Efficient Localization of Human Actions and Moments in Videos

Dissertation by

By

Victor Escorcia

In Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy

King Abdullah University of Science and Technology

Thuwal, Kingdom of Saudi Arabia

June, 2019
The dissertation of Victor Augusto Escorcia Castillo is approved by the examination committee.

Committee Chairperson: Prof. Bernard Ghanem
Committee Member: Prof. Peter Wonka, Prof. Helmut Pottmann, Prof. Kate Saenko
ABSTRACT

Efficient Localization of Human Actions and Moments in Videos
Victor A. Escorcia Castillo

We are stumbling across a video tsunami flooding our communication channels. The ubiquity of digital cameras and social networks has increased the amount of visual media content generated and shared by people, in particular videos. Cisco reports that 82% of the internet traffic would be in the form of videos by 2022. The computer vision community has embraced this challenge by offering the first building blocks to translate the visual data in segmented video clips into semantic tags. However, users usually require to go beyond tagging at the video level. For example, someone may want to retrieve important moments such as the “first steps of her child” from a large collection of untrimmed videos; or retrieving all the instances of a home-run from an unsegmented video of baseball. In the face of this data deluge, it becomes crucial to develop efficient and scalable algorithms that can intelligently localize semantic visual content in untrimmed videos.

In this work, I address three different challenges on the localization of actions in videos. First, I develop deep-based action proposals and detection models that take a video and generate action-agnostic and class-specific temporal segments, respectively. These models retrieve temporal locations with high accuracy in an efficient manner, faster than real-time. Second, I propose the new task to retrieve and localize temporal moments from a collection of videos given a natural language query. To tackle this challenge, I introduce an efficient and effective model that aligns the text query to individual clips of fixed length while still retrieves moments spanning multiple clips. This approach not only allows smooth interactions with users via natural language
queries but also reduce the index size and search time for retrieving the moments. Lastly, I introduce the concept of actor-supervision that exploits the inherent compositionality of actions, in terms of transformations of actors, to achieve spatiotemporal localization of actions without the need of action box annotations. By designing efficient models to scan a single video in real-time; retrieve and localizing moments of interest from multiple videos; and an effective strategy to localize actions without resorting in action box annotations, this thesis provides insights that put us closer to the goal of general video understanding.
Dedicated to my family and the memory of my grandparents and my uncle.
Our work is just the expression of a cummulus of decisions and random events in our life, as such I’m profoundly grateful with the universe for all the individuals and opportunities that appeared along this journey.

In particular, I thank my advisor Prof. Bernard Ghanem for his encouragement to aim the highest academic standard, his invaluable support during all the stages of my Ph.D., and for teaching me that we must be independent thinkers. It was a great joy to discuss any research topic with him and establishing new connections or perspectives not outlined before. I also thank Prof. Helmut Pottmann, Prof. Peter Wonka, and Prof. Kate Saenko being part of my thesis the committee and for their candid feedback during the final stretch.

From the beginning of my career, I was blessed to share stage with great individuals. Among those, Prof. Juan Carlos Niebles who enrolled me on “research” and planted the seed of not letting the details to hinder the big picture. On a least senior level but equally important, it was terrific to “grow up” next to Fabian, Shyamal and Humam. They were an excellent complement for me, but the most important was to count them as friends to share thoughts, jokes and fun moments. Last but not the least, it was invaluable to lead peers like Erik and Mattia - without strong research background - capable to get the things done despite of the odds.

I had the fortune to spend two summers in industry with charming people. One in the Netherlands with the friendly crew of Qualcomm AI, I learnt a lot from the pragmatic approach of Prof. Cees Snoek and the Morpheus team. In particular, it was fulfilling to explore new topics and flavours with Dr. Mihir Jain. And the second with Adobe, San Francisco - CA, next to Dr. Bryan Russell and Prof. Josef Sivic. It was fantastic to appreciate how rich, clean and organized their research approach is, besides of its delightful personalities.
It would be unfair to not highlight all the people behind me during this journey. From my IVUL family, I had the pleasure to overlap with the magnificent Adel, the grrreat Modar, the expert Silvio, Matthias, Jean, Chen, Abdullah, Alejandro, Frost, Salman, Motasem, Hani, Merey, Tina Smith and others alumni and members. We had great nerdy talks, but our gatherings are way more memorable with and without Perudo/Coup. Besides of my life at the lab, I had the pleasure to meet amazing characters - my KAUST family - David Evangelista, Ali Thabet, Sandra Medina, Diego, David Conchouso, Christel, Mafe, Francy, Rodrido, Juan Carlos, Arpies, Edward, Jhonathan (Muñeco), Mary, Liliana, Valentina, Monica Torsello, Tsiky, and many others. They helped me to release the stress from research with laughs, fun gatherings, delicious meals, and memorable moments.

Latest but not the least important, my deepest gratitude goes to my family, girlfriend and friends in Colombia and around the world for their encouragement and support throughout these years. My siblings, Alvaro and Andrea, who always have expressed their affection and gave me the chance to be uncle. My parents, Judith and Alvaro, who inspire me to pursue long-term goals without giving up as pain appears, and to accomplish our duties in the correct manner. And my girlfriend Giada who gave color and curls to my days at KAUST, made this ride unforgettable, and offered me her undeniable support.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examination Committee Page</td>
<td>2</td>
</tr>
<tr>
<td>Copyright</td>
<td>3</td>
</tr>
<tr>
<td>Abstract</td>
<td>4</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>7</td>
</tr>
<tr>
<td>List of Figures</td>
<td>13</td>
</tr>
<tr>
<td>List of Tables</td>
<td>15</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>16</td>
</tr>
<tr>
<td>1.1 Motivation and Overview</td>
<td>16</td>
</tr>
<tr>
<td>1.2 Related Work</td>
<td>19</td>
</tr>
<tr>
<td>1.3 Contributions and Outline</td>
<td>20</td>
</tr>
<tr>
<td><strong>2 Temporal Localization of Actions in a Single Video</strong></td>
<td>22</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>22</td>
</tr>
<tr>
<td>2.2 Related Work</td>
<td>25</td>
</tr>
<tr>
<td>2.3 Deep Action Proposals</td>
<td>27</td>
</tr>
<tr>
<td>2.3.1 Architecture</td>
<td>27</td>
</tr>
<tr>
<td>2.3.2 Training</td>
<td>30</td>
</tr>
<tr>
<td>2.4 Experimental Analysis</td>
<td>33</td>
</tr>
<tr>
<td>2.4.1 Recall Analysis</td>
<td>33</td>
</tr>
<tr>
<td>2.4.2 DAPs for Temporal Action Localization</td>
<td>39</td>
</tr>
<tr>
<td>2.4.3 Run-time Performance</td>
<td>41</td>
</tr>
<tr>
<td>2.4.4 Qualitative results</td>
<td>42</td>
</tr>
<tr>
<td>2.5 SST: Single Stream Temporal Proposal Generation</td>
<td>44</td>
</tr>
<tr>
<td>2.5.1 Model</td>
<td>45</td>
</tr>
<tr>
<td>2.5.2 Training</td>
<td>48</td>
</tr>
<tr>
<td>2.5.3 Experiments</td>
<td>50</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Motivation</td>
<td>17</td>
</tr>
<tr>
<td>2.1</td>
<td>Temporal action localization in a single video</td>
<td>23</td>
</tr>
<tr>
<td>2.2</td>
<td>Deep action proposals for temporal localization</td>
<td>24</td>
</tr>
<tr>
<td>2.3</td>
<td>DAPs architecture</td>
<td>27</td>
</tr>
<tr>
<td>2.4</td>
<td>Effects of multiple hyperparameters in the localization performance</td>
<td>34</td>
</tr>
<tr>
<td>2.5</td>
<td>Comparison against other action proposal approaches</td>
<td>36</td>
</tr>
<tr>
<td>2.6</td>
<td>Generalization of action proposals generated by DAPs</td>
<td>38</td>
</tr>
<tr>
<td>2.7</td>
<td>Qualitative results of DAPs proposals</td>
<td>43</td>
</tr>
<tr>
<td>2.8</td>
<td>SST motivation</td>
<td>44</td>
</tr>
<tr>
<td>2.9</td>
<td>SST architecture</td>
<td>45</td>
</tr>
<tr>
<td>2.10</td>
<td>Schematic exemplifying training sample generation</td>
<td>49</td>
</tr>
<tr>
<td>2.11</td>
<td>Comparison of action proposals</td>
<td>51</td>
</tr>
<tr>
<td>2.12</td>
<td>Robustness of action proposals over time</td>
<td>53</td>
</tr>
<tr>
<td>2.13</td>
<td>Qualitative results of SST proposals</td>
<td>54</td>
</tr>
<tr>
<td>2.14</td>
<td>End-to-end temporal action detection motivation</td>
<td>56</td>
</tr>
<tr>
<td>2.15</td>
<td>SS-TAD architecture for temporal action detection</td>
<td>57</td>
</tr>
<tr>
<td>2.16</td>
<td>Memory modules for temporal action detection</td>
<td>61</td>
</tr>
<tr>
<td>2.17</td>
<td>Relevance of semantically-constrained memory modules</td>
<td>66</td>
</tr>
<tr>
<td>2.18</td>
<td>Qualitative action localization results</td>
<td>68</td>
</tr>
<tr>
<td>3.1</td>
<td>Moments retrieval from a video collection given a natural language query</td>
<td>70</td>
</tr>
<tr>
<td>3.2</td>
<td>Overview of Clip Alignment with Language model</td>
<td>71</td>
</tr>
<tr>
<td>3.3</td>
<td>Clip Alignment with Language architecture</td>
<td>73</td>
</tr>
<tr>
<td>3.4</td>
<td>System for indexing and retrieval</td>
<td>78</td>
</tr>
<tr>
<td>3.5</td>
<td>Qualitative results of moments retrieval from a video corpus</td>
<td>84</td>
</tr>
<tr>
<td>4.1</td>
<td>Actor-supervision motivation</td>
<td>95</td>
</tr>
<tr>
<td>4.2</td>
<td>Actor-supervised architecture</td>
<td>99</td>
</tr>
<tr>
<td>4.3</td>
<td>Generation of actor proposals</td>
<td>100</td>
</tr>
<tr>
<td>4.4</td>
<td>Actor pooling schematic</td>
<td>102</td>
</tr>
</tbody>
</table>
LIST OF TABLES

2.1 Localization accuracy as a function of the number of proposals 40
2.2 Temporal action localization performance 41
2.3 Runtime comparison among action proposal methods 42
2.4 Comparison of action proposals methods 52
2.5 Temporal action detection results 65

3.1 Dataset settings and statistics for our video corpus moment retrieval task 81
3.2 Video corpus moment retrieval results - exhaustive setting 83
3.3 Efficient retrieval of moments with re-ranking 85
3.4 Runtime and index size benchmark 87
3.5 Feature ablation for video corpus moment retrieval 88
3.6 Single video moment retrieval results 90
3.7 Single video moment retrieval results on DiDeMo 92

4.1 Comparison of spatiotemporal action proposals 108
4.2 Impact of actor attention layer 110
4.3 Comparison of spatiotemporal action localization approaches 111
4.4 Localization results across multiple IOU values 112

A.1 Text to video retrieval results 135
Chapter 1

Introduction

1.1 Motivation and Overview

The ubiquity of digital cameras and social networks has increased the amount of visual media content generated and shared by people, in particular videos. Cisco reports that 82% of the internet traffic would be in the form of videos by 2022 [1], while Youtube claims an impressive number of 400 video hours uploaded every minute to their servers [2]. We are stumbling across a video tsunami flooding our communication channels with data in the form of pixels and without precise sense of the value of this information. In the face of this data deluge, it becomes crucial to develop efficient and scalable algorithms that can intelligently sift, scan and outline semantic visual content in the videos. Such algorithms would contribute and facilitate the development of various applications like large-scale video indexing and retrieval, intention-aware autonomous driving, robots for healthcare assistance, among others.

Let consider the three tasks depicted in Figure 1.1. Humans can accomplish them almost easily (with the proper will). A person can sift through a video and find the temporal endpoints when “polishing shoes” occurs; she can also scan a collection of videos and return only the videos and the temporal segments enclosing moments displaying “the girl jumps up and down”; and it would not be a concern for her to outline the actor executing the action, as well. The fact that we do these tasks almost effortlessly hinders the difficulty to solve this task from a computer’s perspective. Let briefly revisit some of the challenges associated with these tasks.
Figure 1.1: **Overview.** This work tackles the problem of automatic localization of human actions and moments of interest in videos in an extensive manner. Concretely, it considers the temporal localization of human actions in a single untrimmed video (Task 1); the retrieval and temporal localization of moments associated with natural language queries from a large video collection (Task 2); and outlining the actor performing the actions (Task 3).

**Diversity.** Videos in the wild capture a wide diversity of the visual word in terms of objects and scenes. Youtube exhibits videos a diverse set of actors and scenes; from people’s everyday life and holidays, to video games and cartoons with creatures antropomorphic behaviors. Developing algorithms that deal with this humonguous variability is a heroic, but ill-defined challenge. Thus, this work takes an anthropocentric approach and focuses its interest on the localization of human actions or...
moments of interest for humans. Interestingly, people are not only a relevant subject of interest for utilitarian purposes, but also covers a significant part of the current video footage [2].

Despite of narrowing the focus to human actions and moments of interest for people, the diversity of the visual world is still significantly wide. An automatic algorithm should account from people with different gender, race, age; interacting with a varied set of objects; executing the actions with their own style and pace; on videos shot from different viewpoints. Fortunately, rapid progress has been made in the area of visual recognition, in particular with deep convolutional networks achieving unprecedent performance to classify thousands of visual categories [3, 4, 5]. This family of models not only performs as accurate as humans, but also introduces a modular approach to tackle visual perception tasks as their parameters can be easily adjusted to other tasks without the need of excesive labeled data [6, 7]. This work builds on top of this disruptive paradigm to deal with the diversity of the visual world. At the same time, it presents complementary findings in relation to the representation power of deep models in the context of videos.

Untrimmed unsegmented videos. The understanding of human movement and behavior has been a long-standing goal in computer vision [8, 9]. Due to the engineering challenges of dealing with videos, most of the work has focused on segmented video clips around human actions of interest [10, 11, 12, 13, 14, 15, 16, 17]. However, humans perceive the worlds continously. We identify and keep track of relevant actions on untrimmed videos by itself [18]. Thus, it is pertinent to design computational models operating in the same way. Furthermore, untrimmed unsegmented videos pose a unique opportunity to model transitions between idle or irrelevant movements and actions of interest as well as studying long-term relationships and understanding in videos. Inspired by the pervasity and uniqueness of untrimmed videos, this work devotes a significant part to develop automatic algorithms to process them. Moreover,
the large-scale nature of videos demands the development of efficient models that can keep up with the impressive rate of growth [1, 2]. This is precisely the focus of this work which covers the retrieval and localization of human actions and moments in a single video and a video collection.

1.2 Related Work

This dissertation draws inspiration from the large body of work in action recognition [12, 13, 14, 15, 16] as well as early work in the intersection of video and language [19, 20]. Herath et al. [21] offer a more comprehensive review of the analysis of human actions in video. In what follows, we briefly introduce the closest work related to this work.

Temporal action localization. A small number of work pre-deep-learning era were interested on the temporal localization of actions in untrimmed videos [22, 23, 24]. Multi-class classifier running on a sliding window fashion determines the occurrence of actions based on hand-crafted features [10, 14], and high-order pooling techniques [25, 26]. Due to the high-computational demand of the previous approach, [27, 28] formulate the localization problem as a two-stage system. During the first-stage (action proposals) one could quickly reject segments of the video unlikely to have actions. Followed by a classification-stage over the segments retrieved by the first-stage. Contemporary with them, this work (Chapter 2) demonstrates how to generate action proposals in a single-pass over the video by leveraging recurrent modules, and without relying on expensive multi-scale feature extractors. Moreover, this dissertation also shows that fusing the two-stages into a single end-to-end model results in further speed-up and accuracy on the temporal localization of actions.

Temporal moments retrieval from language queries. Recently, datasets of short video clips with accompanying natural language have emerged [29, 30]. Followed
by recent approaches leveraging detected concepts in videos to formulate scalable retrieval in a text-to-video fashion [31, 32, 33, 34]. On the other hand, [35, 36] introduce the problem of localizing moments, temporal snippets, associated with a natural language query in a single untrimmed video. In Chapter 3, I defy the scalability of retrieving and localizing moments from a video collection.

**Spatiotemporal action localization.** Outlining the actor executing an action in videos is a well-established problem in video understanding [37, 38, 13, 15]. Much progress was been made over the last years thanks to deep models and exhaustive annotations [39, 40, 41]. Due to the tedious and expensive annotation process, effective ways to scale the localization problem without action boxes annotations are much needed as I illustrate in Chapter 4.

1.3 Contributions and Outline

This dissertation presents automatic algorithms for understanding the visual information present in videos. In particular, visual content associated with human actions or moments of interest. In the following chapters, I introduce models that efficiently sift, retrieve and localize actions or moments in a single or a collection of videos.

In Chapter 2, I introduce new deep-based action proposals and detection models for temporal localization in a single untrimmed video. These models takes a video and generate action-agnostic and class-specific temporal segments, respectively. They generates the detections in real-time by performing a single pass over the video in roughly 700 FPS. The content of this chapter is based on my seminal single pass action proposal work, Escorcia *et al.* [42], as well as its latter enhancements [43], and its application as an efficient temporal action detector [44]. The last two were developed in close collaboration with Shyamal Buch who was the first author in those publications.
Chapter 3 presents a new task involving vision and language understanding, retrieve and localize temporal moments from a collection of videos given a natural language query. To tackle this challenging task, I introduce an efficient and effective model that aligns the text query to individual clips of fixed length while still retrieves moments spanning multiple clips. This approach not only allows smooth interactions with users via natural language queries, but also reduce the index size and search time for retrieving the moments. The content of this chapter was developed in collaboration with Adobe Research where I served as project leader and principal developer.

In Chapter 4, I introduce the concept of actor-supervision to achieve spatiotemporal localization of actions in segmented videos, such as the outputs from the previous Chapters. Actor-supervision exploits the inherent compositionality of actions, in terms of transformations of actors, to reduce the amount of exhaustive annotations required to train deep model. This Chapter’s content was developed in collaboration with Qualcomm Research where I served as project leader and principal developer. These insights also constitute the basis for two US patents [45, 46].

Finally, a summary of the major finding and a discussion about the path forward in presented in Chapter 5.
Chapter 2

Temporal Localization of Actions in a Single Video

2.1 Introduction

Nowadays, the ubiquity of digital cameras and social networks has increased the amount of visual media content generated and shared by people. For example, Cisco reports that 82% of the internet traffic would be in the form of videos by 2022 [1]. In the face of this data deluge, it becomes crucial to develop efficient and scalable algorithms that can intelligently parse/browse videos to discover semantic information. This Chapter focuses on the task of quickly localizing temporal segments in untrimmed, unsegmented videos that are likely to contain human activities of interest. Outlining segments of interest in a video, and possibly associating an action label to them, is useful to retrieve snippets of a home-run being scored within a long baseball game or extract relevant moments during the construction of a new skyscraper.

Motivated by the large-scale nature of the problem, we develop temporal localization algorithms that retrieves well aligned segments around human actions with a much faster computational runtime than previous methods (refer to Figure 2.2).

The idea of outlining visual regions with semantic content has been explored on a single image or segmented video clips [47]. The key insight to achieve an efficient runtime resided on formulating the localization problem as a cascade system, by quickly rejecting a region where is not likely to find human faces during early stages of the cascade [47]. Object proposals scale up this approach for multiple classes [48, 6], and turned out to be one of the ingredients for the modern success of object detection.
Figure 2.1: **Overview.** This Chapter focuses on the temporal localization of actions in untrimmed videos by means of its enclosing temporal segment. In particular, given an action of interest such as “polishing shoes” we seek to retrieve all the green segments showcasing the action while disregarding irrelevant content.

at large scale with an appealing runtime [49, 5]. Efficient and effective localization enabled the development other visual tasks in images, such as instance segmentation, object tracking, and image captioning [50, 51, 52, 53]. We argue that the development of temporal localization algorithms should be put in the forefront of human activity understanding to push forward the parsing of untrimmed videos.

Inspired by object proposal approaches for localizing object in a single image [54, 48], Jain *et al.* introduced the concept of class-agnostic action proposals in videos which correspond to spatiotemporal tubelets where it is likely to find an action [55]. It is tempting to think that retaining the temporal extend of these tubelets would result in good temporal segments confining actions. However, it was recently shown that the temporal footprint of some methods can be as accurate as sampling temporal proposals uniformly in the video [28]. Moreover, [15] evaluated their performance in short video clips depicting simple or repetitive actions, which makes it difficult to gauge their scalability to untrimmed video sequences containing complex activities [56, 57]. Given the current state of spatiotemporal action proposals, it is worth exploring how temporal-only action proposals can contribute to the analysis of unsegmented videos.

Contemporary work has explored the generation of temporal action proposals directly from videos [28, 58, 27]. These approaches focus on exploring a large number of segments in the video at multiple scales (i.e. proposals with varied durations) and ranking them with an action agnostic classification pipeline. Unlike these methods, we
Figure 2.2: **Our method DAPs** efficiently retrieves segments with varied duration likely to enclose actions without exhaustively exploring multiple temporal scales. This Chapter focus on effective temporal localization of actions in a single video 10x faster than contemporaneous approaches.

leverage the capacity of deep learning models with memory blocks to extract action proposals at different temporal scales in a single-pass through the video. Refer to Figure 2.2. This is done by encoding a video sequence as a discriminative sequence of states, from which action likely segments can be localized with varied duration inside a video sequence.

Efficient action proposals are a natural precursor for temporal action detection in videos as depicted in Figure 2.1. The temporal proposal generates the relevant segments which can be further examined by an action classifier that determines the action categories occurring inside them. Yet effective, these coupling (proposals+classifiers) is not fully efficient for the down-stream task as the two stages are designed independently. We showcase that integrating both approaches into a single end-to-end model not only achieves faster speed-up but also better localization accuracy.

**Contributions of this Chapter**

In this chapter, we propose a (i) new approach for temporal action proposal generation specifically targeting untrimmed videos. This is done by training a well-suited memory network to reliably output the temporal location and scale of action proposals. Our model is able to generate proposals accross multiple temporal scales with
a single pass through the video and to generalize well to new unseen actions. This translates into a computationally efficient and runs at 134-300 FPS. (ii) We also demonstrate that effective and real-time temporal detection of actions in videos is possible by coupling our proposal model and an action classifier into a single monolithic end-to-end architecture.

2.2 Related Work

We summarize the most recent work on topics related to the task of action proposal generation and our proposed methodology.

Object Proposals. Exhaustively running computationally intensive object classifiers with a sliding window approach is not as common as it was eight years ago. Instead, the use of generic or class-specific object proposals is now a cornerstone in the object detection pipeline. These proposal algorithms retrieve high-quality candidate regions that are likely to contain an object (high recall), before classification is performed [59, 5, 60]. This approach has proven to be an effective and scalable way to find possible locations of an object in an image.

The latest trend in this area is designing algorithms with high ranking quality i.e. achieving high object recall with less number of bounding boxes, preferably with a small computational overhead and the potential to scale to hundreds of object categories [48, 61, 62]. Here, discriminative methods based on deep learning models have helped improve the ranking quality of proposal approaches [63, 62, 49, 64]. Inspired by this work, we extend the use of deep and recurrent networks to temporal action proposal generation by introducing a new architecture.

Spatiotemporal Action Proposals. Recently, ideas from the area of object proposals have been extrapolated to action recognition in the video domain [65, 60, 55, 52, 67, 68, 69, 70]. Most of these methods produce spatio-temporal object segments
to perform spatio-temporal detection of simple or cyclic actions in short video sequences, hence their scalability to real-world scenarios is uncertain. These methods rely on straddling of voxels [66, 55], reasoning over dense trajectories [65, 68, 69], or non real-time object proposals [52, 70], which increase their computational cost and reduce their competitiveness at large scale.

Temporal Action Proposals. Very recently, work emerged that focused on temporal segments which are likely to contain human actions [28, 58, 27]. Similar to grouping techniques for retrieving object proposals, Mettes et al. create a hierarchy of fragments by hierarchical clustering based on semantic visual similarity of contiguous frames [58]. The main disadvantages of this approach are its strong dependence on an unsupervised grouping method that diminishes its repeatability [60] and the absence of an actionness score for each fragment in the hierarchy. In comparison, we use a supervised method that learns to generate segments on a video and predict their action likelihood. Most closely related with our approach are methods that use category-independent classifiers to explore many segments in video and exhaustively evaluate segments of multiple temporal scales [28, 27]. Our method improves over previous ones by using a powerful deep learning model that allows for less windows to be scanned and multiple temporal scales to be considered simultaneously in a single pass through the video. We leverage long-short term memory cells to learn an appropriate encoding of the video sequence as a set of discriminative states. We experimentally show that this representation is able to regress the temporal location and duration of relevant segments on the original sequence, while running at 134 FPS.

Temporal Action Detection. In contrast to object detection methods, the dominant approach for action detection is still to use a sliding window approach [71, 56, 72] combined with action classifiers trained on multiple features [73, 74, 75]. Previous approaches have reduced the computational overhead of sliding window search by us-
Figure 2.3: **Our Deep Action Proposals (DAPs) architecture** effectively encodes a segment of visual observations (of length $L$ frames) into discriminative states from which it is possible to localize $K$ proposals $\{s_i\}_{i=1}^K$ with confidences $\{c_i\}_{i=1}^K$ inside the segment. We generate several segments, where it is possible to find actions, along an untrimmed video by sliding our architecture with step size $\delta$.

We introduce *Deep Action Proposals* (DAPs), a new network for the task of temporal action proposal generation. From a long input video sequence, we aim to retrieve temporal segments with multiple durations where it is likely to find actions of interest. Figure 2.3 summarizes our model architecture, which is described in detail in Section 2.3.1. Section 2.3.2 describes the training procedure.

**2.3 Deep Action Proposals**

Let consider a segment of length $L$ frames, highlighted in dark yellow in Figure 2.3. Our DAPs network encodes the stream of visual observations into discriminative
states, from which we infer the temporal location and duration \( \{s_i\}_{i=1}^{K} \) of \( K \) action proposals inside the stream. Each proposal \( s_i \) is associated with a confidence score \( c_i \). Our network integrates the following modules:

**Input.** Our model processes all the frames \( X = \{x_i\}_{i=1}^{L} \) enclosed by the segment of length \( L \) highlighted in Figure 2.3.

**Visual encoder module.** It encapsulates the visual information from a group of frames into a meaningful low dimensional representation. We achieve this by processing non-overlapping clips of \( d \) frames with a 3D convolutional network pre-trained for action classification [78], C3D. This convolutional network, CNN, efficiently encodes RGB and motion information over a small temporal resolution, \( d = 16 \) frames. In this manner, we discretize the input stream into \( T = L/d \) non-overlapping time steps and each time step is represented by a C3D activation \( v_t = \phi_W(x_i)_{i=td}^{(t+1)d} \), \( v_t \in \mathbb{R}^D \) and \( W \) corresponds to all the parameters of the CNN. In practice, we use the activations from the penultimate layer of C3D, fc7 [78], and reduce its dimensionality to 500 dimensions using PCA to decrease training time.

**Sequence encoder.** Encode the sequence of visual codes \( V = \{v_t\}_{t=1}^{T} \) as a discriminative sequence of hidden states. We achieve this with a long-short term memory network, LSTM, which models the sequential information in a principled and effective manner [79, 80, 81]. In particular, we use the following recurrent equation:

\[
\begin{align*}
 i_t &= \sigma(W_{vi}v_t + W_{hi}h_{t-1} + w_{ui} \odot u_{t-1} + b_i) \\
 r_t &= \sigma(W_{vr}v_t + W_{hr}h_{t-1} + w_{ur} \odot u_{t-1} + b_r) \\
 u_t &= r_t \odot u_{t-1} + i_t \odot \tanh(W_{vu}v_t + W_{hu}h_{t-1} + b_u) \\
 o_t &= \sigma(W_{vo}v_t + W_{ho}h_{t-1} + w_{uo} \odot u_{t-1} + b_o) \\
 h_t &= o_t \odot \tanh(u_t)
\end{align*}
\] (2.1)
where $i_t, r_t, u_t, o_t \in \mathbb{R}^{D'}$ correspond to the internal representation of the input, forget vector, cell state, and output of the LSTM, respectively; $\sigma(z) = \frac{1}{1 + \exp(-z)}$ correspond to the sigmoid function; and $\odot$ denotes an element-wise product. The matrices $W_\ast$, cell vectors $w_c$ and the bias vectors $b_\ast$ correspond to the parameters of our sequence encoder.

**Localization module.** It predicts the location of $K$ proposals inside the stream based on an affine transformation of the last state of the *sequence encoder*. In this way, our model output segments of different lengths after scanning $L$ frames of the video without exhaustively scanning multiple window sizes independently. In practice, our segments are given by $s = W_{os} o_{t=T} + b_o$, where $W_{os} \in \mathbb{R}^{2K \times D'}$; and are parametrized in terms of its central frame and duration.

**Prediction module.** It predicts the confidence that the $k$-th proposal is associated with an action. In practice, $c$ is the output of a sigmoid function over an affine transformation of the last state of the sequence encoder *i.e.* $c = \sigma(W_{oc} o_{t=T} + b_c)$, where $W_{oc} \in \mathbb{R}^{K \times D'}$.

**Inference: parsing arbitrary length videos.** So far we described the computation of our DAPs network over $L$ frames inside a video. Our architecture can parse arbitrary length videos as well by sliding our DAPs network for each segment of length $L$ with a step size $\delta$, as depicted in Figure 2.3. Every time our model processes a video stream with $L$ frames, it places $K$ segments of varied duration inside it with their respective action likelihoods. Note that our visual encoder only encodes each frame once, as it only processes non-overlapping clips with $d$ frames. Thus, we can store and re-use this representation when we perform a forward-pass through the sequence encoder. In this manner, our algorithm efficiently scans the whole video sequence in *only* one pass with a single scale (or window) of size $L$, while still producing proposals.
of different duration.

2.3.2 Training

We seek to have our DAPs model correctly generating: (i) temporal proposals tightly covering the extent of the actions occurring in a video sequence; (ii) high confidence values when a proposal is matching an action; and (iii) low confidence values when a proposal covers background regions. We achieve these by defining a training objective based on supervised learning that resembles the perception of human annotators in regard to the endpoints of human actions.

Let $\mathcal{X} = \{X, a\}_{i=1}^{M}$ be a training set of $M$ video sequences each with $N_i$ temporal action annotations $a = \{a_j\}_{j=1}^{N_i}$ and length $T$ frames; and denote the mapping from $X$ into $K$ proposals as $(s, c) = g_\theta(X)$; where $g$ is the non-linear function summarizing all the computations of the DAPs’ modules described before, and $\theta = \{W_{**}, w_{**}, b_{*}\}$ groups all the parameters of our architecture. The following training loss, minimized with respect to the model parameters $\theta$, achieves our three goals (i - iii):

$$
\mathcal{L}_\theta = \sum_{(X,a) \in \mathcal{X}} \alpha \mathcal{L}_{\text{match}}(s, a, Y) + \mathcal{L}_{\text{conf}}(c, y)
$$

(2.2)

$$
Y \in \{0, 1\}^{K \times |a|}, \ y = Y 1
$$

$$
\mathcal{L}_{\text{match}}(s, a, Y) = \sum_{k=1}^{K} \sum_{j=1}^{|a|} Y_{kj} ||s_k - a_i||_2^2
$$

(2.3)

$$
\mathcal{L}_{\text{conf}}(c, y) = \sum_{k=1}^{K} w_1 y_k \log c_k + w_0 (1 - y_k) \log(1 - c_k)
$$

(2.4)

where $Y$ represents a binary assignment matrix between our $K$ proposals and the temporal annotations $a$ of a given sequence. A positive link $Y_{kj} = 1$ implies that the $s_k$ proposal tightly matches the annotation $a_j$, thus it is a linked proposal for training.
purposes.

Goal (i) is achieved through equation 2.3 which penalizes linked proposals distant from their corresponding action annotation through the squared euclidean distance. Note that proposals spanning background regions i.e. \( Y_{kj} = 0 \) \( \forall j \) do not contribute to the loss function through this term.

Goals (ii, iii) are enforced with the equation 2.4 which takes the form of a weighted binary cross-entropy function. Thus, the likelihood of linked proposals is encourage to be high, while non-linked proposals should score low. The weights \( w_0, w_1 \) are hyper-parameters calculated according to the frequency of positive and negative proposals in the training set. Finally, \( \alpha \) balances the contribution of the terms in the loss function. We optimize equation 2.2 using stochastic gradient descend, as our architecture and loss function are differentiable almost everywhere.2

In this work we consider assignment matrices where each proposal \( s_k \) is at most linked to the best matching action annotation, \( a_{j^*} \), based on their temporal intersection over union (tIoU); and the link is also a function of the tIoU(\( s_k, a_{j^*} \)). Formally, \( Y_{kj^*} = 1(\text{tIoU}(s_k, a_{j^*}) \geq \beta) \) for \( j^* = \arg\max \text{tIoU}(s_k, a) \). We set \( \beta = 0.5 \) using cross-validation.

**Anchor proposals.** Summarize the statistics of common duration and location of actions that usually occur inside a segment of length \( L \). We found that using anchor proposals, \( \hat{s} \), as proxies for the outputs of the localization module was beneficial to improve the stability during training. Intuitively, anchors constraint the permutations along the rows of \( Y \) after each forward-pass hence they reduce the noise from alternating assignment matrices.

More specifically, we incorporated the anchor proposals into our formulation by: (i) interpreting the outputs of the localization module, \( s \), as deformations for each anchor, \( \hat{s}_k = \{ \hat{s}_k \}_{k=1}^K \), rather than absolute locations; (ii) computing the assignment
matrix, $Y$, between anchors and (absolute) action locations $\hat{s}, a$; (iii) computing the ground-truth deformations for each $\hat{s}_k$ with respect to the best matching action annotation $a_{j^*}$; and (iv) using ground-truth deformations as opposed to (absolute) action locations for $a$ in equation 2.2. In practice, we obtain the location and duration of each anchor proposal by clustering the ground-truth annotations with $K$-means which gives rise to a diverse set of anchors inside $X$.

**Implementation Details.** We use the public implementation of C3D network [78] as the backbone of our visual encoder module. We employ the weights pre-trained in Sports-1M [82], and extracted the activations from the second fully-connected layer, $fc7$. In contrast with [27], we did not fine-tune the visual encoder module for the action proposal tasks to reduce training time. By cross-validation, we found that employing a single layer in our sequence encoder and $d_t = 256$ achieve a good trade-off between accuracy and run-time. We found that ADAGRAD, an accelerated stochastic gradient descent method with adaptive rate of change for each parameter, improves the training dynamic. We employ a learning rate of $10^{-4}$, and $\alpha = 1.0$ to balance the contribution of $L_{\text{match}}$ and $L_{\text{conf}}$. The normalize the location of the center and duration of the anchors, $\hat{s}$, and the annotations, $a$, with respect to $L$.

We build $X$ by sampling sequences of length $L$ frames from a long untrimmed video dataset with temporal annotations. For the THUMOS-14 dataset, which compresses around 11 hours of video and more than 3000 annotations, we generated a large corpus instances with multiple actions and more than 500 thousands instances. In practice, we extract sequences $X$ densely for all the training videos and cluster them according to maximum $tIoU$ between the annotations $a$ and the priors $\hat{s}$. We sample sequences from each cluster, to ensure that they are equally represented.

Similar to [28, 27], our action proposals are learned in a category-agnostic fashion which it is of special interest for scalable temporal localization of actions.
2.4 Experimental Analysis

We validate the quality of our approach on labeled untrimmed videos from the challenging THUMOS-14 benchmark, which contains over 24 hours of video from about 20 sport action categories. This part comprises 413 videos divided into 200 validation videos and 213 test videos. We train our DAPs model using 180 out of 200 videos from the validation set and hold out 20 videos for validation. We report results on the 213 test videos with temporal annotations. To study the generalization capability of our model across datasets, we also test on the validation set of the ActivityNet benchmark (release 1.2) [57], which comprises 76 hours of video and 100 action classes. No fine tuning is done on this benchmark.

Metrics. Inspired by [60], we assess the quality of our temporal proposals with the same metrics adapted to the video domain. Specifically, we use Average Recall (AR) to measure the temporal proposal quality for a limited number of proposals. We compute AR for a $tIOU$ between 0.5 to 1, as a function of the number of proposals. We expect the best proposal approach to achieve the best recall by generating tight temporal proposals at a fixed number of proposals. We also measure the recall at a fixed number of proposals, as a function of $tIoU$. This metric measures the localization quality of temporal proposals. We consider 1000 proposals for this.

In Section 2.4.2, we investigate the impact of applying action proposals in the context of action detection. Following the standard evaluation protocol, we measure the mean Average Precision (mAP) at 50% $tIoU$. We use the official toolkit provided by THUMOS-14 [56].

2.4.1 Recall Analysis

In this section, we analyze recall performance of our method. Specifically, we (i) showcase an ablation study for multiple variants of our approach, (ii) compare the
Figure 2.4: **Impact of DAPs’ hyperparameters.** We study the impact of the number of proposals $K$ per segment, and the length of the segment $L$ in a held-out portion of the validation set of THUMOS-14. We find that the performance of our model is stable with respect to the number of proposals (top). On the other hand, we find that the choice of the sequence length $L$ is more critical (bottom).

performance of our architecture with respect to previous and concurrent temporal proposal methods, and (iii) study the ability of our approach to generalize to action categories unseen during training.

**Variants of our approach.** We study the impact of two hyperparameters of our DAPs architecture on 20 videos from the THUMOS-14 validation set. Figure 2.4 plots AR (first and third columns) and Recall at 1000 proposals (second and fourth columns) of our algorithm for different numbers of proposals per stream ($K$) and four
different segment lengths ($L$).

As Figure 2.4 shows (two leftmost columns), our model is not very sensitive to the number of anchor proposals $K$ for a segment length $T = 512$ frames. Our experiments show that larger $K$ does not necessarily translate into better performance. We hypothesize that this behavior is a result of using $K$-means to select the anchors. This result suggests that the difference between selecting multiple anchors per segment might not be predictable, so we resort to choose this hyper-parameter by cross-validation. We choose $K = 64$ for the rest of our experiments, as a reasonable tradeoff between capacity and AR. In fact, our DAPs model with $K = 64$ achieves the highest average recall rate for more than 100 proposals and about 100% recall at a 50% $tIoU$ with 1000 proposals, as shown in Figure 2.4 (second columns).

Next, we assessed the impact of the length of the sequence $L$ on the performance of our architecture. We evaluated with $L \in \{160, 256, 512, 1024\}$ frames which covers $\{75, 92, 98, 99\}$% of the annotations in the validation set respectively. The results suggest that $L$ is a crucial hyperparameter for achieving high recall. From Figure 2.4 (bottom), we found that for $tIoU$ of 50% at 1000 proposals the recall correlates with statistics of annotations. Therefore, we conclude that our model learns correctly to retrieve actions inside the range of $L$. Based on this analysis, we chose a value of $L = 512$ frames for other experiments.

In the experiments to follow, we report the results of our DAPs algorithm with $K = 64, L = 512$ which offers a good trade-off between accuracy, scalability, and run-time performance.

**Comparison with other approaches.** We compare the performance of our algorithm against contemporaneous approaches designed to retrieve temporal proposals, namely *Sparse-prop* [28], *BoFrag* [58], and *SCNN-prop* [27]. For completeness, we also compare to a representative spatio-temporal proposal method, *APT* [69]. For
Figure 2.5: Comparison against other action proposal approaches. Our DAPs network achieves better results than most of the previous temporal and spatio-temporal approaches on THUMOS-14 in terms of Average Recall as well as in terms of recall of 1000 proposals for a wide range of tIoU. This result evidences the importance of effectively encoding the visual sequence as a discriminative sequence of states. In comparison with the contemporary SCNN-prop work, our network achieves competitive performance without exploring multiple window scales nor fine-tuning the visual encoder.

APT, we generate its temporal predictions by retaining the temporal dimension from its spatio-temporal tubelets. We obtain APT results by running the public implementation provided by the authors. For all other methods, temporal proposals were kindly provided by the authors.

Figure 2.5 illustrates the AR and recall of 1000 proposals in the THUMOS-14 benchmark. Notably our DAPs network outperforms all other methods in both metrics (except for SCNN-prop). We hypothesize that it improves upon them by effectively encoding the sequence of visual codes as a discriminative set of states from where it is plausible to regress proposals with multiple durations. Notably, our approach achieves a better performance without exhaustively exploring multiple temporal window sizes which suggests that our network is effectively encoding multiple action durations instead of sticking to a fixed length. It is worth to mention that the AR of DAPs with 1.6k proposals is better than or comparable to APT and Sparse-Prop results with an order of magnitude more of predictions. In contrast with the
contemporary SCNN-prop network, our network achieves encouraging results without exploring 5 additional scales or finetuning the visual encoder in the THUMOS-14 dataset. Figure 2.5 (right) shows that our approach also produces segments tightly covering the annotations up to 85% tIoU. From this point onwards, DAPs and SCNN-prop methods exhibit a greater decreasing slope than the other algorithms. This effect could be partly created by the use of tIoU to define the supervisory signals.

On the other hand, we found that all supervised methods outperform the unsupervised ones (BoFrag [58] and APT [69]) by a considerable margin, especially at high tIoU values. This suggests that supervised methods are not over-fitting on their training set, and that their “actioness function” is generalizing to unseen videos with similar characteristics. We believe that developing the concept of “actioness score” for unsupervised approaches may help to boost the ranking of their proposals.

**Is the network able to generalize the concept of an action?** Proposal approaches are similar to classifier cascades in the sense that they reduce the computational cost of evaluating powerful classifiers on regions that can be “easily” rejected [47]. According to Hosang et al. [60], the main difference between these two methods is that classifier cascades are not purposely designed to generalize beyond the categories they are trained on. Along these lines, we study the generalization capabilities of our DAPs network to validate that it is a proper proposal approach. We do that by applying our model, trained on 20 sports categories from THUMOS-14, on ActivityNet, a richer and more diverse dataset in terms of action categories. For example, just nine actions from THUMOS-14 have a reasonable correspondence with the hundred activity categories in ActivityNet. Moreover, this dataset includes many categories unrelated with sports, such as Preparing pasta, Playing saxophone, Shoveling snow, to name a few.

Figure 2.6 (left) quantitatively summarizes the generalization capability of our approach. We show average recall results of our method on four datasets: ActivityNet...
Figure 2.6: **Generalization of DAPs’ proposals.** We study the generalization power of our DAPs network for unseen actions by evaluating its performance on ActivityNet. Interestingly, the AR performance of our network does not decrease significantly, at 600 proposals, on videos where action durations comes from a similar distribution, ActivityNet ≤ 1024 frames line. This suggests that discriminative sequence of states learned by our model capture common patterns that allows it to localize and score segments of unseen actions. On the right, we appreciate segments retrieved by our method focus on *brushing teeth* and *shoveling snow* actions, clearly not related with any sport.

(all 100 categories), ActivityNet ∩ THUMOS-14 (on 9 categories shared between both benchmarks), ActivityNet ≤ 512 frames (videos of unseen categories with annotations up to 1024 frames), and THUMOS-14. By comparing the performance on ActivityNet and THUMOS-14, the generalization power of DAPs might not seem encouraging. However, we find that 42% of the activity annotations in ActivityNet span more than 1024 frames (i.e. twice the size of our temporal sequences L), hence it will be difficult for our model to achieve a high AR in this scenario. Since the distribution of activity durations in ActivityNet is very different to the one in THUMOS-14, a drop in recall performance is not surprising. In fact, Hosang et al. make a very similar observation in the context of generalizing 2D object proposals in images across datasets [60].

Following up on this observation, we study the performance of our approach on ActivityNet ∩ THUMOS-14, where we only consider annotations from common classes seen in training; and ActivityNet ≤ 512 frames, where we only consider annotations of unseen classes that have similar duration statistics observed in THUMOS-14, i.e. an-
notations that span up to 512 frames. When evaluating on these two datasets, DAPs performance is quite similar in both cases, especially when more proposals are retrieved. In fact, it achieves an AR of 50.9% and 41.9% for 600 proposals respectively, which are close to our performance on THUMOS-14 for the same number of proposals. This suggests that DAPs does exhibit a desired level of generalization for unseen actions. Note that, ActivityNet videos are 50% shorter than THUMOS-14 videos on average so it is natural that our method produces less number of proposals.

Figure 2.6 (right) shows qualitative examples of temporal proposals retrieved by our network for activities not related to action categories used in training. We hypothesize that the network can generalize to these activities by discovering common underlying patterns in the encoded visual sequence that helps it to localize a proposal, as well as, score its likelihood.

2.4.2 DAPs for Temporal Action Localization

Inspired by the success of object detection approaches combining object proposal methods with object classifiers, we study the benefit of applying our temporal action proposals in an action localization pipeline. To this end, we classify the action proposals generated by our approach and competing proposal methods using the same state-of-the-art action classifier trained on THUMOS-14 \cite{83}. In this section, we describe the action classifier, assess the impact of the number of proposals on the localization performance, and compare our method against state-of-the-art approaches.

**Action Classifier.** We adopt the recent approach of Xu et al. \cite{83}, which trains a linear classifier from CNN features aggregated with VLAD. To this end, we use the activations from the fc7 layer from a 3D CNN \cite{78} as our features. First, We learn the codebook using K-means with \( K = 256 \). Then, we encode the fc7 features that belong to each temporal segment using VLAD with power and \( L_2 \)-normalization.
Table 2.1: Localization accuracy as a function of the number of proposals on THUMOS-14. We evaluate the performance of different proposal methods using mAP at K number of proposals, mAP@K. Our approach outperforms competing methods by a significant margin for all number of proposals.

<table>
<thead>
<tr>
<th>Method</th>
<th>K=50</th>
<th>K=100</th>
<th>K=200</th>
<th>K=500</th>
<th>K=1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>APT</td>
<td>4.1</td>
<td>5.2</td>
<td>6.2</td>
<td>6.8</td>
<td>6.4</td>
</tr>
<tr>
<td>BoFrag</td>
<td>5.3</td>
<td>6.6</td>
<td>7.0</td>
<td>8.5</td>
<td>8.3</td>
</tr>
<tr>
<td>Sparse-prop</td>
<td>5.7</td>
<td>6.3</td>
<td>7.6</td>
<td>8.2</td>
<td>8.0</td>
</tr>
<tr>
<td>DAPs (Ours)</td>
<td><strong>8.4</strong></td>
<td><strong>12.1</strong></td>
<td><strong>13.9</strong></td>
<td><strong>12.5</strong></td>
<td><strong>12.0</strong></td>
</tr>
</tbody>
</table>

Finally, we train a one-vs-all linear SVM classifier with $C = 100$. At test time, we run our activity classifier over all the generated action proposals and obtain an action confidence score for each of them. We apply non-maximum suppression with a 30% $tIoU$ to eliminate near-duplicate detections. As in common detection procedures, we generate a final prediction score by multiplying the classifier and proposal scores.

**Localization results.** Table 2.1 shows quantitative detection results comparing our proposal approach against competing methods. Following temporal action localization convention, we report the mAP (mean AP) score at 50% $tIoU$. We consistently outperform the competing methods by a significant margin. This reaffirms that our method produces high-quality action proposals within a budget number of proposals.

Interestingly, BoFrag generates good localization results despite its modest recall performance. This suggests that BoFrag is producing proposals with a small number of hard negatives, which allows the activity classifier to keep the number of false positives low. We also observe that all methods tend to saturate after using more than 500 proposals. This is in part due to having an action classifier not-tailored for each of the proposal methods. Therefore, it is plausible to fluster the action classifier when the ratio of true positives starts to decrease.

**State-of-the-art comparison.** Table 2.2 summarizes the results of the state of the art on temporal action localization in THUMOS-14. Our method achieves a signifi-
Table 2.2: Temporal action localization comparison in THUMOS-14. We report the performance of our method at 200 proposals. Our method achieves a competitive results with respect to previous and contemporary approaches.

<table>
<thead>
<tr>
<th>Method</th>
<th>[24]</th>
<th>[28]</th>
<th>[84]</th>
<th>[27]</th>
<th>This work</th>
</tr>
</thead>
<tbody>
<tr>
<td>mAP</td>
<td>2.0</td>
<td>13.5</td>
<td>15.0</td>
<td>19.0</td>
<td>13.9</td>
</tr>
</tbody>
</table>

significantly higher performance than Karaman et al. [24] which uses sliding window with a unique fixed temporal length. We attribute this improvement to the fact that our approach scans the video in a much more efficient way. We obtain a similar performance to Caba Heilbron et al. [28] and Oneata et al. [84]. This result is encouraging given that our action proposal pipeline operates at a much faster rate. In contrast to Shou et al. [27] (SCNN-prop), our results are promising considering that our classification stage is based on VLAD aggregation of CNN features and a SVM classifier. We could expect further improvements by fine tuning a powerful CNN network on this task.

2.4.3 Run-time Performance

By definition, action proposals should reduce the effort of applying an accurate and computationally expensive classifier on a large number of windows in a video. This means that a good action proposal method is expected to achieve a high recall rate in the shortest amount of time possible. Table 2.3 summarizes the run-time performance of different proposal methods. Specifically, we compute the average run-time over all testing videos on a Titan-X GPU and report the time in terms of the average length of videos in THUMOS-14 (3 minutes). The authors of other methods kindly provided the run-time of their approach.

Table 2.3 shows that our algorithm is the fastest method to generate temporal action proposals. This is due to: (i) an effective and efficient window scanning approach; and (ii) the use of hardware acceleration units (GPUs) to speed-up computation. A preliminary comparison with SCNN-prop, which also benefits from GPUs, shows a rel-
Table 2.3: **Runtime comparison.** Our DAPs network is the fastest action proposal method. We report the average time needed to apply DAPs to an average length video from *THUMOS-14* (3 minutes). Methods that we could not benchmark appear with “-”, while N.A. refers to methods that do not require a specific stage, see text for more details.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Feature</th>
<th>Proposal</th>
<th>Total</th>
<th>Speedup</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>APT</td>
<td>2828.5</td>
<td>5120.3</td>
<td>7948.8</td>
<td>1.0</td>
<td>0.68</td>
</tr>
<tr>
<td>BoFrag</td>
<td>90</td>
<td>5.5</td>
<td>95.5</td>
<td>83.23</td>
<td>1.88</td>
</tr>
<tr>
<td>Sparse-prop</td>
<td>191.1</td>
<td>342.5</td>
<td>533.6</td>
<td>14.9</td>
<td>10.2</td>
</tr>
<tr>
<td>SCNN-prop</td>
<td>N.A</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>60</td>
</tr>
<tr>
<td>DAPs (Ours)</td>
<td>N.A</td>
<td><strong>1.34</strong></td>
<td><strong>1.34</strong></td>
<td><strong>5931.9</strong></td>
<td><strong>134.1</strong></td>
</tr>
</tbody>
</table>

ative improvement of 123.5%. Disregarding implementation details that can increase the performance of both approaches, the improvement on speed-up is a consequence of an effective encoding that reduces the exploration of multiple temporal scales on overlapping regions.

### 2.4.4 Qualitative results

Figure 2.7 shows the top ranked proposal retrieved from videos of *THUMOS-14* as well as two sample videos with the best-matched proposals out of 100 segments. We include examples where our method succeeds (True positive proposals) and fails (False positive proposals) to match the ground truth with a $tIoU$ of 50%. We observe that our method can produce tight segments around actions. We detect several failure cases in actions like *Shot put* where either the annotation is ambiguous or is hard to establish the temporal boundaries of the action. Interestingly, our method can retrieve segments semantically relevant around miss-labeled or incomplete actions in *THUMOS-14*. For example, the fourth row in Figure 2.7 shows a proposal that matches an action where a woman attempts to perform *Pole vault* but stops.
Figure 2.7: Qualitative examples of our action proposals in sample videos from THUMOS-14. The first five rows show the top ranked proposal, its nearest ground-truth action and the corresponding mapping to time (seconds). The first three rows show examples where our approach generates tightly segments around action instances. On the other hand, the next two rows correspond to failures modes of our model such as an unlabeled occurrence of an incomplete action (fourth row). The last two rows visualize the best-matched segments retrieved by our approach in two different videos.
2.5 SST: Single Stream Temporal Proposal Generation

Our DAPs architecture posed a new way to address the generation of segments of interest in untrimmed and unsegmented videos in an efficient manner. However, its performance is still slightly lower that the state-of-art approach based on exploring multiple sliding windows [27]. In this Section, we introduce some architectural innovations into our network that close the gap with [27] and improve the efficiency of our work. In particular, we streamline the proposal generation by unrolling our sequence encoder over the entire video, and repurposing the final layer such that segments are predicted every time step. In this manner, the network scans an arbitrary length video in a single pass, without the need of processing overlapping temporal windows separately. These innovations accompanied of an appropriate training scheme result in a much more efficient and effective architecture.

In this section, we introduce a new architecture for Single Stream Temporal (SST) action proposals generation that runs over the video in a single pass, without the use of overlapping temporal sliding windows. We repurpose our sequence encoder and output layers such that varied length action proposals are generated every time step.
Figure 2.9: **SST architecture.** We extract C3D features from the input video stream, with a time resolution $d = 16$ frames for each “time step”. These features are the input to the sequence encoder model which outputs $K$ proposals, at each time step $t$, with a confidence vector $c_t$. The longest proposal is of length $Kd$.

as opposed to every $T$ frames. In addition to the architectural adjustment, we introduce a new training scheme that allows us to streamline the generation of proposals in very long input video sequences, without compromising model performance. We validate our innovations in the THUMOS-14 dataset, showcasing state-of-the-art performance on the proposal generation task while being 2x faster than the fastest action proposal method at that time, DAPs [42].

### 2.5.1 Model

We seek to generate temporal action proposals $Y$ of varied length by scanning an untrimmed and unsegmented videos $X$ in single stream with *no sliding windows*. Figure 2.9 illustrates an schematic of our model. We discuss each of the building blocks of our SST architecture in what follows.
Input. At test time, our model parses an untrimmed video sequence $X = \{x_l\}_{l=1}^L$ of $L$ frames entirely without any sliding window strategy.

Visual encoder. Encapsulate the visual information from a group of frames into a meaningful low dimensional representation. We achieve this by processing non-overlapping clips of $d$ frames with a 3D convolutional network pre-trained for action classification [78], C3D. This convolutional network, CNN, efficiently encodes RGB and motion information over a small temporal resolution, $d = 16$ frames. In this manner, we discretize the input stream into $T = L/d$ non-overlapping time steps and each time step is represented by a C3D activation $v_t = \phi_W(x_i)^{(t+1)d}_{i=td}$, $v_t \in \mathbb{R}^{D}$ and $W$ corresponds to all the parameters of the CNN. In practice, we use the activations from the penultimate layer of C3D, fc7 [78], and reduce its dimensionality to 500 dimensions using PCA to reduce training time.

Sequence encoder. Accumulate the visual information as the video progresses, and it carries over an internal representation relevant to decide if action has taken place at each time step. Our sequence encoder is setup in a multiple input and multiple output fashion, consuming the output from the visual encoder and generating an output at each time step $t$. Given by the following recurrent equation:

$$
\begin{align*}
    r_t &= \sigma(W_{vr}v_t + W_{hr}h_{t-1} + b_r) \\
    u_t &= \sigma(W_{vu}v_t + W_{hu}h_{t-1} + b_u) \\
    m_t &= \tanh(W_{vm}v_t + W_{hm}h_{t-1} + b_m) \\
    h_t &= (1 - u_t) \odot h_{t-1} + u_t \odot m_t
\end{align*}
$$

where $r_t, u_t, m_t, h_t \in \mathbb{R}^{D'}$ correspond to the internal representation of the reset, update, memory, and output vectors, respectively; $\sigma(z) = \frac{1}{1+\exp^{-z}}$ correspond to the sigmoid function; and $\odot$ denotes an element-wise product. The matrices $W_{*s}$ and the bias vectors $b_s$ correspond to the parameters of our sequence encoder.
The recurrent equation 2.5 is called Gated Recurrent Unit (GRU) [85]. In comparison with the LSTM cells, presented in the previous Section (c.f. Equation 2.1), a GRU has less parameters which is beneficial from a runtime perspective. We found that GRU offers a slightly better performance than LSTM over a wide range of hyperparameters, consistent with similar findings in other domains [86]. Therefore, we exploit this module on the rest of this Chapter.

Output. It produces the confidence scores for $K$ candidate action proposals ending at time step $t$. Naturally, a high confidence score means a strong belief that an action ended at a particular time step. As illustrated in Figure 2.9, the output of our architecture at each time step corresponds to several proposals of varied duration. Concretely, we output confidence scores $c_t \in \mathbb{R}^K$ for a set of $K$ proposals $P_t = \{\tau^t_1, \ldots, \tau^t_K\}$ at each time step $t$, where $\tau^t_i = (\tau^{S(t,i)}, td)$ corresponds to the temporal endpoint frames of the proposals. The confidence score is proportional to an affine transformation of the output hidden state from our sequence encoder given by $c_t = \sigma(W_o h_t + b_o)$, where $W_o \in \mathbb{R}^{K \times D'}$. Note that $c_t$ changes over time because it is a function of $h_t$, and the affine transformation $(W_o, b_o)$ is not a time-variant function. These properties ensure that we can effectively unroll our model over long videos without any windowing strategy.

In this Chapter, we use fixed endpoints for our proposal segments at each time step with duration sampled from a linear scale. This corresponds to a set of durations $\{d, 2d, \ldots, Kd\}$ frames. Thus, the set of starting endpoints of our proposal is given by $\{\tau^{S(t,i)}\}_{i=1}^K = \{td - d, td - 2d, \ldots, td - Kd\}$ which can be simply written as $\tau^{S(t,i)} = d(t-i)$. Other parametrizations for the endpoints, $\tau^t_i$, like canonical durations given by non-parametric models such as K-means, or boundary deformations are straightforward to add into our formulation.

After processing a long video with $L$ frames, our architecture outputs a collection
of actions proposals $P = \{P_1, \ldots, P_T\}$. Consistent with the previous Section, we apply similar standard post-processing techniques to select the top proposals, such as thresholding by confidence score and non-maxima suppression.

### 2.5.2 Training

Our training objective is similar to the one described in Section 2.3.2, we seek to generate temporal proposals: (i) with high confidence values when a proposal is matching an action; and (ii) low confidence values when a proposal covers background regions. We rely on supervised learning as well, given that its performance is significantly better than the existent unsupervised objectives.

Let $\mathcal{X} = \{X, a\}_{i=1}^M$ be a training set of $M$ video sequences each with $N_i$ temporal action annotations $a = \{a_j\}_{j=1}^{N_i}$ and length $L_i$; and denote the mapping from $X$ into $P$ proposals with their respective confidence scores $C$ as $(P, C) = g_\theta(X)$; where $g$ is the non-linear function summarizing all the computations of the SST architecture described before, and $\theta = \{W_{**}, w_{**}, b_*\}$ groups all the parameters of our architecture. The following training loss, minimized with respect to the model parameters $\theta$, achieves our goals (i - ii):

$$
\mathcal{L}_\theta = \sum_{(X, a) \in \mathcal{X}} \sum_{t=0}^{L_i} \mathcal{L}_{\text{conf}}(c_t, y_t) \\
y_t \in \{0, 1\}^K, \ y_{tk} = 1[\max_{a_j \in a} \text{IoU}(\tau^t_k, a_j) \geq \beta] \\
\mathcal{L}_{\text{conf}}(c, y) = - \sum_{k=0}^{K} w_{1k} y_k \log c_k + w_{0k} (1 - y_k) \log(1 - c_k)$$

where $y_t$ is a binary $K$ dimensional vector representing if the temporal proposals at time step $t$ are (or not) linked to any temporal action annotation.

Our training loss, Equation 2.6, explicitly simulates testing operation by unrolling over a long sequence of frames, $L_i$, for each video (inner summation). Note that the
Figure 2.10: Training examples are generated densely by extracting temporal segments of length $L_w$ in a sliding window fashion with stride $s$. Our dense sampling of long training instances offers a good trade-off between training speed and simulating the unroll over very long videos.

The term $L_{\text{conf}}$ in Equation 2.8 fulfils our goals (i and ii). It encourages the likelihood of linked (unlinked) proposals to be high (low), or penalizes it in the opposite case. This equation takes the form of a weighted binary cross-entropy function. The weights $w_{0k}, w_{1k}$ are hyperparameters calculated according to the frequency of negative and positive proposals in the training set for each proposal duration $k$.

In this work, we consider that an action proposal $\tau^t_k$ is linked to an action if it has enough overlap with any temporal annotation. In practice, an action proposal is linked, $y_{tk} = 1$, if its best temporal intersection over union w.r.t. all the action annotations is greater than $\beta$, Equation 2.7. Here $y_{tk}$ refers to the $k$-th entry of the binary vector $y_t$. We set $\beta = 0.5$, in agreement with the setup described in Section 2.3.2.

We optimize equation 2.6 using stochastic gradient descend, as our architecture and loss function are differentiable. We add dropout and $\ell_2$ regularization on the learned parameters.

In practice, we unroll Equation 2.6 for a fixed number of frames $L_w$ for all the videos in $\mathcal{X}$ to perform efficient mini-batch training. Briefly, we generate the training sequences by densely sampling segments longer than the temporal proposals we aim
to detect, and padding short sequences if needed. Figure 2.10 illustrates our scheme to generate the training sequences.

We set $L_w \gg Kd$ to achieve a good trade-off between training speed and simulating the operation over long sequences, expected during testing. The (dense) striding parametrized by $s$ serves as a data augmentation technique with two purposes. First, it allows a dense generation of training data from the same video. Second, it allows for each time step in the original video sequence to be considered multiple times with different contexts. For instance, consider time step $t = i$ in Figure 2.10. The visual content and ground-truth observed at time $i$ is part of different training sequences $X_0, \ldots, X_3$. Thus, during training the information at $i$ is analyzed within the context of four different examples $X_0, \ldots, X_3$. This practice encourages the predictions and encoding to be robust to the specific initializations of the hidden state. We set $s = d = 16$, corresponding to the finest stride over non-overlapping groups of frames.

Note that the windowing strategy described before only obeys computational reasons during training. Thus, SST still processes a long video sequence using a single stream at test time.

2.5.3 Experiments

We empirically evaluate the effectiveness of our temporal proposal method for the task of proposal generation.

Dataset. To train and evaluate our model, we use the temporal action localization subset from the THUMOS-14 dataset [56], which contains 20+ hours of video with 200 validation and 213 test untrimmed video sequences. We use the validation videos as our training set, and adopt the experimental settings described in Section 2.4.

We perform an 80%-20% split over the training examples in the dataset to cross-validate the hyperparameters for our model, ensuring that the distribution of activity
Figure 2.11: Quantitative comparison of temporal action proposals. SST improves upon prior work without the need of overlapping sliding windows at test time \textit{i.e.} by processing the input in a single stream. \textit{(left)} SST has higher average recall and requires fewer proposals. \textit{(center)} The difference is notable when average recall is computed over a higher tIOU range (0.7-0.95). \textit{(right)} SST achieves large improvements in Recall at 1000 proposals around tIoU $\approx 0.8$.

classes is approximately the same. For our generalizability analysis, we leverage subsets of unseen classes in the ActivityNet dataset [57].

**Comparisons.** We compare our SST model with Deep Action Proposals (DAPs) [87], the proposal stage of S-CNN (SCNN-prop) [27], BoFrag [58], and Sparse-Prop [28].

**Implementation details.** We generate training data with $L_w = 2048$. We vary the number of recurrent layers and hidden state size, as well as number of proposals $K$. We implement the model and training/validation pipeline using Lasagne/Theano and Caffe, with training executed on GeForce TITAN X (Maxwell) GPUs. We optimize our model parameters with \textit{ADAM} update rule [88], with an initial learning rate of $5 \cdot 10^{-2}$ annealed after every $l = 5$ epochs.

**Temporal Proposal Generation** The task of temporal proposal generation consists of taking an input video and producing a set of temporal intervals that are likely to contain human actions. A successful proposal method should be able to retrieve action intervals with very \textit{high recall} and \textit{high temporal overlap} (tIoU) with the ground-truth temporal segments that correspond to actions. Additionally, it is key
Table 2.4: **Trade-off runtime v.s. accuracy of different temporal action proposals.** Our method offers comparable performance for lower tIoU thresholds, outperforms for higher thresholds, and offers a significant boost in speed for generating proposals.

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall tIoU= 0.6</th>
<th>Recall tIoU= 0.8</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAPs</td>
<td>0.916</td>
<td>0.573</td>
<td>134</td>
</tr>
<tr>
<td>S-CNN-prop</td>
<td><strong>0.938</strong></td>
<td>0.524</td>
<td>60</td>
</tr>
<tr>
<td>SST (This work)</td>
<td>0.920</td>
<td><strong>0.672</strong></td>
<td>308</td>
</tr>
</tbody>
</table>

for the model to have *fast runtime*. We evaluate these three aspects of our method in the following.

First, we consider the ability of our model to retrieve proposals with high recall. This is measured with the Average Recall for a fixed number of proposals. The average corresponds to the mean recall over a range of tIoUs. We plot Average Recall against number of retrieved proposals in Figure 2.11(center) for tIoUs in the range 0.7 - 0.95, and Figure 2.11(left) for tIoUs in the range 0.5 - 1.0 (for consistency with [87]). We observe that our model outperforms all existing state-of-the-art methods for low and high number of proposals. Also, we note that when operating at the high overlap regime in Figure 2.11(center), our model more significantly outperforms prior work.

Second, we consider the ability of the model to retrieve proposals with high tIoU overlap. Figure 2.11(right) plots proposal recall for our method in comparison to prior work. We note that our model performs comparably with competing approaches at the lower tIoU range, but more importantly, our method performs better at the higher tIoU regime. This shows that our proposals tightly align the ground-truth temporal action annotations.

Finally, we study the runtime speed of our method in comparison to alternative approaches in the literature. To achieve this, we measure runtime speed in frames per second (FPS). In comparison to prior work that relies on multi-scale temporal
Figure 2.12: **Recall stability and robustness** of our model against *(left)* Proposal Temporal Position, *(center)* Video Length, and *(right)* Groundtruth Annotation Length. We observe that SST is indeed able to handle long, untrimmed video sequences without compromising performance.

sliding windows [27] or a single scale temporal sliding windows [87] (Section 2.3.1), our single-pass model achieves significantly faster processing speeds, as shown in Table 2.4.

**Robustness to Video Length.** An important goal for our architecture is the ability to handle very long testing video sequences. As outlined above, the idea is that our recurrent model should be able to unroll over the entire duration of the testing video, regardless of its duration, so that the proposals are generated in a single pass through the video. We achieve this by two aspects of our model: dense predictions at each time step, and a data augmentation scheme that encourages robustness into the model with respect to video length.

We analyze the performance of our model to highlight its robustness from three perspectives. For this analysis, we select the operating point of 1000 retrieved proposals. First, we study the recall stability with respect to the temporal location of the proposal middle frame, which we plot in Figure 2.12 (left). Note that our model processes the entire video by unrolling a single recurrent network, so the longer the video, the more time steps the recurrent network processes. We observe that SST recall performance is stable and nearly independent of the temporal location of the proposal.

Second, we study the recall stability with respect to video length. Here, we compute recall per video and compute average recall for videos with similar length. We
Figure 2.13: **Qualitative results of our SST model on THUMOS-14**, time measured in seconds. (a) Proposals generated by SST on a long, untrimmed video sequence. Each groundtruth annotation is coupled with the best proposal on top. We observe that performance of the network is maintained for the full duration. (b-d) Performance of the top-ranked temporal action proposal retrieved for a given input video sequence, coupled with the nearest groundtruth annotation. SST provides tight localization bounds with high tIoU. (e-f) False-positive results for the top-ranked retrievals. In particular, (e) illustrates an issue where the model sometimes rank “umbrella proposals” which encompass several short, consecutive action sequences with a slightly higher confidence than the individually-localized proposals.

Also observe stable behavior with respect to video length as shown in Figure 2.12 (center).

Finally, we analyze recall performance with respect to proposal length. We would like to analyze if longer action sequences are harder to detect than shorter actions. We plot recall against proposal length in Figure 2.12 (right). Again, we observe that recall performance is also stable with respect to length of the ground-truth annotations we wish to localize.
We also note that some videos in the THUMOS testing set are particularly long, with durations of over 20 minutes. This corresponds to video sequences of more than 30-50 thousand frames. Our qualitative evaluation confirms our empirical observation that our network can unroll over very long testing videos without compromising its performance.

**Qualitative Results.** We visualize sample proposals from our model in Figure 2.13. We see that the model localizes the groundtruth annotations well - our top confidence proposals satisfy tIoU criteria well, and our highest overlap proposals have high confidences. We observe that among false positive detections, a common case occurs where our model outputs a high-confidence “umbrella proposal” over short action sequences tightly packed together with brief periods of non-action between them (although the model does output other high-confidence proposals correctly localizing those proposals, just with slightly lower confidence).

## 2.6 SS-TAD: Single Stream Temporal Action Detection

So far we have discussed two efficient approaches to generate temporal segments where it is likely to find a human action in a video. While these approaches are key for the localization problem, users often require to localize the occurrence of particular actions such as “elder person falls down” to trigger particular alarms. In this section, we describe how to quickly parse a video to determine all the temporal bounds of interest as well as the action categories undergoing inside them. This is commonly known as temporal action detection.

Recent progress in this area can be broadly categorized into three types of approaches, as shown in Figure 2.14. The first group [89, 90] performs analysis at the level of individual frames or groups of frames, applies temporal smoothing, and merges into detections. The second group [27, 87] does a first series of passes through the video to generate temporal proposals, applies a classifier to each proposal to ob-
Figure 2.14: **There have been a few dominant approaches for temporal action detection.** (a) frame-level analysis followed by a separate merging step to obtain detections, (b) proposals generation followed by a separate classification over the proposals. (c) We present a new end-to-end model (SS-TAD) that outputs action detections directly from a single-pass over the input video stream.

There have been a few dominant approaches for temporal action detection. (a) frame-level analysis followed by a separate merging step to obtain detections, (b) proposals generation followed by a separate classification over the proposals. (c) We present a new end-to-end model (SS-TAD) that outputs action detections directly from a single-pass over the input video stream.

The trade-off between these two is that the first group tends to offer simpler models that do not require multi-stage training at the cost of detection performance, while the latter approach offers superior performance, but requires separate training of the proposals and classifier stages and may require multiple passes through the data at test time. This motivates approaches in the third group [91]: modular architectures that can offer the end-to-end trainability and efficiency of the first group of approaches, while providing the superior performance of the latter.

Here, we introduce a novel approach within this group that significantly outperforms earlier approaches both in accuracy and efficiency.

In this section, we introduce an end-to-end architecture (SS-TAD) for efficient temporal action detection in untrimmed videos. Our architecture exploits recurrent memory modules trained with semantic constraints resembling the dominant approaches (frame-level classification and proposals), described before. Our model processes the video in a single pass, and directly outputs the temporal bounds and corresponding action classes for the detections. Furthermore, we demonstrate experimentally that our efficient architecture achieves state-of-the-art performance on
Figure 2.15: SS-TAD model architecture. Given an input video stream, we represent each non-overlapping “time step” \( t \) with a visual encoding over \( d \) frames. This visual encoding is the input to two recurrent memory modules that are semantically-constrained to learn proposals and classification-based features. These features are combined before providing the final temporal action detection output.

temporal action detection, while simultaneously providing a high FPS processing speed.

2.6.1 Model

We seek to generate temporal action detections from an input untrimmed video. Figure 2.15 provides an overview of our model and approach. Given an input video sequence \( X = \{x_i\}_{i=1}^L \) with \( L \) frames, our model should provide as output (1) the temporal boundaries and (2) the corresponding action class of any activities contained within. Importantly, the model must disregard irrelevant background information while still retaining relevant action information. In this part, we introduce the technical details of Single Stream Temporal Action Detection (SS-TAD), our new efficient model for end-to-end temporal action detection.

The temporal action detection task is the natural product of two main sub-tasks: (1) temporal action proposals, which provides temporal bounds where non-background actions are occurring, and (2) local action classification, which provides frame or time-step resolution classification. Rather than providing explicit solutions to these sub-tasks as done in prior work, we propose an efficient model design that focuses on
providing the detections directly with information gained by *implicitly* solving these sub-tasks during inference.

Our model consists of three main components: our input visual encoding, our two recurrent memory modules, and our final output which we detail next.

**Visual Encoder.** We capture lower-level spatiotemporal visual information from the input video frames by leveraging a 3D-Convolutional (C3D) network [78]. C3D effectively captures visual and motion information over small time clips with \( d = 16 \) frames, and it constitutes an effective building block in prior work for temporal action detection [27, 43, 87]. Since we process each frame only once, we essentially discretize the input video stream into \( T = L/d \) non-overlapping time steps, similar to the setup described in Section 2.5. At each time step the visual encoder returns a low dimensional representation given by \( v_t = \phi_W(\{x_i, \ldots, x_{i+d-1}\}) \). In practice, we replace the \( fc8 \) layer with a linear layer to reduce the dimensionality of \( v_t \) to 500 dimensions. We initialize this layer with the PCA matrices released publicly by [87].

**Semantically-Constrained Recurrent Memory Modules.** A key component of our model is the introduction of recurrent memory modules, which are semantically constrained during training as a way to induce better training and test-time performance. The purpose of this component is to accumulate evidence as the video sequence progresses relevant to both distinguishing background from action and between classes. Each memory module consists of a multi-layer gated recurrent unit GRU network [85]. Each module takes as input the visual encoder output \( v_t \) and the hidden state representation from the previous time step. The output of each memory module is the sequence of hidden state vectors of the final GRU layer in that corresponding module.

As illustrated in Figure 2.16 a key aspect of the design of our memory modules is that they are *semantically constrained* during the training process. This means that
our loss function encourages the hidden state representation of these modules to be useful to solve intermediate semantically meaningful tasks.

Our final design incorporates two such memory modules, operating in parallel. We combine both output embeddings together before proceeding to generate the final output detections. We describe the intermediate representation and only during training outputs of these module as follows.

**Semantic Constraints: Memory Module (P).** Capture and accumulate relevant information with regard to temporal proposals, which allows the network to discern if the video under consideration contains background or action over several temporal scales.

Let denote by $h_t^{(\text{prop})}$ the hidden state representation of its final hidden layer. This corresponds to the only internal representation held at testing.

During training, we constrain this module such that a vector with confidence scores corresponding for $K$ proposals is generated from the hidden state of the recurrent network. In practice, the vector of proposals confidence is the output from a fully connected layer given by:

$$m_t^{(\text{prop})} = \sigma(W_P h_t^{(\text{prop})} + b_P), \quad m_t^{(\text{prop})} \in [0, 1]^K.$$

We adopt the convention described in Section 2.5, at a given time step $t$ we have right-aligned proposals $P_t = \{\tau_t^k\}_{k=1}^K$ and the endpoints are given by $\tau_t^k = (d(t - k), td)$. Similarly, we followed the same approach to generate the supervision signal, a proposal is linked to an action if it has enough overlap with any temporal annotation otherwise it is associated with background.

We emphasize that these intermediate semantic outputs $m_t^{(\text{prop})}$ are not directly used during inference for the overall detection task.

**Semantic Constraints: Memory Module (C).** While the previous memory module captures the “actionness” of multiple temporal intervals ending at a time step $t$, the classification-focused memory module focuses on retaining an internal representation for precise class encoding of the current time step.
Let denote by $h_t^{(cls)}$ the hidden state representation of its final hidden layer. This corresponds to the only internal representation held at testing.

At training time we enforce that the information in $h_t^{(cls)}$ is relevant to recover the class information at that particular time step. In practice, we output a $(C + 1)$-dimensional vector of class-score confidence from a fully connected layer given by $m_t^{(cls)} = \text{Softmax}(W_C h_t^{(cls)} + b_C)$. Each dimension in $m \in [0, 1]^{C+1}$ denotes the confidence scores for each class of interest and an extra dimension for the background class. Note again that $m_t^{(cls)}$ is not generated during inference, and the state embedding $h_t^{(cls)}$ is what is actually forwarded to the final detection task.

Output Detections. At each time step $t$, we output a $C+1$-dimensional vector of confidence score for each proposal representing the likelihoods that the $k$-th proposal is associated with any of the $C$ classes of interest or background. Formally, we create a joint embedding $h_t^{\text{det}} = h_t^{(prop)} || h_t^{(cls)}$ by concatenating the feature representation outputs from the two memory modules, taken from their final recurrent layer, and compute the confidence score as $d_t = \text{Softmax}(W_O h_t^{\text{det}} + b_O)$, $d_t \in [0, 1]^{K(C+1)}$. We used the same parametrization described in the Memory Module P for the temporal endpoints of our detections $\tau_k^t \in P_t$. The only differences are: (i) the computation of their confidence score $d_t$; and (ii) the ground-truth signal used during training is class-specific rather than class-agnostic.

2.6.2 Training

We seek to generate temporal detections such that their predicted classes match the actual classes of the temporal action annotations; and segments spanning background regions are disregarded easily. In addition to formulate a training loss enforcing these goals, we encourage that the learned representation from the memory modules is relevant for two sub-task (temporal proposals and time-step classification) closely
Figure 2.16: **Semantically-constraint memory modules.** Each recurrent memory module consists of multiple gated-recurrent units (GRUs), and provides the final hidden state encoding $h$ as output. We apply semantic constraints on our recurrent memory modules during training to improve overall detection performance.

related to the temporal detection problem. We rely on supervised learning given that its performance is significantly better than the existent unsupervised objectives.

Let $X = \{X, a\}_{i=1}^{M}$ be a training set of $M$ video sequences each with $N_i$ class-agnostic temporal action annotations $a = \{a_j\}_{j=1}^{N_i}, Y_a \in \{0, 1\}^{N_i \times (C+1)}$; and denote the mapping from $X$ into temporal detections and intermediate outputs as $(P, D, M_C, M_P) = g_\theta(X)$; where $g$ is the non-linear function summarizing all the computations of our SS-TAD architecture; $P = \{P_i\}_{i=1}^{T_i=L/d}$ corresponds to the set of detections over all time steps, similarly $D, M_C, M_P$ correspond to the set of confidence scores of the detections and the intermediate representations, respectively; and $\theta = \{W_s, b_s\}$ groups all the parameters of our architecture.

Our loss function is composed by multiple terms. We describe each of them next, and present the overall loss afterwards.

**Detection Loss.** Our main detection loss $L_{det}$ encourages that the class with the highest confidence score of a given proposal matches the actual class of its corresponding temporal action annotation. Thus, it is applied to the final detections $d_t$
output at each time step $t$. This loss is form by two terms:

$$\mathcal{L}_{\text{det}}(d_t, y_t^{\text{det}}, o_t) = \mathcal{L}_{\text{detcls}}(d_t, y_t^{\text{det}}) + \lambda_{\text{detloc}} \mathcal{L}_{\text{detloc}}(d_t, y_t^{\text{det}}, o_t)$$  \hspace{1cm} (2.9)

$$\mathcal{L}_{\text{detcls}}(d, y) = -\sum_{k=1}^{K} y_k^T \log d_k$$  \hspace{1cm} (2.10)

$$\mathcal{L}_{\text{detloc}}(d, y, o) = \frac{1}{2} \sum_{k=1}^{K} \left( \frac{(y_k^T d_k)^2}{o_k^a} - 1 \right) \mathbb{1}[y_k^T o_k > 0]$$  \hspace{1cm} (2.11)

The first term, Equation (2.10) penalizes miss-classification errors for each temporal detection $\tau_k^t \in P_t$. It takes the form of a multi-class cross-entropy loss between the confidence score $d_k \in [0, 1]^{C+1}$ and its corresponding ground-truth $y_k \in \{0, 1\}^{C+1}$ of the $k$-th detection.

Equation (2.11) encourages detections with tighter localization bounds to be more accurate. Similar to [27], we make the optimal confidence score associated with the correct class a direct function of the overlap $o_k \in [0, 1]^{C+1}$ between ground-truth temporal annotation and the output temporal detection. Note that this term only considers candidate detections associated with ground-truth actions and disregard those associated with the background class, $\mathbb{1}[y_k^T o_k > 0]$.

**Semantic Constraints: Loss Functions.** We encourage a better state representations of action context in our recurrent modules through the loss function. We define a loss function $\mathcal{L}_{\text{prop}}, \mathcal{L}_{\text{cls}}$ over the only during training outputs of memory modules P and C, respectively.

During training, we constrain the proposals-focused memory module according to a multi-label loss function on the semantic output $m_t^{\text{(prop)}}$. We enforce that the confidence scores of $K$ proposals ending at that time step mimic ground-truth “actionness” score i.e. whether each proposal spans any action or a background region.
The loss is given by:

\[ L_{\text{prop}}(m^{(\text{prop})}, y^{(\text{prop})}) = - \sum_{k=1}^{K} w_{1k} y_{k}^{(\text{prop})} \log m_{k}^{(\text{prop})} + w_{0k} (1 - y_{k}^{(\text{prop})}) \log (1 - m_{k}^{(\text{prop})}) \] (2.12)

which corresponds to a weighted binary cross-entropy loss between \( m^{(\text{prop})} \in [0, 1]^K \) and \( y^{(\text{prop})} \in 0, 1^K \) at each time step \( t \). The weights \( w_{0k}, w_{1k} \) are calculated proportional to the frequency of positive and negative proposals in the training set for each proposal duration \( k \).

Similarly, we encourage the classification-focused memory module \( m^{(\text{cls})}_t \) to recover the ground-truth action class occurring at each time step through:

\[ L_{\text{cls}}(m^{(\text{cls})}, y^{(\text{cls})}) = - \sum_c w_c y_c^{(\text{cls})} \log m_c^{(\text{cls})}, \] (2.13)

which corresponds to a multi-class weighted cross-entropy loss between the classification vector \( m^{(\text{cls})} \in [0, 1]^{C+1} \) and the ground-truth class vector \( y^{(\text{cls})} \in \{0, 1\}^{C+1} \). \( w_c \) takes into account the imbalance of the number of instances w.r.t. the frequency of the background class in the training set.

Our overall training loss corresponds to a weighted sum of the previous losses over all training examples

\[ \mathcal{L}_\theta = \sum_{(X,y) \in \mathcal{X}} \sum_t (\lambda_{\text{prop}} \mathcal{L}_{\text{prop}} + \lambda_{\text{cls}} \mathcal{L}_{\text{cls}} + \lambda_{\text{det}} \mathcal{L}_{\text{det}}) \] (2.14)

where \( \lambda_* \) are weighting parameters to normalize the contribution of each loss component to the overall training loss. We found that dynamically adjusting the terms \( \lambda_{\text{prop}}, \lambda_{\text{cls}} \) improves training dynamics and testing performance. In particular an effective schedule gives more importance to the semantic constraints w.r.t. the detection loss earlier in training, but focus more on the detection loss in the later
stages. In this manner, we effectively train the network with a curriculum learning approach by enforcing semantic constraints on easier sub-tasks before relaxing them to focus training on the main task, detection. We present experimental results of this finding in the following section.

Our end-to-end architecture and corresponding loss functions are fully differentiable, enabling training with backpropagation. Note that our single-pass design for the model implies that the recurrent modules must be robust when we unroll them over long input sequences at test time. We encourage this robustness by adopting a data augmentation mechanism during training as described in Section 2.5.2. Briefly, this approach involves dense sampling of overlapping long training window segments.

2.6.3 Experiments

We empirically evaluate the effectiveness of our unified end-to-end architecture for the task of temporal action detection.

**Dataset.** To evaluate our model against prior work, we use the temporal action localization subset of the THUMOS-14 dataset [56]. This subset with 22+ hours of video consists of a validation set with 200 and test set of 213 long, untrimmed videos annotated with the temporal intervals that depict human actions. As is standard practice, we leverage the “validation set” as our training data, performing an 80-20 split for hyperparameter optimization. To enable direct comparisons with prior work, we adopt their experimental settings [90, 43].

**Implementation details.** We implement our model using PyTorch, with training executed on Titan X Maxwell GPUs. We initialize the conv1 through fc7 layers of the visual encoder with pretrained weights released by the authors of [78] from the Sports1M dataset. For training efficiency, we fix layers prior to fc8. We vary the number of stacked recurrent layers and hidden state size in our memory modules,
as well as the number of temporal intervals \( K \) considered at each time step. We optimize our model parameters with the ADAM update rule and a learning rate of 0.001. For our best performing model, we relax the semantic constraints by a factor \((d_{sc} = 0.5)\) every 5K batch iterations for MM-P \((\lambda_{prop})\) and every 8.5K for MM-C \((\lambda_{cls})\), with a batch size of 128 training instances.

**Performance.** We summarize the results of applying our model to the task of temporal action detection on THUMOS-14 in Table 2.5. We observe that our model provides state-of-the-art performance for this task for both low and high overlap thresholds, which indicates that the output detections are more precise than prior state-of-the-art approaches.

**Relevance of semantically-constrained modules.** We verify the efficacy of our semantically-constrained memory modules by performing an ablation study on our architecture. We examine three variants of the architecture: (1) semantic constraints with relaxation during training (our full model), (2) semantic constraints with no relaxation, and (3) no semantic constraints. Other hyperparameters are fixed for fair comparison. As we see in Figure 2.17 adding semantic constraints reduces the number of training epochs needed to reach a particular performance level. However,

<table>
<thead>
<tr>
<th>Category</th>
<th>24</th>
<th>Ours</th>
<th>Category</th>
<th>24</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseball Pitch</td>
<td>0.149</td>
<td>0.193</td>
<td>Hammer Throw</td>
<td>0.191</td>
<td>0.416</td>
</tr>
<tr>
<td>Basketball Dunk</td>
<td>0.201</td>
<td>0.385</td>
<td>High Jump</td>
<td>0.2</td>
<td>0.22</td>
</tr>
<tr>
<td>Billiards</td>
<td>0.076</td>
<td>0.046</td>
<td>Javelin Throw</td>
<td>0.182</td>
<td>0.52</td>
</tr>
<tr>
<td>Clean and Jerk</td>
<td>0.248</td>
<td>0.541</td>
<td>Long Jump</td>
<td>0.348</td>
<td>0.717</td>
</tr>
<tr>
<td>Cliff Diving</td>
<td>0.275</td>
<td>0.639</td>
<td>Pole Vault</td>
<td>0.321</td>
<td>0.489</td>
</tr>
<tr>
<td>Cricket Bowling</td>
<td>0.157</td>
<td>0.151</td>
<td>Shotput</td>
<td>0.121</td>
<td>0.16</td>
</tr>
<tr>
<td>Cricket Shot</td>
<td>0.138</td>
<td>0.103</td>
<td>Soccer Penalty</td>
<td>0.193</td>
<td>0.264</td>
</tr>
<tr>
<td>Diving</td>
<td>0.176</td>
<td>0.269</td>
<td>Tennis Swing</td>
<td>0.193</td>
<td>0.123</td>
</tr>
<tr>
<td>Frisbee Catch</td>
<td>0.153</td>
<td>0.22</td>
<td>Throw Discus</td>
<td>0.244</td>
<td>0.074</td>
</tr>
<tr>
<td>Golf Swing</td>
<td>0.182</td>
<td>0.205</td>
<td>Volleyball Spiking</td>
<td>0.046</td>
<td>0.108</td>
</tr>
</tbody>
</table>

Table 2.5: Temporal action detection results on THUMOS-14. SS-TAD achieves state-of-the-art performance (mAP) on action detection at different temporal overlap thresholds, while still maintaining an efficient single-pass, end-to-end design. ("-" denotes not reported).
Ablation analysis of semantically-constrained (S.C.) memory modules in SS-TAD. We plot relative mAP compared to our best performing model against number of training epochs. Our best performing model leverages both semantically-constrained memory modules while also relaxing constraints during training. See text for details.

if we enforce these throughout training, the overall performance of the model suffers after convergence, compared against the model trained with no semantic constraints. Overall, our full model strikes a balance between the two: by enforcing semantic constraints on the memory modules early, the model learns to tackle the overall detection task quicker, and by relaxing the semantic constraints gradually the model converges to a higher performance value.

Runtime. Since our model has a single-stream, end-to-end design, it is also very efficient. We benchmark our model on a GeForce GTX Titan X (Maxwell) GPU, and find that it operates at 701 FPS. In other words, it can process the input frame data from a 60 minute, 30 FPS video in \( \sim 2.5 \) minutes. For comparison, contemporary approaches [90, 43] in Table 2.5 have significantly lower FPS for the temporal action detection task, between 60 - 500 FPS. Both methods are bounded by their independent proposal or classification stage [27], respectively. This highlights the benefits of our end-to-end monolithic design. Summing up, SS-TAD offers a significant boost in both performance and processing speed.
Qualitative Results. In Figure 2.18(a), we provide example positive detections from our model on action classes such as BasketballDunk, Diving, FrisbeeCatch, and BaseballPitch (Fig. 2.18a)i-iv, respectively). We observe that our model is able to provide strong temporal localization of actions in videos overall, while still maintaining a single-stream, end-to-end design. Importantly, SS-TAD is able to provide localizations over long, untrimmed video sequences. Additionally, in Fig. 2.18a)iii-iv, we see that the model is able to correctly localize the actions across different camera angles and temporal distortions (e.g. slow-motion editing). Furthermore, as we observe in Figure 2.18(c), SS-TAD is able to provide localization and correct classification of different actions occurring in the same video. However, there remain challenging cases for the model. Videos with high frequencies of camera cuts, shots over oblique angles during the action frame, or strong temporal editing lead to weaker localization or misclassification. Two of such examples are shown in Figure 2.18b).

2.7 Summary

We explore the limits of efficient and effective temporal localization of actions in a single untrimmed video. We present Deep Action Proposals (DAPs) the first architecture that effectively generates temporal segments where it is likely to find human actions in a single-pass over a long video sequence. Rather than exhaustively exploring candidate actions over multiple scales, DAPs predicts proposals after encoding a single scale which corresponds to a more efficient generation scheme. By introducing some architectural enhancements, without refraining to our core principles, SST (2nd version of DAPs) achieves even faster runtime and better localization results than the state-of-the-art. We showcase that temporal action proposals paired with an action classifier yields good temporal detection of actions in untrimmed videos. However, the overall efficiency for the downstream task is compromised as results from two independent stages. Thus, we presented SS-TAD, a novel monolithic architecture for
Figure 2.18: **Qualitative results of SS-TAD** on THUMOS-14, including both (a,c) positive and (b) negative detections. Detections are marked as positive above if the temporal overlap (tIoU) with groundtruth is greater than 0.5 and the correct action label is provided.

temporal action detection. By exploiting the compositionality of the task in relation to action proposals and “per-frame” classification, we develop an end-to-end solution with a much faster runtime than multi-stage approaches and better accuracy. We expect that our insights boost the development of new applications to efficiently sift over unsegmented videos. In the next Chapter, we extend the temporal localization problem from a single video to multiple videos, and leverage natural language to not limit the search to a pre-defined set of action categories.
Chapter 3

Temporal Localization of Moments in Video Collections with Natural Language

3.1 Introduction

Consider the natural language query shown in Figure 3.1. Recent work has introduced the task of natural language moment retrieval in video [36, 35], where the goal is to return a relevant moment in an untrimmed, unsegmented single video corresponding to a natural language query. While current methods retrieve moments from a single video, users often have large stores of untrimmed, unsegmented videos that they want to query. In this Chapter, we propose the task of temporally localizing relevant moments in a large corpus of videos given a natural language query. Progress on this task could enable applications in video search and retrieval, such as video editing and surveillance.

Our task is challenging as we need to efficiently and accurately find both the video and the exact moment in the video that aligns with a natural language query. While one could attempt to scale prior approaches for localizing a relevant moment in a single, untrimmed video given a natural language query [95, 36, 35, 96, 97, 98] to a large video corpus, such an attempt would face two difficulties. First, we need the ability to index and efficiently retrieve relevant moments in videos. As current efficient indexing techniques rely on approximating the Euclidean distance between descriptors [99, 100, 101], they cannot be readily plugged into video moment retrieval systems that rely on computing similarities using, often complicated, neural network
Given a natural language query, we seek to find relevant videos from a large corpus of untrimmed, unsegmented videos and temporally localize relevant moments within the returned videos.

Second, the index size needs to scale efficiently relative to the size of the video corpus. While the Moment Context Network (MCN) allows for efficient retrieval due to the model’s use of Euclidean distance for comparing language and video features, it requires indexing and storing all possible-length moments in a video. Such a requirement yields large and non-practical video index sizes. While indexing only action proposals may be a solution to reducing the index size, such methods may discard relevant moments that a user may want to query.

In this work, we propose Clip Alignment with Language (CAL), a model that represents a video moment as a series of short video clips and aligns a natural language query to the moment’s clips with a clip-alignment cost. Our approach is illustrated in Figure 3.2. Our clip-alignment cost compares language and clip features using squared-Euclidean distance, which allows for efficient indexing and retrieval of the video clips. Moreover, aligning language features to short video clips within a video moment allows for finer temporal alignment compared to methods that extract only an aggregate feature from the entire video moment. At query time, we propose a two-stage approach consisting of efficient retrieval followed by more expensive re-ranking to maintain recall accuracy. We achieve efficiency by an approximate strategy that
Figure 3.2: **Our approach.** aligns natural language queries to a sequence of short video clips that compose the candidate moment.

retrieves relevant candidate clips for a language query using efficient approximate nearest neighbour search. Then, for re-ranking, we apply the full clip-alignment cost on all variable-length moments in the temporal proximity of the retrieved candidate clips. Furthermore, representing moments as a series of short video clips allows us to overcome the need for indexing all possible variable-length moments while at the same time retrieving any possible moment in a video.

Contributions of this Chapter

The contributions are twofold: we propose (i) the task of natural language video corpus moment retrieval and (ii) a model (CAL) that aligns video clips to a language query while allowing for efficient retrieval followed by re-ranking in the large-scale video corpus moment retrieval setting. We demonstrate the effectiveness of our approach by extending three datasets to the video corpus retrieval setting: DiDeMo [35], Charades-STA [36], and ActivityNet-captions [103]. We show that our CAL model in an exhaustive setting outperforms MCN [35] on all criteria across all datasets, yielding an 8%-85% and 11%-47% boost for average recall and median rank, respectively. Furthermore, for a corpus of 1M videos, we achieve $5 \times$ faster retrieval and $8 \times$ smaller index size over MCN.
3.2 Related work

Our work lies at the intersection of natural language processing and video, an area that has received much recent attention. Our work is closest to the tasks of, given a natural language query, retrieving short video clips from a large collection and localizing moments in a single untrimmed, unsegmented video. We describe related work for both tasks.

**Video clip retrieval with natural language.** Recently, datasets of short video clips with accompanying natural language have emerged. Examples include the MPII movie description dataset as part of the large scale movie description challenge (LSMDC) dataset [29] and the MSR-VTT dataset [30]. Example recent approaches leverage detected concepts in videos [31], hierarchical alignment and attention [32], learning a mixture of embedding experts [33], and dual deep encoding for zero-example retrieval [34]. However, all of these approaches do not search for moments within untrimmed, unsegmented videos.

**Localizing moments in a single video with natural language.** Datasets of videos with temporally aligned text [36, 35, 96, 103, 104] have been used for aligning movie scripts, textual instructions, and sentences in a paragraph with a single video [105, 106, 107, 108], video object segmentation [109], and retrieving moments in a single video given a text query [95, 110, 36, 35, 96, 97, 98]. Our work is closest to the latter. As we will discuss in Section 3.3, the MCN [35] and CTRL [36] models aggregate features over a video moment before comparing to a feature for the language query. Our clip-based alignment approach allows for finer alignment between the moment and query. More recent approaches have integrated alignment of clips with language queries inside a neural network as part of a temporal modular network [97] or joint alignment with temporal attention [95, 110, 98, 111]. As we
will show, these approaches are not amenable to efficient search and retrieval at large scale. Our approach overcomes both limitations and allows for efficient indexing and retrieval over large video collections.

### 3.3 Clip Alignment with Language (CAL)

Our goal is, given a natural language query $q$, to return a video $v \in \mathcal{V}$ from a corpus $\mathcal{V}$ and temporal endpoints $\tau = (\tau^S, \tau^E)$ that temporally localize the language query in the video where $\tau^S$ and $\tau^E$ are start and end points, respectively. If the video corpus $\mathcal{V}$ comprises a single video, then the task is single video moment retrieval (as proposed in [36, 35]). If it is a collection of videos, the task is video corpus moment retrieval (our proposed task). Our approach for the video corpus moment retrieval task consists of two stages – efficient retrieval followed by more expensive re-ranking. We first describe our Clip Alignment with Language (CAL) model and then describe how it is used for efficient two-stage retrieval with re-ranking (Section 3.3.1).

We cast the temporal localization problem as one of retrieving a sequence of
relevant short clips from a video. Let video $v$ be comprised of a sequence of $N_v$ short uniform-length clips $v = \{c_1, \cdots, c_{N_v}\}$ ordered in time with corresponding temporal endpoints $T(v) = \{\tau_1, \cdots, \tau_{N_v}\}$. Depending on the dataset, the clips may be 3-5 seconds in duration. We seek to return a relevant moment $m_{i,j}^{(v)} \subseteq v$ from a video $v$ consisting of a consecutive sequence of clips $m_{i,j}^{(v)} = \{c_i, \cdots, c_j\}$ for $1 \leq i \leq j \leq N_v$ with moment temporal endpoints $T^{(M)}_{i,j} = (T^{(S)}_i, T^{(E)}_j)$ for $\tau_i, \tau_j \in T(v)$, which closely correspond to the ground truth temporal endpoints $\tau_q$ for input natural language query $q$. Our temporal localization problem can be formulated as an optimization over an alignment cost $C$,

$$\min_{v \in V, i,j} \quad C\left(m_{i,j}^{(v)}, q\right),$$

where we aim to find the best video $v$ and sequence of clips in the video $m_{i,j}^{(v)} = \{c_i, \cdots, c_j\}$ minimizing the alignment cost between the clips and language query $q$. In general, alignment cost $C$ may align two variable-length sequences of features extracted over video clips and words in a sentence. In this work, we consider a special case of alignment by learning to match a single feature extracted over a language query to a sequence of features extracted over video clips. Figure 3.2 illustrates our overall approach.

Let $f^{(v)} = \{f_1^{(v)}, \cdots, f_{N_v}^{(v)}\}$ be a set of features for the clips $\{c_1, \cdots, c_{N_v}\}$ of a video $v$ and $f^{(q)}$ be a feature for the language query $q$. As we may have a variable number of clips in a moment $m_{i,j}^{(v)}$, we define the alignment cost for the moment as the average squared-Euclidean distance between the language feature and the moment’s clip features,

$$C\left(m_{i,j}^{(v)}, q\right) = \frac{1}{Z} \sum_{k=i}^{j} \left\| f_k^{(v)} - f^{(q)} \right\|^2,$$

where $Z = j - i + 1$ is the number of clips in the moment. To prevent the degeneracy of always returning single-clip moments, we enforce that moments have at least two clips, i.e., $i < j$. We used shorter-length clips and observed that this requirement
does not degrade performance in practice.

Our alignment cost has two advantages over previous ones. First, our cost is separable with respect to the video clips, i.e., our cost is expressed as a sum of terms over clips, allowing for finer clip alignment. Second, the video clips are indexable since the terms in the cost are Euclidean distances, which can be computed efficiently. We discuss the advantages of both properties and relate to prior work next.

**Discussion.** In prior work [36, 35], the language feature is compared to an aggregated feature over the video moment,

\[
C_{\text{agg}}(m_{i,j}^{(v)}, q) = \Phi(\Psi(f_i^{(v)}, \ldots, f_j^{(v)}), f^{(q)}), \tag{3.3}
\]

where \(\Psi\) aggregates the clip features \(f_i^{(v)}, \ldots, f_j^{(v)}\) into an embedded feature for the candidate moment and \(\Phi\) compares the aggregated video moment and language features. In MCN [35], squared-Euclidean distance (\(\Phi\)) is used to compare aggregated video moment (\(\Psi\)) and language features. In CTRL [36], aggregated video moment features (\(\Psi\)) and language features pass through a neural network (\(\Phi\)). One drawback of these formulations is that the language feature is compared to an aggregated feature over the entire moment and does not have the ability to align to the individual clips in the moment.

In recent work [95, 110, 36, 97, 98, 111], a joint model over language and video features is used to return the alignment cost,

\[
C_{\text{joint}}(m_{i,j}^{(v)}, q) = \Phi\left(\left( f_i^{(v)}, \ldots, f_j^{(v)} \right), f^{(q)} \right), \tag{3.4}
\]

where \(\Phi\) is a neural network. These neural networks perform early fusion and incorporate an attention mechanism into the model. While these approaches have achieved early success for single video moment retrieval, they currently cannot perform ef-
efficient indexing and retrieval at large scale (e.g., over millions of untrimmed and unsegmented videos) due to their reliance on a neural network for comparing video and language features, i.e., it would be too expensive to compute at test time for a large video corpus.

**Model details.** Figure 3.3 illustrates our model. Clip features \( f^{(v)} \) are computed and compared to language features \( f^{(q)} \) for query \( q \) using squared-Euclidean distance and then averaged. For each clip feature \( f^{(v)}_k \), we concatenate visual features computed over the temporal extent of the clip with a context feature and (optionally) temporal endpoints for the moment, which are then passed through a multilayer perceptron (MLP). As in MCN [35], for context features, we average pool clip features over the entire video. The language feature \( f^{(q)} \) is computed as in MCN [35], where the output of the last hidden layer of an LSTM with word embedding features for each query word as inputs passes through a linear mapping. We use pre-computed features for the visual and word-embedding features, so our model parameters comprise the MLP, LSTM, and hidden-layer linear mapping. Note that our CAL model has the same number of parameters as MCN, which allows for direct comparison of the two approaches.

**Training.** We seek to have our CAL model rank correctly aligned video and language query training examples better than misaligned examples. To achieve this goal, we define a ranking loss for our training objective. Let \( \mathcal{P} = \left\{ \left( m^{(v)}_{i,j}, q \right) \right\}^N_{k=1} \) be a training set of \( N \) aligned video moment and natural language query pairs. For a positive training example \( p \in \mathcal{P} \), we define an intra-video negative set \( \mathcal{N}_{\text{intra}}^{(p)} \) consisting of video moments in the training example video not aligned to the language query training example. Similarly, we define an inter-video negative set \( \mathcal{N}_{\text{inter}}^{(p)} \) consisting of video moments from completely different videos in the training set. We define a set \( \Gamma \) where each member is the triple \( (p, n, n') \in \Gamma \) such that \( p \sim \mathcal{P} \), \( n \sim \mathcal{N}_{\text{intra}}^{(p)} \), and
\( n' \sim N^{(p)}_{\text{inter}} \). We optimize a training loss \( \mathcal{L}_\theta \) for model parameters \( \theta \), where the loss is a sum of ranking losses over intra- and inter-video negatives for all sampled triples,

\[
\mathcal{L}_\theta = \sum_{(p,n,n') \in \Gamma} \mathcal{L}^R(\tilde{C}_p, \tilde{C}_n) + \lambda \mathcal{L}^R(\tilde{C}_p, \tilde{C}_{n'}), \tag{3.5}
\]

where \( \mathcal{L}^R(x, y) = \max(0, x - y + b) \) is a ranking loss, \( \tilde{C}_p = C(m^{(v)}_{i,j}, q) \) is the alignment Cost (3.2) for positive training example \( p = (m^{(v)}_{i,j}, q) \) (similarly \( \tilde{C}_n \) and \( \tilde{C}_{n'} \) for intra- and inter-negative training examples \( n \) and \( n' \), respectively), and \( b \) and \( \lambda \) are margin and weighting hyperparameters, respectively. We set \( b = 0.1 \) and \( \lambda = 0.4 \) using cross validation. We optimize Loss (3.5) using stochastic gradient descent with momentum by uniform sampling over positive and intra-/inter-negative triples.

### 3.3.1 Efficient retrieval with re-ranking

For inference, one can evaluate Cost (3.2) exhaustively over all possible moments in all videos. While this routine was used in MCN [35] to localize moments in a single video, this exhaustive strategy does not efficiently scale to localizing moments in a large video corpus. To achieve efficient retrieval while maintaining recall accuracy, we propose a two-stage approach consisting of an efficient retrieval stage followed by a more expensive re-ranking stage.

Our CAL model allows for efficient indexing and retrieval of video moments for a natural language query since it relies on comparing video and language features with a sum of Euclidean distances. This is important for our application as we may potentially want to search through a large corpus comprising millions of untrimmed, unsegmented videos. As noted in our earlier discussion, approaches that align video and language features with neural networks currently do not extend to large-scale indexing applications, which is a key difference from our approach.

Our strategy for implementing the efficient retrieval stage with our approach is to
Figure 3.4: Our system for indexing and retrieval, based on CALarchitecture, allows for efficient retrieval and storage of video moments in a database. See text for details.

index video clip features \( f^{(v)} \) for each video \( v \). At query time, the system retrieves moments in a greedy fashion by retrieving top clips corresponding to the language feature \( f^{(q)} \) for query \( q \). For the re-ranking stage, we score and re-rank the set of moments containing the retrieved clips with the more expensive Cost (3.2). To boost recall during re-ranking, we re-train our CAL model using the top-retrieved moments from the retrieval stage. This strategy is illustrated in Figure 3.4. While this efficient retrieval with re-ranking strategy is not guaranteed to retrieve the best moment in terms of Cost (3.2), we are able to effectively return the correct moment in practice (c.f., Section 3.4). Moreover, our approach allows for retrieval of any moment from any video, which is in contrast to proposal-based methods [102] that discard clips from videos.

While MCN [35] can also index features corresponding to video moments, our approach offers an advantage with respect to the index size. For our clip-alignment approach, only \( N \) clips are indexed for a video. For MCN, all possible-length moments must be indexed as the model relies on aggregated features over the moments. Assuming maximum moment length of \( K \) clips results in an index of size \( NK - \frac{1}{2}K(K - 1) \).
for a video. For the datasets considered in this Chapter, this results in $2 \times -8 \times$ increase in the index size. This increase is expected to get even worse when longer, more complex moments need to be considered, thus increasing the value of $K$.

### 3.4 Experiments

In this section, we show qualitative and quantitative results on our proposed task of retrieving relevant moments from a large corpus of videos for a natural language query. We start by showing results on our proposed video corpus moment retrieval task in an exhaustive setting (Section 3.4.1). Next, we show results using efficient retrieval with re-ranking (Section 3.4.2). We show additional results in the supplemental.

#### 3.4.1 Video corpus moment retrieval

Our first experiment consists of exhaustively evaluating a method over an entire video corpus. More specifically, given a language query, we evaluate the alignment cost exhaustively over all possible moments in all videos. We describe in detail our evaluation setup and results.

**Datasets.** We evaluate on three datasets that have natural language sentences aligned in time to videos and have been proposed for the single video moment retrieval task: DiDeMo [35], Charades-STA [36], and ActivityNet-captions [103]. These datasets have a large number of temporally aligned natural language sentences with large (open) vocabulary. Moreover, the videos depict general scenes and are not constrained to a specific scene type. DiDeMo consists of unedited video footage from Flickr with sentences aligned to unique moments in the video (i.e., the sentences are referring). There are 10642 videos and 41206 sentences in the dataset and we use the published splits over videos (train–8511, val–1094, test–1037). Note that moment start and end points are aligned to five-second intervals and that the max-
imum annotated video length is 30 seconds. Charades-STA builds on the Charades dataset [112] consisting of unedited videos of humans acting from scripts. There are 6670 videos and 16124 sentences in the dataset and we use the published splits over videos (train–5336, test–1334). The videos are typically longer in length than the ones in DiDeMo and sentences from the scripts are aligned in time and may not be referring. ActivityNet-captions builds on the ActivityNet dataset [113] consisting of YouTube video footage. There are 14926 videos and 71942 sentences in the dataset and we use the published splits over videos (train–10009, val–4917). Videos are typically longer in length than DiDeMo and Charades-STA and may be edited; the sentences may not be referring.

We adapt the DiDeMo, Charades-STA, and ActivityNet-captions datasets used for single video moment retrieval to our video corpus moment retrieval task. Specifically, a method must correctly identify both the video and the moment within the video corresponding to a ground truth natural language query.

**Evaluation criteria.** We adopt the criteria proposed in TALL [36], where average recall at $K$ ($R@K$) is reported over all language queries. We measure recall for a particular language query by determining whether one of the top $K$-scoring retrieved moments sufficiently overlaps with the ground truth annotation (recall will be 0 or 1). A retrieved moment sufficiently overlaps with a ground truth annotation if the ratio of the temporal intersection over union (IoU) exceeds a specified threshold. We average the recall values across all language queries to obtain the average recall at $K$. We report $R@K$ over all retrieved moments from the video corpus for $K \in \{1, 10, 100\}$ and IoU $\in \{0.5, 0.7\}$. In addition, we report the median rank for the correct retrieval. While the annotations are not exhaustive (i.e., a given natural language query may appear in a video but not be annotated), reporting over different values of $K$ allows us to take into account the missing annotations. Finally, note that DiDeMo [35] has
Table 3.1: **Dataset settings and statistics.** Right – oracle upper bound. See text for details.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Clip length</th>
<th>Max moment length</th>
<th>Stride length</th>
<th>Avg. video length</th>
<th>IoU=0.5</th>
<th>IoU=0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiDeMo [35]</td>
<td>2.5 secs.</td>
<td>6 clips</td>
<td>5 secs.</td>
<td>29 secs.</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>Charades-STA [36]</td>
<td>3 secs.</td>
<td>8 clips</td>
<td>3-6 secs.</td>
<td>31 secs.</td>
<td>99.62</td>
<td>88.79</td>
</tr>
<tr>
<td>ActivityNet-captions [103]</td>
<td>5 secs.</td>
<td>26 clips</td>
<td>5-40 secs.</td>
<td>120 secs.</td>
<td>89.26</td>
<td>80.24</td>
</tr>
</tbody>
</table>

We account for the multiple annotations by requiring that a correct detection must overlap with at least two of the human judgements with the specified IoU, which can be satisfied for all sentences in the val and test sets.

**Implementation details.** To obtain candidate moments in a video, we need to specify the clip length, maximum number of clips in a moment, and how frequently to extract clips in a video (temporal stride). Table 3.1 shows the settings for the video clip length, maximum moment length, and temporal stride used for the evaluated datasets. We set the values for each dataset to maximize an oracle detector where a sequence of (non-overlapping) clips are aligned with the ground truth moments, while minimizing computational cost. We set the temporal stride to 5 seconds for all moments in DiDeMo and proportionally to the moment length \( d \) in the other datasets computed as \( 0.3 \times d \) (rounded to the nearest clip boundary) as longer-length moments do not need fine temporal stride. Given the settings in Table 3.1, the number of candidate moments for each dataset are: DiDeMo – 21,777, Charades-STA – 49,465, ActivityNet-captions – 460,265. For approaches, such as MCN, that index all possible moments, these numbers would be the index sizes for the evaluated datasets.

We report the performance of the oracle detector in Table 3.1. While the oracle’s returned endpoints align to clip boundaries and do not have the ability to exactly align to ground truth endpoints, we note that the oracle detector still achieves high performance. Also note that humans may not generally agree on temporal endpoints [114].
For all approaches, we evaluate their alignment cost for every moment in a video and perform non-minimum suppression with temporal IoU threshold chosen empirically for each dataset (DiDeMo – 1.0, Charades-STA – 0.6, ActivityNet-captions – 0.5).

Our model uses ResNet-152 features [4] computed over the video clips. We computed ResNet\_pool5 features over video frames extracted at 5 fps and max-pooled the features over the clips. Empirically, we observed max pooling outperformed average pooling. We used Glove word-embedding features [115] for the words in the language query. For temporal endpoints, we normalized the start and end points relative to the video length as in MCN [35] to obtain temporal endpoint features (TEFs). For stochastic gradient descent, we set momentum to 0.95 and used a schedule of lowering an initial learning rate of 0.05 by a factor of 0.1 every 30 epochs; training stopped at 108 epochs. We formed mini-batches with 128 positive/negative examples. We selected intra-negatives such that their overlap with the ground-truth moment is lower than a given IoU value. For DiDeMo, we used IoU=1 since the ground truth is aligned to five-second intervals; for Charades-STA and ActivityNet-captions we used IoU=0.35. Similarly, inter-negatives were selected from the same temporal location as the ground-truth moment, whenever possible, in another video selected at random from the entire dataset.

**Baselines.** We compare our CAL model to the MCN baseline [35] run exhaustively over all moments in the corpus in addition to chance and moment frequency prior baselines. For chance, we return moments across all videos sampled from a uniform distribution. We compute the moment frequency prior as in Hendricks et al. [35] for each dataset by discretizing the range of video-length-normalized start and end points and histograming the training ground truth moments. We output the probability for each video’s moment; ties across different videos are broken by sampling a uniform distribution. We train the MCN model using the same procedure as for single video
Table 3.2: **Video corpus moment retrieval quantitative results (exhaustive setting)**. We show average recall for top \( K \) retrievals and median retrieval rank (MR, lower is better) on DiDeMo [35], Charades-STA [36], and ActivityNet-captions [103] datasets for different baselines and our model. Top section - IoU=0.5, bottom section - IoU=0.7. More details in text.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( K=1 )</td>
<td>( K=10 )</td>
<td>( K=100 )</td>
</tr>
<tr>
<td></td>
<td>MR ↓</td>
<td>MR ↓</td>
<td>MR ↓</td>
</tr>
<tr>
<td>IoU=0.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chance</td>
<td>0.00 0.10 1.99 4233</td>
<td>0.01 0.09 1.09 6393</td>
<td>0.00 0.02 0.18 46718</td>
</tr>
<tr>
<td>Moment prior</td>
<td>0.02 0.22 2.34 2527</td>
<td>0.02 0.17 1.63 4906</td>
<td>0.01 0.05 0.47 32597</td>
</tr>
<tr>
<td>TEF-only</td>
<td>0.05 0.32 2.58 2426</td>
<td>0.04 0.34 2.87 3809</td>
<td>0.01 0.05 0.70 24447</td>
</tr>
<tr>
<td>MCN</td>
<td>0.36 2.15 12.47 1057</td>
<td>0.08 0.32 2.96 6540</td>
<td>0.02 0.18 1.26 24658</td>
</tr>
<tr>
<td>This work (TEF)</td>
<td>0.74 3.90 16.51 831</td>
<td>0.15 0.75 4.39 5486</td>
<td>0.01 0.21 1.58 16150</td>
</tr>
<tr>
<td>This work (TEF)</td>
<td>( K=1 )</td>
<td>( K=10 )</td>
<td>( K=100 )</td>
</tr>
<tr>
<td></td>
<td>MR ↓</td>
<td>MR ↓</td>
<td>MR ↓</td>
</tr>
<tr>
<td>IoU=0.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chance</td>
<td>0.00 0.02 0.64 13434</td>
<td>0.00 0.03 0.39 17070</td>
<td>0.00 0.01 0.06 136371</td>
</tr>
<tr>
<td>Moment prior</td>
<td>0.02 0.17 1.99 3324</td>
<td>0.01 0.05 0.56 11699</td>
<td>0.00 0.03 0.26 82488</td>
</tr>
<tr>
<td>TEF-only</td>
<td>0.03 0.27 2.12 3209</td>
<td>0.01 0.16 1.57 8737</td>
<td>0.01 0.03 0.39 57919</td>
</tr>
<tr>
<td>MCN</td>
<td>0.28 1.55 9.03 1423</td>
<td>0.04 0.31 1.75 10262</td>
<td>0.01 0.09 0.70 40474</td>
</tr>
<tr>
<td>This work (TEF)</td>
<td>0.58 2.81 12.79 1148</td>
<td>0.06 0.42 2.78 8627</td>
<td>0.01 0.10 0.90 26652</td>
</tr>
<tr>
<td>This work (TEF)</td>
<td>( K=1 )</td>
<td>( K=10 )</td>
<td>( K=100 )</td>
</tr>
<tr>
<td></td>
<td>MR ↓</td>
<td>MR ↓</td>
<td>MR ↓</td>
</tr>
</tbody>
</table>

Results. Quantitative results are shown in Table 3.2. First, we observe that our CAL model without TEF is on par or outperforms MCN across all datasets on all criteria; for some criteria there is greater than twofold increase in accuracy. These results indicate the effectiveness of our approach on visual and language cues alone without temporal endpoints. When we include TEF, accuracy for CAL improves and outperforms all baselines across all datasets, validating the effectiveness of our approach on our newly proposed task. In particular, we obtain an 8%-85% and 11%-47% boost over MCN with TEF for average recall and median rank, respectively. We note that the performance is low for all methods as annotations are not exhaustive and there are many more candidate moments to search over than in the single video retrieval task, illustrating the great difficulty of the video corpus retrieval task.

Qualitative results are shown in Figure 3.5. Notice how we are able to retrieve relevant moments for the different language queries. For example, the queries “the person
is eating a sandwich” and “person drink out of the glass” retrieves well-localized moments depicting people eating or drinking, including single ground truth annotated moments for the queries. The query “person pour sauce on first piece of meat in skillet” shows example failures of our system. While the top retrieval is correct, the other retrievals depict different parts of the language query, such as “sauce”, “meat”, and “pour”, but not the entire query.

**What is the effect of TEF on the datasets?** We analyze the effect of incorporating the temporal endpoint features (TEFs) into the model. For our analysis, we train the MCN model using only the language features and TEF on all the datasets, *i.e.* , the model does not see appearance features from the video. During testing, we run the model exhaustively over all moments in the corpus. Results are shown in Table 3.2 (“TEF-only”). We observe that the TEF-only baseline is competitive and on par or outperforms the moment frequency and chance baselines. Moreover, on the single video moment retrieval task (*c.f.*, Section 3.4.4), we observe that the TEF-only baseline is a competitive baseline for each dataset and outperforms many of the other baselines that use video appearance features. The fact that the TEF-only baseline performs so well indicates that there is a strong bias in the datasets as only knowing the language query and the relative position of the moment in the video can allow for reasonably high accuracy. This fact was also observed in early datasets for visual
Table 3.3: **Efficient retrieval with re-ranking quantitative results.** We show average recall for top $K$ retrievals on DiDeMo [35] and Charades-STA [36] datasets for different baselines and our model. Top section - IoU=0.5, bottom section - IoU=0.7. More details in text.

<table>
<thead>
<tr>
<th>Retrieval stage</th>
<th>Re-ranking stage</th>
<th>DiDeMo [35] (test)</th>
<th>Charades-STA [36] (test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IoU=0.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEE</td>
<td>MCN (TEF)</td>
<td>0.53</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.00</td>
<td>0.67</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.52</td>
<td>1.75</td>
</tr>
<tr>
<td>MCN</td>
<td>MCN (TEF)</td>
<td>0.92</td>
<td>0.19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.83</td>
<td>1.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17.50</td>
<td>4.09</td>
</tr>
<tr>
<td>CAL</td>
<td>MCN (TEF)</td>
<td>0.98</td>
<td>0.23</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.94</td>
<td>1.41</td>
</tr>
<tr>
<td></td>
<td>CAL (TEF)</td>
<td>22.83</td>
<td>5.89</td>
</tr>
<tr>
<td>CAL</td>
<td>CAL (TEF)</td>
<td>1.07</td>
<td>0.30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.45</td>
<td>1.63</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22.60</td>
<td>6.03</td>
</tr>
<tr>
<td></td>
<td>CAL (TEF,re-train)</td>
<td>1.29</td>
<td>0.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.71</td>
<td>1.91</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22.51</td>
<td>6.85</td>
</tr>
<tr>
<td>IoU=0.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEE</td>
<td>MCN (TEF)</td>
<td>0.46</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.64</td>
<td>0.53</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.37</td>
<td>1.63</td>
</tr>
<tr>
<td>MCN</td>
<td>MCN (TEF)</td>
<td>0.64</td>
<td>0.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.67</td>
<td>0.71</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13.12</td>
<td>2.58</td>
</tr>
<tr>
<td>CAL</td>
<td>MCN (TEF)</td>
<td>0.69</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.63</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>CAL (TEF)</td>
<td>17.89</td>
<td>3.78</td>
</tr>
<tr>
<td>CAL</td>
<td>CAL (TEF)</td>
<td>0.72</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.86</td>
<td>1.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17.60</td>
<td>3.71</td>
</tr>
<tr>
<td>CAL</td>
<td>CAL (TEF,re-train)</td>
<td>0.85</td>
<td>0.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.95</td>
<td>1.32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17.73</td>
<td>4.49</td>
</tr>
<tr>
<td>Approx. CAL</td>
<td>CAL (TEF,re-train)</td>
<td>0.80</td>
<td>0.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.95</td>
<td>1.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11.59</td>
<td>2.55</td>
</tr>
</tbody>
</table>

question answering (VQA) [116] and suggests future work to mitigate such dataset bias.

### 3.4.2 Efficient retrieval with re-ranking

For our second experiment, we evaluate the efficient retrieval and re-ranking system described in Section 3.3. In the *retrieval stage*, we retrieve the top 200 moments using a given method. In the *re-ranking stage*, we then re-rank the top-retrieved moments using a given method.

**Evaluation criteria.** Similar to the exhaustive retrieval setting (Section 3.4.1), we report average recall at $K \in\{1, 10, 100\}$ on the DiDeMo and Charades-STA datasets for the video corpus moment retrieval task. Note that we do not report median rank as only the top retrieved moments are considered.

**Baselines and ablations.** We use MCN or CAL for the retrieval stage, followed by MCN or CAL with TEF for the re-ranking stage. We also consider using MEE [33] for the retrieval stage as it performs well on the LSMDC benchmark [29] and outperforms
other recent methods [34] on MSR-VTT [30] (c.f., supplemental). We used the publicly available implementation of MEE to retrieve videos from the corpus and turned off flow, face, and audio features in MEE for fair comparison. We tried MEE pre-trained on LSMDC and MSR-VTT, which performed near chance on our task; retraining MEE on the target datasets performed best. During retrieval with MEE, we maintain comparable number of moments for the re-ranking stage by retrieving the top videos such that there are 200 available moments within the retrieved videos. Finally, we consider the approximate retrieval setting where we retrieve the top 200 clips given a language query and consider moments around the retrieved clips for the re-ranking stage (Approx. CAL).

Re-training the re-ranking stage. For re-training, we take the top retrieved moments from the retrieval stage and sample inter-video negatives from the retrieved moments (instead of over all possible videos). We sample inter-video negatives using an exponential distribution over the moment’s rank from the retrieval stage. Finally, we fine tune the re-ranking model initializing with the original model’s parameters.

Results. We report quantitative results in Table 3.3. Our CAL for retrieval followed by CAL with TEF and re-training for re-ranking performs best across all criteria on Charades-STA and for $K \in \{1, 10\}$ on DiDeMo (we are on par for $K = 100$), demonstrating the effectiveness of our approach for retrieval with re-ranking. Moreover, our two-stage approach outperforms the exhaustive approaches in Table 3.2 for $K \in \{1, 10\}$. Note that CAL for retrieval is on par or outperforms MCN and MEE using the same method for the re-ranking stage across all criteria on both datasets. We also tried retrieving 200 videos with MEE followed by MCN with TEF for re-ranking and found that our approach outperforms this baseline on all criteria for Charades-STA and for $K \in \{1, 10\}$ on DiDeMo (we are on par for $K = 100$). However, note that this baseline has access to significantly (21×–33×) more moments for the re-ranking
Table 3.4: **Runtime and index size.** Top – exhaustive retrieval; bottom – efficient retrieval with re-ranking.

<table>
<thead>
<tr>
<th>Method</th>
<th>Runtime (s)</th>
<th>Index size (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCN</td>
<td>144.7</td>
<td>63.3</td>
</tr>
<tr>
<td>CAL</td>
<td>24.6</td>
<td>7.45</td>
</tr>
<tr>
<td>MCN / MCN (TEF)</td>
<td>144.7 / 0.4</td>
<td>63.3</td>
</tr>
<tr>
<td>CAL / CAL (TEF)</td>
<td>24.6 / 0.3</td>
<td>7.45</td>
</tr>
<tr>
<td>Approx. CAL / CAL (TEF)</td>
<td>1.0 / 0.3</td>
<td>7.45</td>
</tr>
</tbody>
</table>

stage, which aids in improving recall. Finally, for approximate retrieval, we maintain similar recall as our best model for $K \in \{1, 10\}$ on both datasets, demonstrating its effectiveness in an efficient retrieval setting.

**Runtime and index size.** We report runtime and the retrieval index size for a video corpus containing 1M videos each containing 20 clips with max moment length of 14 clips for the different methods in Table 3.4. We observe that CAL is $5 \times$ faster and has $8 \times$ smaller index size than MCN. Finally, Approx. CAL with approximate nearest neighbor search [117] has fastest run-time ($111 \times$ speed up) and smallest index size, demonstrating its efficiency.

### 3.4.3 Feature ablation

In this section, we show quantitative results of different visual and language features on our proposed task of retrieving relevant moments from a large corpus of videos for a natural language query. For this purpose, we evaluate the alignment cost exhaustively over all possible moments in all videos on DiDeMo (val set) (c.f. Section 3.4.1).

**Which video features perform best?** Table 3.5 (top part) shows an ablation over video features. We use our CAL model as the base model with Glove language features and refrain for using temporal endpoint features, to not introduce a confounding factor. We evaluate the VGG features [16], along with ResNet-152 features [4]. We use
Table 3.5: **Video (top) and language (bottom) feature ablation.** DiDeMo (val) results on our CAL model. See text for details.

<table>
<thead>
<tr>
<th>Modality</th>
<th>Feature</th>
<th>IoU=0.5 K=1</th>
<th>K=10</th>
<th>K=100</th>
<th>MR ↓</th>
<th>IoU=0.7 K=1</th>
<th>K=10</th>
<th>K=100</th>
<th>MR ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual</td>
<td>VGG [16]</td>
<td>0.44</td>
<td>2.68</td>
<td>11.28</td>
<td>1356</td>
<td>0.30</td>
<td>1.89</td>
<td>8.62</td>
<td>1944</td>
</tr>
<tr>
<td></td>
<td>ResNet-152</td>
<td><strong>0.71</strong></td>
<td><strong>4.00</strong></td>
<td><strong>16.48</strong></td>
<td><strong>882</strong></td>
<td><strong>0.48</strong></td>
<td><strong>2.92</strong></td>
<td><strong>12.94</strong></td>
<td><strong>1214</strong></td>
</tr>
<tr>
<td>Language</td>
<td>ELMo [119]</td>
<td>0.49</td>
<td>3.03</td>
<td>13.84</td>
<td>1077</td>
<td>0.33</td>
<td>2.13</td>
<td>10.41</td>
<td>1559</td>
</tr>
<tr>
<td></td>
<td>FastText [120]</td>
<td><strong>0.73</strong></td>
<td>3.64</td>
<td>15.44</td>
<td>959</td>
<td><strong>0.57</strong></td>
<td>2.51</td>
<td>12.02</td>
<td>1315</td>
</tr>
<tr>
<td></td>
<td>Glove [115]</td>
<td>0.71</td>
<td><strong>4.00</strong></td>
<td><strong>16.48</strong></td>
<td><strong>882</strong></td>
<td>0.48</td>
<td><strong>2.92</strong></td>
<td><strong>12.94</strong></td>
<td><strong>1214</strong></td>
</tr>
</tbody>
</table>

the same process outlined in Section 3.4.1 to compute the ResNet and VGG feature. Similar to [35], we extracted \(fc7\) features from VGG-16 pre-trained on Imagenet [5].

We observe that ResNet-152 features perform best across all evaluation criteria. The strength of ResNet features for this task is consistent with the finding that ResNet without fine tuning is a good stand-alone base feature for image recognition tasks [118].

**Which language features perform best?** Table 3.5 (bottom part) shows an ablation over language features. We use our CAL model as the base model with Glove language features and refrain for using temporal endpoint features, to not introduce a confounding factor. We evaluate Glove [115], FastText [120], and ELMO [119] word embedding features using their publicly available code. We observe that Glove and FastText word embedding features perform best. Since Glove performs similarly to FastText, we used Glove for all experiments in our work.

### 3.4.4 Single video moment retrieval

This Section shows quantitative results of our CAL architecture on the existing task of retrieving a moment from a single video given a natural language query. These results give more insights about the overall moment retrieval task, and present an exhaustive and unprecedented evaluation relevant to advance this field.
Datasets. We evaluate on three datasets that have natural language sentences aligned in time to videos and have been proposed for the single video moment retrieval task DiDeMo [35], Charades-STA [36], and ActivityNet-captions [103]. They correspond to the same datasets used in Section 3.4. Note the previous work for single video retrieval usually report findings in a single dataset or at most two of them.

Evaluation criteria. We adopt the criteria proposed in [36], where average recall at \( K \) (R@\( K \)) is reported over all language queries. This corresponds to the same criteria presented in Section 3.4 in the context of searching over a single video. Note that DiDeMo [35] uses a hit-or-miss criteria to compare retrieval methods. For completeness, we also report results and compare to prior work on the original criteria proposed by DiDeMo [35] at the end of this section.

Implementation details. We use the same setup described in Section 3.4 to generate the candidate moments, but limit the search of each query to its corresponding video. Table 3.1 summarizes the setup of the video clip length, maximum moment length, and temporal stride used for the evaluated datasets. For all approaches, we evaluate the alignment cost for every moment in a video and suppress moments that are near lower-cost moments, i.e. those that have temporal IoU greater than a given value chosen empirically (DiDeMo – 1.0, Charades-STA – 0.6).

The visual stream was trained on mean-pooled VGG features like [35] (VGG), or with ResNet-152 features [4] (ResNet). Following the protocol outlined in Section 3.4.

Baselines. We compare our CAL model to several baselines: MCN [35], CTRL [36], ACRN [98], TGN [95], TMN [97], Xu et al. [111]. We also compute a moment frequency prior as in Hendricks et al. [35] for each dataset by discretizing the range of video-length-normalized start and end points and histogramming the training ground
Table 3.6: **Single video moment retrieval quantitative results.** We show average recall on DiDeMo [35] and Charades-STA [36] with RGB and temporal endpoint features. See text for more details.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IoU=0.5</td>
<td>IoU=0.7</td>
<td>IoU=0.5</td>
</tr>
<tr>
<td></td>
<td>R@1</td>
<td>R@5</td>
<td>IoU=0.5</td>
</tr>
<tr>
<td>Frequency Prior</td>
<td>26.11</td>
<td>88.51</td>
<td>21.89 70.53</td>
</tr>
<tr>
<td>TEF-only (Glove,TEF)</td>
<td>26.90</td>
<td>87.85</td>
<td>21.65 70.33</td>
</tr>
<tr>
<td>ACRN [35] (VGG,Glove)</td>
<td>28.45</td>
<td>61.43</td>
<td>16.76 41.13</td>
</tr>
<tr>
<td>CTRL [35] (C3D,Skip-thought)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>L2L [35] (C3D,word2vec)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TGN [35] (Inception-v4,Glove)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MCN [35] (VGG,Glove)</td>
<td>27.01</td>
<td>64.55</td>
<td>16.51 46.65</td>
</tr>
<tr>
<td>CAL (VGG,Glove)</td>
<td>31.24</td>
<td>66.83</td>
<td>20.95 48.44</td>
</tr>
<tr>
<td>MCN [35] (ResNet,Glove)</td>
<td>31.26</td>
<td>71.35</td>
<td>21.29 55.82</td>
</tr>
<tr>
<td>CAL (ResNet,Glove)</td>
<td>36.53</td>
<td>71.96</td>
<td>25.58 54.84</td>
</tr>
<tr>
<td>MCN [35] (ResNet,Glove,TEF)</td>
<td>40.85</td>
<td>90.52</td>
<td>30.50 76.94</td>
</tr>
<tr>
<td>CAL (ResNet,Glove,TEF)</td>
<td>39.18</td>
<td>88.63</td>
<td>29.00 75.73</td>
</tr>
</tbody>
</table>

For the DiDeMo dataset, we reached out to the authors to get their raw outputs so we could compare directly on our DiDeMo criteria; we report obtained results in Table 3.6.

**Results.** Table 3.6 shows quantitative results over the three datasets. We separate the results into four categories – frequency-prior and language-only baselines, “non-indexable” baselines, approaches that are “indexable”, and “indexable” approaches that incorporate temporal endpoint features (TEFs). We list the base features that are used for each method. Finally, for MCN and our CAL model, we evaluate over all datasets; no previous work has shown results on all three datasets.

We observe that with VGG features, our CAL model outperforms ACRN and MCN on all criteria. ResNet features further improve performance. Adding TEF further improves performance, but it gives a slight edge towards MCN. The TEF allows the models to leverage the strength of the frequency prior and suggests that further work is needed on improving the joint video-language representation. On Charades-STA, the frequency prior baseline outperforms CTRL on all criteria, while CAL with and without TEF outperforms both baselines. MCN and CAL achieve results on par with [111] without TEF feature or an attention mechanism. Moreover,
CAL aided by TEF outperforms this baseline over all criteria with absolute gap of +3.8-11.8%. Finally, we observe that the TEF-only model outperforms all the vision+language approaches on ActivityNet-captions. As in the other datasets, MCN and CAL offers complementary information to this baseline with a slight edge toward CAL. To sum up, we found that (i) CAL achieves competitive results or outperforms the state-of-the-art for the single video moment retrieval task; (ii) the conclusions drawn from this task are slightly different than those outline by our new corpus moment retrieval task (c.f. Section 3.4.1); thus (iii) it is important to analyze new models accordingly.

What is the effect of TEF on the datasets? As stated before, we observe that the TEF-only baseline is a strong baseline for each dataset. For our analysis, we train the MCN model using only the language features and TEF on all the datasets, i.e. the model does not see appearance features from the video. Results are shown in Table 3.6 (“TEF-only”). The TEF-only baseline gives a significant boost over the moment frequency prior baseline on the Charades-STA dataset, while performing similarly to the moment frequency prior on DiDeMo. Moreover, the TEF-only baseline outperforms many of the other baselines that use video appearance features in the single video moment retrieval task. This confirms our earlier observation regarding the need of further study to diagnose and mitigate this bias in all datasets.

Results with original DiDeMo criteria. We also show quantitative results on the original DiDeMo [35] evaluation criteria for single video moment retrieval in Table 3.7. The Rank@k criterion represents that among the top-k predictions is possible to find one that overlaps with a ground truth moment with an IoU of 1.0. Note that DiDeMo [35] has multiple temporal segments for each sentence corresponding to different human judgements. Instead of consolidating all the segments of a text query into a single temporal segment, [35] uses a consensus strategy that takes into
<table>
<thead>
<tr>
<th>Method</th>
<th>Validation set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mIoU</td>
<td>Rank@1</td>
</tr>
<tr>
<td>TEF-only (Glove, TEF)</td>
<td>25.88</td>
<td>18.94</td>
</tr>
<tr>
<td>TMN [97] (VGG,Glove)</td>
<td>30.14</td>
<td>18.71</td>
</tr>
<tr>
<td>ACRN [98] (VGG,Glove)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TGN [95] (VGG,Glove)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MCN [35] (VGG,Glove)</td>
<td>24.39</td>
<td>12.53</td>
</tr>
<tr>
<td>CAL (VGG,Glove)</td>
<td>28.97</td>
<td>17.05</td>
</tr>
<tr>
<td>MCN [35] (ResNet,Glove)</td>
<td>28.36</td>
<td>16.36</td>
</tr>
<tr>
<td>CAL (ResNet,Glove)</td>
<td>34.68</td>
<td>21.08</td>
</tr>
<tr>
<td>MCN [35] (VGG,Glove,TEF)</td>
<td>35.82</td>
<td>23.59</td>
</tr>
<tr>
<td>CAL (VGG,Glove,TEF)</td>
<td>33.12</td>
<td>21.26</td>
</tr>
<tr>
<td>MCN [35] (ResNet,Glove,TEF)</td>
<td><strong>38.78</strong></td>
<td><strong>25.67</strong></td>
</tr>
<tr>
<td>CAL (ResNet,Glove,TEF)</td>
<td>37.24</td>
<td>24.44</td>
</tr>
</tbody>
</table>

Table 3.7: Single video moment retrieval results on DiDeMo using [35]'s evaluation criterion. mIoU corresponds to the mean intersection over union between the most confident prediction and all the ground-truth temporal segments of a given moment. See text for more details. Note that TGN [95] could not have used [35]'s criterion (It reports R@\{1,5\} IoU=1 instead). We include their numbers for completeness.

account outliers in the annotations as follows,

\[
\text{Rank}(P, A) = \min_{A' \in \binom{A}{3}} \frac{1}{3} \sum_{a \in A'} \text{Rank}(P, a),
\] (3.6)

where \(P\) corresponds to the ordered set of predictions, \(A\) is the set of all annotations associated to a given text query, and \(A'\) is the set of all triads of annotations in \(A\). Similarly, the mIoU criterion measures the tightness of the top-1 prediction with the ground-truth moments in terms of temporal intersection over union,

\[
\text{mIoU}(p, A) = \max_{A' \in \binom{A}{3}} \frac{1}{3} \sum_{a \in A'} \text{IoU}(p, a),
\] (3.7)

where \(p\) corresponds to the top-1 prediction in the ordered set of predictions \(P\). The previous criteria are computed for each text query. Thus the overall performance is the average among all the queries in the subset of interest. In practice, we use the
public evaluation code released by [35].

We observe that our CAL model outperforms MCN and ACRN in the most stringent criteria (mIoU and Rank@1) with VGG and Glove features. Along the same lines of the previous results, ResNet and TEF features further improve performance of MCN and CAL. Again, TEF gives a slight edge towards MCN suggesting an interesting line for future work where clip alignment and TEF features complement further for the task of single video moment retrieval. Interestingly, TEF allows MCN and CAL to leverage the strength of the frequency prior and achieve performance competitive or better than TMN and TGN. This result suggests analyzing prior attention-based models (e.g., TMN and TGN) carefully as the architectures might be attending to the frequency prior of the datasets instead of learning visual-text patterns.

### 3.5 Summary

We have shown a simple yet effective approach for aligning video clips to natural language queries for retrieving moments in untrimmed unsegmented videos. Our approach allows for efficient indexing and retrieval of video moments on our newly proposed task of search through large video collections. We have quantitatively evaluated on three benchmark datasets extended to our task and shown the effectiveness of our approach over prior work on our proposed task in terms of accuracy and search index size. Our work opens up the possibility of effectively searching video at large scale with natural language interfaces.

So far we have formulated schemes for localizing actions and moments of interest over untrimmed unsegmented videos. However some applications, such as robotics scenarios, demand to outline the pixels of the actor executing the action. Thus, next Chapter focuses on the spatiotemporal localization of actions over trimmed or segmented videos such as those generated in Chapters [2][3].
Chapter 4

Spatiotemporal Localization of Actions through Actor-Supervision

4.1 Introduction

The goal of this Chapter is to localize and classify actions like *skateboarding* or *walking with dog* in video by means of its enclosing spatiotemporal tube, as depicted in Figure 4.1. Empowered by action proposals [15, 121, 122], deep learning [39, 40] and carefully labeled datasets containing spatiotemporal annotations [11, 123, 124], progress on this challenging topic has been considerable [41, 125]. However, the dependence on deep learning and spatiotemporal boxes is also hampering further progress, as annotating tubes inside video is tedious, costly and error prone [126]. We strive for action localization without the need for spatiotemporal video supervision.

Others have also considered action localization without spatiotemporal supervision [127, 128, 129]. Recently, Li et al. [129] proposed a deep learning based model for action classification with an attention LSTM. The attention component highlights regions in the video that correspond to high-responses of certain action class labels. Unfortunately, this scheme does not ensure high-localization accuracy as the model may learn to attend only to discriminative parts of the action, such as the legs and the skateboard for the action *skateboarding*, but not the entire actor. [127] and [128] circumvent this issue by relying on human detectors, trained on images, to guide the classification of action proposals inside a multiple instance learning framework. This framework selects the best candidate proposal in the video guided by multiple cues, in
Figure 4.1: **We propose actor-supervision** as a means for weakly-supervised action localization in video space and time. During the learning stage, our method relies on action labels at the video level only.

particular the detected human actors, which is then used to learn an action classifier. These works are shallow and were not designed to exploit the representation learning principle of deep architectures. Our work unifies these alternatives. It infuses the pragmatic and arguably more accurate scheme of localization from detected actors into a novel end-to-end trainable deep architecture.

**Contributions of this Chapter**

We introduce the concept of actor-supervision to exploit the relevance of actors to steer the localization of actions in videos without using spatiotemporal annotations of the training videos. Instead of using the detected actors to select among candidate regions *a posteriori* ([127](#) [128](#)), we exploit the detections to define the candidate proposals *a priori*. We propose a new architecture that learns to determine the action performed by our proposals only from action labels at the video level. Our technical contributions are twofold: (i), we introduce actor proposals; a means to generate candidate tubes that are likely to contain an action and that do not require any action video annotations for training. We derive our proposals from a detector for human and non-human actors, intended for images, combined with Siamese sim-
ilarity matching to account for actor deformations over time. (ii), we introduce an actor-supervised architecture; an end-to-end deep model that determines the actions of our actor proposals without the need of action boxes annotations. It exploits a new differentiable operation, actor pooling, which summarizes the visual information spanned by an actor; and an attention mechanism to distill the information required to perform spatiotemporal action localization in videos. Experiments on four human and non-human action datasets show that our actor proposals and actor attention register an absolute (and relative) improvements up to 8.4% (23.7%) in Recall and 10.4% (27.5%) in mAP, respectively. Taken together, our actor supervision is the state-of-the-art for action localization from video class labels and is even competitive to some box-supervised alternatives.

4.2 Related work

Typical approaches for action localization first generate spatiotemporal action proposals and then classify them with the appropriate action label. We discuss work related to these two aspects of action localization and group them by the amount of supervision needed.

**Action proposals**

*Supervised action proposals* generate box proposals and classify them per action for each individual frame. In addition to video-level class labels, bounding-box ground-truth for each action instance across all the frames is required. In [39, 121], the box proposals come from object proposals ([130, 131]), and a two-stream convolutional network (conv-net) is learned to classify these boxes into action classes. More recently, action boxes are generated by video extensions of modern object detectors ([132, 133]), as in [134, 135, 136, 122]. For all these works, once the action boxes per frames are established they are linked together to create action proposals per video, for example via dynamic programming based on the Viterbi algorithm ([39]).
Unsupervised action proposals do not require any class labels or bounding box ground-truth. A sliding window sampling is unsupervised but has an exponentially large search space. More efficient methods generate action proposals by grouping super-voxels based on low-level cues such as color or image gradients ([15], [137], [138]). Clustering of motion trajectories is also an effective choice to hierarchically build proposals ([139], [140], [141]).

Weakly-supervised action proposals do not rely on box-level ground-truth for all the video frames ([13], [142], [143], [144]). Instead, they exploit object detectors trained on images to get detections. [144] use a human detector and motion scores to locate boxes and compute an actionness score for each of them. Linking of boxes is formulated as a maximum set convergence problem. [13] rely on an upper-body detector per frame and links them in a tube by tracking optical flow feature points. For the linking of our human and non-human actor detectors, we prefer a similarity based tracker ([145], [146]), which is more robust to deformations and can recover from loose and imprecise detection boxes.

Full supervision results in more precise boxes but scales poorly as the number of action classes grows. Unsupervised proposals are more scalable, but boxes are often less precise. Our approach achieves the best of both worlds. We obtain box precision by using an actor detector and then link the boxes from consecutive frames by Siamese similarity matching, making them robust to deformations. At the same time, our approach is action-class agnostic and hence more scalable.

Proposal classification

Supervised classification is the default in the action localization literature. Methods train classifiers using box-supervision for each action class and apply it on each action proposal for each test video, e.g. [39], [136], [41], [135], [40], [134], [121]. Others, who rely on unsupervised or weakly-supervised action proposals, also train their action proposal classifiers in a supervised fashion using bounding-box ground-truth across
frames \([137, 139, 140]\).

Unsupervised classification has been addressed as well. \([141]\) classify their unsupervised proposals using tube-specific and class agnostic detectors, trained via two-stage transductive learning. \([147]\) start with supervoxel segmentation and automatically discover action classes by discriminative clustering. It localizes actions by knapsack optimization. \([148]\), classify action proposals in a zero-shot fashion by encoding them into a semantic word embedding spanned by object classifiers. \([149]\), capture actors, relevant object detections and their spatial relations in a word embedding. All the training happens on images and text, no videos are needed.

Weakly-supervised classification refrains from box-supervision for classifying action proposals. A considerable reduction in annotation effort may be achieved by replacing boxes with point annotations and unsupervised action proposals \([126]\), but it still demands manual labor. An alternative is to rely on human body parts \([150]\) or human detectors trained on image benchmarks \([151, 152]\) to steer the localization in video; either by defining the search space of most likely action locations, \(e.g.\) \([127]\), or by selecting the most promising action proposal \([128]\).

We also rely on human (and non-human) actor detectors but exploit them to generate a limited set of actor proposals. Among those, we select the best ones per action, based on an actor attention mechanism that only requires action class labels. Without the need for box annotations per video frame, we achieve results not far behind the supervised methods and much better than unsupervised methods.

### 4.3 Actor-Supervision

To deal with the inherent difficulty of spatiotemporal action localization without box supervision, we introduce actor supervision. We exploit the fact that actors are precursors of actions. Actions result from an actor going through certain transformations, while possibly interacting with other actors and/or objects in the process. This means
Figure 4.2: **Actor-supervised deep architecture** for spatiotemporal localization of actions. Blocks in yellow generate actor proposals where it is likely to find actions in the videos. Our actor attention module (blue blocks) determines the action occurring in each actor proposal and sorts them based on the relevance of the actor class. Actor-supervision allows to solve this task without the need of action boxes.

that actors not only locate the action in the video, but also one can learn to rank the potential actor locations for a given action class. Based on these premises, we design a novel end-to-end architecture for spatiotemporal action localization guided by actor supervision.

Figure 4.2 illustrates the two pillars of our architecture, namely *actor proposals* and *actor attention*. In a nutshell, our approach enables the localization of actions with minimal supervision by (i) infusing the concept of actors in the architecture, through existing knowledge and progress in object detection and object tracking; and (ii) introducing a powerful attention mechanism suitable for learning a meaningful representation of actions. In the following subsections we disclose full details of each pillar.

**Actor proposals.** Our actor proposals receive a video stream and generate a set of tubes, parameterized as a sequence of boxes $\mathcal{T} = \{\mathcal{B}_i\}$. The tubes outline the most likely spatiotemporal regions where an action may occur based on the presence of an actor. It contains two modules: actor detection and actor linking, as shown in Figure 4.3 and detailed next.
Figure 4.3: **We generate actor proposals** by fusing an actor detector and an actor linking module. During linking, we track relevant detections with a Siamese similarity network to robustly overcome actor deformations. After an actor tube $B_i$ is formed, we filter out detections with high similarity with the boxes of the tube. Notably, our actor linking can handle missed actor boxes in the fourth frame without requiring annotations for fine tuning.

**Actor detection.** This module generates spatial locations where the actor of interest appears in the video. Respecting the requirements of our setup, this module adopts a pre-trained conv-net for object detection which predicts bounding boxes over all the frames of the video. Despite the huge progress in object detection, the predictions are still imperfect due to false positive errors or missed detections of the actor. Missed detections typically occur when the actor undergoes a significant deformation, which is common in actions. For example, when performing a cartwheel in floor gymnastics, the shape of the actor changes when he/she is flipping upside down. In these cases, actor deformations, characteristic of the performance of the action, may involve significant visual changes that do not fit the canonical model of the object category of the actor.

**Actor linking.** This stage carefully propagates the predictions of our actor detector throughout the video to generate an actor proposal tube. It complements the detector by filling the gaps left during the performance of the action, without demanding any annotation to tune the detector. In this way, our module is more robust to missed detections and consistently retrieves complete actor tubes associated with actors. We attain this goal with the aid of a robust similarity-based tracker along with a scheme to filter and select the boxes enabling detection boxes and tracker coordination. The
similarity tracker exploits the temporal coherence between neighboring frames in the video, generating a box-sequence for every given box. In practice, we employ a pre-trained similarity function learned by a Siamese network which strengthens the matching of the actor between a small neighborhood in adjacent frames. Once it is learned, this similarity function is transferable and remains robust against deformations of the actor ([145, 146]). The filtering and selection scheme selects the best scoring detection boxes and sequentially feeds them to the tracker, which propagates them into box-sequences $B_i$, also called tubes. This scheme also filters out the candidate detections, similar to the boxes generated by the tracker, reducing the amount of computation.

Section 4.4.2 describes the implementation details about the conv-net architectures used to generate our actor proposals.

**Actor attention.** The second pillar of our approach is responsible for assigning action labels to the actor proposals without any action box supervision. It takes into account the visual appearance inside the actor proposals, and scores them based on action classification models trained on video-level class labels only. The outcome of this module is a set of ranked proposals where it is likely to find particular actions in the video. Figure 4.2 illustrates the inner components of our actor attention which are detailed next.

**Video encoder.** The encoder transforms the video stream into a suitable space where our attention module can discern among different actions. In practice, we use a conv-net, which encodes video frames as response maps that also comprise spatial information. Without loss of generality, an input video with $T$ frames and shape $T \times 3 \times W \times H$ produces a tensor of shape $T \times C \times W' \times H'$, where $C$ is the number of response maps in the last layer of the video encoder. $W'$ and $H'$ correspond to scaled versions of the original width and height, respectively, due to the pooling layers or
Figure 4.4: **Actor pooling** computes a smooth fixed sized representation of each actor proposal. It crops the feature maps around each actor box and aligns a sampling grid $X \times Y$ to compute the representation via bilinear interpolation with the kernel $k(d)$. The final representation of the actor proposal is obtained through temporal average pooling. We illustrate the procedure for a single actor proposal over one slice of the feature maps.

**Actor pooling.** We introduce a new pooling operation that takes as input the response maps from the video encoder and the set of actor proposals, and outputs a fixed size representation for each actor proposal. This module identifies the regions associated with each actor proposal in the response maps, and extracts a smooth representation for them. Our operation extends the bilinear interpolation layer ([153, 154]), which operates over feature maps of images and bounding boxes, to deal with feature maps of videos and spatiotemporal tubes. Concretely, given an input feature map $U$ of shape $T \times C \times W' \times H'$ and a set of actor proposals $A$ of shape $T \times P \times 4$, which represents coordinates of the bounding boxes of $P$ actor proposals of length $T$. We interpolate the features of $U$ to produce an output feature map $V$ of shape $T \times P \times C \times X \times Y$ where $X,Y$ are the hyper-parameters representing the size of the desired output features of each actor box. For each actor box, we perform bilinear interpolation by projecting the bounding box onto the corresponding $U_{t,:,:,:}$ and computing a uniform sampling grid of size $X \times Y$, inside the actor box, associating
each element of $V$ with real-valued coordinates into $U$. We obtain $V$ by convolving with a sampling kernel $k(d) = \max(0, 1 - |d|)$:

$$V_{t,p,c,i,j} = \sum_{i' = 1}^{W'} \sum_{j' = 1}^{H'} U_{t,c,i',j'} k(i' - x_{t,p,i,j}) k(j' - y_{t,p,i,j})$$  \hspace{1cm} (4.1)$$

Finally, we average pool the contribution of all the output features belonging to the same actor proposal, which gives us a tensor $Z$ of shape $P \times C \times X \times Y$ corresponding to the final output of our actor pooling.

Figure 4.4 illustrates the inner details of actor pooling. Note that the bilinear kernel $k$ ensures that only the four pixels adjacent to the point $(x_{t,p,i,j}, y_{t,p,i,j})$, which belongs to the sampling grid defined by each actor box, contributes to the final representation. This contrasts with the recently proposed Tube of Interest Pooling ([136, 155]), which takes the maximum value over a bin cell. In this sense, our actor of interest pooling not only generates a smooth representation for the actors, but also yields less sparse gradients during backpropagation. This operation is amenable to acceleration in GPUs by ignoring the double summation and computing each entry in $V$ via bilinear interpolation.

**Actor proposal classification.** We classify each actor proposal according to a pre-defined set of action classes. This module learns to map the fixed size representation of each actor proposal into the space of actions of interest. In practice, we employ a fully connected layer where the number of outputs corresponds to the number of classes. During training, the main challenge is to learn an appropriate mapping of the actor representation into the action space without the obligation of annotations for each actor proposal. For this reason, we propose an attention mechanism over the actor proposals that bootstraps the action labels at the video level. In this way, we encourage the network to learn the action classifier by focusing on the actors that contribute to an appropriate classification. In this work, we explore the use
Our actor-supervised architecture is the first end-to-end approach for weakly-supervised spatiotemporal action localization that update the parameters of the video encoder as training progresses. Dashed lines correspond to gradient data flow for the parameters update. Notably, we do not enquiry for any supervisory signal different than action labels at the video label.

of an attention mechanism based on top-$k$ selection. It encourages the selection of the $k$ most relevant actor proposals per class that contribute to perform a correct classification. In practice, we choose the top-$k$ highest scores from the fully connected layer for each action category, and average them to form a single logit vector for each video. Subsequently, we apply a softmax activation on the logits of each video.

We train our actor attention using the cross-entropy loss between the output of the softmax and the video label. We fit all the parameters of the actor attention module employing back-propagation and stochastic gradient descent. In the case of the top-$k$ selection module, we use a binary mask during the back-propagation representing the subgradients of the selection process.

Figure 4.5 illustrates the modules involved during the training of our actor supervised architecture. These are (i) the video encoder; (ii) our actor pooling; (iii) the actor attention classifier; and (iv) the attention mechanism. The diagram highlights the end-to-end characteristics of our architecture, and the capability to be trained in a weakly-supervised setup, only from class labels at the video level. In contrast with Li et al. [129], gradients flow backwards down to pixels updating the parameters of our video encoder. Our experiments showcase the benefits of our design vs. using pre-computed features as [129]. In this regard, our work is the first deep network for weakly-supervised spatiotemporal action localization trained on raw visual data.
4.4 Experiments

4.4.1 Datasets and evaluation

We validate our approach on four public benchmarks for spatiotemporal action localization in videos.

**UCF-Sports ([156])**. This dataset consists of 150 videos from TV sport channels representing 10 action categories such as weightlifting, diving, golf-swing. We employ the evaluation protocol established by [142], but without using the box annotations in the training set.

**JHMDB ([157])**. This dataset showcases 21 action categories such as push-up, shooting, etc.; and consists of 928 videos from movies and consumer videos from internet portals. Unless stated otherwise, we employ the standard evaluation protocol using three splits. We refrain from using the box annotations in the training set to tune our model.

**THUMOS13 ([11])**. This dataset incorporates untrimmed videos and multiple action instances per video. It consists of a subset of 3,294 videos derived from UCF101 featuring 24 action categories. We use the training and testing partition from split 1 of UCF101 for evaluating our approach. Note that we do not rely on the spatiotemporal box annotations of the training set.

**A2D ([124])**. The actor-action dataset comprises 3,782 videos from Youtube designed to model the relationship between actors and actions in videos. This dataset considers actions, such as flying, jumping, climbing, etc., as performed by various actors, such as ball, cat, baby, etc. Again, we do not use the spatiotemporal annotations of the training set.

**Evaluation.** Following the standard protocol for action localization, we report the intersection over union (IoU) to measure the degree to which a candidate tube is associated with a given spatiotemporal action ground-truth annotation. Depending
on the task and dataset of interest, we report the result in terms of Recall or mean Average Precision (mAP). To evaluate action classification performance, we employ the evaluation setup typical for action localization using mean average precision (mAP) over all available classes, given an overlap threshold of 0.2 (THUMOS13) and 0.5 (UCF-Sports, JHMDB).

### 4.4.2 Implementation details

**Actor proposals.** We use a single-shot multi-box detector to detect an actor of interest in every frame [132]. We train the detector on all MS-COCO categories and limit the detections used to the actors of interest according to the action categories defined in each dataset [152]. The base network is an InceptionV2 network pre-trained on ILSVCR-12 [5]. We only track the detections selected by our actor linking forward-and-backward over the entire video using a multi-scale fully-convolutional Siamese-tracker [146] trained on the ALOV dataset ([158]). The base network of the tracker corresponds to the first four convolutional blocks of VGG-16. The actor selector ignores detections predicted by the actor detector greedily when those have a high spatial affinity with boxes generated by the tracker. In practice, we use an overlap threshold of 0.7.

**Actor attention.** We only consider the RGB stream to encode the visual appearance of the videos. Our video encoder corresponds to the convolutional stages of VGG-16, for fair comparison with previous work [129], pre-trained on ILSVCR-12 [5]. The grid size for the bilinear interpolation of actor pooling is $3 \times 3$. During training, our attention module focuses on the $k = 3$ most relevant actors out of 10 actor tubes for classifying the video. We train our entire actor attention module end-to-end from RGB streams to video labels, adjusting the parameters of the visual encoder as training progresses, as opposed to using pre-computed features like [129]. Due to memory constraints, we employ segment partition introduced by [159] to allocate more than
Figure 4.6: **Actor linking** outperforms the Viterbi linking algorithm by Gkioxari and [39], used to connect sparse detection in time. We attribute its success to the use of similarity-based matching to handle deformations of the actor that the Viterbi linking is unable to fix.

We set the learning rate to 1e-2 and employ a momentum factor of 0.99 to train our model in a single GPU with a batch size of four videos.

### 4.4.3 Results

**Actor linking versus Viterbi linking.** In our fist experiment we validate the relevance of our actor linking with respect to the more traditional Viterbi linking for the generation of actor proposals from the predictions of our actor detector [39]. As shown in Figure 4.6 our actor linking achieves an improvement in Recall of +19.9% and +21.6% for 0.2 and 0.5 IoU in THUMOS13. We attribute these results to the capability of the similarity-based matching to accommodate for deformation of the actor, that the Viterbi linking approach is unable to fix. Previous approaches *e.g.* [39, 13, 40] employ supervision at the level of boxes and length of the tubes to overcome this issue. This clearly limits their application under the weakly-supervised setup evaluated in this work. We conclude that actor linking, by similarity-based matching, aids spatiotemporal action localization with weak supervision.

**Actor proposals versus others.** Table 4.1 compares our actor proposals with previous supervised and unsupervised action proposals. Compared to the action proposals by [144], our approach achieves an improvement of +34.2% in terms of
Table 4.1: **Action proposal comparison** in terms of Recall. \cite{121} and \cite{122} use video supervision from action boxes and action labels, while the rest do not use any video supervision. Our actor proposals achieve better Recall compared to previous unsupervised and weakly-supervised methods.

<table>
<thead>
<tr>
<th></th>
<th>UCF-Sports</th>
<th>THUMOS13</th>
<th>JHMDB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IoU=0.5</td>
<td>IoU=0.5</td>
<td>IoU=0.1</td>
</tr>
<tr>
<td>\cite{121}</td>
<td>98.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>\cite{122}</td>
<td>96.8</td>
<td>61.4</td>
<td>-</td>
</tr>
<tr>
<td>\cite{141}</td>
<td>-</td>
<td>-</td>
<td>54.5</td>
</tr>
<tr>
<td>\cite{139}</td>
<td>89.4</td>
<td>35.5</td>
<td>-</td>
</tr>
<tr>
<td>\cite{137}</td>
<td>91.9</td>
<td>32.8</td>
<td>-</td>
</tr>
<tr>
<td><strong>This work</strong></td>
<td><strong>93.6</strong></td>
<td><strong>43.9</strong></td>
<td><strong>88.7</strong></td>
</tr>
</tbody>
</table>

Recall on THUMOS13. This result evidences the benefit of our actor detection and linking scheme. Interestingly, our approach improves upon previous unsupervised work by +1.7% and +8.4% in terms of Recall on UCF-Sports and THUMOS13, respectively. These methods, \cite{137}, \cite{139}, are based on grouping techniques over low-level primitives such as color and motion, which reaffirms our intuition about the relevance of actors as a strong semantic cue for the localization of the actions. Figure 4.7 illustrates the recall of our actors proposals for a varying number of proposal in comparison with previous unsupervised approaches. It provides further evidence on the quality of our proposals, especially when considering only a limited number of proposals.

The state of the art for action proposals generation are fully supervised approaches based on a mix of convolutional-recurrent networks and supervised instance level tracking \cite{122}, \cite{121}, respectively. Our approach achieves competitive results or even outperforms them in JHMDB-split1, following the evaluation protocol of \cite{121}, without relying on additional action box supervision. Although supervised approaches offer proposals with better quality in two out of the three datasets studied, they do so at the expense of extra annotations. It that sense, these methods have a limited scalability potential, and it opens up a spot for our actor proposals.
Figure 4.7: **Actor proposals outperform unsupervised action proposals.** We attribute its success to the use of actors as semantic cue relevant for the grounding of actions. Notably, we retrieve relevant action tubes from a much smaller pool, which is advantageous in the context of retrieval and spatiotemporal localization of actions.

**Non-human actor proposals.** We also analyze the quality of our proposals for generating spatiotemporal tubes for non-human actors. Figure 4.8 summarizes our findings including two qualitative results taken from the A2D dataset. The leftmost example shows we are able to generate proposals for highly articulated actors like *birds*. The example in the center exemplifies a common failure case for the *ball* actor. In this case, the ball changes significantly in appearance during the execution of the action, including several full occlusions caused by the human. From our quantitative analysis, we appreciate that the actor with the highest recall is *baby*, which is not directly represented in the training set of our actor detector. The analysis also reveals that the gap in recall at 0.5 IoU between adult actors and animal actors like *bird* and *dog* is at most +2%. Except for *ball*, the recall for all the actors using at most 50 actor proposals is greater than 85%. Therefore, we conclude that our method is general and applicable to both human and non-human actors.

**Impact of actor attention.** We first validate the impact of fine-tuning the video encoder in an end-to-end fashion, as training progresses. Table 4.2 (a) summarizes our findings on THUMOS13. When the video encoder is fine-tuned we achieve an
Figure 4.8: **Non-human actor proposals** on A2D. Qualitative visualizations of action proposals for *Bird* and *Ball*. The recalls at IoU=0.5 are consistently high for all the actor classes except for *Ball*, which is understandable due to its common shape and small size, which invite many occlusions. Best viewed in color.

Table 4.2: **Impact of actor attention.** (a) Fine-tuning the video encoder adds 4.4% in mAP as opposed to a fixed visual representation as done in [129]. Similarly, (b) in our weakly-supervised setup actor pooling adds 7.9% compared to the tube pooling of [136].

<table>
<thead>
<tr>
<th>Video encoder</th>
<th>Pooling</th>
<th>THUMOS13 mAP@0.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed</td>
<td>Actor pooling</td>
<td>41.4</td>
</tr>
<tr>
<td>Fine tuned</td>
<td>Actor pooling</td>
<td><strong>45.8</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Video encoder</th>
<th>Pooling</th>
<th>THUMOS13 mAP@0.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine tuned</td>
<td>Tube pooling</td>
<td>37.9</td>
</tr>
<tr>
<td>Fine tuned</td>
<td>Actor pooling</td>
<td><strong>45.8</strong></td>
</tr>
</tbody>
</table>

improvement of 4.4% in mAP, compared to when employing a fixed visual representation, as done in [129]. We also analyze the relevance of our actor pooling operation in Table 4.2(b). To this end, we compare our actor pooling with the tube pooling of [136], by replacing the former with the latter in our actor attention module. For this experiment, we also fine-tune the video encoder of the modified network using tube pooling. The result shows an improvement of +7.9% in mAP when actor pooling is used. We found that the pooling operation introduced in this work is more robust to variations in the optimization setup and improves consistently upon the operation described by [136] in a weakly-supervised scenario. We hypothesize that actor pooling is more robust due to the use of a smooth representation with less sparse gradients as mentioned in section 4.3 while training in a regime without box level supervision.
Table 4.3: **Comparison of spatiotemporal action localization approaches** with decreasing amount of supervision. The top half shows supervised approaches, whereas the bottom half shows weakly-supervised approaches relying on action class labels only. Actor-supervision achieves state-of-the-art performance among the weakly-supervised approaches and sometimes even outperforms supervised methods. The performance marked with * denotes use of RGB frames solely, as we do.

<table>
<thead>
<tr>
<th>Action Supervision</th>
<th>THUMOS13</th>
<th>UCF-Sports</th>
<th>JHMDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boxes Points Labels</td>
<td>mAP@0.2</td>
<td>mAP@0.5</td>
<td>mAP@0.5</td>
</tr>
<tr>
<td>[41]</td>
<td>✓</td>
<td>✓</td>
<td>77.2</td>
</tr>
<tr>
<td>[134]</td>
<td>✓</td>
<td>✓</td>
<td>73.5</td>
</tr>
<tr>
<td>[136]</td>
<td>✓</td>
<td>✓</td>
<td>47.1</td>
</tr>
<tr>
<td>[137]</td>
<td>✓</td>
<td>✓</td>
<td>48.1</td>
</tr>
<tr>
<td>[121]</td>
<td>✓</td>
<td>✓</td>
<td>46.8</td>
</tr>
<tr>
<td>[39]</td>
<td>✓</td>
<td>✓</td>
<td>37.8</td>
</tr>
<tr>
<td>[139]</td>
<td>✓</td>
<td>✓</td>
<td>26.5</td>
</tr>
<tr>
<td>[141]</td>
<td>✓</td>
<td>✓</td>
<td>34.8</td>
</tr>
<tr>
<td>[126]</td>
<td>✓</td>
<td>✓</td>
<td>5.5</td>
</tr>
<tr>
<td>[160] (from [129])</td>
<td>✓</td>
<td>*5.5</td>
<td>-</td>
</tr>
<tr>
<td>[161] (from [126])</td>
<td>✓</td>
<td>13.6</td>
<td>-</td>
</tr>
<tr>
<td>[129]</td>
<td>✓</td>
<td>36.9</td>
<td>-</td>
</tr>
<tr>
<td>[128]</td>
<td>✓</td>
<td>37.4</td>
<td>37.8</td>
</tr>
<tr>
<td>This work</td>
<td>✓</td>
<td>*45.8</td>
<td>*48.2</td>
</tr>
</tbody>
</table>

**Actor-Supervision versus others.** Table 4.3 compares multiple approaches with a varying degree of supervision. Based on these results, we note that actor supervision achieves the state-of-the-art among all approaches that localize action from an action class label only. It improves upon [129] by +8.1% and upon [128] by +10.4% on the THUMOS13 and UCF-Sports benchmarks, respectively. Compared to [128], our approach gives more relevance to the actors during the localization stage instead of using them as cues to improve the ranking of existing action proposals. We hypothesize that our attention mechanism is more effective than the one in [129], because actors are a more powerful cue for guiding the localization of actions than individual pixels.

Table 4.4 illustrates the results of our approach for a broad range of IOU thresholds in contrast with the state-of-the-art approach for weakly-supervised action localization of [128]. These results complements the global trend presented in Table 4.3.
Table 4.4: **Localization results across multiple IOU values.** Actor-supervision achieves better localization for a large range of IOU thresholds w.r.t. the state-of-the-art weakly-supervised approach [128].

<table>
<thead>
<tr>
<th>IOU thresholds</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCF-sports</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>This work</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UCF-sports</td>
<td>87.7</td>
<td>81.7</td>
<td>64.4</td>
<td>54.5</td>
<td>37.8</td>
<td>17.5</td>
</tr>
<tr>
<td>This work</td>
<td>84.2</td>
<td>84.2</td>
<td>84.2</td>
<td>64.2</td>
<td>48.2</td>
<td>40.6</td>
</tr>
<tr>
<td>THUMOS13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>This work</td>
<td>53.4</td>
<td>45.8</td>
<td>38.0</td>
<td>30.7</td>
<td>19.3</td>
<td>6.2</td>
</tr>
</tbody>
</table>

showcasing that our architecture achieves better localization accuracy, on the more stringent IOU thresholds (high values). Note as well that our actor-supervision outperforms this multiple instance learning approach in the most challenging dataset, THUMOS13, over all the IOU values reported. This result reaffirms the effectiveness of using actor information to achieve precise localization without resorting on action boxes annotations.

Actor-supervision also outperforms several approaches with varying levels of supervision on the challenging THUMOS13 benchmark [144, 139, 126]. Compared to [121, 136], our visual representation is limited to the RGB video stream, which clearly leaves room for improvement when combined with richer modalities such as optical flow or the use of 3D convolutions in our video encoder ([17]). Similarly, we compare actor-supervision on JHMDB with [39], using their RGB stream only, and observe comparable results (35.8%±2.7 versus 37.9%). This proves the relevance of our approach in another challenging benchmark against a strong competitor aided by box-supervision during training.

The state-of-the-art in action localization is dominated by fully-supervised approaches resembling conv-net architectures well established for generic object detection [41, 134], which require strong levels of supervision. These approaches are unable to be trained in the weakly supervised regime presented in this work. Interestingly, our approach not only outperforms other weakly-supervised methods but it also has
Figure 4.9: **Qualitative results on the THUMOS13 dataset.** Top row shows three successful cases by visualizing the ground-truth and action tubes as well as two highlighted frames. These include action sequences that have deformations of actor as well as multiple actors with complex background. Bottom row visualizes three failed cases which show that crowded background, occlusions and temporally untrimmed action sequences are the most challenging scenarios. Best viewed in color.

This Chapter introduces a weakly-supervised approach for the spatiotemporal localization of actions, driven by actor supervision. We show that exploiting the inher-
ent compositionality of actions, in terms of transformations of actors, disregards the dependence on spatiotemporal annotations of the training videos. In the proposal generation step, we introduce actor supervision in the form of an actor detector and similarity-based matching to locate the action in the video as a set of actor proposals. Then, our proposed actor attention learns to classify and rank these actor proposals for a given action class. This step also does not require any per frame box-level annotations. Instead, we design an attention based mechanism that chooses the most relevant actor proposal only for class labels at the video level. Moreover, we introduce a novel actor pooling operation that summarizes the representation of each actor in a more effective way than recent pooling strategies for the weakly-supervised setup of our interest. Our approach outperforms the state-of-the-art among weakly-supervised works and even achieves results that are better or competitive to some of the fully-supervised methods.


Chapter 5

Conclusion

This dissertation summarizes my take on the pursuit of general algorithms for video understanding, in particular on the localization of human actions and moments of interest. My main contributions are two fold: (i) the development of efficient models for temporal localization in a single or a collection of untrimmed unsegmented videos; and (ii) the use of actor-supervision as an effective approach for outlining actions in trimmed clips without the need of expensive action boxes.

In Chapter 2, I developed the first action proposal model that processes the video in a single pass. Our approach not only outperforms previous multi-scale methods, but also achieves a faster run-time, 134-300 FPS. This Chapter also showcased that developing an end-to-end solution integrating proposal and classification of actions is more optimal for the purpose of temporal action detection.

In Chapter 3, I extended the temporal localization problem from a single video to multiple videos and incorporated the use of natural language queries. This constitutes a new task for the computer vision community called “corpus moment retrieval from natural language queries”. A new model was proposed by expressing the alignment cost of any variable length moment in terms of matching a given query to short video clips. I demonstrated that this model offers interesting advantages in terms of efficiency, scalability and performance in relationship to existing alternatives.

Finally, Chapter 4 presents an approach for outlining the actor performing the human actions of interest in trimmed video sequences. A major limitation for the development and scalability of this kind of models is the strong dependence on exhaustive
action boxes annotations during training. To this end, I introduce actor-supervision to exploit the inherent compositionality of actions, in terms of actors transformations. Concretely, I developed a new actor-supervised architecture that leverages the power of transfer learning inside a novel weakly-supervised framework and achieves state-of-the-art results in action localization without action boxes annotations.

To sum up, this thesis has presented a clear path that delivered efficient localization of human actions and moments of interest in videos. The models developed here respond to the current video growth requirements and needs as they are efficient, scalable, and enable interactivity with the user by responding to natural language queries.

Despite of the rapid progress in video understanding, presented here as well as by contemporary work, many challenges still remain before finding the holy-grail of video understanding like humans. I have identified two opportunities where more progress is needed:

First, tackling the problem in a multi-modal fashion. Videos are more than stream of pixels, for instance, it is common to record audio waves along with the visual information. Thus, it is pertinent to develop algorithms that understand videos and human actions beyond the visual data. Similar to our endeavor of localizing temporal moments of interest by aligning visual information to natural language queries, c.f. Chapter 3, it is time to explore the interplay between visual data and audio, speech or even heart rate signals from the actors. In particular, we could think on training an embedding between visual and heart rate signals that predicts if a person is stressed out or not. Interestingly, recent attempts on aligning audio waveforms and visual data [162, 163] not only showcase more integral solutions to parse videos but also suggest that multi-modal training is a powerful initialization strategy. From these holistic, utilitarian and opportunistic perspectives, it is time to work on video

\footnote{Nowadays, smartwatches and IoT devices record the heart rate, temperature and other useful signals of the actors and the environment.}
Figure 5.1: Challenges ahead. Despite of the progress presented in this work, we still far from developing algorithms with a rich social perception of all the human activities, objects, scenes and interactions in the real-world. Refer to text for details. Taken from [https://cuesa.org/foodwise-kids](https://cuesa.org/foodwise-kids), copyright belongs to the authors.

algorithms that learn and digest multiple modalities.

Second, tackling the understanding problem beyond a single task. Currently, most of the work on human activity understanding, including this thesis, focus on categorizing/localizing moments of interest in video. Although we have witnessed progress by tackling a well-defined unique task (c.f. Chapter 2 - localizing temporal action instances, and Chapter 4 - localizing actors and their actions), we are leaving aside interesting aspects of the social perception problem of human actions. Let consider the snapshot depicted in Figure 5.1. As today we would be able to outline this piece from an untrimmed video and label it as “market transaction”. We could also outline certain actors, but we would miss all the social relationships depicted here. For example, humans could easily: (i) describe the actor on the left as the seller, and the family on the right as the buyers during this action. (ii) Identify the rightmost woman as not belonging to the family group despite its proximity on the pixel space. (iii) Recognize that the buyers are most likely acquiring vegetables, and (iv) infer that the woman is explaining the leftmost child why that is a wise
choice. (v) Identify that they are in a market place on a public street, and (vi) infer that the same place may serve for a different activity at another moment. In other words, video understanding is more than outlining regions of interest and associating a single action label to them. Actors perform actions based on the context around them, and their actions in context determine the role of the actors and the scene. In the future we should develop algorithms that expose all these pieces from a video; actor, objects (iii), scenes (v), their relationships (ii, iv), and scene/actors roles (i, vi); i.e. tackling the understanding of videos as a multi-task problem with multiple instances (objects, scenes and actors), executing/participating on multiple actions, and performing different roles over time.

To make further progress on our quest for the holy-grail of video understanding, let hit play and unroll more insights from videos in a holistic manner.
REFERENCES


APPENDICES
A Temporal Localization in Video Collections

Supplemental Details

We supplement the presentation of our new task of retrieval of relevant video moments from a large corpus of untrimmed videos, Chapter 3, with the following items:

- A video summarizing our work. We show additional qualitative results generated by our Clip Alignment with Language model (CAL).
  
  
  The codec of the video is H264 - MPEG-4 AVC with resolution 1280 × 720 and frame rate of 60 fps.

- Details about our CAL architecture (Section A.1).

- Results of Mixture of Embedding Experts [33] (MEE) (Section A.2).

A.1 Model architecture details

Our Clip Alignment with Language model architecture was built on top of the insights of MCN [35]. In particular, the multilayer perceptron (MLP) of the visual stream is formed by two linear layers with a ReLU non-linearity in between. The number of hidden units is 500 and 100, respectively. Note that the size of the embedding space corresponds to the number of units in the second linear layer which matches those in the linear layer of the text query stream. The LSTM layer contains 1000 hidden units and follows the recurrent equation presented by Donahue et al. [164]. Note that we did not observe significant improvements in performance by increasing the depth and capacity of our architecture.
Figure A.1: Our Clip Alignment with Language architecture with tensor shapes. See text for details.

Figure A.1 illustrates our CAL architecture with the tensor shapes across the model. In this particular instantiation, we assume the feature vector produced by the CNN is of size 2048. Note that we tile the output of the Average pool clip features, and the temporal endpoints to match the number of clips in a given moment (for example, we tile to length two to match the two clips in the example illustrated in Figure A.1) before the Concat operation.

We train our architecture in an end-to-end fashion using supervised learning by updating all the parameters except for the CNN weights (c.f., training paragraph of Section 3.3).

A.2 Mixture of Embedding Experts for text-to-video retrieval

In this section, we provide more details about the Mixture of Embedding Experts baseline (MEE) used in section 4.2 (c.f., main submission) [33]. This model belongs to the family of methods of video clip retrieval with natural language described in
Table A.1:  **Text to video retrieval results.** We show Recall@K on MSR-VTT [30], DiDeMo [35], and Charades-STA [36] with RGB features. See text for more details.

Section 2 in the main submission. In a nutshell, given a natural language query, MEE retrieves the most similar trimmed video clip that aligns with the given query. This approach, by itself, falls short of addressing our proposed task of retrieving relevant moments from a large corpus of untrimmed, unsegmented videos. Thus, we paired MEE with a model for localizing moments in a single video in a two-stage fashion to fulfill the requirement of our task.

We chose MEE over other methods as it performs well on LSMDC benchmark and outperforms other recent methods [34] on MSR-VTT [30]. Table A.1 “(MSR-VTT)” shows the quantitative comparison in the text to video retrieval task on the MSR-VTT corpus. We used the publicly available implementation of MEE to retrieve videos from the corpus and turned off flow, face and audio features in MEE for fair comparison.

For our two-stage retrieval baseline of moments from a video corpus for a natural language query, we tested MEE models pre-trained on MSR-VTT and the corresponding dataset. Table A.1 “(DiDeMo, Charades-STA)” summarizes the results of these experiments. We observed that MEE pre-trained on MSR-VTT performed near chance; while re-training MEE on the target datasets performed the best. Thus, we
used the latter setup in the rest of the experiments in our main submission.
B Actor-Supervision Supplemental Details

We supplement the presentation of our Actor-Supervision architecture, presented in Chapter 4 with the following items:

- A video summarizing our work. It showcases more qualitative results generated by our actor-supervised architecture. There you can appreciate that our approach works well in videos with multiple actors, involving considerable deformations and for non-human actors.


  The codec of the video is H264 - MPEG-4 AVC, its resolution is 1280 × 720 and the frame rate 60. We recommend using the VLC media player [https://www.videolan.org/vlc/index.html](https://www.videolan.org/vlc/index.html).

- In Section B.1, we describe the inner details of our actor proposal algorithm and comment about its computational complexity.

- Section B.2 gives more details about the training of our Actor-Supervision architecture, and robustness of our actor pooling operation.

B.1 Actor Proposals Algorithm

The algorithm describes all the interactions between the inner blocks involved for the generation of our actor proposals, described in this work.

Regarding the time complexity of our approach, our object detector and Siamese-tracker run at 32 and 60 FPS, respectively, on an Intel-Xeon E5-2687 with a GTX
Algorithm 1 Actor proposals generation

1: **Input:** maximum number of proposals $N$
2: **Output:** $\mathcal{T}$
3: $\mathcal{D} \leftarrow$ run actor detector over all frames
4: $\mathcal{T} \leftarrow \emptyset$
5: $i \leftarrow 0$
6: while $\mathcal{D} \neq \emptyset \land i < N$ do
7: \quad $b_i \leftarrow$ select actor with highest score from $\mathcal{D}$
8: \quad $\mathcal{B}_i \leftarrow$ actor tracker tracks $b_i$ forward and backward throughout the video
9: \quad Push $\mathcal{B}_i$ onto $\mathcal{T}$
10: $\mathcal{D} \leftarrow$ filter actors in $\mathcal{D}$ with high similarity with boxes in $\mathcal{B}_i$
11: $i \leftarrow i + 1$

After direct contact with [137], we establish that our method generates less number of proposals (100), is more accurate (+10% Recall at 0.5 IOU) and 7.91 times faster on THUMOS13. In conclusion, our action proposal approach generates more precise candidates and does so in less time.

B.2 Actor-supervised architecture

Training details of our actor-attention stream We initialize the weights of our actor classifier module with Xavier technique ([165]), and our video encoder with the weights from a VGG-16 model pre-trained on Imagenet-ILSVCR-2012 ([151]). Our actor attention stream is trained for 20 epochs annealing the learning rate by a factor of 0.75 after eleven epochs. We use a momentum factor of 0.99 and an initial learning rate of 0.01. As input pre-processing, we employ the segment based strategy suggested by [159]. In our case, we randomly sample 16 frames uniformly spaced per video. Additionally, we apply a random horizontal flipping of all the sampled frames. Finally, we normalize the input frames such that the intensity values lie on the range between $[-0.5, 0.5]$ using standard scaling with mean $[0.485, 0.456, 0.406]$ and standard deviation $[0.229, 0.224, 0.225]$ for the RGB channel.
Figure B.1: The violin plot clearly shows that using actor pooling in regards of the tube pooling operation \cite{136} results in better localization performance. The middle line in each violin represents the median mAP and its shape is determined by the density function of the sample points in the experiment. The experiment was carried out in THUMOS13 using the training-testing partition from the split 2 of UCF101. More details of the experiment are provided in the text.

**Robustness of actor pooling** As we mentioned in Chapter 4, the use of our actor pooling yields better results than the tube pooling operation \cite{136} in the weakly-supervised setup of our interest. Figure B.1 compares the robustness of our actor pooling operation in regards to the tube pooling operation. The violin plot, Figure B.1, summarizes the statistics of the top-15 results among forty experiments for each trial i.e. our actor supervised architecture using either actor pooling or tube pooling. One experiment constitutes a variation of the grid size (3x3 or 5x5), or the optimization hyper-parameters, such as learning rate, momentum, etc., while maintaining the other hyper-parameters of our actor supervised architecture intact. In this manner, we can compare both operations beyond a single hyper-parameter configuration without introducing other confounding variables. For this experiment, we use the THUMOS13 dataset employing the training-testing partition from the split 2 of UCF101 \cite{11}, ensuring that we do not overfit on the standard partition used for comparing different methods.

We verify that our actor pooling statistically improves upon the tube pooling
operation with a $p$-value of 1%. Thus, we reaffirm that our actor pooling consistently achieves better localization performance than tube pooling beyond a single hyperparameter configuration.