Learning to localize goal-oriented actions with weak supervision

Dimitri Zhukov

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à École Normale Supérieure

Learning to localize goal-oriented actions with weak supervision

Soutenue par
Dimitri ZHUKOV
Le 16 Décembre 2021

École doctorale n°386
Sciences Mathématiques de Paris Centre

Spécialité
Informatique

Composition du jury :

Matthieu Cord Sorbonne University	*Président du jury*
Efstratios Gavves University of Amsterdam	*Rapporteur*
Jason Corso University of Michigan	*Rapporteur*
Gül Varol Ecole des Ponts ParisTech	*Examinateur*
Ivan LAPTEV Inria	*Directeur de thèse*
Josef SIVIC Czech Technical University, CIIRC	*Directeur de thèse*

ENS | PSL

Abstract

The goal of this thesis is to develop methods for automatic understanding of video content. We focus on instructional videos, that demonstrate how to perform complex tasks, such as making an omelette or hanging a picture. Such videos usually show a sequence of goal-oriented actions, or steps, required to complete the task. Instructional videos demonstrate thousands of highly diverse tasks and are usually several minutes long, making supervised learning approach impractical, as human annotations are costly and hard to scale. In this thesis, we focus, instead, on weakly-supervised and unsupervised scenarios. In our first contribution, we investigate learning visual models for the steps of tasks, using only the sequential order of steps, instead of strong supervision via temporal annotations. We propose a model that allows to share the information between tasks on the sub-step level, effectively increasing the amount of training data available for each step. This is done by decomposing each step into a set of components, which are shared across tasks. To demonstrate the benefits of sharing, we collect a new dataset of instructional videos, CrossTask, and perform a series of experiments with our model and the baselines on this newly collected data. In our second contribution, we present a model that learns to isolate task-related actions from the surrounding background. We build on the observation that actions, unlike the background, often follow a specific temporal order and can be predicted by other actions in the same video. Our model is trained using a proxy task of predicting the order between different clips, taken from the same video, and learns to assign higher actionness scores to the clips, that contain stronger order clues. We apply our method to the tasks of action vs background classification, action proposal generation and action localization. Our final contribution is focused on 3D reconstruction and spatial language grounding in instructional videos. Given a set of narrated instructional videos, sharing a common scene, our goal is to reconstruct the 3D scene and learn language grounding within this 3D scene. We approach this problem by, first, performing 3D reconstruction from each individual video, and, then, aligning all reconstructions in the 3D space. This allows to reduce the computational complexity of the problem, as well as to cope with appearance variations across videos. Finally, we use narrations in videos, as well as the obtained 3D reconstruction, to learn 3D language grounding in unsupervised way. We evaluate our method on car maintenance videos, showing that our method successfully associates textual instructions with objects in the 3D scene.

Resumé

Le but de cette thèse est de développer des méthodes pour la compréhension automatique des vidéos. Nous nous intéressons des vidéos d'instructions, qui démontrent des tâches humaines, composées de plusieurs actions, comme, par exemple, faire une omelette ou accrocher une peinture. Ces vidéos étant long et variés, des annotations manuelles sont difficiles à obtenir à la grande échelle. Pour cette raison, nous considérons le cas de l'apprentissage faiblement ou non supervisée. Dans un premier temps, nous proposons une méthode d'apprentissage des actions seulement à partir d'un script pour chaque tâche, au lieu des segments temporels annotés dans chaque vidéo. Afin de réduire la quantité de données d'entrainement, nous décomposons chaque action en ensemble de composantes et partageons ces composantes entre les tâches. Nous évaluons notre méthode dans une série d'expérience sur un nouveau jeu de données, CrossTask. Notre deuxième contribution est une méthode qui permet d'isoler des actions liées aux tâches de leur contexte. Nous observons que les actions apparaissent dans la vidéo dans l'ordre spécifique, ce qui n'est pas le cas pour le reste de la vidéo et proposons un modèle qui permet de vérifier l'ordre des fragments de la vidéo, ainsi qu'y attribuer des valeurs d'actionness en fonction de leurs utilités pour la vérification de l'ordre. Notre modèle est entraîné sur la tâche de la vérification de l'ordre sans avoir besoin de l'annotation manuelle. Nous évaluons notre méthode sur les tâches de la classification entre les actions et leur contexte, de la génération des propositions d'actions et de leur localisation. Notre dernière contribution est dédiée à la reconstruction 3D et au raisonnement spatial à partir des vidéos d'instructions. Nous considérons le cas où toutes les vidéos partagent la même scène, ayant pour l'objectif de reconstruire cette scène en 3D et d'apprendre y associer le langage naturel. Nous commençons par la reconstruction des scènes 3D à partir des vidéos individuelles. Ces reconstructions sont ensuite alignées dans l'espace 3D commun. Cela permet de réduire la complexité computationnelle du problème, ainsi que de réduire l'effet des différences visuelles entre les vidéos, qui compliquent la reconstruction directe à partir de l'ensemble de vidéos. Muni de la reconstruction 3D et des sous-titres des vidéos, nous entraînons, finalement, un modèle qui trouve une correspondance entre le texte et des différentes parties de la scène. Nous effectuons une série d'expérience sur les vidéos de la maintenance de voiture et démontrons que notre modèle est capable d'associer des instructions textuelles aux objets correspondants dans la scène 3D.

Acknowledgments

First, I would like to thank my PhD advisors, Ivan and Josef, for giving me an opportunity to pursue a thesis in Willow team and for providing me with ideas and guidance over these years. None of the contributions, presented in this thesis, would be possible without their help.

I thank Efstratios Gavves and Jason Corso for accepting the role of rapporteurs of my thesis, and Matthieu Cord and Gül Varol for agreeing to participate in the jury.

I am grateful to Francis Bach, Alessandro Rudi and Pierre Gaillard for being part of my comité de suivi doctoral. I would also like to thank Francis for his technical advice and for his feedback.

I would like to thank all my colleagues from Willow and Sierra teams, in particular, my office neighbors, Thomas, Adrian, Pascal and Remi. I thank Antoine and Makarand for our collaboration. I would especially like to thank Jean-Baptiste for co-supervising my research internship, for his collaboration in most of my projects, and for his invaluable advise in how to manage my thesis. I would like to thank Kim, Sabrine, Hélène and Mathieu for helping me with administrative procedures.

I would like to thank all researchers with whom I had luck to collaborate: David Fouhey, Gökberk Cinbiş, Johannes Schönberger, Bugra Tekin and Marc Pollefeys.

Finally, I thank my family for being so supportive. This thesis is dedicated to my wife, Viktoriia, for her constant patience and support.

Contents

Chapter 1

Introduction

1.1 Goal

The goal of this thesis is to develop methods and models for understanding videos of complex human tasks. Such videos, that we refer to as instructional videos, typically show one or multiple persons, performing a sequence of actions, required to complete the task, such as cooking an omelette or changing car battery. Figure 1.1 shows an example of an instructional video for the task of making pancakes. Understanding the content of instructional videos requires answering questions such as what actions are performed at every given moment, which parts of the video are important within the context of the task, and where exactly each action occurs within the scene.

Instructional videos are usually several minutes long and include multiple visual events, such as performing actions, demonstrating objects, as well as video segments, which are irrelevant for the task. For this reason, we focus on the methods, that are able to produce fine-grained representations of the video content. For example, instead of recognizing the task in the video as making an omelette, we want to be able to recognize each atomic step of the task, such as cracking and whisking eggs, and localize it within the video.

Video hosting platforms such as YouTube provide a vast amount of instructional videos, focused on different tasks, ranging from cooking to furniture assembly to car repairs. As

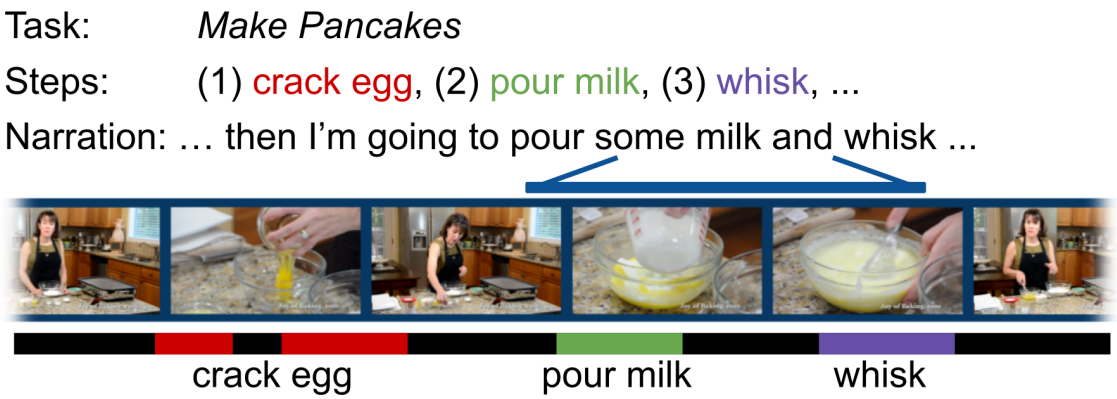


Figure 1.1: An example of instructional video for the task *how to make pancakes*. The video shows a sequence of *steps*, *i.e.* actions, required to complete a task. Each step is represented by one or multiple segments of the video, while the remaining parts of the video are not important for understanding of the task (*background* in black). Narration of the video may but not guarantied to mention each step.

by the time of writing this thesis, the largest dataset, focusing around the web instructional videos, HowTo100M (Figure 1.2), introduced by [Miech et al. 2019], includes more than 1 million videos, more than 23 thousands different tasks, and a total of 15 years of video recording. In order for our methods to work on such scale, they must require only a small amount of human annotations or no annotations at all. Instead of relying on accurate human annotations, our approach is to use available metadata and make natural assumptions about the videos, in order to reduce the amount of manual annotations, required to learn high quality representations. Some examples of such assumptions, that we make in our work, are listed below.

First, we may assume that the task is known. The task, shown in a YouTube video can often be deduced from its title and its description, without requiring to watch the video itself. Moreover, we may assume that the list of steps, required to complete the task is also provided. Indeed, given a name of the task, we can search for it on the how-to websites, such as WikiHow.com, and parse them in order to obtain a step-by-step guide.

Second, instructional videos are often accompanied by author's commentary, explaining what happens in the video. We may assume that narrations in instructional videos describe the actions, occurring in the video. Furthermore, we may assume that the narrations are roughly synchronized with the visual stream. This means that people

Figure 1.2: **HowTo100m dataset**. The largest existing dataset of narrated YouTube videos, at the time of writing this thesis, collected by querying different tasks, such as *how to grill steak* or *how to paint furniture*, HowTo100m encompases a variety of topics, ranging from *food* to *computers and electronics*. Each video is accompanied by narration in the form of subtitles, generated either manually or with the help of automatic speech recognition.

tend to explain what and how they are doing around the same time as they are doing it.

Finally, in many cases the actions that constitute a task, must be performed in a particular order. For example, in order to make an omelette, you need to break the eggs, before whisking them. Of course, many real world tasks allow variations in the order in which the steps are performed. Nevertheless, knowing a rough order of steps can simplify automatic understanding of instructional videos even in the case of such tasks.

1.2 Motivation

Web instructional videos are a large and readily available source of knowledge about various human activities. Such videos, posted on web platforms like YouTube are intended for teaching a human viewer how to complete all sorts of useful tasks. To achieve this goal, a typical instructional video demonstrates the objects involved in the task, describes what actions must be performed and how to perform them, as well as what the final result should look like. In order to better communicate the process of solving a task to an unskilled viewer, the video stream in instructional videos is often accompanied by

a commentary in the form of natural language. Both visual demonstration and verbal explanation are standard means for teaching human new skills, but can we use the same videos to teach machines the same skills?

First of all, why would we want to make machines understand human tasks? Imagine a robot, cooking your breakfast or repairing your car. Or smart glasses, such as Microsoft HoloLens, analyzing your actions and providing useful visual and linguistic hints, that would help you successfully complete the task. These and other useful applications may potentially be engineered for a single task, by manually translating human knowledge into the machine code. However, the numbers of tasks, that people have to address in their daily life and in their professional activities scale in thousands. This motivates a research towards an automatic understanding of human tasks.

Can we use the same source of knowledge, that people use, namely the instructional videos, to teach machines? The answer to this question is not straightforward. While vision and language are commonly used sources of information for a human, they are not directly digestible by computer and require development of suitable computer vision and natural language processing methods.

1.3 Challenges

Automatic video understanding has seen much progress over the recent years, with the main focus on supervised action recognition in trimmed videos. Such methods, however, are ill suited for the case of instructional videos, which are usually long, untrimmed and hard to annotate. In this section we explain in detail these and other challenges that arise, when learning from instructional videos.

Unscalable annotations. Supervised learning require high quality human annotations, that may be hard to obtain. This is especially pronounced in the case of instructional videos due to their long duration and a high number of distinct visual events that occur in a single video. Training a model to recognize such events requires annotations on sub-video scale, instead of annotating a video as a whole. This implies that a human annotator has to watch the entire video and annotate small fragments of it. Furthermore,

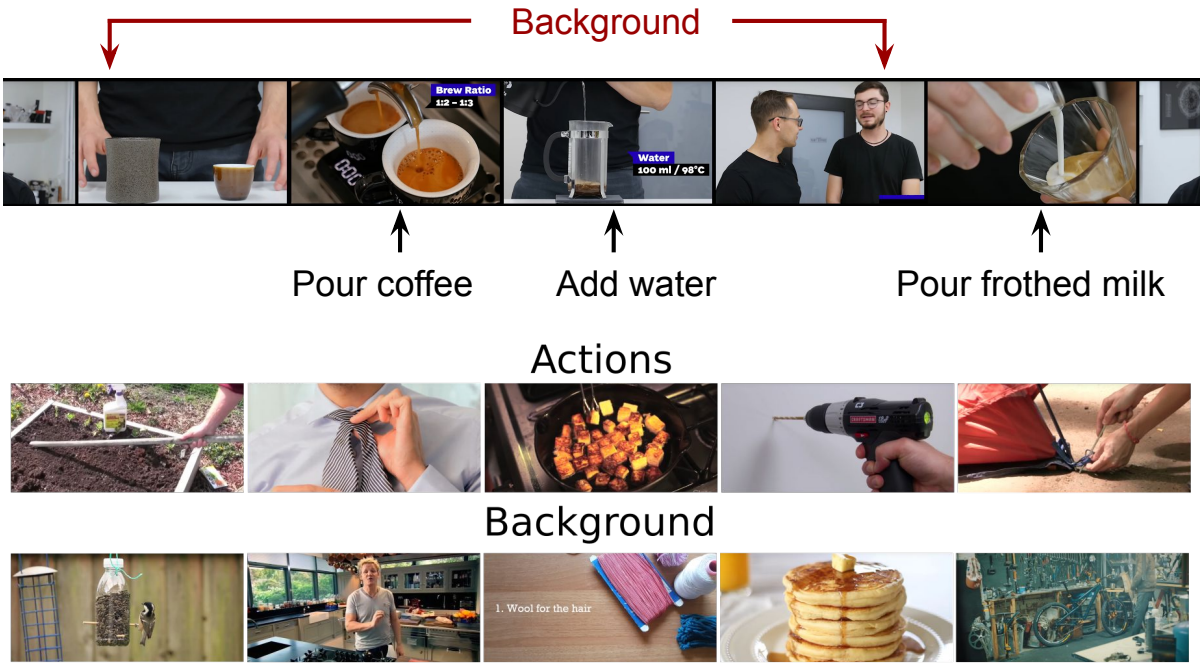


Figure 1.3: Examples of actions and background in instructional videos

some aspects of annotation process may be highly ambiguous. For example, what level of granularity do we need? Should we annotate actions like cutting onions and cutting tomatoes with the same label "cut ingredients", despite their difference in meaning and visual appearance? Or do we want to use different labels and ignore possible similarities? What should be considered as an atomic action? Another question is how to define the temporal extents of each action. What defines the beginning and the end of an action? These questions make the task of annotating the videos challenging and poorly defined.

Variation in appearance. In this thesis we face high level of visual variation, typical for computer vision problems. This includes viewpoint variations, variations in appearance of objects and actions, differences in backgrounds and in scene layouts, different illumination and other sources of variation. The videos can be filmed in first or third person perspective. Finally, the task, shown in the video, itself may be a subject of variations. For example, the actions that constitute the task may differ, and some actions may be optional.

Noisy data. Web instructional videos are oriented towards human audience and are

not designed to be used as training data for computer vision models. Furthermore, there is usually no quality control for such videos. This introduces a set of challenges for automatic video understanding. First, a typical video contains background segments, which are not related to the task (Figure 1.3). It is crucial for a model to separate such background from meaningful content. Second, a person may have multiple unsuccessful attempts to perform an action, before performing it correctly. Actions and objects may be occluded, and some actions may be partially or completely cut out of the video, if deemed trivial by the author.

Limited data. As mentioned above, instructional videos are available on web in high quantities. This abundance is partly explained by the variety of different tasks. However, finding more than a hundred video demonstrations on YouTube proofs challenging even for the most popular tasks. Less popular or more fine-grained tasks may provide only a few video examples. Taking into account the level of noise and variation in the data, this constitutes a significant challenge for computer vision models, which usually require a high number of training examples even in the supervised case.

1.4 Contributions

The contributions of this thesis are threefold. First, we propose a method for learning steps of tasks, that benefits from information sharing between different tasks in order to alleviate the data limitations. To demonstrate the benefits of such sharing, we introduce a new dataset of instructional videos, CrossTask, that contains clusters of similar tasks (see Figure 1.4), that can benefit from sharing representations. Second, we introduce an unsupervised method for learning to separate the task-related actions within instructional videos from temporal background. Third, we propose an approach for reconstructing a common 3D scene from multiple videos, and grounding language in the obtained 3D reconstruction in unsupervised way.

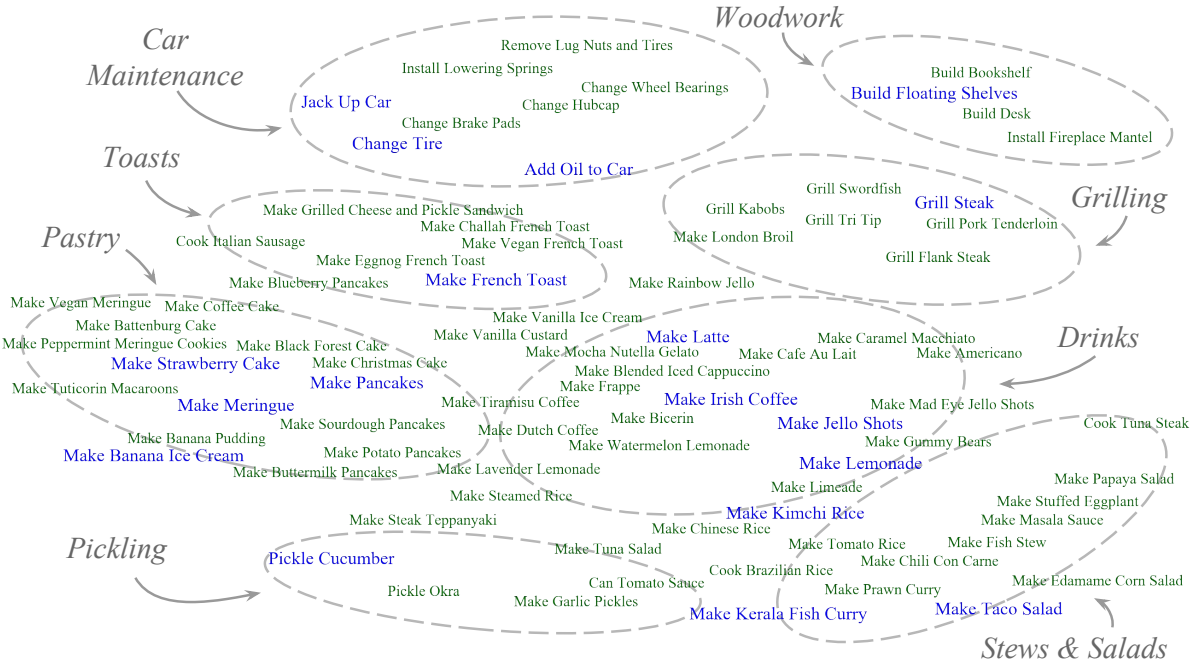


Figure 1.4: **CrossTask dataset.** This dataset includes narrated instructional videos, grouped by tasks. Videos from primary tasks are selected manually and annotated with temporal action segments. Videos from related tasks are collected automatically using YouTube search engine and are not provided with action-level annotations.

1.4.1 Cross-task weakly supervised learning from instructional videos

One of the challenges for automatic task understanding from instructional videos is a limited number of available videos per task. However, by decomposing a task into a set of individual steps, we may notice that many tasks share similarities. Same actions and objects may be involved in different tasks. For example, whisking eggs is a common action, involved in many cooking tasks. In this work we propose a model that allows to explicitly share the information between tasks in the form of step components, representing primitive concepts, such as "whisk" and "egg". Our model represents each step on the level of its components. Component representations are then aggregated to represent the step. Since the same component may be a part of different steps across many tasks, this approach effectively increases the amount of available training data for each step. We also provide a new dataset of instructional videos, CrossTask, specifically designed

to include many similar tasks that may benefit from sharing.

1.4.2 Learning actionness via long-range temporal order verification

Web instructional videos usually contain a significant amount of a background noise. In a typical instructional video, task-related action segments interchange with the video segments, which are not important for understanding of the task. Being able to distinguish action segments from background is an important part of action localization. In this work we propose an unsupervised approach for isolating actions from background in instructional videos. Our approach is based on the observation that task-related actions usually appear in a particular order, which makes it possible to deduce the order between the corresponding video segments. On the contrary, the background segments appear in arbitrary parts of the video and do not contain any information, that would allow to predict their order relative to the other parts of the video. We formalize this observation in a form of a proxy task of predicting order between different segments of the same video. Our model learns to assign score to each segment of the video based on how easy it is to predict the correct order between this segment and other segments of the same video. We evaluate our model on two different tasks: frame-wise action vs background classification and temporal action proposal generation and compare against multiple unsupervised baselines to demonstrate the benefits of our approach.

1.4.3 Reconstructing and grounding narrated instructional videos in 3D

Our third and final contribution is focused around the spatial relations within instructional videos. Spatial relations in instructional videos are generally very noisy and are often not important for understanding the task. For example, in a cooking video, different ingredients and equipment may be located anywhere in the kitchen. However, in some cases, understanding spatial relations can prove very helpful. In this contribution, we investigate one such case, the car engine maintenance, with the goal of grounding natural

language instructions to the corresponding locations within the 3D model of car engine. This is achieved in two steps. First, we obtain a common 3D reconstruction of the car engine from multiple videos. Second, we use narrations to learn natural language grounding in 3D without manual supervision. We evaluate our model on the task of grounding natural language instructions from WikiHow.com in the 3D model and comparing predicted locations with ground truth.

1.5 Publications

This thesis is based on three following publications.

- [Zhukov et al. 2019]: Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David Fouhey, Ivan Laptev and Josef Sivic. Cross-task weakly supervised learning from instructional videos. In Proc. CVPR, 2019. (Chapter 3)
- [Zhukov et al. 2020]: Dimitri Zhukov, Jean-Baptiste Alayrac, Ivan Laptev and Josef Sivic. Learning actionness via long-range temporal order verification. In Proc. ECCV, 2020. (Chapter 4)
- [Zhukov et al. 2021]: Dimitri Zhukov, Ignacio Rocco, Ivan Laptev, Josef Civic, Bugra Tekin, Johannes Schönberger, Marc Pollefeys. Reconstructing narrated instructional videos in 3D. ArXiv, 2021. (Chapter 5)

The following work has been done in collaboration with A. Miech and is excluded from thi thesis.

- [Miech et al. 2019]: Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In ICCV, 2019.

1.6 Dataset

We have publicly released the CrossTask dataset[1]. We provide YouTube links, video features, used in [Zhukov et al. 2019], as well as the annotations. The annotations include

task labels for all videos, action scripts for all tasks, as well as temporal action segments for 2750 videos.

1.7 Outline

This thesis is organized into six chapters including this introduction. In Chapter 2 we provide a literature review of the previous work, related to the subject of this thesis. In Chapter 3 we present our first contribution on cross-task learning from instructional videos and the CrossTask dataset. In Chapter 4 we describe our second contribution on learning actionness from instructional videos via long-range temporal order verification. Chapter 5 provides details on our third contribution on reconstructing and grounding narrated instructional videos in 3D. Finally, in Chapter 6, we summarize our contributions and discuss the limitations of proposed methods, as well as open problems.

[1]https://github.com/DmZhukov/CrossTask

Chapter 2

Literature review

2.1 Action recognition

Actions play an essential role in instructional videos and are crucial for their understanding. Action recognition requires understanding both the visual content within the still images (*e.g.* appearance of objects and relations between them), and how it changes over time (*e.g.* change in object states). In Section 2.1.1 we describe representations and models, used for action recognition. We follow this up by discussing the most common datasets, used to train and evaluate these models in Section 2.1.2.

2.1.1 Video representation

Object recognition in videos requires jointly modeling spatial and temporal information. [Laptev 2005] introduced space-time interest point (STIP) descriptors, by extending the notion of image interest points to the spatio-temporal domain. [H. Wang et al. 2011] proposed dense trajectory (DT) descriptors, obtained by densely sampling points within the frames, tracking them in the video with the help of the optical flow, and, finally, computing local descriptors, such as histogram of oriented gradient (HOG), histogram of optical flow (HOF) and motion boundary histograms (MBH) along these trajectories. [H. Wang et al. 2013] later proposed improved dense trajectories (IDT), that allow to

reduce the effects of the camera movement.

Following the success of convolutional neural networks (CNN) in image recognition [Krizhevsky et al. 2012], much effort was put into developing similar deep architectures for action recogntion. [Karpathy et al. 2014b] explored three different ways to adapt CNNs for video representation. First, they proposed to expand the filters of the first convolutional layer to process information from a fixed temporal window. Second, they ran two 2D convolutional networks with shared parameters on different frames, and then fused their outputs. Finally the authors considered adding temporal dimension to all convolutional filters in the network. [Bilen et al. 2016] applied 2D CNN, pretrained on the task of image recognition, to action recognition in the video, by first producing a dynamic image, that summarizes the motion within the video, via rank pooling [Fernando et al. 2015], and then finutune the network on such dynamic images.

[Simonyan et al. 2014] proposed a two-stream network, that consists of two 2D CNNs. First branch of the network takes RGB frames as an input and represents the visual appearance within the frames. Second branch takes a sequence of optical flow images and models the motion. The outputs of both networks are then fused to predict the action. This model has shown a significant improvement, compared to the previous approaches, and similar two-stream CNNs were later used by [Feichtenhofer et al. 2017; Limin Wang et al. 2016b].

[S. Ji et al. 2013; Tran et al. 2015] proposed to model spatio-temporal structure in the videos with 3D space-time convolutions. [Varol et al. 2017] later investigated learning long-range temporal structure in the videos with 3D CNNs. [Carreira et al. 2017] introduced two-stream inflated 3D convolutional network (I3D). Similarly to [Simonyan et al. 2014], the network consists of two separate branches, that take RGB frames and optical flow as an input respectively. However, unlike [Simonyan et al. 2014], both branches are 3D CNNs. The architecture of this 3D CNN is derived from 2D CNN, by adding temporal resolution to 2D convolution filters. Furthermore, the authors proposed a way to initialize the network with the original 2D CNN, pretrained on images. This is done by "inflating" 2D kernels, *i.e.* copying their values along the temporal axis. This initialization allows to reduce the amount of videos, required to train a 3D CNN.

Aiming at reducing the number of parameters in 3D convolutional networks, [Qiu et al. 2017] proposed P3D ResNet, where 3D convolution kernels are factored as a composition of spatial and temporal kernels. This was further investigated by [Tran et al. 2018] and [Xie et al. 2018], who demonstrated that such decomposed 2+1D convolutional networks can outperform ordinary 3D CNNs, despite having significantly less parameters.

2.1.2 Datasets

The KTH dataset [Schüldt et al. 2004] includes 2391 manually recorded videos with 6 types of actions (walking, jogging, running, boxing, hand waving and hand clapping), performed by 25 different subjects. Manual recording of videos allows for a high level of control over their content (*e.g.* what actions occur in the video). However, its high cost has motivated attempts at collecting videos from readily available sources of data, such as movies and web video sharing platforms. In particular, [Laptev et al. 2008] proposed Hollywood dataset, that consists of 687 videos and 8 types of actions, collected from movies. This dataset was later extended to Hollywood2 by [Marszalek et al. 2009] to include 1694 videos with 12 action classes.

[J. Liu et al. 2009] introduced a dataset of 1600 videos with 11 action classes, collected from YouTube. Using video sharing platforms as a source of data paved the way to larger datasets with higher action diversity, most notably, HMDB-51 [Kuehne et al. 2011] (6.8K videos, 51 action class) and UCF-101 [Soomro et al. 2012] (13K videos, 101 action class). HMDB-51 was later extended to J-HMDB [Jhuang et al. 2013], by providing annotations of human joints. [Karpathy et al. 2014b] introduced Sports-1M, that contains 1 million videos with 487 action classes, focused around sports. The authors aimed at collecting sufficiently large amount of data, required to train deep convolutional networks for action recognition. ActivityNet dataset [Caba Heilbron et al. 2015] contains 28K videos with 203 action classes from 7 different domains: *Personal Care*, *Eating and Drinking*, *Household*, *Caring and Helping*, *Working*, *Socializing and Leisure* and *Sports and Exercises*.

Charades dataset [Sigurdsson et al. 2016] was collected with the idea of representing boring actions, that play an important role in daily human life. The authors crowdsourced the data collection, resulting in 9848 videos for 157 action classes, filmed by 267 partic-

ipants in their home environment. Charades-Ego [Sigurdsson et al. 2018] uses the same approach to the data collection, but provides paired first and third-person videos for joint learning of first and third-person representations. The dataset includes 4000 paired videos, involving 112 different subjects. Driven by the same motivation of learning boring daily actions, [Fouhey et al. 2018] proposed to mine them from lifestyle VLOGs, widely available on sites such as YouTube, resulting in 114K videos with 10.7K different participants.

Kinetics-400 [Carreira et al. 2017] is the dataset of 300K short clips, collected from YouTube, and features 400 different action classes. This dataset was later extended to Kinetics-700, that includes 650K videos and 700 classes [Carreira et al. 2019]. Another large-scale dataset, EPIC-KITCHENS [Damen et al. 2018] is currently the largest source of egocentric data, containing 40K crowdsourced videos, focused on cooking activities.

So far we have described the general purpose datasets, that play a key role in training and evaluating models for action recognition, action detection and other tasks of video understanding. Datasets, specifically focusing around instructional videos, are discussed in detail in Section 2.3.6.

2.2 Weakly-supervised learning from videos

In this section we provide an overview of weakly-supervised methods for action localization in the videos. We begin by describing existing approaches, where supervision is provided in the form of video-level labels in Section 2.2.1. In Section 2.2.2, we consider a more challenging scenario, where each video contains many action labels. This is relevant to understanding instructional videos, that typically show complex activities, composed of many different individual actions.

2.2.1 Action detection with video-level annotations

Weakly-supervised action detection addresses the problem of recognizing actions in videos, and predicting their temporal extents, while using supervision in the form of video-level

labels for training. This means, that at the training time, we know only what action occurs in the given video, but do not know when exactly it occurs. This problem is similar to weakly-supervised object detection in images [Jie et al. 2017; Kantorov et al. 2016; Singh et al. 2017; P. Tang et al. 2018], where the supervision is provided in the form of image-level labels, instead of bounding boxes.

Action detection from video-level supervision was first addressed by [Sun et al. 2015]. The authors proposed to train the model on images with action labels first, and, then, to transfer the learnt representations to the video domain. UntrimmedNets [Limin Wang et al. 2017] introduced two types of selection modules, which are used to select action proposals for each class: attention-based soft selection and hard selection, based on multiple instance learning (MIL). MIL approach to weakly-supervised localization was also used in a later work by [Paul et al. 2018]. Hide-and-Seek framework [Singh et al. 2017] applies to weakly supervised detection of both objects in images and actions in videos. The key idea of this framework is to randomly mask out parts of the input, thus forcing the network to look for the relevant parts in the remaining video. Autoloc, proposed by [Shou et al. 2018], introduced outer-inner-contrastive loss, that allows to train a boundary predictor. [Nguyen et al. 2018] combined a video classification loss with a sparsity loss, in order to extract a sparse subset of representative frames from the video, which is then used to generate action proposals. [Ma et al. 2020] considered a different type of weak supervision, where a single annotated frame is provided for each action in each video. The proposed approach is based on pseudo-label mining. First, the model is trained on the manually labeled frames. Then, the training set is expanded by taking into consideration the frames, which are sufficiently close in time to the ground truth frame for the given action, and such that the output of the model for the given frame is relatively high.

2.2.2 Action localization with action transcripts

In this section, we provide an overview of the existing methods for weakly-supervised localization in long untrimmed videos, that may contain multiple action instances of different classes. Unlike Section 2.2.1, the methods, described below address the problem, where each video may include a high number of different action labels. To address

this challenge, most of such methods rely on additional supervision. Examples of such supervision include approximate action boundaries and the temporal order of action appearance in the video.

Instructional videos, which are the main focus of this thesis, fall into this category of videos. However, instructional videos have their own specifics. For example, different instructional videos, that address the same task, share a common underlying structure. We can assume, in particular, that such videos share similar actions, and that the actions are performed in a similar order. Such assumptions can be explicitly incorporated in action localization methods. This is generally not the case for the type of videos, that we consider in this section. To make this distinction clearer, the methods for action localization, designed specifically for instructional videos, are described in Section 2.3.1, while the present section addresses the general case.

Discriminative clustering methods, such as DIFFRAC [Bach et al. 2007], are a popular choice for weakly-supervised action localization in videos. These methods consist in minimizing a discriminative loss with respect to both the parameters of the model and the label assignment. This is usually done by minimizing the loss iteratively, alternating between updating the parameters of the model and updating the label assignment. One benefit of the discriminative clustering is its flexibility. In particular, it allows to impose any constraints on the label assignment, as long as the linear optimization problem can be solved efficiently under these constraints. Weakly-supervised localization methods, based on the discriminative clustering, usually leverage available weak supervision in the form of constraints on the solution. In the domain of video understanding, this approach is used in works such as [Alayrac et al. 2016, 2017; Bojanowski et al. 2013, 2014, 2015; Duchenne et al. 2009; Miech et al. 2017; Seguin et al. 2016].

In particular, [Duchenne et al. 2009] tackled the problem of action localization in movies, by using movie scripts as a source of supervision. First, the actions are mined from the script, by using part of speech (POS) tagging and named entity recognition (NER) to find triplets of the form (person, verb, noun), (*e.g.* "Jane opens door"). Then, the authors align subtitles with the movie script in order to obtain an approximate temporal window for each entry in the script. The previously discovered action triplets are then localized within the video via discriminative clustering with the temporal constraints, derived

from the previously obtained temporal windows. [Bojanowski et al. 2013] proposed an approach, similar to [Duchenne et al. 2009] for joint modeling of actors and actions in the movies.

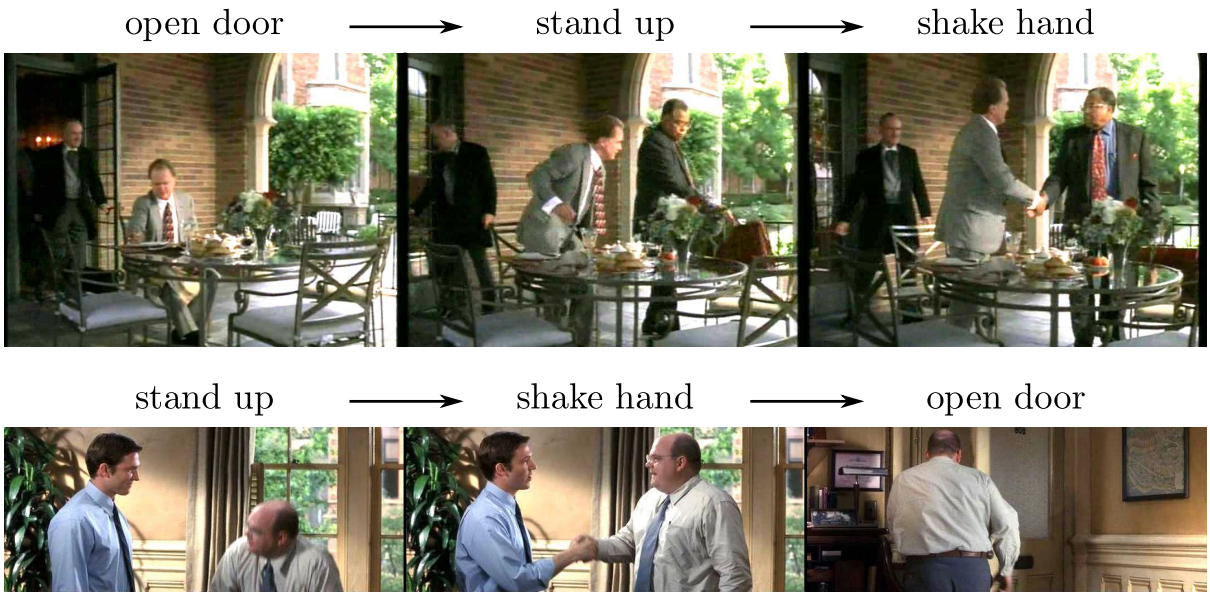


Figure 2.1: Examples of videos with annotated action transcripts. Such transcripts provide information on which action labels are present in the video, and what is their order of appearance. This illustration is taken from [Bojanowski et al. 2014].

[Bojanowski et al. 2014] considered another type of weak supervision for action localization in the videos, the temporal ordering constraints, illustrated on Figure 2.1. Such supervision is easier to obtain, than the exact temporal action segments. The problem is solved via alternating optimization, by updating the weights of the model, while freezing the current label assignment, and updating the label assignment with fixed model weights. The label assignment is updated by solving a linear optimization problem under the ordering constraints, which can be done efficiently via dynamic programming. [Bojanowski et al. 2015] extends this approach to the case of two sequences of continuous vector variables, instead of a sequence of discrete labels and a video. This is applied to the problem of aligning video with text. Weak supervision from ordered action transcripts, first introduced by [Bojanowski et al. 2014], became increasingly popular, and was later used, in particular, by [Alayrac et al. 2016; Chang et al. 2019; Ding et al. 2018; Huang et al. 2016; Richard et al. 2017, 2018b].

An approach to weakly-supervised instance-level segmentation, based on discriminative clustering was proposed by [Seguin et al. 2016]. This method relies on supervision in the form of tracked object bounding boxes, instead of pixel-level annotations. This approach is formalized as minimizing a squared loss under a set of linear constraints. In particular, the authors impose a lower bound on the number of pixels within a bounding box, assigned to the corresponding class. They, furthermore, impose an upper bound on the distance between a pixel, assigned to the given class and the corresponding bounding box.

[Bojanowski et al. 2013, 2014, 2015; Seguin et al. 2016] formulate the learning task as a quadratic minimization problem and solve it via Frank-Wolfe algorithm [Frank et al. 1956; Jaggi 2013]. Frank-Wolfe algorithm allows to minimize a convex function over a convex domain, by optimizing a series of linear functions over the same domain. One drawback of this method is that it requires to compute and store a matrix of size $N \times N$, where N is the number of training examples, making it prohibitive in the case of sufficiently large training set. [Miech et al. 2017] proposed a more scalable approach to joint learning of actors and actions, compared to [Bojanowski et al. 2013], by decomposing the set of constraints into a Cartesian product of constraints for each individual video, and adapting Block-Coordinate Frank-Wolfe algorithm [Lacoste-Julien et al. 2013] to find the optimal solution.

A different approach to the weakly supervised action localization was proposed by [Huang et al. 2016]. They adapt Connectionnist Temporal Classification (CTC) framework, originally introduced in the context of speech recognition, to segment the actions within the videos under the temporal ordering constraints, similar to [Bojanowski et al. 2014]. This framework allows to estimate the probability of a given label ordering, by summing up the probabilities of all possible label assignments, that satisfy this order. At training time, the objective is to minimize the negative log-likelihood of the correct ordering. The gradient of this log-likelihood can be computed via dynamic programming, allowing to efficiently optimize the objective.

[Richard et al. 2017] combined an RNN that produces frame-wise predictions with a Hidden Markov Model (HMM), that models the sequence of actions. Similar to [Bojanowski et al. 2014; Huang et al. 2016], this method assumes a weak supervision in the form of an ordered label sequence. The proposed learning approach consists in alternating between

updating the RNN, given current label assignment, and updating the label assignment with the HMM. HMM, in particular, allows to impose the ordering constraints on the solution, by putting certain transition probabilities to zero. [Richard et al. 2018b] uses a similar approach, but replaces the HMM with a combination of the network outputs with an action length model. The length of each action is modeled with Poisson distribution, and the parameter λ of this distribution is updated after each iteration. [Li et al. 2019] also uses a combination of an RNN with an HMM on top of its outputs, and introduces a novel constrained discriminative forward loss.

A similar two-step approach, where the model and the current assignment are progressively updated, was considered by [Ding et al. 2018], who proposed to train a CNN, namely Temporal Convolutional Feature Pyramid Network (TCFPN), under weak supervision in the form of ordered action transcripts. At each iteration, the current segmentation is updated via a procedure, called iterative soft boundary assignment, where a boundary between adjacent actions within the video is shifted, if the estimated probability of the first action at the border is significantly larger than the probability of the second action.

[Chang et al. 2019] introduced a method for aligning the video with its ordered transcript, titled discriminative differentiable dynamic time warping (D3TW). The problem is formulated as a dynamic time warping problem, where the goal is to learn the distance function, that assigns a cost of aligning an action with a frame. This is done by minimizing a negative log-likelihood function, given a video and the most likely label assignment (*i.e.* the label assignment, that minimizes the overall alignment cost). Instead of computing the most likely assignment by solving a minimization problem, for example, by dynamic programming, the authors propose a continuous relaxation of the minimum operator in order to make it differentiable. The whole training procedure is thus reduced to the gradient descent, instead of a two-step approach, similar to [Bojanowski et al. 2014] and [Richard et al. 2017].

[Richard et al. 2018a] addresses the task of action segmentation, where supervision is provided in the form of an unordered list of actions per video, which is a weaker source of supervision, compared to [Bojanowski et al. 2014; Huang et al. 2016; Richard et al. 2017] (see Figure 2.2). The authors take a generative approach and decompose the probability of a given segmentation into three factors, including the probability of a given ordering of

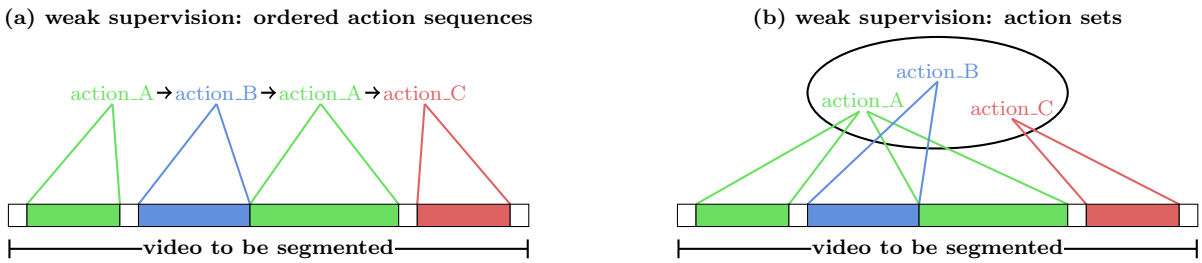


Figure 2.2: Action set supervision, used in [Richard et al. 2018a] and [Li et al. 2020], compared to the ordered transcript supervision, used in [Bojanowski et al. 2014; Huang et al. 2016; Richard et al. 2017]. This illustration is taken from [Richard et al. 2018a].

actions, the probability of action length, similar to [Richard et al. 2018b], and the frame probability, given the action. Each of these components is modeled separately.

The problem of action segmentation from unordered action sets was further investigated by [Li et al. 2020]. Unlike [Richard et al. 2018a], [Li et al. 2020] proposed to jointly train a network, that produces frame-wise action predictions, and an HMM, that models action lengths and co-occurrences. To this end, the authors propose set constrained Viterbi algorithm, that allows to infer the optimal segmentation under the set constraints (*i.e.* every action from the set must appear in the segmentation). The training is performed in a similar way, as in [Richard et al. 2017], in that the current model is used to produce a segmentation, that is later used as pseudo-ground truth to update the model weights.

2.3 Instructional videos

In this section, we describe different problems, tied to the automatic understanding of instructional videos, methods, proposed for their solution, as well as existing datasets for training and evaluation of computer vision models. We start with an overview of the problems and methods for instructional video understanding in Sections 2.3.1-2.3.5, followed by an overview of instructional video datasets in Section 2.3.6.

2.3.1 Supervised step localization

Instructional videos are inherently action-oriented. Actions, that compose a task, often referred to as *sub-activities* or *steps*, play an essential role in such videos. Naturally, recognition and localization of the steps is one of the most important and extensively studied problems in the domain of instructional videos. Earlier works focus on recognition and localization of steps in the supervised setup.

[Kuehne et al. 2014] draws inspiration from speech recognition, by proposing to model the steps in the videos with an HMM. This allows to model the dependencies between steps (*i.e.* , which step is more likely to follow the current step). [Richard et al. 2016] proposed an N-gram language model for step sequences, combined with linear classifiers of steps in the video and a Poisson distribution to model the length of each step. These works rely on human annotations in the form of the temporal extents of steps in the videos. The difficulty and the cost of obtaining such annotations motivates the research towards weakly-supervised and unsupervised step localization.

2.3.2 Weakly-supervised and unsupervised step localization

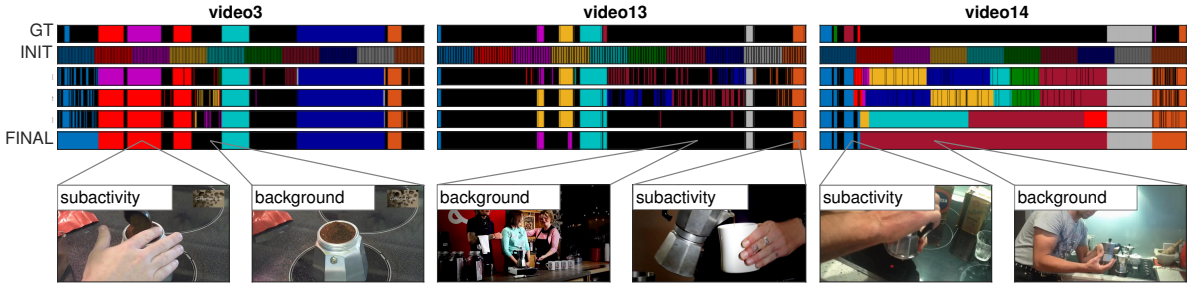


Figure 2.3: Results of step segmentation from [F. Sener et al. 2018] on three "making coffee" videos. The model predicts step labels, including the background label, for all frames of the video. Background recognition is complicated by the fact that the background often includes unimportant manipulations with the key objects, as shown in the third example. This illustration is taken from [F. Sener et al. 2018]

Many methods, designed for action localization in long untrimmed videos with multiple action instances, such as [Huang et al. 2016; Kuehne et al. 2017; Richard et al. 2017, 2018a], discussed in Section 2.2.2, were applied to step localization in instructional videos. These methods, however, do not take the task structure into account. Focusing more

specifically on the instructional videos, [F. Sener et al. 2018] proposed an unsupervised method for learning and localizing the steps, using Generalized Mallows Model to model the sequence of steps. This model assigns probabilities to all possible orderings of the steps, with higher probabilities assigned to orderings, which are closer to a "canonical" order. The model assigns a label (either one of the steps, or the background) to each frame in the video, as shown in Figure 2.3.

[Kukleva et al. 2019] introduced a two-step approach to unsupervised learning of steps. The first step is to learn a continuous temporal embedding of frame features, by predicting the relative time of the frame occurrence in the video. Frame features are then clustered in this latent embedding space, in order to recover a set of steps and localize them in the videos. The main idea behind this approach is that different steps generally occur in different moments in the videos (*e.g.* , "pour coffee" usually appears after "grind coffee"). Therefore, learning to predict time of appearance of each frame may help to discriminate different steps.

The approach, proposed by [Elhamifar et al. 2019], combines an HMM to model step sequences within videos, with subset selection to find a common subset of steps for all videos of a given task. Similarly, [Elhamifar et al. 2020] applies subset selection to the problem of unsupervised step localization, but extends it to the case of multiple tasks, allowing for a task-agnostic step localization. This is done by incorporating a task prediction module and task-conditioned video representation into the model.

[Naing et al. 2020] addressed the problem of learning the tasks from incomplete video demonstrations, where each video in the training set may omit one or multiple steps of the task. In practice, web instructional videos often omit certain steps, for example, as a result of video editing, aimed at reducing the duration of the video. The paper proposes a solution to this problem, based on optimizing a subset selection criterion.

In Chapter 4 we present a method for unsupervised actionness estimation in instructional videos, that allow to localize the boundaries of the steps. This method can be coupled with step classification in order to solve the problem of step localization in the presence of the background.

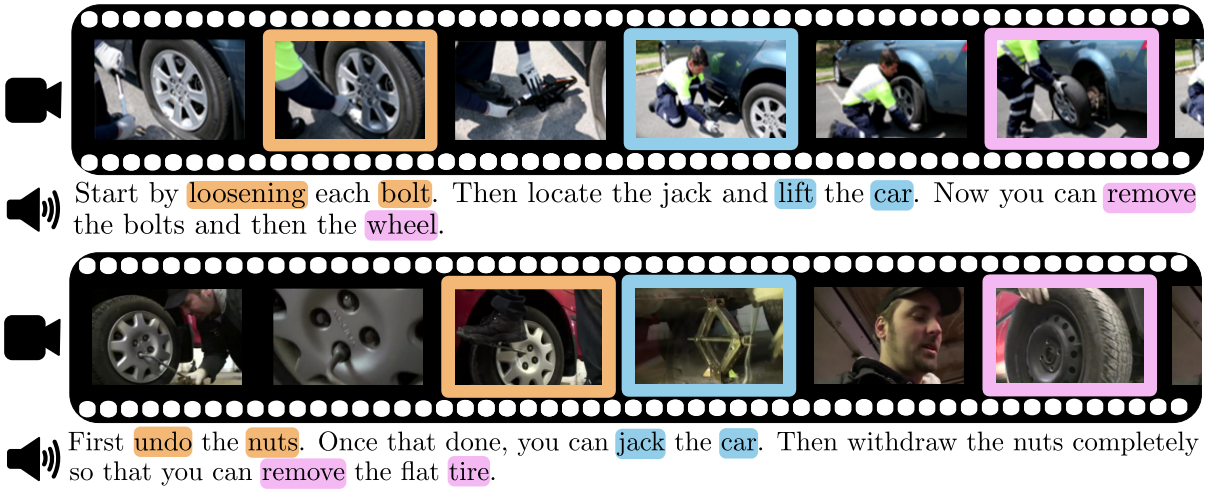


Figure 2.4: Narration in instructional videos can be used to guide the step localization. Frames, containing the steps and the corresponding parts of narration are highlighted with the same color. This illustration is taken from [Alayrac et al. 2016].

2.3.3 Step localization from text and video

Narration in instructional videos was extensively used to learn and localize the steps. [Yu et al. 2014] first proposed to localize actions within the instructional videos with the help of narration, by searching for direct object (*dobj*) relations (typically, verb-noun pairs) within the subtitles. This work, however doesn't aim at step localization, and only uses narrations to mine action examples from videos.

[Malmaud et al. 2015] tackled the task of step localization in the cooking videos, by, first, parsing clean manual text recipes to recover a list of steps, then, aligning these steps with subtitles with the help of an HMM, and, finally, refining the step localization in the video with the help of object detectors. This approach, however, requires clean recipes, pre-defined list of action verbs and pre-trained object detectors, which makes it hard to extend beyond cooking videos.

Less restrictive approaches were proposed by [O. Sener et al. 2015] and [Alayrac et al. 2016]. In particular, [O. Sener et al. 2015] represents each frame of the video as a bag of visual and linguistic "atoms", where the visual atoms are extracted from videos via clustering and linguistic atoms are represented by the words with the highest TF-IDF scores in subtitles. They, then, model instructional videos with a Beta Process Hidden

Markov Model, assuming that each step of the task has different probability of generating each visual and linguistic atom. Fitting this model allows to discover the steps and to localize them within each video. This method doesn't have any restrictions and can be applied to any instructional video. One of its limitations, however, lies in the fact that it requires a discrete bag of words representation of input video, and doesn't allow the use of more advanced continuous video and language representations.

[Alayrac et al. 2016] proposed a two-step approach to discover and localize the steps in narrated videos (see Figure 2.4). First, a common sequence of steps is recovered from subtitles. This is done by extracting a sequence of dobj relations from the subtitles of each video, and then apply sequence alignment in order to extract a common sub-sequence. Second, these steps are localized in the video via discriminative clustering, where the assignment of steps and the step classifiers are updated together in order to minimize the loss function. In order to regularize this problem, the authors use the sequence of steps, previously recovered from the subtitles, to impose two types of constraints. First, the order of appearance of steps in the video should be the same as in the recovered step sequence. This is similar to [Bojanowski et al. 2014], the main difference being that the steps must have the same order of appearance in all videos. Second type of constrains makes use of the time-stamps, available for the subtitles, and is based on the assumption, that the narration and the video are roughly aligned. This means that the step must appear in the video within a temporal window centered around the time, when it appears in the narration. The authors use DIFFRAC to solve this constraint discriminative clustering problem, and simultaneously learn to recognize the visual appearance of the steps and to localize them within the video. This approach has several limitations. First, it doesn't allow for variations in the step order across the videos. Second, successfully discovering steps from language with the help of dobj relations requires high quality of subtitles.

The same two types of temporal constraints from narration were used by [Fried et al. 2020], who proposed a step learning method, based on a Hidden semi-Markov Model, instead of discriminative clustering.

Contrary to [Alayrac et al. 2016], who, first, recover a sequence from narration and then localize them in the video, and to [O. Sener et al. 2015], who simply concatenate

both types of input, [Shen et al. 2021] proposed to discover the steps within narration and within the video jointly, while taking into account possible inconsistencies between two modalities. The authors, first, discover a sequence of step "prototypes" within each modality via clustering, and then align both sequences to recover a sub-sequence of steps, that best align between the narration and the video. This is done via differentiable weak sequence alignment, proposed in the paper.

Our step localization method, described in Chapter 3 uses similar temporal constraints, as in [Alayrac et al. 2016]. However, unlike [Alayrac et al. 2016], we do not aim to discover the steps within the narration, and, instead, rely on manually provided script for each task, which allows to successfully recover the steps from text, even when the quality of the subtitles is relatively low (in particular, if the subtitles are generated with automatic speech recognition (ASR)), at the cost of requiring manual annotations on the task level.

2.3.4 Joint video and language modeling

Figure 2.5: Narration in the instructional video is roughly aligned with the video content. This can be used to train models for joint text and video understanding. This illustration is taken from [Miech et al. 2020].

In the previous section, we have considered methods, that use language as guidance to localize steps in instructional videos. In this section, we consider other works that make use of language and visual data in the instructional videos.

[Miech et al. 2019] proposed to use the correspondences between the narration and the visual content in instructional videos, in order to learn joint text and video embedding. Typically, joint embedding is trained on a set of clip-caption pairs, where the captions, describing the visual content, are provided by annotators. However, obtaining such manual captions on large scale is expensive. The authors, instead, used subtitles, produced by ASR, as a source of free, albeit noisy, captions, and trained a joint embedding on top of

pre-trained video representations and word vectors. This approach was further improved in [Miech et al. 2020], by learning the entire embedding from scratch in an end-to-end fashion, instead of relying on pre-trained features. Furthermore, the approach, proposed by [Miech et al. 2020], takes into account the weak nature of the text supervision, by applying multiple-instance learning approach to the problem. MIL loss allows to take into account the possible misalignment between paired clips and subtitles, as shown in Figure 2.5. The authors demonstrate that the obtained embedding can be successfully applied to the problem of text retrieval in the long untrimmed videos. Moreover, they show that the video representation, trained this way, can be used in various downstream tasks such as action recognition.

Similarly, [Sun et al. 2019] proposed to train a joint text-video model, called VideoBERT, on the paired clips and captions, obtained from narrated cooking videos. The model is based on BERT, an encoder-decoder language model, and allows, in particular, to generate captions, given video clips at test time. A similar approach was taken by [Zhu et al. 2020], who trained the proposed ActBERT model on HowTo100M clip-caption pairs. Unlike previous approaches, ActBERT explicitly models actions, by extracting all verbs in the subtitles and using them as labels to train a 3D convolutional network on the corresponding clips. This network is then used to obtain a global action feature for the input clips, which is then provided to ActBERT as a part of the input.

[Doughty et al. 2020] investigated the role of adverbs and how they can affect the visual appearance of actions in instructional videos. Three pairs of adverbs were studied: *quickly/slowly*, *finely/coarsely*, and *partially/completely*. The authors propose to learn a joint embedding of video clips and actions in a common embedding space and model each adverb as a linear transformation in the action embedding space. These transformations, called action modifiers are learnt on all co-occurrences of the verbs and adverbs in the subtitles and the corresponding clips. The authors evaluate the model by applying to video-to-adverb retrieval and adverb-to-video retrieval.

[Huang et al. 2017] considered the problem of unsupervised reference resolution in cooking videos. Given a video and the corresponding transcript, the goal is to construct a reference graph over the transcript, by connecting entities (such as “dressing”) with the corresponding actions (such as “mix yogurt with black pepper”). While this is a purely

linguistic problem, the idea behind visual reference resolution comes from the observation, that the visual appearance may help to identify such references within the language. The authors tackle the problem via an action graph, that connects the visual and linguistic cues.

Following this work, [Huang et al. 2018] proposed an approach to the reference-aware visual grounding in instructional videos. Given a video with an aligned caption, and a set of bounding boxes within the video, the goal is to associate each entity within the caption with the corresponding bounding box. Since previously mentioned entities are often referred by the pronouns in the captions (*e.g. first, mix the greens*, and *next, take it*), one challenge is to understand what each pronoun is referring to. The authors demonstrate, in particular, that solving such linguistic ambiguities improves the results of grounding text entities in the video, and vice versa.

In Chapter 5 we address the problem of 3D language grounding in instructional videos. While this is a novel problem, we draw inspiration from the previous works, by using automatically generated subtitles to train our model.

2.3.5 Related problems

[Alayrac et al. 2017] introduced a new task of joint discovery of object states and actions, and proposed a method for its solution, based on the discriminative clustering. The motivation behind the joint modeling of states and actions is based on the observation, that the change in the object state is usually a result of an action, and observing the same object in different states implies that a state-changing action occurred in-between, therefore serving as a clue for action localization, and vice versa.

Action anticipation was also investigated in the domain of instructional videos. Unlike action classification, where the task is to predict the label of observed action, and action localization, where the task is to localize actions within the video, in the case of action anticipation, the goal is to predict what will happen after a certain point in time, given an incomplete video demonstration.

[Farha et al. 2018] proposed a method for anticipating future actions in the cooking

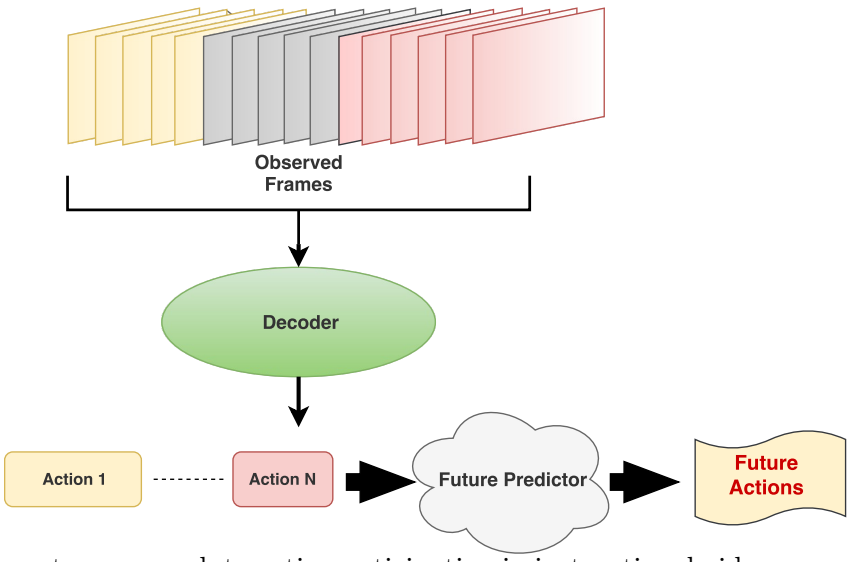


Figure 2.6: Two-step approach to action anticipation in instructional videos proposed by [Farha et al. 2018]. An observed part of the video is, first, translated into a sequence of action labels, from which future actions are, then, deduced. This illustration is taken from [Farha et al. 2018].

videos within a 5 minutes interval. This is done by, first, recognizing steps of the task within the observed part of the video, and, then, deducing the next steps, as illustrated in Figure 2.6. The approach, proposed by [Farha et al. 2017], instead allows for learning to anticipate the future steps directly from the data in an end-to-end fashion, and doesn't require to first infer the steps in the observed part of the video.

Contrary to the previous works, which require multiple video examples of each task at training time, [F. Sener et al. 2019] and [F. Sener et al. 2021] tackled the problem of action anticipation in the cooking videos in a zero-shot scenario, where the model is tasked with making predictions for a previously unseen dish.

Another problem related to instructional videos is the skill determination. It consists in assessing the correctness of task execution in a video demonstration. This task was first addressed in such works as [Q. Zhang et al. 2011; Zia et al. 2015, 2018, 2016] in the context of surgical tasks. The general case was, first, considered by [Doughty et al. 2018], who evaluated the skill determination in four different domains, including surgery, dough rolling, drawing and chopstick using. The proposed approach is based on a siamese network, with a TSN backbone, that takes a fixed amount of frames from the video as an input. At training time, the model is fed two videos with the first video having higher

skill rank than the second, with the objective to minimize the ranking loss for a given pair of videos. This approach was improved by [Doughty et al. 2019], who proposed an attention-based representation of each video, instead of sampling random frames.

[Chang et al. 2020] considered the problem of procedure planning, where, given a start visual observation (e.g. cracked eggs in a bowl), the goal is to produce a sequence of actions, leading towards a desired visual goal (e.g. an omelette). To this end, the authors propose to learn an embedding of observed states and actions into a space of semantic states and semantic actions respectively. Next, they learn forward dynamics, that produce the next semantic state, given the current semantic state and action. Finally, the procedure planning is performed by recovering semantic states from the start and the goal observations, and using forward dynamics, reconstruct a path from the start to goal state.

2.3.6 Instructional video datasets

In the previous sections we described various problems, related to the instructional videos, as well as the methods proposed for their solution. In this Section, we describe the existing publicly available datasets of instructional videos, that are used to train and evaluate computer vision models. To make a clear distinction from other video datasets, described in Section 2.1.2, below we only consider the datasets of video demonstrations of complex human activities with a well-defined goal, which requires to perform a sequence of actions to achieve. Furthermore, we only consider the datasets, which provide multiple video examples for each task, therefore allowing understanding of the task structure.

One of the first attempts at studying instructional videos was done by [Rohrbach et al. 2012], who collected the MPII dataset of 44 cooking videos with 14 different dishes, performed by 12 different persons in the same environment. The videos were provided with annotated action segments, with a total of 65 action classes, as well as annotated human poses. However, this dataset considers complex human tasks simply as a source of actions, and is more oriented towards action recognition and detection, as opposed to the understanding of the task. In particular, the dishes used for data collection are vaguely defined. For example, the “soup” dish includes different recipes of the soup with

Dataset	Num. vids	Annot. vids	Num. tasks	Avail. annot.	Comments
MPII [Rohrbach et al. 2012]	44	44	14	Windows	Cooking, same kitchen
50salads [Stein et al. 2013]	54	54	1	Windows	Cooking, single task
Breakfast [Kuehne et al. 2014]	2K	2K	10	Windows	Cooking
[O. Sener et al. 2015]	1.2K	85	17	Windows	
What's Cookin'? [Malmaud et al. 2015]	180K	-	-	Recipes	Cooking, no steps no task labels
[Alayrac et al. 2016]	150	150	5	Windows	
YouCook2 [L. Zhou et al. 2018a]	2K	2K	89	Windows	Cooking, Manual captions
How2 [Sanabria et al. 2018]	79K	-	-	Aligned subtitles	No steps no task labels
COIN [Y. Tang et al. 2019]	12K	12K	180	Windows	
ProceL [Elhamifar et al. 2019]	720	720	12	Windows	
HowTo100M [Miech et al. 2019]	1.22M	-	23K	-	No steps
CrossTask [Zhukov et al. 2019]	4.7K	2.7K	83	Windows	

Table 2.1: **Comparison of instructional video datasets.** Third column indicates the number of videos with annotated step windows. Forth column indicates the number of tasks if task labels are provided.

different ingredients, including a packet soup.

[Stein et al. 2013] introduced 50salads, a dataset, that includes 54 videos of 27 person preparing the same dish, a mixed salad, with a total of 17 different step labels, such as "add oil", "cut lettuce" and "mix ingredients". This dataset, however, is limited by the fact that it only consists of a single task.

[Das et al. 2013] proposed YouCook dataset, consisting of 88 cooking videos. Unlike the datasets, described above, this dataset was collected from YouTube, and, thus, provides a higher variety of scenes and people. This dataset was originally designed for the task of video description. [L. Zhou et al. 2018a] proposed the YouCook2 dataset, that is more suited for the fine-grained understanding of cooking activities. It contains 2000 videos for 89 different cooking recipes, annotated with temporal action segments. All segments are provided with manual captions, such as "grill the tomatoes in a pan".

Breakfast dataset, proposed by [Kuehne et al. 2014] also focuses on the cooking videos, providing a total of 1989 videos for 10 different tasks. The videos provide recordings of actions performed by 52 subjects in 18 different kitchens. Hence, the variation of environments across videos is relatively limited. The annotations include action segments for coarse action labels (*e.g.* "pour milk"), as well as for finer-grained action labels (*e.g.* "grab milk", "twist cap" and "open cap").

[Malmaud et al. 2015] introduced a dataset of 180K narrated cooking videos with provided English speech transcripts, collected from YouTube, by searching with "Cooking" and "Recipe" tags. All videos are provided with text recipes. However, the videos are not grouped by the task and are not provided with annotated action segments.

[O. Sener et al. 2015] collected a dataset of 1.2K videos for 17 different tasks. In order to collect the data, the authors, first, searched for the most popular tasks, related to the physical world, on WikiHow. The videos were, then, collected by querying the name of each task on YouTube and collecting the top 100 videos per task. Since the output of the YouTube search may contain outliers. The authors propose an unsupervised approach to the outlier detection, based on clustering video descriptions. Discarding all videos outside the dominant cluster yields at least 50 videos per task, which, however, may

still include some outliers. The authors provide annotations for 5 videos per task. This dataset stands out from the previous ones, by including tasks from different domains, not limited to cooking. Such non-cooking tasks include *Unclog a Bathtube Drain* and *Tie a Tie*. Most of the tasks (13 out of 17), however, are cooking tasks, which reflects the abundance of such tasks and their relative popularity on WikiHow.

A similar dataset, proposed by [Alayrac et al. 2016] includes 5 tasks, with 30 videos per task, collected from YouTube. The included tasks are *Making a coffee*, *Changing car tire*, *Performing cardiopulmonary resuscitation (CPR)*, *Jump-starting a car* and *Repotting a plant*. While significantly smaller than [O. Sener et al. 2015], this dataset has several advantages. First, the videos are selected manually, in order to ensure, that there are no outliers. Second, all 150 videos are annotated with the step segments. Finally, all videos are provided with accurate manual closed-captions.

The dataset of [Alayrac et al. 2016] was extended by [Elhamifar et al. 2019], who proposed ProceL dataset. It includes 720 videos and 12 tasks, with 60 videos per task. 5 tasks are the same as in [Alayrac et al. 2016]. However, the corresponding sets of videos were extended to include 60 videos for each task. The newly introduced tasks are *Set up Chromecast*, *Assemble clarinet*, *Replace iPhone battery*, *Tie a tie (Windsor knot)*, *Replace toilet*, *Make peanut butter jelly sandwich* and *Make smoked salmon sandwich*. All videos are provided with step segment annotations, and with subtitles, generated by ASR.

Sanabria et al. 2018 proposed How2 dataset with a focus on language understanding. It consists of 79K instructional videos, collected from YouTube using keywords. Similarly to [O. Sener et al. 2015] and [Alayrac et al. 2016], it covers different topics, such as *Cooking*, *Gardening* and *Health*, but doesn't provide task labels for the videos. However, unlike the aforementioned datasets, it doesn't provide task labels for the videos, and the annotated step segments. The videos are provided with English subtitles, aligned with the videos on the word level, as well as their translations in Portuguese.

Closer to the main focus of this thesis, COIN dataset [Y. Tang et al. 2019] was designed specifically for the task of step localization. It contains 12K videos and 180 tasks, grouped by 12 different domains, such as *Vehicles*, *Housework* and *Dishes* (see Figure 2.7). The annotated step segments are provided for all videos.

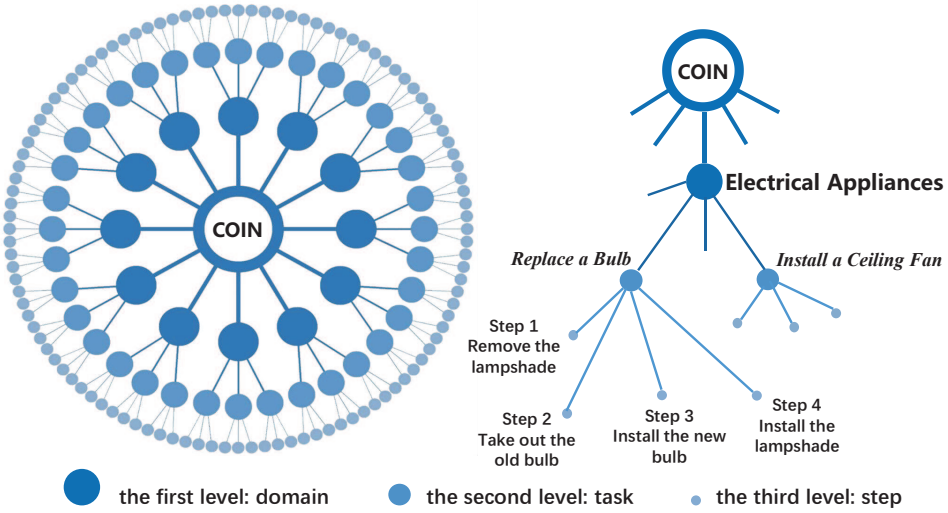


Figure 2.7: COIN includes instructional videos for 180 tasks, divided into 12 categories, with annotated step segments for each video. This illustration is taken from [Y. Tang et al. 2019].

HowTo100M dataset [Miech et al. 2019], designed for joint video and text modeling, was build with the idea to cover all physical tasks from WikiHow. To this end, the authors, first, collected all 120K tasks from WikiHow, and manually selected 23.6K visual tasks, discarding non-physical tasks, such as *How to make friends*. Videos were collected from YouTube, by querying each task in the YouTube search, taking top 200 videos from YouTube output, and removing videos without subtitles (either manual, or produced by ASR). This results in 1.22M unique videos. HowTo100M inherits the hierarchy of tasks from WikiHow, as shown in Figure 2.8. The highest level of this hierarchy is represented by 12 different domains, such as *Food and Entertainment*, *Home and Garden* and *Hobbies and Crafts*. Lower levels of this hierarchy are progressively more fine-grained, with the concrete tasks at the bottom level (e.g. *Food and Entertainment* $\rightarrow$ *Breakfast* $\rightarrow$ *Pancakes* $\rightarrow$ *How to Make Strawberry Pancakes*). This dataset, however, doesn't provide ground truth step annotations.

One of the contributions of this thesis is the CrossTask dataset, described in Chapter 3. This dataset contains 4.7K videos and 83 task. It is designed to overcome the limited amount of YouTube videos, available for each task, by drawing from similar tasks. For this reason, the tasks are divided into two parts. The first part consists of 18 *primary* tasks. These tasks are provided with a total of 2750 videos, filtered manually to elim-

inate outliers, and annotated with temporal step segments. The remaining videos are distributed across 65 *related* tasks with 30 videos per task. These videos are not filtered manually, and are, instead, collected automatically, by taking top 30 videos from YouTube search output. Table 2.1 provides a brief summary of the datasets, described above, in comparison with CrossTask.

Figure 2.8: First two levels of How100M hierarchy. For each category, the total number of tasks is given in blue, and the total amount of videos is in red. This illustration is taken from [Miech et al. 2019]

Chapter 3

Cross-task weakly supervised learning from instructional videos

In this chapter, we address the problem of localizing the steps of complex tasks from narrated instructional videos, where the annotations are provided in the form of ordered list of steps for each task. In order to benefit from the variety of available tasks, we propose a method that allows sharing the information across tasks on a sub-step level. We achieve this by representing each step of each task by a bag of *components*. For example, "cut lemon" can be represented by two components "cut" and "lemon". These components are shared across all tasks during training, allowing the model to leverage the visual similarity between different steps, such as "cut lemon" and "cut tomato". Our method allows to mitigate the data limitations for a single task by drawing from other tasks that may include visually similar concepts.

To demonstrate the benefits of our approach we collect a new dataset of 4.7K narrated instructional videos for 83 different tasks. We demonstrate gradual improvements by training without sharing information across tasks, with sharing of step labels and finally with sharing of step components. We show that our model consistently outperforms existing weakly-supervised action localization methods. Furthermore, we show that our model can recognize previously unseen steps in novel tasks, if they include previously observed components.

3.1 Introduction

Suppose you buy a fancy new coffee machine and you would like to make a latte. How might you do this? After skimming the instructions, you may start watching instructional videos on YouTube to figure out what each step entails: how to press the coffee, steam the milk, and so on. In the process, you would obtain a good visual model of what each step, and thus the entire task, looks like. Moreover, you could use parts of this visual model of making lattes to help understand videos of a new task, e.g., making filter coffee, since various nouns and verbs are shared. The goal of this chapter is to build automated systems that can similarly learn visual models from instructional videos and in particular, make use of shared information across tasks (e.g., making lattes and making filter coffee).

The conventional approach for building visual models of how to do things [Carreira et al. 2017; Simonyan et al. 2014; H. Wang et al. 2013] is to first annotate each step of each task in time and then train a supervised classifier for each. Obtaining strong supervision in the form of temporal step annotations is time-consuming, unscalable and, as demonstrated by humans' ability to learn from demonstrations, unnecessary. Ideally, the method should be weakly supervised (i.e., like [Alayrac et al. 2016; Huang et al. 2016; Kuehne et al. 2017; O. Sener et al. 2015]) and jointly learn *when* steps occur and *what* they look like. Unfortunately, any weakly supervised approach faces two large challenges. Temporally localizing steps in the input videos for each task is hard as there is a combinatorial set of options for the step locations; and, even if the steps were localized, each visual model learns from limited data and may work poorly.

We show how to overcome these challenges by sharing across tasks and using weaker and naturally occurring forms of supervision. The related tasks let us learn better visual models by exploiting commonality across steps as illustrated in Figure 3.1. For example, while learning about *pour water* in *making latte*, the model for *pour* also depends on *pour milk* in *making pancakes* and the model for *water* also depends on *put vegetables in water* in *making bread and butter pickles*. We assume an ordered list of steps is given per task and that the videos are instructional (i.e., have a natural language narration describing what is being done). As it is often the case in weakly supervised video learning [Alayrac et al. 2017; Huang et al. 2016; O. Sener et al. 2015], these assumptions constrain the

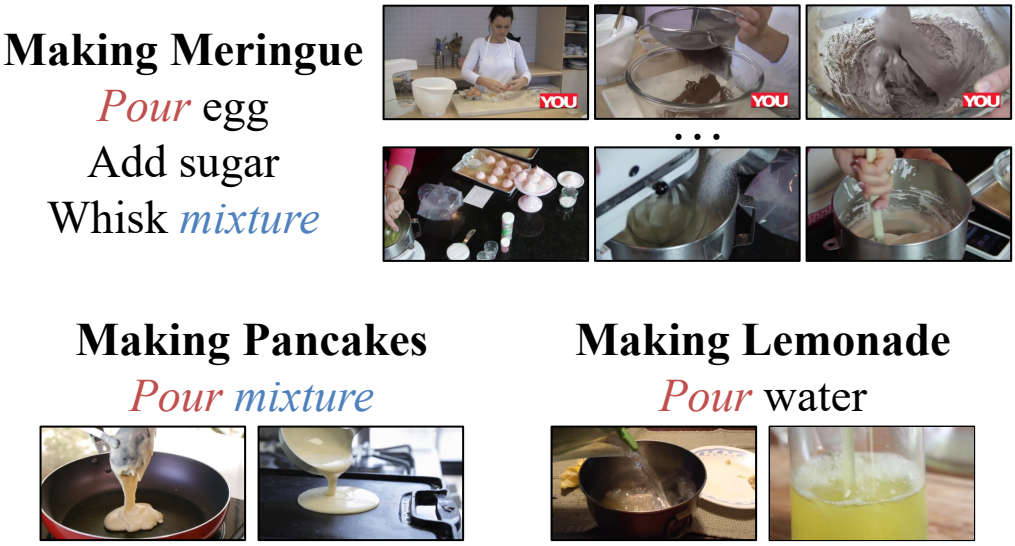


Figure 3.1: Our method begins with a collection of tasks, each consisting of an ordered list of steps and a set of instructional videos from YouTube. It automatically discovers both where the steps occur and what they look like. To do this, it uses the order, narration and commonalities in appearance across tasks (e.g., the appearance of *pour* in both *making pancakes* and *making meringue*).

search for when steps occur, helping tackle a combinatorial search space.

We formalize these intuitions in a framework, described in Section 3.4, that enables compositional sharing across tasks together with temporal constraints for weakly supervised learning. Rather than learning each step as a monolithic weakly-supervised classifier, our formulation learns a component model that represents the model for each step as the combination of models of its components, or the words in each step (e.g., *pour* in *pour water*). This empirically improves learning performance and these component models can be recombined in new ways to parse videos for tasks for which it was not trained, simply by virtue of their representation. This component model, however, prevents the direct application of techniques previously used for weakly supervised learning in similar settings (e.g., DIFFRAC [Bach et al. 2007] in [Alayrac et al. 2017]); we therefore introduce a new and more general formulation that can handle more arbitrary objectives.

Existing instructional video datasets do not permit the systematic study of this sharing. We gather a new dataset, CrossTask, which we introduce in Section 3.5. This dataset

consists of ∼4.7K instructional videos for 83 different tasks, covering 374 hours of footage. We use this dataset to compare our proposed approach with a number of alternatives in experiments described in Section 3.6. Our experiments aim to assess the following three questions: how well does the system learn in a standard weakly supervised setup; can it exploit related tasks to improve performance; and how well can it parse previously unseen tasks.

The contributions described in this chapter include: **(1)** A component model that shares information between steps for weakly supervised learning from instructional videos; **(2)** A weakly supervised learning framework that can handle such a model together with constraints incorporating different forms of weak supervision; and **(3)** A new dataset that is larger and more diverse than past efforts, which we use to empirically validate the first two contributions. We make our dataset and our code publically available[1].

3.2 Related work

Learning the visual appearance of steps of a task from instructional videos is a form of action recognition. Most work in this area, e.g., [Carreira et al. 2017; Simonyan et al. 2014; H. Wang et al. 2013], uses strong supervision in the form of direct labels, including a lot of work that focuses on similar objectives [Damen et al. 2018; Fang et al. 2018; Fouhey et al. 2018]. We build our feature representations on top of advances in this area [Carreira et al. 2017], but our proposed method does not depend on having lots of annotated data for our problem.

We are not the first to try to learn with weak supervision in videos and our work bears resemblances to past efforts. For instance, we make use of ordering constraints to obtain supervision, as was done in [Bojanowski et al. 2014, 2015; Huang et al. 2016; Kuehne et al. 2017; Richard et al. 2017]. The aim of our work is perhaps closest to [Alayrac et al. 2016; Malmaud et al. 2015; O. Sener et al. 2015] as they also use narrations in the context of instructional videos. Among a number of distinctions with each individual work, one significant novelty of our work is the compositional model used, where instead of learning a monolithic model independently per-step as done in [Alayrac et al. 2016;

[1] https://github.com/DmZhukov/CrossTask

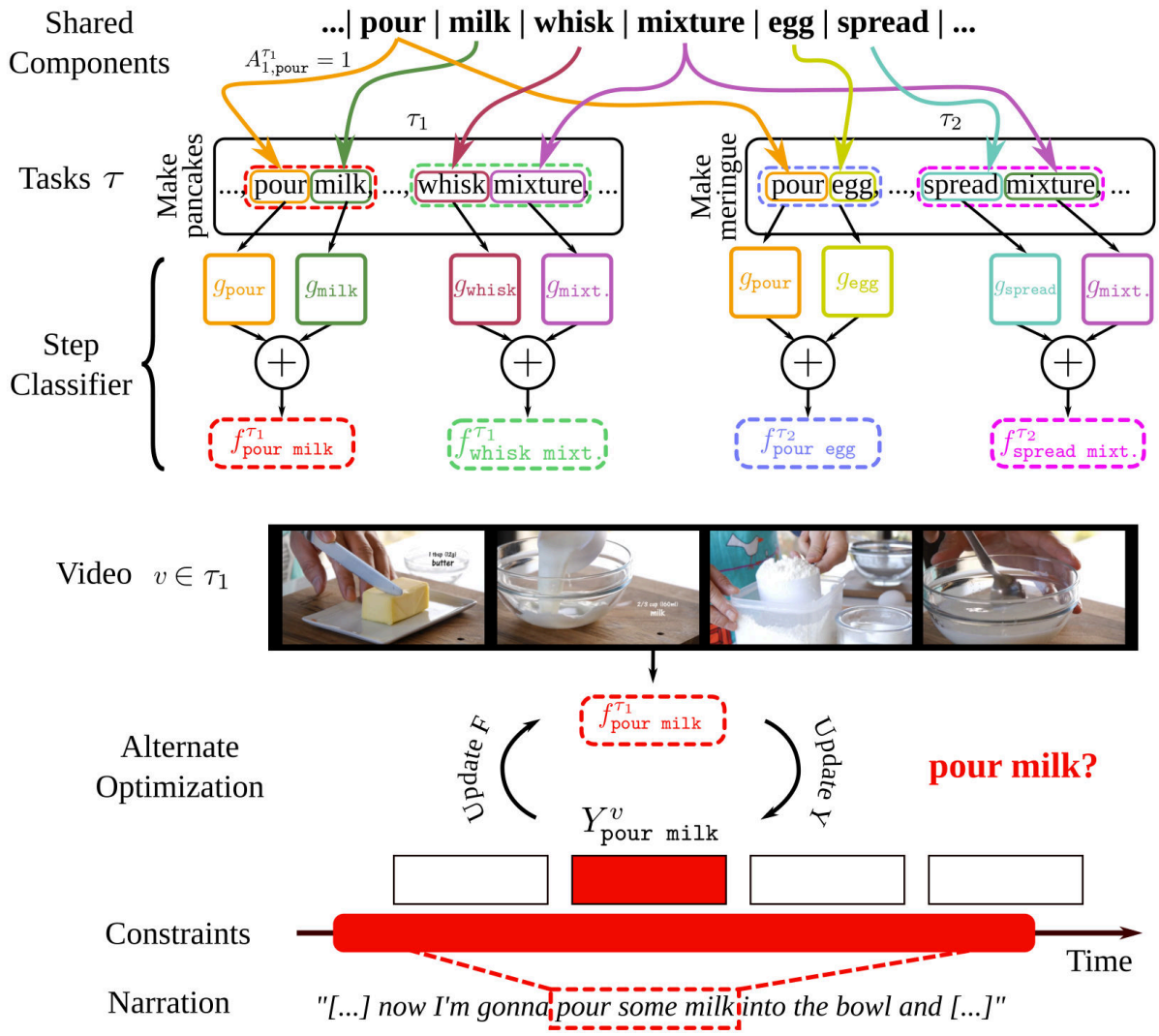


Figure 3.2: Our approach expresses classifiers for each step of each task in terms of a component model (e.g., writing the *pour milk* as a *pour* and *milk* classifier). We thus cast the problem of learning the steps as learning an underlying set of component models. We learn these models by alternating between updating labels for these classifiers and the classifiers themselves while using constraints from narrations.

O. Sener et al. 2015], the framework shares components (e.g., nouns and verbs) across steps. This sharing improves performance, as we empirically confirm, and enables the parsing of unseen tasks.

In order to properly evaluate the importance of sharing, we gather a dataset of instruc-

tional videos. These have attracted a great deal of attention recently [Alayrac et al. 2016, 2017; Huang et al. 2018, 2017; Malmaud et al. 2015; O. Sener et al. 2015; L. Zhou et al. 2018a] since the co-occurrence of demonstrative visual actions and natural language enables many interesting tasks ranging from coreference resolution [Huang et al. 2017] to learning person-object interaction [Alayrac et al. 2017; Damen et al. 2014]. Existing data, however, is either not large (e.g., only 5 tasks [Alayrac et al. 2017]), not diverse (e.g., YouCookII [L. Zhou et al. 2018a] is only cooking), or not densely temporally annotated (e.g., What's Cooking? [Malmaud et al. 2015]). We thus collect a dataset that is: **(i)** relatively large (83 tasks, 4.7K videos); **(ii)** simultaneously diverse (Covering car maintenance, cooking, crafting) yet also permitting the evaluation of sharing as it has related tasks; and **(iii)** annotated for temporal localization, permitting evaluation. The scale, and relatedness, as we demonstrate empirically contribute to increased performance of visual models.

Our technical approach to the problem builds particularly heavily on the use of discriminative clustering [Bach et al. 2007; L. Xu et al. 2004], or the simultaneous constrained grouping of data samples and learning of classifiers for groups. Past work in this area has either had operated with complex constraints and a restricted classifier (e.g., minimizing the L2 loss with linear model [Alayrac et al. 2017; Bach et al. 2007]) or an unrestricted classifier, such as a deep network, but no constraints [Bojanowski et al. 2017b; Caron et al. 2018]. Our weakly supervised setting requires the ability to add constraints in order to converge to a good solution while our compositional model and desired loss function requires the ability to use an unrestricted classifier. We therefore propose an optimization approach that handles both, letting us train with a compositional model while also using temporal constraints.

Finally, our sharing between tasks is enabled via the composition of the components of each step (e.g., nouns, verbs). This is similar to attributes [Farhadi et al. 2009; Ferrari et al. 2007], which have been used in action recognition in the past [J. Liu et al. 2011; Yao et al. 2011]. Our components are meaningful (representing, e.g., "lemon") but also automatically built; they are thus different than pre-defined semantic attributes (not automatic) and the non-semantic attributes (not intrinsically meaningful) as defined in [Farhadi et al. 2009]. It is also related to methods that compose new classifiers from

others, including [Guadarrama et al. 2013; Misra et al. 2017; Yatskar et al. 2017] among many others. Our framework is orthogonal, and shows how to learn these in a weakly-supervised setting.

3.3 Overview

Our goal is to build visual models for a set of **tasks** from instructional videos. Each task is a multi-step process such as *making latte* consisting of multiple **steps**, such as *pour milk*. We aim to learn a visual model for each of these steps. Our approach uses **component models** that represent each step in terms of its constituent **components** as opposed to a monolithic entity, as illustrated in Figure 3.2. For instance, rather than building a classifier solely for *whisk mixture* in the context of *make pancakes*, we learn a set of classifiers per-component, one for *whisk*, *spread*, *mixture* and so on, and represent *whisk mixture* as the combination of *whisk* and *mixture* and share *mixture* with *spread mixture*. This shares data between steps and enables the parsing of previously unseen tasks, which we both verify empirically.

We make a number of assumptions. Throughout, we assume that we are given an ordered list of steps for each task. This list is our only source of manual supervision and is done once per-task and is far less time consuming than annotating a temporal segmentation of each step in the input videos. At training time, we also assume that our training videos contain audio that explains what actions are being performed. At test time, however, we do not use the audio track: just like a person who watches a video online, once our system is shown how to make a latte with narration, it is expected to follow along without step-by-step narrations.

3.4 Modeling instructional videos

We now describe our technical approach for using a list of steps to jointly learn the labels and visual models on a set of narrated instructional videos. This is weakly supervised since we provide only the list of steps, but not their temporal locations in training videos.

Problem formulation. We denote the set of narrated instructional videos $\mathcal{V}$. Each video $v \in \mathcal{V}$ contains a sequence of N_v segments of visual features $X^v = (x_1, \ldots, x_{N_v})$ as well as narrations we use later. For every task τ we assume to be given a set of videos V_τ together with a set of ordered natural language steps K_τ.

Our goal is then to discover a set of classifiers F that can identify the steps of the tasks. In other words, if τ is a task and k is its step, the classifier f_k^τ determines whether a visual feature depicts step k of τ or not. To do this, we also learn a labeling Y of the training set for the classifiers, or for every video v depicting task τ, a binary label matrix $Y^v \in \{0,1\}^{N_v \times K_\tau}$ where $Y^v_{tk} = 1$ if time t depicts step k and 0 otherwise. While jointly learning labels and classifiers leads to trivial solutions, we can eliminate these and make meaningful progress by constraining Y and by sharing information across the classifiers of F.

3.4.1 Component classifiers

One of the main focuses of this chapter is in the form of the step classifier f. Specifically, we propose a component model that represents each step (e.g., "pour milk") as a combination of components (e.g., "pour" and "milk"). Before explaining how we formulate this, we place it in context by introducing a variety of alternatives that vary in terms of how they are learned and formulated.

The simplest approach, a **task-specific step model**, is to learn a classifier for each step in the training set (i.e., a model for *pour egg* for the particular task of *making pancakes*). Here, the model simply learns $\sum_\tau K_\tau$ classifiers, one for each of the K_τ steps in each task, which is simple but which permits no sharing.

One way of adding sharing would be to have a **shared step model**, where a single classifier is learned for each unique step in the dataset. For instance, the *pour egg* classifier learns from both *making meringues* and *making pancakes*. This sharing, however, would be limited to exact duplicates of steps, and so while *whisk milk* and *pour milk* both share an object, they would be learned separately.

Our proposed **component model** fixes this issue. We automatically generate a vo-

cabulary of **components** by taking the set of stemmed words in all the steps. These components are typically objects, verbs and prepositions and we combine classifiers for each component to yield our steps. In particular, for a vocabulary of M components, we define a per-task matrix $A^\tau \in \{0,1\}^{K_\tau \times M}$ where $A^\tau_{k,m} = 1$ if the step k involves components m and 0 otherwise. We then learn M classifiers $g_1, \ldots, g_M$ such that the prediction of a step f^τ_k is the average of predictions provided by component classifiers

$$f^\tau_k(x) = \sum_m A^\tau_{km} g_m(x) / \sum_m A^\tau_{km}. \tag{3.1}$$

For instance, the score for *pour milk* is the average of outputs of g_{pour} and g_{milk}. In other words, when optimizing over the set of functions F, we optimize over the parameters of $\{g_i\}$ so that when combined together in step models via (3.1), they produce the desired results.

3.4.2 Objective and constraints

Having described the setup and classifiers, we now describe the objective function we minimize. Our goal is to simultaneously optimize over step location labels Y and classifiers F over all videos and tasks

$$\min_{Y \in \mathcal{C}, F \in \mathcal{F}} \sum_\tau \sum_{v \in \mathcal{V}(\tau)} h(X^v, Y^v; F), \tag{3.2}$$

where $\mathcal{C}$ is the set of temporal constraints on Y defined below, and $\mathcal{F}$ is a family of considered classifiers. Our objective function per-video is a standard cross-entropy loss

$$h(X^v, Y^v; F) = -\sum_{t,k} Y^v_{tk} \log \left(\frac{\exp\left(f^\tau_k(x^v_t)\right)}{\sum_{k'} \exp(f^\tau_{k'}(x^v_t))} \right). \tag{3.3}$$

Optimizing (3.2) may lead to trivial solutions (e.g., $Y^v = 0$ and F outputting all zeros). We thus constrain our labeling of Y to avoid this and ensure a sensible solution. In particular, we impose three constraints:

At least once. We assume that every video v of a task depicts each step k at least once, or $\sum_t Y_{tk}^v \geq 1$.

Temporal ordering. We assume that steps occur in the given order. While not always strictly correct, this dramatically reduces the search space and leads to better classifiers.

Temporal text localization. We assume that the steps and corresponding narrations happen close in time, e.g., the narrator of a *grill steak* video may say "just put the marinated steak on the grill". We automatically compare the text description of each step to automatic YouTube subtitles. For a task with K_τ steps and a video with N_v frames, we construct a $[0,1]^{N_v \times K_\tau}$ matrix of cosine similarities between steps and a sliding-window word vector representations of narrations. More details are provided in Appendix A Since narrated videos contain spurious mentions of tasks (e.g., "before putting the steak on the grill, we clean the grill") we do not directly use this matrix, but instead find an assignment of steps to locations that maximizes the total similarity while respecting the ordering constraints. The visual model must then more precisely identify when the action appears. We then impose a simple hard constraint of disallowing labelings Y^v where any step is outside of the text-based interval (average length 9s)

3.4.3 Optimization and inference

We solve problem (3.2) by alternating between updating assignments Y and the parameters of the classifiers F.

Updating Y. When F is fixed, we can minimize (3.2) w.r.t. Y independently for each video. In particular, fixing F fixes the classifier scores, meaning that minimizing (3.2) with respect to Y^v is a constrained minimization of a linear cost in Y subject to constraints.

Updating F. When Y is fixed, our cost function reduces to a standard supervised classification problem. We can thus apply standard techniques for solving these, such as stochastic gradient descent. Detailed description of our optimization procedure is provided in Appendix A.

Initialization. Our objective is non-convex and has local minima, thus a proper initialization is important. We obtain such an initialization by treating all assignments that

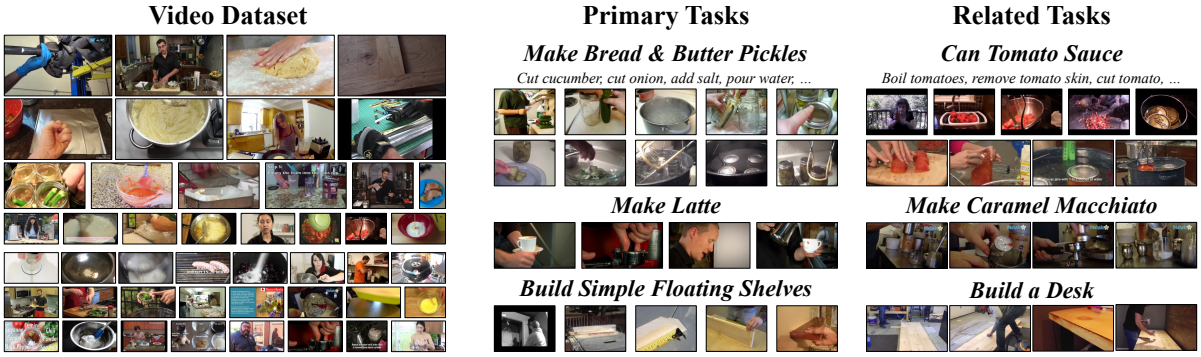


Figure 3.3: Our new dataset, used to study sharing in a weakly supervised learning setting. It contains primary tasks, such as *make bread and butter pickles*, as well as related tasks, such as *can tomato sauce*. This lets us study whether learning multiple tasks improves performance.

satisfy the temporal text localization constraints as ground-truth and optimizing for F for 30 epochs, each time drawing a random sample that satisfies the constraints.

Inference. Once the model has been fit to the data, inference on a new video v of a task τ is simple. After extracting features, we run each classifier f on every temporal segment, resulting in a $N_v \times K_\tau$ score matrix. To obtain a hard labeling, we use dynamic programming to find the best-scoring labeling that respects the given order of steps.

3.4.4 Implementation details

Networks: Due to the limited data size and noisy supervision, we use a linear classifier with dropout for regularization. Preliminary experiments with deeper models did not yield improvements. We use ADAM [Kingma et al. 2014] with the learning rate of 10^{-5} for optimization. *Features:* We represent each video segment x_i using RGB I3D features [Carreira et al. 2017] (1024D), Resnet-152 features [He et al. 2016] (2048D) extracted at each frame and averaged over one-second temporal windows, and audio features from [Hershey et al. 2017] (128D). *Components:* We obtain the dictionary of components by finding the set of unique stemmed words over all step descriptions. The total number of components is 383. *Hyperparameters:* Dropout and the learning rate are chosen on a validation data set.

Table 3.1: A comparison of CrossTask with existing instructional datasets. Our dataset is both large and more diverse while also having temporal annotations.

	Num. Vids	Total Length	Num. Tasks	Not only Cooking	Avail. Annots
[Alayrac et al. 2017]	150	7h	5		Windows
[O. Sener et al. 2015]	1.2K+85	100h	17		Windows
[L. Zhou et al. 2018a]	2K	176h	89		Windows
[Malmaud et al. 2015]	180K	3,000h			Recipes
CrossTask	4.7K	375h	83		Windows

3.5 CrossTask dataset

One goal of this chapter is to investigate whether sharing improves the performance of weakly supervised learning from instructional videos. To do this, we need a dataset covering a diverse set of interrelated tasks and annotated with temporal segments. Existing data fails to satisfy at least one of these criteria and we therefore collect a new dataset (83 tasks, 4.7K videos) related to cooking, car maintenance, crafting, and home repairs. These tasks and their steps are derived from wikiHow, a website that describes how to solve many tasks, and the videos come from YouTube.

CrossTask dataset is divided into two sets of tasks to investigate sharing. The first is **primary tasks**, which are the main focus of our investigation and the backbone of the dataset. These are fully annotated and form the basis for our evaluations. The second is **related tasks** with videos gathered in a more automatic way to share some, but not all, components with the primary tasks. One goal of our experiments is to assess whether these related tasks improve the learning of primary tasks, and whether one can learn a good model only on related tasks.

3.5.1 Video collection procedure

We begin the collection process by defining our tasks. These must satisfy three criteria: they must entail a sequence of physical interactions with objects (unlike e.g., *how to get into a relationship*); their step order must be deterministic (unlike e.g., *how to play chess*); and they must appear frequently on YouTube. We asked annotators to review

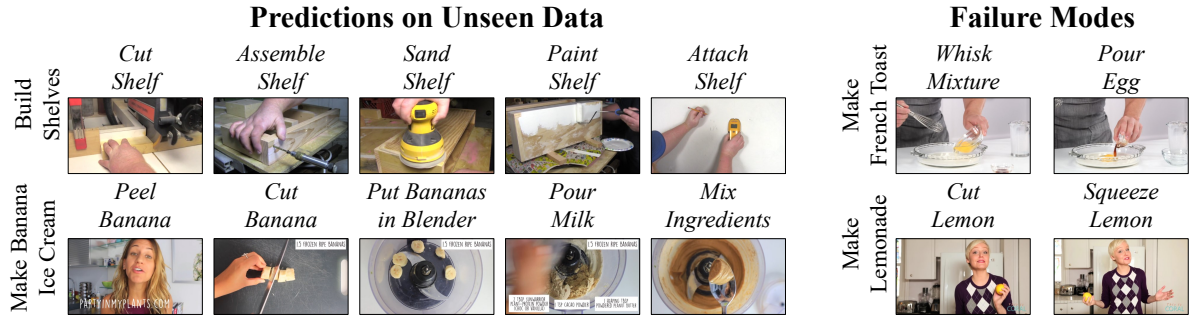


Figure 3.4: Predictions on unseen data as well as typical failure modes. Our method does well on steps with distinctive motions and appearances. Failure modes include (top) features that cannot make fine-grained distinctions between e.g., egg and vanilla extract; and (bottom) models that overreact to particular nouns, preferring a more visible lemon over a less visible lemon actually being squeezed.

the tasks in five sections of wikihow to get tasks satisfying the first two criteria, yielding $\sim$ 7K candidate tasks, and manually filter for the third criteria.

We select 18 primary tasks and 65 related tasks from these 7K candidate tasks. The primary tasks cover a variety of themes (e.g., auto repair to cooking to DIY) and include *building floating shelves* and *making latte*. We find 65 related tasks by finding related tasks for each primary task. We generate potential related tasks for a primary task by comparing the wikiHow articles using a TF-IDF on a bag-of-words representation, which finds tasks with similar descriptions. We then filter out near duplicates (e.g., *how to jack up a car* and *how to use a car jack*) by comparing top YouTube search results and removing candidates with overlaps, and manually remove a handful of irrelevant tasks.

We define steps and their order for each task by examining the wikiHow articles, beginning with the summaries of each step. Using the wikiHow summary itself is insufficient, since many articles contain non-visual steps and some steps combine multiple physical actions. We thus manually correct the list yielding a set of tasks with 7.4 steps on average for primary tasks and 8.8 for related tasks.

We then obtain videos for each task by searching YouTube. Since the related tasks are only to aid the primary tasks, we simply take the top N results from YouTube, without any manual verification. For primary tasks, we ask annotators to filter a larger pool of top results while examining the video, steps, and wikiHow illustrations, yielding at least

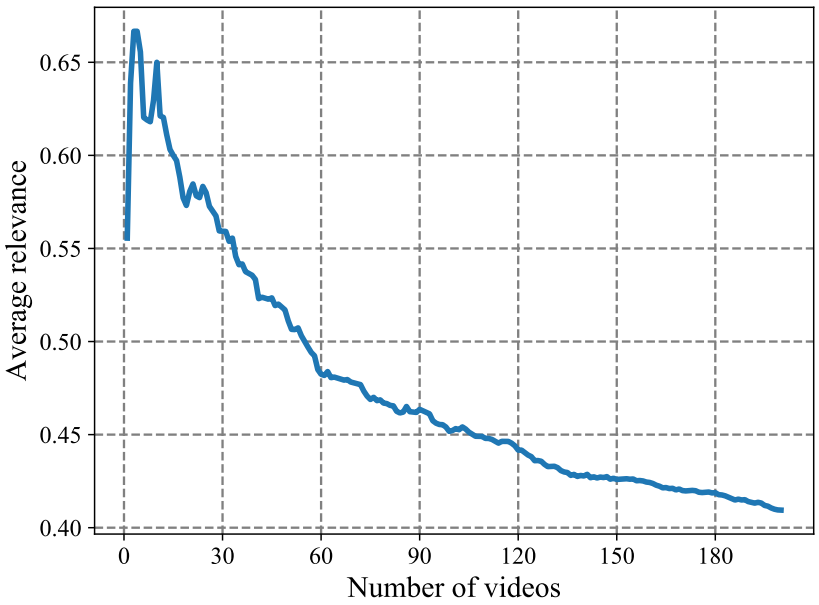


Figure 3.5: Average relevance of videos as a function of the number of videos collected from YouTube. Taking top 30 videos per task results in 56% relevant videos. Attempting to collect more videos results in a noisy dataset with many irrelevant videos.

80 videos per task.

The choice of the number of videos N per task relies on the following trade-off. Training the model on a large number of videos for a given task may lead to better performance. On the other hand, large N results in many videos being unrelated to the queried task, which may hurt the performance of the model. To investigate the influence of N on the purity of the data, we have annotated videos from YouTube search output as relevant or irrelevant to a task for all primary tasks. We define the average relevance as the ratio between the number of relevant videos and the total number of videos. Figure 3.5 shows the average relevance for different values of N. The relevance rapidly decreases with N, making the data unusable without manual cleaning. For each related task we take top 30 videos from YouTube, which is a reasonable compromise between the amount of data and the level of noise.

Since the videos are collected automatically for each task, they may be shared between tasks. The primary tasks have little in common and do not share any videos. The

Table 3.2: Statistics for primary tasks.

Task	Number of videos	Number of steps	Average length	Missing steps	Background	Order consistency
Make Kimchi Rice	120	6	4:47	21%	70%	0,69
Pickle Cucumber	106	11	5:35	48%	75%	0,85
Make Banana Ice Cream	170	5	4:04	38%	80%	0,98
Grill Steak	228	11	5:26	46%	75%	0,95
Jack Up Car	89	3	4:13	39%	81%	1,00
Make Jello Shots	182	6	4:15	21%	72%	0,87
Change Tire	99	11	4:52	27%	62%	0,97
Make Lemonade	131	8	3:44	28%	69%	0,80
Add Oil to Car	137	8	5:39	33%	85%	0,92
Make Latte	157	6	3:52	43%	71%	0,89
Build Floating Shelves	153	5	5:23	34%	58%	0,96
Make Taco Salad	170	8	4:44	41%	79%	0,66
Make French Toast	252	10	4:10	23%	68%	0,80
Make Irish Coffee	185	5	3:13	13%	74%	0,77
Make Strawberry Cake	86	9	5:36	25%	63%	0,82
Make Pancakes	182	8	4:34	19%	70%	0,89
Make Meringue	154	6	4:42	23%	67%	0,98
Make Fish Curry	149	7	5:31	25%	69%	0,74
Average	**153**	**7**	**4:57**	**31%**	**72%**	**0,86**

related tasks, however, may be similar to each other and to the primary tasks (*e.g.* *Make Sourdough Pancakes* and *Make Pancakes*). We found that the primary and related tasks share about 2.6% of videos. However, we stress the fact that such duplicate videos are provided with different supervision (different ordered list of steps) for each task. Our videos come from 3007 different YouTube channels, with 1.6 videos per channel on average. 70% of videos from primary tasks do not share channels with videos from related tasks. We, thus, conclude, that the difference between videos collected for primary and related tasks is sufficiently large.

3.5.2 Annotations and statistics

Task localization annotations. Since our focus is the primary tasks, annotators mark the temporal extent of each primary task step independently. We do this for our 18 primary tasks and make annotations publically available.

Dataset. This results in a dataset containing 2763 videos of 18 primary tasks comprising 213 hours of video; and 1950 videos of 65 related tasks comprising 161 hours of video. We contrast this dataset with past instructional video datasets in Table 3.1. Our dataset is simultaneously large while also having precise temporal segment annotations.

Table 3.3: Weakly supervised recall scores on test set (in %). Our approach, which shares information across tasks, substantially and consistently outperforms non-sharing baselines. The standard deviation for reported scores does not exceed 1%.

	Make Kimchi Rice	Pickle Cucumber	Make Banana Ice Cream	Grill Steak	Jack Up Car	Make Jello Shots	Change Tire	Make Lemonade	Add Oil to Car	Make Latte	Build Shelves	Make Taco Salad	Make French Toast	Make Irish Coffee	Make Strawberry Cake	Make Pancakes	Make Meringue	Make Fish Curry	Average
Supervised	19.1	25.3	38.0	37.5	25.7	28.2	54.3	25.8	18.3	31.2	47.7	12.0	39.5	23.4	30.9	41.1	53.4	17.3	31.6
Uniform	4.2	7.1	6.4	7.3	**17.4**	7.1	14.2	9.8	3.1	10.7	22.1	5.5	9.5	7.5	9.2	9.2	19.5	5.1	9.7
[Alayrac et al. 2016]	**15.6**	10.6	7.5	14.2	9.3	11.8	17.3	13.1	**6.4**	12.9	27.2	9.2	15.7	8.6	16.3	13.0	23.2	7.4	13.3
[Richard et al. 2018a]	7.6	4.3	3.6	4.6	8.9	5.4	7.5	7.3	3.6	6.2	12.3	3.8	7.4	7.2	6.7	9.6	12.3	3.1	6.7
Task-Specific Step-Based	13.2	17.6	19.3	19.3	9.7	12.6	30.4	16.0	4.5	19.0	29.0	9.1	29.1	**14.5**	22.9	29.0	32.9	7.3	18.6
Proposed	13.3	**18.0**	**23.4**	**23.1**	16.9	**16.5**	**30.7**	**21.6**	4.6	**19.5**	**35.3**	**10.0**	**32.3**	13.8	**29.5**	**37.6**	**43.0**	**13.3**	**22.4**
Gain from Sharing	0.2	0.4	4.1	3.8	7.2	3.9	0.3	5.6	0.1	0.6	6.3	0.9	3.2	-0.7	6.6	8.7	10.1	6.0	3.7

Table 3.2 provides some statistics for the primary tasks. The videos are quite long, with an average length of 4min 57sec, and depict fairly complex tasks, with 7.4 steps on average. Less complex tasks include *jack up a car* (3 steps); more complex ones include *pickle cucumbers* or *change tire* (11 steps each). The order consistency, defined as in [Alayrac et al. 2018], shows how well the order is respected in the videos. For example, if the order of steps in a video is $2 \rightarrow 1 \rightarrow 3$, the order consistency for this video would be equal to $\frac{2}{3}$. The amount of background is defined as an average number of frames, which are not assigned to any step, divided by a total number of frames.

Challenges. In addition to being long and complex, the videos in CrossTask dataset are challenging since they do not precisely show the ordered steps we have defined. While the average order consistency is high (86%), thus justifying the use of hard ordering constraints, it varies across the tasks and has a relatively low value for some of the tasks (*e.g. Make Kimchi Rice* and *Make Taco Salad*). The high amount of background (72% in average) motivates the use of methods, that aren't limited to a dense segmentation of a video and allow frames to remain unlabeled. On average, 31% of steps are not depicted due to variances in procedures and omissions. The amount of missing steps is close to 50% for *Grill Steak* and *Pickle Cucumber*, making these tasks especially challenging for our method.

3.6 Experiments

Our experiments aim to address the following three questions about cross-task sharing in the weakly-supervised setting: **(1)** Can the proposed method use related data to improve performance? **(2)** How does the proposed component model compare to sharing alternatives? **(3)** Can the component model transfer to previously unseen tasks? Throughout, we evaluate on the large dataset introduced in Section 3.5 that consists of primary tasks and related tasks. We address (1) in Section 3.6.1 by comparing our proposed approach with methods that do not share and show that our proposed approach can use related tasks to improve performance on primary asks. Section 3.6.2 addresses (2) by analyzing the performance of the model and showing that it outperforms step-based alternatives. We answer (3) empirically in Section 3.6.3 by training only on related tasks, and show that we are able to perform well on primary tasks.

3.6.1 Cross-task learning

We begin by evaluating whether our proposed component model approach can use sharing to improve performance on a fixed set of tasks. We fix our evaluation to be the 18 primary tasks and evaluate whether the model can use the 65 related tasks to improve performance.

Metrics and setup. We evaluate results on 18 primary tasks over the videos that make up the test set. We quantify performance via *recall*, which we define as the ratio between the number of correct step assignments (defined as falling into the correct ground-truth time interval) and the total number of steps over all videos. In other words, to get a perfect score, a method must correctly identify one instance of each step of the task in each test video. All methods make a single prediction per step, which prevents the trivial solution of assigning all frames to all actions.

We run experiments 20 times, each time making a train set of 30 videos per task and leaving the remaining 1863 videos for test. We report the average. Hyperparameters are set for all methods using a fixed validation set of 20 videos per primary task that are never used for training or testing.

Baselines. Our goal is to examine whether our sharing approach can leverage related tasks to improve performance on our primary task. We compare our method to its version without sharing as well as to a number of baselines. (1) Uniform: simply predict steps at fixed time intervals. Since this predicts steps in the correct order and steps often break tasks into roughly equal chunks, this is fairly well-informed prior. (2) [Alayrac et al. 2016]: the weakly supervised learning method for videos. This is similar in spirit to our approach except it does not share and optimizes a L2-criterion via the DIFFRAC [Bach et al. 2007] method. (3) [Richard et al. 2018a]: the weakly supervised learning method that does not rely on the known order of steps. (4) Task-Specific Steps: Our approach trained independently for each step of each task. In other words, there are separate models for *pour egg* in the contexts of *making pancakes* and *making meringue*. This differs from [Alayrac et al. 2016] in that it optimizes a cross-entropy loss using our proposed optimization method. It differs from our full proposed approach since it performs no sharing. Note, that the full method in [Alayrac et al. 2016] includes automatic discovery of steps from narrations. Here, we only use the visual model of [Alayrac et al. 2016], while providing the same constraints as in our method. This allows for a fair comparison between [Alayrac et al. 2016] and our method, since both use the same amount of supervision. At test time, the method presented in [Richard et al. 2018a] has no prior about which steps are present or the order in which they occur. To make a fair comparison, we use the trained classifier of the method in [Richard et al. 2018a], and apply the same inference procedure as in our method.

Qualitative results. We illustrate qualitative results of our full method in Figure 3.4. We show a parses of unseen videos of *Build Shelves* and *Make Banana Ice Cream* and failure modes. Our method can handle well a large variety of tasks and steps but may struggle to identify some details (e.g., vanilla vs. egg) or actions.

Quantitative results. Table 3.3 shows results summarized across steps. The uniform baseline provides a strong lower bound, achieving an average recall of 9.7% and outperforming [Richard et al. 2018a]. Note, however, that [Richard et al. 2018a] is designed to address a different problem and cannot be fairly compared with other methods in our setup. While [Alayrac et al. 2016] improves on this (13.3%), it does substantially worse than our task-specific step method (18.6%). We found that predictions from [Alayrac

et al. 2016] often had several steps with similar scores, leading to poor parse results, which we attribute to the convex relaxation used by DIFFRAC. This was resolved in the past by the use of narration at test time; our approach does not depend on this.

Our full approach, which shares across tasks, produces substantially better performance (22.4%) than the task-specific step method. More importantly, this improvement is systematic: the full method improves on the task-specific step baseline in 17 tasks out of 18.

Verifying optimizer on small-scale data. We now evaluate our approach on the smaller 5-task dataset of [Alayrac et al. 2016]. Since here there are no common steps across tasks, we are able to test only the basic task-specific step-based version. To make a fair comparison, we use the same features, ordering constraints, as well as constraints from narration for every K as provided by the authors of [Alayrac et al. 2016], and we evaluate using the F1 metric as in [Alayrac et al. 2016]. As a result, the two formulations are on par, where [Alayrac et al. 2016] versus our approach result in 22.8% versus 21.8% for K=10 and 21.0% versus 21.1% for K=15, respectively. While these scores are slightly lower compared to those obtained by the single-task probabilistic model in [F. Sener et al. 2018] (25.4% at K=10 and 23.6% at K=15), we are unable to compare using our full cross-task model on this dataset. Overall, these results verify the effectiveness of our optimization technique.

3.6.2 Experimental evaluation of cross-task sharing

Having verified the framework and the role of sharing, we now more precisely evaluate how sharing is performed to examine the contribution of our proposed compositional model. We vary two dimensions. The first is the granularity, or at what level sharing occurs. We propose sharing at a component level, but one could share at a step level as well. The second is what data is used, including (i) independently learning primary tasks; (ii) learning primary tasks together; (iii) learning primary plus related tasks together.

Table 3.4 reveals that increased sharing consistently helps and component-based sharing extracts more from sharing than step-based (performance increases across rows). This

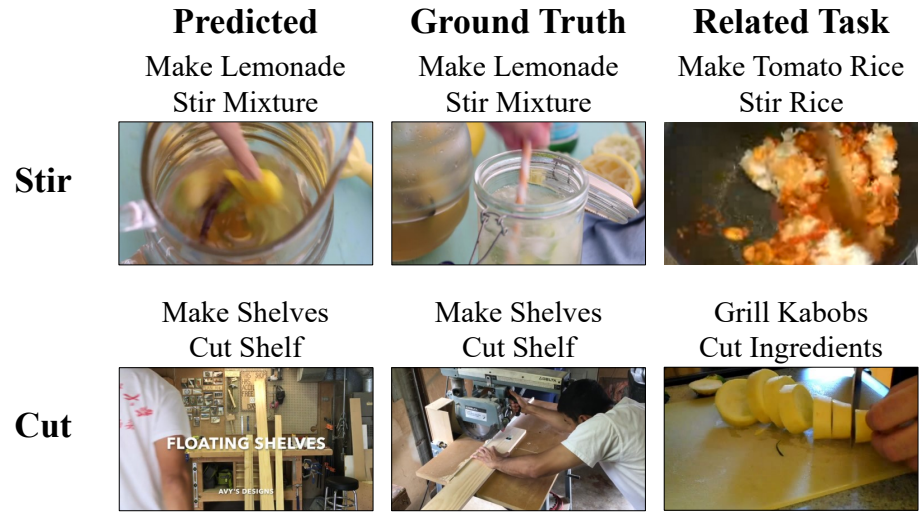


Figure 3.6: Components that share well and poorly: while stir shares well between steps of tasks, cut shares poorly when transferring from a food context to a home improvement context.

Table 3.4: Average recall scores on the test set for our method when changing the sharing settings and the model.

	Unshared Primary	Shared Primary	Shared Primary + Related
Step-based	18.6	18.9	19.8
Component-based	**18.7**	**20.2**	**22.4**

gain over step-based sharing is because step-based sharing requires exact matches. Most commonality between tasks occurs with slight variants (e.g., *cut* is applied to steak, tomato, pickle, etc.) and therefore a component-based model is needed to maximally enable sharing.

We illustrate some qualitative examples of steps benefiting and least benefiting from sharing in Figure 3.6. Typically, sharing can help if the component has distinctive appearance and is involved in a number of steps: steps involve stirring, for instance, have an average gain of 15% recall over independent training because it is frequent (in 30 steps) and distinctive. Of course, not all steps benefit: *cut shelf* is harmed (47% independent $\rightarrow$ 28% shared) because *cut* mostly occurs in cooking tasks with dissimilar contexts.

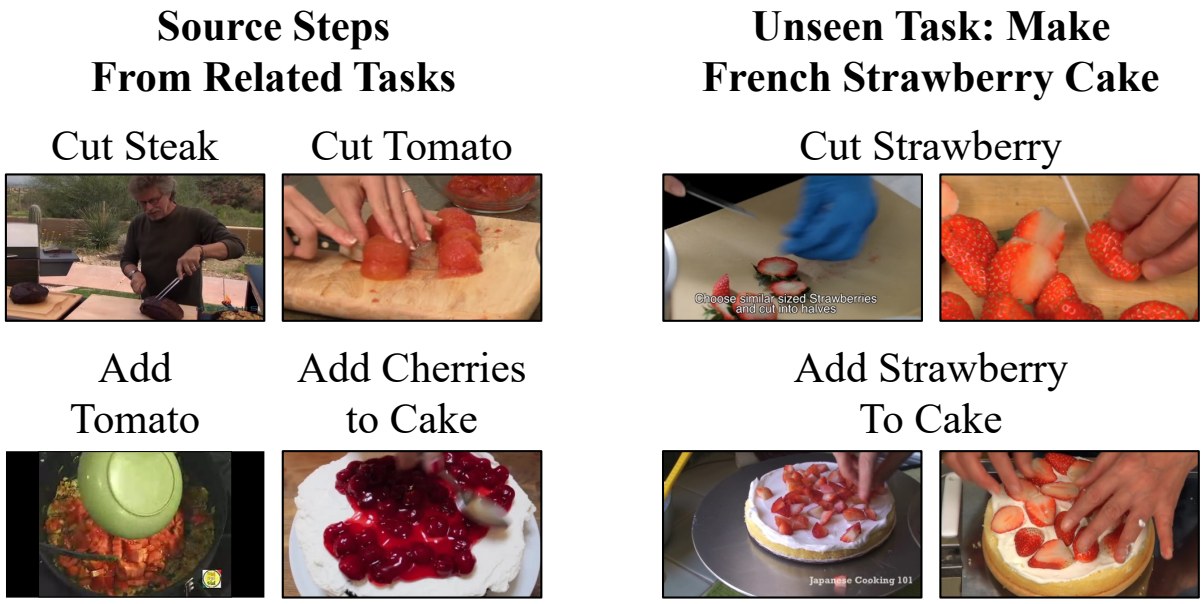


Figure 3.7: Examples of identified steps for an unseen task. While the model has not seen these steps and objects e.g., strawberries, its knowledge of other components leads to reasonable predictions.

3.6.3 Novel task transfer

One advantage of shared representations is that they can let one parse new concepts. For example, without any modifications, we can repeat our experiments from Section 3.6.1 in a setting where we never train on the 18 tasks that we test on but instead on the 65 related tasks. The only information given about the test tasks is an ordered list of steps.

Setup. As in Section 3.6.1, we quantify performance with recall on the 18 primary tasks. However, we train on a subset of the 65 related tasks and never on any primary task.

Qualitative results. We show a parse of steps of *Make Strawberry Cake* in Figure 3.7 using all related tasks. The model has not seen *cut strawberry* before but has seen other forms of cutting. Similarly, it has seen *add cherries to cake*, and can use this step to parse *add strawberries to cake*.

Quantitative results. Figure 3.8 shows performance as a function of the number of related tasks used for training. Increasing the number of training tasks improves perfor-

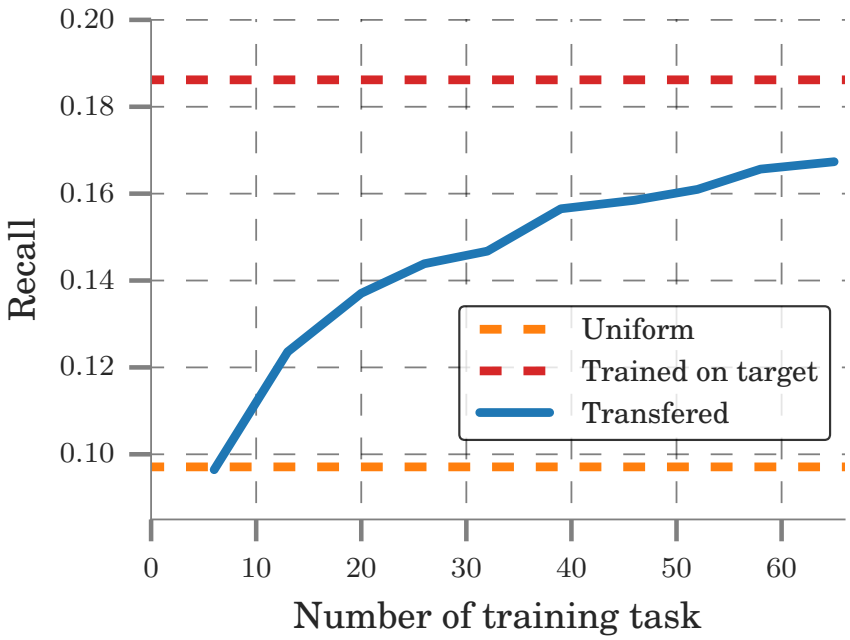


Figure 3.8: Recall while transferring a learned model to unseen tasks as a function of the number of tasks used for training. Our component model approaches training directly on these tasks.

mance on the primary tasks, and does not plateau even when 65 tasks are used.

3.7 Conclusion

In this chapter, we have introduced an approach for weakly supervised learning from instructional videos and a new CrossTask dataset for evaluating the role of sharing in this setting. Our component model has been shown ability to exploit common parts of tasks to improve performance and was able to parse previously unseen tasks. Future work would benefit from improved features as well as from improved versions of sharing.

Chapter 4

Learning actionness via long-range temporal order verification

In the previous chapter, we have presented the model for learning the steps of complex tasks, that is explicitly designed to leverage similarity between steps of different tasks. One limitation of this model is its inability to discriminate between the appearance of the steps and parts of the video that do not contain any steps, which we refer to as *background*. Instead, we rely on a simple assumption that each step is only 1 second long and the remaining video is covered by the background. This assumption is justified by the fact that most of the video (72% of total duration in CrossTask dataset) is covered by the background. However, this approach doesn't allow to accurately determine temporal extent of each step within a video.

In this chapter, we propose a method, that allows to separate steps from their background in instructional videos in an unsupervised way. Our approach is based on the observation, that the steps often appear in a particular temporal order, while the background segments usually do not contain any visual clues that would help to guess their order of appearance in the video. This means that given a set of video fragments, demonstrating steps of the task, it is easy to determine their order of appearance within the video, and it is difficult to determine the order of appearance of the background fragments. We formalize this observation in the form of a model that learns to assign an *actionness* score to video

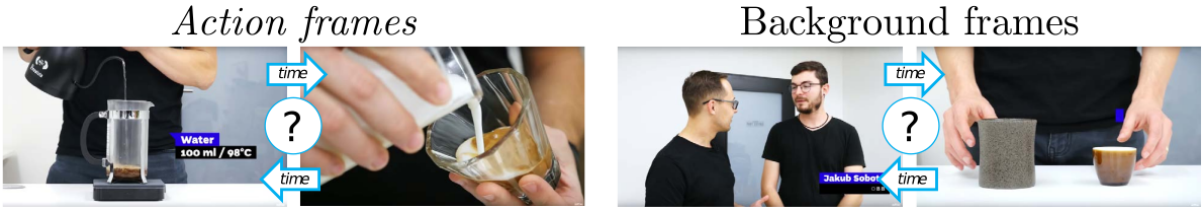


Figure 4.1: We show pairs of action frames and background frames from the same video. Can you predict the order of frames within each pair? While the order is relatively easy to guess for actions, the same task is more difficult for the background. We use this observation and exploit the predictability of temporal order as a measure of actionness.[1]

fragments by verifying their order of appearance. To demonstrate the effectiveness of our model, we evaluate it in three different scenarios. First, we use it to classify action versus background frame by frame. Second, we use it to generate action proposals. Third, we combine it with several step localization methods, including our method, described in Chapter 3, and demonstrate that our background model improves step localization in every case.

4.1 Introduction

Learning from web videos is becoming increasingly popular in computer vision as such videos are available in large quantities, and cover diverse activities and scenes. In particular, instructional videos have been recently explored as a rich source for many tasks and goal-driven sequences of actions [Alayrac et al. 2016; Huang et al. 2018; Miech et al. 2019; Y. Tang et al. 2019; L. Zhou et al. 2018a; Zhukov et al. 2019]. While the quantity and diversity of video data appears crucial for training current recognition models [Carreira et al. 2017; Miech et al. 2020, 2019], the manual annotation of actions in large-scale video data requires large efforts [Carreira et al. 2017] and may not scale well to the large number of possible actions. This is particularly true for the task of temporal action localization where sparse actions should be isolated in video streams from the large portion of "background" with no actions. For example, action frames in the CrossTask dataset [Zhukov et al. 2019] with typical instructional videos represent only 25.9% of the

[1]**Quiz answers:** Action frames are shown in correct temporal order; Background frames are shown in reverse temporal order.

total video length.

In our work we address the above challenges and aim to develop a self-supervised approach for separating a large and diverse set of actions from their background. We observe that actions typically do not happen in isolation and are often surrounded by other related actions. Moreover, action sequences often demonstrate a consistent order (taking off a car wheel should be preceded by lifting the car), hence, many actions can be identified by the predictability of their order with respect to other actions in the same video. On the contrary, the order of background frames is often hard to predict, hence, the low predictability for the order could be used to signify the background. We illustrate this idea with a quiz in Figure 4.1.

Temporal order has been explored as a supervisory signal by a number of recent works [Fernando et al. 2017; Misra et al. 2016; Wei et al. 2018; B. Zhou et al. 2018]. The goal of such methods, however, is to learn video representations by verifying the order for a short range of consequtive frames. We here address a different task and learn actionness [W. Chen et al. 2014] by exploiting long-range relations between video clips on the scale of minutes. To this end, we propose a new model and a method to learn actionness scores via a self-supervised proxy task of order verification. The model assigns high actionness scores to clips which order is easy to predict from other clips in the video. Our method is self-supervised and requires no manual annotation. Given this property, we use a very large HowTo100M dataset [Miech et al. 2019] with diverse and unlabeled instructional videos to learn an action-agnostic model for actionness. We show interesting insights of our method and demonstrate improved performance of action localization when combining our model with recent weakly-supervised approaches.

Contributions. This work makes the following contributions: (1) We develop a new model for action-agnostic action/background classification and propose to learn it via a self-supervised proxy task of long-range order verification. (2) We demonstrate a successful application of our method to the tasks of frame-wise action/background classification and action proposal generation evaluated on datasets with instructional videos COIN [Y. Tang et al. 2019] and CrossTask [Zhukov et al. 2019]; (3) We further demonstrate the benefit of our model for action localization by combining it with recent weakly-supervised methods of step localization on the CrossTask dataset; and (4) We provide ablation stud-

ies that give insights about our approach.

4.2 Related work

4.2.1 Self-supervised learning

Our work exploits the natural source of supervision in videos: the temporal order between frames. The proposed method is, hence, related to a large body of recent work on self-supervised learning, where the supervisory signal is obtained directly from the data and does not require manual annotation. The variety of recently proposed self-supervised tasks in the image domain include prediction of image rotation [Gidaris et al. 2018], spotting artifacts [Jenni et al. 2018], image colorization [Larsson et al. 2017; R. Zhang et al. 2016], cross-channel prediction [R. Zhang et al. 2017], inpainting [Pathak et al. 2016] and predicting relative position of patches [Doersch et al. 2015; Noroozi et al. 2016]. In the video domain, motion has been used as a cue for learning video representations in [Agrawal et al. 2015; Dwibedi et al. 2019; Pathak et al. 2017; X. Wang et al. 2015]. More related to our work, previous methods [Fernando et al. 2017; Lee et al. 2017; Misra et al. 2016; Wei et al. 2018; D. Xu et al. 2019] explore the temporal order, either by predicting the exact order of consecutive frames [Lee et al. 2017; D. Xu et al. 2019] or verifying their partial order [Fernando et al. 2017; Misra et al. 2016; Wei et al. 2018]. Our work builds on these ideas but brings two important innovations. First, in contrast to previous work that exploits temporal order to learn local video representations, we address a different task of action/background classification. As actions are often separated by minutes, our task requires reasoning about *long-range* temporal order, as opposed to *short-range* frame permutations explored by previous methods. Our second innovation is, hence, a new method that exploits long-range order verification for video clips and enables to model relations between actions in 10-20 minutes long videos.

4.2.2 Learning from instructional videos

Instructional videos have recently been in the focus of numerous works in the context of action localization [Alayrac et al. 2016; F. Sener et al. 2018; Zhukov et al. 2019], joint

learning of object states and actions [Alayrac et al. 2017], joint modeling of video and language [Miech et al. 2019; Sun et al. 2019] and visual reference resolution [Huang et al. 2018, 2017]. Some of this work exploits specific properties of instructional videos, such as the approximate temporal alignment between narrations and the visual content [Alayrac et al. 2016; Miech et al. 2019; Sun et al. 2019; Zhukov et al. 2019], and the order consistency [Alayrac et al. 2016; F. Sener et al. 2018; Zhukov et al. 2019]. Similarly, we rely on the partial order between actions. Our novelty is to use the order verification as a proxy task to discover most relevant parts of the video. To demonstrate the value of our approach, we combine it with the previous methods [Miech et al. 2020, 2019; Zhukov et al. 2019] for the task of weakly-supervised step localization in instructional videos and demonstrate consistent improvements.

4.2.3 Action proposals

We apply our method to generate action proposals. Action proposals is an essential part of many methods for action detection, explored by a number of recent papers [Escorcia et al. 2016; Gao et al. 2018; J. Ji et al. 2019; Lin et al. 2019, 2018; Y. Liu et al. 2019; Y. Zhao et al. 2017]. A popular approach to generate action proposals is to estimate an *actionness* score for each temporal unit and then apply some sort of temporal grouping and non-maxima suppression. The notion of actionness was first introduced in [W. Chen et al. 2014] as a confidence measure of intentional bodily movement of biological agents. Most works [Gao et al. 2018; Lin et al. 2019, 2018; Y. Liu et al. 2019; Y. Zhao et al. 2017] address actionness with supervised methods based on manual annotation of a known and limited set of action classes. This is done by training a binary classifier for estimating actionness score as first proposed in the context of spatial action detection [Limin Wang et al. 2016a]. Contrary to this approach, we aim to learn an actionness score without manual supervision by relying on generic assumptions about action order. Our definition of actions is narrower than in [W. Chen et al. 2014]. In particular, we only consider *goal-oriented actions*, necessary to perform specific manipulation tasks. This definition excludes actions such as gesticulation and conversations.

4.3 Unsupervised learning of actionness score

Given a large corpus of instructional videos depicting complex tasks, our goal is to automatically discover which segments of the videos are the most relevant for the successful completion of the tasks. We refer to these relevant segments as actions. Example of a complex task would be "*building a shelve*" and a relevant action would be "*drilling a hole*". Formally, we learn an *actionness* scoring function S that takes as input a video clip $x \in \mathbb{R}^{T\times H\times W\times 3}$ containing T frames of height H and width W and outputs a score $S(x) \in \mathbb{R}$ that is high when x corresponds to a relevant action and is low otherwise, *e.g.* on background scenes that are not relevant for completing the task.

We propose to learn S in a self-supervised manner through the pretext task of long range temporal order verification, which consists in predicting whether or not a set of video clips spanning a long temporal interval are in the correct order. The intuition is that one needs to isolate the relevant segments of the video that allow to best identify the correct ordering of events to be able to solve that task. We use that observation to train our actionness model S.

This section formally describes our method for joint order verification and actionness prediction. Section 4.3.1 introduces the model used. Section 4.3.2 details the training procedure, that allows to train the actionness score S via order verification.

4.3.1 Models for actionness and order verification

We represent each video clip x by a d-dimensional feature vector $h = f(x) \in \mathbb{R}^d$ obtained from a pre-trained video network f that we keep fixed throughout the work. In practice we use averaged pooled I3D representation [Richard et al. 2017] pretrained in [Miech et al. 2020], with $d = 1024$. For simplicity, we only refer to the feature vector h in the following (implicitly assuming h is associated with a video clip x). The actionness score $S(x)$ is estimated by a linear function on h, *i.e.* $S(x) = w^\top h + b$, where $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are learnt weights and biases, respectively. To predict the order of clips, we also introduce a model F that takes as input two clips x and x' and outputs the confidence $F(x, x') \in \mathbb{R}$ that x happens before x'. We force F to be antisymmetric (*i.e.* $F(x, x') = -F(x', x)$),

by defining $F(x, x') = a^\top(h - h')$, where $a \in \mathbb{R}^d$ and $h' = f(x')$. Our choice of simple linear models is practicaly motivated by the fact that in our experiments we did not see improvements from using more complex models. Next, we describe the training strategy used to train S and F using order verification.

4.3.2 Training with ordering verification

Actionness through order verification. Our goal is to learn actionness score in an unsupervised manner. As explained earlier we believe that actions contain more information than background in terms of predicting what happens before or what may come next in instructional videos as they carry more information about the global temporal structure of the video than the background. In this section, we use that observation to automatically differentiate actions from background. In short, the idea is to train a network to predict if a set of clips are in the correct order, a task for which it's trivial to get *free* supervision as correct ordering is naturally present in the video. In order to do so, we allow the model to softly select which pairs of clips from the set are best to perform that prediction, *i.e.* those that are most informative in terms of their relative order in the video. Hence, by learning to predict order through weighted relative ordering of pairs of clips, we expect the model to pay more attention on important actions and therefore learn a good actionness score S. Details are given next.

Order verification task. As illustrated in Figure 4.2, given a sequence X of M video clips $X = (x_i)_{i=1}^M$ randomly sampled from the *same* video, the task of order verification is to predict whether or not the clips in X are in the correct order, *i.e.* the same order as the original video.

Ground truth generation. We create positives and negatives for the verification task by simply randomly sampling M clips from a video and either **(i)** create a positive sequence by sorting clips in the order of their appearance in the video (label $y = 1$) or **(ii)** reverse completely the original sequence (negative label $y = 0$).

Order verification prediction. We seek to predict if the sequence X is in the correct order through our pairwise model F that can predict the relative ordering of a given pair

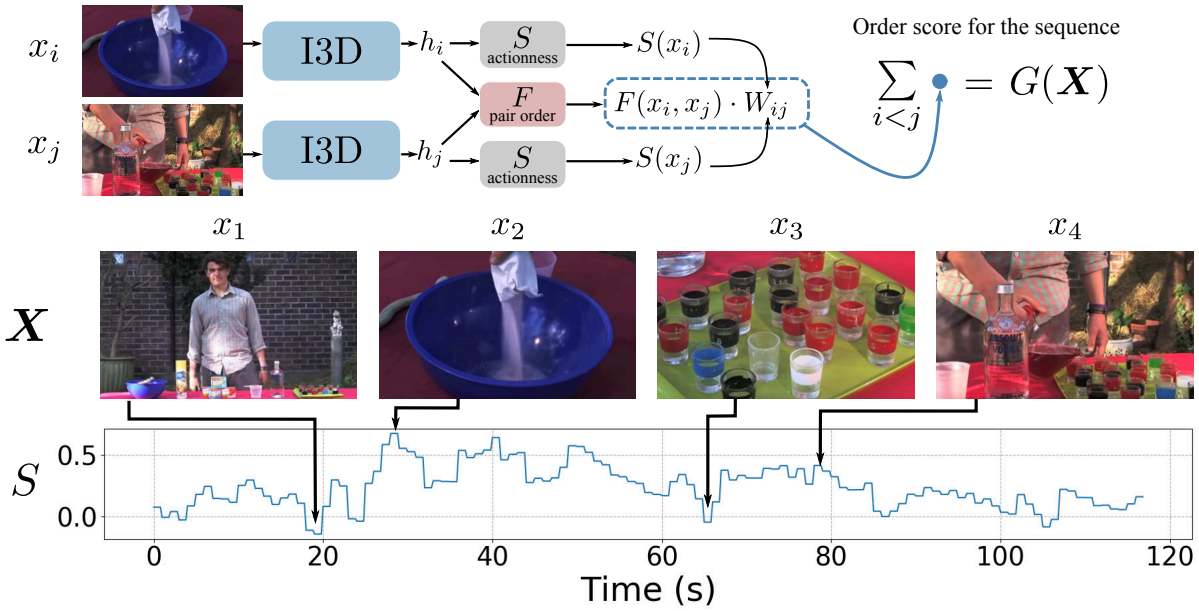


Figure 4.2: Given a sequence of clips $X = (x_1, x_2, ..., x_M)$ extracted from the same video as input, our model produces two types of outputs: confidence $S(x_i)$ that video clip x_i displays an action, and confidence $F(x_i, x_j)$ that x_i occurs before x_j in the original video. We combine these scores together to produce an order score $G(X)$ that reflects the model confidence that the sequence X is displayed in the correct order. We generate training data for G for *free* by simply maintaining or reverting order of videos. By doing so, the model automatically learns to put more weight $W_{ij} \propto \exp(S(x_i)+S(x_j))$ to clips from which it is easy to predict whether clip x_i has happened before clip x_i. We argue that clips with such properties are more likely to be actions that allow S to become an *actionness* score. **Top.** Architecture for producing the actionness score S and its training. **Bottom.** Evolution of the learnt actionness score S throughout a video. Note how the model learned to put higher weights to frames that correspond to actions such as *adding sugar* (x_2) or *pouring* (x_4) and low score on clips that are not relevant to the completion of the task (*i.e. background*) such as a *man standing still* (x_1) or a clip only showing *empty glasses* (x_3).

of clips (x_i, x_j). To make a more accurate prediction, it is better to aggregate scores over many pairs from the entire sequence X. For this reason, we predict the confidence $G(X)$ that $X = (x_i)_{i=1}^{M}$ is in the correct order as a weighted average of all pair-wise predictions:

$$G(X) = \sum_{i<j} W_{ij} F(x_i, x_j). \tag{4.1}$$

Note that the sum indices $i < j$ are to make sure that (i) we only compare pairs in the order given by the sequence and (ii) we do not compare a clip to itself. Finally the weights W_{ij} are defined with a softmax over all pairs of clips:

$$W_{ij} = \frac{\exp(S(x_i) + S(x_j))}{\sum\limits_{i'<j'} \exp(S(x_{i'}) + S(x_{j'}))}, \tag{4.2}$$

where $S(x_i)$ and $S(x_j)$ are the actionness score of our model for clips x_i and x_j, respectively. Because of the softmax (4.2) we have $\sum_{i<j} W_{ij} = 1$ and $W_{ij} \geq 0$. Hence the weights W_{ij} can be seen as a way to softly select the contribution of every individual pair (x_i, x_j) since they control the contribution of the individual order pairwise prediction $F(x_i, x_j)$ in the global order score $G(X)$ (see (4.1)). This contribution to $G(X)$ is proportional to the sum $S(x_i) + S(x_j)$ of actionness score of both clips x_i *and* x_j. This is to match our intuition that *both* clips should be depicting a relevant action to facilitate the order prediction. By learning G we will therefore indirectly learn S as described by the training objective below.

Training objective. Given a sequence X and associated label y indicating whether or not the sequence of clips is in the correct order, we use the binary cross-entropy loss $\mathcal{L}$ as follows

$$\mathcal{L}(X, y) = -y \log(\sigma(G(X))) - (1 - y) \log(1 - \sigma(G(X))), \tag{4.3}$$

where σ is the sigmoid function. This loss will enforce that when X is in the correct order (*i.e.* $y = 1$), then $G(X)$ should be maximized. To do so, the actionness model S needs to be *high* on clips x_i and x_j from which it's *easier* to predict their correct ordering (meaning high value of $F(x_i, x_j)$) so that their contribution in $G(X)$ will be maximal. Conversely, when X is in the incorrect order (*i.e.* $y = 0$), then $G(X)$ should be minimized. Again, to do so the actionness model S needs to be *high* on clips x_i and x_j from which it's

easier to predict their incorrect ordering (meaning low value of $F(x_i, x_j)$) so that their contribution in $G(X)$ will be maximal. Following the standard procedure, we minimize the expected loss $\mathcal{L}$ on our training dataset.

4.4 Experiments

The experimental setup is described in Section 4.4.1. We then provide in Section 4.4.2 an ablation study of highlighting the most important components of our method. Finally, in Section 4.4.3, we show how we can use our learned actionness score for applications such as action proposals or weakly supervised action localization.

4.4.1 Experimental setup

Input processing. Given a clip x containing T frames, we extract $h(x)$ using the I3D backbone from [Miech et al. 2020], pre-trained on HowTo100M for the task of joint embedding of videos and subtitles. We extract the features at `Mixed_5c` in a fully convolutional manner and perform a global average pooling to have a single feature vector $h(x)$ of dimension 1024. Note that this network was trained without requiring manual annotations. To reduce computation cost during training, we pre-extract these features and directly work in feature space. In particular, this means that we do not finetune the I3D backbone for our task.

Training dataset. Due to the large variety of actions in instructional videos, the variety of their visual appearance as well as the order in which they are performed in videos, we require a large amount of data for our self-supervised task. For this reason, we train our model on HowTo100M [Miech et al. 2019], a large dataset containing more than 1.2 million videos depicting around $23,000$ different tasks and was collected without manual annotation.

Training details. We optimize the objective (4.3) using the Adam optimizer [Kingma et al. 2014] on a single GPU with batch size 1024, initial learning rate of 10^{-4} that we decay by a factor 0.9 at every epoch. We train for a total of 15 epochs. In addition, since

HowTo100M is biased towards some specific domains (*e.g.* 40% of videos are cooking), we resample the data, using the task taxonomy of HowTo100M. Precisely, we consider all subcategories of depth 3, *i.e.* subcategories of the principal categories, such as Food and Cars, and sample equal number of videos from each subcategory. We also remove subcategories with less than 3000 videos.

Evaluation datasets. We use two instructional video datasets, COIN [Y. Tang et al. 2019] and CrossTask [Zhukov et al. 2019]. Both datasets contain untrimmed videos that have been temporally annotated with action labels corresponding to the different steps of the task. In the following, time intervals without any action labels are considered as background. We use the official COIN test subset of 2797 videos for evaluation. Since there is no official test subset of CrossTask, we randomly split it into a training (2062 videos) and test sets (688 videos). In addition, we made sure to discard all COIN and CrossTask videos from our training set.

Evaluation tasks. We use three different tasks for evaluation, detailed next.

Background vs. action classification. In this task, videos from the respective test sets are split into non overlapping 0.2s segments. For each segment we assign a binary label: positive (action) if the segment overlaps with an annotated action interval and negative (background) otherwise. The goal is to classify each segment as an action or a background. We use average precision (AP) as an evaluation metric for this task.

Action temporal proposal generation. The task is to generate a set of proposals, that overlap well with ground truth action intervals. To generate proposals from the outputs of the network $S(x)$, we use the Temporal Actionness Grouping (TAG) method [Y. Zhao et al. 2017]. We set the Intersection over Union (IoU) threshold for non-maximum suppression to 0.8. We follow the evaluation protocol of [Lin et al. 2018] using the implementation provided in [Lin et al. 2019] to compute Average Recall (AR) at multiple IoU thresholds: $[0.5 : 0.05 : 0.95]$. Finally, we report AR as a function of the Average Number (AN) of proposals per video AR@AN as done in [Lin et al. 2018].

Action step localization. Third task is step localization on CrossTask. Given an ordered list of actions for a video, the goal is to assign each action to exactly one frame. We use

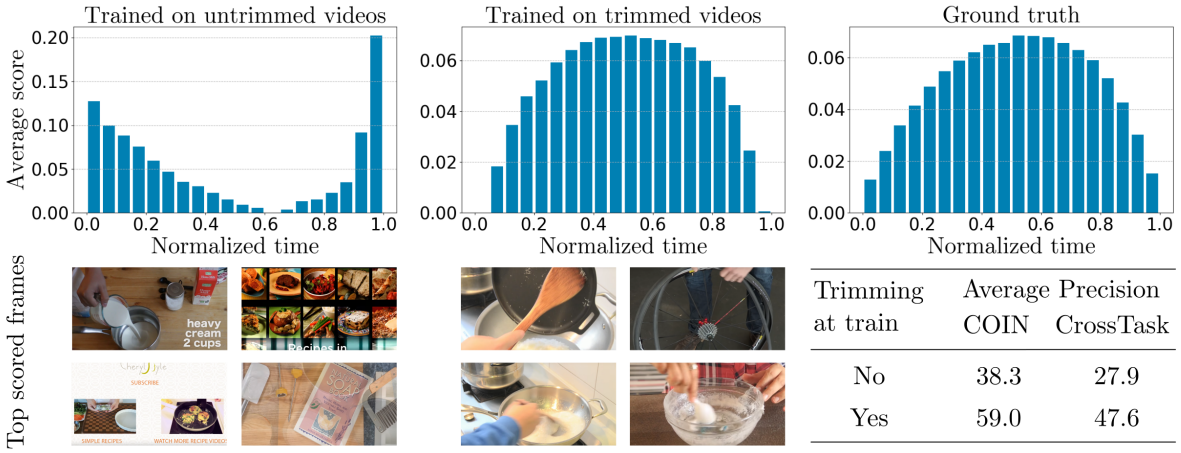
the same evaluation protocol as in [Zhukov et al. 2019] and report average recall.

4.4.2 Ablations studies

This section ablates the important components of our method. We report performance using *background vs action classification* task on COIN and CrossTask.

Video trimming. When first training our method, we notice that the model could learn to be good at the self-supervised ordering task by putting high score on intro and outro segments of videos. This can be intuitively explained by the fact that the beginning and end of instructional videos are distinctive: they start with some typical introduction and often finish with credits. This effect is demonstrated on the left part of Figure 4.3: when trained on untrimmed videos, the actionness score $S(x)$ is high for beginning and end of videos simply because the model can easily discriminate if a frame is from intro (beginning) or outro (end) and hence can safely select these frames to predict relative ordering of pairs. Top scoring frames illustrate that the model picks up on typical credit frames. This led to bad performance as these segments do not contain relevant actions. To alleviate this effect, we employ a simple but effective strategy: during training we trim off 30% of frames in the beginning and 30% of frames in the end of the video. Note that we only do that at training and do not trim test videos to keep the evaluation protocol consistent. Figure 4.3 shows that the temporal distribution of scores that we obtain by this technique better matches the ground truth one, as expected. In particular it's interesting to note that our model is still able to assign frames to actions even for frames that are in the beginning or the end when relevant (*i.e.* this trimming did not handicaped the model for true actions that happen early or late in the video). Finally, the table in Figure 4.3 demonstrates the effectiveness of this approach on our two evaluation datasets.

Long vs. short range order verification. The driving hypothesis of this work is that learning actionness score S is possible thanks to the long range order consistency between actions in instructional videos. We claim that this is not true for short-term ordering between frames as used in previous work [Misra et al. 2016], as in that case order verification can be done via low level visual cues regardless of whether images depict actions or background. We verify that claim by training our model at different

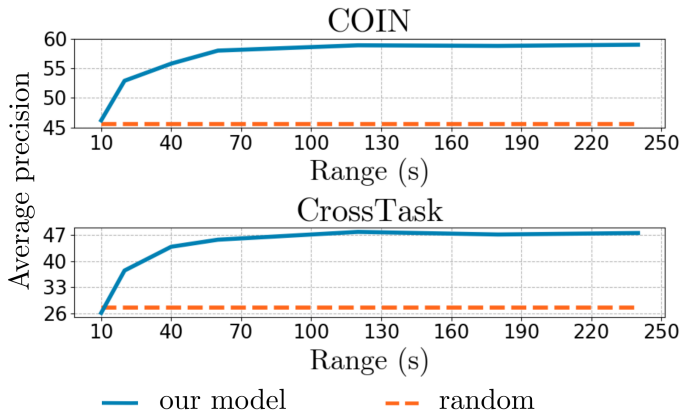


Trimming at train	Average Precision COIN	CrossTask
No	38.3	27.9
Yes	59.0	47.6

Figure 4.3: *Actionness scores at different temporal locations of COIN videos test set.* (**left**) scores of the actionness model S, trained on untrimmed videos. (**center**) scores of the model S, trained on trimmed videos. (**right**) Distribution of actions in the videos. When trained on untrimmed videos, the score concentrates in the beginning and in the end of the video. When trained on untrimmed videos, the score distribution is closer to the ground truth action label distribution which leads to significant increase in background vs. action classification performance.

temporal scales. Formally, we set d to be the temporal window length in which we are going to sample $M = 5$ segments of length $d/10$. In other words, d corresponds to the maximum distance we can have between two clips from X, and is a good measure of the *range* at which we perform the order verification task. Figure 4.4 (left) shows the results for different values of d. For small values of d, the model shows poor performance, compared to larger values. This demonstrates that our method works better on the scale of several minutes (long range), rather than 10-20 seconds (short range).

Number of segments per video. In Figure 4.4 (right), we study the effect on performance when training our model with different number of segments M per video. Results are given for $M \geq 3$ ($M = 2$ makes training of S impossible since there are no pairs to select from). We observe that overall the method is not too sensitive to M. However, only sampling 3 segments per video may often lead to a situation where all selected segments are background, hence selecting larger value for M leads to better results. Figure 4.4 also shows a decline in precision for $M > 5$ on both datasets. This can be explained by the fact that for large values of M there is a high probability of having at least one pair of



Number of segments	Average precision COIN	CrossTask
3	59.4	43.1
4	59.9	46.0
5	59.5	47.2
6	59.0	47.2
7	58.2	46.5
8	57.6	45.6
9	57.1	45.1

Figure 4.4: **Left: long vs. short range.** Action vs. background average precision of our model on COIN (top) and on CrossTask (bottom) as a function of the range d (in seconds) spanned by the sequence X. Duration of each sampled clip equals $d/10$. Performance increases with the range used for the order prediction task, confirming our hypothesis that long range action dependencies are better to learn about actionness. **Right: number of segments.** Average precision of our model on COIN and CrossTask for different number of segments M sampled per video.

segments, for which the temporal order is easy to guess. This may decrease the ability of our model to learn temporal order for other more subtle pairs. Based on that study, we use $M = 5$ in all of our other experiments.

4.4.3 Actionness score for practical applications

Action vs. Background classification. In Table 4.1, we compare against five methods: **(i)** chance baseline, **(ii)** chance baseline with trimming, **(iii)** hand detector, **(iv)** optical flow and **(v)** a supervised model. **(i)** simply assigns random uniform action scores to segments in the video. Following our observation from Section 4.4.2 about trimming, **(ii)** does the same as **(i)** but also assigns to background the segments that occur in the first and last 30% portion of the video. **(iii)** assigns the actionness score to the maximum score of a hand detector [Shan et al. 2020] computed for each frame. This baseline is based on the assumption that the actions correlate with the appearance of hands. **(iv)** assumes that the actions correlate with motion and estimates an actionness score as an average magnitude of optical flow at each frame. Finally for the supervised topline

Table 4.1: **Action vs. Background.** Frame-wise average precision of background separation on COIN and CrossTask datasets.

Dataset	(i) Chance	(ii) Chance (trim@0.3)	(iii) Hand Detector	(iv) Optical Flow	Ours	(v) Supervised
COIN	45.6%	53.5%	50.6%	47%	**59.0%**	70.7%
CrossTask	27.5%	32.6%	28.4%	30.3%	**47.6%**	56.2%

(v), we train in a supervised manner a linear layer on top of our feature representation $h(x)$ for the binary action vs. background classification task. As we also use a linear layer for S, **(v)** provides an upper bound of performance, that can possibly be achieved with our approach. Methods **(ii-iv)** provide simple, yet meaningful baselines in the absence of existing unsupervised methods for the considered task. The results are shown in Table 4.1. Baselines **(ii-iv)** show only a marginal improvement over Chance, which illustrates the difficulty of the task. **(ii)** provides the strongest baseline on both COIN and CrossTask, highlighting the importance of intro and outro segments for action vs. background classification. Our method shows an improvement over the baselines on both datasets.

Figure 4.5 shows the average precision for every domain in the COIN dataset. We compare our results to the chance baseline. Our model outperforms chance for every individual domain. The gap between our method and chance ranges from 9% for Sport to up to 26% for Drink and Snack. This shows that our method allows to separate actions from background in diverse instructional videos.

Temporal Action Proposals. Following the evaluation protocol described in Section 4.4.1, we compare to 6 different methods: **(i)** chance baseline, **(ii)** chance baseline with trimming, **(iii)** hand detector, **(iv)** optical flow, **(v)** supervised and **(vi)** temporal prior. **(i-v)** are the same as for the Action vs. Background classification task. For **(vi)**, we use a temporal prior to generate action segments: it consists in sampling proposal start and length from a prior distribution, obtained from ground truth action intervals. In details, we compute the empirical distribution of the normalized start time of actions as well as their duration. We then randomly sample segments from that distribution by first sampling a start time and then a duration. Note that this baseline has access to more annotation than our proposed approach, since we do not have access to any temporal annotation. Figure 4.6 shows the average recall for CrossTask and COIN, as a

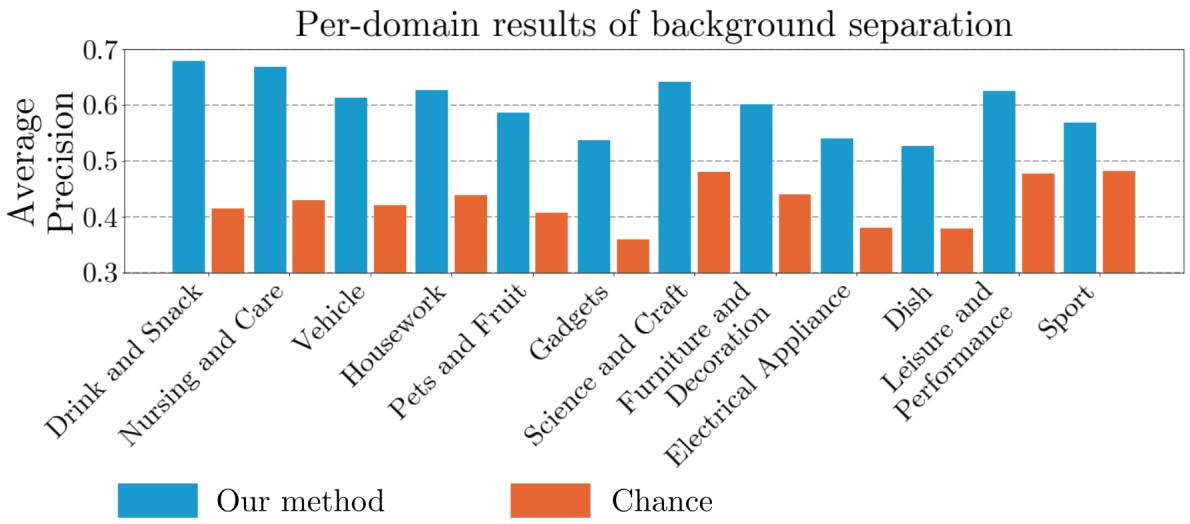


Figure 4.5: Average precision of frame-wise Action vs. Background classification for every COIN domain. Our method outperforms chance in every individual domain.

function of average number of proposals per video (AN). On both datasets we outpeform baselines **(i-iv)** for most values of AN. More interestingly, we also significantly outpeform the temporal prior **(vi)** approach despite using less annotation information. Finally it is worth noting that the gap between our method and the supervised **(v)** topline is not large (less than between our method and **(vi)**).

Step localization. In this experiment, we explore how our actionness model can improve weakly supervised action localization. This task is particularly relevant since presence of background is one of the main challenges for weakly supervised action localization methods. To do so, we augment various action localization methods from [Zhukov et al. 2019], [Miech et al. 2019] and [Miech et al. 2020] with our actionness score $S(x)$ and evaluate on the CrossTask dataset following the protocol described in [Zhukov et al. 2019]. These methods work in a similar way, described next. First, step classifiers are applied to every frame of the video. Then, each step is assigned to exactly one short clip, using a dynamic programming to solve: $(t_1^*, ..., t_K^*) = \arg \max_{t_1 < ... < t_K} \sum_{t=1}^{T} \sum_{k=1}^{K} f_k(x_t)$, where $f_k(x_t)$ is the output of the step classifier k for the t-th input clip x_t, and t_1, ..., t_K are the clips ids assigned to steps 1, ..., K, respectively. [Zhukov et al. 2019], [Miech et al. 2019] and [Miech et al. 2020] differ only in the form of the classifier $f_k(x)$. We augment these

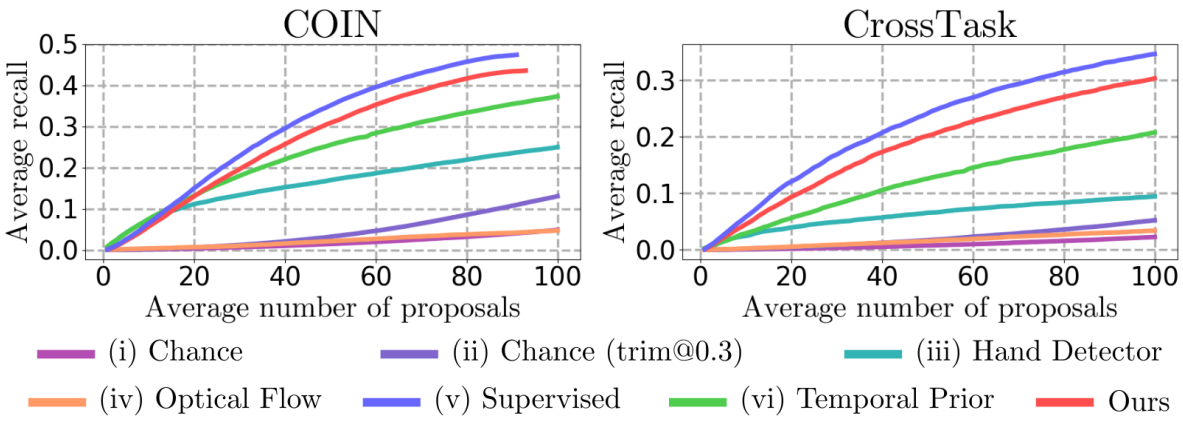


Figure 4.6: **Action proposal.** Average recall versus average number of proposals per video.

methods with our actionness score by simply adding it to the objective during inference:

$$(t_1^*, ..., t_K^*) = \arg \max_{t_1 < ... < t_K} \sum_{t=1}^{T} \left[(1-\alpha) \sum_{k=1}^{K} f_k(x_t) + \alpha S(x_t) \right], \tag{4.4}$$

where $S(x_t)$ is the actionness score of our model for clip x_t and α is a combination parameter. Intuitively, the role of our score is to lower the confidence of background clips and increase the score on foreground action clips.

Results are provided in Table 4.2. α is selected independently for each method, which equals 0.1 for [Miech et al. 2019] and 0.8 for other methods. We use the same value of α for all test tasks. Combining the baseline method with our score improves the performance in every case. The gap in performance is particularly large for the method from Zhukov *et al.* [Zhukov et al. 2019]. This can be in part attributed to the fact that this method does not try to model the background for a given frame (indeed a simple constraint imposes that the score should sum to one across time for all actions without trying to explicitly lower score on detected background frames). Adding our score to the outputs of the model resolves this problem and leads to a large improvement (+4.6% recall). Other methods in Table 4.2 rely instead on a joint video and text embedding pretrained on the large scale HowTo100M dataset to score every segment of the video against the text embeddings of the action description. These text-video embeddings approach lead to much stronger base model. However, given a frame, there is no guarantee that the model will explicitly

	+ **actionness**	Make Kimchi Rice	Pickle Cucumber	Make Banana Ice Cream	Grill Steak	Jack Up Car	Make Jello Shots	Change Tire	Make Lemonade	Add Oil to Car	Make Latte	Build Shelves	Make Taco Salad	Make French Toast	Make Irish Coffee	Make Strawberry Cake	Make Pancakes	Make Meringue	Make Fish Curry	Average
[Zhukov et al. 2019]		13.3	18.0	23.4	23.1	16.9	16.5	30.7	21.6	4.6	19.5	35.3	10.0	32.3	13.8	29.5	37.6	**43.0**	13.3	22.4
[Zhukov et al. 2019]		**18.4**	**24.9**	**25.6**	**24.1**	**19.0**	**29.6**	**33.8**	**30.0**	**7.7**	**23.7**	**45.0**	**13.4**	**36.1**	**23.7**	**34.3**	**41.9**	42.0	**15.8**	**27.0**
[Miech et al. 2019]		33.5	27.1	**36.6**	37.9	24.1	35.6	32.7	35.1	30.7	28.5	43.2	19.8	**34.7**	33.6	**40.4**	**41.6**	**41.9**	**27.4**	33.6
[Miech et al. 2019]		**33.6**	**28.6**	35.4	**38.5**	**25.0**	**37.3**	**35.1**	**41.2**	**30.9**	**30.1**	**45.1**	**21.4**	33.7	**34.3**	39.1	41.2	40.3	26.3	**34.0**
I3D [Miech et al. 2020]		28.7	**37.9**	**42.8**	36.3	22.0	42.9	**27.4**	43.1	30.8	32.7	42.8	27.5	**34.0**	**33.7**	44.3	48.0	**46.0**	33.9	36.4
I3D [Miech et al. 2020]		**31.1**	37.2	42.6	**37.4**	**23.5**	**43.4**	27.1	**43.4**	**32.2**	**35.9**	**46.0**	**29.4**	33.9	36.6	**45.6**	**49.7**	45.2	**37.0**	**37.6**
S3D [Miech et al. 2020]		31.5	36.0	46.5	38.5	25.2	45.0	33.3	**48.1**	**38.4**	**37.0**	48.1	34.2	**38.7**	41.9	**44.6**	**48.2**	**52.2**	**38.0**	40.3
S3D [Miech et al. 2020]		**34.1**	**40.0**	**48.7**	**40.3**	**30.7**	**46.1**	**34.5**	45.9	38.1	35.9	**50.0**	**35.4**	38.1	**42.6**	42.6	45.9	51.6	37.8	**41.0**

Table 4.2: Step localization results on CrossTask with and without our actionness score.

be looking for actions as scores can still be high if its visual content partially matches the description of an action. For example, if the action is *season steak*, the presence of a *steak* and the object *salt* in the frame can increase the similarity score even if the action is not visible. Interestingly, combining these much stronger base models with our score still improves performance by a significant margin (+1.2% recall and +0.7% recall for the best S3D model), hence setting a new state-of-the-art on the CrossTask benchmark [Zhukov et al. 2019].

4.4.4 Qualitative results

To provide more insight about the kind of signal captured by our model S, we show clips from HowTo100M with high and low actionness scores in Figure 4.7. In order to obtain these examples, we run our model on 50,000 randomly sampled videos from HowTo100M. To show the variety of different tasks, we illustrate the top 10 highest scoring clips within each of the four largest HowTo100M categories: Food and Entertaining, Home and Garden, Hobbies and Craft and Cars & Other Vehicles. Finally, we also give the 10 lowest scoring clips across all categories to illustrate the type of background discovered by our model.

Figure 4.8 illustrates the highest scoring COIN clips according to our model that do not intersect with any action segments in COIN annotation, and the lowest scoring COIN clips that lie within action segments in COIN annotation. Interestingly, all the highest scoring background clips in the top row of Figure 4.8 contain valid actions, that are

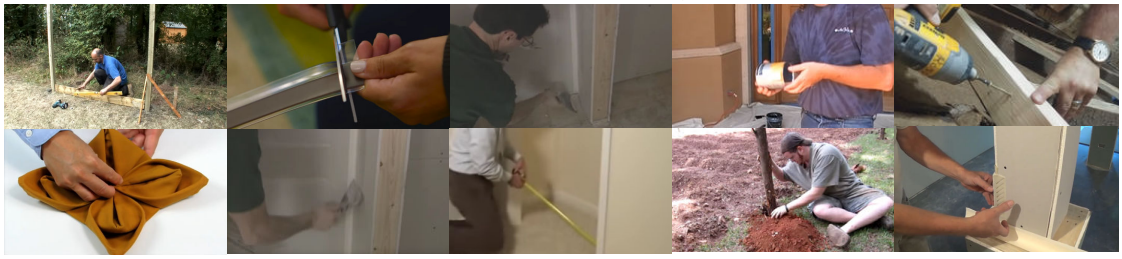
not included in the COIN annotation. Finally, when looking at the clips that score low according to our model but are labeled as action in the COIN dataset (bottom row of Figure 4.8), we see that only two clips actually depict an action: clip 6 (reveal the glue from the face) and clip 8 (scratch the hair carefully). The rest of the clips do not contain any action and are labeled as action due to imprecise annotation of action boundaries.

4.5 Conclusion

In this chapter, we have presented a self-supervised method that can separate actions from background without resorting to any form of manual annotation. It does so by leveraging the assumption that frames that depict key actions are more informative when it comes to predict what may come next or what happens in the past. Equipped with our method, we managed to improve the state-of-the-art on a challenging action localization benchmark. As future work we notably plan to investigate if our method would generalize to broader domains than instructional videos. Another potential direction would be to jointly discriminate background and actions while also clustering similar actions together, thus paving the way for unsupervised action discovery.

Highest scoring clips (Food and Entertainment)

Highest scoring frames (Home and Garden)



Highest scoring frames (Hobbies and Craft)

Highest scoring frames (Cars & Other Vehicles)

Lowest scoring frames (all categories)

Figure 4.7: First four rows show highest actionness scoring clips from the top 4 categories of HowTo100M: Food and Entertaining, Home and Garden, Hobbies and Craft and Cars & Other Vehicles. Bottom row illustrates the lowest scoring clips according to our actionness score S.

Figure 4.8: **Top:** Highest scoring COIN clips, which are not annotated as action. **Bottom:** Lowest scoring COIN clips, which are annotated as action

Chapter 5

Reconstructing and grounding narrated instructional videos in 3D

In Chapters 3 and 4, we have considered methods, that focus on the temporal aspect of the instructional videos, while ignoring spatial information. In this chapter, instead, we focus on the spatial information, contained in the instructional videos. In particular, we propose a method to learn language grounding in the 3D scene from narrated instructional videos in an unsupervised way. We consider instructional videos that share a common underlying 3D scene, and assume that the locations associated with key objects and actions are the same across videos. Our approach consist of two principal steps. First, we produce a common 3D reconstruction and register all videos to it. In order to reduce the computational complexity of this task, we, first, produce a separate 3D reconstruction for each video, using COLMAP [Schonberger et al. 2016], and then align the obtained reconstructions in the 3D space. Second, we propose a model that grounds language in the obtained 3D reconstruction, by associating text input with a specific voxel in a discretized 3D space. We train this model in an unsupervised way, by using video narrations, coupled with a supervisory signal, automatically obtained from the video. We consider three ways to build a such supervisory signal, based on simple assumptions, such as photographer's bias (*i.e.* camera is focused on the object of narration), correlation between hand position and the object of narration, as well as local joint text-video embedding from [Miech et al.

2020]. In order to validate our approach, we collect a set of car maintenance videos for 6 different car models. Each video focuses on car engine area, demonstrating tasks, such as "changing car battery" and "measuring oil level". Our choice of videos is justified by the fact that all videos for a specific car model share near identical scene (the engine bay area). We evaluate our method on this data and demonstrate, that it grounds textual instructions in associated parts of the 3D reconstruction.

5.1 Introduction

Imagine you need to perform a task on a product you own, such as replacing the drum unit in your laser printer or re-filling the coolant fluid tank in your car, but you do not know how to proceed. In the past few years, instructional videos have become a popular internet resource for learning how to perform such kind of tasks. However, finding the answer to this particular problem would require searching for the appropriate video, identifying the key steps, and finally, putting it into practice by performing the right sequence of actions involving the correct object parts at the right locations. Wouldn't it be great to have a virtual assistant guide you through the steps in augmented reality showing you *which* tasks to perform and *where* to perform them? In order to produce such an assistant, a method for relating natural language to the actual 3D objects in the world is needed. In this chapter we show that it is possible to obtain 3D reconstructions and to learn associations between natural language and corresponding 3D locations (3D language grounding) using raw instructional videos and no manual annotations.

As we have seen in the previous chapters, instructional videos have proven to be a useful source of data for learning to temporally localize a set of actions in a video strip, given the set of action labels (i.e. weakly-supervised temporal action localization) [Alayrac et al. 2018; Elhamifar et al. 2020; Huang et al. 2016; Kukleva et al. 2019; Malmaud et al. 2015; Richard et al. 2017, 2018a; F. Sener et al. 2018; Zhukov et al. 2019]. This is useful for downstream tasks such as video summarization or video retrieval. However, with the advent of smartphones and headsets equipped with 3D sensors the problem of spatial action localization in 3D gains relevance [H. Chen et al. 2015; Jonghyun Kim et al. 2019; Martín-Gutiérrez et al. 2017; Pratt et al. 2018]. While raw video data is

Figure 5.1: Illustration of the proposed method. Given a set of instructional videos showing the same object (e.g. same car model), we propose a two-step 3D reconstruction approach where, first, each video is independently reconstructed (R) and, second, reconstructions are aligned (A) in a single model. Using the reconstructed 3D model together with the narrations from each individual video and no manual supervision, we learn a language grounding model (G) that can determine the corresponding 3D region for a given text query such as "air filter".

useful to learn visual representations of actions, it is difficult to directly learn about 3D structure from 2D frame data. Therefore, in order to solve the problem of spatial action localization in 3D, a suitable 3D spatial representation of the object and the scene are necessary. However, producing a 3D reconstruction from a set of narrated instructional videos presents several challenges.

In recent years, significant progress has been made in the field of 3D reconstruction. Structure-from-motion (SfM) pipelines [Moulon et al. 2012, 2016; Schonberger et al. 2016; Sweeney n.d.; Wu 2013] leverage robust local features for obtaining correspondences between images despite large illumination and viewpoint changes. This has allowed to perform 3D reconstruction from unstructured photo-collections from the web, where images may be taken months or years apart [Agarwal et al. 2011; Frahm et al. 2010; Snavely et al. 2006]. However, the task of reconstructing a set of narrated instructional videos poses some additional challenges. First, besides the large illumination and viewpoint changes, these videos may contain significant appearance changes. This is due to the fact that, while the objects that appear in different videos belong to the same brand and model, they might present variations due to different options or sub-versions of the same product. Therefore, obtaining correspondences between such videos is an intermediate problem between traditional instance-level matching (the *same* physical object is imaged) [Jégou et al. 2008; Lowe 1999; Snavely et al. 2006] and category-level matching (semantically related objects are imaged, such as two different species of dogs) [Ham et al. 2016; Min et al. 2019; Rocco et al. 2018a,b]. Despite the robustness of the local descriptors used in current SfM pipelines, these are not designed to handle such strong appearance changes and therefore are prone to failure. As a second challenge, because the computational cost of 3D reconstruction grows quadratically with the number of total frames, jointly reconstructing a set of many videos may be unfeasible, or may impose a strong limit in the number of frames to be sampled from each video.

In this chapter we address the above difficulties and propose a method that is (i) capable of finding correspondences across instructional videos despite the large appearance changes, and (ii) reduces the global computational complexity of 3D reconstruction by reconstructing each video independently and then combining the results. Given the obtained 3D reconstructions and corresponding video narrations, we next learn 3D language

grounding. Our approach to 3D language grounding leverages both narrations and visual information in instructional videos to associate the input text to different parts of the 3D scene, and doesn't require manual supervision (see Figure 5.1).

Contributions. In summary, our contributions are the following. **I.** We develop an approach for correspondence estimation that combines the strengths of local features and dense flow in order to obtain reliable matches across instructional videos. **II.** We propose a two-step 3D reconstruction approach that reduces the overall computational complexity by reconstructing each video independently, and then combining the results into a 3D alignment graph as a proxy to a single aligned 3D reconstruction. We evaluate obtained reconstructions on the task of keypoint transfer. **III.** We propose an unsupervised method, that makes use of the obtained 3D models and narrations in instructional videos to learn language grounding in 3D.

5.2 Related work

3D reconstruction. The standard approach to obtain a 3D reconstruction from a set of images is to employ an incremental or global structure-from-motion pipeline such as COLMAP [Schonberger et al. 2016], Theia [Sweeney n.d.], VisualSfM [Wu 2013] or OpenMVG [Moulon et al. 2016]. In all cases, the first step is to find correspondences between the set of input images. This is typically done using local features such as SIFT [Lowe 1999], which are robust to moderate changes in illumination and viewpoint present in instance-level 3D reconstruction. In this chapter, we propose a different 3D reconstruction approach targeted to the reconstruction of narrated instructional videos, which have the additional challenge of large appearance changes due to differences in sub-versions of the same object type or due to different conditions of conservation (wear, cleanliness, etc.)

Learning from instructional videos. Learning from instructional videos has been recently addressed in the context of problems such as action discovery [Alayrac et al. 2018; Elhamifar et al. 2020; Kukleva et al. 2019; Richard et al. 2018a; F. Sener et al.

2018] and localization [Huang et al. 2016; Richard et al. 2017; L. Zhou et al. 2018a; Zhukov et al. 2019], modeling of object states [Alayrac et al. 2017] and visual reference resolution [Huang et al. 2018, 2017]. A particular attention has been drawn to the *narrated* instructional videos, *i.e.* instructional videos, accompanied by comments in the form of natural language. It was shown, that narrations in such videos generally describe the visual content and can be used as an additional guidance for learning the visual models [Alayrac et al. 2018; Malmaud et al. 2015; Zhukov et al. 2019], or for joint modeling of video and language [Miech et al. 2020, 2019; Zhu et al. 2020]. In this chapter, we make use of narration for object grounding in 3D, assuming that object mentions in narration coincide with their appearance on the screen.

Related to our work, Damen et al., [Damen et al. 2014] consider the problem of task-relevant object discovery from videos. Unlike [Damen et al. 2014] we do not aim for a discovery of objects, and only attempt to localize objects in the 3D scene. On the other hand we consider a more general setup: our videos depict different object instances in uncontrolled scenarios and are sourced from a mix of egocentric and third-person cameras.

Visual language grounding. Visual language grounding requires associations between the text the most relevant parts of an image. A standard approach to this task involves learning a joint embedding of language and visual data and comparing the similarity between the text and image regions in the joint embedding space [Karpathy et al. 2014a; Plummer et al. 2018; Liwei Wang et al. 2019, 2016; M. Wang et al. 2016]. Most works focus on learning to ground language in images in supervised settings, where the regions of images, relevant to the input text are provided at training time. On the contrary, [Engilberge et al. 2018; Xiao et al. 2017; F. Zhao et al. 2018] consider weakly-supervised scenario where only the correspondences between text and images are provided during training, while the exact regions, where the text is grounded, are unknown.

Closer to our work, Russell *et al.* [Russell et al. 2013] attempt to localize text from Wikipedia articles in a 3D scene. This work focuses on museum scenes and the objects, such as paintings and frescos. This is done in two steps. First, an object caption is used as a query to search for related images with Google image search. Retrieved images are then registered to the 3D scene and their extents are used as a clue for object localization

in 3D. The authors of this work leverage the *photographer's bias*, *i.e.* the fact that objects of interest are usually perfectly framed by the extents of the image. Our solution for the 3D text grounding is based on a similar observation. However, we focus on instructional videos and unlike [Russell et al. 2013], we use the same video data both for reconstruction and for language grounding, and do not rely on query expansion via external image search.

5.3 3D reconstruction of instructional videos

In this section we present our approach for reconstructing narrated instructional videos in 3D. In order to address the limitations of the standard instance-level 3D reconstruction pipeline (i.e. limited robustness to appearance changes and large computational complexity) we propose an approach based on the *divide and conquer* strategy, where each video is reconstructed independently and then the resulting 3D models are combined into a 3D alignment graph that allows to obtain a final consistent model. In this way we have two advantages over the joint reconstruction approach. First, only a few correspondences are needed *across* different videos, in order to estimate the similarity transformation that aligns the 3D models reconstructed from each video. Second, the computational complexity of the 3D reconstruction is reduced from $O(n^2N^2)$ to $O(N^2)$, where N is the number of frames sampled from each video, and n is the number of videos. In the following sections, we describe in detail our approach for reconstructing instructional videos in 3D. An overview of the approach is presented in Figure 5.2.

5.3.1 Per-video reconstruction

In order to avoid the large computational complexity and matching challenges of joint 3D reconstruction, we reconstruct each video independently. For each video, we proceed in the following way. First, N video frames $\{v_i\}_{i=1,\dots,N}$ are randomly sampled. Then, CNN-based local features $(f_i, p_i) = L(v_i)$ are extracted for every frame using the feature transform L, where $f_i \in \mathbb{R}^{d_L \times N_L}$ is the set of d_L-dimensional feature descriptors, $p_i \in \mathbb{R}^{2 \times N_L}$ is the set of 2D key-point positions, and N_L is the number of extracted local features. Then, exhaustive matching is performed between the features of every video frame v_i. Only matches that are mutually nearest-neighbors are retained. Finally, these

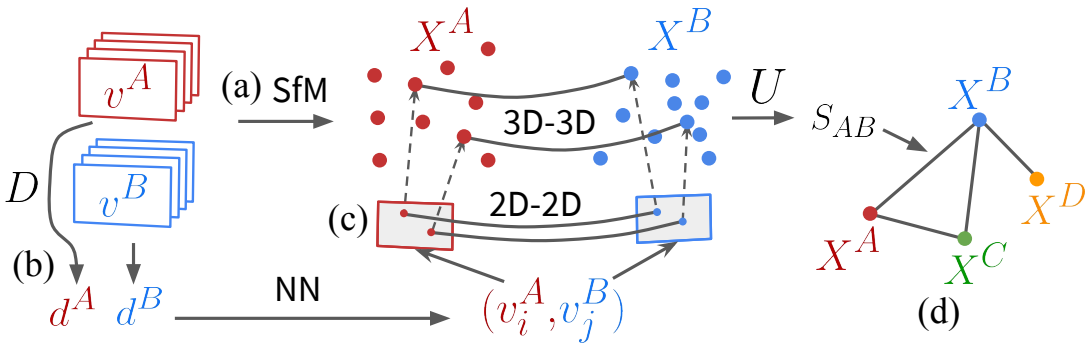


Figure 5.2: Proposed approach for reconstructing instructional videos in 3D. (a) Given two videos, v^A and v^B, each is independently reconstructed, producing the 3D models X^A and X^B. (b) Furthemore, image-level descriptors d^A and d^B are extracted with D, and used to obtain candidate pairs of frames (v_i^A, v_j^B). (c) Then, 2D-2D correspondences between these pairs of frames are obtained by matching local features with global constraints imposed by dense flow estimation, and 3D-3D correspondences are established from the 2D-2D correspondences by leveraging the individually reconstructed 3D models, X^A and X^B, and the camera parameters (extrinsics and intrinsics) for v_i^A and v_j^B. (d) The aggregated 3D-3D correspondences from several image pairs are passed to the solver U that robustly estimates a similarity transformation. Finally, a graph is constructed where each node corresponds to a 3D model, and each edge corresponds to a successful alignment between a pair of point-clouds. By traversing this graph, and composing the transformations along a given path, one can compute the similarities between any pair of connected nodes.

matches are input into an SfM pipeline to produce a 3D model represented as a cloud of 3D points $X = \{(X_i, Y_i, Z_i)\}_{i=1,\ldots,N_p}$ for the given set of video frames[1]. An illustration of this approach is presented in Figure 5.2a.

5.3.2 Formation of frame-pairs across videos

Once every video is independently reconstructed, the goal is to align them to a common frame of reference, which involves estimating a 3D similarity transformation. Then, given a pair of videos (v^A, v^B), the first step involves estimating a candidate set of matching video frames in the form of $M = \{(v_i^A, v_j^B)\}$, where i and j are the frame indices of matching frames of v^A and v^B respectively. In order to achieve this, we follow the approach that is typically used for image retrieval. This consists of, first, extracting image-level descriptors $d_i^A = D(v_i^A)$ and $d_j^B = D(v_j^B)$ using the image-level description

[1]In some cases, several 3D models will be produced due to the fact that videos might have edits, and show completely different scenes at different times. Small models are discarded.

function D for every frame v_i^A in v^A and v_j^B in v^B, where $d_i^A, d_j^B \in \mathbb{R}^{d_I}$ are d_I-dimensional descriptors. Then, the top N_M matches between videos v^A and v^B are selected by nearest-neighbor search between the sets of image level descriptors $\{d_i^A\}$ and $\{d_j^B\}$. An illustration if this approach is presented in Figure 5.2b.

5.3.3 Establishing 2D correspondences between videos

Once candidate matching frames $\{(v_i^A, v_j^B)\}$ between a pair of videos (v^A, v^B) are established, the next step consists of robustly estimating 2D-2D correspondences between them. In order to handle the large appearance differences that may be present in these two frames, we propose to use a combination of learnt local features and dense flow to robustly obtain matches that are simultaneously well localized and compliant with a global transformation. Given a pair of frames (v_i^A, v_j^B), mutual matches between the sets of CNN-based local features (f_i^A, p_i^A) and (f_j^B, p_j^B) are first computed. Using local features allows for a very precise localization of the matches, but does not enforce any global consistency among different matches. In practice, this may lead to a high fraction of incorrect correspondences that may cause the subsequent alignment estimation step to fail. In order to reduce the fraction of incorrect correspondences while retaining the correct correspondences, we filter the matches produced by local features using a dense transformation field $\mathcal{T}_{ij}^{AB} = G(v_i^A, v_j^B)$ obtained from a dense flow model G, where $\mathcal{T}_{ij}^{AB} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ is a 2D-2D mapping between the pixels of v_i^A and those of v_j^B, and G is a CNN-based dense flow estimation model. Only matches between local features that are consistent with the dense flow mapping $\mathcal{T}_{ij}^{AB}$ up to certain tolerance threshold t are retained. An illustration if this approach is presented in Figure 5.2c.

5.3.4 Estimating 3D transformation between videos

Once 2D-2D correspondences between frames of different videos are established, these can be used to, first, establish 3D-3D correspondences, and, second, determine a 3D similarity transformation between these videos.

Given a pair of videos (v^A, v^B), 2D-2D correspondences are first aggregated from all its

matching frames $\{(v_i^A, v_j^B)\}$. Then, these 2D-2D correspondences are converted to 3D-3D correspondences by leveraging the camera parameters (intrinsic and extrinsic) together with the 3D models X^A and X^B obtained from their individual 3D reconstructions (cf. Section 5.3.1). Then, the set of 3D-3D correspondences is input to a solver U that robustly estimates the similarity transformation S_{AB} between 3D models X^A and X^B. However, some models may be difficult to align directly due to very large viewpoint changes, which may cause the matching method presented in Section 5.3.3 to fail. To overcome this issue, we propose to construct a *3D alignment graph* where every node represents a 3D model X^m and every edge represents a successful alignment between the 3D models of the two nodes that constitute the edge. Then, the alignment between any two connected models in the graph can be estimated by computing the shortest path [Dijkstra et al. 1959] and then composing the pairwise transformations along the path. Note that by traversing the graph it is possible to align 3D models across large changes of viewpoint or appearance which would not be possible by direct matching. In order to register the models to a common reference frame, any node can be selected as reference and the similarities to every other node computed by traversing the graph. An illustration if this approach is presented in Figure 5.2d. Figure 5.3 demonstrates how the 3D alignment graph is built and used.

5.3.5 Application to keypoint transfer

In this section we describe how the proposed 3D alignment graph can be used to perform keypoint transfer among models. Because manual annotation is costly, annotation transfer has been used in the past as a way to obtain annotation on new examples in an automatic way [Guillaumin et al. 2014; S. Kim et al. 2017; X. Wang et al. 2019]. In our case, we consider the problem of transferring annotations across several instructional videos showing the same object model. In particular, consider that we have a source video v^s, and a set of target videos $\{v^{t_1}, v^{t_2}, \ldots, v^{t_{N_t}}\}$, where N_t is the number of target videos. Consider as well that we are interested in a set of N_k keypoints. Then, we manually annotate these N_k keypoints on several frames of the source video v^s and triangulate their 3D positions $K^s \in \mathbb{R}^{3\times N_k}$ in the source video reference frame, where the column of K^s correspond to the 3D coordinates of each different keypoint. Then, we can apply

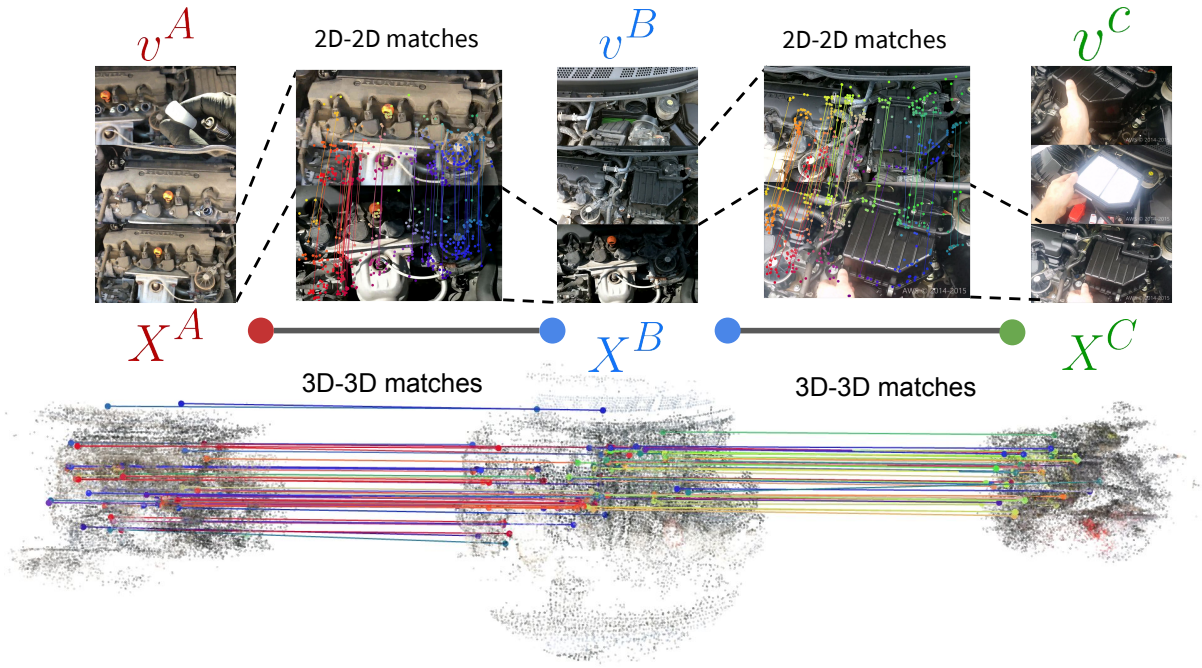


Figure 5.3: Illustration of the proposed alignment method and the 3D alignment graph. Given two videos v^A and v^B, pairs of matching frames are found using global image descriptors, and are then matched using our approach to obtain 2D-2D matches (top). These 2D-2D matches are then translated into 3D-3D matches between the 3D point clouds X^A and X^B corresponding to v^A and v^B, respectively (bottom). These 3D-3D matches are used to estimate the similarity transformation S_{AB} between X^A and X^B, using the function U. Furthermore, the constructed 3D alignment graph can be used to align two videos v^A and v^C which have little overlap (i.e. they focus on different parts of the object), by traversing the edges of the graph using one or more intermediate videos and 3D models (v^B and X^B in this example).

the similarity transformation S_{s,t_i} between the source video v^s and each target video v^{t_i} (obtained by traversing the 3D alignment graph) to transfer the keypoints K^s to each target video v^{t_j}:

$$K^{t_i} = S_{s,t_i} K^s \tag{5.1}$$

Finally, these transferred 3D keypoint annotations can be projected to any 2D frame $v_j^{t_i}$ of the target video v^{t_i} by using the estimated projection matrix $P_j^{t_i}$ (obtained from the product of the intrinsic and extrinsic camera parameters).

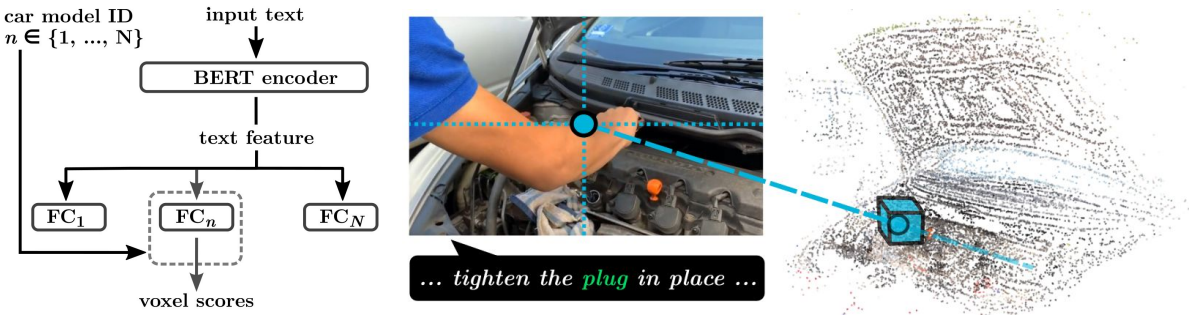


Figure 5.4: **Left.** Illustration of our 3D text grounding model. We train a separate model for each car make. All models share the same text encoder (BERT) and differ in the last grounding layer. The grounding layer is a single fully connected layer with the output dimension equal to the number of voxels. Given the input text, the model produces the confidence scores for all voxels in the 3D model. **Middle and Right.** Illustration of our training procedure. Given a word sequence from subtitles (bottom), and a 2D point in the frame coordinates (blue dot), we backproject the point onto the surface of the 3D model and find the corresponding voxel (blue cube). The pair (text, voxel) is then used as a training pair. In this example, we use the center of the frame as the approximate supervision to ground the text narration on the 3D model.

5.4 Text grounding in 3D

Given a 3D scene reconstruction, our goal is to obtain 3D text grounding, i.e., to identify the parts of the 3D model corresponding to a given text query. We follow [Alayrac et al. 2016; Miech et al. 2019] and assume video narrations to be approximately synchronized with video frames of corresponding instructional videos and to describe their visual content. We consider alternative methods for localizing narrations in 2D frames and then backproject such location onto the 3D model. While the individual backprojected 3D positions may be noisy, e.g. because of the misalignment of the narration and the video, we obtain accurate positions by accumulating evidence in 3D from multiple videos.

Our approach is summarized in Figure 5.4. We first employ a text encoder model B, that produces a $d-$dimensional text feature descriptor $f_s = B(s)$ from a given input text string s. We then use a classifier C to predict the distribution of scores $c = C(f_s) \in \mathbb{R}^n$ over a set of N_V voxels that discretize the 3D model space, for the given text features f_s. In the following, we describe our approach for training such classifier and text encoder in an unsupervised way from narrated instructional videos.

5.4.1 Generation of training data

We propose to automatically generate training samples for learning text grounding in 3D scenes. Our objective is to obtain training pairs (s_i, k_i) formed by the text s_i and the corresponding location in the scene represented by the 3D voxel label $k_i \in [0, N_V]$. Towards this goal we first split an instructional video v and its corresponding transcribed narration s into a set of temporal segments $\{(v_i, s_i)\}_{i=1,\dots,n}$. To obtain k_i for each segment, we first select a 2D position $p_i = (x_i, y_i)$ in video v_i, and then use the camera parameters (intrinsics and extrinsics) together with the reconstructed 3D model to obtain the corresponding 3D point P_i of the scene. We build a pair (s_i, k_i) by choosing a 3D voxel k_i containing P_i.

To obtain 2D image locations p_i, we ground narrations s_i in video frames by following three alternative approaches.

Center of frame. The first approach is to assign p_i to the frame center. This simple strategy turns out to be effective in instructional videos where narrated objects and actions are typically centered in the frame by the cameraman.

Hand detector. Our second approach follows the assumption that narrated objects are being manipulated by the user. We hence assign $p_i = H(v_i)$, where H is a position of a hand in the video obtained with a hand detector [Shan et al. 2020].

MIL-NCE. Our third approach builds on a pre-trained model for local text-video similarity. We use a joint text-video embedding model trained on a large set of narrated instructional videos with MIL-NCE [Miech et al. 2020]. This model projects video and text to a joint embedding space, which allows us to compute similarity between a local spatio-temporal video region and a text fragment. To ground the text narration within the frame, we select a position in a feature map with the highest text-video similarity score.

In all three cases described above, the selected 2D locations p_i within frames are reprojected to the surface of the reconstructed 3D model to obtain P_i and corresponding voxels k_i.

5.4.2 Learning 3D grounding model

Given a set of training pairs with narrations and corresponding voxels $\{(s_i, k_i)\}_{i=1,\ldots,n}$, the classifier C is trained to predict the correct voxel label k_i given input text s_i. We train C by minimizing the cross-entropy loss over all training pairs:

$$-\sum_{i=1}^{n} \log\left(C(B(s_i))[k_i]\right), \tag{5.2}$$

where $C(B(X))[k]$ is the score of voxel k, given input text X, according to our model C. This loss is minimized when C assigns the maximum score of one to the correct voxel.

In practice, we are interested in training language grounding models for different 3D reconstructions. For example, for car maintenance videos, we build a separate 3D reconstruction for each specific car model, such as Ford Focus or Honda Civic, as the 3D shape of the car and the engine can be substantially different. However, while the classifier C is different for all 3D reconstructions, we choose to share the same text encoder B for all reconstructions. This allows us to benefit from all available text data, when training the text representation module. The entire model trained in a multi-task setting is shown in Figure 5.4. Additional implementation details are provided in Section 5.5.

5.5 Experimental results

To evaluate our proposed method for 3D reconstruction and 3D grounding, we collect a dataset of narrated instructional videos related to car maintenance tasks. The tasks we consider are the following: replacing the air filter, changing the spark plugs, replacing the car battery, measuring the oil level, adding coolant fluid and checking the break fluid. The car models we consider are: Honda Civic (8th gen), Honda Accord (7th gen), Ford Focus (Mk3), Ford Explorer (5th gen), Toyota Corolla (E140) and Toyota Prius (XW20). We select these car models as they are common in North America. We also focus our data collection on English-spoken videos. We have collected a total of 174 videos, with the number of videos per car model between 23 and 36. The number of videos per task varies between 14 (measure oil level) and 53 (change spark plugs).

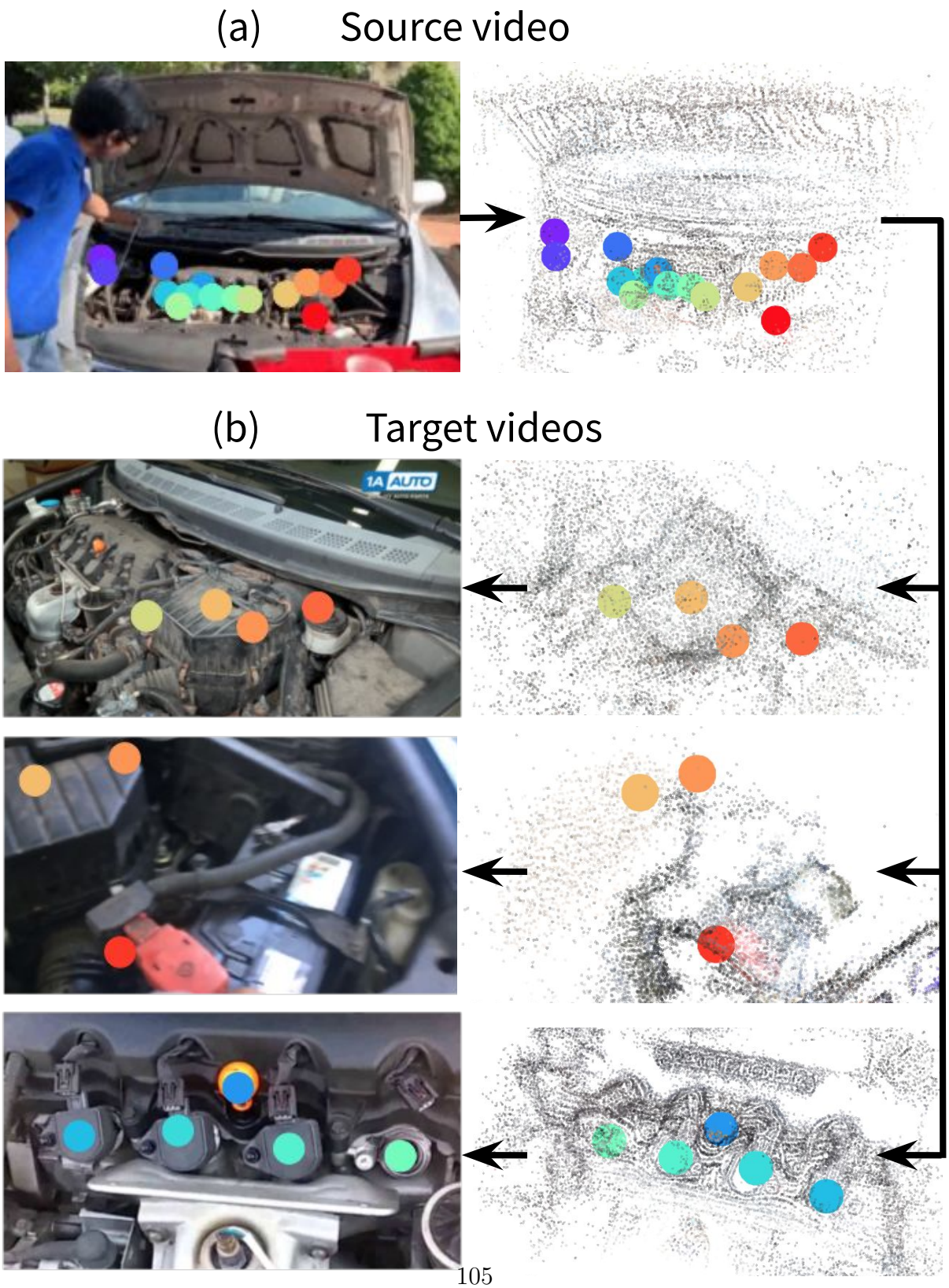


Figure 5.5: Example of keypoint transfer results. (a) 2D keypoints are annotated in several frames of the source video and triangulated to 3D keypoints in the 3D reference frame of source video. (b) These source keypoints are automatically transferred to the 3D reference frames of several target videos using the proposed alignment graph. The 3D keypoints can be finally projected to 2D keypoints for every registered frame in the target videos.

In the following we first describe implementation details and then present results obtained for the keypoint transfer and 3D text grounding tasks. Our dataset, pre-trained models and code will become publicly available.

Implementation details. We use R2D2 [Revaud et al. 2019] learnt local features as the feature extraction function L and GLU-Net [Truong et al. 2020] as the learnt dense flow estimation function G. The image level description-function D is done using a NetVLAD [Arandjelovic et al. 2016] model with a VGG16 backbone [Simonyan et al. 2015]. The solver U is composed by a RANSAC loop [Fischler et al. 1981] that performs a transformation fitting using the least-squares method proposed by Umeyama [Umeyama 1991].

For the text grounding, we use a pre-trained BERT model as the text encoder B. The output dimension d of the text encoder is set to 1024. A linear classifier is used for the grounding layer C, where the output dimensionality corresponds to the number N_v of voxels. The text enconder B is finetuned together with the grounding classifier C in an end-to-end manner. For the hand detector H, we use [Shan et al. 2020]. When using MIL-NCE [Miech et al. 2020] we obtain local similarity scores for any location of the frame by removing the last average pooling layer from the S3D backbone of the visual branch of the model and obtain a features map of size 14×8 for each selected frame of the video.

We define a set of voxels as follows. First, consider a rectangular region, that encompasses the 3D model. This region is divided into 20 equal parts by each axis, resulting in a set of 8000 voxels. Since the shape of a car engine is relatively flat, we may expect that most of these voxels do not intersect with any parts of the model. In order to reduce the total number of voxels, and only retain the voxels, that intersect with objects of interest in the 3D scene, we select top N_v voxels, that contain most of the training points. In practice N_v is set to 500.

5.5.1 Keypoint transfer

For evaluating the task of keypoint transfer we annotate multiple frames for each Honda Civic video. We select one video as source (v^s), and the rest as target ($\{v^{t_i}\}_{i=1,\dots,N_t}$). Then, the annotations are used to triangulate the 3D keypoint positions K^s for the source video and for each of the target videos K^t_i, where $i = 1, \dots, N_t$. Then, we estimate a ground-truth similarity transformation between the source video and the target videos by fitting a similarity transformation using the solver U on the triangulated keypoints:

$$S^{GT}_{s,t_i} = U(K^s, K^{t_i}). \tag{5.3}$$

Finally, we can assess the quality of the similarity transformations S_{s,t_i} that we estimated using the proposed 3D alignment graph by computing the percentage of correctly transferred keypoints (PCK) in 3D coordinates. For evaluation, we use 89 pairs of videos for the Honda Civic car model, where the source and target keypoints K^s and K^t have been verified to have consistent relative distances and angles (a consistent overall 3D shape). Our results are presented in Figure 5.6, where we plot the mean PCK over these 89 evaluation pairs vs. a varying tolerance threshold calibrated in centimeters. Results show that our proposed matching approach using a combination of R2D2 learnt local features and GLU-Net dense flow is superior to these two methods employed individually. Nevertheless, some video pairs are very difficult to match directly due to extreme viewpoint changes and our direct matching approach can only align about 50% of the keypoints for a 10cm tolerance value. In this cases the 3D alignment graph can be used to compute the similarity transformation via other intermediate videos. Results show that by using the 3D alignment graph, 80% of keypoints are correctly localized within a tolerance of 8cm. Examples of keypoint transfer using the 3D alignment graph are shown in Figure 5.5 and, thanks to the 3D alignment graph, demonstrate transfer across an impressive range of viewpoints and appearance variation across the different car instances.

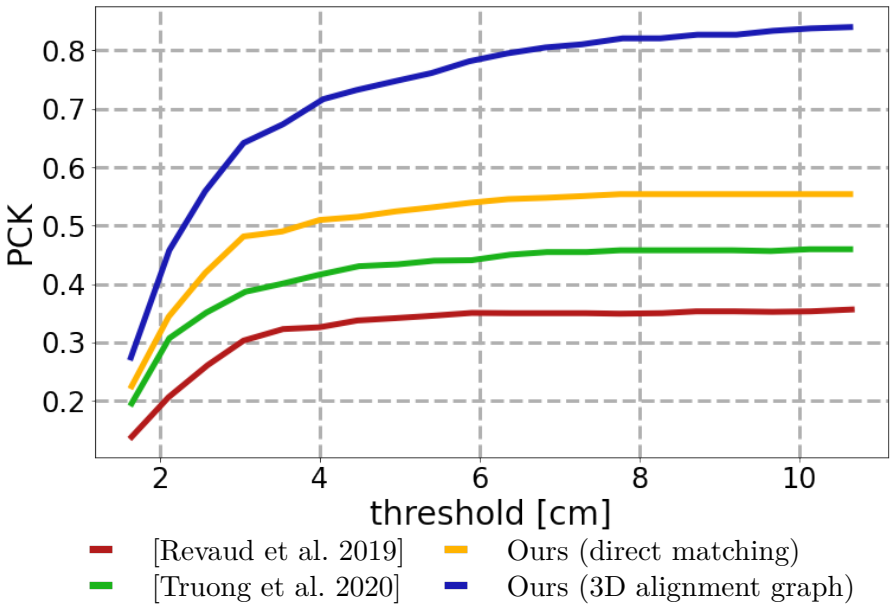


Figure 5.6: Quantitative evaluation of keypoint transfer. We plot the mean PCK curve over a set of 89 video pairs. The proposed 3D alignment graph correctly aligns 80% of keypoints within a 8cm tolerance.

5.5.2 Text grounding in 3D

Evaluation setup. To demonstrate the effectiveness of 3D text grounding, we select six different objects: *air filter*, *brake fluid reservoir*, *negative battery cable*, *positive battery cable*, *dipstick* and *spark plugs*, and annotate their positions within each 3D model. Each object is annotated with a single 3D point. We manually parse the WikiHow.com car repair articles and select a set of textual instructions, related to each object. These instructions are object-centric, each refering to a specific location within the engine, where one of the objects is located. Examples of such instructions include "remove the dipstick" (*dipstick*), "use a wrench to loosen the positive cable clamp and take the cable off of the terminal" (*positive battery cable*) and "lift the filter out of the housing" *(air filter*). At test time, we use these instructions as queries, with the goal to ground each query to the correct 3D location, defined by the corresponding point of interest. Since our model returns the scores over a set of voxels, instead of a single 3D point, we define the grounding by the model as the center of a highest scoring voxel.

Evaluation metric. We measure the performance of the grounding by the PCK score. This score is computed as follows. For each car model and for each query, we compute the distance between the predicted 3D point and the ground truth 3D point that corresponds to the object within the query. Then for each threshold value from 1cm to 100cm we compute PCK as the number of queries grounded correctly (*i.e.* the distance between the predicted point and the ground truth point is below the given threshold value), divided by a total amount of queries for all models.

Results and discussion. We evaluate our models, trained with three different types of supervision, as described in Section 5.4, and compare their performance against a chance baseline ("Chance"), which implies grounding each query to a random point in the 3D model. Results are shown in Figure 5.7. Interestingly, the best results are provided by the center of frame method ("Center of frame") confirming the previously observed photographer bias [Russell et al. 2013]. Supervision from the Hand detector ("Hand detector") suffers from a large amount of noise when the hands are visible in the video but are not near the object. The hand detector also provides less supervision than other methods, since hands may not appear within the frame at the time when the object is mentioned by the narration. Finally, in many cases, the appearance of hands in the videos coincide with an object mention in narration, when an action over the object is being performed. However, an action often involves using tools, such as a wrench or a screwdriver. Therefore, the position of the hand is often shifted with respect to the object, contributing to the noise in the supervision. We found MIL-NCE feature ("MIL-NCE") not well suited for the task. While, in some cases, this feature is able to provide good localization, it is usually very coarse, as MIL-NCE is not designed to accurately localize queries within the image. Results for different object classes are shown in Table 5.1. These results indicate that our model (here the best performing variant "Center of frame" is used) consistently outperforms the chance baseline for 5 object classes out of 6, and only fails on the dipstick object. Figure 5.8 demonstrates qualitative results for different queries and different car models.

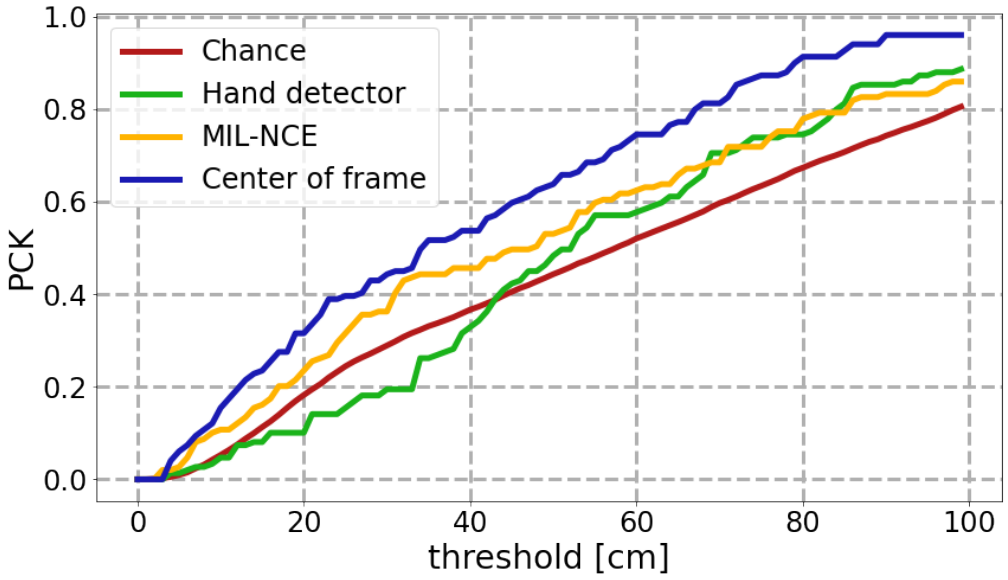


Figure 5.7: Quantitative evaluation of 3D text grounding. We plot PCK curve over a set of 6 different car models and 6 different objects of interest. For each object we consider a set of text queries, related to the corresponding location in the 3D model, and predict the 3D location of the object based only on the the language query. We compare our model trained with different types of supervision against a Chance baseline, where each query is grounded in a random point on the 3D model.

Object	Air filter	Brake fluid reservoir	Negative battery cable	Positive battery cable	Dipstick	Spark Plugs	Average
Chance	0.26	0.19	0.07	0.05	0.40	0.54	0.29
Our method	**0.55**	**0.23**	**0.42**	**0.20**	0.35	**0.69**	**0.41**

Table 5.1: PCK grouped by object classes. We compare our model, trained on centers of frames, against the Chance baseline. Here the PCK error threshold is set to 30cm.

5.6 Conclusion

In this chapter, we have presented a method for 3D reconstruction of instructional videos and localizing the associated narrations in 3D. Our method is resistant to the differences in appearance of objects depicted in the videos and computationally efficient. Building on our 3D reconstruction, we propose an unsupervised approach to 3D language grounding. Our work opens-up the possibility of providing visual hints, based on textual instructions for augmented reality applications.

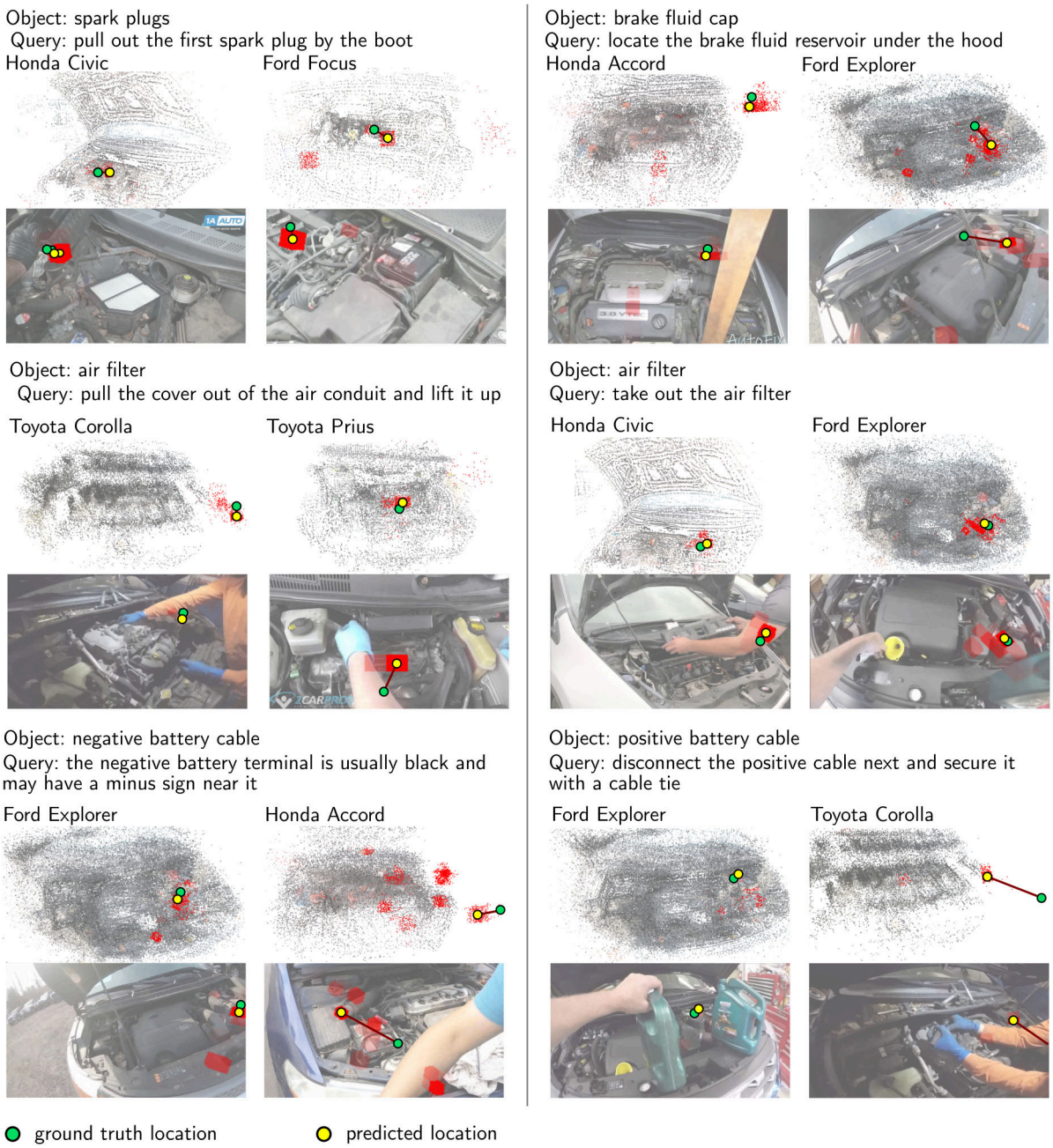


Figure 5.8: Examples of 3D text grounding with our model. Our model predicts a heatmap over a set of voxels with higher values assigned to locations that correspond to the query text. For clarity, we visualize the predicted voxels in random frames from the videos. The confidence that a given text query corresponds to a given location is shown in red with higher opacity corresponding to higher confidence. The ground truth 3D location is shown by a green dot.

Chapter 6

Discussion and perspectives

In this chapter, we summarize contributions of the thesis and discuss potential directions for future work.

6.1 Summary of contributions

Cross-task weakly supervised learning from instructional videos. In chapter 3 we presented a method for step localization in instructional videos, that allows for sharing information between steps of different tasks, by representing each step as a bag of components and sharing representations of each individual component across tasks. We have collected a new dataset of instructional videos, CrossTask, containing 4.7K videos for 83 different tasks from various domains, ranging from cooking to furniture assembly. This dataset includes many similar tasks by design, allowing us to demonstrate the benefits of sharing. Finally, we have evaluated our method for step localization on this newly collected dataset and compared it against a number of weakly-supervised baselines. Since our model provides a full control over the sharing pattern between steps, we were able to assess the benefits of sharing by evaluating step localization, obtained with three versions of our method: without any sharing, with sharing on step level, and with sharing on the level of step components, demonstrating gradual improvement over our baselines. We have, furthermore, demonstrated that our approach can be used to

localize previously unseen steps in new tasks.

Learning actionness via long-range temporal order verification. In Chapter 4 we tackled a problem of unsupervised actionness estimation in instructional videos. We have used an assumption that the action sequences, unlike their background, come in a particular order and contain visual clues, that allow to predict their order of appearance. Based on this assumption, we have proposed a method that learns to predict the actionness score, by solving a proxy task of temporal order verification. Our method doesn't require any human supervision, thus, allowing us to train the model on the large-scale dataset of instructional videos, HowTo100M [Miech et al. 2019]. We have evaluated our method on two datasets: CrossTask and COIN [Y. Tang et al. 2019] on the task of frame-wise action vs background classification. We have furthermore applied our method to the problems of temporal action proposal generation and action localization in instructional videos. In order to demonstrate the benefits of our background modeling for action localization, we have combined our actionness score with action localization methods proposed in Chapter 3 and in [Miech et al. 2020, 2019] and have demonstrated an improvement, compared to each of our baselines.

Reconstructing and grounding narrated instructional videos in 3D. Finally, in Chapter 5 we focused on the problem of spatial language grounding in instructional videos. We have considered the case of instructional videos that share a similar scene. Proposed approach consists in two steps. First, we have reconstructed a common 3D scene. This was done by running 3D reconstruction with COLMAP [Schonberger et al. 2016] on each individual video, and then aligning all reconstructions in the 3D space by estimating a rigid transformation between each pair. Second, we have trained a 3D language grounding model, using narration from the videos as the source of training data. We have evaluated our method on a set of car maintenance videos and have shown that our approach allows to associate textual instructions with corresponding locations in the 3D scene.

6.2 Perspectives

We conclude this thesis by discussing potential directions for future research.

Automatic parsing of textual guides. Our method presented in Chapter 3 relies on a manually provided list of steps for each task. [Alayrac et al. 2016] propose, instead, to recover a list of steps directly from the video narrations. This method, however, relies heavily on the quality of subtitles, which, in practice, are often noisy, especially in the case of subtitles, produced by ASR. Language is not the principle mean of conveying the information about the task in instructional videos, which results in a poor quality of subtitles in such videos. On the contrary, textual guides, that can be found on how-to sites, such as WikiHow.com, are significantly cleaner, and often provide some structure, that helps to separate the steps. Such sites may potentially be a better source of data for automatic extraction of step sequence. However, unlike subtitles for instructional videos, it is significantly harder to obtain many "recipes" for a given task, as WikiHow only provides a single description of each task. This makes the sequence alignment, applied to the subtitles in [Alayrac et al. 2016], impossible in this case. A possible solution to this problem may come from the same observation that we have used in Chapter 3: different tasks often share similar steps. The approach to localizing the steps within the text may then be equivalent to the extreme case of our cross-task learning method, where we're only provided with a single example per task, while the total number of available tasks is of the order of 10K.

End-to-end learning of actionness score. Our model for actionness estimation, presented in Chapter 4, uses I3D backbone, pretrained on the task of joint text-video embedding [Miech et al. 2020] and fixed during training. We therefore rely on existing video features and only learn the joint actionness and order verification module. It may be beneficial for the performance of our model to finetune or even train the video representation on our task from scratch, especially, given that the task of order verification was successfully used to learn video representations in the past (*e.g.* by [Misra et al. 2016]). Training our model end-to-end, however, is not straightforward. First, training a deep 3D-convolutional network on the scale of HowTo100M presents a significant computational challenge. An additional challenge may rise from the fact that our method is based on the assumed relation between actionness and order clues. As we have shown in Chapter 4, this assumption doesn't always hold. In particular, we have shown, that intro and outro parts of the videos contain strong order clues, but no actions. This results in undesirable shortcuts even in the case of pre-trained video features, if these

parts are not preemptively removed from videos. Due to a much higher capacity of the model with non frozen backbone, training end-to-end may introduce more undesirable shortcuts. This would require modification of our method to properly take into account the weak nature of our core assumption.

Unified framework for spatio-temporal localization in instructional videos. Our methods, presented in Chapters 3, 4 and 5, address different aspects of spatio-temporal step localization in instructional videos: **(i)** step classification, **(ii)** background modeling and **(iii)** spatial localization within the 3D scene, respectively. This allows to solve the problem of spatio-temporal localization step by step. It may be beneficial, however, to model spatial and temporal localization of steps jointly to solve this task, instead of relying on independent methods for temporal and spatial localization. In particular, learning spatial language grounding from narrated videos may be easier, if we can localize the temporal segments of the video, where actions occur, instead of just taking random segments of video and corresponding narrations, that are not guaranteed to be groundable. On the other hand, the temporal localization may benefit from knowing where exactly an action may occur within a frame. Such knowledge would allow to ignore spatial background noise when representing video for temporal action localization. This can be achieved, for example, by alternating between updating the weights of language grounding model and updating the weights of step classifier during training.

Mining visual events in instructional videos. In this thesis we have addressed the problem of automatic instructional video understanding. As discussed in Chapter 1, this problem has a number of practical applications. However, the utility of instructional videos goes beyond understanding complex human tasks. Instructional videos offer a large amount of diverse video-linguistic data. Many visual events that occur in the instructional videos may be hard to collect explicitly from websites such as YouTube. For example, it is difficult to find video examples for "boring" actions, such as "pouring water" with YouTube search. On the other hand, a lot of examples for this action can be found in instructional videos. The idea of mining instructional videos for various actions is similar to [Fouhey et al. 2018], who proposed to use lifestyle vlogs as a source of daily interactions. The methods, proposed in this thesis may be used to facilitate collecting action examples from instructional videos. In particular, our method, presented

in Chapter 4 can be used to collect high-scoring action proposals from large scale set of videos, such as HowTo100M. These candidate action clips may then be culled manually in order to obtain a clean large-scale dataset of short action clips. Such dataset may be used to learn action representations, as well as for pre-training purposes. In particular, as shown by [Miech et al. 2020], instructional videos may be a good source of pre-training data for various down-stream tasks.

A. Appendix of Chapter 3

In this appendix, we provide more details on our step localization method, presented in Chapter 3, together with some additional results of our method. In Section A.1 we describe details of our narration-based temporal constraints and the optimization procedure. Section A.2 provides additional quantitative and qualitative results of our method, including classifier scores and localized steps. We also provide more examples and analysis of failure cases.

A.1 Modeling instructional videos

A.1.1 Temporal text localization

In this section we explain in detail how we obtain temporal constraints from subtitles of the video. We assume that each step in the video occurs roughly at the same time as it is mentioned in the narration. Step localization in narrations is challenging for several reasons. First, the same step may be described in different ways (*e.g. cut steak* and *slice meat*). It may contain a reference (*e.g. cut it*). Second, a mention of a step doesn't guarantee, that the step occurs at the same time (*e.g.* Let the steak rest before *cutting it*). Since most of the videos in our dataset are unprofessional, the narrator doesn't usually follow a strict scenario and often talks about unrelated topics. Finally, most of the subtitles are produced by YouTube automatic speech recognition, and, therefore, contain errors and lack punctuation.

As described in Section 3.5.1, we provide a short textual description of each step. These

descriptions are matched to the text within a sliding window over the subtitles, in order to find where each step is mentioned. More formally, let f be a function, mapping a sequence of words of variable length into $\mathbb{R}^D$. Applying this function to the text within a sliding window of size w yields a matrix $U \in \mathbb{R}^{L \times D}$, where L is the number of words in the subtitles. Applying the same function to the description of each step gives us a matrix $V \in \mathbb{R}^{K \times D}$, where K is a total number of steps. Assuming that $\sum_{d=1}^{D} U_{ld}^2 = 1$ for any $l = 1 \dots L$, and that $\sum_{d=1}^{D} V_{kd}^2 = 1$ for any $k = 1 \dots K$ (i.e., the features are unit-norm), $S = UV^T \in \mathbb{R}^{L \times K}$ is a matrix of cosine similarities between vector representations of subtitles and descriptions of steps.

We find the best matching $A \in \{0, 1\}^{L \times K}$ between the steps and the subtitles, that satisfies the ordering of the steps, by solving a linear problem

$$\min_{A \in \mathcal{A}} \sum_{l,k} S_{l,k} A_{l,k} \tag{A.1}$$

where $\mathcal{A}$ is a set of assignments that satisfy at-least-one and ordering constraints. This problem can be efficiently solved via dynamic programming, as described in Section A.1.2. Imposing the ordering constraints during step localization helps to avoid spurious mentions of steps, that do not follow the scenario of a task.

We try different choices of mapping f. The first is TF-IDF representation of the text within sliding window. The second is a Word2Vec-like word embedding [Mikolov et al. 2013], followed by a max-pooling over sliding window. We obtain our word embedding by training the Fasttext model [Bojanowski et al. 2017a] with dimension 100. The model is trained on a corpus of subtitles of 2 million YouTube videos for 6729 tasks from wikiHow.

Finally, we propose a way to learn a better aggregation function than max-pooling for the word vectors. Assume that we are given a set of sentences $\mathcal{I}$ of various lengths. Each sentence i is represented by $X^i \in \mathbb{R}^{M_i \times d_0}$, where M_i is the number of words in a sentence and $X_m^i \in \mathbb{R}^{d_0}$ is the feature vector corresponding to m-th word in the sentence. Assume that for each sentence i we are also given a set of sentences $S_i \subset \mathcal{I}$ with similar meaning

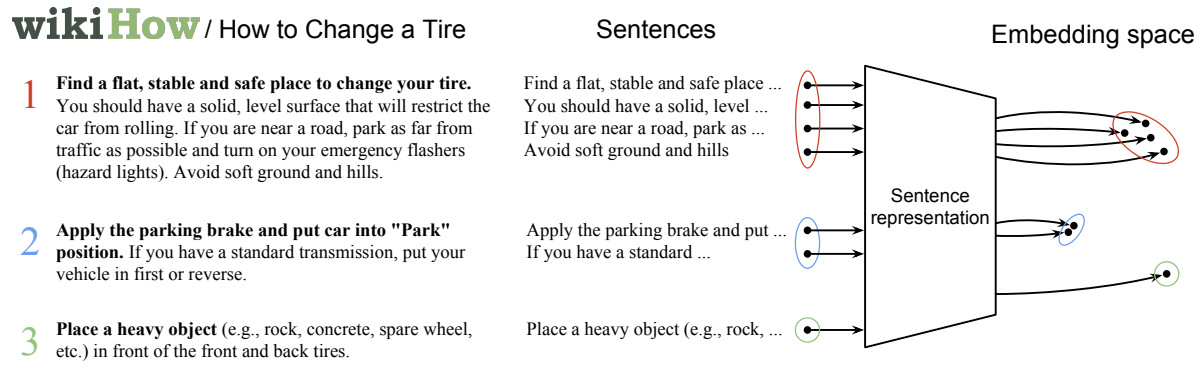


Figure A.1: Example of three wikiHow steps for the *Change a Tire* task. Our method learns a similarity function that pulls representations of the sentences from the same paragraphs closer together, and pushes the sentences from different paragraphs away from each other

and a set of sentences $D_i \subset \mathcal{I}$ with different meaning. f is learnt by minimizing loss

$$\sum_{i \in \mathcal{I}} \frac{1}{|S_i|} \sum_{j \in S_i, k \in D_i} \max[0, \text{sim}(f(X^i), f(X^k)) - \text{sim}(f(X^i), f(X^j)) + h], \tag{A.2}$$

where $\text{sim}(a, b) = \frac{a^T b}{\|a\| \|b\|}$ is the cosine similarity function, and, h is the margin constant. This can be understood as pushing representations of sentences with similar meaning closer together, as opposed to sentences with different meaning. We take f in the form of 1D convolution with kernel length 1 and a number of filters d, followed by the global max-pooling, and by a linear mapping $\mathbb{R}^d \to \mathbb{R}^d$. We take $d = 300$ and $h = 0.1$.

We train our model on the set of sentences from wikiHow. Descriptions of the tasks on wikiHow are organized into step paragraphs, as shown on figure A.1. We assume that sentences within the same paragraph describe similar concepts, while sentences from different steps of the same task have different meanings. For each sentence i, S_i is defined as a set of sentences from the same paragraph, and D_i is defined as a set of sentences from other paragraphs within the same wikiHow page.

We evaluate alternative text representations by comparing obtained constraints with the ground truth on the set of primary tasks. The results are shown in Table A.1. Our

Table A.1: Precision and recall of the constraints obtained with our method, averaged over 18 primary tasks.

	Precision (%)	Recall (%)
Max-pooled word vectors	11.6	10.4
TF-IDF	13.3	11.4
Proposed	**15.9**	**13.9**

aggregation function, trained on wikiHow outperforms TF-IDF and max-pooled word vectors both in terms of precision and recall.

A.1.2 Constrained linear optimization

Our optimization procedure, inference and temporal text localization require solving a linear problem of the form

$$\min_{Y \in \mathcal{C}} \sum_{t,k} S_{tk} Y_{tk}, \tag{A.3}$$

where $S \in \mathbb{R}^{T \times K}$ and $\mathcal{C}$ is the set of all assignments from $\{0,1\}^{T \times K}$ that satisfy the *ordering* and *at-least-once* constraints. At-least-once constraints mean that every step k should be picked at least once: $\sum_t Y_{t,k} \geq 1$ for any $k = 1 \dots K$. Ordering constraints mean that the step $k-1$ should precede step k for any $k \geq 2$. This problem can be solved efficiently via dynamic programming. First, we rewrite the problem in the form:

$$\min_{y \in \tilde{C}} \sum_{t} S_{t y_t}, \tag{A.4}$$

where $y_t \in \{0, 1, \dots, K\}$ is the step label at time t (0 stands for background, i.e. when no step is selected). The ordering constraints impose that $y_{t+1} \in \{0, z_t\}$, where $z_t = \max(y_1, \dots, y_t)$ is the last non-background step. We define state x_t at time t as a pair (y_t, z_t). Note, that for a given state x_t, the only possible x_{t-1} that satisfies the constraints are (z_t, z_t), (z_t-1, z_t-1) and $(0, z_t-1)$ if $y_t \neq 0$, and (z_t-1, z_t-1) and $(0, z_t-1)$ otherwise. We denote this set of possible previous states as $\mathcal{P}(x_t)$. The minimum cumulative cost

for state $x_t = x$ at time t is

$$V(x,t) = \min_{x_1,\ldots,x_{t-1} \mid x_t = x} \sum_{\tau=1}^{t} S_{\tau y_\tau}. \tag{A.5}$$

Define $C(x_t, x_{t-1}) = S_{ty_t}$ if $x_{t-1} \in \mathcal{P}(x_t)$ and $C(x_t, x_{t-1}) = +\infty$ otherwise (for simplicity we denote $x_0 = (0,0)$). This allows to rewrite (A.5) in the recursive form:

$$V(x,t) = \min_{x'}(C(x,x') + V(x', t-1)). \tag{A.6}$$

We compute $V(x,t)$ recursively for $t = 1,\ldots,T$, using (A.6). In practice, computing $V(x,t)$ given $V(x', t-1)$ for all x' requires minimization only over $x' \in \mathcal{P}(x)$ and can be done in O(1). Since there are $2K$ possible states, the complexity of computing $V(x,t)$ for all x and t is $O(KT)$. To satisfy *at-least-once* constraints, the final state x_T must be either (K,K), or $(0,K)$. To get the optimal assignment, we take $x_T^* = \underset{x \in \{(K,K),(0,K)\}}{\arg\min} V(x,T)$ and find $x_t^* = \underset{x \in \mathcal{P}(x_{t+1}^*)}{\arg\min} V(x, t-1)$ recursively for every $t = T-1,\ldots,1$.

A.1.3 Optimization for discriminative clustering

The discriminative clustering problem

$$\min_{Y \in \mathcal{C}, F \in \mathcal{F}} -\sum_{t,k} Y_{t,k} \log\left(\frac{\exp(f_k(x_t))}{\sum\limits_{k'} \exp(f_{k'}(x_t))}\right), \tag{A.7}$$

introduced in Section 3.4 can't be solved efficiently with standard techniques, such as projected gradient descent, because the projection over our constraint set $\mathcal{C}$ is computationally expensive.

Our optimization method can be applied to a broader class of problems of the form

$$\min_{Y \in \mathcal{C}, \theta \in \mathbb{R}^m} \sum_{tk} Y_{tk} F_{tk}(\theta). \tag{A.8}$$

Given solution (Y^l, θ^l) at l-th iteration, we define a quadratic upper bound for $F(\theta)$ in the neighbourhood of θ^l: $F_{tk}(\theta) \leq \tilde{F}_{tk}(\theta; \theta^l)$, where

$$\tilde{F}_{tk}(\theta; \theta^l) = F_{tk}(\theta^l) + \nabla F_{tk}(\theta^l)(\theta - \theta^l) + \frac{1}{2\delta K}||\theta - \theta^l||^2. \tag{A.9}$$

Note, that $\sum\limits_{t,k} Y_{tk} F_{tk}(\theta) \leq \sum\limits_{t,k} Y_{tk} \tilde{F}_{tk}(\theta; \theta^l)$ for any (Y, θ) and that $\sum\limits_{t,k} Y^l_{tk} F_{tk}(\theta^l) = \sum\limits_{t,k} Y^l_{tk} \tilde{F}_{tk}(\theta^l)$. This means, that for any (Y^{l+1}, θ^{l+1}), s.t. $\sum\limits_{t,k} Y^{l+1}_{tk} \tilde{F}_{tk}(\theta^{l+1}) \leq \sum\limits_{t,k} Y^l_{tk} \tilde{F}_{tk}(\theta^l)$, the same inequality holds for F:

$$\sum_{t,k} Y^{l+1}_{tk} F_{tk}(\theta^{l+1}) \leq \sum_{t,k} Y^l_{tk} F_{tk}(\theta^l). \tag{A.10}$$

For the problem

$$\min_{Y \in \mathcal{C}, \theta \in \mathbb{R}^m} \sum_{t,k} Y_{tk} \tilde{F}_{tk}(\theta) \tag{A.11}$$

it is possible to find a global minimum. The minimization with respect to θ yields

$$\theta^*(Y) = \theta^l - \delta K \frac{\sum\limits_{t,k} Y_{tk} \nabla F_{tk}(\theta^l)}{\sum\limits_{t,k} Y_{tk}}. \tag{A.12}$$

Since $\sum\limits_{t,k} Y_{tk} = K$, this expression can be simplified as

$$\theta^*(Y) = \theta^l - \delta \sum_{t,k} Y_{tk} \nabla F_{tk}(\theta^l). \tag{A.13}$$

Substituting θ with $\theta^*(Y)$ in (A.11) leads to the problem

$$\min_{Y \in \mathcal{C}} \sum_{t,k} [F_{tk}(\theta^l) - \frac{\delta}{2}||\nabla F_{tk}(\theta^l)||^2] Y_{tk}. \tag{A.14}$$

This is a linear problem that can be solved as described in Section A.1.2. Denote Y^* a solution of (A.14). We obtain $\theta*$ by substituting Ys with Y^* in (A.13). The pair (Y^*, θ^*) is a global minimum for (A.11). We take this pair as a new solution (Y^{l+1}, θ^{l+1}).

The described optimization procedure can be seen as alternating between solving a linear problem (A.14) for Y and a gradient descent step with the learning rate δ

$$\theta^{l+1} = \theta^l - \delta \sum_{t,k} Y_{tk}^{l+1} \nabla F_{tk}(\theta^l). \tag{A.15}$$

A.2 Experiments

A.2.1 Comparison of evaluation metrics

At test time we predict one temporal unit per step and assume a correct detection if it falls within a ground truth interval for the corresponding step. This is motivated by the fact that in weakly supervised context, when no information about exact temporal extents of steps is given during the training, prediction of step time intervals is an ill-posed problem. Indeed, even people do not always agree on the action boundaries. Predicting punctual steps, defined as the most consistent and distinguishable frames in the videos, allows to avoid this problem. Although our model is trained for this punctual prediction, it may still be used to predict temporally extended steps, for example, by thresholding model's confidence of each step for each frame. Does this yield reasonable predictions? To answer this question we evaluate our model, using mAP metric. Table A.2 contains results, averaged over the primary tasks and compared to baselines. Here recall stands for the same evaluation procedure, as described in Section 3.6 (global non-maximal suppression + recall). Note that both proposed ways of evaluation yield highly correlated results with recall roughly equal to mAP $\times$ 2. The only exception is Richard'18 that under-performs in the case of recall. This may be caused by the fact that, unlike other methods in Table A.2, it is trained to predict step intervals, and not punctual steps.

A.2.2 Importance of temporal text constraints

Temporal constraints, obtained from narration, provide a very noisy supervision. In case of our primary tasks, the intersection over union between the ground truth of the steps and the corresponding temporal text constraints is only 7.9%. Moreover, 61% of ground

Metric	Random	Richard'18	Alayrac'16	Ours (no sharing)	Ours (with sharing)
Recall	8.27	6.7	13.3	18.6	22.4
mAP	4.3	5.5	6.9	8.9	11.0

Table A.2: Results of cross-task learning, evaluated with mAP and recall metrics and averaged over primary tasks. Standard deviation does not exceed 0.3% for mAP and 1% for recall.

truth steps lie entirely outside of the constraint intervals. This means that our method is unable to assign a step to a correct frame even with a perfect classifier, if it is forced to satisfy these constraints. This is likely to be a disadvantage during training which tries to fit the step model to incorrect temporal intervals. Could it be better to learn our model without temporal constraints? We answer this question by training and evaluating our solution in the same setup as before, but without text constraints at the training time. The resulting recall is 17%, compared to 22.4%, when training with text constraints. This gain of 5.4% shows the importance of text guidance during training even at the presence of the high level of noise.

A.2.3 Qualitative results

Figures A.2-A.7 shows the outputs of our model for videos for several tasks. We show the outputs of classifiers for each step at each frame of the video, as well as the inferred solution and compare it with the ground truth. Figure A.2 illustrates two kinds of error, caused by our assumptions. First,the step *Whisk Mixture* is localized in the area of low confidence for this step. This is caused by the next step, *Pouring Egg*, that precedes *Whisking Mixture* in this particular video. Second, false detection for *Topping Toast* is due to the absence of this step in the video, while we assume that every step is present. Note that although step *Top Toast* doesn't appear in the video, the classifier puts high and well localized scores in the end of the video. This is because videos usually end with a demonstration of a final product on a plate. The model captures this visual consistency between the videos and takes it for the final step.

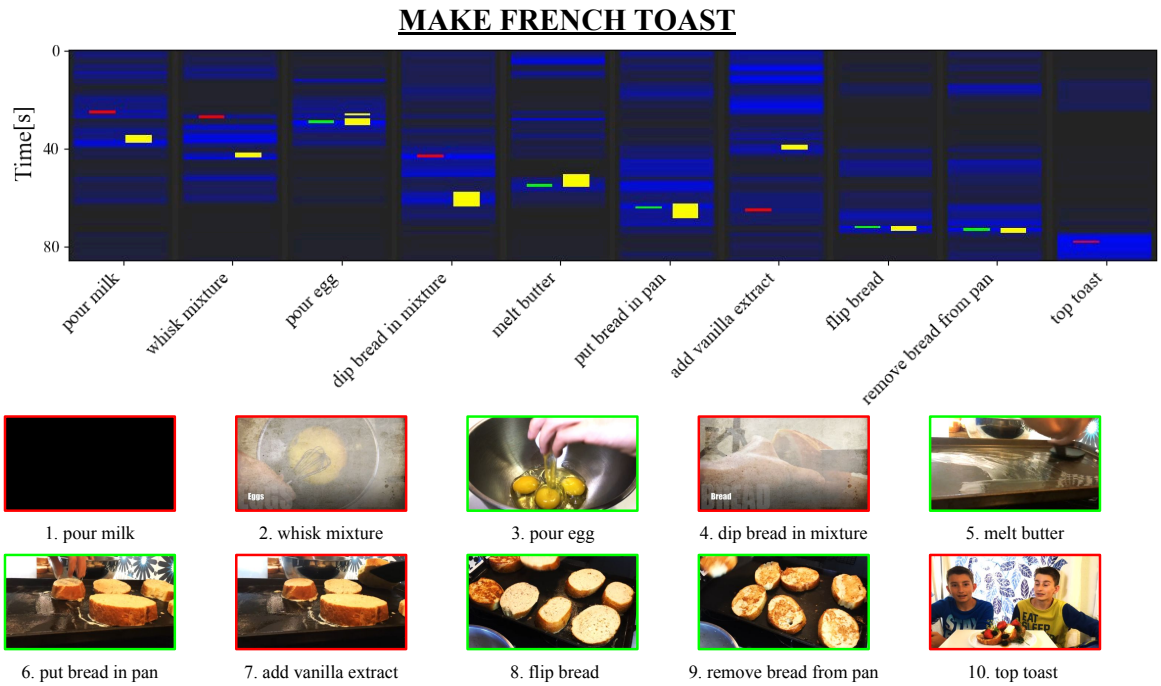


Figure A.2: Example of obtained solution for *Make French Toast* task. Outputs of the classifier are shown in blue. Correctly localized steps are shown in green. False detections are shown in red. Ground truth intervals for the steps are shown in yellow. Failure cases include false localization due to the ordering constraints (*Pour milk*, *Whisk mixture* and *Dip bread*) and due to a missing step (*Top toast*).

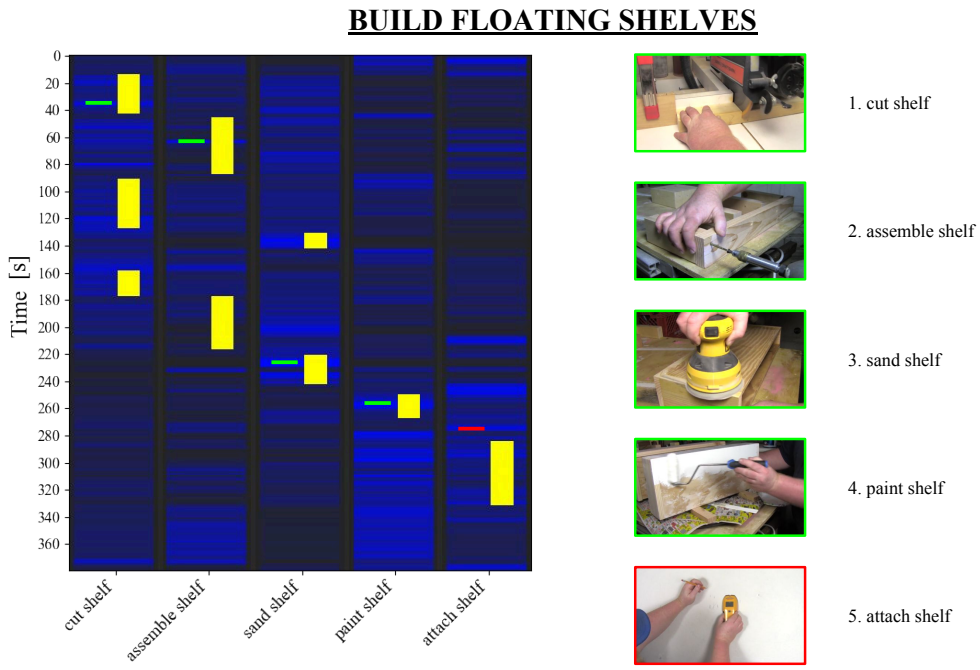


Figure A.3: Example of obtained solution for *Build Floating Shelves* task. Outputs of the classifier are shown in blue. Correctly localized steps are shown in green. False detections are shown in red. Ground truth intervals for the steps are shown in yellow.

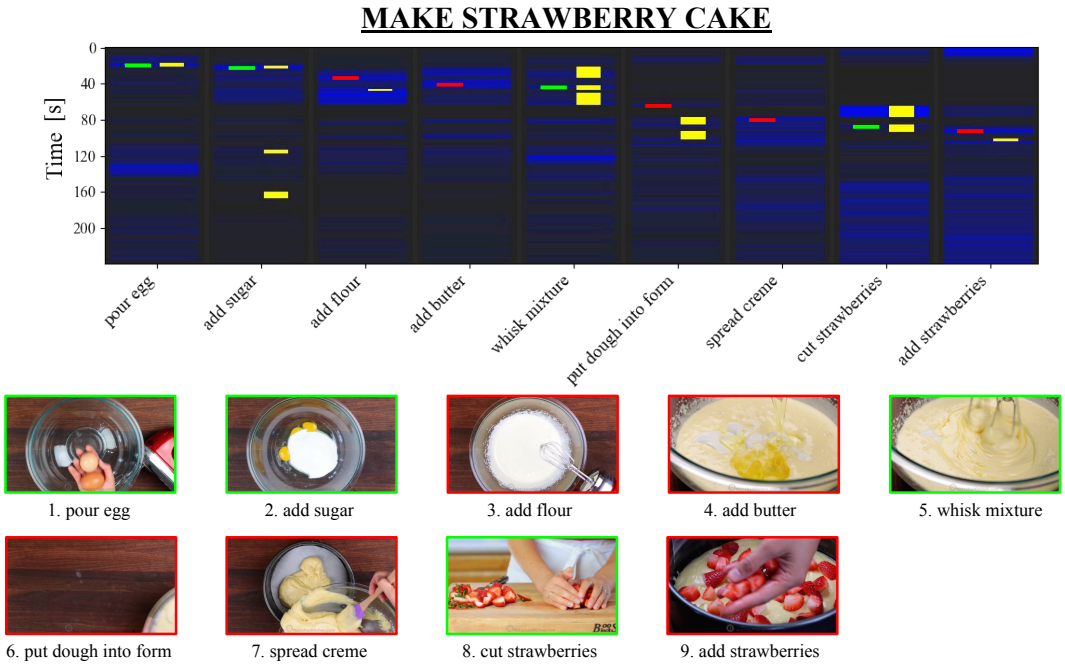


Figure A.4: Example of obtained solution for *Make Strawberry Cake* task. Outputs of the classifier are shown in blue. Correctly localized steps are shown in green. False detections are shown in red. Ground truth intervals for the steps are shown in yellow.

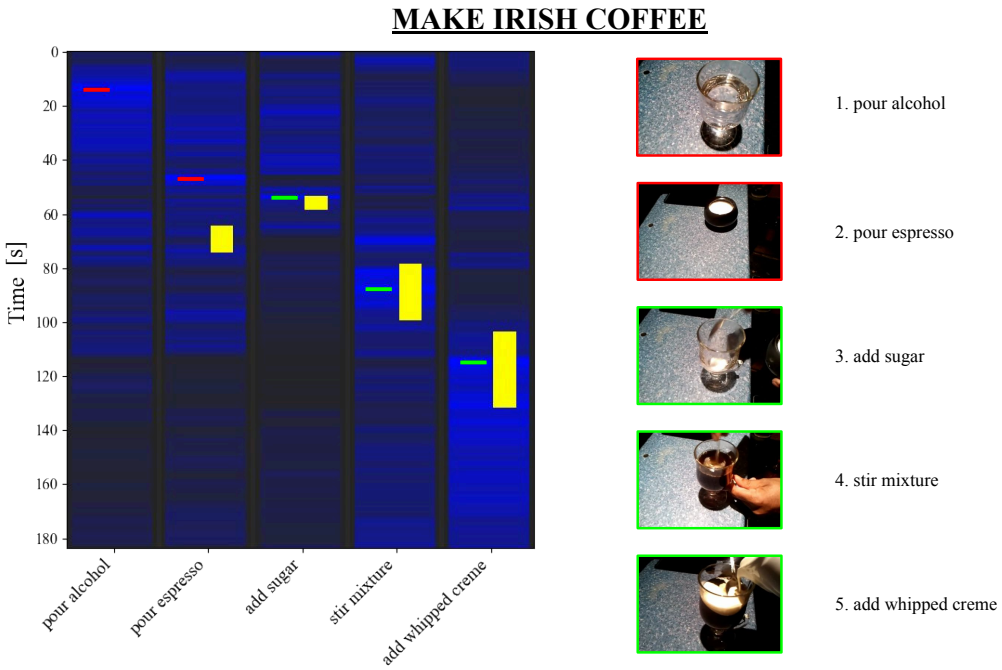


Figure A.5: Example of obtained solution for *Make Irish Coffee* task. Outputs of the classifier are shown in blue. Correctly localized steps are shown in green. False detections are shown in red. Ground truth intervals for the steps are shown in yellow.

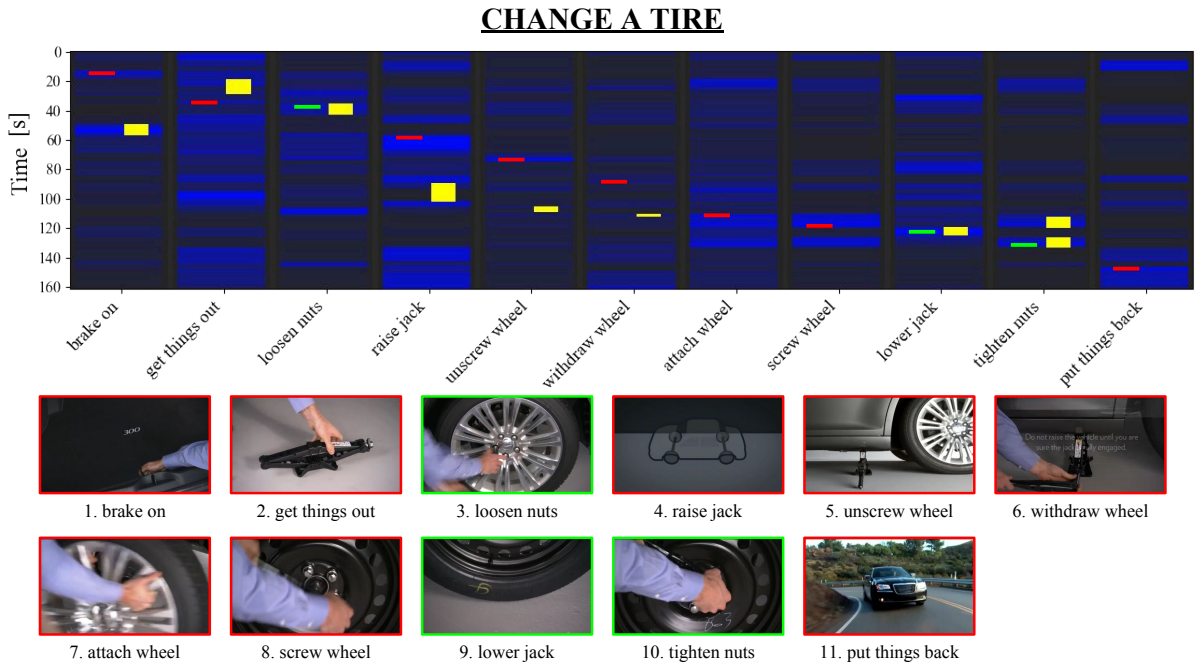


Figure A.6: Example of obtained solution for *Change a Tire* task. Outputs of the classifier are shown in blue. Correctly localized steps are shown in green. False detections are shown in red. Ground truth intervals for the steps are shown in yellow.

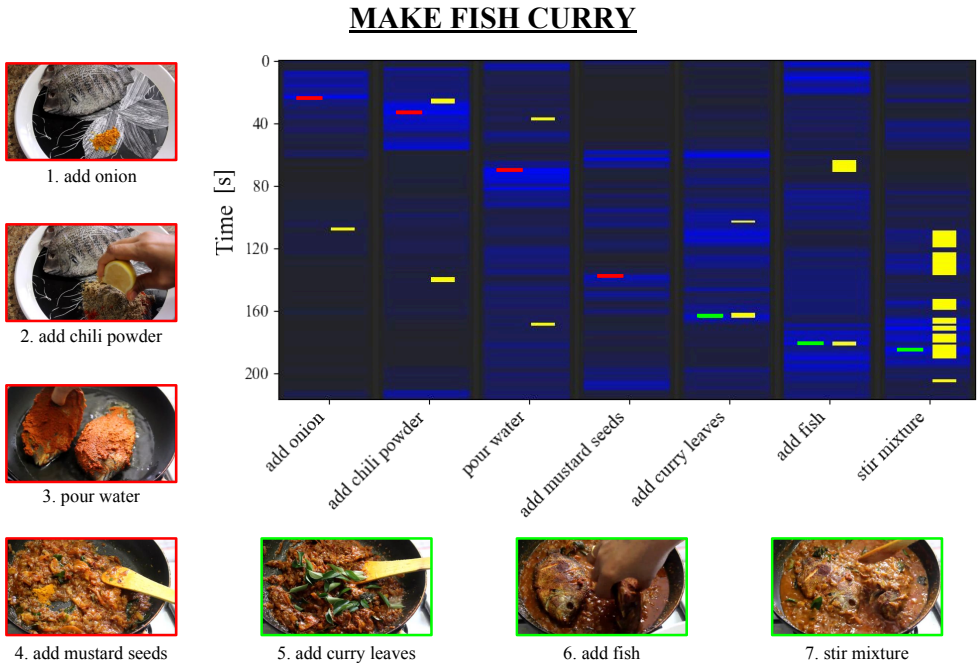


Figure A.7: Example of obtained solution for *Make Fish Curry* task. Outputs of the classifier are shown in blue. Correctly localized steps are shown in green. False detections are shown in red. Ground truth intervals for the steps are shown in yellow.

Figure A.8: Erroneous predictions, involving wrong objects and actions. Correct object/action is in green. Our method is not capable of distinguishing particular kinds of objects, especially liquids and powders, due to the nature of the features. Examples for the wrong action components show that in many cases the method captures a static context in which object occurs, rather than performed action.

Bibliography

Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S. M., & Szeliski, R. (2011). Building Rome in a day. *Communications of the ACM*, *54*(10), 105–112 (cit. on p. 94).

Agrawal, P., Carreira, J., & Malik, J. (2015). Learning to see by moving. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 72).

Alayrac, J.-B., Bojanowski, P., Agrawal, N., Sivic, J., Laptev, I., & Lacoste-Julien, S. (2016). Unsupervised learning from narrated instruction videos. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 26, 27, 33, 34, 35, 40, 42, 48, 50, 52, 62, 64, 65, 70, 72, 73, 102, 115).

Alayrac, J.-B., Bojanowski, P., Agrawal, N., Sivic, J., Laptev, I., & Lacoste-Julien, S. (2018). Learning from narrated instruction videos. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, *40*, 2194–2208 (cit. on pp. 62, 92, 95, 96).

Alayrac, J.-B., Sivic, J., Laptev, I., & Lacoste-Julien, S. (2017). Joint discovery of object states and manipulation actions. *ICCV* (cit. on pp. 26, 37, 48, 49, 52, 58, 73, 96).

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., & Sivic, J. (2016). NetVLAD: CNN architecture for weakly supervised place recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 106).

Bach, F. R., & Harchaoui, Z. (2007). DIFFRAC: A discriminative and flexible framework for clustering. *NIPS* (cit. on pp. 26, 49, 52, 64).

Bilen, H., Fernando, B., Gavves, E., Vedaldi, A., & Gould, S. (2016). Dynamic image networks for action recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 22).

Bojanowski, P., Bach, F., Laptev, I., Ponce, J., Schmid, C., & Sivic, J. (2013). Finding actors and actions in movies. *ICCV* (cit. on pp. 26, 27, 28).

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017a). Enriching word vectors with subword information. *TACL* (cit. on p. 120).

Bojanowski, P., & Joulin, A. (2017b). Unsupervised learning by predicting noise. *ICML* (cit. on p. 52).

Bojanowski, P., Lajugie, R., Bach, F., Laptev, I., Ponce, J., Schmid, C., & Sivic, J. (2014). Weakly supervised action labeling in videos under ordering constraints. *ECCV* (cit. on pp. 26, 27, 28, 29, 30, 34, 50).

Bojanowski, P., Lajugie, R., Grave, E., Bach, F., Laptev, I., Ponce, J., & Schmid, C. (2015). Weakly-supervised alignment of video with text. *ICCV* (cit. on pp. 26, 27, 28, 50).

Caba Heilbron, F., Escorcia, V., Ghanem, B., & Carlos Niebles, J. (2015). ActivityNet: A large-scale video benchmark for human activity understanding. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 23).

Caron, M., Bojanowski, P., Joulin, A., & Douze, M. (2018). Deep clustering for unsupervised learning of visual features. *ICCV* (cit. on p. 52).

Carreira, J., Noland, E., Hillier, C., & Zisserman, A. (2019). A short note on the Kinetics-700 human action dataset. *arXiv* (cit. on p. 24).

Carreira, J., & Zisserman, A. (2017). Quo vadis, action recognition? a new model and the Kinetics dataset. *CVPR* (cit. on pp. 22, 24, 48, 50, 57, 70).

Chang, C.-Y., Huang, D.-A., Sui, Y., Fei-Fei, L., & Niebles, J. C. (2019). D3TW: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 27, 29).

Chang, C.-Y., Huang, D.-A., Xu, D., Adeli, E., Fei-Fei, L., & Niebles, J. C. (2020). Procedure planning in instructional videos. *ECCV* (cit. on p. 39).

Chen, H., Lee, A. S., Swift, M., & Tang, J. C. (2015). 3D collaboration method over HoloLens™ and Skype™ end points. *Proceedings of the 3rd International Workshop on Immersive Media Experiences*, 27–30 (cit. on p. 92).

Chen, W., Xiong, C., Xu, R., & Corso, J. J. (2014). Actionness ranking with lattice conditional ordinal random fields. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 71, 73).

Damen, D., Doughty, H., Maria Farinella, G., Fidler, S., Furnari, A., Kazakos, E., Moltisanti, D., Munro, J., Perrett, T., Price, W., & Wray, M. (2018). Scaling egocentric vision: The EPIC-KITCHENS dataset. *ECCV* (cit. on pp. 24, 50).

Damen, D., Leelasawassuk, T., Haines, O., Calway, A., & Mayol-Cuevas, W. (2014). You-Do, I-Learn: Discovering task relevant objects and their modes of interaction from multi-user egocentric video. *BMVA* (cit. on pp. 52, 96).

Das, P., Xu, C., Doell, R. F., & Corso, J. J. (2013). A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 41).

Dijkstra, E. W. et al. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, *1*(1), 269–271 (cit. on p. 100).

Ding, L., & Xu, C. (2018). Weakly-supervised action segmentation with iterative soft boundary assignment. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 27, 29).

Doersch, C., Gupta, A., & Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 72).

Doughty, H., Damen, D., & Mayol-Cuevas, W. (2018). Who's better? who's best? pairwise deep ranking for skill determination. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 38).

Doughty, H., Laptev, I., Mayol-Cuevas, W., & Damen, D. (2020). Action modifiers: Learning from adverbs in instructional videos. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 36).

Doughty, H., Mayol-Cuevas, W., & Damen, D. (2019). The pros and cons: Rank-aware temporal attention for skill determination in long videos. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 39).

Duchenne, O., Laptev, I., Sivic, J., Bach, F., & Ponce, J. (2009). Automatic annotation of human actions in video. *ICCV* (cit. on pp. 26, 27).

Dwibedi, D., Aytar, Y., Tompson, J., Sermanet, P., & Zisserman, A. (2019). Temporal cycle-consistency learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 72).

Elhamifar, E., & Huynh, D. (2020). Self-supervised multi-task procedure learning from instructional videos (cit. on pp. 32, 92, 95).

Elhamifar, E., & Naing, Z. (2019). Unsupervised procedure learning via joint dynamic summarization. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (cit. on pp. 32, 40, 42).

Engilberge, M., Chevallier, L., Pérez, P., & Cord, M. (2018). Finding beans in burgers: Deep semantic-visual embedding with localization. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 96).

Escorcia, V., Heilbron, F. C., Niebles, J. C., & Ghanem, B. (2016). DAPs: Deep action proposals for action understanding. *The European Conference on Computer Vision (ECCV)* (cit. on p. 73).

Fang, K., Wu, T.-L., Yang, D., Savarese, S., & Lim, J. J. (2018). Demo2Vec: Reasoning object affordances from online videos. *CVPR* (cit. on p. 50).

Farha, Y. A., Ke, Q., Schiele, B., & Gall, J. (2017). Long-term anticipation of activities with cycle consistency. *arXiv* (cit. on p. 38).

Farha, Y. A., Richard, A., & Gall, J. (2018). When will you do what? - anticipating temporal occurrences of activities. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 37, 38).

Farhadi, A., Lim, I. E., Hoiem, D., & Forsyth, D. (2009). Describing objects by their attributes. *CVPR* (cit. on p. 52).

Feichtenhofer, C., Pinz, A., & Wildes, R. P. (2017). Spatiotemporal multiplier networks for video action recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 22).

Fernando, B., Bilen, H., Gavves, E., & Gould, S. (2017). Self-supervised video representation learning with odd-one-out networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 71, 72).

Fernando, B., Gavves, E., Oramas, J. M., Ghodrati, A., & Tuytelaars, T. (2015). Modeling video evolution for action recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 22).

Ferrari, V., & Zisserman, A. (2007). Learning visual attributes. *NIPS* (cit. on p. 52).

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395 (cit. on p. 106).

Fouhey, D. F., Kuo, W.-C., Efros, A. A., & Malik, J. (2018). From lifestyle vlogs to everyday interactions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 24, 50, 116).

Frahm, J.-M., Fite-Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.-H., Dunn, E., Clipp, B., Lazebnik, S. et al. (2010). Building Rome on a cloudless day. *ECCV* (cit. on p. 94).

Frank, M., & Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly* (cit. on p. 28).

Fried, D., Alayrac, J.-B., Blunsom, P., Dyer, C., Clark, S., & Nematzadeh, A. (2020). Learning to segment actions from observation and narration. *ACL*, 2569–2588 (cit. on p. 34).

Gao, J., Chen, K., & Nevatia, R. (2018). CTAP: Complementary temporal action proposal generation. *The European Conference on Computer Vision (ECCV)* (cit. on p. 73).

Gidaris, S., Singh, P., & Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. *ICLR* (cit. on p. 72).

Guadarrama, S., Krishnamoorthy, N., Malkarnenkar, G., Venugopalan, S., Mooney, R., Darrell, T., & Saenko, K. (2013). YouTube2Text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. *ICCV* (cit. on p. 53).

Guillaumin, M., Küttel, D., & Ferrari, V. (2014). Imagenet auto-annotation with segmentation propagation. *IJCV*, *110*(3), 328–348 (cit. on p. 100).

Ham, B., Cho, M., Schmid, C., & Ponce, J. (2016). Proposal flow. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 94).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *CVPR* (cit. on p. 57).

Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., Slaney, M., Weiss, R. J., & Wilson, K. (2017). CNN architectures for large-scale audio classification. *2017 IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP), 131–135. https://doi.org/10.1109/ICASSP.2017.7952132 (cit. on p. 57)

Huang, D.-A., Buch, S., Dery, L., Garg, A., Fei-Fei, L., & Niebles, J. C. (2018). Finding "it": Weakly-supervised reference-aware visual grounding in instructional videos. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 37, 52, 70, 73, 96).

Huang, D.-A., Fei-Fei, L., & Niebles, J. C. (2016). Connectionist temporal modeling for weakly supervised action labeling. *ECCV* (cit. on pp. 27, 28, 29, 30, 31, 48, 50, 92, 96).

Huang, D.-A., Lim, J. J., Fei-Fei, L., & Carlos Niebles, J. (2017). Unsupervised visual-linguistic reference resolution in instructional videos. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 36, 52, 73, 96).

Jaggi, M. (2013). Revisiting Frank-Wolfe: Projection-free sparse convex optimization. *ICML* (cit. on p. 28).

Jégou, H., Douze, M., & Schmid, C. (2008). Hamming embedding and weak geometric consistency for large scale image search. *ECCV* (cit. on p. 94).

Jenni, S., & Favaro, P. (2018). Self-supervised feature learning by learning to spot artifacts. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 72).

Jhuang, H., Gall, J., Zuffi, S., Schmid, C., & Black, M. J. (2013). Towards understanding action recognition. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 23).

Ji, J., Cao, K., & Niebles, J. C. (2019). Learning temporal action proposals with fewer labels. *The IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 73).

Ji, S., Xu, W., Yang, M., & Yu, K. (2013). 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(1), 221–231. https://doi.org/10.1109/TPAMI.2012.59 (cit. on p. 22)

Jie, Z., Wei, Y., Jin, X., Feng, J., & Liu, W. (2017). Deep self-taught learning for weakly supervised object localization. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 25).

Kantorov, V., Oquab, M., Cho, M., & Laptev, I. (2016). ContextLocNet: Context-aware deep network models for weakly supervised localization. *ECCV* (cit. on p. 25).

Karpathy, A., Joulin, A., & Li, F. F. F. (2014a). Deep fragment embeddings for bidirectional image sentence mapping. *Advances in Neural Information Processing Systems*, 1889–1897 (cit. on p. 96).

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014b). Large-scale video classification with convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 22, 23).

Kim, J. [Jonghyun], Jeong, Y., Stengel, M., Akşit, K., Albert, R., Boudaoud, B., Greer, T., Kim, J. [Joohwan], Lopes, W., Majercik, Z. et al. (2019). Foveated AR: Dynamically-foveated augmented reality display. *ACM Transactions on Graphics (TOG)*, *38*(4), 1–15 (cit. on p. 92).

Kim, S., Min, D., Ham, B., Jeon, S., Lin, S., & Sohn, K. (2017). FCSS: Fully convolutional self-similarity for dense semantic correspondence. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 100).

Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (cit. on pp. 57, 78).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *NIPS* (cit. on p. 22).

Kuehne, H., Arslan, A., & Serre, T. (2014). The language of actions: Recovering the syntax and semantics of goal-directed human activities. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 31, 40, 41).

Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., & Serre, T. (2011). HMDB: A large video database for human motion recognition. *2011 International Conference on Computer Vision*, 2556–2563. https://doi.org/10.1109/ICCV.2011.6126543 (cit. on p. 23)

Kuehne, H., Richard, A., & Gall, J. (2017). Weakly supervised learning of actions from transcripts. *CVIU* (cit. on pp. 31, 48, 50).

Kukleva, A., Kuehne, H., Sener, F., & Gall, J. (2019). Unsupervised learning of action classes with continuous temporal embedding. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 32, 92, 95).

Lacoste-Julien, S., Jaggi, M., Schmidt, M., & Pletscher, P. (2013). Block-coordinate Frank-Wolfe optimization for structural SVMs. *ICML* (cit. on p. 28).

Laptev, I. (2005). On space-time interest points. *International Journal of Computer Vision* (cit. on p. 21).

Laptev, I., Marszalek, M., Schmid, C., & Rozenfeld, B. (2008). Learning realistic human actions from movies. *CVPR* (cit. on p. 23).

Larsson, G., Maire, M., & Shakhnarovich, G. (2017). Colorization as a proxy task for visual understanding. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 72).

Lee, H.-Y., Huang, J.-B., Singh, M., & Yang, M.-H. (2017). Unsupervised representation learning by sorting sequences. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 72).

Li, J., Lei, P., & Todorovic, S. (2019). Weakly supervised energy-based learning for action segmentation. *The IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 29).

Li, J., & Todorovic, S. (2020). Set-constrained Viterbi for set-supervised action segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 30).

Lin, T., Liu, X., Li, X., Ding, E., & Wen, S. (2019). BMN: Boundary-matching network for temporal action proposal generation. *The IEEE International Conference on Computer Vision (ICCV)* (cit. on pp. 73, 79).

Lin, T., Zhao, X., Su, H., Wang, C., & Yang, M. (2018). BSN: Boundary sensitive network for temporal action proposal generation. *The European Conference on Computer Vision (ECCV)* (cit. on pp. 73, 79).

Liu, J., Kuipers, B., & Savarese, S. (2011). Recognizing human actions by attributes. *CVPR* (cit. on p. 52).

Liu, J., Luo, J., & Shah, M. (2009). Recognizing realistic actions from videos "in the wild". *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 1996–2003. https://doi.org/10.1109/CVPR.2009.5206744 (cit. on p. 23)

Liu, Y., Ma, L., Zhang, Y., Liu, W., & Chang, S.-F. (2019). Multi-granularity generator for temporal action proposal. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 73).

Lowe, D. (1999). Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, *2*, 1150–1157 vol.2. https://doi.org/10.1109/ICCV.1999.790410 (cit. on pp. 94, 95)

Ma, F., Zhu, L., Yang, Y., Zha, S., Kundu, G., Feiszli, M., & Shou, Z. (2020). SF-Net: Single-frame supervision for temporal action localization. *ECCV* (cit. on p. 25).

Malmaud, J., Huang, J., Rathod, V., Johnston, N., Rabinovich, A., & Murphy, K. (2015). What's Cookin'? interpreting cooking videos using text, speech and vision. *NAACL* (cit. on pp. 33, 40, 41, 50, 52, 58, 92, 96).

Marszalek, M., Laptev, I., & Schmid, C. (2009). Actions in context. *CVPR* (cit. on p. 23).

Martín-Gutiérrez, J., Mora, C. E., Añorbe-Díaz, B., & González-Marrero, A. (2017). Virtual technologies trends in education. *EURASIA Journal of Mathematics, Science and Technology Education*, *13*(2), 469–486 (cit. on p. 92).

Miech, A., Alayrac, J.-B., Bojanowski, P., Laptev, I., & Sivic, J. (2017). Learning from video and text via large-scale discriminative clustering. *ICCV* (cit. on pp. 26, 28).

Miech, A., Alayrac, J.-B., Smaira, L., Laptev, I., Sivic, J., & Zisserman, A. (2020). End-to-end learning of visual representations from uncurated instructional videos. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 35, 36, 70, 73, 74, 78, 84, 86, 91, 96, 103, 106, 114, 115, 117).

Miech, A., Zhukov, D., Alayrac, J.-B., Tapaswi, M., Laptev, I., & Sivic, J. (2019). Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (cit. on pp. 12, 19, 35, 40, 43, 45, 70, 71, 73, 78, 84, 85, 86, 96, 102, 114).

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 26* (pp. 3111–3119). Curran Associates, Inc. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf. (Cit. on p. 120)

Min, J., Lee, J., Ponce, J., & Cho, M. (2019). Hyperpixel flow: Semantic correspondence with multi-layer neural features. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (cit. on p. 94).

Misra, I., Gupta, A., & Hebert, M. (2017). From red wine to red tomato: Composition with context. *CVPR* (cit. on p. 53).

Misra, I., Zitnick, C. L., & Hebert, M. (2016). Shuffle and learn: Unsupervised learning using temporal order verifications. *ECCV* (cit. on pp. 71, 72, 80, 115).

Moulon, P., Monasse, P., & Marlet, R. (2012). Adaptive structure from motion with a contrario model estimation. *ACCV* (cit. on p. 94).

Moulon, P., Monasse, P., Perrot, R., & Marlet, R. (2016). OpenMVG: Open multiple view geometry. *International Workshop on Reproducible Research in Pattern Recognition*, 60–74 (cit. on pp. 94, 95).

Naing, Z., & Elhamifar, E. (2020). Procedure completion by learning from partial summaries. *BMVC* (cit. on p. 32).

Nguyen, P., Liu, T., Prasad, G., & Han, B. (2018). Weakly supervised action localization by sparse temporal pooling network. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 25).

Noroozi, M., & Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. *ECCV* (cit. on p. 72).

Pathak, D., Girshick, R., Dollar, P., Darrell, T., & Hariharan, B. (2017). Learning features by watching objects move. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 72).

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., & Efros, A. A. (2016). Context encoders: Feature learning by inpainting. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 72).

Paul, S., Roy, S., & Roy-Chowdhury, A. K. (2018). W-TALC: Weakly-supervised temporal activity localization and classification. *Proceedings of the European Conference on Computer Vision (ECCV)* (cit. on p. 25).

Plummer, B., Kordas, P., Kiapour, M., Zheng, S., Piramuthu, R., & Lazebnik, S. (2018). Conditional image-text embedding networks. *ECCV*, 258–274 (cit. on p. 96).

Pratt, P., Ives, M., Lawton, G., Simmons, J., Radev, N., Spyropoulou, L., & Amiras, D. (2018). Through the HoloLens™ looking glass: Augmented reality for extremity reconstruction surgery using 3D vascular models with perforating vessels. *European radiology experimental*, *2*(1), 1–7 (cit. on p. 92).

Qiu, Z., Yao, T., & Mei, T. (2017). Learning spatio-temporal representation with pseudo-3D residual networks. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 23).

Revaud, J., De Souza, C., Humenberger, M., & Weinzaepfel, P. (2019). R2D2: Reliable and repeatable detector and descriptor. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. (Cit. on pp. 106, 108).

Richard, A., & Gall, J. (2016). Temporal action detection using a statistical language model. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 31).

Richard, A., Kuehne, H., & Gall, J. (2017). Weakly supervised action learning with RNN based fine-to-coarse modeling. *CVPR* (cit. on pp. 27, 28, 29, 30, 31, 50, 74, 92, 96).

Richard, A., Kuehne, H., & Gall, J. (2018a). Action sets: Weakly supervised action segmentation without ordering constraints. *CVPR* (cit. on pp. 29, 30, 31, 62, 64, 92, 95).

Richard, A., Kuehne, H., Iqbal, A., & Gall, J. (2018b). Neuralnetwork-viterbi: A framework for weakly supervised video learning. *CVPR* (cit. on pp. 27, 29, 30).

Rocco, I., Arandjelović, R., & Sivic, J. (2018a). End-to-end weakly-supervised semantic alignment. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 94).

Rocco, I., Cimpoi, M., Arandjelović, R., Torii, A., Pajdla, T., & Sivic, J. (2018b). Neighbourhood consensus networks. *NeurIPS* (cit. on p. 94).

Rohrbach, M., Amin, S., Andriluka, M., & Schiele, B. (2012). A database for fine grained activity detection of cooking activities. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 1194–1201 (cit. on pp. 39, 40).

Russell, B. C., Martin-Brualla, R., Butler, D. J., Seitz, S. M., & Zettlemoyer, L. S. (2013). 3D wikipedia: Using online text to automatically label and navigate reconstructed geometry. *ACM* (cit. on pp. 96, 97, 109).

Sanabria, R., Caglayan, O., Palaskar, S., Elliott, D., Barrault, L., Specia, L., & Metze, F. (2018). How2: A large-scale dataset for multimodal language understanding. *Proceedings of the Workshop on Visually Grounded Interaction and Language (ViGIL). NeurIPS* (cit. on pp. 40, 42).

Schonberger, J. L., & Frahm, J.-M. (2016). Structure-from-motion revisited. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 91, 94, 95, 114).

Schüldt, C., Laptev, I., & Caputo, B. (2004). Recognizing human actions: A local svm approach. *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, *3*, 32–36. https://doi.org/10.1109/ICPR.2004.1334462 (cit. on p. 23)

Seguin, G., Bojanowski, P., Lajugie, R., & Laptev, I. (2016). Instance-level video segmentation from object tracks. *CVPR* (cit. on pp. 26, 28).

Sener, F., Saraf, R., & Yao, A. (2021). Learning video models from text: Zero-shot anticipation for procedural actions. *arXiv* (cit. on p. 38).

Sener, F., & Yao, A. (2018). Unsupervised learning and segmentation of complex activities from video. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 31, 32, 65, 72, 73, 92, 95).

Sener, F., & Yao, A. (2019). Zero-shot anticipation for instructional activities. *The IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 38).

Sener, O., Zamir, A. R., Savarese, S., & Saxena, A. (2015). Unsupervised semantic parsing of video collections. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on pp. 33, 34, 40, 41, 42, 48, 50, 52, 58).

Shan, D., Geng, J., Shu, M., & Fouhey, D. F. (2020). Understanding human hands in contact at internet scale. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 82, 103, 106).

Shen, Y., Wang, L., & Elhamifar, E. (2021). Learning to segment actions from visual and language instructions via differentiable weak sequence alignment. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10156–10165 (cit. on p. 35).

Shou, Z., Gao, H., Zhang, L., Miyazawa, K., & Chang, S.-F. (2018). AutoLoc: Weakly-supervised temporal action localization in untrimmed videos. *Proceedings of the European Conference on Computer Vision (ECCV)* (cit. on p. 25).

Sigurdsson, G. A., Gupta, A., Schmid, C., Farhadi, A., & Alahari, K. (2018). Actor and observer: Joint modeling of first and third-person videos. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 24).

Sigurdsson, G. A., Varol, G., Wang, X., Farhadi, A., Laptev, I., & Gupta, A. (2016). Hollywood in homes: Crowdsourcing data collection for activity understanding. *ECCV* (cit. on p. 23).

Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. *NIPS* (cit. on pp. 22, 48, 50).

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *ICLR* (cit. on p. 106).

Singh, K. K., & Lee, Y. J. (2017). Hide-and-Seek: Forcing a network to be meticulous for weakly-supervised object and action localization. *2017 IEEE International Conference on Computer Vision (ICCV)*, 3544–3553. https://doi.org/10.1109/ICCV.2017.381 (cit. on p. 25)

Snavely, N., Seitz, S. M., & Szeliski, R. (2006). Photo tourism: Exploring photo collections in 3D. *Siggraph* (pp. 835–846). (Cit. on p. 94).

Soomro, K., Zamir, A. R., & Shah, M. (2012). Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv* (cit. on p. 23).

Stein, S., & McKenna, S. J. (2013). Combining embedded accelerometers with computer vision for recognizing food preparation activities. *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing)* (cit. on pp. 40, 41).

Sun, C., Myers, A., Vondrick, C., Murphy, K., & Schmid, C. (2019). VideoBERT: A joint model for video and language representation learning. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (cit. on pp. 36, 73).

Sun, C., Shetty, S., Sukthankar, R., & Nevatia, R. (2015). Temporal localization of fine-grained actions in videos by domain transfer from web images. *Proceedings of the 23rd ACM international conference on Multimedia* (cit. on p. 25).

Sweeney, C. (n.d.). Theia multiview geometry library: Tutorial & reference. (Cit. on pp. 94, 95).

Tang, P., Wang, X., Wang, A., Yan, Y., Liu, W., Huang, J., & Yuille, A. (2018). Weakly supervised region proposal network and object detection. *Proceedings of the European Conference on Computer Vision (ECCV)* (cit. on p. 25).

Tang, Y., Ding, D., Rao, Y., Zheng, Y., Zhang, D., Zhao, L., Lu, J., & Zhou, J. (2019). Coin: A large-scale dataset for comprehensive instructional video analysis. *Pro-*

ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (cit. on pp. 40, 42, 43, 70, 71, 79, 114).

Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning spatiotemporal features with 3D convolutional networks. *ICCV* (cit. on p. 22).

Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., & Paluri, M. (2018). A closer look at spatiotemporal convolutions for action recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 23).

Truong, P., Danelljan, M., & Timofte, R. (2020). GLU-Net: Global-local universal network for dense flow and correspondences. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 106, 108).

Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Computer Architecture Letters*, *13*(04), 376–380 (cit. on p. 106).

Varol, G., Laptev, I., & Schmid, C. (2017). Long-term temporal convolutions for action recognition. *PAMI* (cit. on p. 22).

Wang, H., Kläser, A., Schmid, C., & Liu, C.-L. (2011). Action recognition by dense trajectories. *CVPR* (cit. on p. 21).

Wang, H., & Schmid, C. (2013). Action recognition with improved trajectories. *ICCV* (cit. on pp. 21, 48, 50).

Wang, L. [Limin], Qiao, Y., Tang, X., & Van Gool, L. (2016a). Actionness estimation using hybrid fully convolutional networks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 73).

Wang, L. [Limin], Xiong, Y., Lin, D., & Van Gool, L. (2017). UntrimmedNets for weakly supervised action recognition and detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 25).

Wang, L. [Limin], Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., & Van Gool, L. (2016b). Temporal segment networks: Towards good practices for deep action recognition. *ECCV* (cit. on p. 22).

Wang, L. [Liwei], Li, Y., Huang, J., & Lazebnik, S. (2019). Learning two-branch neural networks for image-text matching tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(2), 394–407. https://doi.org/10.1109/TPAMI.2018.2797921 (cit. on p. 96)

Wang, L. [Liwei], Li, Y., & Lazebnik, S. (2016). Learning deep structure-preserving image-text embeddings. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 96).

Wang, M., Azab, M., Kojima, N., Mihalcea, R., & Deng, J. (2016). Structured matching for phrase localization. *ECCV*, 696–711 (cit. on p. 96).

Wang, X., & Gupta, A. (2015). Unsupervised learning of visual representations using videos. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (cit. on p. 72).

Wang, X., Jabri, A., & Efros, A. A. (2019). Learning correspondence from the cycle-consistency of time. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 100).

Wei, D., Lim, J. J., Zisserman, A., & Freeman, W. T. (2018). Learning and using the arrow of time. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 71, 72).

Wu, C. (2013). Towards linear-time incremental structure from motion. *2013 International Conference on 3D Vision - 3DV 2013*, 127–134. https://doi.org/10.1109/3DV.2013.25 (cit. on pp. 94, 95)

Xiao, F., Sigal, L., & Jae Lee, Y. (2017). Weakly-supervised visual grounding of phrases with linguistic structures. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 96).

Xie, S., Sun, C., Huang, J., Tu, Z., & Murphy, K. (2018). Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. *Proceedings of the European Conference on Computer Vision (ECCV)* (cit. on p. 23).

Xu, D., Xiao, J., Zhao, Z., Shao, J., Xie, D., & Zhuang, Y. (2019). Self-supervised spatiotemporal learning via video clip order prediction. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 72).

Xu, L., Neufeld, J., Larson, B., & Schuurmans, D. (2004). Maximum margin clustering. *NIPS* (cit. on p. 52).

Yao, B., Jiang, X., Khosla, A., Lin, A. L., Guibas, L., & Fei-Fei, L. (2011). Human action recognition by learning bases of action attributes and parts. *ICCV* (cit. on p. 52).

Yatskar, M., Ordonez, V., Zettlemoyer, L., & Farhadi, A. (2017). Commonly uncommon: Semantic sparsity in situation recognition. *Proceedings of the CVPR* (cit. on p. 53).

Yu, S.-I., Jiang, L., & Hauptmann, A. (2014). Instructional videos for unsupervised harvesting and learning of action examples. *ACM MM* (cit. on p. 33).

Zhang, Q., & Li, B. (2011). Video-based motion expertise analysis in simulation-based surgical training using hierarchical Dirichlet process hidden Markov model. *Proceedings of the 2011 international ACM workshop on Medical multimedia analysis and retrieval* (cit. on p. 38).

Zhang, R., Isola, P., & Efros, A. A. (2016). Colorful image colorization. *The European Conference on Computer Vision (ECCV)* (cit. on p. 72).

Zhang, R., Isola, P., & Efros, A. A. (2017). Split-brain autoencoders: Unsupervised learning by cross-channel prediction. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 72).

Zhao, F., Li, J., Zhao, J., & Feng, J. (2018). Weakly supervised phrase localization with multi-scale anchored transformer network. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 96).

Zhao, Y., Xiong, Y., Wang, L., Wu, Z., Tang, X., & Lin, D. (2017). Temporal action detection with structured segment networks. *The IEEE International Conference on Computer Vision (ICCV)* (cit. on pp. 73, 79).

Zhou, B., Andonian, A., Oliva, A., & Torralba, A. (2018). Temporal relational reasoning in videos. *Proceedings of the European Conference on Computer Vision (ECCV)* (cit. on p. 71).

Zhou, L., Xu, C., & Corso, J. J. (2018a). Towards automatic learning of procedures from web instructional videos. *AAAI* (cit. on pp. 40, 41, 52, 58, 70, 96).

Zhu, L., & Yang, Y. (2020). ActBERT: Learning global-local video-text representations. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 36, 96).

Zhukov, D., Alayrac, J.-B., Cinbis, R. G., Fouhey, D., Laptev, I., & Sivic, J. (2019). Cross-task weakly supervised learning from instructional videos. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 19, 40, 70, 71, 72, 73, 79, 80, 84, 85, 86, 92, 96).

Zhukov, D., Alayrac, J.-B., Laptev, I., & Sivic, J. (2020). Learning actionness via long-range temporal order verification. *ECCV* (cit. on p. 19).

Zhukov, D., Rocco, I., Laptev, I., Sivic, J., Schönberger, J. L., Tekin, B., & Pollefeys, M. (2021). Reconstructing and grounding narrated instructional videos in 3D. *arXiv* (cit. on p. 19).

Zia, A., Sharma, Y., Bettadapura, V., Sarin, E. L., Clements, M. A., & Essa, I. (2015). Automated assessment of surgical skills using frequency analysis. *International Conference on Medical Image Computing and Computer-Assisted Intervention* (cit. on p. 38).

Zia, A., Sharma, Y., Bettadapura, V., Sarin, E. L., & Essa, I. (2018). *Video and accelerometer-based motion analysis for automated surgical skills assessment.* (Cit. on p. 38).

Zia, A., Sharma, Y., Bettadapura, V., Sarin, E. L., Ploetz, T., Clements, M. A., & Essa, I. (2016). *Automated video-based assessment of surgical skills for training and evaluation in medical schools.* (Cit. on p. 38).

RÉSUMÉ

Le but de cette thèse est de développer des méthodes pour la compréhension automatique des vidéos d'instructions, qui démontrent des tâches humaines, comme, par exemple, faire une omelette ou accrocher une peinture. Nous proposons, d'abord, une méthode d'apprentissage des actions seulement à partir d'un script pour chaque tâche, au lieu des annotations manuelles. Notre modèle permet de réduire la quantité de données d'entraînement, en partageant l'information entre les tâches. Nous évaluons notre approche sur un nouveau jeu de données, CrossTask. Nous présentons, ensuite, une méthode non supervisée pour isoler les actions, liée à une tâche de leur contexte. Finally, we learn to associate natural language instructions with the corresponding objects within the 3D scene, reconstructed from the videos. Finalement, nous proposons une approche pour associer des instructions textuelles avec des objets correspondants dans la scène 3D, reconstruite à partir des vidéos.

MOTS CLÉS

Vision par ordinateur Reconnaissance d'actions Compréhension de vidéos Apprentissage non supervisé Apprentissage faiblement supervisé Vidéos d'instruction

ABSTRACT

The goal of this thesis is to develop methods for automatic understanding of video content. We focus on instructional videos that demonstrate how to perform complex tasks, such as making an omelette or hanging a picture. First, we investigate learning visual models for the steps of tasks, using only a list of steps for each task, instead of costly and time consuming human annotations. Our model allows us to share the information between the tasks on the sub-step level, effectively multiplying the amount of available training data. We demonstrate the benefits of our method on a newly collected dataset of instructional videos, CrossTask. Next, we present a method for isolating task-related actions from the surrounding background, that doesn't rely on human supervision. Finally, we learn to associate natural language instructions with the corresponding objects within the 3D scene, reconstructed from the videos.

KEYWORDS

Computer vision Action recognition Video understanding Unsupervised learning Weakly supervised learning Instructional videos