluating on the test set [50, 18, 17, 15]. The majority of ablation experiments are performed using a single-stream configuration, using optical flow frames, except where otherwise mentioned.

The only exception to the aforementioned train/evaluate strategy is for experiments used to select the hyperparameters described in Section 3 (*e.g.* T_{inc} and s) . For these experiments, we instead use a 70/30 split of the THUMOS validation set to perform training and validation respectively, so as not to evaluate on the test split. This ensures that we don't overfit the hyperparameters of our method to the test data. In order to enforce that the 70/30 split of videos contains the appropriate number of instances of each class, we employ the simple strategy of repeatedly sampling random 70/30 splits until the number of training samples for each class is between 1.05 and 3.38 times the number of instances in validation.

Figure 3 shows sample frames from videos within the THUMOS'14 dataset. THUMOS performance is measured in the provided THUMOS'14 scoring tool using the mean Average Precision (mAP) metric, performed at several different temporal Intersection over Union (tIoU) thresholds.

Table 1 shows the final performance of our algorithm on THUMOS'14, as compared to other published algorithms. To achieve these final numbers, we use the I3D backbone in a two-stream configuration, taking both the RGB and optical flow frames as inputs.

tIoU	0.1	0.2	0.3	0.4	0.5	0.6	0.7
Karaman *et al.* [29]	4.6	3.4	2.4	1.4	0.9	-	-
Oneata *et al.* [37]	36.6	33.6	27.0	20.8	14.4	-	-
Wang *et al.* [46]	18.2	17.0	14.0	11.7	8.3	-	-
Caba *et al.* [6]	-	-	-	-	13.5	-	-
Richard *et al.* [39]	39.7	35.7	30.0	23.2	15.2	-	-
Shou *et al.* [41]	47.7	43.5	36.3	28.7	19.0	10.3	5.3
Yeung *et al.* [53]	48.9	44.0	36.0	26.4	17.1	-	-
Yuan *et al.* [54]	51.4	42.6	33.6	26.1	18.8	-	-
Escorcia *et al.* [10]	-	-	-	-	13.9	-	-
Buch *et al.* [5]	-	-	37.8	-	23.0	-	-
Shou *et al.* [40]	-	-	40.1	29.4	23.3	13.1	7.9
Yuan *et al.* [55]	51.0	45.2	36.5	27.8	17.8	-	-
Buch *et al.* [4]	-	-	45.7	-	29.2	-	9.6
Gao *et al.* [16]	60.1	56.7	50.1	41.3	31.0	19.1	9.9
Hou *et al.* [22]	51.3	-	43.7	-	22.0	-	-
Dai *et al.* [9]	-	-	-	33.3	25.6	15.9	9.0
Gao *et al.* [17]	54.0	50.9	44.1	34.9	25.6	-	-
Xu *et al.* [50]	54.5	51.5	44.8	35.6	28.9	-	-
Zhao *et al.* [59]	60.3	56.2	50.6	40.8	29.1	-	-
Huang *et al.* [23]	-	-	-	-	27.7	-	-
Yang *et al.* [51]	-	-	44.1	37.1	28.2	20.6	12.7
Chao *et al.* [8]	59.8	57.1	53.2	48.5	42.8	33.8	20.8
Alwassel *et al.* [2]	-	-	51.8	42.4	30.8	20.2	11.1
Gao *et al.* [15]	-	-	-	-	29.9	-	-
Lin *et al.* [32]	-	-	53.5	45.0	36.9	28.4	20.0
Gleason *et al.* [18]	52.1	51.4	49.7	46.1	37.4	26.2	15.2
Yang *et al.* [52]	-	-	51.8	41.5	32.1	22.9	14.7
Murtaza *et al.* [33]	-	-	54.9	47.2	41.5	37.5	**31.6**
Ours	**67.41**	**66.96**	**62.6**	**56.87**	**48.99**	**39.19**	27.82

Table 1. Comparison to other algorithms on the THUMOS'14 based on the mAP metric at various temporal IoUs. Missing entries indicate that results are not available. The best performance at each tIoU is indicated in bold.

Inclusion	Localization	0.1	0.5	0.7
No	No	59.09	32.19	16.08
Yes	No	61.53	37.04	21.03
No	Yes	60.45	35.9	19.25
Yes	Yes	**64.56**	**48.35**	**28.22**

Table 2. The effects of chunk inclusion and localization on performance at 3 different tIoU thresholds. All results are attained using optical flow as the only input modality. We also note that for this table, when inclusion or temporal localization are used, we also apply inclusion thresholding or temporal refinement respectively. Best results are in bold.

Inclusion	Localization	0.1	0.5	0.7
No	No	61.68	31.03	11.46
Yes	No	64.06	40.55	15.68
No	Yes	61.4	37.51	21.0
Yes	Yes	**64.56**	**48.35**	**28.22**

Table 3. The result of training the system using inclusion and localization loss but selectively ignoring those results during evaluation. This is to demonstrate that the performance improvements are not solely a result of multi-task learning.

4.2.2 Ablation experiments

Inclusion and Localization Chunk inclusion and localization (*i.e.* the losses $\mathcal{L}_{inc}$ and $\mathcal{L}_{loc}$) are the key novelties of our approach, providing strong improvements to our algorithm's performance. We demonstrate the importance of these tasks by performing two separate experiments. The first of these, shown in Table 2, indicates our performance when we remove each of these components from training entirely. In both cases, performance drops significantly. Unsurprisingly, both localization loss and inclusion loss help with temporal localization so the greatest impact can be seen at the higher tIoU values where precise start and end times are most important.

We also perform another set of experiments where the inclusion and localization values are learned, but are not used during inference. These results are shown in Table 3. Here we see the importance of performing the inclusion thresholding and temporal refinement, since removing either one significantly decreases performance. Additionally, the improvements from each are not independent since activating both of them together yields a bigger increase in performance than they each add individually; for instance, tIoU at 0.7 goes up about 4 points when inclusion thresholding is turned on and 10 points when temporal refinement is used, but nearly 17 points when both are used together.

Input Modality I3D can be trained using only the raw RGB frames of the input chunks, optical flow frames, or both at once. In Table 4, we compare the results of training on these three modalities. Our key observation is that optical flow significantly out-performs RGB, while combining both modalities into a two-stream network is optimal, similar to the observations made by [7] in the original I3D paper. The choice of modality also has important ramifications for the run-time of the algorithm, which we explore in more detail in Section 4.3.

tIoU	0.1	0.5	0.7
RGB	61.56	42.56	22.85
Flow	64.56	48.35	**28.22**
Joint (RGB+Flow)	**67.41**	**49.0**	27.82

Table 4. Classification accuracy for the three different input modalities on the THUMOS'14 dataset. Each row represents one of the modalities tested at 3 different temporal IoU thresholds. Best results are in bold.

tIoU	0.1	0.5	0.7
2 Skips	58.68	35.93	20.00
1 Skip	61.57	38.46	21.26
No Skips	**64.56**	**48.35**	**28.22**

Table 5. The proposed aggregation algorithm compared to an alternate aggregation algorithm in which chunks can be matched even if there are chunks in between of a different class. "1 Skip" and "2 Skip" refer to algorithms that merge chunks with one or two different entries between them, and "No Skips" is the algorithm we describe in Section 3.3. Best results are in bold.

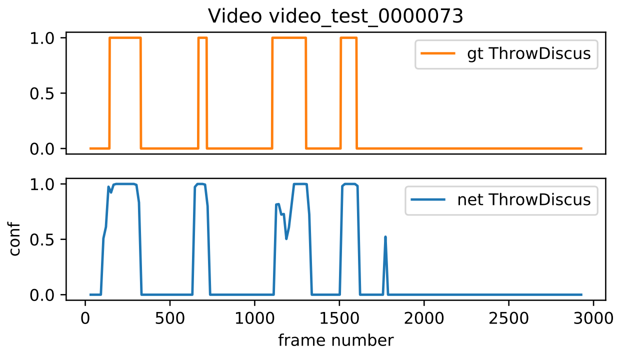

Figure 4. A plot of the network's per-chunk predictions (bottom) vs. the ground truth (top) on a sample video in the test set. The bottom plot shows the confidence value of each chunk in the sequence. Chunk confidence is plotted from the location of the chunk's center frame. For more intuitive visualization, the confidence values are suppressed to zero when the ThrowDiscuss class is not the maximal response for the chunk.

Aggregation In order to motivate our choice of aggregation algorithm, we present in Figure 4, a plot that shows the predictions of our chunks mapped against the ground truth of a given video. What we find is that, in general, chunks appear to do a good job classifying activities on their own, which motivates our choice to aggregate by connecting adjacent chunks.

In Table 5, we compare our chosen chunk aggregation algorithm to a slightly modified algorithm where, instead of merely connecting adjacent chunks, we also connect chunks

Mode	# GPUs	sec/chunk	Forward FPS	Total FPS
RGB	1	0.055	235.35	235.35
Flow	1	0.046	291.67	22.77
RGB+Flow	1	0.108	126.11	20.65
RGB	8	0.026	467.74	467.74
Flow	8	0.0208	612.0	121.93
RGB+Flow	8	0.0467	268.54	97.17

Table 6. Timing analysis of our algorithm under various configurations. "Forward FPS" refers to the frame rate of a forward pass through the network, while "Total FPS" includes the time to process input (*i.e.* optical flow computation). Timing of disk access was not included in this analysis.

that are of the same class with up to one or two differing chunks in between. We find that this actually decreases performance, which suggests that chunks being misclassified in the middle of activities are not a serious issue in THUMOS'14, and allowing skipping only hurts performance by merging chunks that should not be merged.

4.3. Timing-Performance Trade-offs

Here, we analyze the inference speed of our system, and show that with the right modifications our system can still achieve near state-of-the-art performance while running at speeds well above real time on a single GPU. Table 6 shows the results of these experiments. When optical flow is used, the vast majority of computation time is spent extracting the optical flow frames, and therefore multiple GPUs are required to perform real time inference. On the other hand, when inference is performed using only RGB frames, it is possible to perform computations significantly faster. With RGB input, a single GPU is enough to achieve processing speeds of 235 FPS, significantly faster than real time. In terms of performance, Table 4 shows that RGB, while worse than optical flow or RGB+optical flow, still achieves results competitive with the previous state-of-the-art. Disk read times were not considered in this timing analysis.

Thus, in addition to being simple, our algorithm is very well suited to real-world applications where speed would be an important factor. If, in addition, detection needs to be performed on live streamed video, then merely introducing a several-second lag would be sufficient to detect new activities as they happen. Further, since optical flow computation is the only stage of the algorithm with significant computational overhead, it may also be possible to achieve the state-of-the-art performance of optical flow with speeds much closer to that of RGB by estimating the flow or motion information directly from compressed video [49, 58, 48].

Method	P-Miss	N-MIDE
Gleason et al. [18]	67.50%	0.239
Ours	**61.71%**	**0.181**

Table 7. ActEV validation dataset results. In the left column we report weighted average probability of miss (weighted p-miss), where the weighting is applied to account for any class imbalance. In the right column we report the N-MIDE metric, where lower N-MIDE scores refer to better temporal localization. Both metrics are reported with a fixed rate of false alarm of 0.15.

Method	P-Miss
Aakur et al. [1]	93.4%
Xu et al. [50]	91.30%
Gleason et al. [18]	75.03%
MUDSML	**69.85%**
Ours	76.44%

Table 8. ActEV test dataset results. Here we report the weighted probability of miss of our system, which was evaluated on an independent evaluation server [36]. The metric is reported at a rate of false alarm of 0.15. To the best of our knowledge the top performer, known as MUSDML, has not published their approach.

4.4. ActEV: Spatio-Temporal Detection

4.4.1 Dataset

The ActEV dataset [36] is a spatio-temporal activity detection dataset featuring 18 different activities over 64 training videos, 54 validation videos, and 246 test videos with annotations withheld. All videos within the ActEV dataset are high resolution (1200×720 or 1920×1080) yet the subjects performing actions within the videos are very tiny by comparison, generally ranging from 20 to 180 pixels in height. Therefore, actions within the ActEV dataset tend to be spatially sparse, taking up only a very small portion of the scene. One of the primary challenges of the ActEV dataset is to intelligently avoid processing extraneous pixels. We do this by introducing a proposal-based method for generating chunks, as discussed in Section 3.4.

4.4.2 Results

In this section we report the results of our method on the ActEV validation and test sets. The primary metric for ActEV is probability of miss (p-miss) at a false alarm rate of 0.15 which is computed using the provided evaluation tool downloaded from Github [27]. We also report the N-MIDE metric on validation for which lower scores correspond to better temporal localization. Both of these metrics are described in detail in TRECVID 2018 [3].

Table 7 shows the results of our algorithm on the ActEV validation data using the method described in Section 3.4. In the table we included validation results using the TRI3D system described in [18], which, to our knowledge are the best published results on ActEV. The public leaderboard

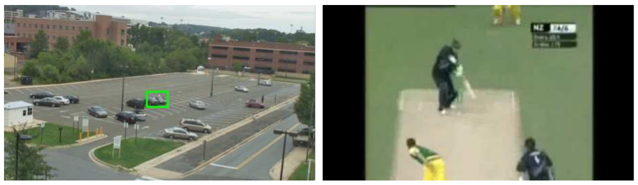
Figure 5. Spatial sparsity in ActEV. On the left is an example frame from ActEV, and on the right is an example frame from THUMOS'14. The green box in the ActEV image shows the spatial location of an instance of the `Closing` activity which covers only a small percentage of the overall image. Contrast this to the THUMOS'14 image where the activity `Cricket_Bowling` takes up a much greater proportion of the image. This figure was original presented in [18].

for the 2018 ActEV challenge was available at [36] where the top performing system achieves a probability of miss of 69.85% at a rate of false alarm of 0.15. Since the top performing system is unpublished as of the submission of this manuscript, it is difficult to comment on where the gain in performance comes from. This is particularly true since teams were allowed and encouraged to collect and use proprietary data to improve performance. Our system significantly out-performed the only other published systems which we know to have been evaluated on ActEV test [1, 50].

We observe that while we achieve a significant improvement over TRI3D [18] on the validation dataset, our system performs slightly worse on the test split as shown in Table 8. Since the annotations are not available for the test split it is difficult to draw any conclusions from this phenomenon, however, by contrasting our approach to TRI3D we have some thoughts on this.

On the validation dataset, our method outperforms TRI3D on all but 3 of the 18 activities: `activity_carrying`(+6.1%), `Closing`(+5.4%), and `Loading`(+5.4%), where the deltas reported here are the differences in average p-miss at 0.15 rate of false alarm. This indicates that the discrepancy is likely not due to some sort of biasing of our method towards certain actions. We also rule out the detector and clustering algorithm as being the cause of the discrepancy as both TRI3D and our method use the same approach. The difference between our algorithm and TRI3D at the core is that we use short, overlapping chunks, which are then aggregated, as opposed to varying length cuboids which are not aggregated. Our hypothesis is that the activities in the test dataset are more difficult to determine given the short temporal spans, and instead benefit from the high level context provided by long cuboids. We leave it to future work to test this hypothesis by extending our system to incorporate chunks of varying length.

5. Conclusion

We have presented a simple but effective approach to activity detection and classification in untrimmed videos. We have shown, through empirical analysis on the THUMOS'14 and ActEV datasets, that it is in fact possible to achieve results in untrimmed activity detection comparable to state-of-the art simply by making use of strong existing activity classification systems, and that with the addition of two auxiliary tasks it is possible to push these results significantly farther.

Acknowledgments

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 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] S. Aakur, D. Sawyer, and S. Sarkar. Fine-grained action detection in untrimmed surveillance videos. In *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, pages 38–40. IEEE, 2019. 7, 8

[2] H. Alwassel, F. Caba Heilbron, and B. Ghanem. Action search: Spotting actions in videos and its application to temporal action localization. In *The European Conference on Computer Vision (ECCV)*, 2018. 5

[3] G. Awad, A. Gov, A. Butt, K. Curtis, Y. Lee, y. Gov, J. Fiscus, D. Joy, A. Delgado, A. Smeaton, Y. Graham, W. Kraaij, G. Qunot, J. Magalhes, and S. Blasi. Trecvid 2018: Benchmarking video activity detection, video captioning and matching, video storytelling linking and video search. 04 2019. 7

[4] S. Buch, V. Escorcia, B. Ghanem, L. Fei-Fei, and J. Niebles. End-to-end, single-stream temporal action detection in untrimmed videos. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2017. 5

[5] S. Buch, V. Escorcia, C. Shen, B. Ghanem, and J. C. Niebles. Sst: Single-stream temporal action proposals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 5

[6] F. Caba Heilbron, J. Carlos Niebles, and B. Ghanem. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 5

[7] 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. 1, 2, 3, 5, 6

[8] Y.-W. Chao, S. Vijayanarasimhan, B. Seybold, D. A. Ross, J. Deng, and R. Sukthankar. Rethinking the faster r-cnn architecture for temporal action localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5

[9] X. Dai, B. Singh, G. Zhang, L. S. Davis, and Y. Q. Chen. Temporal context network for activity localization in videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2, 5

[10] V. Escorcia, F. C. Heilbron, J. C. Niebles, and B. Ghanem. Daps: Deep action proposals for action understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 2, 5

[11] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. *CoRR*, abs/1812.03982, 2018. 1

[12] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *Advances in neural information processing systems*, pages 3468–3476, 2016. 2

[13] C. Feichtenhofer, A. Pinz, and R. P. Wildes. Spatiotemporal multiplier networks for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4768–4777, 2017. 2

[14] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1933–1941, 2016. 2

[15] J. Gao, K. Chen, and R. Nevatia. Ctap: Complementary temporal action proposal generation. In *The European Conference on Computer Vision (ECCV)*, 2018. 5

[16] J. Gao, Z. Yang, and R. Nevatia. Cascaded boundary regression for temporal action detection. 2017. 2, 5

[17] J. Gao, Z. Yang, C. Sun, K. Chen, and R. Nevatia. Turn tap: Temporal unit regression network for temporal action proposals. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 2, 5

[18] J. Gleason, R. Ranjan, S. Schwarcz, C. D. Castillo, J.-C. Cheng, and R. Chellappa. A proposal-based solution to spatio-temporal action detection in untrimmed videos. *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 141–150, 2018. 2, 3, 4, 5, 7, 8

[19] 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*, pages 6546–6555, 2018. 2

[20] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017. 4

[21] R. Hou, C. Chen, and M. Shah. Tube convolutional neural network (t-cnn) for action detection in videos. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[22] R. Hou, R. Sukthankar, and M. Shah. Real-time temporal action localization in untrimmed videos by sub-action discovery. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2017. 5

[23] J. Huang, N. Li, T. Zhang, G. Li, T. Huang, and W. Gao. Sap: Self-adaptive proposal model for temporal action detection based on reinforcement learning. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018. 5

[24] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013. 2

[25] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. `http://crcv.ucf.edu/THUMOS14/`, 2014. 2, 5

[26] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. 4

[27] D. Joy. Actev scorer. `https://github.com/usnistgov/ActEV_Scorer/tree/v0.3.0`, 2018. 7

[28] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid. Action tubelet detector for spatio-temporal action localization. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[29] S. Karaman, L. Seidenari, and A. Del Bimbo. Fast saliency based pooling of fisher encoded dense trajectories. In *ECCV THUMOS Workshop*, 2014. 2, 5

[30] 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 IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. 2

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 2

[32] T. Lin, X. Zhao, H. Su, C. Wang, and M. Yang. Bsn: Boundary sensitive network for temporal action proposal generation. In *The European Conference on Computer Vision (ECCV)*, 2018. 5

[33] F. Murtaza, M. H. Yousaf, S. Velastin, and Y. Qian. End-to-end temporal action detection using bag of discriminant snippets (bods). *IEEE Signal Processing Letters*, PP:1–1, 12 2018. 5

[34] D. Mllner. Modern hierarchical, agglomerative clustering algorithms, 2011. 4

[35] F. Negin and F. Bremond. Human action recognition in videos: A survey. *INRIA Technical Report*, 2016. 2

[36] NIST. Activity extended video (actev) prize challenge, https://actev.nist.gov/prizechallenge. 2, 5, 7, 8

[37] D. Oneata, J. Verbeek, and C. Schmid. The lear submission at thumos 2014. 2014. 2, 5

[38] 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. 2, 4

[39] A. Richard and J. Gall. Temporal action detection using a statistical language model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 5

[40] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S.-F. Chang. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1417–1426, 2017. 2, 5

[41] Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[42] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014. 2

[43] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of The IEEE International Conference on Computer Vision (ICCV)*, 2015. 2

[44] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1510–1517, 2018. 2

[45] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *International journal of computer vision*, 103(1):60–79, 2013. 2

[46] L. Wang, Y. Qiao, and X. Tang. Action recognition and detection by combining motion and appearance features. 2014. 2, 5

[47] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4305–4314, 2015. 2

[48] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016. 2, 7

[49] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl. Compressed video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6026–6035, 2018. 7

[50] H. Xu, A. Das, and K. Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2, 5, 7, 8

[51] K. Yang, P. Qiao, D. Li, S. Lv, and Y. Dou. Exploring temporal preservation networks for precise temporal action localization. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018. 5

[52] K. Yang, P. Qiao, D. Li, S. Lv, and Y. Dou. Exploring temporal preservation networks for precise temporal action localization. In *AAAI*, 2018. 5

[53] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 5

[54] J. Yuan, B. Ni, X. Yang, and A. A. Kassim. Temporal action localization with pyramid of score distribution features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 5

[55] Z.-H. Yuan, J. C. Stroud, T. Lu, and J. Deng. Temporal action localization by structured maximal sums. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 5

[56] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702, 2015. 2

[57] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Joint Pattern Recognition Symposium*, pages 214–223. Springer, 2007. 3

[58] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang. Real-time action recognition with deeply transferred motion vector cnns. *IEEE Transactions on Image Processing*, 27(5):2326–2339, 2018. 7

[59] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin. Temporal action detection with structured segment networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 5