Understanding Human Actions in Video

by

Jonathan Stroud

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2020

Doctoral Committee:

Assistant Professor Jia Deng, Co-Chair
Professor Rada Mihalcea, Co-Chair
Professor Jason Corso
Professor SangHyun Lee

Jonathan C. Stroud

stroud@umich.edu

ORCID iD: 0000-0002-9505-4508

ACKNOWLEDGEMENTS

Thank you to my labmates and friends from the Vision and Learning Lab, who have inspired me to work hard every day. Thank you for the discussions, both about research and about life, that opened my mind and kept me entertained throughout the years. In particular, thank you to Alejandro Newell, Weifeng Chen, Dawei Yang, Lanlan Liu, Johnny Chao, Kaiyu Yang, Ankit Goyal, Hei Law, Zach Teed, Mingzhe Wang, Lahav Lipson, Emily Walters, Zehuan Yuan, Lei Huang, and Zhefan Ye.

Thank you to my thesis committee, Jia Deng, Rada Mihalcea, Jason Corso, and SangHyun Lee, who provided valuable advice as I prepared this dissertation. In particular, thank you to my co-chairs, Jia Deng and Rada Mihalcea, for their constant support and encouragement.

Thank you to my co-authors and those who have contributed to my work, specifically Jia Deng, Zehuan Yuan, Tong Lu, Kaiyu Yang, David Ross, Chen Sun, Rahul Sukthankar, Olga Russakovsky, Ryan McCaffrey, Rada Mihalcea, Haochen Li, Cordelia Schmid, Santiago Castro, Mahmoud Azab, Cristina Noujaim, Ruoyao Wang, George Toderici, Carl Vondrick, Bryan Seybold, Austin Meyers, Zhichao Lu, Xuehan Xiong, Yinxiao Li, Arsha Nagrani, and Meghana Thotakuri.

Thank you to my qualifying exam committee, Rada Mihalcea, Mingyan Liu, and Gregory Wakefield, for your feedback and for allowing me to make it this far.

Thank you to Jia Deng, my thesis advisor and my mentor throughout my PhD. Thank you for teaching me your methodical approach to research and solving prob-

lems. Thank you for bringing me into your lab and showing me why Computer Vision is important.

Thank you to Jake Abernethy, with whom I co-founded the Michigan Data Science Team. I am incredibly proud of what we built and working with you made it enjoyable. Thank you to all of the wonderful students and advisors I have worked with during my time in MDST, particularly Alex Chojnacki, Jared Webb, Arya Farahi, Cyrus Anderson, Eric Schwartz, Jenna Wiens, Danai Koutra, Allie Cell, Josh Gardner, Michael Kovalcik, Wesley Tian, Seth Saperstein, Mukai Wang, Eris Llangos, Cory Laban, and Rohit Mogalayapalli.

Thank you to David Ross, my two-time internship host who has served as my co-author, manager, and mentor. Thank you for your endless positivity and support, and for being a perfect role model.

Thank you to Dave Clausen, the first and funniest computer science teacher I ever had, and still the only one who has ever let me borrow their guitar.

Thank you to Alex Ihler, who introduced me to research and supported my interest in machine learning as an undergraduate.

Thank you to my parents, Jean Stroud and Jeff Stroud, who have supported me in my love of computers since I was in diapers[1]. Thank you to my brothers, Josh Stroud and Jordan Stroud, who make the world a lot more fun to live in.

Thank you to the countless friends I have made during my time at Michigan. I simply can't list you all, but you know who you are. In particular, thank you to my roommates, Tim, Ian, Alejandro, Oskar, Alex, Max, Richard, Lauren, and Mo, for all of the board game nights and excellent food.

Last but not least, thank you to Mel. You are the single most inspiring, empa-

[1]Proof: `https://youtu.be/xxPt0lp08uo`

thetic, and hard-working person I know, and I truly don't know if I would have made it here without you. I can never thank you enough.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

ABSTRACT

Understanding human behavior is crucial for any autonomous system which interacts with humans. For example, assistive robots need to know when a person is signaling for help, and autonomous vehicles need to know when a person is waiting to cross the street. However, identifying human actions in video is a challenging and unsolved problem. In this work, we address several of the key challenges in human action recognition. To enable better representations of video sequences, we develop novel deep learning architectures which improve representations both at the level of instantaneous motion as well as at the level of long-term context. In addition, to reduce reliance on fixed action vocabularies, we develop a compositional representation of actions which allows novel action descriptions to be represented as a sequence of sub-actions. Finally, we address the issue of data collection for human action understanding by creating a large-scale video dataset, consisting of 70 million videos collected from internet video sharing sites and their matched descriptions. We demonstrate that these contributions improve the generalization performance of human action recognition systems on several benchmark datasets.

CHAPTER I

Introduction

As human beings, we are constantly acting and interacting with the world around us. This serves several important functions. First, and most obviously, this is how we enact change in our environment; we build things, and we and work to achieve our goals. Second, this is how we learn about our environment; we actively explore the world around us, conduct small experiments, and test our predictions about the future. Finally, this is how we communicate our knowledge, beliefs, and desires to others; we signal what we know, both explicitly with our words, and implicitly with our actions. Humans are a product of the actions they perform.

Because actions are such a key part of human life, understanding these actions is a crucial skill for any intelligent system which coexists with humans. At the very least, such a system would need to be able to perform **action recognition**, that is, identifying what actions are being performed. Action recognition allows systems to respond to actions in their environment. For example, an autonomous car could identify when someone is waiting to cross the street, and then stop to allow them to cross. Broadly, action recognition allows an intelligent system to better understand their environment as more than just a collection of people and objects. Action recognition allows the system to see how these people and objects interact with one

another, identify their intent, and respond to their needs.

Action recognition in computer vision has been studied extensively for several decades, and significant strides have been made towards accurate and robust action recognition systems [125, 188]. In recent years, such systems typically have fallen into the framework of deep learning, which leverages large amounts of data to train multi-layer model architectures. However, there are many unanswered questions that arise from this framework, specifically regarding two key areas: datasets and models. Datasets have been shown to be crucially important for training deep learning systems. Such datasets need to be large and diverse, and sufficiently general datasets can be used to train models which then transfer well to many down-stream tasks [72]. However, in the domain of video action recognition, datasets are particularly expensive to collect because of the sheer amount of time it takes to watch and label video content. This is particularly problematic because of the models that are often used for video action recognition, which typically have many more parameters, and therefore require more data to train, than models that perform other computer vision tasks. One reason for this is that models for video action recognition must be able to understand low-level motion (that is, movements that take place over a fraction of a second) as well as long-term context (over periods of minutes or hours). This means that models need to recognize complex patterns of appearance, motion, and context, all simultaneously, and integrate these signals effectively. How to do this effectively is still very much an unsolved problem.

In this work, we present several novel model architectures and datasets for action recognition. In Chapters II, III, and IV, we contribute novel model architectures which address the issues of instantaneous motion interpretation (II), and long-term context (III, IV). In Chapter V, we address the challenge of collecting large-scale data

for video action recognition, and demonstrate how such data can then improve the performance of models. Additionally, our various approaches for action recognition each approach a progressively more challenging version of the task, each demonstrating a progressively richer representation of actions. In Chapter II, we model action recognition simply as a video classification task, where each short video clip contains exactly one action from a pre-defined list of human actions. In Chapter III, we treat action recognition as a detection task, where multiple actions may take place concurrently and at different points in a longer video. Finally, in Chapters IV and V, we treat action recognition as a natural language grounding task, where the actions are no longer taken from a pre-defined list, but instead are described in an open-ended fashion using natural text. This progression represents an advance towards more practically useful video action recognition systems, that is, ones which apply to realistic scenarios with fewer assumptions.

1.1 Scope of this Work

Throughout our work, we focus on understanding actions using only their visual content, as opposed to using audio signals or other additional cues. This is primarily for simplicity, as the vast majority of human actions can be recognized from visual cues alone and therefore adding audio needlessly complicates the model architectures. While it is probable that additional cues from audio may be useful for many action recognition systems, this question is beyond of the scope of our work. Our work primarily focuses on building visual representations of actions, and using these representations to perform recognition and understanding.

In addition, our models all act on videos, as opposed to static images. This is because actions inherently involve motion and the passage of time, and therefore

Figure 1.1: Modeling actions requires modeling motion. From the single image above, it is ambiguous whether the action is "opening a door" or "closing a door". Motion is necessary to model such actions.

cannot be fully addressed without modelling their temporal component. To see this, consider the examples from Figure 1.1. These examples each present a single frame from a video which demonstrate an ambiguous action, such as "opening a door" or "closing a door" . Because the temporal component is missing, it is nearly impossible to accurately determine which action is being performed in each frame. However, just a small amount of motion information would make this task possible. Because of the tight coupling between actions and time, a large portion of our work is concerned with representing video content in a way which preserves the temporal properties of actions. While single-image action recognition is itself an interesting line of research, we do not consider this task in our work.

1.2 Applications

In addition to its importance to visual intelligence more broadly, action understanding is an area rich with immediate practical applications. Here, we briefly present a few illustrative examples. While our work does not directly address these applications, we demonstrate in our experiments that our models are capable of

learning a broad vocabulary of actions, and therefore can be used in a wide range of real-world applications, including those listed below.

1.2.1 Video Retrieval

Video retrieval is a classic task at the intersection of computer vision and database design. In this task, we seek to query a database of videos using key terms about their content. For example, we may search the database for videos containing for broad concepts such as "basketball", or more narrow concepts such as "Michael Jordan dunking a basketball". This task has practical utility for users on social media and video sharing websites, and can also be useful for creative applications such as video editing.

Current large-scale approaches to video retrieval rely heavily on user-generated metadata such as video titles, descriptions, and tags. The accuracy of video retrieval therefore depends on the accuracy of the user-generated metadata. However, annotating video content is cumbersome, particularly now that video content is becoming easier to capture and share online. For reference, more than 500 hours of video are uploaded to YouTube every minute [49], and it is likely that the majority of this content has limited annotations available for retrieval.

Because of the difficulty of collecting accurate video labels at scale, recent work has considered the task of content-based video retrieval [201, 41, 164]. In this task, only video content, rather than user-annotated metadata, is used to retrieve results from the database. While this task is much more challenging than its predecessor, it is a much more flexible framework for video retrieval, as it removes any reliance on specific video tags or categories. Conceptually, all of the video representations we build in our work could be used for video retrieval, and in Chapter IV, we present a model for text-to-clip retrieval, a related task.

1.2.2 Pedestrian Detection

Computer vision, in general, is of significant practical importance for autonomous driving, which involves many perceptual tasks such as lane identification, and sign and vehicle recognition. Current systems are quite successful at driving in highly predictable environments such as highways, but significant progress needs to be made before autonomous vehicles can drive in crowded city environments without human intervention.

Action recognition specifically is useful in this situation, and in any mode of driving where pedestrians may be present. In city driving, it is crucial to understand the motions and intentions of pedestrians, which may make sudden moves, such as crossing the street in front of the vehicle. To prevent collisions in such situations, the system can monitor nearby pedestrians and infer their intent from their actions, such as looking both ways before crossing the street.

1.2.3 Human-robot Interaction

Naturally, a robot that interacts with humans needs to be able to recognize and respond to human actions. Ideally, we expect a robot assistant to perform and recognize new actions from only a small number of demonstrations, or from a description alone. Current systems are far from achieving this level of ability, but our work presents a necessary step towards this goal.

1.3 Related Work

Action recognition is a well-studied problem in computer vision which has drawn the attention of researchers for decades. In recent years, approaches towards video action recognition have generally fallen into the framework of deep learning, in which feature representations of videos are learned in an end-to-end fashion, using deep

neural networks trained on large amounts of labeled data. Broadly, this means that there are two main areas in which such deep learning approaches can make progress. First, there are the models themselves, and specifically how these models are designed and optimized for a given dataset. Second, there are the datasets on which these models are trained, and in particular how datasets can be made larger and more varied to meet the needs of data-intensive deep learning models. Here, we give a broad overview of the progress in these two areas in recent years. For a more complete picture of the related work, we additionally provide a related work section in each of the forthcoming chapters.

1.3.1 Models

In the simplest case, models for video action recognition can simply treat a video as a bag of images, paying no attention to the sequence in which these images appear. This approach often provides a strong baseline on many action recognition tasks, but they fail to identify actions for which temporal context is important. To fully reason about videos, models for action recognition must carefully integrate temporal information over a wide range of timescales, all the way from instantaneous motion that occurs over fractions of a second, to long-term dependencies between moments that are minutes or hours apart. In this work, we demonstrate advances in integrating both short-term context (Chapter II) as well as long-term context (Chapters III, IV).

1.3.2 Short-term Context

One common approach is to simply augment single-image CNNs (2D CNNs) to allow for motion feature learning, specifically by changing the input modality to be a representation of local motion such as optical flow. When combined with a standard 2D CNN which takes RGB video frames as input, the approach is called two-stream

CNNs [143]. However, this is not a purely end-to-end approach, and therefore can suffer if the input motion representation fails to capture useful features for action recognition. To remedy this, 2D CNNs can be generalized to include 3D filters (which operate across small groups of adjacent frames) as opposed to 2D (single-frame) filters. This approach is called 3D CNNs [67], and conceptually, 3D filters should allow CNNs to model motion. However, 3D CNNs have many more parameters and therefore require more data to train than their 2D counterparts. Additionally, 3D CNNs lack some of the machinery used in optical flow estimation, specifically the ability to find correspondences between pairs of adjacent frames. Therefore, it is not clear whether 3D CNNs can and do learn sufficiently generalizable motion representations, which is the focus of our work on Distilled 3D Networks (Chapter II).

1.3.3 Long-term Context

Approaches which examine short-term context, such as optical flow estimation and 3D CNNs, do not scale well to long timescales. Therefore, prior work which examine long- and short-term context have taken markedly different directions. To integrate long-term context, prior work often begins with pre-extracted frame-level feature vectors, and treats the sequence of features as a multivariate time series. Commonly, this series is processed using a recurrent model such as an LSTM [58]. In principle, such a model can integrate information over arbitrary long sequences, however, in practice, these models often fail to identify such relationships. Another approach is to use graphical models such as CRFs [80]. However, these approaches generally require the system to solve hard optimization problems during inference that do not scale to arbitrarily-long temporal contexts. In Temporal Hourglass Networks (Chapter III) we provide an end-to-end learnable network architecture which can integrate long-term context at multiple scales throughout a video. In Compositional Tem-

poral Grounding (Chapter IV) we introduce another end-to-end architecture which explicitly integrates information across multiple sub-actions within a long video.

1.3.4 Datasets

Deep architectures for video action recognition require large amounts of data to train. As a result, there has been signficant effort over the past decade to construct ever-larger datasets. In 2011, the state of the art dataset contained fewer than 10,000 videos, while in 2019 the Kinetics-700 dataset was released with 650,000 labeled videos [79, 11]. These advances in dataset size have tracked with performance on many down-stream tasks, and enabled data-intensive models such as 3D CNNs to emerge as the dominant paradigm. However, Kinetics required over 10,000 human hours to annotate [13], and it is unlikely that this figure could be scaled by another factor of 10 or 100, which would require over one million hours of human time.

1.3.5 Webly-Supervised Learning

To combat the need for large labeled datasets, much prior work has turned towards the internet as a source of free labeled datasets. In general, these approaches use metadata found on the Internet, such as using search results, as a form of labels. These approaches have consistently demonstrated that webly-supervised learning is scalable, and has the potential to outperform strongly-supervised methods if given a large enough pool of data. In Web Video Text (Chapter V), we collect the largest video dataset ever used for webly-supervised learning, and demonstrate state-of-the-art results on various downstream tasks.

CHAPTER II

Distilled 3D Networks for Video Action Recognition

Motion is often a necessary cue for recognizing actions. For example, it may be difficult to tell two actions apart from a single frame, like "open a door" and "close a door", because the interpretation of the action depends on the direction of motion. To handle this, recent work treats recognition from motion as its own task, in which a "temporal stream" observes only a hand-designed motion representation as input, while another network, the "spatial stream", observes the raw RGB video frames [143]. However, when the spatial stream is a 3D Convolutional Neural Network, it has spatiotemporal filters that can respond to motion in the video [12, 189]. Conceptually, this should allow the spatial stream to learn motion features, a claim echoed in the literature [169, 90, 122]. However, we still see strong gains in accuracy by including a "temporal" 3D CNN which takes an explicit motion representation, typically optical flow, as input. For example, we see a 6.6% increase in accuracy on HMDB-51 when we ensemble a 3D CNN that takes RGB frames with a 3D CNN that takes optical flow frames [12]. It is unclear why both streams are necessary. Is the temporal stream capturing motion features which the spatial stream is missing? If so, why is the 3D CNN missing this information? In this chapter, we examine the spatial stream in 3D CNNs to see what motion representations they learn, and we

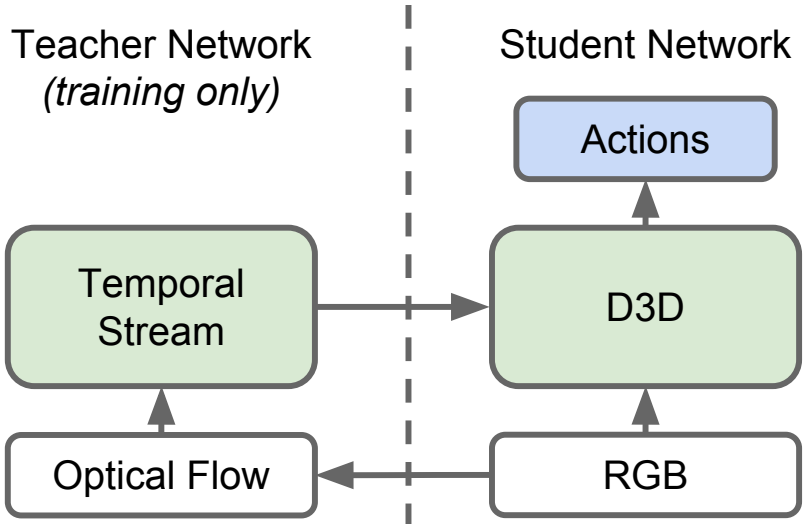


Figure 2.1: Distilled 3D Networks (D3D). We train a 3D CNN (the *student*) to recognize actions from RGB video while also distilling knowledge from a network (the *teacher*) that recognizes actions from optical flow sequences. The teacher network is only used during training, so optical flow is not needed for inference.

introduce a method, depicted in Figure 2.1, that combines the spatial and temporal streams into a single RGB-only model that achieves comparable performance.

Because 3D CNNs include temporal filters, we hypothesize that they should be able to produce motion representations such as optical flow. Recent work has shown that it is possible for 3D CNNs to learn optical flow, but in these studies, the network structure is designed specifically for this purpose [112]. Instead of designing a network specifically for learning motion representations, we study a network that is designed for action recognition, and we test whether it is capable of producing motion representations. To do this, we train 3D CNNs on an optical flow prediction task, described in Section 2.2.1, and we demonstrate experimentally that 3D CNNs are indeed capable of learning motion representations in this way.

However, while 3D CNNs are capable of learning motion representations when optimized for optical flow prediction, it is not necessarily true that these motion representations will arise naturally when 3D CNNs are trained to perform other

tasks, such as action recognition. To answer whether this is the case, we evaluate the same state-of-the-art 3D CNNs on the optical flow prediction task, but we use models with fixed spatiotemporal filters that are trained on an action recognition task. We find that these models underperform those that are fully fine-tuned for optical flow prediction, suggesting that 3D CNNs have much room for improvement to learn higher-quality motion representations.

To improve these motion representations, we propose to distill knowledge from the temporal stream into the spatial stream, effectively compressing the two-stream architecture into a single model. In Section 2.3, we train this Distilled 3D Network (D3D) by optimizing an auxiliary loss which encourages the spatial stream to match the temporal stream's output, a technique often used for model compression [57]. During inference, we only use the distilled spatial stream, and we find that D3D achieves improved performance on the optical flow prediction task. This suggests that distillation improves motion representations in 3D CNNs.

We apply D3D to five datasets using three backbone architectures, and we find in Section 2.5 that D3D strongly outperforms single-stream baselines, achieving accuracy on par with the two-stream model with only a single stream. We train and evaluate D3D on Kinetics [72], and we show that the weights learned by distillation also transfer to other tasks, including HMDB-51 [79], UCF-101 [150], and AVA [48]. D3D does not require any optical flow computation during inference, making it less computationally expensive than two-stream approaches. D3D can also benefit from ensembling for better performance, still without the need for optical flow. We compare D3D to a number of strong baselines, and D3D outperforms these approaches.

In summary, we make the following contributions:

1. We investigate whether motion representations arise naturally in the spatial

stream of 3D CNNs trained on action recognition.

2. We introduce a method, Distilled 3D Networks (D3D), for improving these motion representations using knowledge distillation from the temporal stream.
3. We demonstrate that D3D achieves competitive results on Kinetics, UCF-101, HMDB-51, and AVA, without the need to compute optical flow during inference.

2.1 Related Work

We broadly categorize video action recognition methods into two approaches. First, there are 2D CNN approaches, where single-frame models are used to process each frame individually. Second, there are 3D CNN approaches, where a model learns video-level features using 3D filters. As we will see, both categories of methods often take a two-stream approach, where one stream captures features from appearance, and another stream captures features from motion. Our work considers Two-Stream 3D CNNs.

2.1.1 2D CNNs for Action Recognition

Many approaches leverage the strength of single-image (2D) CNNs by applying a CNN to each individual video frame and pooling the predictions across time [143, 25, 141]. However, naïve average pooling ignores the temporal dynamics of video. To capture temporal features, Two-Stream Networks introduce a second network called the temporal stream, which takes a sequence of consecutive optical flow frames as input [143]. The outputs of these networks are then combined by late fusion, or in other approaches by early fusion, by allowing the early layers of the spatial and temporal streams to interact [28]. Other methods have taken different approaches to incorporating motion by changing the way the features are pooled across time, for example, with an LSTM or CRF [25, 141]. These approaches have proven very

effective, particularly in the case where video data is limited and therefore training a 3D CNN is challenging. However, recently released large-scale video datasets have spurred advances in 3D CNNs [72].

2.1.2 3D CNNs for Action Recognition

Single-frame CNNs can be generalized to video by expanding the filters to three dimensions and applying them temporally, an approach called 3D CNNs [67]. Conceptually, 3D filters should allow CNNs to model motion, but this comes at a cost; 3D CNNs have more parameters and therefore require more data to train. Large-scale video datasets such as Sports-1M enabled the first 3D CNNs, but these were often not much more accurate than 2D CNNs applied frame-by-frame, calling into question whether 3D CNNs actually model motion [71]. To compensate, many 3D CNN approaches use additional techniques for incorporating motion. In C3D, motion is incorporated using Improved Dense Trajectory (IDT) features, which leads to a substantial improvement of 5.2% absolute accuracy on UCF-101 [169, 176]. In I3D, S3D-G, and R(2+1)D, using a two-stream approach leads to absolute improvements of 3.1%, 2.5%, and 1.1% on Kinetics, respectively [12, 189, 171]. The fact that 3D CNNs benefit from a hand designed motion representation suggests that they do not learn to model motion naturally when trained on action recognition tasks. More evidence has shed light on this, for example recent work discovered that 3D CNNs are largely unaffected in accuracy on Kinetics when their input is reversed [189]. In addition, it has been shown that using only a single frame from Kinetics videos with C3D achieves only 5% lower accuracy than using all frames [61]. These results suggest that 3D CNNs do not sufficiently model motion, a hypothesis we explore further in this work.

2.1.3 Uses of Optical Flow

If 3D CNNs do not model motion when trained on action recognition, we naturally ask whether motion is even necessary for this task, and if not, what other benefits optical flow may offer. Recent work has explored several possible explanations for why optical flow is so effective for 3D CNNs [136]. One hypothesis is that optical flow is invariant to texture and color, making it difficult to overfit to small video datasets. To support this, recent work demonstrates that action recognition performance is not well correlated with optical flow accuracy, except near motion boundaries and areas of small displacement [136]. This work, as well as others, have shown that better or cheaper motion representations can be used in place of optical flow, suggesting that, while motion representations are important, optical flow itself is not crucial [27, 199, 204, 38, 136]. However, optical flow has been shown to be useful as a source of additional supervision, which is shown by ActionFlowNet [112]. This work, like ours, trains a 3D CNN to incorporate motion by using an auxiliary task. However, our work uses a different auxiliary task, distillation, which we show is more effective.

2.1.4 Incorporating Motion in 3D CNNs

Many other approaches incorporate motion information into 3D CNNs using changes to the network architecture. Motion Feature Networks, Optical Flow-Guided Features, and Representation Flow all accomplish this by introducing modules into the network which explicitly compute motion representations [90, 161, 122]. These approaches typically add machinery into the 3D CNN architecture that is "missing" from 3D filters, such as the ability to match to motion templates or find correspondences between spatial locations in pairs of nearby frames, which were common in pre-deep learning approaches to action recognition [134]. Alternatively, several ap-

proaches have proposed to replace the optical flow inputs for the temporal stream with a CNN which produces a learned motion representation. For example, Hidden Two-Stream and TVNet use a motion representation that is trained end-to-end for action recognition [27, 204]. In our work, we show that distillation is more effective at improving accuracy than these architectural changes. However, distillation is not in conflict with these changes, and can in fact be applied in combination with any network architecture. Furthermore, the approaches which introduce new modules do not answer whether "vanilla" 3D CNNs are capable of learning motion representations. In our work, we present a study which demonstrates that 3D CNNs do have this ability, and show that distillation improves these representations.

2.1.5 Distillation

In this work we propose to incorporate motion representations into 3D CNNs using distillation. Distillation was first introduced as a way of transferring knowledge from a teacher network to a (typically smaller) student network by optimizing the student network to reconstruct the output of the teacher network [9, 57]. Recent work on distillation has demonstrated that this technique is widely applicable and can be used to transfer knowledge between different tasks or modalities [32, 199, 128, 99, 39]. Our work is related to Motion Vector CNNs, which distill knowledge from the temporal stream into a new motion stream which uses a cheaper motion representation in place of optical flow [199]. By contrast, our work distills the temporal stream into the spatial stream, which allows us to avoid using hand-designed motion representations altogether.

The most similar work to ours is concurrent work on Motion-Augmented RGB Streams (MARS) [20]. This work proposes a similar distillation approach, but ours presents several additional analyses which shed light on the method. Specifically, in

Section 2.2, we propose a flow prediction task to study the motion representation capacity of 3D CNNs, and we demonstrate the effect of distillation on this ability. In addition, we show that our approach can transfer to spatio-temporal action localization (Section 2.5.4) as well as different backbone architectures (Table 2.7). Finally, in our ablation studies in Section 2.5.5 we propose and evaluate some alternatives to distillation, and we show that distillation outperforms these alternatives.

2.2 Motion Representations in 3D CNNs

Two-stream methods rely on optical flow, a hand-designed motion representation, in order to learn features from motion. This begs the question: are 3D CNNs capable of learning sufficient motion representations on their own? To answer this, we train a spatial stream 3D CNN to produce optical flow. If the spatial stream is able to produce optical flow, it suggests that the temporal stream is unnecessary, since it does not have access to any information that the spatial stream cannot learn to produce on its own. On the other hand, if the 3D CNN is not able to produce optical flow, it could be due to one of two possibilities. First, it could be a fundamental limitation of 3D CNNs, that is, they are unable to learn optical flow from video. Second, it could suggest a limitation in the training procedure, that is, they are able to learn optical flow, but do not.

We will show that the second possibility is true: 3D CNNs do not learn motion representations such as optical flow naturally, and the issue lies with the training procedure. Specifically, we demonstrate that 3D CNNs do not learn sufficiently accurate optical flow when trained on action recognition, and that they can learn much more accurate optical flow when trained explicitly to do so.

2.2.1 Optical Flow Decoder

To predict optical flow, we use the hidden features from an intermediate layer in a 3D CNN and pass them through a decoder, as depicted in Figure 2.2. Since our goal is to evaluate the motion representations in the hidden features, we constrain the decoder such that it is unable to learn motion patterns beyond what is already learned by the 3D CNN. Specifically, the decoder contains no temporal convolutions, and operates on a single frame at a time.

In our experiments, the optical flow decoder is designed to mimic the optical flow prediction network from PWC-Net [159], but without the cost volume and warping layers. For more details on the architecture of this decoder, please refer to the appendix.

The output of the decoder is a motion representation introduced by Im2Flow [38], which consists of three channels that encode optical flow: $(mag, \sin\theta, \cos\theta)$, where mag and θ are the magnitude and angle, respectively, of the flow vector at each pixel. The decoder is trained to minimize the squared error between the predicted and target optical flow. For numerical stability, we weight the loss for the $\sin\theta, \cos\theta$ channels by mag. This encoding and training procedure have been shown in prior work to be more effective than directly regressing the optical flow vectors.

To match prior work, we use TV-L1 optical flow [198] as the motion representation [38, 175, 123]. TV-L1 optical flow is commonly used as the input to the temporal stream in many two-stream approaches [12, 136]. Therefore, it is known to be a useful motion representation for action recognition, and reconstructing it with a 3D CNN demonstrates how well the 3D CNN can capture useful motion representations.

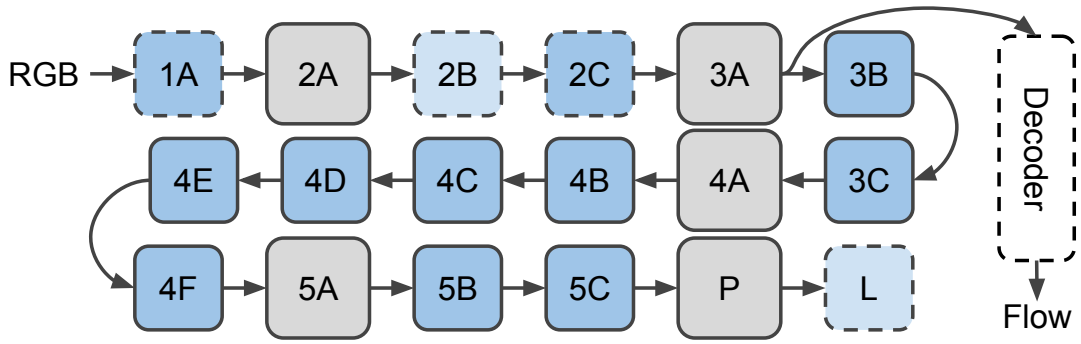


Figure 2.2: The network used to predict optical flow from 3D CNN features. We apply the decoder at hidden layers in the 3D CNN (depicted here at layer 3A). This diagram shows the structure of I3D/S3D-G, where blue boxes represent convolution (dashed lines) or Inception blocks (solid lines), and gray boxes represent pooling blocks [12, 189]. Layer names are the same as those used in Inception [163].

2.2.2 Evaluation Metrics

After training the optical flow decoder, we evaluate the learned optical flow using endpoint error (EPE), a common metric that is adopted in prior work [38, 175, 123].

We evaluate in two settings. In the first setting, we freeze the 3D CNN and train the decoder. This setting tests what motion representations are learned by the 3D CNN naturally by training on action recognition. In the second setting, we fine-tune the decoder and 3D CNN end-to-end. This setting tests what motion representations *can* be learned by a 3D CNN when optimized specifically for this purpose.

In Section 2.5.1, we demonstrate much better results in the second setting than in the first, suggesting there is room for improvement in the training procedure for spatial stream 3D CNNs. We also demonstrate that our proposed distilled method achieves improvements in this direction.

2.3 Distilled 3D Networks

Our goal is to incorporate motion representations from the temporal stream into the spatial stream. We approach this using distillation, that is, by optimizing the spatial stream to behave similarly to the temporal stream. Our approach uses the

learned temporal stream from the typical two-stream pipeline as a teacher network, and the spatial stream as a student network. During training, we distill the knowledge from the teacher network into the student network, as depicted in Figure 2.1. This is accomplished by introducing a new loss function, which penalizes the outputs of the spatial stream if they are dissimilar to those of the temporal stream. More concretely, we train the network parameters θ to minimize the sum of two losses L_a and L_d,

$$L(\theta) = L_a(\theta) + \lambda L_d(\theta) \tag{2.1}$$

where the action classification loss L_a is the cross-entropy and the distillation loss L_d is the mean squared error between the pre-softmax outputs of the spatial stream $f_s(x;\theta)$ and that of the fixed temporal stream $f_t(x)$, i.e.

$$L_d(\theta) = \frac{1}{N}\sum_{i=0}^{N-1}(f_s(x^{(i)};\theta) - f_t(x^{(i)}))^2, \tag{2.2}$$

where $\{x^{(0)}, ..., x^{(N-1)}\}$ are the video clips. The hyperparameter λ allows us to flexibly rescale the contribution of the distillation loss. In our experiments, we find that $\lambda = 1$ conveniently serves as a good setting in many cases. Note that we use a mean squared error loss, as opposed to the cross-entropy loss proposed in prior work [57]. We find that this approach achieves similar results, and can be more flexibly applied to intermediate layers in the network.

We refer to a spatial stream f_s trained using distillation as a Distilled 3D Network (D3D). For inference, we discard the temporal stream f_t, skipping the optical flow step and relying only on RGB input. As we show in Section 2.5, D3D is able to achieve accuracy on par with two-stream methods without the need for two separate spatial and temporal streams. In addition, unlike other approaches for incorporating motion representations, we add no additional computational overhead to the spatial stream [122, 184, 161, 90]. We use S3D-G as the backbone architecture for both

the spatial and temporal stream, since it achieves comparable accuracy at lower computational cost than competing architectures such as I3D and Non-local I3D [12, 184].

2.3.1 Implementation Details

We train D3D in two steps. First, we train the temporal stream using TV-L1 optical flow inputs. Second, we train the spatial stream using the distillation procedure described in Section 2.3. For inference, we discard the temporal stream.

When training the temporal stream, we use the same hyperparameters as those described in prior work [189]. When training the spatial stream, we also use the same hyperparameters as prior work, with the only change being the addition of our distillation loss. We use scaling parameter $\lambda = 1$ unless otherwise specified. We train the model for 140k steps on 56 GPUs with a batch size of 6 clips per GPU. For more details, please refer to prior work on S3D-G [189].

2.4 Datasets

We train and evaluate D3D on several datasets in Section 2.5.

2.4.1 Kinetics

Kinetics is a large-scale video classification dataset with approximately 500K 10-second clips annotated with one of 600 action categories [72, 10]. Kinetics has two variants: Kinetics-600 is the full dataset, and Kinetics-400 is an approximate subset containing 400 categories.

Kinetics consists of publicly available YouTube videos, which can be deleted by their owners at any time. Thus, Kinetics, like similar large-scale Internet datasets, gradually decays over time. Our experiments were conducted on a snapshot of the Kinetics dataset captured in October 2018, when Kinetics-400 contained 226K of

the original 247K training examples (-8.4%) and Kinetics-600 contained 369K of the original 393K training examples (-6.1%). The change in both training and validation sets generates a small discrepancy between experiments conducted at different times. We explicitly denote results on the original Kinetics dataset with an asterisk (*) in all tables and provide the list of videos available at the time of our experiments to enable others to reproduce our results.

2.4.2 HMDB-51 and UCF-101

HMDB-51 and UCF-101 are action classification datasets composed of brief video clips, each containing one action [79, 150]. HMDB-51 contains 7,000 videos from 51 classes, and UCF-101 contains 13,320 videos from 101 classes. For both datasets, we report classification accuracy on the first test split.

2.4.3 AVA

AVA is a large-scale spatiotemporal action localization dataset that consists of 430 15-minute movie clips [48]. Each clip contains bounding box annotations at 1-second intervals for all actors in frame, and each actor is annotated with one or more action labels. In our experiments, we train on AVA v2.1, and report results on the validation set.

2.5 Experiments

In the following experiments, we demonstrate that D3D outperforms single-stream models and achieves accuracy on par with that of two-stream models that require explicit optical flow computation.

Features	Modality	EPE
All zeros	-	2.92
S3D-G	RGB	2.08
D3D	RGB	1.76
S3D-G+FT	RGB	1.34
S3D-G	Flow	0.63

Table 2.1: Effect of feature extractor on optical flow prediction. "All zeros" is a trivial decoder. "S3D-G" and "S3D-G+FT" refer to the 3D CNN with and without end-to-end fine-tuning. We add the optical flow decoder to the "3A" layer of S3D-G and train it to predict optical flow. Fine-tuning vastly improves performance, showing that motion representations can be improved during training.

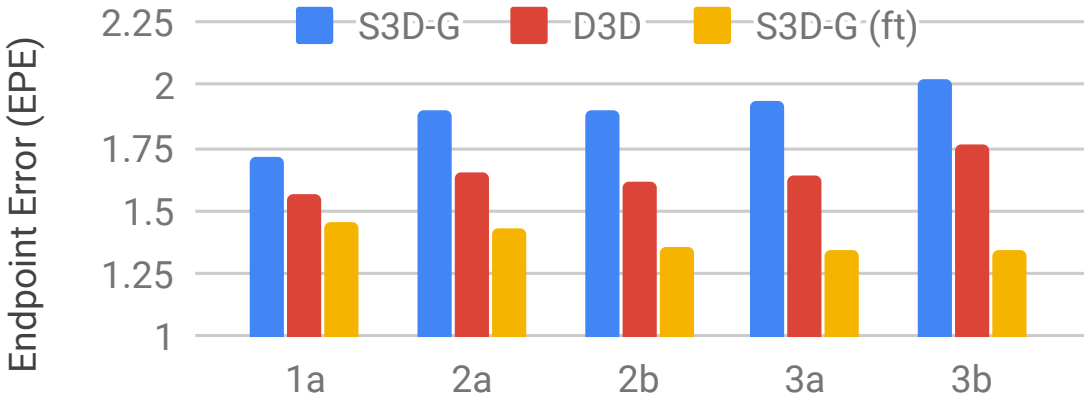


Figure 2.3: Predicting optical flow from multiple layers in S3D-G and D3D. The horizontal axis indicates which layer (see Figure 2.2) is used as input to the decoder. D3D features are able to more accurately reproduce optical flow across the board. Fine-tuning S3D-G end-to-end for flow prediction (indicated "ft") serves as a lower bound.

2.5.1 Predicting Optical Flow

In this experiment, we decode optical flow from the intermediate layers of a 3D CNN as described in Section 2.2.1. For the 3D CNN, we use the spatial stream of S3D-G, which is pretrained on Kinetics-400 and takes RGB videos as inputs. We train the decoder on 2 GPUs with a batchsize of 6 clips per GPU for 100K iterations, and otherwise use the same hyperparameters as S3D-G [189]. We measure performance using endpoint error (EPE) between the predicted and ground truth optical flow.

Fixed vs. Finetuning In Table 2.1, we demonstrate that the decoder can reproduce optical flow, but also that there is significant room for improvement. To bracket performance, we evaluate three baselines: (1) a trivial flow model that predicts "All zeros", (2) a decoder that is trained end-to-end with the 3D CNN, and (3) a decoder trained on the activations of a temporal stream model, which is provided TV-L1 flow as input. Compared to the baselines, the decoder trained on spatial stream S3D-G is able to approximately estimate optical flow. However, we find that the decoded flow is improved by finetuning the model end to end, meaning that motion representations could be improved by changing the training procedure of the 3D CNN.

Distillation and Flow Prediction In Figure 2.3, we compare the flow prediction performance of S3D-G and D3D when the decoder is applied at earlier layers. We observe lower error across the board when attempting to predict optical flow from D3D activations versus S3D-G activations.

While distillation improves optical flow prediction, it does not improve it to the same extent as full end-to-end fine-tuning. This shows that the two objectives, flow prediction and distillation, are complimentary but not completely overlapping. As we will show in Section 2.5.5, distillation improves action recognition accuracy while fine-tuning does not. This result leads to an important finding: improving

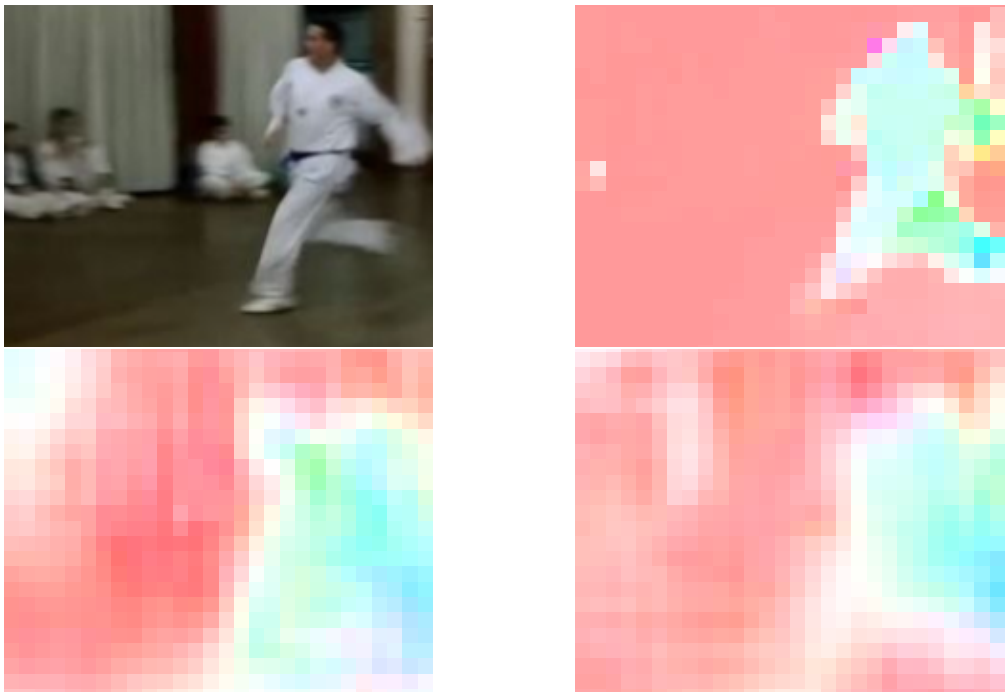


Figure 2.4: Examples of optical flow produced by S3DG and D3D (without fine-tuning) with the decoder applied at layer 3A. Top left: RGB image. Top right: TV-L1 optical flow. Bottom left: S3D-G predicted flow. Bottom right: D3D predicted flow. The color and saturation of each pixel corresponds to the angle and magnitude of motion, respectively. Optical flow is displayed at 28×28px, the output resolution of the decoder. Both S3D-G and D3D miss fine details, but D3D makes fewer mistakes.

motion representations directly does not improve action recognition performance, but improving action recognition performance does improve motion representations. Therefore, in order to improve action recognition performance, it is not sufficient to optimize directly for better optical flow prediction. Distillation takes an alternative approach. By imitating the behavior of the temporal stream, we are able to capture the motion features that are used by the temporal stream while ignoring those that are not.

In Figure 2.4, we give examples of optical flow estimates given using our method. Both S3D-G and D3D can capture coarse motion, but miss fine details. Results using D3D appear to have slightly more accurate motion boundaries, a quality which is known to be useful for temporal stream action recognition [27, 136], explaining the

Method	Modality	Kinetics-400
ARTNet [179]	RGB+Flow	72.4*
TSN [170]	RGB+Flow	73.9*
R(2+1)D [171]	RGB+Flow	75.4*
NL I3D [184]	RGB	77.7*
SAN [8]	RGB+Flow+Audio	77.7*
I3D [12]	RGB	70.6 / 71.1*
I3D [12]	Flow	62.1 / 63.9*
I3D [12]	RGB+Flow	72.6 / 74.1*
S3D-G [189]	RGB	74.0 / 74.7*
S3D-G [189]	Flow	67.3 / 68.0*
S3D-G [189]	RGB+Flow	76.2 / 77.2*
D3D	RGB	75.9
D3D+S3D-G	RGB	76.5

Table 2.2: D3D on Kinetics-400. All numbers given are top-1 accuracy on the validation set. "D3D+S3D-G" refers to an ensemble of D3D and S3D-G. Numbers marked with an asterisk (*) are reported on the full Kinetics-400 set, those without are reported on the subset available as of October 2018 as described in Section 2.4.

quantitative improvements in Table 2.1 and Figure 2.3. We provide more qualitative examples in the appendix.

These results confirm our original hypothesis: 3D CNNs provided with RGB input have a limited natural tendency to capture the motion signal present in optical flow when trained on action classification. The ability to capture motion signal can be significantly enhanced with modified training objectives, such as distillation loss or by fine-tuning for optical flow prediction.

2.5.2 Distillation on Kinetics

Kinetics-400. In Table 2.2, we compare D3D with several competitive baselines. We report accuracy for I3D and S3D-G trained and evaluated on the reduced Kinetics-400 dataset described in Section 2.4. These replications were run with code provided by the original authors and use identical settings to the published papers. Direct comparison with S3D-G shows that the distillation procedure leads to a 1.9% improvement in top-1 accuracy, without any additional computational cost during inference. Per-class accuracy is provided in the appendix. Furthermore, we ensemble

Method	Modality	Kinetics-600
I3D [10]	RGB	73.6 / 71.9*
S3D-G [189]	RGB	76.6
S3D-G [189]	Flow	69.7
S3D-G [189]	RGB+Flow	78.6
D3D	RGB	77.9
D3D+S3D-G	RGB	79.1

Table 2.3: D3D on Kinetics-600. All numbers given are top-1 accuracy on the validation set. "D3D+S3D-G" refers to an ensemble of D3D and S3D-G. Numbers marked with an asterisk (*) are reported on the full Kinetics-600 set, those without are reported on the subset available as of October 2018 as described in Section 2.4. Results on I3D use different settings than in Table 2.2 [10].

D3D with S3D-G ("D3D+S3D-G") by averaging their softmax scores, and achieve a small boost in performance over the two-stream S3D-G approach which uses optical flow. Our ensemble achieves better performance than the two-stream equivalent, without the need to compute optical flow.

Kinetics-600. In Table 2.3, we compare D3D with baseline methods on Kinetics-600. Both the teacher and student network are trained using Kinetics-600 in these experiments. We achieve a 1.3% improvement in single-model performance using D3D, and further improvements by ensembling D3D and S3D-G together, outperforming two-stream S3D-G without the need for optical flow.

2.5.3 Transfer to UCF101, HMDB51

We demonstrate that D3D transfers to other action recognition datasets by fine-tuning D3D on UCF-101 and HMDB-51. For these experiments, we initialize the model using D3D pretrained on Kinetics. However, during fine-tuning, we use only the action classification loss, and not distillation. This avoids the temporal stream altogether, during both training and inference. While we could potentially benefit from applying distillation during fine-tuning as well, these experiments demonstrate that it is not necessary to do so. Each model is fine-tuned for 10k steps on 10 GPUs

Method	UCF-101	HMDB-51
P3D [128]	88.6	-
C3D [169]	82.3	51.6
Res3D [170]	85.8	54.9
ARTNet [179]	94.3	70.9
I3D [12]	95.6	74.8
R(2+1)D [171]	96.8	74.5
S3D-G [189]	96.8	75.9
I3D Two-Stream [12]	98.0	80.7
ActionFlowNet [112]	83.9	56.4
MFNet [90, 122]	-	56.8
Rep. Flow [122]	-	65.4
MV-CNN [199]	86.4	-
TVNet+IDT [27]	95.4	72.6
Hidden Two-Stream [204]	97.1	78.7
D3D (Kinetics-400 pretrain)	97.0	78.7
D3D (Kinetics-600 pretrain)	97.1	79.3
D3D Ensemble	97.6	80.5

Table 2.4: Fine-tuning D3D on UCF-101 and HMDB-51. Our numbers are top-1 accuracy on test split 1 for both datasets. "D3D Ensemble" refers to an ensemble of the two D3D models with different pretraining. No distillation is performed during fine-tuning.

with a batch size of 6 per GPU, as described in [189].

In Table 2.4, we demonstrate that fine-tuning D3D outperforms many competitive baselines. The models in the top section of the table are strong baselines based on 3D CNNs, including S3D-G, which serves as a direct comparison to show that the benefit of distillation during pretraining persists after fine-tuning. The models in the middle section of the table all specifically address the problem of learning motion features without the use of optical flow. D3D outperforms all baselines and achieves essentially equal performance to Hidden Two-Stream when pretrained on Kinetics-400. Hidden Two-Stream uses two I3D models plus an optical flow prediction network, so for fair comparison we also ensemble two D3D models together, and show that this ensemble outperforms Hidden Two-Stream [204].

Method	Pretraining	AVA
I3D w/ RPN [43]	Kinetics-600	21.9
I3D w/ RPN + JFT [43]	Kinetics-400	22.8
S3D-G w/ ResNet RPN [48]	Kinetics-400	22.0
D3D w/ ResNet RPN	Kinetics-400	23.0

Table 2.5: Performance on AVA using different backbone networks. All numbers are frame-mAP on the validation set. Models with "+ ResNet RPN" use a separate pretrained RPN stream based on ResNet, while the others use the 3D features directly for the RPN. The S3D-G baseline includes changes over the previously published numbers, described in Section 2.5.4.

2.5.4 Transfer to AVA

We fine-tune D3D on the spatiotemporal localization dataset AVA, and demonstrate that D3D transfers to this new task. We use a similar approach to the baseline described in the original AVA paper [48], but adopt some changes introduced by a top entry in the 2018 AVA competition [43]. Like the AVA baseline, we use a Faster RCNN-style approach, with a pretrained region proposal network (RPN) based on ResNet, and video feature extractor backbone network based on 3D CNNs. Unlike this work, we use D3D in place of I3D as the backbone network. We also adopt the three key changes introduced in the competition entry [43]. First, we regress only one set of bounding box offsets per region proposal, rather than a different set of offsets per action class. Second, we train for 500k steps using synchronous training on 11 GPUs using a higher learning rate. Third, we add cropping and flipping augmentation during training. Unlike [43], we do not remove the ResNet RPN in either D3D or the S3D-G baseline.

In Table 2.5, we compare the use of D3D as a backbone network with S3D-G and I3D. Our approaches use 50 RGB frames and no optical flow. Direct comparison between S3D-G and D3D shows that using D3D leads to a 1% improvement in Frame-mAP over S3D-G. We also see comparable gains over I3D, and we still outperform the I3D-based approach when it includes additional ResNet features pretrained on

Method	Kinetics-400
S3D-G spatial stream	74.0
S3D-G temporal stream	67.3
S3D-G with 3D CNN flow	69.7
S3D-G with flow loss	74.3
D3D distilled at layer 2C	74.4
D3D distilled at layer 4C	74.5
D3D distilled from spatial stream	74.3
D3D	75.9

Table 2.6: Ablation studies. All numbers given are top-1 accuracy on the reduced Kinetics-400 validation set described in Section 2.4. D3D using our proposed approach outperforms all other approaches listed. See Section 2.5.5 for details.

JFT, an internal Google dataset [158].

2.5.5 Ablation study

In the top section of Table 2.6, we experiment with two alternative approaches to distillation, and demonstrate that D3D outperforms both alternatives. In both cases, we make slight modifications to prior work, described below, to allow for fair comparison with distillation.

S3D-G with 3D CNN Flow. Recent approaches, such as TVNet and Hidden Two-Stream networks, improve the temporal stream by learning their motion representations end-to-end [204, 27]. To compare, we use the first few layers of S3D-G as an optical flow prediction network, and use this learned flow as input to the temporal stream. We use the optical flow prediction network as described in Section 2.2.1, and train this end-to-end with an S3D-G temporal stream. we use S3D-G pretrained to predict actions from optical flow. In our experiments, we find that this approach outperforms the temporal stream applied to TV-L1 optical flow, but still underperforms the spatial stream and D3D.

S3D-G with Flow Loss. Similar to ActionFlowNet [112], we use optical flow prediction as an auxiliary task to improve the spatial stream. We use the flow prediction

network described in Section 2.2.1, but we optimize the model to jointly minimize the flow prediction loss and action classification loss. This is a more direct way of encouraging the network to learn motion representations. However, we find that this does not generally lead to better results on action classification, and distillation gives significantly better results. This is possibly due to the fact that the flow loss is dominated by background pixels, which take up most of the field of view but are not typically important cues for action recognition.

Distillation at Other Layers. The middle section of Table 2.6 demonstrates applying the distillation loss at intermediate layers. We find that applying the distillation loss at intermediate layers is not as effective as at the network outputs.

Distilling from the Spatial Stream. In the bottom section, "D3D distilled from spatial stream" uses the S3D-G spatial stream as the teacher network in place of the temporal stream. This shows that distillation alone does not explain the improvement of D3D over S3D-G. Crucially, we only see benefits when distilling from the temporal stream.

Different Backbones. Distillation is agnostic to the 3D CNN architecture, and therefore can be used in combination with any architecture. In Table 2.7, we show that D3D improves I3D, S3D-G, and a modified version of S3D-G which includes 2 non-local blocks [184]. More details about non-local S3D-G are given in the appendix. In all cases, we use S3D-G as the teacher network, showing that distillation can still work with cross-model transfer.

Ensembling D3D with Spatial and Temporal Streams. In Tables 2.2, 2.3, and 2.4, we demonstrate that it is beneficial to ensemble D3D with an additional spatial stream model. However, in Table 2.8, we find that there is no similar benefit when ensembling D3D with a temporal stream model. This suggests that D3D

Method	Modality	Kinetics-400
I3D [12]	RGB	70.6
S3D-G [189]	RGB	74.0
NL S3D-G	RGB	74.7
D3D (I3D)	RGB	72.3
D3D (S3D-G)	RGB	75.9
D3D (NL S3D-G)	RGB	76.0

Table 2.7: Backbone architectures. All numbers given are top-1 accuracy on the validation set. "D3D (I3D)" and "D3D (NL S3D-G)" refer to D3D with I3D and Non-Local S3D-G as the backbone architectures, respectively. Distillation gives a boost in performance in all architectures.

Method	Modality	Kinetics-600
S3D-G	RGB	76.6
D3D	RGB	77.9
D3D+S3D-G	RGB+Flow	77.6
D3D+S3D-G	RGB+RGB	79.1

Table 2.8: Ensembling. D3D benefits from ensembling with an additional spatial stream, but not a temporal stream.

already captures the signal present in S3D-G Flow, otherwise we would expect to see benefits by performing this ensemble.

2.6 Conclusions

We introduce D3D, a distilled 3D CNN which does not require optical flow during inference and still outperforms two-stream approaches. D3D does not require any changes to the network architecture, and therefore can be used in combination with any backbone network. Furthermore, we show that D3D transfers to other action recognition datasets without the need for further distillation. Finally, we study the ability to predict optical flow with 3D CNNs, and we show that while 3D CNNs have some limited capacity to learn motion representations, D3D improves these representation, and distillation is a more effective objective than directly optimizing for optical flow prediction. Our work shows that the optical flow stream can be discarded during inference for no penalty, calling into question whether optical flow

is really necessary for action recognition. However, further work in this area needs to be done to see whether optical flow can be avoided during training as well.

2.7 Acknowledgements and Citation

Work from this chapter was completed while interning at Google Research, and is co-authored by David A. Ross, Chen Sun, Rahul Sukthankar, and Jia Deng. This work is scheduled to appear in the 2020 IEEE Winter Conference on Applications of Computer Vision (WACV 2020). This chapter can be cited as [153]. We thank our colleagues for their helpful feedback, including Cordelia Schmid, George Toderici, and Carl Vondrick.

CHAPTER III

Temporal Hourglass Networks for Temporal Action Localization

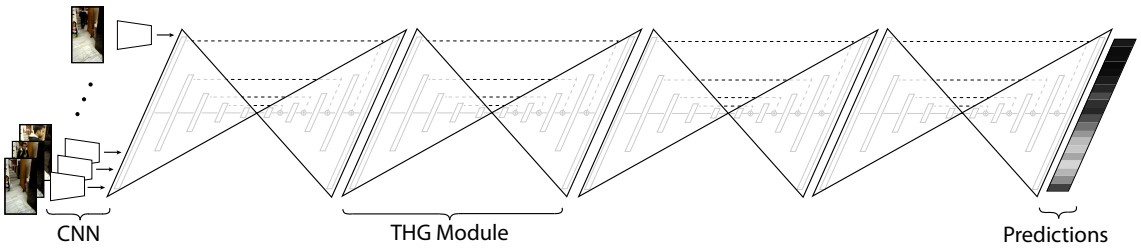


Figure 3.1: Sketch of the Temporal Hourglass Network (THG) architecture. THGs use temporal convolutions at multiple scales to perform video-level inference. We apply multiple THG modules in sequence to perform repeated top-down, bottom-up inference.

3.1 Introduction

In *action detection*, we must not only identify *if* an action occurs in a particular video, we must also identify *when* it takes place. This task requires a rich understanding of the visual cues present in each video frame, such as the pose of the person performing the action and the location of the objects the person is interacting with. However, frame-level features alone are not sufficient to perform this task. In order to perform precise localization of actions, we additionally require contextual understanding of the video that is both *local* and *global*. Local context between neighboring video frames provides cues about instantaneous motion, allowing us to distinguish

between actions like "sit down" and "stand up" that appear (Figure 3.2). Global context provides cues about the natural sequence and interplay between actions in each clip, allowing us to disambiguate actions via their sequential relationships with other actions. By considering both the rich visual characteristics of each individual video frame, as well as their global context in the video, we can achieve the level of understanding necessary to perform accurate action detection.

We propose *Temporal Hourglass Networks (THGs)*, a novel convolutional neural network architecture which is able to reason about videos in both of these crucial aspects (Figure 3.1). Our model builds a rich understanding of an entire video clip by performing temporal convolutions on frame-level image features at multiple temporal resolutions simultaneously, and gradually incorporates these multi-scale features at every timestep. This architecture is a temporal analog to Hourglass Networks [111], which similarly apply convolutions at multiple scales simultaneously, but on images rather than videos. Hourglass networks elegantly incorporate both local and global context for pixel-level prediction in images, making them incredibly versatile. In their original application, hourglass networks were used to predict the location of human pose keypoints [111], but they have since been applied to a number of other tasks, including instance segmentation [110] and depth estimation [16]. These tasks, along with many other tasks in computer vision, require both a global and local understanding of each image. The hourglass network provides both types of context. For this reason, we hypothesize that *temporal* hourglass networks will similarly be able to incorporate both local and global context for action detection in videos.

As is also the case with hourglass networks, our model can be repeatedly applied in sequence, such that the output of each THG is the input to the next. Each THG "module" performs a single round of bottom-up, top-down inference, which

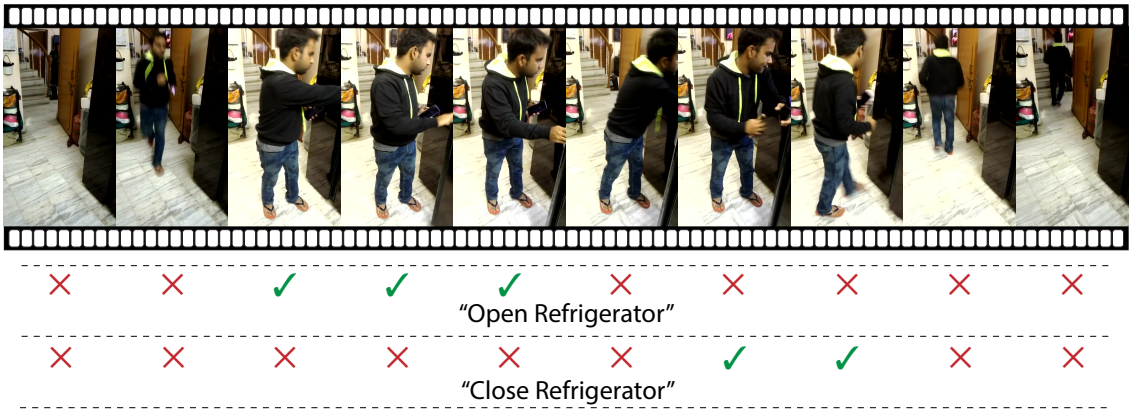


Figure 3.2: Global context in action detection. The natural sequence of actions provides an important cue for detection, in this case "close the fridge" naturally follows "open the fridge".

creates features with access to both local and global context. Repeating this process many times allows many iterations of features to be produced and improved upon during inference. Conceptually, this allows the THG to perform video-level inference *repeatedly*, providing a rich understanding of the global context necessary for action detection.

We implement THGs for action detection and demonstrate their performance on *Charades*, a recent benchmark dataset for action detection [142]. This dataset is particularly challenging, because each video contains several (often overlapping) actions, and occur in typical indoor scenes. This is in stark contrast to prior datasets for action detection, such as *THUMOS* [46] and *ActivityNet* [54], where action instances do not overlap, and actions often take place in atypical environments, such as sports games or musical performances. All action classes in Charades are composed of a verb and an object, and each verb is matched with multiple objects and vice-versa. This ensures that, to classify each action, it is not sufficient to simply detect the corresponding object. This is often not the case in previous action detection datasets, for example in ActivityNet, the direct object "guitar" only appears in

the action "playing guitar", so detecting the guitar object is sufficient to detect the action. This distinction makes Charades an ideal benchmark for action detection.

In summary, this work introduces Temporal Hourglass Networks, a novel convolutional neural network architecture for action detection. Furthermore, we achieve competitive performance on the challenging Charades dataset.

3.2 Related Work

Action detection and recognition have a rich history of prior work [125, 188]. Initial approaches relied on hand-designed descriptors, such as space-time interest points (STIP), Histograms of Optical Flow (HOF), and Motion Boundary Histograms (MBH), among others [85, 74, 86, 21, 114]. More recently, several approaches have relied on mid-level representations constructed from these descriptors [134, 148, 65, 82, 83, 195]. Trajectory-based descriptors, such as Trajectons and Improved Dense Trajectories (IDT) have also shown considerable success [103, 177, 22, 70]. While powerful, these descriptors cannot be learned in an end-to-end fashion, limiting their effectiveness on complex action detection tasks, where the detector must identify actors and objects in a wide variety of scenes and conditions. Despite this limitation, trajectory-based methods long remained the state-of-the-art method for classifying and detecting human actions.

Convolutional Neural Networks. More recently, approaches based on CNNs have seen success because of their ability to be trained end-to-end for action classification and detection. Initial approaches simply classified each frame individually, relying on postprocessing to incorporate temporal context as an afterthought. Many of these approaches are based on the two-stream design of Simonyan and Zisserman, where two parallel CNNs extract features from color frames and optical flow

channels [144]. The reason for incorporating optical flow is that it provides some local context, indicating the direction of motion behind neighboring frames. While optical flow is able to provide some cues about instantaneous motion, it does not provide global context, limiting the amount of temporal context available during prediction [71, 87, 194, 166]. Other versions apply a sliding-window CNN template across the video, often at multiple timescales, and detect actions by classifying each window location [180, 181]. While this is intended to permit inference at multiple time-scales, many sliding-window classifiers simply perform frame-wise classification within the window, the same method employed by two-stream networks, and rely on action duration priors to select which scale to use. In a similar vein, some approaches first predict action proposals, and then classify these proposals using sliding-window architectures [147, 190]. Despite the severe limitations of these frame-by-frame methods, they remain competitive with existing models that account for temporal context.

Graphical Models. Some approaches have leveraged context by modeling the temporal structure of actions and training these models jointly with frame-level CNN features. These models may explicitly define actions as a sequence of action parts [33], use of existing sequence models for language [124, 130], or implicitly model the sequence of actions using graphical models such as CRFs [165, 19, 156, 140]. As in THGs, graphical models are able to reason about video-level context. However, graphical models rely on fixed message-passing updates to perform inference with this context, while THGs learn their own inference function. In principle, THGs can learn to approximate the message-passing procedure, but are strictly more flexible.

Recurrent Neural Networks. Another common approach for incorporating context for action is to use recurrent neural networks, such as LSTMs [157, 146, 151, 25, 197]. In principle, RNNs can remember information over arbitrary long sequences,

and bi-directional RNNs can pass this information both backwards and forwards through the video. However, in practice it is difficult for RNNs to remember information over long sequences, and LSTMs applied to Charades have not outperformed frame-by-frame baselines [140].

3D and Temporal Convolutions. Another class of models, which includes THGs, avoids graphical models and RNNs, and instead uses convolutions to reason about context.

Many of these approaches utilize 3D convolutions [67, 169, 139]. 3D CNNs are difficult to train, as they have many more parameters and require much more memory than traditional CNNs. To combat these difficulties, 3D CNN architectures typically rely on high amounts of temporal downsampling early in the processing pipeline, which limits the temporal resolution afforded by the CNN features. Recently, Shou et al. proposed Convolutional De-Convolutional Networks (CDC), which somewhat avoids this limitation by attaching several layers of successive temporal upsampling and spatial convolutions on top of the first few layers of a 3D CNN [137]. Also notable is R-C3D, which effectively uses 3D CNN features to create coarse feature maps, and uses these to classify and fine-tune action proposals for precise localization [192]. This method, however, relies on scoring a high number of action proposal windows, adding to the high cost of 3D convolutions.

Other approaches use temporal convolutions in favor of 3D convolutions [160, 88]. Notably, Temporal Convolution Networks (TCNs) apply temporal convolutions to frame-wise CNN features in an encoder-decoder style architecture for action segmentation [88]. This allows TCNs to incorporate video-level contex. However, TCNs do not maintain high-resolution skip connections, meaning that they must sacrifice local context in order to incorporate more global context, and vice versa. THGs do

not have to optimize for this tradeoff, because they maintain high-resolution features throughout.

3.3 Temporal Hourglass Networks

We propose Temporal Hourglass Networks (THGs), a novel CNN architecture that detects actions at each frame in a video. The goal of this architecture is to perform repeated inference at multiple temporal scales, which we accomplish with temporal convolutions applied to frame-level image features. Our model first extracts these frame-level image features by applying CNNs to each video frame individually (Sec 3.3.2). Second, we apply several THG "modules" in sequence, which perform repeated inference over the entire video (Sec 3.3.1). Finally, we use the features produced by the final THG module to predict the presence or absence of each action at each video frame (Sec 3.3.3).

3.3.1 THG Module

Each THG module (Figure 3.3) takes as input d features for each of L video frames, which we concatenate to form an $L \times d$ feature map. We successively apply n alternating layers of temporal convolutions and temporal max-pooling, producing a series of feature maps with sizes $\{L \times d, \frac{L}{2} \times d, \frac{L}{4} \times d, \ldots, \frac{L}{2^n} \times d\}$. Each of these feature maps is additionally passed through a *skip connection*, which applies temporal convolutions, but not max-pooling. These skip connections therefore maintain features at a high resolution, which is typically lost to repeated max-pooling. The $1 \times d$ video-level feature vector is then passed through alternating layers of temporal convolutions and nearest-neighbor upsampling until it is restored to its original temporal resolution. At each level during this phase, we add the skip connections element-wise to the corresponding feature map of the same size, which reincorporates

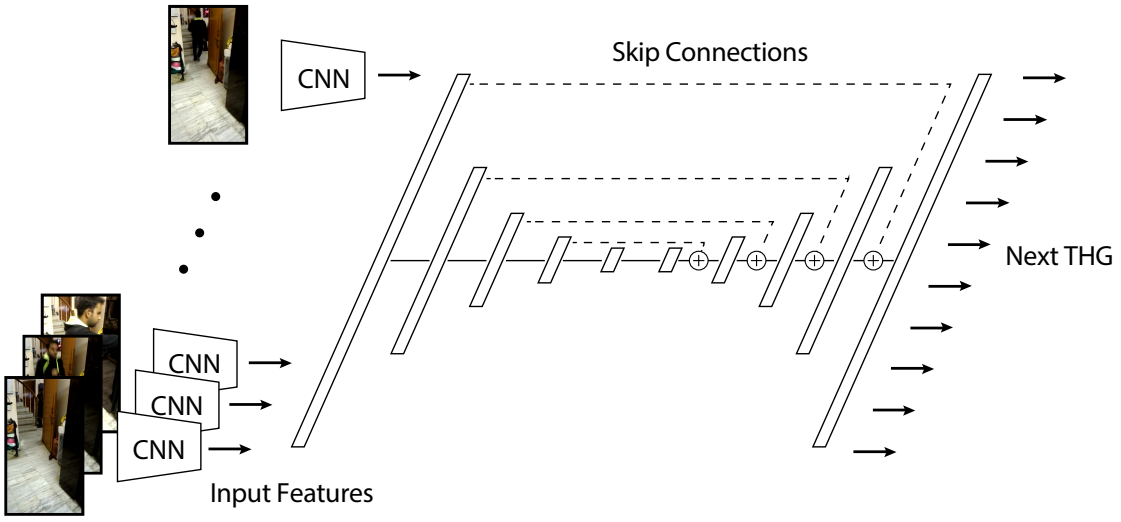


Figure 3.3: Detailed look at a single Temporal Hourglass (THG) module. Each connection between blocks is a residual unit with temporal convolutions, optionally followed either by max-pooling (in the earlier layers) or nearest-neighbor upsampling and element-wise addition (in the later layers, denoted $\oplus$). Skip connections (dashed lines) also contain residual units.

these local features with the global context afforded by the video-level feature vector.

We adopt residual units [53] as the primary building block of the THG architecture. Residual units learn an *additive* transformation of their inputs, that is, $\text{Resid}(x) = x + f(x)$. This has been repeatedly shown to greatly increase performance and ease optimization when training very deep CNNs [52]. In each layer where we apply temporal convolutions, we apply a residual unit with the following configuration: $f(x) : \text{Conv}(1, d/2) - \text{BN} - \text{ReLU} - \text{Conv}(3, d/2) - \text{BN} - \text{ReLU} - \text{Conv}(3, d) - \text{BN} - \text{ReLU}$, where BN is Batch Normalization [63], ReLU is the rectified linear activation, and $\text{Conv}(k, d)$ is a temporal convolutional layer with d filters of width k. The first $\text{Conv}(1, d/2)$ layer serves to perform dimensionality reduction, while the following convolutional layers apply temporal convolutions in this reduced-dimension space.

The complexity of the THG is controlled by several flexible hyperparameters.

This includes the feature dimension d, which we typically set to $d = 256$ and keep constant through all layers of the hourglass module. Additionally, we can control the hourglass depth n, which we set to $n = \log(L)$, so that the feature map at the thinnest part of the hourglass has length $\frac{L}{2^{\log(L)}} = 1$. Finally, we can tune the input length L, which we fix to $L = 64$ during training. Note, however, that THGs are fully convolutional in time, meaning that L can be made arbitrarily large during inference, even when fixed during training.

When we stack multiple THGs, the output of one THG simply becomes the output to the next. Each module has the same structure, regardless of its position in the network. We do not share weights between modules.

3.3.2 Input features

We use the two-stream architecture of Simonyan and Zisserman to extract features from each video frame [144]. This architecture consists of two parallel CNNs, called *spatial-* and *motion*-CNN, which extract features from RGB video frames and dense optical flow, respectively. Both the spatial-CNN and motion-CNN follow the standard VGG16 architecture.

In addition to RGB and optical flow, we include two streams for *pose* and *object* features (Figure 4a). These channels are produced by two standard (spatial) hourglass networks that are trained for human pose estimation and object detection. Intuitively, objects and human pose are useful features for action detection, and we can make use of these pre-existing systems to locate key features in complex videos.

Pose Heatmaps. We detect the locations of 17 human body joints using the multi-person pose system of Newell et al. [110]. This system uses an hourglass network to produce heatmaps which estimate the probability that a given human joint appears at each pixel in the input image. In practice, these estimates are highly accurate,

even in cases of severe occlusion and motion blur.

Object Heatmaps. We produce heatmaps for 45 object classes using the same stacked hourglass architecture of Newell et al. . In these heatmaps, we predict whether a given object class appears at each pixel in the input image. We train this detector is on a collection of object instances from Imagenet [133] and MSCOCO [94], comprising of over 157K examples. These objects were specifically chosen because they appear in the Charades dataset.

Pose and Object Streams. We train two additional CNNs, *pose-CNN* and *object-CNN*, to produce frame-wise feature vectors from pose and object heatmaps. These CNNs have the following architecture: $\text{Conv}(1 \times 1, 64) - \text{BN} - \text{ReLU} - \text{Conv}(1 \times 1, 64) - \text{BN} - \text{ReLU} - \text{Resid}(d) - \text{Pool} - \text{Resid}(d) - \text{Pool} - \text{Resid}(d) - \text{Pool} - \text{Resid}(d)$, where the residual block $\text{Resid}(d)$ is a standard spatial residual unit with d output features.

Late Fusion. We train separate THG architectures for each of the four input feature types. After training, we fuse the predictions of each model by averaging their outputs. In principle, it is possible to train all THG streams jointly along with their respective input CNNs. However, the associated GPU memory costs are prohibitive.

3.3.3 Prediction

Common human actions, like "open a door" are composed of a verb ("open") and a direct object ("door"). This structure is informative, when a less common action class, like "open a refrigerator", has few examples, but shares a concept with a common action class, like "open a door". The verb "open" is shared between these two classes, and therefore we can transfer knowledge between them. We incorporate this structure by predicting actions and their associated verbs and objects separately

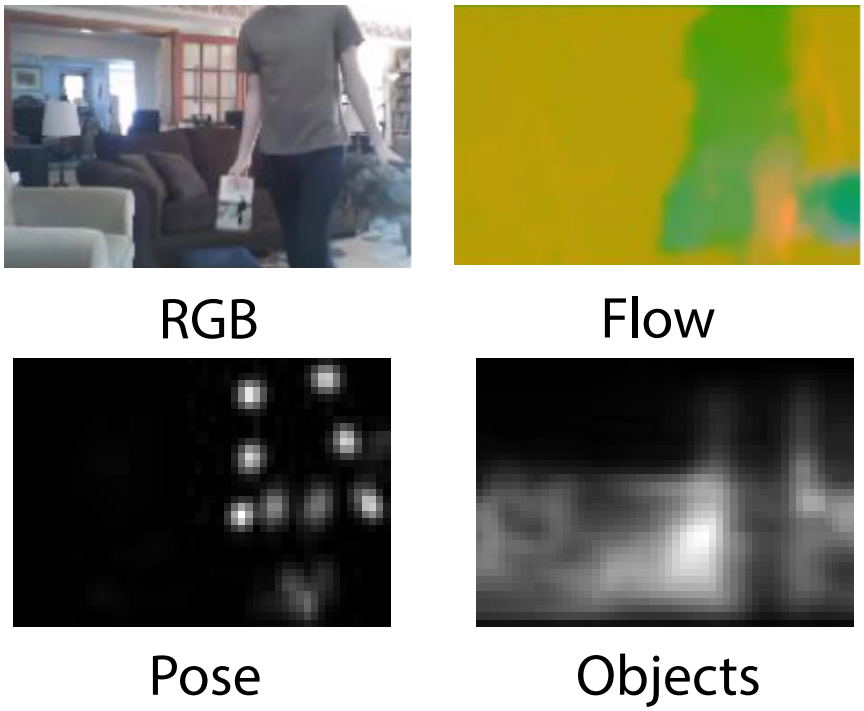


Figure 3.4: Input streams. In addition to RGB frames and optical flow, we include heatmaps for human pose and objects.

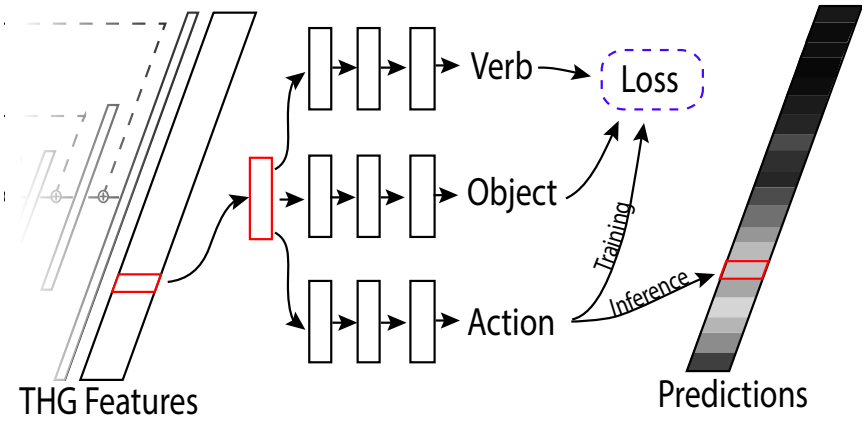


Figure 3.5: Output module. We apply a loss to actions, verbs, and objects separately during training, and only use action predictions during inference.

during training (Figure 4b). For each individual output frame, we apply three parallel output streams, which each consist of three fully-connected layers in the following configuration: FC(256) – BN – ReLU – FC(256) – ReLU – FC(K), where $FC(d)$ is a fully-connected layer with d outputs and K is the number of classes for that particular output stream. During training, we apply a separate loss to each of these

three output streams. During inference, we discard the verb and object predictions, and only consider the action predictions.

3.4 Training Details

During training, the input to the THG is $L = 64$ video frames, which are evenly-spaced from the input video. The THG produces $L \times K$ outputs for $K = 157$ action classes, to which we apply a sigmoid transformation to obtain action predictions.

In the Charades dataset, the median video duration is 30 seconds, meaning that we typically observe frames at slightly higher than 2 frames per second. For each frame, we apply a sigmoid transformation to the network output to produce a prediction for each action between 0 and 1. We then apply a binary cross-entropy loss to each prediction. We use a mini-batch size of 4 videos and optimize using RMSprop [168]. The learning rate is 10^{-4} for 42 epochs and then drops to 10^{-5}. We do not adopt any regularization strategies such as dropout or weight decay, and we train for a maximum of 50 epochs.

Pretraining. Before training the THG, we pretrain the Spatial- and Motion-CNNs on Charades to predict actions frame-by-frame. We then extract their learned features, and train a separate THG for each feature type, without fine-tuning the Spatial-CNN and Motion-CNN. We do not pretrain pose-CNN and object-CNN, and instead train these end-to-end jointly with their respective THG streams.

Data Augmentation. For RGB and optical flow, we extract features for 10 crops in each frame. These include the center crop, 4 corners and their flipped versions. We randomly pick one crop during training, and always use the center crop during testing. We extract 128 frames per video and randomly use 64 of them during training. For object and pose heatmaps, we have random flipping, cropping and

random contrast to each video during training. Each random crop is 85% of the original frame size, and then rescaled with bilinear sampling to a fixed size of 64×64. We randomly perturb the contrast by 15%. We observe that data augmentation is crucial to prevent overfitting.

Intermediate Supervision. When stacking multiple THG modules, we produce predictions independently at each module, and compute a loss for each resulting set of predictions. During backpropagation, we average these losses when computing gradients. Prior work has shown that this approach allows gradient information to more directly reach the earlier layers of the network, and motivates the network to learn relevant features earlier in the pipeline. This strategy was employed previously in stacked hourglass networks [111].

Post-Processing. Prior work on Charades [142, 192] has employed a post-processing step which smooths all predictions by averaging in a temporal window of ± 15 frames around each timepoint. We find that this step slightly reduces the performance of the THG across the board in all experiments. In certain cases, this step has accounted for large improvements in performance (9.6 This is likely because the temporal hourglass already has the ability to reason about local context, and can itself learn to smooth predictions in regions of high uncertainty. We do not use post-processing in experiments using the THG.

3.5 Experiments

In our experimental evaluation of THGs, we perform comparisons with state-of-the-art systems (Sec. 3.5.3) and perform a study on the effect of stacking multiple THG modules with different input features (Sec. 3.5.4).

Our implementation is built in TensorFlow [1] and will be released upon publica-

tion. All experiments are run on a single NVIDIA TitanX GPU.

3.5.1 Charades Dataset

The *Charades* dataset [142] is a recent action detection benchmark which contains nearly 10,000 videos of 157 everyday human actions. Videos in Charades are approximately 30 seconds long and often contain several simultaneous actions from multiple action classes. Each action is composed of one of 33 verbs and one of 38 direct objects, leading to a rich set of action classes. Furthermore, each video is recorded by crowdsourced workers in natural home environments, which leads to a wide variety in scenes and actors across videos. In our experiments, we train and evaluate on the official Charades training and testing splits, which contain 7985 and 1996 videos, respectively.

3.5.2 Evaluation Metric

We report mean Average Precision (mAP), which is the recommended evaluation metric for localization on Charades [140]. We select 25 evenly-spaced time points in each video, that is, $t_0 \dots t_{24}$ such that $t_i = i/T$, where T is the video duration. At each timepoint, we predict a score for each of the 157 actions. Using these scores, we compute the average precision (AP) for each of the 157 action classes individually. To get the mean Average Precision (mAP), we simply take the mean of these computed APs across all classes.

We note that certain classes in this dataset are signficiantly more difficult than others. This is likely due to high class imbalance in the training set, as we observe that the number of positive action instances is highly correlated with the final performance. Because of this, we additionally report the normalized AP (AP_N) for each class, in which the precision is normalized by the average number of positive instances

Depth	mAP
RGB (Single-frame) [140]	8.8
Two-Stream [140]	10.0
Two-Stream+LSTM [140]	8.8
CTF [140]	12.1
R-C3D [192]	12.7
Two-Stream+Audio+Kinetics	17.96
I3D+Kinetics	20.72
THG (ours)	**13.09**
THG++ (ours)	**18.03**

Table 3.1: Comparison with state-of-the-art Mean Average Precision (mAP) on the Charades dataset. Top section: baseline methods. Middle section: top competitors from the Charades Challenge at CVPR 2017. Both competing teams used additional training data from Kinetics. Bottom section: Temporal Hourglass Networks, our submission to the Charades Challenge.

for all classes, rather than total number of positives instances for that class [59]. This has been used for other detection problems to reduce the effect of class imbalance on evaluation.

3.5.3 Comparison with State of the Art

We compare the performance of our full model with that of several established benchmarks (Table 3.1). All results (excluding our own) use the post-processing step described by Sigurdsson et al. [140].

RGB (Single-frame) and *Flow (Single-frame)* denote the performance of Spatial-CNN and Motion-CNN, respectively, applied frame-by-frame to each video. *Two-Stream* is the fusion of Spatial-CNN and Motion-CNN, implemented in the same fashion as Simonyan and Zisserman [144]. *Two-Stream+LSTM* applies a single LSTM layer on top of the Two-Stream features. These baselines are adopted from the implementation of Sigurdsson et al. [140].

Connected Temporal Fields (CTF) [140] employs a fully-connected CRF on top of the Two-Stream video features. This CRF reasons jointly about many aspects of each action, including the verb, object, scene, and the intent of the human actor.

	RGB	Flow	Pose	Objects	Two-Stream	All
Single-Frame	7.7	4.9	5.24	4.20	7.7	-
Single-Frame w/ Post	**8.8**	6.6	5.80	4.39	10.0	-
Ours	8.61	**9.31**	**6.31**	**4.55**	**11.38**	**13.09**

Table 3.2: Effect of input streams. THG improves performance over frame-by-frame methods for all input modalities except RGB.

Region Convolutional 3D Network (R-C3D) [192] is another recent benchmark that uses 3D-CNN features to classify action proposals.

3.5.4 Ablation Study

In these experiments, we perform a comprehensive evaluation of the hyperparameter choices that are involved in designing the THG module. In addition, we explore the effect of the individual input streams and their fused counterparts.

Input Streams. We compare the performance of the single-stack THG with a frame-by-frame classifier (with and without post-processing) for each of the four input modalities in Table 3.2. In addition, we compare fused models for both the Two-Stream modalities (RGB and Flow) and all four modalities together.

We find that the THG module improves upon the single-frame classifier for all four input types. In addition, we find that pose and objects alone are less predictive than RGB and optical flow. However, when we ensemble all four modalities, we are able to improve upon the performance of the two-stream THG.

Stacking. We train four variants of the THG architecture, each with an additional stacked THG module. To ensure that each architecture has equal capacity, we tune the feature dimension d such that each variant has a roughly equal number of total free parameters. In all experiments, we use intermediate supervision, and all four input modalities with late fusion.

We find that THGs do not benefit from an increased number of stacked hourglass

# of THGs	# features	RGB+Flow	All
1	256	11.38	**13.09**
2	200	8.7	10.37
3	175	8.11	8.72
4	150	6.52	7.60

Table 3.3: Effect of stacking. We control the feature dimensionality for multi-stack hourglasses such that each THG has the same order of parameters. We find that adding multiple hourglass modules reduces performance.

Action Class	RGB	RGB+Pose	Change
Sitting at a table	6.80	36.82	(+30.02)
Sitting on sofa/couch	6.15	32.64	(+26.49)
Lying on a bed	5.59	30.84	(+25.24)
Sitting in a chair	7.62	32.44	(+24.82)
Someone is going from standing to sitting	5.65	30.27	(+24.62)
Action Class	RGB+P	RGB+P+O	Change
Opening a refrigerator	8.92	31.33	(+22.40)
Someone is cooking something	23.95	45.82	(+21.87)
Lying on a sofa/couch	20.86	42.68	(+21.82)
Washing a window	9.84	27.46	(+17.62)
Someone is going from standing to sitting	30.27	47.72	(+17.45)

Table 3.4: Effect of Pose and Object heatmap features on performance of individual action classes. We show the five classes with the highest absolute improvement in normalized Average Precision (AP_N). (Top) Pose features are helpful for detecting sitting, standing, and lying down. (Bottom) Object features are useful for detecting interactions with objects.

modules (Table 3.3). This is in stark contrast to prior work on spatial hourglass networks, where increased stacking significantly improved performance [111]. While our experiments control for the total number of parameters, it is possible that the increased number of stacked modules leads to more extreme overfitting.

Additionally, we report performance for each successive module individually, and find that each module has similar performance. This also contradicts the results of spatial hourglass networks, where it was found that each successive module typically has better performance than the last.

3.6 Conclusions

We propose *Temporal Hourglass Networks*, a CNN architecture that uses temporal convolutions to learn from both local and global context in videos. We find

that Temporal Hourglass networks achieve competitive performance on the Charades action detection dataset. While we find that stacking THGs leads to decreased performance, we show that we can leverage features from human pose and object detectors to further improve performance.

3.7 Acknowledgements and Citation

Work from this chapter was published in the proceedings of the Visual Understanding Across Modalities Workshop as part of the Charades Challenge at the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017). This work was co-authored by Kaiyu Yang, Hei Law, and Jia Deng.

CHAPTER IV

Compositional Temporal Visual Grounding of Natural Language Event Descriptions

In this chapter, we consider the task of temporally grounding natural language event descriptions in videos. In this task, we are given a video and a natural language query, and we must locate the point in time that corresponds to the query. For example, we may have a video of a baseball game, along with the query "the batter hits the ball, and a player tags him out at first base." The system must find the point in time that best corresponds with the event described in the query. A system that can accomplish this task would have many applications, such as video retrieval and human-robot interaction.

Like many tasks at the intersection of vision and language, temporal grounding requires that we generalize to both unseen videos *and* unseen queries. Since we make no assumptions about the content of a particular video or query, it is possible for them to depict a completely novel scene or event. This presents a challenge, and one way to overcome it is to leverage our prior knowledge; these modalities have structural properties which can be encoded into a model, which allows us to make better use of training data and generalize more effectively. In our work, we focus on two structural properties inherent to temporal grounding, which we leverage in our model.

Figure 4.1: Example query and video. Queries are often *compositional*, and they impose a *temporal ordering* over their components.

The first property is *compositionality*, that is, a single query may be composed of many events. Consider the example in Figure 4.1, which contains two events ("walks by" and "touches"). Conceptually, there is no limit to the number of events that a query could describe. We can take advantage of this structure by breaking up a query into atomic components, and localizing each of them individually. This makes the problem more manageable; while the full query may be a novel combination of components, it is likely that we will have seen some of these components before.

The second property is *temporal ordering*, that is, each query designates a particular ordering of its components. Often, the events must occur in the video in the same order that they appear in the sentence, like in "*this* then *that*". But this is not always the case, as in the example in Figure 4.1, where the order is reversed. Determining the ordering is non-trivial, as natural language is complex and orderings can be implied from context. However, a model that can determine this ordering can then use it to its advantage, as it allows the model to prune spurious detections which do not match the temporal constraints.

We propose a model, called Compositional Temporal Grounding Network or CTG-Net, which explicitly leverages both compositional and temporal structure for temporal grounding, depicted in Figure 4.2. Specifically, our model first segments the query into discrete sub-events, leveraging compositionality (Section 4.2). Next, we

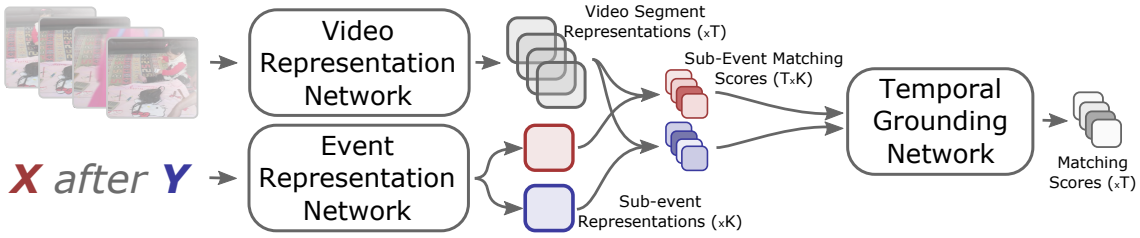


Figure 4.2: Overview of our proposed CTG-Net. We represent the input query as several sub-events (depicted as red and blue boxes) using our Event Representation Network (Section 4.2). We match these sub-events to video segments, and then combine and refine these matchings using our Temporal Grounding Network (Section 4.3).

ground each sub-event in the video, and then refine these groundings to enforce temporal ordering (Section 4.3). We apply our model on three temporal grounding datasets: DiDeMo, TEMPO-TL, and TEMPO-HL [55, 56], and demonstrate that our approach outperforms state-of-the-art methods (Section 4.4). Concretely, we improve Recall@1 from 26.8% to 28.7% on TEMPO-TL and from 20.8% to 21.5% on TEMPO-HL.

4.1 Related Work

Temporal grounding is a recently-introduced task, and it is often referred to in prior work as moment retrieval or action localization with natural language queries [55, 37]. Prior work on this task generally takes one of two approaches: sliding window or single-shot.

Sliding Window. In this approach, we embed the query and video segments into the same low-dimensional space, such that the query embedding is similar to the video embedding for the correct segment, and dissimilar for other segments. To ground a query, we slide its embedding across the video and find the most similar segment. Three notable sliding window approaches include Moment Context Networks (MCN) and Moments Localized with Latent Context (MLLC) by Hendricks

et al. [55, 56], Cross-modal Temporal Regression Localizers (CTRL), introduced by Gao et al. [37], and Activity Concepts-based Localizer (ACL), by Ge et al. [40]. While these approaches have been successful, they fail to account for compositional events. MLLC makes a step towards this goal, by representing each query as two sub-events, the "base" and the "context". However, it is restrictive to assume that every event description has exactly two sub-events. Our approach lifts this restriction by allowing any number of sub-events.

Single-Shot. In this approach, we compare each token in the query to each segment in the video, and then aggregate these comparisons to classify each segment as a correct or incorrect grounding. This includes Temporal GroundNet (TGN), by Chen et al. [15], Moment Alignment Networks (MAN), by Zhang et al. [200], and Temporal Modular Networks (TMN), introduced by Liu et al. [95]. These approaches do leverage compositionality, but only at the level of individual words, and they generally do not account for temporal relationships. Temporal Modular Networks make significant improvements in terms of compositionality, in that they use the parse tree of the query to gradually aggregate more refined groundings. However, TMN is limited by the use of a fixed dependency parser, while we demonstrate that our method can work both with a fixed parser or in an end-to-end architecture.

Compositional Representations. It is well known that events are compositional, and this observation has inspired many prior works in action recognition [131, 34, 93], captioning [193], and temporal grounding [95]. Leveraging compositional structures is central to many tasks in computer vision [29]. To the best of our knowledge, our work is the first to leverage explicit compositional structure for temporal grounding.

Temporal Relationships. There is a large body of work that addresses temporal relationships in natural language, enabled by corpora with labeled temporal rela-

tions [127, 108]. Prior approaches have used formal logic [126, 76] in addition to machine learning [101, 84, 14]. Few papers have approached this problem utilizing the most recent developments in deep learning for NLP, however there has been some recent work on determining temporal ordering from clinical notes [129, 66]. We provide a mechanism for incorporating temporal relations which does not rely on hand-designed features or formal logic, by creating a "position embedding" for each sub-event. We find that this mechanism, while simple, is effective for reasoning about temporal relations.

Weakly-Supervised Temporal Localization. Our work is related to the task of weakly-supervised temporal localization, a well-studied problem in which a system must learn to perform temporal action localization when given only video-level labels [120, 138, 113, 182, 60]. Similarly, in our work, we do not have individual temporal labels for each atomic sub-event, and must learn to localize these without labels. However, we do have temporal labels for each query. In addition, these prior works use action category labels, as opposed to natural language queries, as the description for each labeled event. Only recently has weakly-supervised temporal localization been studied in the context of natural language queries [164].

Relation to MCN and MLLC. Our work is a generalization of Moment Context Networks (MCN) [55]. Specifically, we use the same encoders for embedding words (Section 4.2.2) and video segments (Section 4.2.3), as well as the same distance metric to compute matching scores (Section 4.3). We generalize MCN by allowing multiple sub-events to each be localized simultaneously, incorporating compositional structure, and later combining and refining their matching scores, incorporating temporal structure. Moments Localized with Latent Context (MLLC) [56] similarly builds off of MCN, but is limited to localizing just two sub-events, and does not include

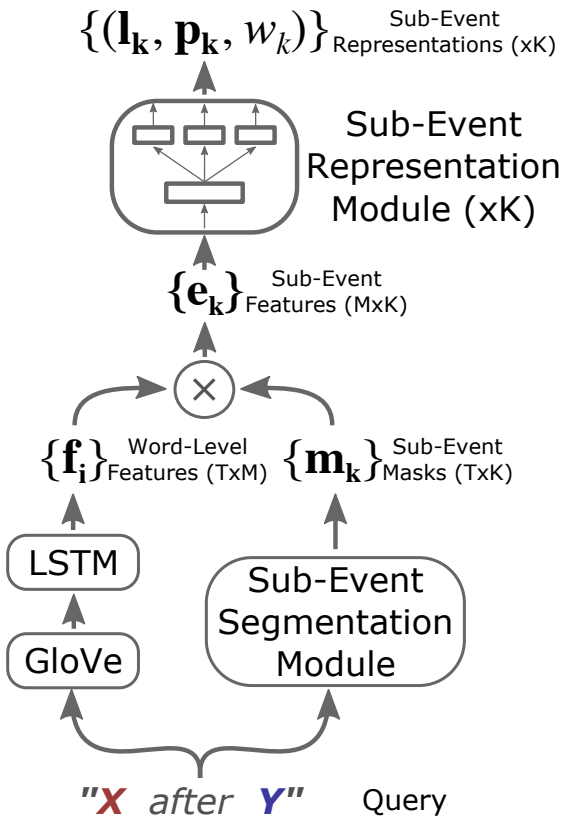


Figure 4.3: Event Representation Network with the parser approach. The Sub-Event Segmentation module produces masks that indicate which words belong to each detected sub-event (Section 4.2.1). The Sub-Event Representation module produces a triplet representation for each sub-event (Section 4.2.2).

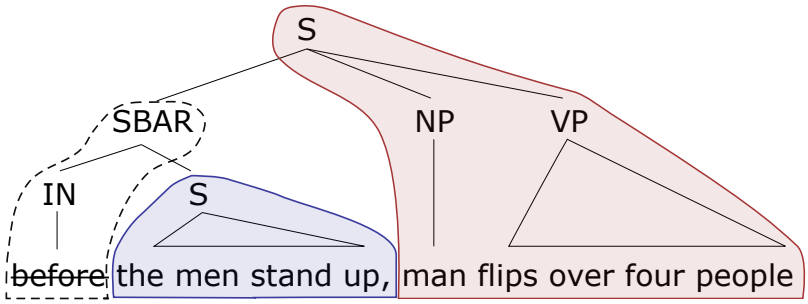


Figure 4.4: Result of segmenting sub-events with a parser. This query contains three clauses (with tags S, SBAR, and S). Each word is assigned to the lowest-level clause to which it belongs, and length-1 clauses are discarded. Two sub-events are detected in this example, depicted by the red and blue outlines.

temporal refinement. Our model, therefore, is more flexible and performs well with both simple and complex queries.

4.2 Event Representation Network

We represent each natural language query as a set of one or more sub-events, where the number of sub-events is chosen flexibly depending on the query. We produce this representation using a novel network architecture, which is depicted in Figure 4.3 and has two primary components. The first component is the *sub-event segmentation module*, which determines the number of sub-events as well as which words belong to each. The second component is the *sub-event representation module*, which creates a vector representation for each sub-event.

4.2.1 Sub-Event Segmentation Module

We propose two methods for segmenting a query into sub-events.

Parser. In this approach, we use the Stanford Parser [75] to segment the sentence into clauses, and we consider each clause to be a sub-event. We define "clauses" to include all clause-level tags in the Penn Treebank (S, SBAR, SINV), as well as fragment tags (FRAG). Since clauses can themselves contain subordinate clauses, we assign each word to the lowest-level clause to which it belongs. For an example of

this, see Figure 4.4. We discard all sub-events that contain only one word, as these are typically due to a conjunction such as "after," which we do not consider to be events.

Bi-directional LSTM. While the parser approach allows us to segment reasonable sub-events, it imposes some limitations. First, all sub-events are contiguous in the query, and cannot overlap. This introduces an issue when a clause contains a pronoun, as we cannot resolve the reference of the pronoun without the broader context of the sentence. By removing the restrictions of contiguity and overlap, we can replace a pronoun with its referent, making it more easy to localize. Another limitation is that the parser is fixed, and therefore any noise in its output cannot be fine-tuned away during training.

As a flexible and end-to-end learnable alternative to the parser, we propose a simple attention mechanism using a bi-directional LSTM. We feed the word-level features $\mathbf{f_i} \in R^M$ (Section 4.2.2) into the Bi-LSTM, and then apply K linear classifiers to its output to get a set of attention masks corresponding to K sub-events. Unlike in the parser approach, this always results in an equal number of sub-events. Specifically, we set $K = 6$, since this is the maximum number of sub-events identified by the parser in our experiments. However, as we will explain in the following section, each sub-event is later associated with a weight, and it is ignored when its weight is zero. This way, we still can flexibly represent the query as any number of sub-events.

Output. Mathematically, both segmentation approaches result in a set of masks $\mathbf{m_k} \in [0, 1]^N$ for $k \in \{1, \ldots, K\}$, where N is the number of tokens in the input query and $K \leq 6$ is the number of detected sub-events.

4.2.2 Sub-Event Representation Module

To create the sub-event representations, we first compute GloVe embeddings [121] for each word in the query, and then pass these into an LSTM [58]. The output of the LSTM is a sequence of feature vectors $\mathbf{f_i} \in \mathbb{R}^M$ for $i \in 1, \ldots, N$ where N is the number of words and M is a hyperparameter corresponding to the hidden feature dimension. To convert these word-level features to sub-event-level features, we pool them via a weighted average using the sub-event masks from Section 4.2.1. That is, $\mathbf{e_k} = \sum_{i=1}^{N} m_{ki}\mathbf{f_i}$, resulting in $\mathbf{e_k} \in \mathbb{R}^M$, the feature vector for the k^{th} sub-event.

We then create a triplet representation for each sub-event. Specifically, we represent the sub-event k as $(\mathbf{l_k}, \mathbf{p_k}, w_k)$, where $\mathbf{l_k} \in \mathbb{R}^{M_{\text{embed}}}$ is the language embedding, $\mathbf{p_k} \in \mathbb{R}^{M_{\text{pos}}}$ is the position embedding, and $w_k \in [0, 1]$ is a scalar weight. The language embedding $\mathbf{l_k}$ is used to co-embed the sub-event features in the same space as the visual features in Section 4.2.3. The position embedding, $\mathbf{p_k}$, represents the position of the sub-event in time, and is used to enforce temporal consistency in Section 4.3.1. The weight w_k allows the model to ignore sub-events which describe something non-visual, or sub-events which are used purely as context for describing another event (Section 4.3). Each of these embeddings is created by passing the sub-event features $\mathbf{e_k}$ through a single fully-connected layer. We normalize each $\mathbf{l_k}$ with L2 normalization, and we normalize the weights w_k with a softmax across the K sub-events.

4.2.3 Video Representation

We represent each video segment as an embedding $\mathbf{v_t} \in \mathbb{R}^{M_{\text{embed}}}$, where $\mathbf{t} = (s, e)$ is the start and end-point of the segment. To create the embeddings, we adopt the approach introduced by [55]. In this approach, either RGB or Optical Flow

frames are passed into a CNN to create frame-level feature vectors. These features are averaged within each segment $\mathbf{t}$ to create *local* features, and averaged across the entire video to get *global* features. We then concatenate the local and global feature vectors along with the start and end-points (s, e) (called temporal end-point features, or TEF) into a $(2D_{\text{video}} + 2)$-length feature vector, where D_{video} depends on whether RGB or Optical Flow features are used. This vector is then fed into a 2-layer multi-layer perceptron (MLP) to get the video segment embedding $\mathbf{v_t}$.

4.3 Temporal Grounding

When queries are compositional, it is not always straightforward to define the expected output of temporal grounding. Consider a query of the form "X after Y." Should the grounding include the temporal extent of both X and Y, or only X? Some datasets (DiDeMo [55]) encode the answer using the former option, opting to ground the full extent of events described by the query. Other datasets (Tempo-TL, Tempo-HL [56]) however, use the latter, since Y in this case is used only to refer to a particular instance of X. We propose a method which is agnostic the particular grounding scheme and can be trained end-to-end for either scenario.

Intuitively, our proposed temporal grounding procedure leverages the sub-event representations, as well as their temporal ordering, to find a matching between the query and the video. We first attempt to localize each individual sub-event in the video. Then, we apply a *refinement network* which updates these locations to ensure temporal consistency, and combines them to compute the final grounding. Concretely, this network takes as input the sub-event representations (Section 4.2.2) and video segment representations (Section 4.2.3), and as output, it produces a score for each time segment t, which corresponds to how well that time segment matches the

query.

To locate each sub-event k, we compare their embeddings $\mathbf{l_k}$ with the embedding of each video segment $\mathbf{v_t}$. We compute the Euclidean distance between each pair, that is, $d_{kt} = \|\mathbf{l_k} - \mathbf{v_t}\|_2$, where pairs with smaller distances between them are considered better matches. The distance d_{kt} therefore serves as a matching score between sub-event k and video segment t.

We then combine these matching scores across sub-events to compute an initial matching score for the entire query. Specifically, we perform a weighted average of the sub-event matching scores, where the weights are given by w_k from the event representation network, that is, $D_t = \sum_{k=1}^{K} d_{kt} w_k$. The weights allow the matching to favor particular sub-events over others, or exclude a sub-event entirely. This is helpful when a sub-event is used only for context, or when a sub-event is not visible in the video.

4.3.1 Refinement Network

The initial matching score D_t accounts for compositional structure, but does not account for *temporal* structure. To leverage temporal structure, we propose an additional step, where the matching scores and expected positions of each sub-event are used to update the matching score for the entire query. This phase allows the model to downweight segments that do not match the expected position of each sub-event in the video, which is encoded using the position embedding $\mathbf{p_k}$ (Section 4.2.2).

More precisely, for all video segments t, we apply a refinement function ϕ which takes as input the matching score D_t, the matching score for a particular sub-event d_{kt}, the position embedding for that sub-event $\mathbf{p_k}$, and the temporal extent of the segment $t = (s, e)$. We apply the refinement function to each sub-event k and add the results to the matching score D_t to get a refined score $\widetilde{D}_t$. That is,

$$\widetilde{D}_t = D_t + \sum_{k=1}^{K} \phi(D_t, d_{kt}, \mathbf{p_k}, t). \tag{4.1}$$

For the refinement function ϕ, we use a 2-layer MLP. The inputs are concatenated along the feature dimension, which results in an input with dimension $1+1+M_{embed}+2$. The final matching score $\widetilde{D}$ is used to rank the candidate segments, and as the final grounding we choose the segment that results in the lowest distance $\widetilde{D}$. That is, $\hat{t} = \arg\min_t \widetilde{D}_t$.

4.3.2 Training

CTG-Net is fully end-to-end differentiable and can be optimized via gradient descent. We adopt the triplet ranking loss $\mathcal{L}_{\text{triplet}}$ from Hendricks et al. [55] and apply it directly to the refined matching scores $\widetilde{\mathbf{D}}$. This loss imposes a penalty when the score for an incorrect segment $\widetilde{D}_{t'}$ is lower than that of a correct segment $\widetilde{D}_{t'}$ (recall that lower scores are better), or if they are within some margin b, that is, $\mathcal{L}(\widetilde{D}_t, \widetilde{D}_{t'}) = \max(0, \widetilde{D}_t - \widetilde{D}_{t'} + b)$. For every positive example, we compare it with two negative examples: one is a segment chosen randomly from the same video $\widetilde{D}_{t_{\text{intra}}}$, and the other is a segment chosen from a random video at the same timepoint $\widetilde{D}_{t_{\text{inter}}}$. The loss is a weighted sum of the ranking losses for these two negative examples, that is, $\mathcal{L}_{\text{triplet}} = \mathcal{L}(\widetilde{D}_t, \widetilde{D}_{t_{\text{intra}}}) + \lambda\mathcal{L}(\widetilde{D}_t, \widetilde{D}_{t_{\text{inter}}})$.

Implementation Details. We optimize CTG-Net using stochastic gradient descent with a batch size of 120, an initial learning rate of 0.05, and we decay this learning rate by a factor of 10 every 33 epochs (50 for Tempo-HL). We multiply the learning rate by a factor of 10 when updating the LSTM used to create word-level features. We train each network for a maximum of 100 epochs, with early stopping if the validation accuracy plateaus.

We choose hyperparameters from MCN [55] where applicable, that is, the word feature dimension $M = 1000$, the embedding dimension $M_{\text{embed}} = 100$, the inter-video loss weight $\lambda = 0.2$. The remaining hyperparameters, including the dimension of the position embedding $M_{\text{pos}} = 100$ and the size of the hidden layer in the refinement MLP $M_\phi = 100$, are selected using the validation set.

As in Hendricks et al. [55], we divide each video into 5-second clips, and consider all segments which consist of one or more contiguous clips. For a 30-second video, this gives 6 possible segments of length 1, 5 possible segments of length 2, and so on. The number of segments for a video with T clips is therefore $T(T+1)/2$.

Late Fusion. We use two sets of visual features, one extracted from RGB video frames, and the other extracted from Optical Flow sequences. For RGB, we pass each frame of the video through a pretrained VGG16 network up to the *fc7* layer [145], producing a feature vector for each frame. For Optical Flow, we use the penultimate layer of a Temporal Segment Network [183] trained on action recognition. For each experiment, we train two networks, one for each of the two visual modalities, and then perform a weighted average of their refined correspondences $\lambda_{\text{RGB}}\widetilde{D}_{\text{RGB}} + (1 - \lambda_{\text{RGB}})\widetilde{D}_{\text{Flow}}$, $\lambda_{\text{RGB}} = 0.3$ to get a fused result. All results include late fusion.

Code. Our implementation is built in PyTorch [118], and uses GloVE embeddings and visual features (both RGB and Flow) provided by the original creators of the datasets [55, 56]. Our implementation will be made publicly available.

4.4 Experiments

DiDeMo. Distinct Describable Moments (DiDeMo) is a recent dataset introduced by Hendricks et al. [55] as a benchmark for temporal grounding. This dataset consists of over 10K unedited videos from Flickr and 40K unique queries. Each query

describes a distinct moment in the corresponding video, and its temporal location is annotated independently by four annotators. The queries describe a rich range of events that are not limited to human activities, such as camera and object motion. The dataset includes a high percentage of queries (18.4%) that include words about temporal relationships, such as "first", "begin", "after", and "final". These properties make DiDeMo a realistic benchmark dataset for temporal grounding.

Tempo-TL and Tempo-HL. Temporal Template Language (Tempo-TL) and Temporal Human Language (Tempo-HL) are two recent temporal grounding datasets which build off of DiDeMo [56]. These datasets contain complex queries which are constructed specifically to test compositional grounding, making them an ideal testbed for our method.

Tempo-TL queries are procedurally constructed from pairs of DiDeMo queries. They are constructed using one of five templates: *X before Y*, *Y, before X*, *X after Y*, *Y, after X*, and *X then Y*, depending on the temporal relationship of the two events. While this procedure sometimes results in unnatural sentences, it enables fine-grained evaluation of grounding under specific temporal relationships. Depending on the template, the system must localize the base moment X ("before" and "after"), or the concatenation of both events X and Y ("then").

Tempo-HL queries are constructed by asking each annotator to describe an event *relative* to an existing query from DiDeMo. The resulting queries include all of the same temporal relationships as Tempo-TL, as well as "while" relationships, which are not covered by Tempo-TL. Because these queries are rewritten in natural language from scratch, they include coreference statements and a wider range of temporal prepositions. Tempo-HL is therefore a much more realistic and challenging dataset.

DiDeMo+Tempo. When training on Tempo-TL or Tempo-HL, we also include

Method	Tempo-TL Splits - R@1				**Average**	
	DiDeMo	Before	After	Then	R@1	R@5
Prior	10.7	17.9	22.4	0.0	12.7	52.6
MCN [55]	24.9	32.3	25.1	27.1	73.4	26.1
TALL [37]	21.0	27.1	26.3	4.8	19.8	64.7
MLLC [56]	25.9	32.0	24.3	25.0	26.8	74.0
CTG-Net-P	**26.8**	34.1	**27.6**	**26.4**	**28.7**	**76.0**
CTG-Net-A	26.6	**35.1**	26.1	23.6	27.9	75.5

Method	Tempo-HL Splits - R@1					**Average**	
	DiDeMo	Before	After	Then	While	R@1	R@5
Prior	19.4	29.3	0.0	0.0	4.7	10.7	37.6
MCN [55]	26.1	26.8	14.9	18.6	10.7	19.4	70.9
TALL [37]	21.8	25.9	14.4	2.5	8.1	14.6	60.7
MLLC [56]	27.4	**32.3**	14.4	19.6	10.4	20.8	71.7
CTG-Net-P	**27.6**	27.9	16.9	18.7	10.5	20.3	71.3
CTG-Net-A	**27.6**	28.6	**18.8**	**20.8**	**11.6**	**21.5**	**72.7**

Table 4.1: Results on (Top) Tempo-TL and (Bottom) Tempo-HL. We compare against prior work, using two variants of our model: with the fixed parser (*CTG-Net-P*), and with the Bi-LSTM attention mechanism (*CTG-Net-A*). For both Tempo-TL and Tempo-HL, we train on the DiDeMo+Tempo-TL and DiDeMo+Tempo-HL training sets, respectively, and evaluate on the test sets. **Average** refers to the average of each metric across the splits (Section 4.4).

training examples from DiDeMo, as suggested by the original creators. This is referred to as "DiDeMo+Tempo-TL" or "DiDeMo+Tempo-HL."

Evaluation. We adopt the suggested evaluation metrics for both DiDeMo and Tempo. In both datasets, four annotators are each given a video and query and are asked to select the temporal segment which corresponds best with the query. In some cases, there is disagreement between the annotators. To account for disagreement, each metric is computed by comparing the prediction to each of the four annotations, and the annotation that most disagrees with the prediction is discarded. Using this method, we compute three metrics: Recall@1 (R@1), Recall@5 (R@5), and Mean Intersection over Union (mIOU). We note that prior works use the names Rank@1 and Rank@5 when computing these metrics. Our Recall@1 and Recall@5 metrics are equivalent to these.

For DiDeMo, the three metrics are computed for all videos in the dataset. For

TEMPO, we use a modification suggested by the original creators [56]: we split the dataset into subsets based on the temporal words ("before", "after", "then", "while") which are present in the queries. We compute the metrics within each subset, and then average the results with equal weight to get a final set of metrics. This allows us to get a fine-grained understanding of how our model performs under different temporal relationships.

4.4.1 Comparison with State of the Art

Tempo-TL. In Table 4.1, we compare CTG-Net with several prior works and baselines on Tempo-TL. All methods outperform the Prior baseline, which always selects the first video segment. Our model outperforms MCN [55] and TALL [37], two sliding-window approaches which do not account for the compositional structures present in Tempo. We also outperform MLLC [56], the previous state-of-the-art on Tempo-TL, which achieves an average R@1 of 26.8% compared to 27.9% for our method with the Bi-LSTM attention mechanism (CTG-Net-A) and 28.7% for ours with the parser (CTG-Net-P). In terms of average R@5, we also find that CTG-Net-P outperforms MLLC (+2.0%, from 74.0% for MLLC to 76.0% for ours), and performs comparably in terms of mIOU (-0.3%, from 42.3% for MLLC to 42.0% for ours). We perform particularly well on queries containing "before" (+3.1%, from 32.0% of MLLC to 35.1% of ours) and and "after" (+3.3%, from 24.3% of MLLC to 27.6% of ours), demonstrating our robustness to different temporal relationships.

Tempo-HL. In Table 4.1, we also perform the same set of comparisons on Tempo-HL. We find that Tempo-HL is more challenging for all temporal words, likely because of the wider range of temporal prepositions and coreferences present in natural language. Our model with the Bi-LSTM (CTG-Net-A) outperforms all prior work on this challenging dataset. We note that our model performs well on the less common

temporal relations “after” and “while”. These relations are particularly challenging because they require the model to perform temporal reasoning about sub-events that occur out of order (“after”) and simultaneously (“while”). Our temporal refinement procedure allows the model to take this into account.

We find that, while our model with the fixed parser (CTG-Net-P) outperforms that with the Bi-LSTM attention mechanism (CTG-Net-A) on Tempo-TL, the opposite is true on Tempo-HL. This is likely due to the rigid, procedurally constructed queries in Tempo-TL, which lend themselves well to parsing. Tempo-HL queries, on the other hand, contain more variation in sentence structure, which may not be accounted for by the parser. However, we find that both methods are competitive, demonstrating that compositional structure is useful, regardless of the method of decomposition. Concretely, CTG-Net-A achieves 21.5% Average R@1 on Tempo-HL, an improvement over MLLC (20.8%), the previous state-of-the-art. As with Tempo-TL, we also find that CTG-Net outperforms MLLC in terms of R@5 (+1.0%, from 71.7% for MLLC to 72.7% for ours), but performs worse in terms of mIOU (-1.4%, from 44.6% for MLLC to 43.2% for ours). This drop in mIOU is because our model, when compared to MLLC, tends to be over-confident when predicting shorter segments. These shorter segments may contain important components of the event, but this is not reflected in the mIOU metric.

DiDeMo. In Table 4.1, we demonstrate that CTG-Net outperforms MLLC on DiDeMo when trained on both DiDeMo and Tempo-TL (R@1 +1.0%, from 25.8% to 26.8%) and on DiDemo and Tempo-HL (+0.2%, from 27.4 to 27.6%). This demonstrates that CTG-Net performs well on the relatively simple queries present in DiDeMo, in addition to the complex queries present in Tempo.

Prior work on DiDeMo does not use the additional training data from Tempo-TL

Method	Tempo-HL - **Average** (Val) R@1	R@5	mIOU
Ours w/o $\mathbf{m_k}$, ϕ	20.75	71.91	41.68
Ours w/o $\mathbf{m_k}$	20.69	71.54	41.72
Ours w/o ϕ	21.24	72.09	42.55
Ours w/o $\mathbf{p_k}$, w_k	21.49	72.46	42.31
Ours w/o $\mathbf{p_k}$	21.37	71.68	42.58
Ours w/o w_k	21.27	71.89	42.88
Ours-A (Full)	**21.83**	**72.98**	**43.25**

Table 4.2: Ablation studies. Top section: to demonstrate the impact of compositional and temporal structure, we remove the sub-event masks $\mathbf{m_k}$ and temporal refinement network ϕ. Middle section: we remove the position embedding $\mathbf{p_k}$ and weights w_k from the sub-event representations. Our full model outperforms all variants. All numbers are reported on the Tempo-HL validation set.

and Tempo-HL, so for fair comparison we additionally report performance of CTG-Net trained on DiDeMo only. When trained on DiDeMo only, CTG-Net-A (27.8%) outperforms several recent baselines, including Temporal Modular Networks (22.9%) [95] and Moment Alignment Networks (27.0%) [200], both of which are competitive recent baselines which leverage compositional reasoning as part of their approach. We additionally achieve competitive results with MCN (28.1%), MLLC (28.4%), and Temporal GroundNet (28.2%) [55, 55, 15]. We observe that, suprisingly, CTG-Net-A performs slightly below MCN and MLLC when trained on DiDeMo only, while the opposite is true when we use additional data from Tempo. A simple explanation for this discrepancy is that our model is more complex, and therefore requires more data than is available in the DiDeMo-only setting. In support of this, we observe that our model performs similarly on DiDeMo when trained using the additional data from Tempo-HL (-0.1%), while other models experience steep *drops* in performance when using this additional data, such as -2.0% for MCN, and -1.0% for MLLC. This demonstrates that our model has higher capacity and is more robust to changes in distributions between datasets, both of which are important qualities in practice.

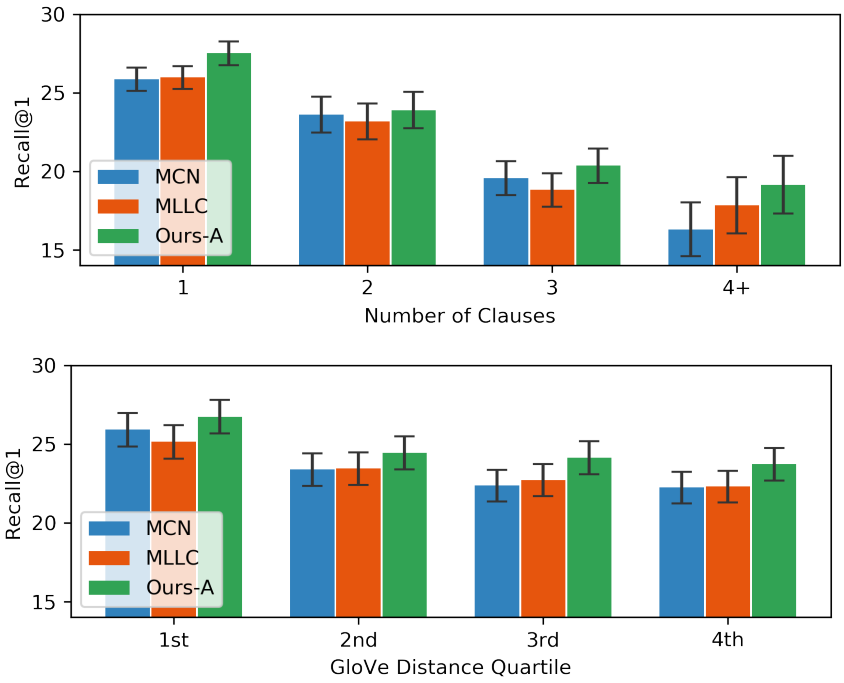


Figure 4.5: Recall@1 on DiDeMo+Tempo-HL as a function of query complexity (above) and query novelty (below). Complexity: queries with a high number of clauses are more complex and are therefore more difficult to ground, and our method outperforms prior work at all levels. Novelty: Queries which are dissimilar to previously-seen queries are also more difficult to ground, and our model outperforms prior work at all levels. Error bars depict one standard deviation.

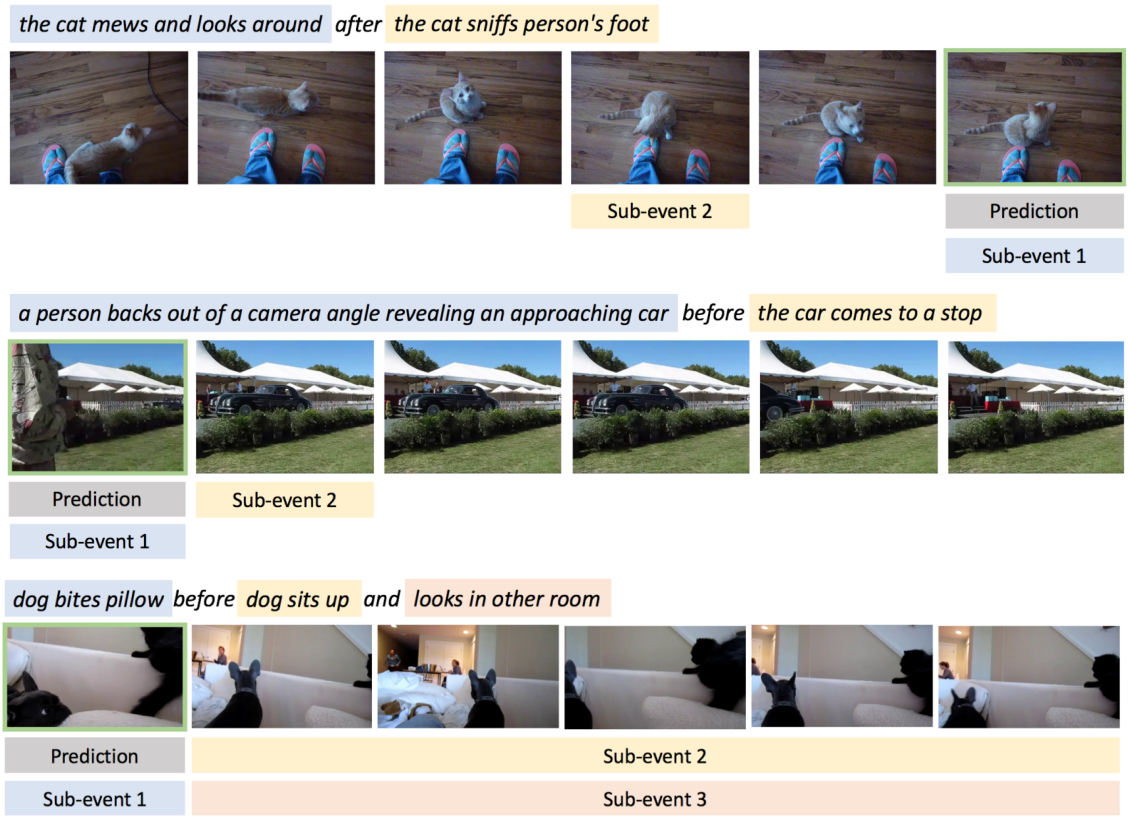


Figure 4.6: Examples of sub-event localization on Tempo-TL. The highlighted portions of each query indicate the sub-event masks, and the colored boxes below the video frames indicate the predicted locations of each sub-event (the location with the lowest matching score d_{kt}). In both cases, CTG-Net correctly identifies both sub-events, both in the query and in the video.

4.4.2 Complex & Novel Queries

We expect CTG-Net to generalize well to a broad range of queries. Two challenging areas are *complex* queries, which have many sub-events, and *novel* queries, which are dissimilar to queries seen during training. To test our model in these scenarios, we evaluate its performance on subsets of queries from DiDeMo and Tempo-HL with increasing complexity and novelty.

Complexity. To measure complexity, we count the number of clauses in each query, using the results of the Stanford Parser [75]. We divide the queries into categories based on the number of clauses, and we give the performance on each split in Fig-

ure 4.5. We find that queries with more clauses are more challenging to ground. We also find that our full model (Ours-A) achieves better performance than MCN and MLLC [55, 56] at all levels, demonstrating that we are able to generalize to both simple and complex queries.

Novelty. To measure novelty, we compute the average GloVe embedding for each query in the test set, find the most similar average embedding in the training set, and take the Euclidean distance between their embeddings. We consider queries with a high GloVe distance to be more novel. We divide the queries into four categories based on the quartile of the GloVe distance and give the performance in Figure 4.5. Novel queries are more challenging, and again we find that our full model achieves better performance at all levels, demonstrating that it is robust to novel queries.

In these experiments, we use Recall@1 over the entire Tempo-HL+DiDeMo test set, *not* the Average R@1 metric (Section 4.4) that separates by temporal words. This allows us to fairly compare queries based only their complexity and novelty, rather than their source dataset. We use our own implementations of MCN and MLLC, and verify that these implementations achieve similar or better performance than that reported by the original authors: specifically we find that MCN and MLLC achieve Average R@1 scores of 20.6% and 20.6% on Tempo-HL, respectively, compared to 19.4% and 20.8% in the original work.

4.4.3 Ablation Study

We perform an in-depth ablation study in Table 4.2. In these experiments, we demonstrate the contribution of the novel components of our model, namely our use of compositional and temporal structure, and our use of weights and position embeddings in the sub-event representation. In all ablation experiments, we use CTG-Net with attention (Ours-A) trained on DiDeMo+TempoHL, and we report

results on the validation set.

Compositional and Temporal Structure. In the top section of Table 4.2, we demonstrate the effect of eliminating compositional and temporal reasoning from our model. *Ours w/o* $\mathbf{m_k}$ refers to our model without sub-events masks. This model still has the temporal refinement step, but it does not have the benefit of receiving refinement updates from multiple sub-events. *Ours w/o* ϕ refers to our model without the temporal refinement step, which still uses compositional structure by combining sub-event matching scores. *Ours w/o* $\mathbf{m_k}$, ϕ has both components removed, and is equivalent to MCN [55]. We find that removing both compositional and temporal structure lead to decreased performance (-1.1%), and that removing compositional structure alone (-1.1%) is more detrimental than removing temporal structure (-0.6%). Interestingly, we see no benefit to including temporal refinement without also including sub-event decomposition, demonstrating that temporal refinement leverages updates from multiple sub-events as intended.

Sub-Event Representation. In the middle section of Table 4.2, we demonstrate the effect of removing two pieces of our proposed sub-event representation. *Ours w/o* $\mathbf{p_k}$ refers to our model without the position embeddings, which are later used as part of the refinement step. *Ours w/o* w_k refers to our model without sub-event weights, meaning that each sub-event is weighted equally under this scheme. *Ours w/o* $\mathbf{p_k}$, w_k refers to a model with neither of these components. We find that removing either of these components leads to decreased performance (-0.46% and -0.56%), but that removing them both does not lead to further decreased performance, demonstrating that there is still utility in including a sub-event decomposition without these components.

Qualitative Examples. In Figure 4.6, we present examples of CTG-Net correctly

identifying the locations of individual sub-events in complex queries. For more examples, please refer to the appendix.

4.5 Conclusions

We demonstrate that *compositional* and *temporal* structure are useful for temporal grounding. Specifically, show that event descriptions are *composed* of sub-events, and that they impose an *ordering* on these sub-events. To leverage these structures, we propose Compositional Temporal Grounding Networks (CTG-Net), and show that this model leads to higher performance on challenging datasets when compared with models which do not leverage such structure. We make our code publicly available.

4.6 Acknowledgements and Citation.

The work presented in this chapter was partially supported by King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No. OSR-CRG2017-3405, and by the Toyota Research Institute (TRI). This work was co-authored with Ryan McCaffrey, Rada Mihalcea, Jia Deng, and Olga Russakovsky. This chapter can be cited individually as [152].

CHAPTER V

Learning Video Representations from Textual Web Supervision

Video representations are typically learned in a fully-supervised fashion. For this approach to be successful, we require large amounts of labeled data, typically on the order of hundreds of thousands of labels. Acquiring these labels can cost tens of thousands of hours of human time to annotate [47, 13], and furthermore, when datasets become large, the benefit of gathering more labels appears to diminish [62]. At a certain point, it becomes too costly to simply label more data to improve performance. In this regime, we look to alternative sources of supervision to learn video representations without costly manual labels.

In this chapter, we draw this supervision from textual metadata available publicly on the Internet. Specifically, we use web videos from popular sites, where videos are associated with freeform text in the form of titles, descriptions, tags, and channel/creator names. These four pieces of textual metadata provide rich information about each video's content. Frequently, they describe the exact types of information which labelers are asked to annotate in labeled datasets, such as objects, scenes, and human actions. For example, consider the title, "Learning how to swim!" or the channel name "PotteryMaker". Both of these indicate what actions will take place in their respective videos, and we can leverage this information to learn representations,

in much of the same way we use labels in supervised learning.

The primary idea behind our approach is to use these pieces of text directly. This stands in contrast to recent work [42] in which the metadata is used indirectly, by using it to infer a class label for each example. Class labels seem like a natural choice for webly-supervised learning, as these are the most common form of supervision in strongly-supervised learning. However, class labels come from a closed vocabulary, while text is open-ended and therefore is necessarily more descriptive. Consider the title "Outdoor free-climbing in Yosemite". If we reduce this title to the class label "rock climbing", we are ignoring important information about the scene and the specific type of action, potentially missing out on valuable supervisory signal. In our experiments, we demonstrate that using text, and using multiple sources of text, translates into improved downstream performance. We compare our method with other webly-supervised approaches, showing that our method produces video representations which improve downstream performance by 2.2% on HMDB-51 [79] (Section 5.4).

Another advantage of this approach is that the amount of available data is immense; e.g. over 500 hours of content is uploaded every minute to YouTube alone [50], and each video is labeled with text. Due to this, we are able to perform experiments on a large scale, and we collect and learn from a dataset of 70 million videos. This dataset, Web Videos and Text (WVT-70M), is over 100 times larger than Kinetics [72], a commonly used large-scale video dataset, and is, to the best of our knowledge, the largest existing video dataset for webly-supervised learning. To acquire this data, we use a text-based video search engine to query for common words and collect a large, uncurated video dataset (Section 5.2).

Our goal with this data is to learn video representations—feature vectors which

encode a video clip—which are then useful for downstream tasks. To learn these representations, we propose a training scheme in which the video representation are used to pair each video with its associated metadata. We use powerful 3D Convolutional Neural Network (3D CNN) architectures to produce these representations, and train the video representations end-to-end on WVT-70M (Section 5.3). We evaluate the representations' effectiveness by fine-tuning them on a suite of downstream tasks. We find that pre-training with this approach significantly improves downstream performance, and that webly-supervised pre-training is complementary to strongly-supervised pre-training (Section 5.4).

Our key finding is that textual metadata is a rich source of supervision which can be acquired freely from public sources. Specifically, in this work, we make the following contributions:

- We collect a large-scale, uncurated dataset (WVT-70M), consisting of 70 million web video clips and their associated metadata, including titles, descriptions, tags, and channel names.
- We propose a method for learning video representations by learning to match these representations with their associated metadata.
- We demonstrate that our approach outperforms other webly-supervised and self-supervised approaches, achieving an improvement of 2.2% on HMDB-51.

5.1 Related Work

Webly-Supervised Learning. Many prior works have leveraged webly-labeled data for visual representation learning, both for images as well as videos. In general, these approaches use metadata found on the Internet to infer weak labels for a set of images or videos, and they differ in how these weak labels are created. The

most commonly-used approach is to use image search results, and label each image with the query that was used to find it [186, 30, 135, 7, 18, 24, 45, 17, 36, 78]. Another approach is to use captions, and label each image with key words present in the caption [115, 69, 91, 107]. Other approaches use user-defined keywords or tags [44, 64, 100, 42] or algorithmically-generated topics [2, 71] to the same end. These approaches have consistently demonstrated that webly-supervised learning is scalable and that it improves performance on downstream tasks, suggesting that webly-acquired class labels provide a valuable source of supervision.

A key observation in our work is that one does not need to infer class labels in order to learn from webly-acquired metadata. In our approach, we instead use the textual metadata directly, allowing for richer information to be used as supervision. This approach is similar to that of concurrent work [92] which uses titles as a form of textual supervision. Our work differs in that we also use other forms of metadata, such as descriptions. In addition, this prior work uses curated data from Kinetics-400, while we introduce an uncurated dataset as our source of videos. These videos provide a more realistic reflection of the webly-supervised videos available in the wild.

Unsupervised and Self-Supervised Learning. Our work is also related to methods of unsupervised and self-supervised learning, which do not use metadata from the Internet, and instead only use the video and its associated audio. Video (without audio) is already a valuable source of self-supervision, and varied approaches have successfully leveraged supervision from clip and frame ordering [106, 31, 89, 187, 191, 73], geometry [35, 68], motion [119, 81], colorization [174], cycle consistency [26, 185], and video prediction [102, 97, 173, 172, 178]. Generally, these approaches are outperformed by those leveraging supervision mined from external metadata, or from

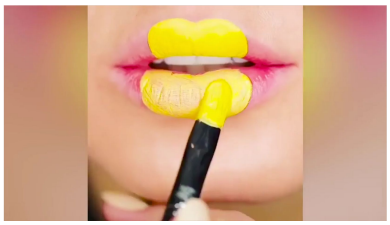
the audio channel.

Audio is a convenient and strong source of supervision: convenient because videos are almost always paired with an audio channel, and strong because the audio is tightly correlated with what is happening in the video. Prior works have leveraged ambient sound [117, 6, 5, 202, 77, 116, 132], dialogue [109], and narration [196, 3, 203, 205, 105, 104, 4], all of which of which serve as useful signals. Those approaches using narration typically do so with instructional videos, such as in the recent HowTo100M dataset [105], since instructional videos typically contain narration which describes the actions being performed. These approaches, like ours, reap the benefits offered by rich, descriptive supervision. However, they rely on a specific genre of video content (instructional videos), which poses a potential limitation. Our work, by contrast, can work with any genre of videos.

5.2 Dataset

To benchmark our approach, we collect a dataset of 70 million videos by searching for common action categories using a text-based web-video search engine. We begin by manually selecting the set of action categories; in our experiments we use the 700 action categories in Kinetics-700 [11]. We choose these categories because they cover a broad range of human actions, and also because this allows for fair comparison with fully-supervised approaches which pre-train on Kinetics (since the specific class categories used are known to have an effect on downstream performance [42]). We use the class names as search terms and collect the resulting videos from the web. We then apply two selection criteria to filter videos. First, we discard videos which are less than 10 seconds long, since we use 10-second clips during training (a choice also made to match Kinetics). Second, we discard videos which were uploaded in the

Title: *Quick Meals On the Grill Episode #1 (Chicken Tenders)*
Desc: *Follow BACKYARD BBQ-R on Social Media ...*
Tags: *Barbeque, smoking, weber, bbq, bbqing ... +17*
Channel: *Backyard 'Bbq-R'*



Title: *Lipstick Tutorial Compilation | New Amazing Lip ...*
Desc: *Lipstick Tutorial Compilation | New Amazing Lip ...*
Tags: *Makeup, DIY, Makeup Tutorial, Tutorial, How To, ... +4*
Channel: *Life & Beauty*

Title: *The First Mowing of The Spring Lawn (timelapse)*
Desc: *It's a nice spring day and the grass needs mowing ...*
Tags: *mow, lawn, grass, time, lapse, mower, push, ... +13*
Channel: *tallt66*

Title: *Lower Carb Chicken Bacon Ranch Mac and Cheese*
Desc: *This high protein, low carb chicken bacon ranch ...*
Tags: *None*
Channel: *Mason Woodruff*

Figure 5.1: Examples of video frames and metadata from WVT-70M. Metadata typically contains references to actions (*mowing*, *bbqing*) as well as objects (*grill*, *lipstick*, *bacon*) which are present in the scene. We collect four types of metadata for each video: titles, descriptions, tags, and channel names. Metadata is truncated where necessary for ease of visualization. All videos used under CC BY 2.0 license.

past 90 days, because older videos are less likely to be deleted in the future, allowing for improved reproducibility of our experiments. In total, we collect 100K videos from each of the 700 queries, resulting in a dataset of 70M videos. From each video, we randomly select a 10-second clip to download and later use for training.

Each video is paired with four pieces of textual metadata: its title, description, tags, and channel name. These were chosen for two reasons. First, these pieces of text are all manually written by the user, as opposed to being automatically generated. This is desirable because the inner workings of automatically generated metadata (such as YouTube's "topics" [2]) are unknown, and could potentially be generated via content-based models trained on our target datasets, allowing these labels to leak into the training set. By relying only on manually-annotated metadata, we avoid this potential issue. Second, from manual inspection, we see that these pieces of text

consistently contain informative references to content in the video. These references are written deliberately by the user, who generally will choose a title, description, and tags which help other users find their video. The user will also select a channel name (an identifier used to represent the user) which is informative, typically one which is indicative of the types of videos that the channel contains. The channel name provides context which the other signals may not, for example, a channel for guitar lessons, "Jeff's Guitar Lessons", may not explicitly say "guitar lesson" in each video title, but the channel name makes this obvious. For some examples of videos and their metadata, see Figure 5.1.

Like many approaches towards webly-supervised learning, we rely on a search engine to collect data. This again raises the question of "leakage" from the test set into the training set: if content-based models (possibly trained on our target datasets) are used to generate the search results, does this introduce the possibility of labels (in the form of search terms) leaking into our training set? In our case, no, since we do not use the search terms as labels, labels cannot leak into the training set through the search engine. This still allows for the possibility that the videos are indirectly "curated", that is, the resulting videos may be more neatly divided into class categories than what could be achieved without content-based search. However, it is still standard practice to use search results for "uncurated" data collection [105], because search provides a practical method for acquiring large amounts of data from the Internet.

In Table 5.2, we compare WVT-70M to other webly-supervised datasets for video representation learning. In terms of the number of videos, WVT-70M is on par with the largest datasets in prior work, with 5M more unique source videos than [42]. We acknowledge that, conceptually, any of these prior datasets could be scaled to much

Dataset	#Videos	Duration (hrs)	Supervision
Sports-1M [71]	1.1M	15K	Topics
Youtube-8M [2]	8M	500K	Topics
HowTo100M [105]	1.2M	136K	Speech
IG-Kinetics [42]	65M	72K	Hashtags
WVT-70M (ours)	70M	194K	Text Metadata

Table 5.1: Datasets for webly-supervised video representation learning. WVT-70M contains 70 million clips, each from a unique source video, and each video is paired with textual metadata.

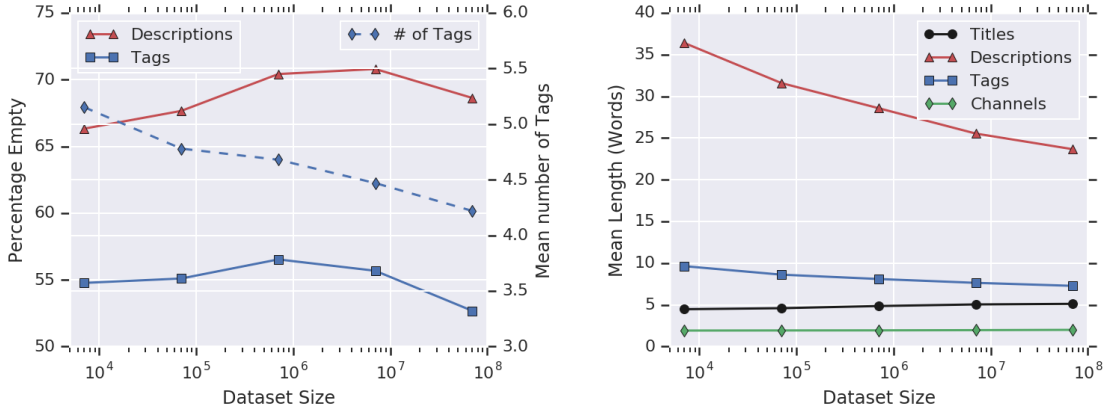


Figure 5.2: Scaling properties of WVT-70M. **Left:** Rate of missing descriptions and tags, and number of tags. Both descriptions and tags are empty for a large number of videos, at all dataset sizes. **Right:** Mean length (in words) of each metadata type. Descriptions and tags tend to get shorter with larger dataset sizes, but titles and channel names tend to get longer.

larger sizes simply by collecting more data, making dataset size a dubious method of comparison. However, it is still important to study how these methods behave when scaled to extreme dataset sizes, and therefore our experiments on 70M videos are a valuable contribution in this space. These experiments are particularly important because there are non-trivial issues associated with scaling webly-supervised learning to extreme dataset sizes. The key issue is that we use search results to collect data, and the quality of these results declines as we move deeper into the search rankings to collect more videos.

To analyze the scaling properties of WVT-70M, we collect increasingly-large sub-

sets of the dataset and measure indicators of their quality, shown in Figure 5.2. The dataset size is scaled up as one would do in practice, by selecting more and more of the top search results from each query, rather than by performing a random sample from the full WVT-70M dataset. The indicators measure, for each piece of metadata, the mean length (in words), the rate of missing-ness (for descriptions and tags, which can be omitted by the user), and the mean number of tags. We find that search results are imbalanced in terms of how these indicators are distributed. Specifically, descriptions and tags tend to get *shorter* with larger dataset sizes, but titles and channel names in fact get *longer*. We also find that the percentage of videos which have any tags or a description stays relatively constant, but the average number of unique tags drops. These analyses indicate that the quality of descriptions and tags tend to decrease, that is, they get shorter and therefore less descriptive, for larger dataset sizes. Notably, we do not see the same for titles or channel names, indicating that these may be a more reliable source of supervision at the largest dataset sizes. This is reflected in our experiments in Section 5.4.2, where we find that using all sources of metadata is helpful for smaller dataset sizes, but that these additional sources of metadata reduce performance when scaled to the largest dataset sizes.

Implementation Details. Since Kinetics videos are also collected from the Internet, we discard videos from WVT-70M which appear in the Kinetics validation or test sets. Since many videos do not contain a description or tags, we code the missing information as an empty string, rather than discarding these videos. We perform all searches in English, so WVT-70M contains primarily (though not exclusively) English-language videos and metadata. However, our approach is extensible to any language.

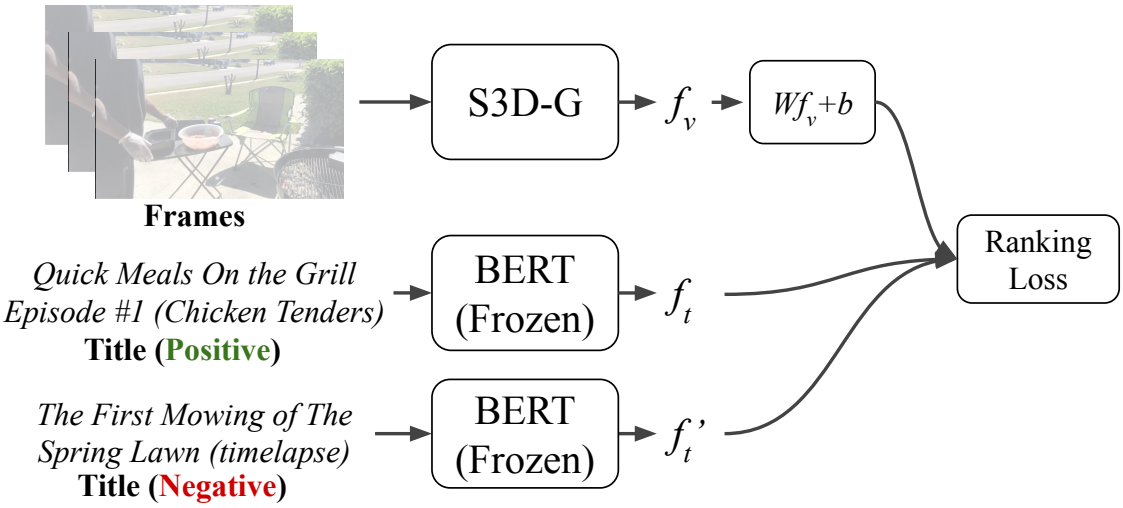


Figure 5.3: Model architecture for webly-supervised learning from textual metadata. We encode the video using S3D-G [189], and the metadata using BERT [23]. We then train the video representation by matching it with the correct metadata representation.

5.3 Model

At a high level, our approach (Figure 5.3) learns video representations by creating representations of the video's metadata, and encouraging the video representations to match these metadata representations. The video representation is a vector $f_v \in \mathbb{R}^{D_v}$, and the metadata representation is a vector $f_t \in \mathbb{R}^{D_t}$, where the vector dimensions D_v and D_t are dependent on the models used to extract each representation and do not need to be the same.

Intuitively, the video and its metadata contain similar information, and therefore their representations f_v and f_t should contain similar information. However, the information contained in the video and its metadata are not exactly the same. The video will always contain information which is not present in the metadata. For example, the description of a rock climbing video will not list every hold the climber uses on their route. Likewise, the text will provide context which is not present in the video, such as listing the time and location where the video was shot. With our

approach, we leverage this observation by encouraging the video representations to be similar, but not the same as, the corresponding metadata representation.

Specifically, the video representations are trained by *predicting* the metadata representations. We predict the metadata representations from the video representations by applying a simple linear transformation, that is $\hat{f}_t = W f_v + b$, where $W \in \mathbb{R}^{D_t \times D_v}$ and $b \in \mathbb{R}^{D_t}$. We then apply a ranking loss which penalizes f_v if $\hat{f}_t$ is similar to the metadata representation for another video f'_t. That is, $\max(0, m + d(\hat{f}_t, f_t) - d(\hat{f}_t, f'_t))$, where d is a distance metric, and m is the minimum allowable margin between $d(\hat{f}_t, f_t)$ and $d(\hat{f}_t, f'_t)$. In our experiments, we set d as the cosine distance, $d(u, v) = 1 - \frac{u^T v}{\|u\|_2 \|v\|_2}$, and choose the margin to be $m = 0.1$ with the validation set.

Negative Examples. For the loss, we require a "negative" metadata representation f'_t, that is, one drawn from a different video than f_v. We draw the negative example f'_t from another video in the dataset uniformly at random. In addition, we use multiple negative examples $\{f'_{ti} \mid i = 1 \dots K\}$ for each positive example, and take the mean of their respective losses to get the loss,

$$\mathcal{L}(f_v, f_t, \{f'_{ti}\}) = \frac{1}{K} \sum_{i=1}^{K} \max(0, m + d(\hat{f}_t, f_t) - d(\hat{f}_t, f'_{ti})). \tag{5.1}$$

In practice, we use $K = 15$, giving a ratio of 1 positive example for every 15 negative examples. We do not perform any hard-negative mining; we find that uniformly sampled negatives are sufficient. These negative examples are taken from the same batch of SGD training for convenience of implementation.

Multiple Sources of Metadata. When using more than one source of metadata for pre-training, we compute separate metadata representations f_t for each source. Then, for each source, we apply a different set of linear transformation parameters W, b to the video representation f_v, to compute a source-specific $\hat{f}_t$. We then separately

compute a loss for each source as in Equation 5.1. The final loss is the sum of these losses.

End-to-End Training. We train the video representation f_v end-to-end with the linear transformation parameters W and b. Since our goal is to learn *video* representations, not text representations, we do not train the metadata representations f_t end-to-end. Instead, we use a pre-trained state-of-the-art text feature extractor to generate these embeddings (Section 5.3.2).

We train the model using stochastic gradient descent, with Nesterov momentum of 0.9 [162] and a weight decay of 1e-5. We apply dropout with a rate of 0.5 to the video features. We use a batch size of 2048 split into chunks of 16 videos across each of 128 accelerators, trained synchronously. The learning rate schedule begins with 1500 warmup steps (exponentially increasing from .001 to 1.0), followed by a cosine-decaying [96] schedule for the remaining steps. We train on 70M videos for only 140K steps in total, which translates into just over 4 full epochs. Due to the accelerators and large batch size, this model takes less than 4 days to train.

5.3.1 Video Representation

We create the video representation $f_v \in \mathbb{R}^{D_v}$ using a 3D Convolutional Neural Network (3D CNN) which operates directly on the RGB video frames. The input to the 3D CNN is therefore a $H \times W \times T \times 3$ tensor which represents the video clip. To get the video representation, we take the final hidden layer of the network and (when necessary) mean-pool across the spatial and temporal dimensions, resulting in a vector of length D_v.

In our work, we use S3D-G [189] as the backbone 3D CNN architecture. We choose this architecture because it outperforms the commonly-used I3D architecture [12] at lower computational cost. We do not train on larger-capacity models

such as R(2+1)D-152 (118M params, 10x that of S3D-G) due to the significant computational cost of training such a model on 70M videos. In addition, our goal in this work is to demonstrate the utility of textual metadata, rather than of any particular backbone 3D CNN. Prior work has shown that pre-training a higher-capacity model on a large dataset leads to a similar change in accuracy as pre-training a lower-capacity model [42], suggesting that our results could be also be applied to larger-capacity models. However, this comparison is beyond the scope of this work.

During training (both pre-training and fine-tuning), we apply the 3D CNN on 64-frame clips drawn uniformly at random from the video at 25fps. We resize the frames to 256px on the shortest edge, and then take a random 224×224 crop. We additionally perform random brightness, contrast, and flipping augmentation. During inference, we use 250-frame clips (using circular padding where necessary), and take a center 224×224 crop.

5.3.2 Metadata Representation

For each piece of textual metadata, we create a metadata representation $f_t \in \mathbb{R}^{D_t}$ using BERT [23], a state of the art text encoder. BERT returns a 768-dimensional embedding for each token in the text, and we take the mean of these token-level embeddings to get a single 768-dimensional representation of the metadata, that is, $D_t = 768$.

Specifically, we use the multilingual, cased version of BERT which was pre-trained on 104 languages, and has 12 layers and 110M parameters. We use the multilingual version because non-English text appears frequently in WVT-70M. Since our goal is to learn *video* representations, we do not fine-tune the BERT model. This also significantly alleviates the computational cost of training; otherwise fine-tuning the text model would dominate the computational cost.

When computing features for tags (where each video can have zero to many tags), we compute a BERT embedding for each individual tag and take the mean of the results. For videos with no tags, we replace it with an empty string. Each of the three other pieces of metadata (titles, descriptions, and channel names) are treated the same.

5.4 Experiments

For many of our experiments, we use a subset of the full 70M-video dataset. These subsets are denoted by the approximate number of videos they include: `500K`, `1M`, `6M`, `12M`, `40M`, and `70M`. These subsets are not selected at random, instead each subset is chosen by selecting a smaller number of the top search results from each query, such that the `500K` subset contains approximately the top 700 results per query (recall that the 70M dataset contains 100K results per query). This reflects the way that such a method could be used in practice; one would search for queries relevant to their particular downstream task and collect as many of the top search results as they can, subject to space or bandwidth constraints.

We do not segment WVT-70M into a validation or test split, and instead evaluate our learned model purely by its performance on downstream tasks. We evaluate on four downstream video classification tasks:

HMDB-51. HMDB-51 [79] is an action recognition dataset consisting of short video clips associated with one of 51 classes. It contains 7000 videos, and is commonly used as a benchmark for video representation learning. We report results on the first test split, except where otherwise noted. When fine-tuning on HMDB-51, we use a learning rate of 1e-3 with a cosine decay schedule, a weight decay of 1e-4, and we train for 1000 iterations.

UCF-101. UCF-101 [149] is a similar action recognition dataset consisting of video clips associated with one of 101 classes. It is larger than HMDB-51, consisting of over 13,000 videos. We report results on the first test split, except where otherwise noted. When fine-tuning on UCF-101, we use a learning rate of 1e-3 with a cosine decay schedule, a weight decay of 1e-3, and we train for 1000 iterations.

Kinetics-400, 600, 700. Kinetics is a widely-used action recognition dataset consisting of 10-second clips drawn from videos annotated with action categories [72]. Kinetics-400, 600, and 700 are increasingly larger versions of the dataset, containing 400, 600, and 700 action categories, respectively [10, 11]. Kinetics contains over 545,000 videos, and due to its scale, it is commonly used to pre-train video representations. We compare against Kinetics as a pre-training scheme, in addition to using it as a downstream task.

Kinetics videos can be deleted by their uploaders at any time, and afterwards can no longer be recovered by researchers. Therefore, Kinetics gradually deteriorates over time, which generates discrepancies between both training and evaluation performed at different times. Our experiments were conducted using a snapshot of the Kinetics dataset collected in February 2020, when Kinetics-400 contained 225K of the original 247K training examples (-8.9%), Kinetics-600 contained 378K of the original 393K training examples (-3.8%), and Kinetics-700 contained 541K of the original 545K training examples (-0.7%).

5.4.1 Different Forms of Metadata

We collect four types of metadata for each video: the title, description, tags, and channel name (Section 5.2). We observe that each type of metadata contains a different level of detail and is affected by different sources of noise (Figure 5.1). Therefore, we expect the different types of metadata to have different impacts on

Supervision	HMDB-51
Scratch	27.9
Titles	43.2
Descriptions	37.7
Tags	36.2
Channel Name	29.1
Titles + Desc.	43.9
Titles + Desc. + Tags	46.5
All	**50.0**

Table 5.2: Sources of metadata used and their effect on downstream performance, as measured on HMDB-51. Each source of metadata contributes individually to the final accuracy. For these experiments, we pre-train on WVT-500K. All reported accuracies are on HMDB-51 split 1.

Title: *HOW TO CHANGE SCOOTER ENGINE OIL ...*
Desc: *HOW TO CHANGE SCOOTER ENGINE OIL ...*
Tags: *N/A*
Channel: *Repair PH*

Title: *Episode 2 - "Scrumptious Scallop Salsa"*
Desc: *INGREDIENTS: 6 scallops, 1 avacado diced, ...*
Tags: *food, recipe, cooking, scallops, salsa, Kalamazoo*
Channel: *ChewiesChow*

Title: *WOW 20 WAXING DEPILATION | VIRAL BEAUTY ...*
Desc: *If you have any questions with our video, please ...*
Tags: *waxinglessons, waxingdepilations, waxdepilations, ...*
Channel: *Viral Beauty*

Figure 5.4: Additional examples of metadata, demonstrating complementary information. One source of metadata is not usually sufficient to fully understand the video content. All metadata used under CC BY 2.0 license.

downstream performance. We investigate which of these are the most useful for pre-training in Table 5.4.1 and Figure 5.4.1. For these experiments, we pre-train the model on WVT-500K and fine-tune on HMDB-51.

We find that all types of metadata are useful sources of supervisory signal for pre-training. Titles are the most effective, achieving an increase in downstream accuracy of 15.3% over a from-scratch baseline. Channel names are the least effective, resulting in only a 1.2% improvement over the baseline. However, we find that these sources of supervision provide complementary signals, and that we achieve the best performance

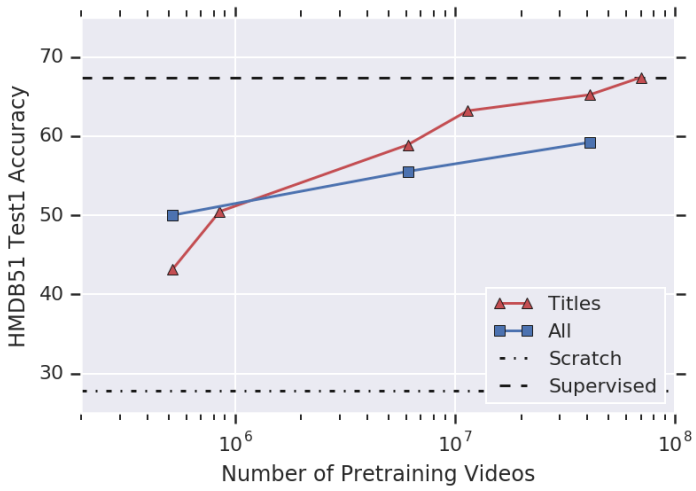
by including all of them during pre-training. This achieves a down-stream accuracy of 50.0% on HMDB-51, a 22.1% improvement over the from-scratch baseline.

In addition, these experiments can be used to show the relative utility of webly-supervised learning and fully-supervised learning. These experiments are conducted using WVT-500K, which is approximately the same size as Kinetics-700 (545K videos). For comparison, a fully-supervised model pre-trained on Kinetics-700 achieves 67.4% accuracy on HMDB-51, a 17.4% improvement over training on all four sources of metadata. As expected, web supervision suffers from noise and therefore is not as effective, video for video, as supervised pre-training. However, web supervision does not incur any labeling cost, making it an effective option for pre-training.

5.4.2 Scaling to 70M Videos

To demonstrate the scalability of our method, we apply it to increasingly large subsets of the full 70M-video dataset in Figure 5.5. We compare two metadata configurations for this experiment: (1) only titles, and (2) all metadata. We find that the titles-only approach scales significantly better than the all-metadata approach; although using all metadata leads to higher downstream accuracy with 500K pre-training videos, this is reversed when using more than 1M pre-training videos. This is likely due to the poor scaling properties of tags and descriptions as shown in Figure 5.2, and suggests that too much noise can become a burden on training.

For the titles-only approach, we find that using more pre-training data sharply improves performance. Using all 70M videos for pre-training achieves an HMDB-51 accuracy of 67.4%, a 13.2% improvement over using 500K videos. In addition, this accuracy is the same as that of an equivalent model trained on Kinetics-700, demonstrating that our approach can match the performance of fully-supervised pre-training, without any labeling cost.



Dataset	Iters	HMDB-51
Scratch	N/A	27.9
500K	20K	43.2
1M	25K	50.5
6M	30K	58.9
12M	50K	63.2
40M	100K	65.2
70M	140K	**67.4**
K700	30K	67.4

Figure 5.5: Performance of our approach on HMDB-51 (split 1) for increasingly larger pre-training dataset sizes, compared to a baseline model trained from scratch and a model pre-trained on Kinetics-700. **Left**: Comparison of titles-only and all-metadata approaches. Titles-only scales better than all-metadata. **Right**: Number of pre-training iterations and resulting accuracy. K700 = Kinetics-700. Our approach with 70M videos matches that of fully-supervised pre-training.

We do not expand the model capacity or adjust the pre-training hyperparameters when scaling to 70M videos. The only difference is the number of pre-training iterations, which we list in Figure 5.5. We found that increasing the number of iterations further lowered down-stream performance, for the smaller-scale datasets. Interestingly, we achieve good performance on the 70M dataset using only 4 epochs of training, while on the 500K dataset we require over 80 epochs of training. This suggests that increased model capacity and longer training could further improve performance for the 70M dataset.

5.4.3 Comparison with Prior Work

In Table 5.4.3, we compare our approach, pre-trained on WVT-70M, against other methods for self-supervised and webly-supervised learning. We strongly outperform all existing methods for self-supervised learning which use video as the only source of supervision, suggesting that the textual metadata provides a supervisory signal that cannot be obtained from video alone. We find that our approach outperforms

	Method	Data	Model	HMDB-51	UCF-101
	Baseline	None	S3D-G	27.9	58.5
Video-only	Geometry [35]	FC	FlowNet	23.3	55.1
	OPN [89]	UCF	VGG	23.8	59.6
	CMC [167]	UCF	CaffeNet	26.7	59.1
	ClipOrder [191]	UCF	R(2+1)D	30.9	72.4
	O3N [31]	UCF	AlexNet	32.5	60.3
	MASN [178]	K400	C3D	33.4	61.2
	DPC [51]	K400	3D-R34	35.7	75.7
	Shuffle&Learn [106]*	K600	S3D	35.8	68.7
	3DRotNet [68]*	K600	S3D	40.0	75.3
	CBT [155]	K600	S3D	44.6	79.5
Video+Audio	AVTS [77]	K600	I3D	53.0	83.7
	MIL-NCE [104]	HT100M	S3D	61.0	91.3
	XDC [4]	IG65M	R(2+1)D	63.1	**91.5**
Webly-Sup.	Sports-1M [71]	S-1M	AlexNet	-	65.4
	Gan et al. [36]	YouTube	VGG	-	69.3
	CPD [92]	K400	3D-R34	57.7	88.7
	Ours	WVT-70M	S3D-G	**65.3**	90.3

Table 5.3: Comparison with self-supervised and webly-supervised pre-training prior work on HMDB-51 and UCF-101. "Data" refers to the source of pre-training videos, however, these approaches do not use the available labels. All numbers are quoted directly from the original authors. Our results are averaged across all three splits of HMDB-51 and UCF-101. *Reimplemented by [155].

all prior methods for webly-spervised approaches on HMDB-51, and performs on-par with state-of-the-art methods on UCF-101 which use audio as a primary source of supervision. Notably, we outperform MIL-NCE [104], a recent method for learning video representations from instructional videos in the HowTo100M dataset [105], on HMDB-51 (+4.3%). We also outperform two prior approaches on UCF-101 (+24.9%, +21.0%) which learn video representations using web supervision from YouTube [71, 36].

In Table 5.4.3, we present results on Kinetics. We find that our pre-training improves performance by 1-3% over a from-scratch baseline, depending on the particular version of Kinetics. These improvements are much smaller than what we found on HMDB-51 and UCF-101, however, this is to be expected, as Kinetics is

Method	Model	K400	K600	K700
Baseline	S3D-G	68.9	74.3	62.2
Baseline [104]	I3D	-	-	57.0
Baseline [42]	R(2+1)D-18	69.3	-	-
MIL-NCE [104]	I3D	-	-	61.1
IG65M [42]	R(2+1)D-18	**76.0**	-	-
Ours	S3D-G	72.0	**76.0**	**63.4**

Pre-training	HMDB-51
70M	67.4
K700	67.4
70M+K400	72.2
70M+K600	74.5
70M+K700	**75.9**

Table 5.4: Experiments on Kinetics. KX = Kinetics-X. **Left:** Comparison with prior work on webly-supervised learning on Kinetics-400, -600, and -700. We use numbers quoted directly from the authors. **Right:** Complementary nature of webly-supervised and fully-supervised learning. We pre-train the model on WVT-70M, then fine-tune it on Kinetics, then apply it to HMDB-51 (split 1).

already a large-scale dataset, and therefore has less to gain from pre-training. We compare against two prior works on Kinetics, MIL-NCE [104] (which uses supervision from narration) and IG65M [42] (which uses hashtags from Instagram). We find that we outperform MIL-NCE on Kinetics-700, however, we underperform IG65M on Kinetics-400. This suggests that hashtags are a stronger source of supervision than textual metadata. This could be due to a number of factors, such as the relative amount of noise in the two types of signals.

5.4.4 Complementary Strong- and Web-Supervision

Webly-supervised learning has the capacity to meet the performance of strongly-supervised learning, without any labels (Section 5.4.2). However, in practice, one would use all sources of supervision available, including labeled datasets. Therefore, we ask whether webly-supervised and strongly-supervised learning can be applied in combination, to further improve the performance on down-stream tasks. We test this in Table 5.4.3 by training in a three-step process: first, we pre-train our model on WVT-70M. Then, we fine-tune this model on Kinetics. Finally, we apply the resulting model to HMDB-51.

We find that strongly-supervised learning and webly-supervised learning are in-

deed complementary. When using both WVT-70M and Kinetics-700 are in combination, the down-stream accuracy on HMDB-51 increases by a further 8.5%. This demonstrates that our method is effective even in situations where labeled data is already plentiful.

5.4.5 Extensions and the Kinetics 2020 Challenge

We achieve first place in the Kinetics-700 Challenge at CVPR 2020 [98] using WVT-70M with improved backbone architectures, extensively tuned hyper-parameters, and model ensembling. The final winning solution includes an ensemble of five backbone models, two of which are pre-trained on WVT-70M. The highest-performing individual RGB model is pre-trained on WVT and achieves 74.9% top-1 validation accuracy on Kinetics-700, compared to 71.2% when trained from scratch and 63.4% in Table 5.4.3. The final model model ensemble achieves a top-1 validation accuracy of 78.6% and a top-1 test accuracy of 77.2%, establishing a new state-of-the-art for the Kinetics-700 dataset.

For more details on this extension, please refer to the presentation [98].

5.5 Conclusions

We demonstrate that textual metadata serves as a useful signal for pre-training video representations, without the need for any manually annotated labels. Specifically, we find that each textual signal is complementary (Section 5.4.1), and that this approach matches the performance of supervised pre-training when scaled to tens of millions of videos (Section 5.4.2). We also show that it outperforms competitive approaches for both self-supervised and webly-supervised learning (Section 5.4.3). Finally, we demonstrate that it is complementary to existing supervised pre-training methods (Section 5.4.4). These findings suggest that textual metadata can be used

as an effective pre-training strategy for a wide variety of downstream tasks.

5.6 Acknowledgements and Citation

Work from this chapter was completed while interning at Google Research, and is co-authored by David A. Ross, Chen Sun, Jia Deng, Rahul Sukthankar, and Cordelia Schmid. Results on the Kinetics 2020 Challenge are co-authored by Zhichao Lu, Xuehan Xiong, Yinxiao Li, and David A. Ross. This chapter can be cited individually as [154].

CHAPTER VI

Conclusion

In this work, we introduced four novel techniques for performing action recognition, each addressing the task from different angles and at different scales. We introduced Distilled 3D Networks (Chapter II), an algorithm for learning motion features from RGB videos, addressing the issue of learning to recognize actions from instantaneous motion. We introduced Temporal Hourglass Networks (Chapter III), which address the opposite issue of learning video features at long timescales. These techniques only consider actions which come from a fixed vocabulary, so we introduce Compositional Temporal Grounding (Chapter IV), which allows actions to be recognized from arbitrary natural language descriptions. Finally, to address the immense need for training data for these techniques, we introduce Web Video Text (Chapter V), which uses paired text and video data freely available on the internet to learn video representations.

These advances represent broad strides in the direction of robust action recognition systems. Using these techniques, we demonstrate improved performance on a wide variety of benchmark datasets, including those with small amounts of labeled data (Section 5.4), and those with open-ended action descriptions (Section 4.4). In addition, we analyze properties of current state of the art models, identifying when

such models fail to learn adequate motion representations (Section 2.5.1). Finally, our work advances the state of the art on the most challenging benchmark dataset in the field (Section 5.4.5).

6.1 Future Directions

While our contributions make significant strides towards learning features from both short and long-term temporal context, there is much research yet to be done on the integration of these two signals. In Chapter II, we leverage 3D CNNs for motion representation learning, however in Chapters III and IV, we find that 3D CNNs are no longer scalable to problems involving long-term temporal context. This suggests a need for cohesive video architectures which can seamlessly deal with both short and long-term temporal context.

Our work, and much work in the field today, focuses on solving action recognition in constrained settings, such as in the case where the action classes are mutually exclusive and known in advance. However, these constraints are rarely met in real world scenarios. Therefore, it is not clear how well these architectures can be directly applied to video "in the wild". To overcome this challenge, we must move beyond fixed action classes and instead allow for new actions to be learned from simple representations, such as demonstrations, examples, or descriptions. Our work in Chapter IV establishes strong results in this area compared to prior work, but it is clear from the overall accuracy of the current approaches that more work is needed.

Finally, we have focused entirely on action *recognition*, while it is debatable whether this constitutes true action *understanding*. One might argue that understanding requires more than simply recognizing an action, for example, the ability to describe or perform an action. To achieve this, the system would need to have an

altogether new form of action representation, one which includes the intention which motivates the action. This area is mostly unexplored in the current literature, as the current datasets are not suited for evaluating such abilities, and therefore it is unclear whether current models are sufficient. Moving forward in this area will likely require significant exploration in the space of both models and datasets for video action recognition.

APPENDICES

APPENDIX A

Supplementary Materials for Distilled 3D Networks

A.1 Predicted Optical Flow Visualizations

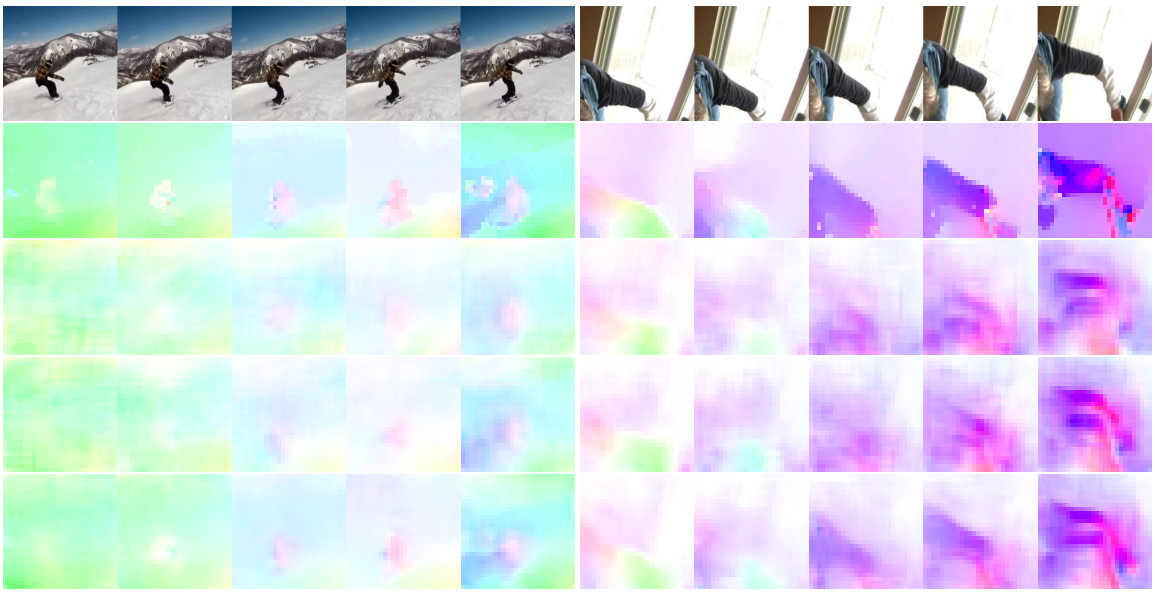


Figure A.1: Examples of optical flow produced by S3DG and D3D by adding the optical flow decoder applied at layer 3A. From top to bottom: RGB Frames, TV-L1 optical flow, S3D-G flow, D3D flow, D3D flow with finetuning. TV-L1 optical flow is shown downsampled to 28×28 px, which is the decoder output resolution used during training.

A.2 Performance on Kinetics-400 Categories

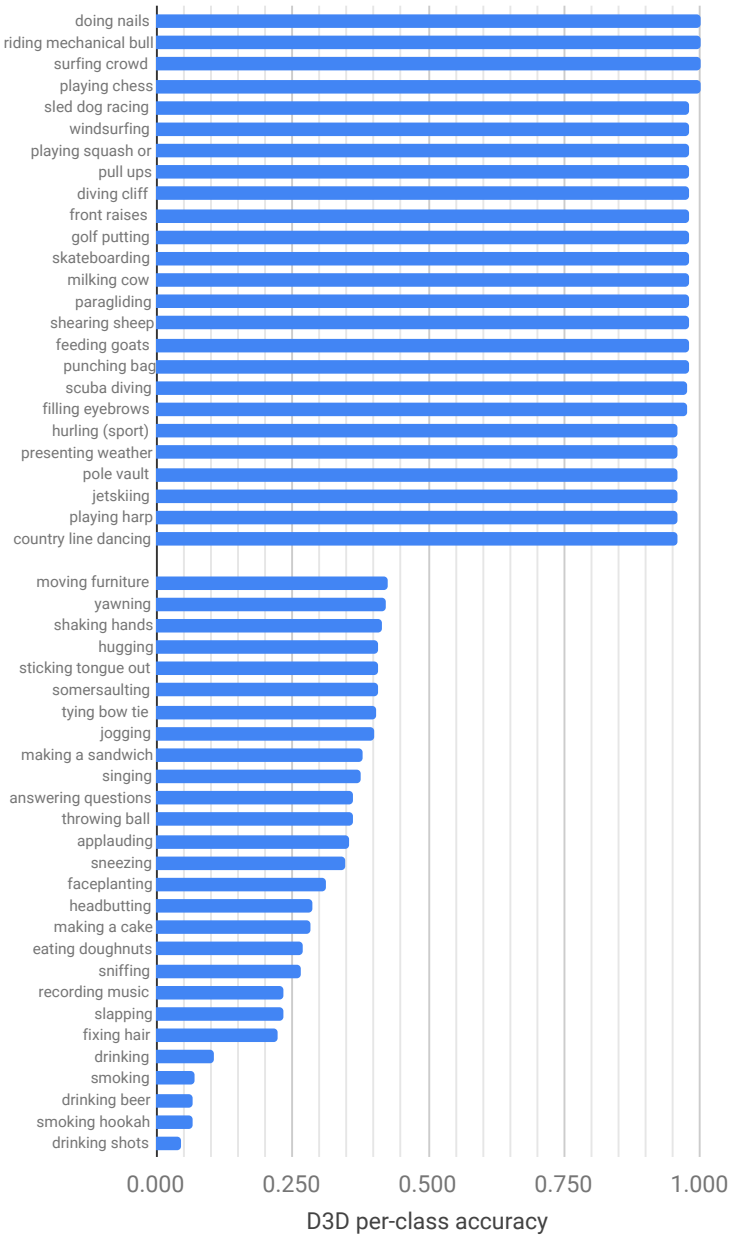


Figure A.2: Accuracy on individual Kinetics-400 categories using D3D. We show the per-class accuracy for D3D trained on Kinetics-400. Only the top and bottom 25 classes are shown.

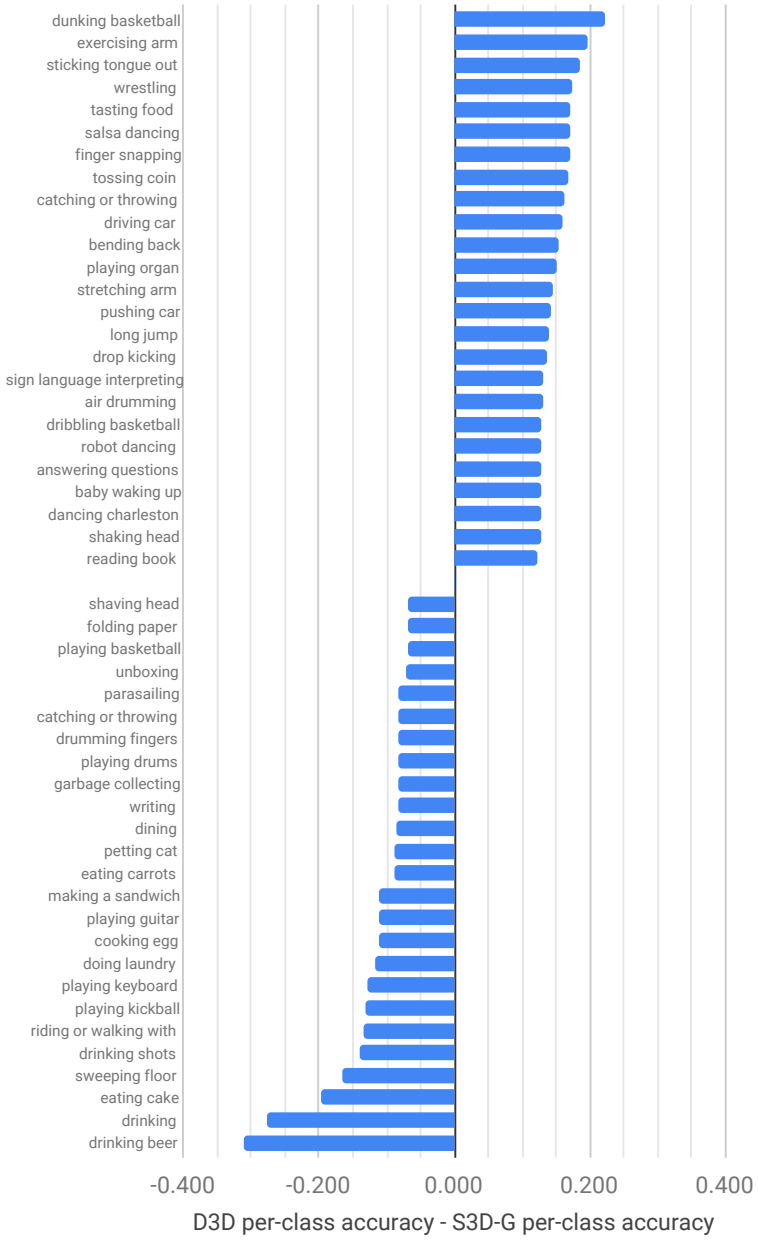


Figure A.3: Accuracy difference on individual Kinetics-400 categories by adding distillation. We compare the difference between per-class accuracy for D3D and per-class accuracy for S3D-G. Only the top and bottom 25 classes are shown. In total, D3D leads to improvements on 203 of the 400 classes (50.8%) and degradations on 103 of the 400 classes (27.3%), with less than a ±.1% difference on the remaining classes.

A.3 Optical Flow Decoders

A.4 Non-Local S3D-G

For our experiments with Non-Local S3D-G (NL S3D-G), we include two non-local blocks [184] into the S3D-G architecture, immediately before blocks 5B and

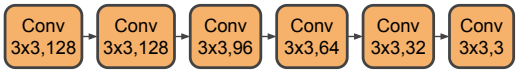


Figure A.4: The optical flow decoder architecture. This is equivalent to that of PWC-Net [159], but with two changes: (1) we do not include warping or cost volume layers, and (2) the output is represented using three channels.

5C. We make no changes to the training procedure or hyper-parameters for these experiments.

We implement non-local blocks similarly to [184], but with two known differences:

1. We do not apply batch norm inside the nonlocal block. Adding batch norm slightly reduced performance.

2. We do not use the sub-sampling trick to reduce the feature map size in the non-local block. This is because the 5X layers in S3D-G already have a small feature map size (7x7x8).

APPENDIX B

Supplementary Materials for Compositional Temporal Grounding

B.1 Qualitative Examples

B.1.1 Comparison with MCN and MLLC

In Figure B.1, we provide examples of challenging instances from Tempo-HL and DiDeMo, and show the temporal segment chosen by our model compared to prior work. Our model reliably localizes difficult queries that are missed by both MCN [55] and MLLC [56].

B.1.2 Examples of Compositional Grounding

In Figure B.2, we provide examples of instances from Tempo-HL and DiDeMo, and show the temporal segments that we identify as sub-events. We show that CTG-Net identifies sensible sub-events, and later uses these to create a final grounding.

B.1.3 Examples Before and After Refinement

In Figure B.3, we provide examples of instances from Tempo-HL and DiDeMo, grounded using CTG-Net before and after the refinement step. Refinement improves groundings.

B.2 Dependency Parser

B.2.1 Example Segmentations

In Figure B.4, we show examples of sub-events identified by CTG-Net with the parser approach. This model identifies sensible sub-events, which can each be grounded in the video.

B.2.2 Distribution of Sub-Events

In Figure B.5, we show the distribution of the number of sub-events identified by the parser in each of the three datasets: DiDeMo, Tempo-TL, and Tempo-HL.

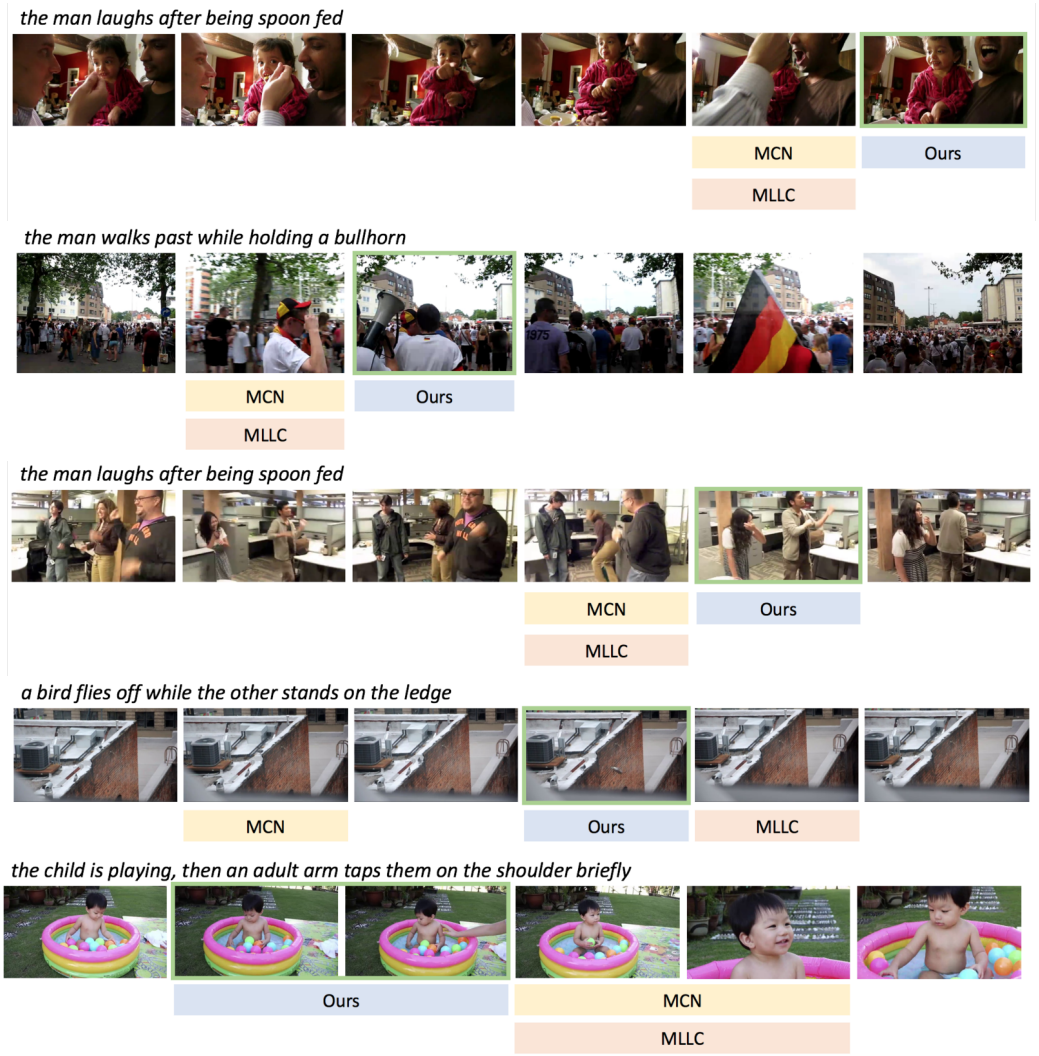


Figure B.1: Example results from the Tempo-HL and DiDeMo training sets. We compare CTG-Net with the Bi-directional LSTM attention mechanism against results from MCN [55] and MLLC [56].

Figure B.2: Example results of temporally grounded sub-events from Tempo-HL and DiDeMo. We CTG-Net model with the dependency parser, and show the individual temporal groundings before combination and refinement. The top 3 examples depict examples of accurate compositional groundings, and the bottom 2 depict incorrect groundings.

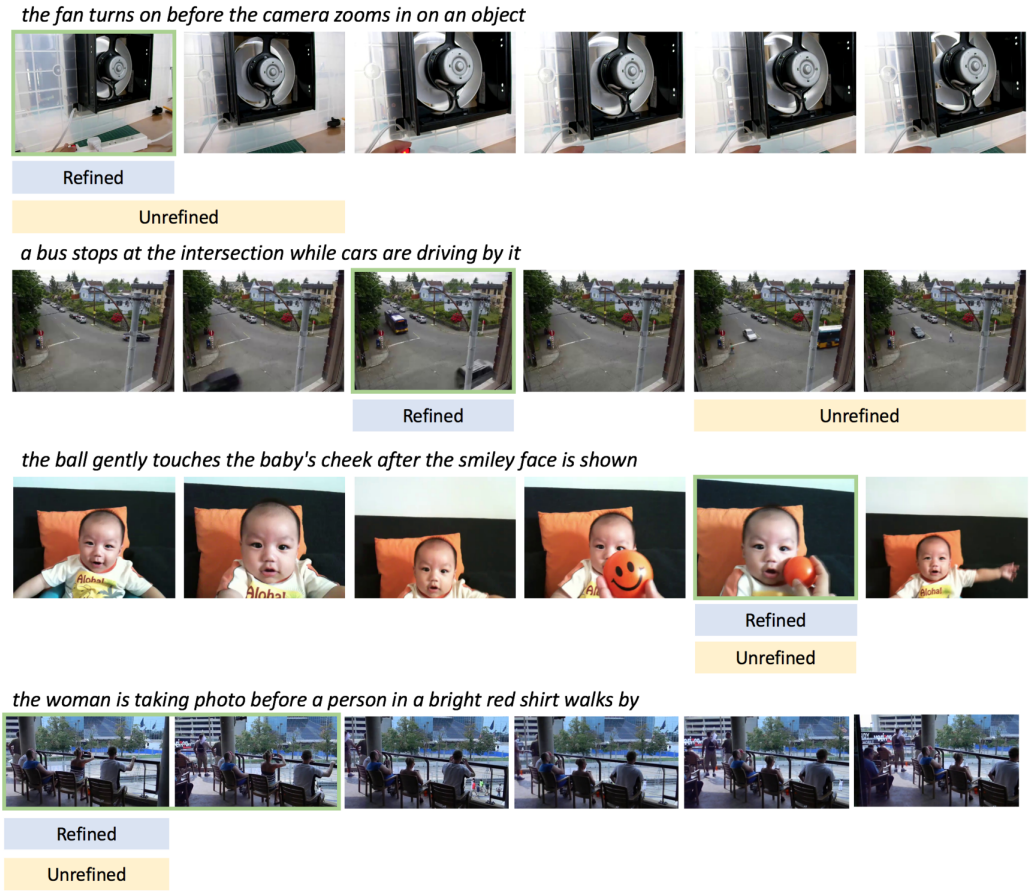


Figure B.3: Example results of CTG-Net before and after the refinement step, shown on Tempo-HL and DiDeMo. In the top 3 examples, we see that the refinement step is able to improve the prediction. In the bottom example, we show a difficult case where the refinement process leads to incorrect predictions.

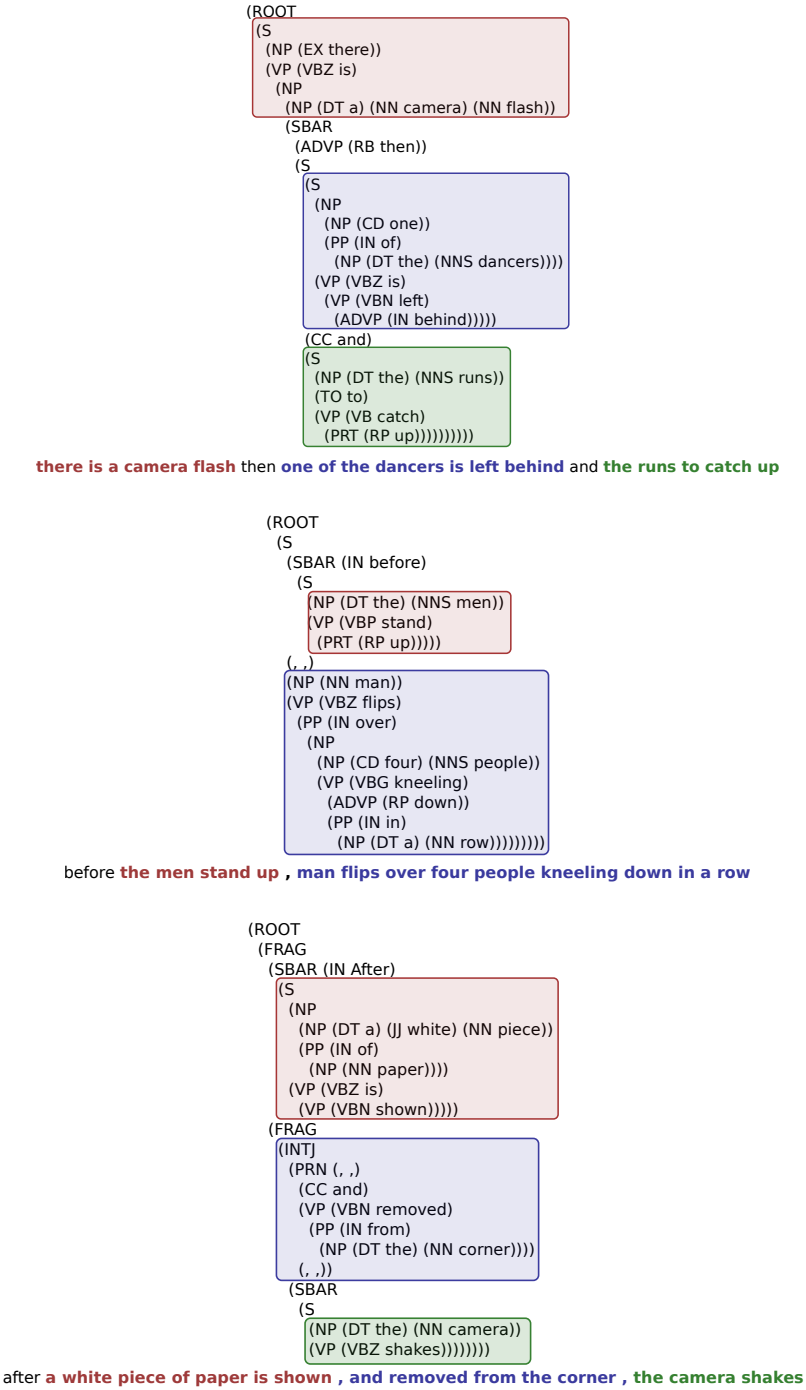


Figure B.4: Example sub-event segmentations produced by the dependency parser.

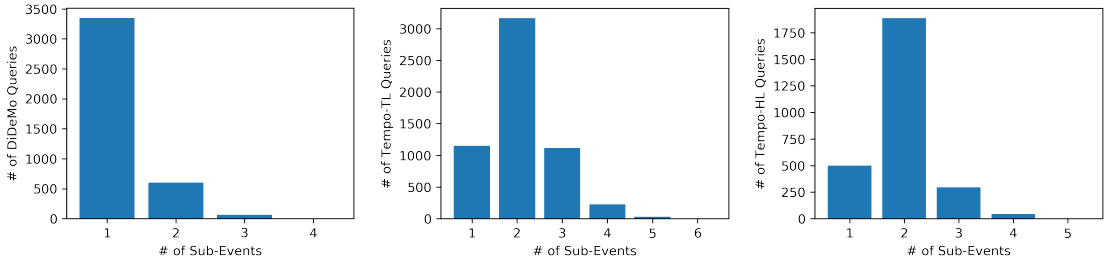


Figure B.5: Distribution of number of sub-events in DiDeMo (left), Tempo-TL (middle) and Tempo-HL (right), as identified by our dependency parser approach. DiDeMo queries are typically shorter and contain few sub-events. Tempo-TL and Tempo-HL are strongly biased towards two sub-events per query.

APPENDIX C

Supplementary Materials for Textual Web Supervision

C.1 Metadata Analysis

We present additional analyses and examples of the metadata in the WVT-70M dataset in Tables C.1, C.1, and C.1.

Metadata	Num. Unique	% Unique
Titles	43.0M	61.5
Descriptions	29.3M	41.9
Tags	34.0M	48.6
Channel Name	21.0M	29.9

Table C.1: Number of unique instances for each metadata type in WVT-70M. All metadata types contain repeats though some are repeated more often than others. Many channels are repeated, and we on average collect 3.3 videos per channel.

Metadata	Min	25	50	75	Max
Titles	0	2	4	6	158
Descriptions	0	0	3	12	4249
Tags	0	0	0	5	161
Channel Name	0	1	2	2	306

Table C.2: Quartiles of length (in words) of each metadata type. All have a long-tailed distribution, meaning that in extreme cases, the metadata may be hundreds or thousands of words long. However, all metadata types also contain examples which are empty or contain zero words. Titles are shortest in the most extreme cases, but longest in the median case.

Metadata	Text	# of Instances
Titles	“Free fire”	92K
	“Dance”	50K
	“Dancing”	47K
	“Baby”	34K
	“Bottle flip”	31K
	“Free Fire”	29K
	“Cute baby”	29K
	“Playing games”	27K
	“Games”	21K
	“Snow”	20K
Tags	“PlayStation 4”	752K
	“Sony Interactive Entertainment”	695K
	“funny”	672K
	“video”	547K
	“mobile”	539K
	“YouTube Capture”	523K
	“#PS4Live”	490K
	“how to”	467K
	“tutorial”	442K
	“fun”	371K

Table C.3: Top ten most often-repeated titles and tags. For titles, these are descriptive and reflect the content of the video. For tags, these often contain automatically-generated metadata which reflect the method by which the video was uploaded.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. *USENIX*, 2016.

[2] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[3] Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Josef Sivic, Ivan Laptev, and Simon Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4575–4583, 2016.

[4] Humam Alwassel, Dhruv Mahajan, Lorenzo Torresani, Bernard Ghanem, and Du Tran. Self-supervised learning by cross-modal audio-video clustering. *arXiv preprint arXiv:1911.12667*, 2019.

[5] Relja Arandjelovic and Andrew Zisserman. Look, listen and learn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 609–617, 2017.

[6] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. Soundnet: Learning sound representations from unlabeled video. In *Advances in neural information processing systems*, pages 892–900, 2016.

[7] Alessandro Bergamo and Lorenzo Torresani. Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach. In *Advances in neural information processing systems*, pages 181–189, 2010.

[8] Yunlong Bian, Chuang Gan, Xiao Liu, Fu Li, Xiang Long, Yandong Li, Heng Qi, Jie Zhou, Shilei Wen, and Yuanqing Lin. Revisiting the effectiveness of off-the-shelf temporal modeling approaches for large-scale video classification. *arXiv preprint arXiv:1708.03805*, 2017.

[9] Cristian Buciluc, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM, 2006.

[10] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018.

[11] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019.

[12] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[13] Carreira, Joao and Noland, Eric. personal correspondence, 2020.

[14] Nathanael Chambers, Shan Wang, and Dan Jurafsky. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 173–176. Association for Computational Linguistics, 2007.

[15] Jingyuan Chen, Xinpeng Chen, Lin Ma, Zequn Jie, and Tat-Seng Chua. Temporally grounding natural sentence in video. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 162–171, 2018.

[16] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. *Neural Information Processing Systems (NIPS)*, 2016.

[17] Xinlei Chen and Abhinav Gupta. Webly supervised learning of convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1431–1439, 2015.

[18] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. Neil: Extracting visual knowledge from web data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1409–1416, 2013.

[19] Yu Cheng, Quanfu Fan, Sharath Pankanti, and Alok Choudhary. Temporal sequence modeling for video event detection. *IEEE International Conference on Computer Vision (CVPR)*, 2014.

[20] Nieves Crasto, Philippe Weinzaepfel, Karteek Alahari, and Cordelia Schmid. Mars: Motion-augmented rgb stream for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7882–7891, 2019.

[21] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. *European Conference on Computer Vision (ECCV)*, 2006.

[22] César Roberto de Souza, Adrien Gaidon, Eleonora Vig, and Antonio Manuel López. Sympathy for the details: Dense trajectories and hybrid classification architectures for action recognition. *European Conference on Computer Vision (ECCV)*, 2016.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[24] Santosh K Divvala, Ali Farhadi, and Carlos Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3270–3277, 2014.

[25] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015.

[26] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1801–1810, 2019.

[27] Lijie Fan, Wenbing Huang, Stefano Ermon Chuang Gan, Boqing Gong, and Junzhou Huang. End-to-end learning of motion representation for video understanding. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

[28] Christoph Feichtenhofer, Axel Pinz, and Richard Wildes. Spatiotemporal residual networks for video action recognition. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[29] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2009.

[30] Rob Fergus, Li Fei-Fei, Pietro Perona, and Andrew Zisserman. Learning object categories from internet image searches. *Proceedings of the IEEE*, 98(8):1453–1466, 2010.

[31] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3636–3645, 2017.

[32] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International Conference on Machine Learning (ICML)*, 2018.

[33] Adrien Gaidon, Zaïd Harchaoui, and Cordelia Schmid. Temporal localization of actions with actoms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2782–2795, 2013.

[34] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Temporal localization of actions with actoms. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2782–2795, 2013.

[35] Chuang Gan, Boqing Gong, Kun Liu, Hao Su, and Leonidas J Guibas. Geometry guided convolutional neural networks for self-supervised video representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5589–5597, 2018.

[36] Chuang Gan, Chen Sun, Lixin Duan, and Boqing Gong. Webly-supervised video recognition by mutually voting for relevant web images and web video frames. In *European Conference on Computer Vision*, pages 849–866. Springer, 2016.

[37] Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. Tall: Temporal activity localization via language query. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5267–5275, 2017.

[38] Ruohan Gao, Bo Xiong, and Kristen Grauman. Im2flow: Motion hallucination from static images for action recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

[39] Nuno C Garcia, Pietro Morerio, and Vittorio Murino. Modality distillation with multiple stream networks for action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 103–118, 2018.

[40] Runzhou Ge, Jiyang Gao, Kan Chen, and Ram Nevatia. Mac: Mining activity concepts for language-based temporal localization. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 245–253. IEEE, 2019.

[41] P Geetha and Vasumathi Narayanan. A survey of content-based video retrieval. In *Citeseer*, 2008.

[42] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12046–12055, 2019.

[43] Rohit Girdhar, João Carreira, Carl Doersch, and Andrew Zisserman. A better baseline for ava. *arXiv preprint arXiv:1807.10066*, 2018.

[44] Yunchao Gong, Qifa Ke, Michael Isard, and Svetlana Lazebnik. A multi-view embedding space for modeling internet images, tags, and their semantics. *International journal of computer vision*, 106(2):210–233, 2014.

[45] Yunchao Gong, Liwei Wang, Micah Hodosh, Julia Hockenmaier, and Svetlana Lazebnik. Improving image-sentence embeddings using large weakly annotated photo collections. In *European conference on computer vision*, pages 529–545. Springer, 2014.

[46] A. Gorban, H. Idrees, Y.-G. Jiang, A. Roshan Zamir, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes, 2014.

[47] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6047–6056, 2018.

[48] Chunhui Gu, Chen Sun, Sudheendra Vijayanarasimhan, Caroline Pantofaru, David A Ross, George Toderici, Yeqing Li, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

[49] James Hale. More than 500 hours of content are now being uploaded to youtube every minute, May 2019.

[50] James Hale. More Than 500 Hours Of Content Are Now Being Uploaded To YouTube Every Minute, 2019.

[51] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[52] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *IEEE International Conference on Computer Vision (CVPR)*, 2016.

[53] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *European Conference on Computer Vision (ECCV)*, 2016.

[54] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. *IEEE International Conference on Computer Vision (CVPR)*, 2015.

[55] Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. Localizing moments in video with natural language. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5803–5812, 2017.

[56] Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. Localizing moments in video with temporal language. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[57] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[58] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[59] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. *European Conference on Computer Vision (ECCV)*, 2012.

[60] De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Connectionist temporal modeling for weakly supervised action labeling. In *European Conference on Computer Vision*, pages 137–153. Springer, 2016.

[61] De-An Huang, Vignesh Ramanathan, Dhruv Mahajan, Lorenzo Torresani, Manohar Paluri, Li Fei-Fei, and Juan Carlos Niebles. What makes a video a video: Analyzing temporal information in video understanding models and datasets. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

[62] Gal Hyams, Dan Malowany, Ariel Biller, and Gregory Axler. Quantifying Diminishing Returns of Annotated Data, 2019.

[63] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[64] Hamid Izadinia, Bryan C Russell, Ali Farhadi, Matthew D Hoffman, and Aaron Hertzmann. Deep classifiers from image tags in the wild. In *Proceedings of the 2015 Workshop on Community-Organized Multimodal Mining: Opportunities for Novel Solutions*, pages 13–18, 2015.

[65] Arpit Jain, Abhinav Gupta, Mikel Rodriguez, and Larry S Davis. Representing videos using mid-level discriminative patches. *IEEE International Conference on Computer Vision (CVPR)*, 2013.

[66] Serena Jeblee and Graeme Hirst. Listwise temporal ordering of events in clinical notes. In *Proceedings of the Ninth International Workshop on Health Text Mining and Information Analysis*, pages 177–182, 2018.

[67] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.

[68] Longlong Jing and Yingli Tian. Self-supervised spatiotemporal feature learning by video geometric transformations. *arXiv preprint arXiv:1811.11387*, 2(7):8, 2018.

[69] Armand Joulin, Laurens van der Maaten, Allan Jabri, and Nicolas Vasilache. Learning visual features from large weakly supervised data. In *European Conference on Computer Vision*, pages 67–84. Springer, 2016.

[70] S. Karaman, L. Seidenari, and A. Del Bimbo. Fast saliency-based pooling of fisher encoded dense trajectories. *THUMOS14 Action Recognition Challenge*, 2014.

[71] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014.

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

[73] Dahun Kim, Donghyeon Cho, and In So Kweon. Self-supervised video representation learning with space-time cubic puzzles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8545–8552, 2019.

[74] Alexander Klaser, Marcin Marszałek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. *British Machine Vision Conference (BMVC)*, 2008.

[75] Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

[76] Savas Konur. An interval logic for natural language semantics. *Advances in Modal Logic*, 7:177–191, 2008.

[77] Bruno Korbar, Du Tran, and Lorenzo Torresani. Cooperative learning of audio and video models from self-supervised synchronization. In *Advances in Neural Information Processing Systems*, pages 7763–7774, 2018.

[78] Hilde Kuehne, Ahsan Iqbal, Alexander Richard, and Juergen Gall. Mining youtube-a dataset for learning fine-grained action concepts from webly supervised video data. *arXiv preprint arXiv:1906.01012*, 2019.

[79] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *International Conference on Computer Vision (ICCV)*. IEEE, 2011.

[80] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[81] Zihang Lai and Weidi Xie. Self-supervised learning for video correspondence flow. *arXiv preprint arXiv:1905.00875*, 2019.

[82] Tian Lan, Yuke Zhu, Amir Roshan Zamir, and Silvio Savarese. Action recognition by hierarchical mid-level action elements. *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[83] Zhengzhong Lan, Ming Lin, Xuanchong Li, Alex G Hauptmann, and Bhiksha Raj. Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. *IEEE International Conference on Computer Vision (CVPR)*, 2015.

[84] Mirella Lapata and Alex Lascarides. Learning sentence-internal temporal relations. *Journal of Artificial Intelligence Research*, 27:85–117, 2006.

[85] Ivan Laptev. On space-time interest points. *International journal of computer vision (IJCV)*, 64(2-3):107–123, 2005.

[86] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. *IEEE International Conference on Computer Vision (CVPR)*, 2008.

[87] Quoc V Le, Will Y Zou, Serena Y Yeung, and Andrew Y Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. *IEEE International Conference on Computer Vision (CVPR)*, 2011.

[88] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. *arXiv preprint arXiv:1611.05267*, 2016.

[89] Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Unsupervised representation learning by sorting sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 667–676, 2017.

[90] Seungeui Lee, Myunggi Lee, Sungjoon Son, Gyutae Park, and Nojun Kwak. Motion feature network: Fixed motion filter for action recognition. In *European Conference on Computer Vision (ECCV)*. IEEE, 2018.

[91] Ang Li, Allan Jabri, Armand Joulin, and Laurens van der Maaten. Learning visual n-grams from web data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4183–4192, 2017.

[92] Tianhao Li and Limin Wang. Learning spatiotemporal features via video and text pair discrimination. *arXiv preprint arXiv:2001.05691*, 2020.

[93] Xiaodan Liang, Liang Lin, and Liangliang Cao. Learning latent spatio-temporal compositional model for human action recognition. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 263–272. ACM, 2013.

[94] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. *European Conference on Computer Vision (ECCV)*, 2014.

[95] Bingbin Liu, Serena Yeung, Edward Chou, De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Temporal modular networks for retrieving complex compositional activities in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 552–568, 2018.

[96] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[97] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016.

[98] Zhichao Lu, Xuehan Xiong, Yinxiao Li, Jonathan Stroud, and David. Ross. Leveraging Weakly Supervised Data and Pose Representation for Action Recognition, 2020.

[99] Zelun Luo, Jun-Ting Hsieh, Lu Jiang, Juan Carlos Niebles, and Li Fei-Fei. Graph distillation for action detection with privileged modalities. In *European Conference on Computer Vision (ECCV)*. IEEE, 2018.

[100] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 181–196, 2018.

[101] Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 753–760. Association for Computational Linguistics, 2006.

[102] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[103] Pyry Matikainen, Martial Hebert, and Rahul Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. *IEEE International Conference on Computer Vision (ICCV) Workshops*, 2009.

[104] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-end learning of visual representations from uncurated instructional videos. *arXiv preprint arXiv:1912.06430*, 2019.

[105] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2630–2640, 2019.

[106] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer, 2016.

[107] Niluthpol Chowdhury Mithun, Rameswar Panda, Evangelos E Papalexakis, and Amit K Roy-Chowdhury. Webly supervised joint embedding for cross-modal image-text retrieval. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1856–1864, 2018.

[108] Philippe Muller and Xavier Tannier. Annotating and measuring temporal relations in texts. In *Proceedings of the 20th international conference on Computational Linguistics*, page 50. Association for Computational Linguistics, 2004.

[109] Arsha Nagrani, Sun Chen, David Ross, Rahul Sukthankar, Cordelia Schmid, and Andrew Zisserman. Speech2action: Cross-modal supervision for action recognition. *CVPR*, 2020.

[110] Alejandro Newell and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. *arXiv preprint arXiv:1611.05424*, 2016.

[111] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. *European Conference on Computer Vision (ECCV)*, 2016.

[112] Joe Yue-Hei Ng, Jonghyun Choi, Jan Neumann, and Larry S Davis. Actionflownet: Learning motion representation for action recognition. In *Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018.

[113] Phuc Nguyen, Ting Liu, Gautam Prasad, and Bohyung Han. Weakly supervised action localization by sparse temporal pooling network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6752–6761, 2018.

[114] Dan Oneata, Jakob J. Verbeek, and Cordelia Schmid. Action and event recognition with fisher vectors on a compact feature set. *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[115] Vicente Ordonez, Girish Kulkarni, and Tamara L Berg. Im2text: Describing images using 1 million captioned photographs. In *Advances in neural information processing systems*, pages 1143–1151, 2011.

[116] Andrew Owens and Alexei A Efros. Audio-visual scene analysis with self-supervised multisensory features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 631–648, 2018.

[117] Andrew Owens, Jiajun Wu, Josh H McDermott, William T Freeman, and Antonio Torralba. Ambient sound provides supervision for visual learning. In *European conference on computer vision*, pages 801–816. Springer, 2016.

[118] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[119] Deepak Pathak, Ross Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. Learning features by watching objects move. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2701–2710, 2017.

[120] Sujoy Paul, Sourya Roy, and Amit K Roy-Chowdhury. W-talc: Weakly-supervised temporal activity localization and classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 563–579, 2018.

[121] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[122] AJ Piergiovanni and Michael S Ryoo. Representation flow for action recognition. *arXiv preprint arXiv:1810.01455*, 2018.

[123] Silvia L Pintea, Jan C van Gemert, and Arnold WM Smeulders. Déja vu. In *European Conference on Computer Vision (ECCV)*. IEEE, 2014.

[124] Hamed Pirsiavash and Deva Ramanan. Parsing videos of actions with segmental grammars. *IEEE International Conference on Computer Vision (CVPR)*, 2014.

[125] Ronald Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010.

[126] Ian Pratt-Hartmann. Temporal prepositions and their logic. *Artificial Intelligence*, 166(1-2):1–36, 2005.

[127] James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. The timebank corpus. In *Corpus linguistics*, volume 2003, page 40. Lancaster, UK., 2003.

[128] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *International Conference on Computer Vision (ICCV)*. IEEE, 2017.

[129] Preethi Raghavan, Eric Fosler-Lussier, Noémie Elhadad, and Albert M Lai. Cross-narrative temporal ordering of medical events. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 998–1008, 2014.

[130] Alexander Richard and Juergen Gall. Temporal action detection using a statistical language model. *IEEE International Conference on Computer Vision (CVPR)*, 2016.

[131] Marcus Rohrbach, Michaela Regneri, Mykhaylo Andriluka, Sikandar Amin, Manfred Pinkal, and Bernt Schiele. Script data for attribute-based recognition of composite activities. *European Conference on Computer Vision (ECCV)*, 2012.

[132] Andrew Rouditchenko, Hang Zhao, Chuang Gan, Josh McDermott, and Antonio Torralba. Self-supervised audio-visual co-segmentation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2357–2361. IEEE, 2019.

[133] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[134] Sreemanananth Sadanand and Jason J Corso. Action bank: A high-level representation of activity in video. *IEEE International Conference on Computer Vision (CVPR)*, 2012.

[135] Florian Schroff, Antonio Criminisi, and Andrew Zisserman. Harvesting image databases from the web. *IEEE transactions on pattern analysis and machine intelligence*, 33(4):754–766, 2010.

[136] Laura Sevilla-Lara, Yiyi Liao, Fatma Guney, Varun Jampani, Andreas Geiger, and Michael J Black. On the integration of optical flow and action recognition. *arXiv preprint arXiv:1712.08416*, 2017.

[137] Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. *arXiv preprint arXiv:1703.01515*, 2017.

[138] Zheng Shou, Hang Gao, Lei Zhang, Kazuyuki Miyazawa, and Shih-Fu Chang. Autoloc: Weakly-supervised temporal action localization in untrimmed videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 154–171, 2018.

[139] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Action temporal localization in untrimmed videos via multi-stage cnns. *arXiv preprint arXiv:1601.02129*, 2016.

[140] Gunnar A Sigurdsson, Santosh Divvala, Ali Farhadi, and Abhinav Gupta. Asynchronous temporal fields for action recognition. *arXiv preprint arXiv:1612.06371*, 2016.

[141] Gunnar A Sigurdsson, Santosh Kumar Divvala, Ali Farhadi, and Abhinav Gupta. Asynchronous temporal fields for action recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[142] Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. *European Conference on Computer Vision (ECCV)*, 2016.

[143] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[144] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *Neural Information Processing Systems (NIPS)*, 2014.

[145] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[146] Bharat Singh and Ming Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. *IEEE International Conference on Computer Vision (CVPR)*, 2016.

[147] Gurkirt Singh and Fabio Cuzzolin. Untrimmed video classification for activity detection: submission to activitynet challenge. *arXiv preprint arXiv:1607.01979*, 2016.

[148] Yale Song, Louis-Philippe Morency, and Randall Davis. Action recognition by hierarchical sequence summarization. *IEEE International Conference on Computer Vision (CVPR)*, 2013.

[149] Khurram Soomro, Amir Roshan Zamir, and M Shah. A dataset of 101 human action classes from videos in the wild. *Center for Research in Computer Vision*, 2, 2012.

[150] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[151] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. *ICML*, 2015.

[152] Jonathan C Stroud, Ryan McCaffrey, Rada Mihalcea, Jia Deng, and Olga Russakovsky. Compositional temporal visual grounding of natural language event descriptions. *arXiv preprint arXiv:1912.02256*, 2019.

[153] Jonathan C. Stroud, David A. Ross, Chen Sun, Jia Deng, and Rahul Sukthankar. D3d: Distilled 3d networks for video action recognition. In *Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2020.

[154] Jonathan C Stroud, David A Ross, Chen Sun, Jia Deng, Rahul Sukthankar, and Cordelia Schmid. Learning video representations from textual web supervision. *arXiv preprint arXiv:2007.14937*, 2020.

[155] Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. Contrastive bidirectional transformer for temporal representation learning. *arXiv preprint arXiv:1906.05743*, 2019.

[156] Chen Sun and Ram Nevatia. Active: Activity concept transitions in video event classification. *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[157] Chen Sun, Sanketh Shetty, Rahul Sukthankar, and Ram Nevatia. Temporal localization of fine-grained actions in videos by domain transfer from web images. *ACM Multimedia Conference*, 2015.

[158] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *International Conference on Computer Vision (ICCV)*. IEEE, 2017.

[159] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

[160] Lin Sun, Kui Jia, Dit-Yan Yeung, and Bertram E Shi. Human action recognition using factorized spatio-temporal convolutional networks. *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[161] Shuyang Sun, Zhanghui Kuang, Lu Sheng, Wanli Ouyang, and Wei Zhang. Optical flow guided feature: A fast and robust motion representation for video action recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

[162] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[163] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015.

[164] Reuben Tan, Huijuan Xu, Kate Saenko, and Bryan A Plummer. wman: Weakly-supervised moment alignment network for text-based video segment retrieval. *arXiv preprint arXiv:1909.13784*, 2019.

[165] Kevin Tang, Li Fei-Fei, and Daphne Koller. Learning latent temporal structure for complex event detection. *IEEE International Conference on Computer Vision (CVPR)*, 2012.

[166] Graham W Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. *European Conference on Computer Vision (ECCV)*, 2010.

[167] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

[168] T. Tieleman and G. Hinton. RMSprop Gradient Optimization. *CSC321 Lecture Slides*, 2014.

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

[170] Du Tran, Jamie Ray, Zheng Shou, Shih-Fu Chang, and Manohar Paluri. Convnet architecture search for spatiotemporal feature learning. *arXiv preprint arXiv:1708.05038*, 2017.

[171] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

[172] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. *IEEE International Conference on Computer Vision (CVPR)*, 2016.

[173] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances in neural information processing systems*, pages 613–621, 2016.

[174] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 391–408, 2018.

[175] Jacob Walker, Abhinav Gupta, and Martial Hebert. Dense optical flow prediction from a static image. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015.

[176] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *International Conference on Computer Vision (ICCV)*. IEEE, 2013.

[177] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[178] Jiangliu Wang, Jianbo Jiao, Linchao Bao, Shengfeng He, Yunhui Liu, and Wei Liu. Self-supervised spatio-temporal representation learning for videos by predicting motion and appearance statistics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4006–4015, 2019.

[179] Limin Wang, Wei Li, Wen Li, and Luc Van Gool. Appearance-and-relation networks for video classification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

[180] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition and detection by combining motion and appearance features. *THUMOS14 Action Recognition Challenge*, 2014.

[181] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. *IEEE International Conference on Computer Vision (CVPR)*, 2015.

[182] Limin Wang, Yuanjun Xiong, Dahua Lin, and Luc Van Gool. Untrimmednets for weakly supervised action recognition and detection. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4325–4334, 2017.

[183] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016.

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

[185] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning correspondence from the cycle-consistency of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2566–2576, 2019.

[186] Xin-Jing Wang, Lei Zhang, Xirong Li, and Wei-Ying Ma. Annotating images by mining image search results. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1919–1932, 2008.

[187] Donglai Wei, Joseph J Lim, Andrew Zisserman, and William T Freeman. Learning and using the arrow of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8052–8060, 2018.

[188] Daniel Weinland, Remi Ronfard, and Edmond Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer vision and image understanding*, 115(2):224–241, 2011.

[189] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *European Conference on Computer Vision (ECCV)*. IEEE, 2018.

[190] Yuanjun Xiong, Yue Zhao, Limin Wang, Dahua Lin, and Xiaoou Tang. A pursuit of temporal accuracy in general activity detection. *arXiv preprint arXiv:1703.02716*, 2017.

[191] Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10334–10343, 2019.

[192] Huijuan Xu, Abir Das, and Kate Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. *arXiv preprint arXiv:1703.07814*, 2017.

[193] Ran Xu, Caiming Xiong, Wei Chen, and Jason J Corso. Jointly modeling deep video and compositional text to bridge vision and language in a unified framework. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[194] Zhongwen Xu, Yi Yang, and Alex G Hauptmann. A discriminative cnn video representation for event detection. *IEEE International Conference on Computer Vision (CVPR)*, 2015.

[195] Angela Yao, Jürgen Gall, and Luc J. Van Gool. A hough transform-based voting framework for action recognition. *IEEE International Conference on Computer Vision (CVPR)*, 2010.

[196] Shoou-I Yu, Lu Jiang, and Alexander Hauptmann. Instructional videos for unsupervised harvesting and learning of action examples. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 825–828, 2014.

[197] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. *IEEE International Conference on Computer Vision (CVPR)*, 2015.

[198] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Joint Pattern Recognition Symposium*. Springer, 2007.

[199] Bowen Zhang, Limin Wang, Zhe Wang, Yu Qiao, and Hanli Wang. Real-time action recognition with enhanced motion vector cnns. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.

[200] Da Zhang, Xiyang Dai, Xin Wang, Yuan-Fang Wang, and Larry S Davis. Man: Moment alignment network for natural language moment retrieval via iterative graph adjustment. *arXiv preprint arXiv:1812.00087*, 2018.

[201] Hong Jiang Zhang, Jianhua Wu, Di Zhong, and Stephen W Smoliar. An integrated system for content-based video retrieval and browsing. *Pattern recognition*, 30(4):643–658, 1997.

[202] Hang Zhao, Chuang Gan, Andrew Rouditchenko, Carl Vondrick, Josh McDermott, and Antonio Torralba. The sound of pixels. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 570–586, 2018.

[203] Luowei Zhou, Chenliang Xu, and Jason J Corso. Towards automatic learning of procedures from web instructional videos. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[204] Yi Zhu, Zhenzhong Lan, Shawn Newsam, and Alexander G Hauptmann. Hidden two-stream convolutional networks for action recognition. *arXiv preprint arXiv:1704.00389*, 2017.

[205] Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David Fouhey, Ivan Laptev, and Josef Sivic. Cross-task weakly supervised learning from instructional videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3537–3545, 2019.