HAL open science

Structured Learning from Videos and Language

Jean-Baptiste Alayrac

▶ To cite this version:

Jean-Baptiste Alayrac. Structured Learning from Videos and Language. Computer Vision and Pattern Recognition [cs.CV]. Ecole normale supérieure - ENS PARIS, 2018. English. NNT : . tel-01885412v1

HAL Id: tel-01885412

https://inria.hal.science/tel-01885412v1

Submitted on 1 Oct 2018 (v1), last revised 2 Sep 2021 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences Lettres – PSL Research University

préparée à l'École normale supérieure

Apprentissage structuré à partir de vidéos et langage

Structured Learning from Videos and Language

École doctorale n°386
Spécialité Informatique

Composition du Jury :

M Kevin Murphy
Google AI
Rapporteur

M Abhinav Gupta
Carnegie Mellon University
Rapporteur

M Josef Sivic
Inria
Directeur de thèse

M Simon Lacoste-Julien
Université de Montréal
Directeur de thèse

M Ivan Laptev
Inria
Directeur de thèse

M Armand Joulin
Fabebook AI Research
Membre du Jury

M Francis Bach
Inria
Membre du Jury

Mme Cordelia Schmid
Inria
Membre du Jury

soutenue par Jean-Baptiste Alayrac
le 17 Septembre 2018

ENS ÉCOLE NORMALE SUPÉRIEURE | PSL RESEARCH UNIVERSITY PARIS

Résumé

Le but de cette thèse est de développer des modèles, des représentations adaptées et des algorithmes de prédiction structurée afin de pouvoir analyser de manière automatique des activités humaines complexes commentées par du langage naturel.

Dans un premier temps, nous présentons un modèle qui, étant donné plusieurs vidéos tutorielles, est capable de découvrir quelle est la liste d'actions nécessaires à l'accomplissement de la tâche ainsi que de localiser ces actions dans le flux vidéo et dans la narration textuelle. Afin d'atteindre cet objectif, nous formulons deux hypothèses. La première est que les gens réalisent les actions au moment où ils les décrivent, *i.e.*, il y a une très forte corrélation temporelle entre le texte et la vidéo. La seconde hypothèse est que ces tâches complexes sont réalisées en suivant un ordre précis d'actions. Muni de ces deux hypothèses, notre modèle résout d'abord un problème de partitionnement dans le texte pour ensuite utiliser ces premiers résultats afin de guider la localisation des actions dans la vidéo. Notre modèle est évalué sur un nouveau jeu de données de vidéos tutorielles qui décrit 5 tâches complexes, telles que 'changer la roue d'une voiture' ou bien 'rempoter une plante'.

Nous proposons ensuite de relier les actions avec les objets manipulés. Plus précisement, on se concentre sur un type d'action particulière qui vise à modifier l'état d'un objet. Par exemple, cela arrive lorsqu'on *sert une tasse de café* ou bien lorsqu'on *ouvre une porte*. Ce type d'action est particulièrement important dans le contexte des vidéos tutorielles. Notre méthode consiste à minimiser un objectif commun entre les actions et les objets. Nous démontrons via des expériences numériques que localiser les actions aident à mieux reconnaître l'état des objets et inversement que modéliser le changement d'état des objets permet de mieux déterminer le moment où les actions se déroulent.

Tous nos modèles sont basés sur du partionnement discriminatif, une méthode qui permet d'exploiter la faible supervision contenue dans ce type de vidéos. Cela se résume à formuler un problème d'optimisation sous contrainte qui peut se résoudre aisément en utilisant l'algorithme de Frank-Wolfe qui est particulièrement adapté au type de contraintes envisagé. Motivé par le fait qu'il est très important d'être en mesure d'exploiter les quelques milliers de vidéos qui sont disponibles en ligne, nous portons enfin notre effort à rendre l'algorithme de Frank-Wolfe plus rapide et plus efficace lorsque confronté à beaucoup de données. En particulier, nous proposons trois modifications à l'algorithme Block-Coordinate Frank-Wolfe: un échantillonnage adaptatif des exemples d'entrainement, une version bloc des 'away steps' et des 'pairwise steps' initialement prévues pour l'algorithme original et enfin une manière de mettre en cache les appels à l'oracle linéaire.

Abstract

The goal of this thesis is to develop models, representations and structured learning algorithms for the automatic understanding of complex human activities from instructional videos narrated with natural language.

We first introduce a model that, given a set of narrated instructional videos describing a task, is able to generate a list of action steps needed to complete the task and locate them in the visual and textual streams. To that end, we formulate two assumptions. First, people perform actions when they mention them, *i.e.*, there is a strong temporal correlation between text and video. Second, we assume that complex tasks are composed of an ordered sequence of action steps. Equipped with these two hypotheses, our model first clusters the textual inputs and then uses this output to refine the location of the action steps in the video. We evaluate our model on a newly collected dataset of instructional videos depicting 5 different complex goal oriented tasks, such as changing car tire or repotting a plant.

We then present an approach to link actions and the manipulated objects. More precisely, we focus on actions that aim at modifying the *state* of a specific object, such as *pouring a coffee cup* or *opening a door*. Such actions are an inherent part of instructional videos. Our method is based on the optimization of a joint cost between actions and object states under constraints. The constraints reflect our assumption that there is a consistent temporal order for the changes in object states and manipulation actions. We demonstrate experimentally that object states help localizing actions and conversely that action localization improves object state recognition.

All our models are based on discriminative clustering, a technique which allows to leverage the readily available *weak supervision* contained in instructional videos. In order to deal with the resulting optimization problems, we take advantage of a highly adapted optimization technique: the Frank-Wolfe algorithm. Motivated by the fact that scaling our approaches to thousands of videos is essential in the context of narrated instructional videos, we also present several improvements to make the Frank-Wolfe algorithm faster and more computationally efficient. In particular, we propose three main modifications to the Block-Coordinate Frank-Wolfe algorithm: gap-based sampling, away and pairwise Block Frank-Wolfe steps and a solution to cache the oracle calls. We show the effectiveness of our improvements on four challenging structured prediction tasks including foreground/background segmentation and human pose estimation.

Acknowledgement

I would first like to thank my great PhD advisors: Josef, Ivan and Simon. Josef, I really enjoyed your plain enthusiasm for research and computer vision. I learnt a lot from your great sense of details. Ivan, I will always remember your great support even until the last deadline minutes. Your commitment for doing good research has pushed me to surpass myself, I will always be grateful for that. Simon, last but not least, I am very happy to had the luck to work with you during these 4 years. From the black board to the climbing walls, you have always been a great teacher. The three of you formed a very complimentary team to which I owe a lot. Thanks!

I thank Kevin Murphy and Abhinav Gupta for agreeing to review my thesis. I also thank Armand Joulin, Francis Bach and Cordelia Schmid for being part of my committee.

I would also want to thank Jean Ponce and Francis Bach for having created such an amazing place to work. Even if I was not directly working with you, I learned a lot from our numerous interactions. I am very proud to have been a member of the mythic WILLOW and SIERRA teams.

I thank Andrew Zisserman to welcome me at Deepmind. I look forward to working with you to explore new topics and never stop learning.

I was lucky enough to collaborate with many great people during my thesis. These collaborations were a very important source of motivation for me. Piotr, thanks a lot for teaching me from the beginning of my PhD the art of rigour which is essential for doing good science. Always great to share a beer with you! Anton, I really enjoyed working with you, our ICML project was a great occasion to learn new things, thank you. Antoine, it was also very nice to work with you, thank you as well for now taking care of sequoia, good

luck! Rémi, I am very happy to have met you. You started as a colleague, you are now a friend. Our collaboration was amazing and I am sure it won't be the last one now that we've embarked together on new adventures. Guilhem, thanks a lot for offering me this last PhD project. It was so much fun to work with you. Finally, thank you Dmitry! I am convinced you are going to do a great PhD.

The lab would not have been the same without all its great members. I really enjoyed sharing moments with all of you: Sofiane Allayen, Relja Arandjelovic, Mathieu Aubry, Dmitry Babichev, Raphael Berthier, Nicolas Boumal, Margaux Bregere, Andrei Bursuc, Lénaic Chizat, Minsu Cho, Théophile Dalens (C408 crew!), Alexandre d'Aspremont, Alexandre Defossez, Rémy Degenne, Vincent Delaitre, Aymeric Dieuleveut, Christophe Dupuy, Mihai Dusmanu,Thomas Eboli, Loïc Esteve, Nicolas Flammarion, Fajwel Fogel, Pierre Gaillard, Damien Garreau, Pascal Germain, Gauthier Gidel, Robert Gower, Yana Hasson, Bumsub Ham, Igor Kalevatykh, Vadim Kantorov, Thomas Kerdreux, Sesh Kumar, Vijay Kumar, Suha Kwak, Yann Labbé, Rémi Lajugie, Loic Landrieu, Zongmian Li, Maxime Oquab, Dmitrii Ostrovskii, Fabian Pedregosa, Julia Peyre, Loucas Pillaud Vivien, Anastasia Podosinnikova, Antoine Recanati, Rafael S. Rezende, Ignacio Rocco, Vincent Roulet, Allessandro Rudi, Damien Scieur, Guillaume Seguin, Nino Shervashidze, Tatiana Shpakova, Robin Strudel (C408 RPZ), Adrien Taylor, Matthew Trager, Gül Varol, Tuan-Hung Vu and Sergey Zagoruyko.

Je me permets maintenant de passer à la belle langue de Molière pour remercier mes proches qui m'ont soutenu à l'extérieur du labo.

Je commence par toi Vincent qui a toujours été présent pour moi que ce soit au labo où à l'extérieur. Nous avons partagé de nombreux moments ensemble. Que ce soit dans les joies ou dans les difficultés, tu as toujours été un soutien très fort pour moi. Je te suis énormément reconnaissant pour cela. Merci beaucoup l'ami!

Merci à tous mes copains et copines qui m'ont permis de m'évader un peu de ma recherche! Entre les colocs d'Alésia et de Picpus, les séances de grimpe, les jeux de sociétés, les weekend à la montagne, les dégustations de whisky,

les soirées parisiennes et mon EVG de l'espace, vous m'avez offert un oxygène incroyable!

Merci à mes parents pour m'avoir appris l'essentiel. Merci à mes soeurs Camille et Marie et leur belles petites familles!

Enfin merci à celle qui a partagé ces longues années de thèse à mes côtés. Malgré les nombreuses deadlines, mes horaires quelque peu décalés et ma passion pour la recherche parfois un peu trop débordante tu as quand même décidé de signer pour la vie! Merci pour être toi!

Finalement, je dédie cette thèse à mon frère Paul ainsi qu'à mes grand parents.

Contents

1 Introduction **1**

1.1 Goal . . . 1

1.2 Motivation . . . 3

1.3 Challenges . . . 4

1.4 Contributions . . . 7

1.4.1 Learning from narrated instructional videos . . . 7

1.4.2 Manipulation actions as change of object states . . . 8

1.4.3 Large-scale optimization techniques . . . 9

1.5 Outline . . . 9

1.6 Publications . . . 10

2 Related work: vision and language **12**

2.1 People and objects . . . 12

2.1.1 Estimating human cues . . . 12

2.1.2 Object detection . . . 15

2.1.3 Object states . . . 17

2.1.4 Human-object interaction . . . 18

2.2 Action recognition in videos . . . 19

2.2.1 Representing videos for action recognition . . . 19

2.2.2 Datasets . . . 22

2.3 Text and video . . . 22

2.3.1 Text as supervision . . . 22

2.3.2 Video captioning . . . 24

2.3.3 Video-Text alignment . . . 26

2.4 Composite activities . . . 27

2.4.1 Composite activities in language . . . 27

2.4.2 Composite activities from video and text . . . 29

2.5 Instructional videos . . . 31

2.5.1 Step discovery and localization . . . 31

2.5.2 Reasoning about objects . . . 34

2.5.3 Datasets 36

3 Background in machine learning **38**
3.1 Supervised learning 38
3.1.1 What is supervised learning? 39
3.1.2 What are the limitations of supervised learning? 40
3.2 Alternatives to supervised learning 41
3.3 Discriminative clustering 43
3.3.1 Framework 44
3.3.2 DIFFRAC 44
3.3.3 Applications with weak supervision 46
3.4 Frank-Wolfe algorithm 47
3.4.1 The algorithm 47
3.4.2 Properties 48
3.4.3 The variants 50
3.4.4 Why is it relevant for discriminative clustering? 50

4 Learning from narrated instructional videos **52**
4.1 Introduction 53
4.2 Related work 55
4.3 New dataset of instructional videos 57
4.3.1 Dataset statistics 58
4.4 Modelling narrated instructional videos 60
4.4.1 Clustering transcribed verbal instructions 61
4.4.2 Discriminative clustering of videos under text constraints 66
4.5 Experimental evaluation 73
4.5.1 Implementation details 73
4.5.2 Results of multiple sequence alignment 75
4.5.3 Results of step discovery from text narrations 76
4.5.4 Results of localizing instruction steps in video 77
4.5.5 Parameter analysis 81
4.5.6 Joint clustering of video and text 84
4.6 Conclusion and future work 85

5 Joint Discovery of Object States and Manipulation Actions **86**
5.1 Introduction 86
5.2 Related work 88
5.3 Modeling manipulated objects 91

5.3.1 Discovering object states 92
5.3.2 Action localization 94
5.3.3 Linking actions and object states 94
5.4 Optimization 95
5.4.1 Relaxation 95
5.4.2 Joint optimization using Frank-Wolfe 96
5.4.3 Linear oracle for object state constraints 96
5.4.4 Joint rounding method 100
5.5 Experiments 102
5.5.1 Dataset and features 102
5.5.2 Weakly supervised object state discovery 104
5.5.3 Object state discovery in the wild 109
5.6 Conclusion and future work 111

6 Minding the Gaps for Block Frank-Wolfe Optimization 113
6.1 Introduction 114
6.2 Background 116
6.2.1 Structured Support Vector Machine (SSVM) 116
6.2.2 Block Coordinate Frank-Wolfe (BCFW) 117
6.2.3 Duality gap 118
6.2.4 Convergence of BCFW 119
6.3 Block gaps in BCFW 120
6.3.1 Adaptive non-uniform sampling 120
6.3.2 Pairwise and away steps 125
6.3.3 Caching 128
6.4 Experiments 132

7 Discussion 136
7.1 Summary of contributions 136
7.2 Perspectives 137
7.2.1 Action step discovery 138
7.2.2 Reasoning about objects and actions 139
7.2.3 Learning and Optimization 139

A Appendix of Chapter 6 141
A.1 Block descent lemma for BCFW 141
A.2 Toy example for gap sampling 142
A.2.1 General idea 142

A.2.2 Explicit construction 143
A.3 Proof of Theorem 2 (convergence of BCFW with gap sampling) 148
A.4 Proof of Theorem 3 (convergence of BCPFW and BCAFW) . . 151
A.5 Proof of Theorem 4 (convergence of BCFW with caching) . . . 158
A.6 BCFW for SSVM with box constraints 161
A.6.1 Problem with box constraints 161
A.6.2 Optimizing w.r.t slack variables 163
A.6.3 Batch setting: optimization w.r.t iterate 164
A.6.4 The block-coordinate setting 166
A.7 Dataset description 167
A.7.1 OCR 167
A.7.2 CoNLL 168
A.7.3 HorseSeg 169
A.7.4 LSP 171

Bibliography **174**

Introduction 1

1.1 Goal

Our objective is to develop models, representations and algorithms for the automatic understanding of complex human activities narrated with natural language. In particular our goal is to discover action steps required to achieve complex tasks. To enable scaling to many tasks, we wish to avoid manual supervision and to use readily available data only such as narrated instructional videos on YouTube. Examples of such video are illustrated in Figure 1.1. A preview of the output of the method designed for that problem – explained in Chapter 4 – is provided in Figure 1.2.

The first focus is therefore to develop models able to cope with the structure of the two input modalities: the visual stream and the textual narration. In addition, these new models should only rely on the weak supervision contained in the observed real data. This supervision is indeed weak, because no precise temporal boundaries delimiting actions are provided in these videos. Often, the list of action steps itself is not given explicitly. Instead, one needs to infer that directly from the instructions given by the demonstrator.

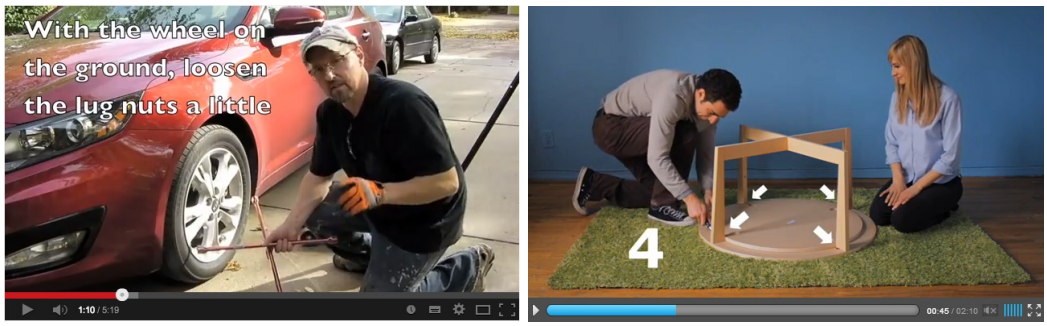


Fig. 1.1.: Example frames from videos depicting complex activities achieving a certain task. **Left:** Changing a tire of the car. **Right:** Assembling an IKEA table. Note the natural language instructions complementing the video. Left:"*With the wheel on the ground, loosen the lug nuts a little.*" Right: "*Install cross brace on the underside of each connection point.*"

Fig. 1.2.: In Chapter 4, given a set of narrated instructional videos of the same task (here 'changing a car tire'), we present a method that recovers the list of action steps required to achieve that task (**left**) and that localizes these action steps in the input videos (**right**).

Second, we want to design appropriate representations for instructional videos. As a consequence, we investigate representations of object states and associated action steps within a joint model. An illustrative output of the method presented in Chapter 5 is given in Figure 1.3. This stands in contrast to other work on action recognition with no explicit notion of objects.

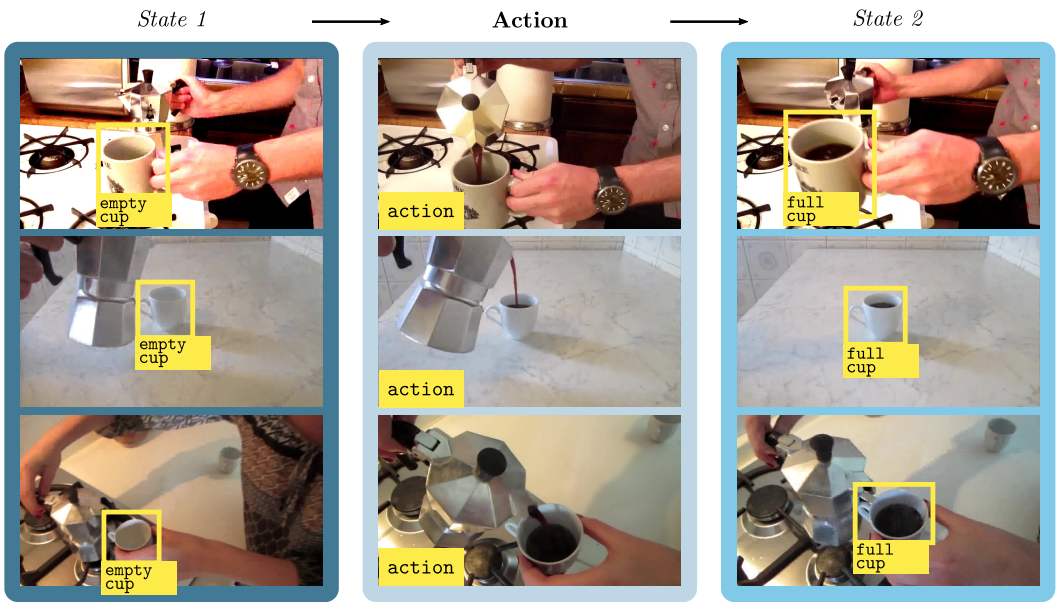


Fig. 1.3.: In Chapter 5, we automatically discover object states such as empty/full coffee cup along with their corresponding manipulation actions by observing people interacting with the objects. Note that the method does not have access to semantic annotation, hence names of states (empty/full) are assigned manually to clarify the illustration.

Finally, we also want to develop algorithms that can scale to a very large number of videos. We estimate there exist instruction videos for tens of thousands of different tasks, and on average around a hundred videos per task are available on YouTube. As these videos usually last a few minutes, this represents several years of footage.

1.2 Motivation

Instructional videos are a popular and widely used resource on the web. For example, querying 'How to change a car tire?' on YouTube alone results in more than 300,000 hits. Websites such as WikiHow[1] contain more than ten thousands different tasks covering various domains, such as 'car maintenance', 'home repair', 'gardening', 'cooking'. Hence, instructional videos represent a rich source of diverse information that is readily available on the web.

Many of those videos are made by amateurs to share their passion and/or maximize the number of views without any incentive to provide supervision for computer vision algorithms. This motivates us to develop algorithms that can work with few or no annotations and that can easily scale up to thousands of videos. Automatic methods that structure this data would enable better indexing in order to help users find the exact information they need.

Most importantly, why should computers learn and understand our activities? One motivation with a wide potential impact is automatic guidance. Imagine a virtual assistant, for example in the form of smart glasses such as Microsoft Hololens, that guides children through simple games to improve their manipulation and language skills or that helps people accomplish unfamiliar tasks. Such visual intelligence capabilities may enable, for example, constructing smart assistant that automatically learn new skills by just observing people as illustrated in Figure 1.4. The main challenges underlying such an approach are presented in the next Section.

[1] https://www.wikihow.com

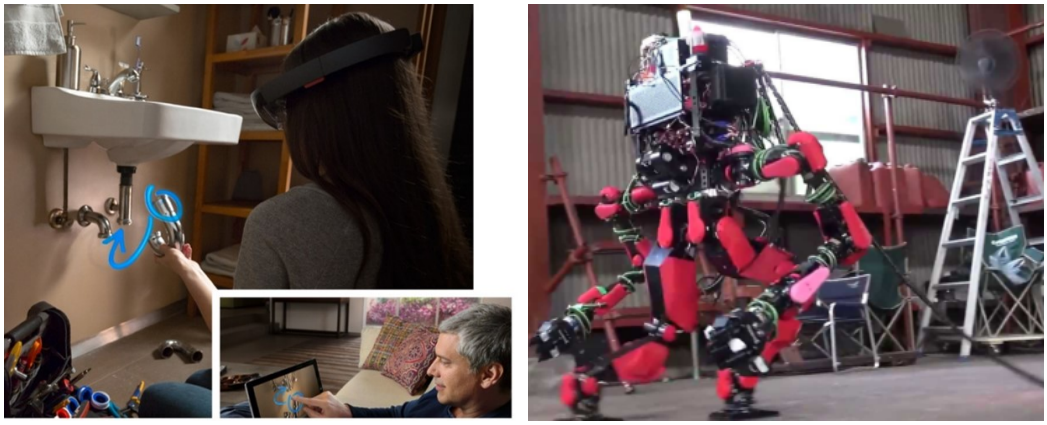


Fig. 1.4.: **Left:** with the recent advance of augmented reality, one could imagine having an automated assistant that could guides people step by step to accomplish unfamiliar tasks. What if this automatic assistant could rely on readily available YouTube data to give better advices? **Right:** a robot from the Darpa challenge. Could these robots directly learn new skills from human demonstrations?

1.3 Challenges

This thesis is addressing key challenges in both visual recognition and machine learning. From the visual recognition perspective, the main challenges are in modeling the structure of the two input modalities. For example, what is the appropriate granularity of representation of both the visual signal and the natural text? How to model sequences of actions that achieve a certain goal? What is the appropriate representation of the relation between the visual and linguistic modalities? From the machine learning perspective, the difficulty lies in learning the structured models with only weak supervision that is from the real-world, incomplete and noisy language and video data. Formalizing learning with weak supervision is a major open research problem. Below, we illustrate some of these challenges in detail.

Noisy natural language. Directly working with the audio signal from the videos is very difficult. A standard method to alleviate this issue is to use Automatic Speech Recognition (ASR) algorithms that turns the audio narration into text. Web platforms such as YouTube already provide this kind of data, which is particularly useful. However, the generated text comes with challenges. First, some words are misspelled. These errors are particularly common on technical words that are crucial in the context of instructional

videos. Second, no punctuation is provided which makes advanced standard Natural Language Processing (NLP) techniques more difficult to use.

Appearance variation. We have to face various factors of appearance variation when dealing with narrated instructional videos. In the textual domain, they are many ways how to express the same action. In the visual domain, there are all the standard challenges such as: difference of viewpoints, illumination changes, various appearance of objects. Finally, some difficulties are intrinsically linked to instructional videos. For example, people may follow different recipes to achieve the same task or use their own technique to perform an important step. This is illustrated in Figure 1.5.

Source of noise in the data. The data we want to use comes from the real world. It is generated by amateurs without any motivations to be used as training data for computer vision algorithms. This is interesting because no bias has been introduced by a pre-defined collection process, however it comes with a set of challenges. People may talk about irrelevant things, such as making jokes in order to attract more viewers. They are usually not professionals, and hence may use non fixed camera setting with poor quality recording device. The beginning and end of such videos may be composed of credits clips that are not relevant for instructions. This different sources of noise make the problem even more challenging. Examples illustrating these different sources of noise are provided in Figure 1.6.

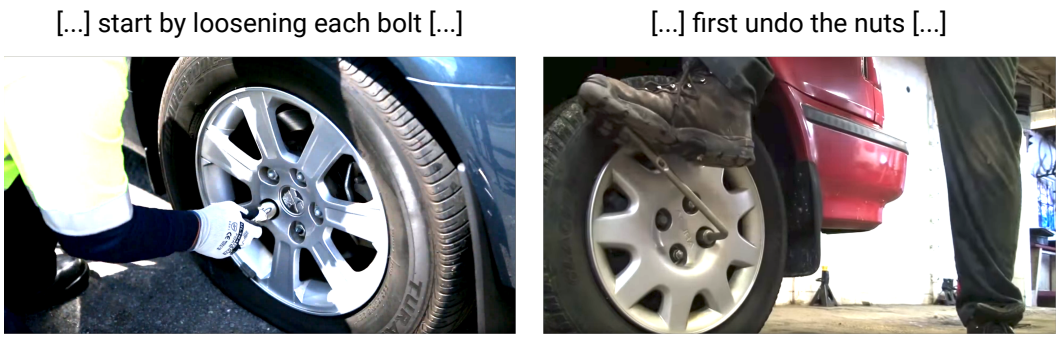


Fig. 1.5.: Challenges arise from different source of variability in instructional videos. First, there are variations in the way people are talking about actions. For example, the same action can be described in very different ways, e.g. 'start by loosening each bolt' (left) or 'first undo the nuts' (right). The variability is also strong in the visual domain due to several factors of variation such as viewpoint, illumination or the way people are performing each step. In this figure we show two examples of the same action step with large variation in visual appearance and textual descriptions.

Visual and textual alignment. We expect the narrator to give spoken instructions that are correlated with what he is showing in the video. However this correlation may not be perfect. First, it is never totally synced as people usually talk about something before actually doing it. Second, people may omit talking about something as they think it is enough to show it. Conversely, they might only orally mention something as they consider this step as being optional. Finally, it is also common to mention important objects and actions as an introduction of the video and then only later refer back to them. An example of such misalignment is given in Figure 1.7.

Learning at scale. As mentioned earlier, a very large amount of instructional videos is available on the web. These videos usually last several minutes, which is quite long when compared to other standard large-scale video datasets [Carreira and Zisserman, 2017] which are usually composed of few

(a) Many videos start with an introduction with some context (e.g. 'Hi, today I am going to talk about ...') that is not relevant for learning instructions.

(b) The object manipulation may not be well visible due to poor framing.

(c) Videos are often shot with a hand-held camera.

(d) Some videos contain credits at the beginning or at the end.

Fig. 1.6.: Typical families of noise encountered in instructional video data. Figures show frames extracted from instructional videos downloaded from the YouTube, typical of what we are focusing on in this thesis. See individual captions for details.

second clips. This requires algorithms that can handle this scale while dealing with few or no annotations.

1.4 Contributions

The contributions of this thesis are threefold. First, we introduce a model for action steps in instructional videos. Second, this model is extended to allow a joint representation of manipulation actions and objects states. Finally, we propose improvements to the speed of the Block Coordinate Frank-Wolfe (BCFW) algorithm [Lacoste-Julien *et al.*, 2013] that can be used to learn models in this thesis. These contributions are outlined in Sections 1.4.1-1.4.3 below.

1.4.1 Learning from narrated instructional videos

Given a set of narrated instructional videos describing the same *task*, such as *changing a car tire*, we propose a model to automatically discover the main **action steps** composing that task, such as **jacking up the car** or **removing the wheel**. In details, we propose an unsupervised method based on discriminative clustering that takes advantage of the complementary nature of

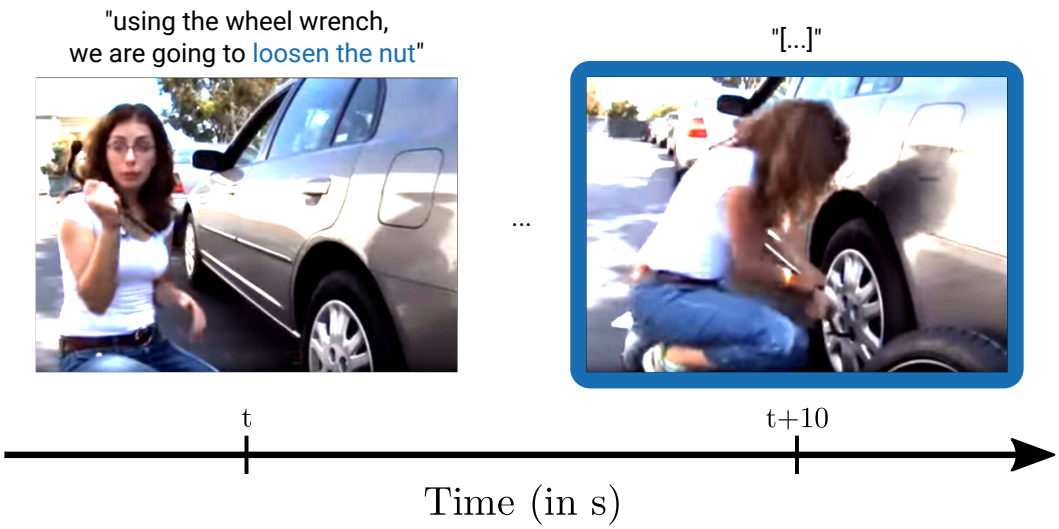


Fig. 1.7.: Challenges arise from the non perfect alignment of the visual and textual stream. As commonly observed, people often mention an action before actually doing it as it is the case here. This misalignment can be up to several seconds.

the input video and associated narration. The problem is formulated as two clustering problems, one in text and one in video, applied one after the other and linked by joint constraints to obtain a single coherent sequence of steps in both modalities.

We also introduce a new dataset of instructional videos for five different tasks (change a car tire, perform CardioPulmonary resuscitation (CPR), jump car's battery, repot a plant and make coffee). The ground truth used for evaluation consists in a list of steps for each task and precise temporal boundaries for these steps in each video. This dataset includes complex interactions between people and objects that are captured in a variety of indoor and outdoor settings.

1.4.2 Manipulation actions as change of object states

Many actions in instructional videos involve manipulating objects aiming to modify a property of the object (e.g. *fill a coffee cup*, *beat eggs*, *close the door*). We refer to these properties as object states. Motivated by this observation, we design a joint model for actions and object states whose goal is to better localize both actions and objects. More precisely, we introduce a method that, given a set of videos depicting a specific action, learns to identify object states and to temporally localize state-modifying actions. Our model is formulated as a discriminative clustering cost with constraints. We assume a consistent temporal order for the changes in object states and manipulation actions, and introduce new optimization techniques to learn model parameters without additional supervision.

To evaluate the effectiveness of our method, we introduce a new dataset with seven classes of manipulating actions temporally localized and object states localization. This kind of dataset is of primary importance for assessing the performance of different approaches.

1.4.3 Large-scale optimization techniques

The two previous contributions build on discriminative clustering algorithms, which typically involve solving a quadratic program (the data term) over linear constraints (the prior over our problem). Over the past few years, the Frank-Wolfe algorithm has become a common method for solving this kind of optimization problems. The main underlying motivation is that the Frank-Wolfe algorithm is well-suited to optimize over the difficult sets of constraints (e.g. ordering constraints) encountered in instructional videos.

Despite this nice property, the Frank-Wolfe approach is a batch algorithm and hence struggles when dealing with large-scale datasets. A nice alternative, called Block Coordinate Frank-Wolfe (BCFW), was recently proposed [Lacoste-Julien *et al.*, 2013] in order to address that concern. In this thesis, we propose three key improvements to the BCFW algorithm for making it faster. The key intuition behind our improvements is that the estimates of block gaps maintained by BCFW reveal the block suboptimality that can be used as an adaptive criterion. First, we sample objects at each iteration of BCFW in an adaptive non-uniform way via gap-based sampling. Second, we incorporate pairwise and away-step variants of Frank-Wolfe into the block-coordinate setting. Third, we cache oracle calls with a cache-hit criterion based on the block gaps. In order to validate our improvements, we conduct a thorough empirical evaluation on four classical structured prediction datasets for the structured SVM problem.

1.5 Outline

This manuscript is organized into seven chapters, including this introduction.

In Chapter 2, we review previous work in computer vision and natural language processing most related to this thesis. The review includes methods for action recognition, video representations, people-object interactions and work that relates vision and text. This naturally leads to the field of interest in the thesis, instructional videos, that is at the intersection of all these domains.

In Chapter 3, we discuss previous work in machine learning and optimization used in this thesis. It notably concerns the different forms of learning in order to highlight the weakly-supervised learning setup used in our work. We also review literature on Frank-Wolfe optimization techniques.

In Chapter 4, we present our first contribution, namely, a model for discovering action steps given narrated instructional videos for a task. Given only narrated instructional videos of the same task, our algorithm finds a list of steps needed to achieve the task and finds the occurrence of these steps both in the input narration and the input video stream.

In Chapter 5 we focus on an essential part of instructional videos: the link between objects and actions. To that end, we propose a novel joint objective that models the effect of action through a change of state of manipulated objects. Finally, we propose to use language that comes from Automated Speech Recognition to automatically find clips were manipulation of objects happens and then apply our method to them.

In Chapter 6, we present an optimization technique that allows our models to scale to a large number of videos. We build upon the Block Coordinate Frank Wolfe algorithm [Lacoste-Julien *et al.*, 2013] and propose a list of improvements that speed up this algorithm.

Finally, in Chapter 7, we provide a summary of our contributions and discuss limitations of this work together with open problems.

1.6 Publications

The first version of the work presented in Chapter 4 appeared in CVPR'16 [Alayrac *et al.*, 2016a], and an extended version was published in PAMI'17 [Alayrac *et al.*, 2017a]. The work described in Chapter 5 was presented in ICCV'17 [Alayrac *et al.*, 2017b]. The material introduced in Chapter 6 appeared in ICML'16 [Alayrac *et al.*, 2016b].

Collaborations have also been conducted during this thesis. In particular, a follow up of [Alayrac *et al.*, 2016b] that enabled BCFW techniques for

discriminative clustering was introduced in ICCV'17 [Miech *et al.*, 2017]. Recently, we also leverage our BCFW improvements for weakly supervised spatio-temporal action detection [Chéron *et al.*, 2018]. Finally, another work concerning structured prediction also appeared in ICLR'18 [Leblond *et al.*, 2018]. However, we do not describe these contributions in this thesis.

2 Related work: vision and language

In this chapter, we review the recent work that is most related to this thesis in the domains of computer vision and natural language processing. We describe important building blocks for instructional videos such as human and object interaction (Section 2.1), action recognition (Section 2.2) and efforts to relate text and vision (Section 2.3). In Section 2.4, we describe related work that models composite activities. Finally in Section 2.5, we review the main topic of interest in this thesis: instructional videos. The purpose of this section is to explain to the reader the context in which this new field is emerging and to justify why it is an important challenge for the community.

2.1 People and objects

People and objects are an inherent part of instructional videos. The detection of human cues such as the pose of the narrator or the identification of the objects used are key in order to better understand and analyze the events that take place. In this section, we review recent work in that domain.

2.1.1 Estimating human cues

Obtaining human cues from images is an important focus of research. This includes work on faces, hands, and full body pose recognition that we review next. Such cues are important for the domain of instructional videos as the narrator is manipulating objects in the scene. Thus, having access to where the person is in the video or in which pose their hands are can help to better capture what is happening.

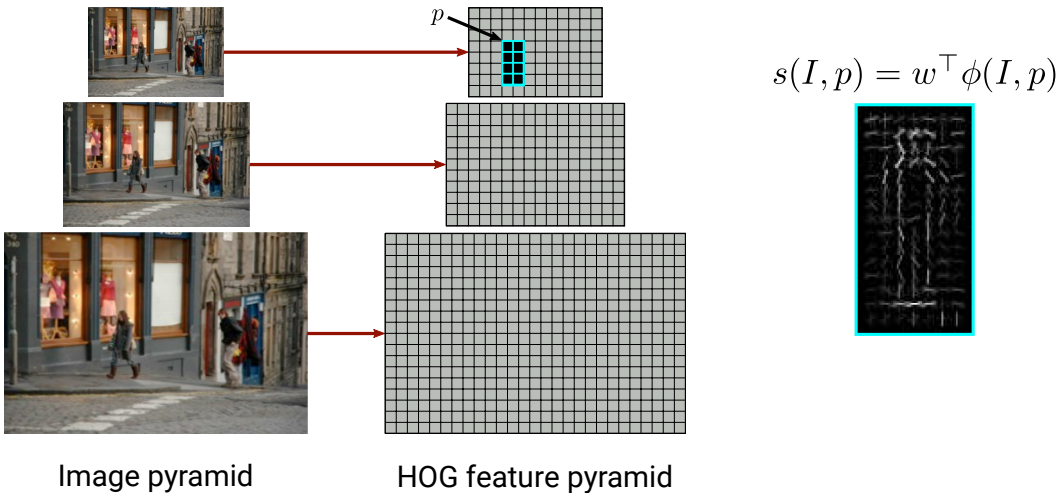
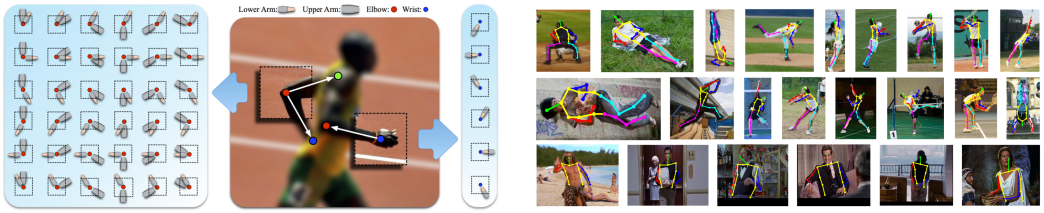


Fig. 2.1.: Illustration of the HOG based human detector of [Dalal and Triggs, 2005]. Given an image, HOG features are computed at different resolutions to obtain a feature pyramid (left). A model w (displayed on the right), is then run exhaustively on all p different positions at all scales (in a sliding window fashion) to output candidate detections that are filtered with NMS. This model is learned on training images containing human bounding boxes (positive) and negative samples using SVM. Illustration taken from the CS231B Stanford course [Girshick, 2013].

Face detection and recognition. One of the first successful method able to detect objects in real time is the work of Viola and Jones, [2001]. It was originally designed to detect faces. It consists in training weak classifiers using AdaBoost type algorithm on cropped images over simple Haar-like features. This classifier is then run in an exhaustive manner on a test image. The method does so through an integral image, which coupled with a cascading technique, enables fast computation. Many face detection approaches have been inspired by this technique [Zhang and Zhang, 2010]. Most recent work for face detection leverages advances in generic object detection that we will review next. In addition to detection, researchers have also focused on recognition. Among this long line of research, the open source Deep Face method [Parkhi *et al.*, 2015] makes use of a large labeled dataset to train a convolutional neural network (CNN) for recognizing faces. This approach is currently the state-of-the-art for face recognition.

Human detection and pose estimation. Detecting full human body is also an important focus of research due to its numerous applications (autonomous driving, video surveillance...). One of the most seminal work in that area is the one of Dalal and Triggs, [2005]. In this paper, the authors demonstrate that using Histograms of Oriented Gradients (HOG) is effective for human



Main intuition Sample results

Fig. 2.2.: Illustration of the human pose detector from [Chen and Yuille, 2014]. **Left**: The main intuition behind the method is that localization of joint (*e.g.*elbow) can help localize neighboring regions (*e.g.*, wrist). The proposed approach does so by learning a set of possible configurations for pairs of joints. A graphical model ensures that the predictions are coherent with this set of plausible configurations. **Right**: sample results obtained by the method on test images. Figure from [Chen and Yuille, 2014].

detection. In a nutshell, their method consists in computing HOG features on the full image at different scales. Then, a simple SVM model is applied in order to score every subwindow at each scale. Non-maxima suppression (NMS) is finally applied to remove overlapping detections. We provide an illustration of this process in Figure 2.1. The idea of using HOG features has been at the core of multiple successful techniques such as Deformable Part Models [Felzenszwalb *et al.*, 2010]. Many benchmarks exist for that task and again most recent successful work takes advantage of advances in generic object detection that we will discuss next. See [Nguyen *et al.*, 2016] for a recent survey. Another important focus of research is pose estimation, which consists of detecting the body's joints. The classical methods try to fit a tree like model to the human body [Andriluka *et al.*, 2009; Yang and Ramanan, 2011]. Thanks to the recent advances of deep learning and the successes of convolutional neural networks, a lot of progress has been made over the past few years.

Chen and Yuille, [2014] propose to couple a graphical model with a CNN representation to both detect joints and also estimate their pairwise relationships. The main intuition behind this method is that localizing a joint (e.g. the elbow) can help locate neighboring joint (e.g. the wrist) as illustrated in Figure 2.2. Toshev and Szegedy, [2014] introduce the idea of directly regressing the 2D positions of the joints with a CNN. The successful Convolutional Pose Machine method [Wei *et al.*, 2016] works by making a sequence of predictions which become more and more accurate with the iterations. This approach is an alternative to explicit modeling of part relationships.

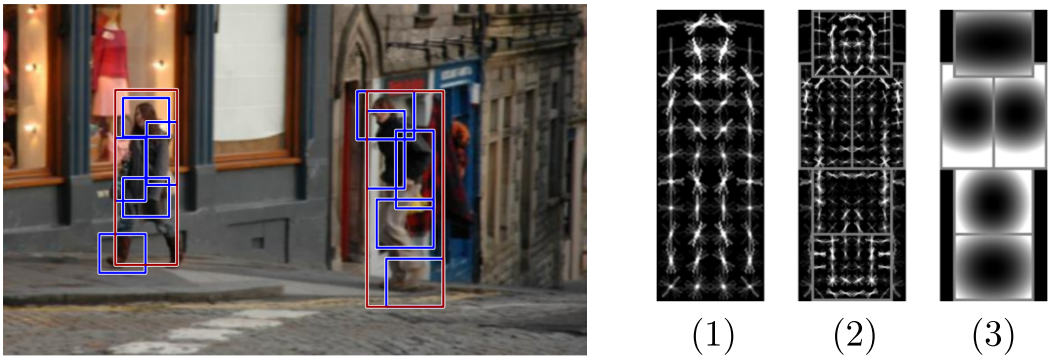


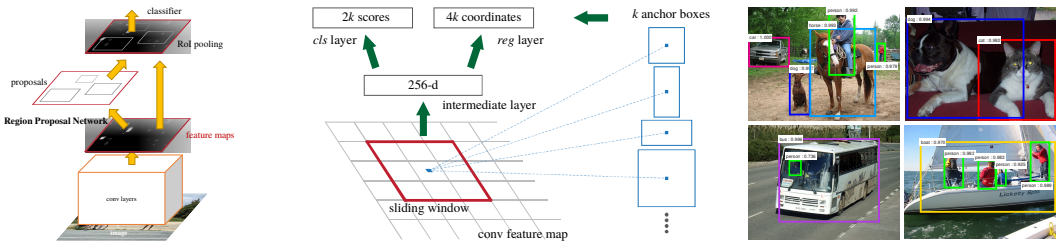
Fig. 2.3.: Illustration from [Felzenszwalb *et al.*, 2010] of the Deformable Part Model. **Left.** Result detections obtained with a person model. **Right.** The model is characterized by a coarse root filter (1), multiple part filters (2) and a prior on the spatial configuration of the different parts (3).

In [Papandreou *et al.*, 2017], the authors combine a multi-person detection method with a 2-D pose estimation technique in order to obtain an effective system for pose estimation in the wild.

Work on hands. Hands are important in the context of instructional videos as a large proportion of shots consist of close up on hands manipulating objects. Most of the existing works that enable hand detection or hand pose estimation have been developed in the context of egocentric videos and often require depth camera such as the Kinect device [Taylor *et al.*, 2016]. This line of work is particularly suited for gesture recognition but is not really adapted to scenarios encountered when dealing with third person YouTube videos. The work of Mittal *et al.*, [2011] introduces a dataset and a method for detecting hands and their orientation in single RGB images. Simon *et al.*, [2017] leverage the panoptic system [Joo *et al.*, 2015] to obtain a hand detector and pose estimator that takes as input only RGB images and is comparable with methods that uses depth sensors in terms of performance.

2.1.2 Object detection

Outside human cue estimation, significant efforts have been made to detect, localize and estimate properties of objects. Objects are a key part of instructional videos as people use tools to modify states of objects. For example, a wheel wrench can be used to tighten the bolts of a car wheel. Hence, being able to detect and recognize objects is crucial to reach the goal of this thesis. Object detection consists in localizing and classifying the object. The two stan-



Faster RCNN Network | Region Proposal Network | Outputs

Fig. 2.4.: Illustration from [Ren *et al.*, 2015] of the Faster RCNN model. **Left**: End-to-end trainable network for object detection. **Middle**: Illustration of the Region Proposal Network, a module that outputs candidate regions in the image that are potential objects. **Right**: Sample detections from the faster RCNN method on the PASCAL VOC 2007 test set.

dard datasets for that task are Pascal VOC [Everingham *et al.*, 2010] which contains 20 categories and the more recent COCO dataset [Lin *et al.*, 2014] which depicts 91 classes. Before the deep learning era, the state-of-the-art technique was the Deformable Part Models (DPM) approach [Felzenszwalb *et al.*, 2010]. Its main idea is based on part-based models [Fischler and Elschlager, 1973; Felzenszwalb and Huttenlocher, 2000] and consists in decomposing an object into parts which have a local appearance template and can deform with respect to each other. A score for the candidate object is obtained by combining appearance and geometric configuration scores. The DPM combines this technique with the Dalal and Triggs, [2005] detector presented earlier. More specifically it obtains scores of a part-based model of an object in a sliding window like manner on a full image. Parts are not specified in the training images. They are instead discovered thanks to the use of a latent SVM training algorithm. We provide an illustration of the DPM approach in Figure 2.3. This method has been recently outperformed by the advances in deep learning. In particular, Girshick *et al.*, [2014] introduces the RCNN method which simply consists in first obtaining object proposals with an off-the-shelf method (such as [Krähenbühl and Koltun, 2014]), and then resizing these proposals to a fixed sized to later classify them with a convolutional neural network. Girshick, [2015] later introduces Fast-RCNN, based on a same principle as RCNN, which additionally leverages the fact that neighboring windows in an image share computation in order to significantly fasten the inference. Ren *et al.*, [2015] proposes to discard the object proposal phase and instead introduce an end-to-end method which can adapt to specific object categories to further improve speed and performance. This is done by introducing a 'Region Proposal Network' illustrated in Figure 2.4.

More recently, He *et al.*, [2017] introduce Mask RCNN which goes further by additionally enabling object segmentation. This method is the current state-of-the-art on the COCO benchmark.

2.1.3 Object states

During an instructional video, objects typically change states while being manipulated. Examples include a door being opened, a nut being loosened or a candle being lightened. Prior work has tackled the problem of recognizing attributes [Farhadi *et al.*, 2009] of an object which can be interpreted in some cases as object states. This line of work often relies on a predefined list of attributes and does not consider the dynamics of the state change. Isola *et al.*, [2015] propose to discover object states and their transformations by analyzing large collections of still images annotated with their states as illustrated in Figure 2.5. On the video side, Wang *et al.*, [2016] propose to model actions as changes of state of the environment at a global level

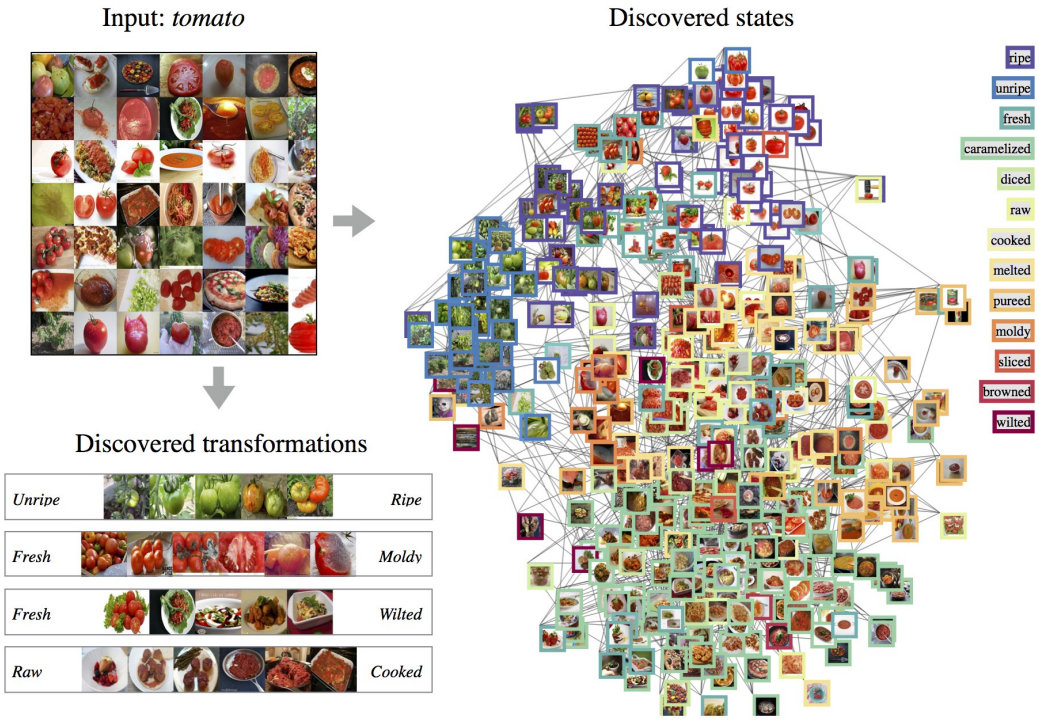


Fig. 2.5.: Illustration from [Isola *et al.*, 2015]. Example input and output obtained with their approach: given a set of images from one class (top-left, here for the tomato category), they cluster the images into various states (right). In addition, continous transformations between opposite states are discovered (bottom-left).

without explicitly trying to locate and identify objects. Fathi and Rehg, [2013] also introduce a method that models actions as changes of object states in the domain of egocentric video. It works by using cues such as gaze, or hand segmentation mask which are not always easy to obtain in challenging third person YouTube videos. We discuss object states in more detail in Chapter 5.

2.1.4 Human-object interaction

The link between human and objects is also crucial for the kind of problems considered in this thesis. The work of Gupta *et al.*, [2009a] demonstrates that jointly modeling object and human actions can benefit both action recognition and object prediction. In particular, the authors show that objects can help disambiguate actions by adding more context and vice versa as shown in Figure 2.6. Co-occurrence of object and actions has also shown benefits for action recognition in still images [Delaitre *et al.*, 2011]. In [Delaitre *et al.*, 2012] and [Fouhey *et al.*, 2014], the authors exploit the way people interact with objects to extract information about a scene, for example what objects are present (semantic) or what is the 3D geometry of the scene. Damen *et al.*, [2014] use consistent manipulation in egocentric videos to discover

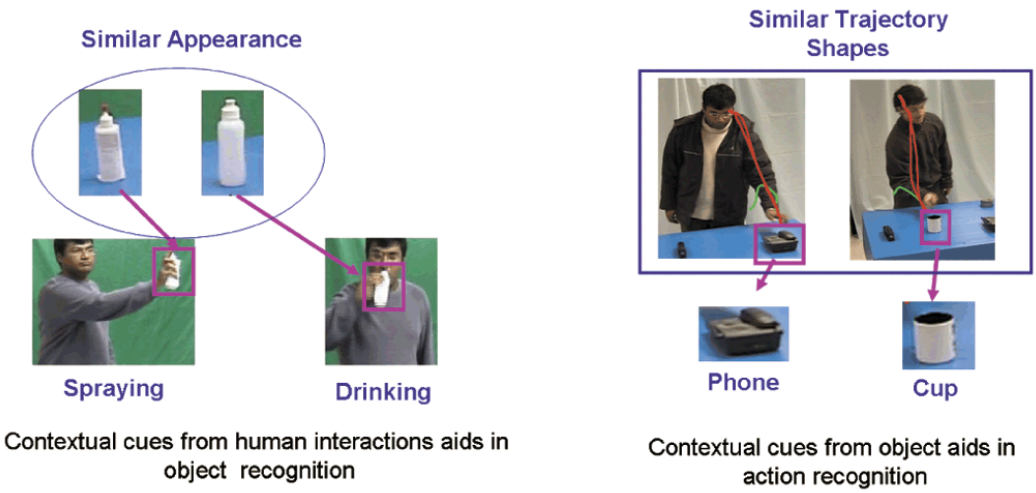


Fig. 2.6.: Illustration from [Gupta *et al.*, 2009a]. The authors show that human-object interaction can help improve both object and action recognition. **Left:** Different actions (*e.g.spraying* or *drinking*) can help disambiguate similar looking objects (*e.g.*a *spray can* and a *bottle*). **Right:** Conversely, actions that are similar in terms of movement (*e.g.answering a phone* and *grabbing a cup*) can be distinguished thanks to object cues.

task relevant objects. An important line of work [Desai *et al.*, 2010; Yao and Fei-Fei, 2010; Prest *et al.*, 2012; Peyre *et al.*, 2017] focuses on predicting visual relationship between human and objects in the form of a triplet (*person*, *action*, *object*), such as '*man holding bike*' or '*man wearing hat*'. Such abilities are essential to build systems able to describe complex scenes. Recently, Goyal *et al.*, [2017] introduce the 'Something Something' dataset which contains a large collection of videos of objects being manipulated. It constitutes a relevant source of data to explore human-object interaction in videos.

2.2 Action recognition in videos

Representing portions of a video to differentiate between various actions such as '*screw*' or '*pour*' is essential for instructional videos. Compared to still images, dynamic videos present a new challenge because of their additional dimension: time. In this section, we discuss some of the most important landmarks in the video action recognition domain. We first discuss the methods and later describe the different datasets that are available for training and evaluating action recognition models.

2.2.1 Representing videos for action recognition

Laptev, [2005] proposes to extend the notion of image interest points that are only spatial to the case of videos which belong to the spatio-temporal domain. Interest points are local regions depicting an important variation in terms of intensities. They were first successfully used in images for several applications such as stereo matching or object recognition. Laptev, [2005] extends the Harris and Förstner interest point detectors [Harris and Stephens, 1988; Förstner and Gülsch, 1987] with a temporal aspect. It works by maximizing a function obtained through convolution of the image with Gaussian filters at different scales. For the video case, Laptev, [2005] introduces separate parameters for the spatial and temporal Gaussian filters. Once these video interest points are obtained, local, spatio-temporal and scale-invariant features are extracted to represent the event.

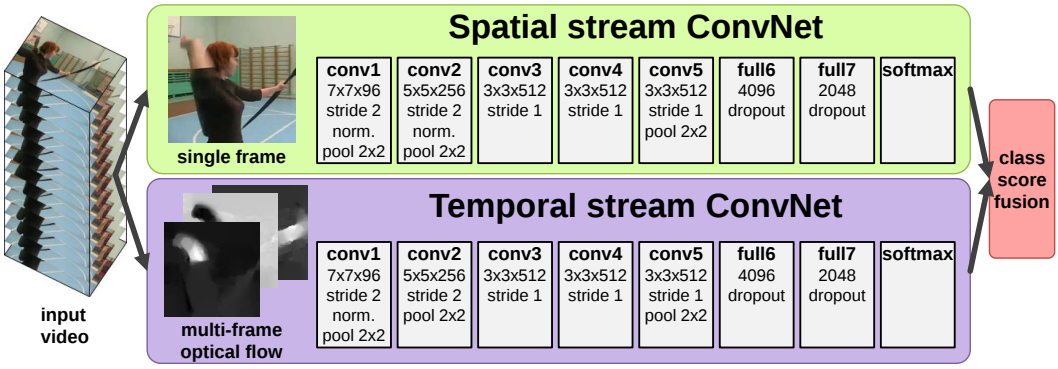


Fig. 2.7.: Two-stream network architecture for action recognition. Figure from [Simonyan and Zisserman, 2014].

Wang *et al.*, [2011] introduce dense trajectories that also build on the idea of describing local spatio-temporal volumes. However the principle to extract such regions is different. Instead of detecting spatio-temporal interest points, Wang *et al.*, [2011] densely sample points in all the frames and then later track these points using optical flow. These tracks, denominated trajectories, are then described by different features to capture shape, appearance and motion information. The authors observe that among all the descriptors, the motion boundary histograms (MBH), seem to perform the best due to their robustness against camera motion. Using this observation, Wang and Schmid, [2013] propose Improved Dense Trajectories (IDT) that estimate camera motion in order to explicitly remove its effects on the feature representation. Thanks to this simple improvement, they obtain significant improvements over the state-of-the-art on multiple action benchmarks.

Due to the astonishing performance of deep models for image classification, many efforts have been made to design deep models for action recognition over the past few years. These methods rely on large action video datasets that typically contain short clips annotated with action labels. We will discuss these datasets next. Among these works, we can distinguish two main branches.

The first one is the 'two stream' approach [Simonyan and Zisserman, 2014]. The idea is to have two convolutional neural networks, one to capture motion between frames (through optical flow) and the other to model visual appearance (through RGB images). The scores of the two networks are fused at the end of the two networks (late fusion) in order to predict the action class. The input of the appearance network is a single RGB image sampled

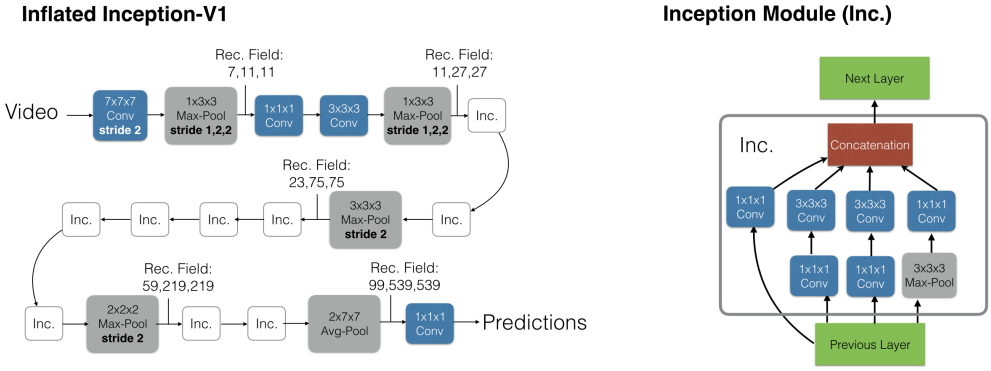


Fig. 2.8.: I3D architecture for action recognition [Carreira and Zisserman, 2017].

from the clip being classified. The input of the motion network is a stack of multiple optical flow images computed over the video. This process is illustrated in Figure 2.7.

The second line of work consists in generalizing convolutional neural networks used for images to the case of video. This is done by learning 3D convolutional kernels to account for the additional temporal dimension. One of such works is the method described in [Tran *et al.*, 2015], where the authors propose a simple architecture with small $3 \times 3 \times 3$ kernels which performs well on two standard action recognition benchmarks.

Despite the use of deep learning, these methods often benefits from being coupled with the standard IDT [Wang and Schmid, 2013] technique. More recently, Carreira and Zisserman, [2017] introduce a method that combines the two previous ideas by proposing a network that contains two streams (optical flow and RGB images) with 3D convolutional kernels. More specifically they introduce the Two-Stream Inflated 3D ConvNet (I3D) architecture (see Figure 2.8) which allows to be initiated from 2D networks pretrained on image classification task through a procedure called inflation. Equipped with this network, this initialization procedure and a new collected video dataset, the authors demonstrate clear superior performance on the HMDB [Kuehne *et al.*, 2011] and UCF101 [Soomro *et al.*, 2012] benchmarks. In particular, the need for combining this representation with more standard methods seems no longer necessary.

2.2.2 Datasets

All the previous advances would not have been possible without the existence of video datasets for action recognition. Below we review some of them. The Hollywood2 dataset [Marszalek *et al.*, 2009] contains around 4000 videos depicting 12 human action classes in 10 types of scenes. The HMDB dataset [Kuehne *et al.*, 2011] has around 7000 videos and 51 action classes. Later, Soomro *et al.*, [2012] introduce the UCF101 dataset containing more than 13K videos for 101 action classes. Recently, the scale of video dataset has increased significantly notably with the Kinetics dataset [Carreira and Zisserman, 2017] which contains more than 300K clips and allowed significant improvements of deep learning techniques [Carreira and Zisserman, 2017].

2.3 Text and video

Narrated instructional videos are composed of a visual stream and a textual stream (the narration). Hence, it is natural to combine these two modalities to better capture the specificity of this type of data. Prior work has also attempted to work with text and video for various applications. In this section, we give a quick overview by describing some relevant approaches. We start by describing methods that use available text as a source of supervision, then we mention the task of video captioning which consists in generating textual description given a video. To finish we present related work that aligns visual and textual streams.

2.3.1 Text as supervision

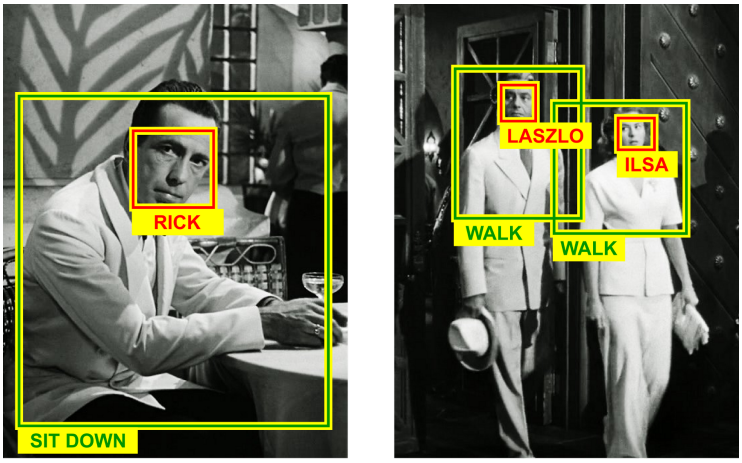
Text is often considered being simpler than visual signal. This is due to the fact that text consists of a sequence of discrete symbols that belong to a known dictionary. A video is instead composed of a sequence of high dimensional continuous images. Using this observation, prior work uses text as a way to supervise learning of a video model. One of the main advantages is that text is often readily available to describe video content and can therefore be considered as a cheap alternative to human defined labels.

In the domain of Internet (Youtube) videos, recent work [Yang and Toderici, 2011] has demonstrated promising results on predicting simple video-level tags. They build their training set in an automatic fashion by directly using the tags provided by YouTube users.

Other text sources for video supervision are movie scripts and subtitles. Movie scripts contain scene descriptions and dialogues that are not temporally aligned with the movie. Indeed, they are usually produced before the movie is actually shot and edited. On the contrary, subtitles contain temporal information and are nowadays easily available on the web.

Everingham *et al.*, [2006] first propose to align script dialogues and subtitles of the TV show 'Buffy, the vampire Slayer' in order to perform face recognition. By doing so, the identity of the speaking character can be extracted. Everingham *et al.*, [2006] leverage that information to obtain a visual model of each character and to detect their appearance in the videos. In addition to the character extraction from the text, the method relies on tracking frontal face detections and speaker identification through lip motion estimation. Sivic *et al.*, [2009] further improve the character detection performance through better visual techniques (kernel face descriptors, profile face detection) while using the same textual processing. More recent works have also looked at this problem such as [Cour *et al.*, 2009] or [Bojanowski *et al.*, 2013] which uses a discriminative clustering approach to disambiguate between characters, a method that we also leverage in this thesis.

In addition to face recognition, prior work has also looked at obtaining supervision for detecting actions from movie scripts. Laptev *et al.*, [2008a] propose to train a simple text SVM classifier based on n-gram representation to extract the portion of the movie script that describes a given action. Thanks to the temporal alignment with the videos, a clip corresponding to the detected text can then be extracted and used as a training sample to learn an action classifier. Because annotating text is easier than annotating action temporal boundaries, the conclusion drawn by this paper was very promising at the time. Marszalek *et al.*, [2009] further extended this paper by improving the text method and most of all introduced a larger movie dataset that constitutes the well known Hollywood2 dataset mentioned earlier. While these methods only provide clip level labels, Duchenne *et al.*,



Rick sits down again and stares off in their direction. Ilsa and Laszlo leave the cafe.

Fig. 2.9.: Illustration of the results obtained by Bojanowski *et al.*, [2013]. Given the weak supervision contained in the *Casablanca* movie script, the method automatically detects characters (*e.g.*Rick, Laszlo and Ilsa) and their actions (*e.g.walk* or *sit down*). Figure from Bojanowski *et al.*, [2013].

[2009] propose to obtain better temporal boundaries without additional supervision. They do so by using a discriminative clustering method in order to detect most representative temporal segments of each retrieved clip. They demonstrate that better modeling temporal boundaries of action enables better test detection performance. Finally, Bojanowski *et al.*, [2013] also show that joint models of actors appearance and their actions can be learned from movies with shooting scripts as shown in Figure 2.9 for the movie *Casablanca*.

2.3.2 Video captioning

Driven by the recent advances in image captioning [Vinyals *et al.*, 2014], others have looked at video captioning. Given a video as input, the goal is to generate a textual description of its content. Applications of such methods include video description for visually impaired. Such systems can also enable better video retrieval techniques by estimating how probable a given description is for a particular video. The text and video representations developed in video captioning literature may be relevant for the domain of narrated instructional videos considered in this thesis.

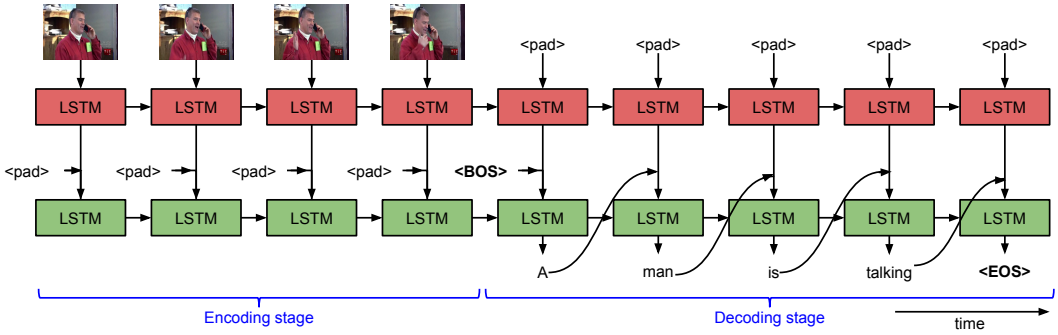


Fig. 2.10.: Illustration of the video captioning system from [Venugopalan *et al.*, 2015a]. A first LSTM (in red) takes as input a video as a sequence of frames (encoding stage) and transmits its outputs to a second LSTM (in green) which outputs a sentence describing the video (decoding stage). Figure from [Venugopalan *et al.*, 2015a].

The most successful recent methods are inspired by Neural Machine Translation and, in particular, the encoder-decoder model. To translate a sentence from a source to a target language, the input sentence is first *encoded* and then *decoded* in the form of the output sequence. In the application considered here, video is considered as being the source and text as being the target. In [Venugopalan *et al.*, 2015b], the authors use a simple encoding scheme by averaging the visual features of the frames composing the videos. A Long Short Term Memory recurrent neural network (LSTM) is used for decoding (to produce the caption). This approach only works for relatively short videos and cannot capture causal events due to the simple averaging scheme. To alleviate this issue other works have proposed to also use recurrent neural networks for the encoding [Venugopalan *et al.*, 2015a] (see Figure 2.10) while others have proposed to use attention mechanisms on top of 3D convolutional features [Yao *et al.*, 2015].

Video captioning is difficult to evaluate as there is currently no good automatic method to say wheter or not a description is relevant for a video. Current evaluation metrics are BLEU or Meteor scores which are used in language and are based on a n-gram distance between ground truth and generated sentences. To address this issue, others also consider other text-video tasks that are easier to evaluate such as video question answering [Tapaswi *et al.*, 2016] or video text retrieval, e.g. with the LSMDC challenge [Rohrbach *et al.*, 2015].

2.3.3 Video-Text alignment

Instructional videos often follow a recipe (ordered textual description of steps) to demonstrate a particular task. It is therefore important to be able to align a visual stream with a textual description. In this section, we review some of the recent approaches that tackle that problem.

Related to our approach presented in Chapter 4, is the work of Bojanowski *et al.*, 2014. In this work, the authors propose to align videos with an ordered list of action classes which belong to a predefined finite dictionary (which can be seen as a simple form of language). The method relies on discriminative clustering [Bach and Harchaoui, 2007], which enables to learn an alignment model without specific alignment annotation. During training, a linear classifier over some predefined visual features is learned. At inference time, dynamic programming is used to find the best alignment between the video and the action list according to the action classifier. In [Bojanowski *et al.*, 2015], the authors further extend that model to align video to a generic sequence of continuous vectors, hence enabling alignment with free form language which can be represented by word embedding vectors. This is illustrated in Figure 2.11.

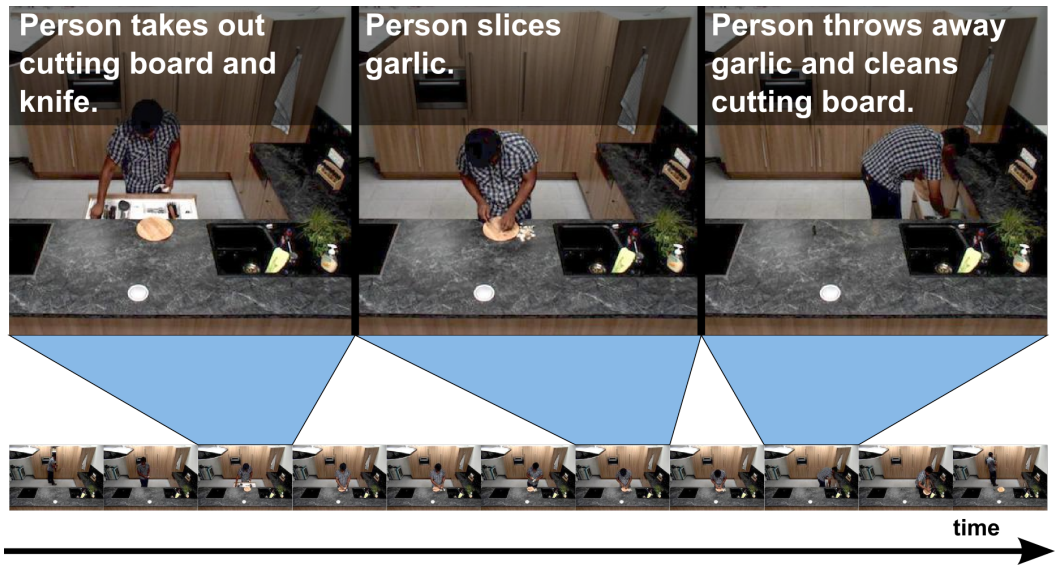


Fig. 2.11.: Output of the method of [Bojanowski *et al.*, 2015] for the task of video-to-text alignment on a cooking video from the TACoS dataset [Regneri *et al.*, 2013].

More recent work have also tackled the problem of aligning videos to a sequence of discrete symbols. In [Huang *et al.*, 2016], the authors introduce the Extended Connectionist Temporal Classification (ECTC) framework. Similar to Bojanowski *et al.*, 2014, they assume that no alignment is provided at training time, instead it has to be inferred during training. Instead of considering the best alignment according to their model ($\max$) they consider all possible alignments (Σ) in order to compute and maximize the conditional probability of the sequence of actions given the video content. They demonstrate that their model can significantly benefit from a few labeled videos by showing that their method is on par with fully supervised approaches with only 1% of the frames being annotated. Richard *et al.*, [2017] propose a method that iteratively finds the alignment between a video and a list of actions with a recurrent neural network.

2.4 Composite activities

Tasks described in instructional videos are inherently composite, i.e. they are composed of multiple actions applied one after the other in order to achieve a goal. Therefore, there exists a causal structure linking the different action steps, the agent and the objects. This structure can take different names depending on the application (script, recipe, scenario, graph of events...). It can answer questions like: what should I do next? how many steps are left? In this section, we describe some prior work in natural language analysis and visual recognition that have tackled the problem of composite activities. The goal is often to first discover the structure relating the actions (*e.g.*, the script) and then use this structure to make more accurate predictions. These works cover various application domains such as cooking, shopping or sport.

2.4.1 Composite activities in language

Chambers and Jurafsky, [2008] propose to discover what they call '*narrative event chains*' from unstructured large-scale news corpora. A narrative event chain is described as being composed of a set of events (typically a verb and its participants) that are partially ordered and centered around a protagonist (the subject). The partial order is defined by a simple binary

relation between events saying whether the first event is *before* the second one or if their relationship is different (simultaneous, undetermined...). The proposed approach is composed of three steps. First, a set of events and their protagonist are extracted from the text. Similarity scores between events are also computed with an unsupervised method. Second, the partial order between events is predicted through a two-stage approach. During the first stage, features are extracted for each event (tense, grammatical aspect...). For the second stage, a supervised temporal classifier is applied on top of these features to predict the *before* relation between events. This classifier is trained on the Timebank Corpus. Finally, during the third step the events are clustered using agglomerative clustering into coherent discrete narrative event chains. This work is a good example of how to extract structured information from a raw source of data.

Later Regneri *et al.*, [2010] notice that many scripts describing day-to-day activities contain a lot of implicit knowledge that cannot be obtained from corpora such as the one used in [Chambers and Jurafsky, 2008]. Instead, they collect a dataset containing descriptions of everyday tasks composed of a sequence of sentences. These descriptions are made by non-specialists through the Amazon Mechanical Turk system. Descriptions are collected for 22 different tasks such as '*eating in a fast food restaurant*' or '*making scrambled eggs*'. The annotators have to describe the task with at least 5 and at most 16 steps but no precise constraint is enforced. In order to discover a directed graph describing the task, the authors first employ Multiple Sequence Alignment (MSA) to jointly align all descriptions of the same task. To use MSA, the authors design a similarity metric adapted to their data (based on WordNet distance between verbs and nouns of different step's descriptions). Then, the authors extract a 'Temporal Script Graph' from the output of the MSA through an ad hoc procedure. The nodes of the graph represent actions (or events) and the oriented edges stand for the causal links between actions. See Figure 2.12 for an illustration of a 'Temporal Script Graph' for the '*eating in a fast food restaurant*' task. This is related to what we do in Chapter 4, except that we apply this technique on less structured textual narrations obtained from readily available Youtube videos.

More recently, Frermann *et al.*, [2014] introduce a hierarchical Bayesian model to address the application introduced in [Regneri *et al.*, 2010]. Instead

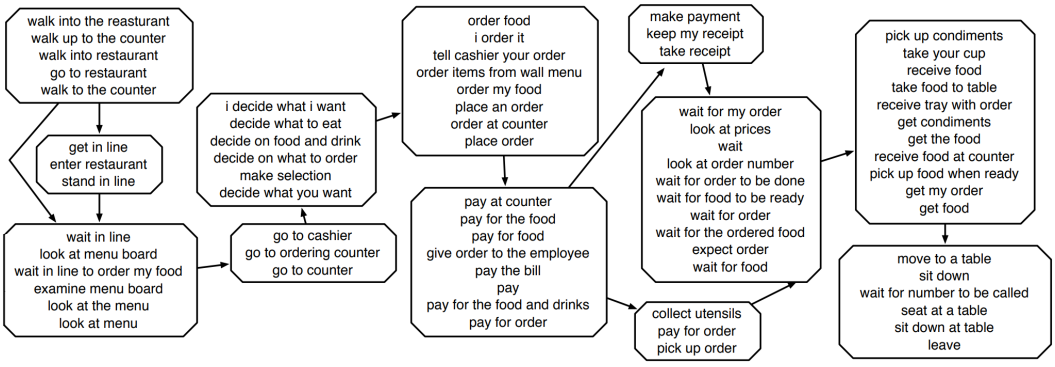


Fig. 2.12.: Illustration of a 'Temporal Script Graph' for the 'Eating in a fast-food restaurant' task extracted with the method of Regneri *et al.*, [2010].

of learning the events and deduce the ordering through a two stage approach as in [Regneri *et al.*, 2010], Frermann *et al.*, [2014] propose to solve both problems jointly. This is done with a generative model based on the Generalized Mallows Model which enables to favor solutions with a linear ordering of the action steps. With this model, they improve performance by 7% when compared with [Regneri *et al.*, 2010] for the task of recovering the correct ordering between events.

2.4.2 Composite activities from video and text

More related to our work are the approaches that extract composite activities text and video.

In [Gupta *et al.*, 2009b], the authors consider sports videos accompanied with linguistic captions. Given that, they aim to discover what is the causal structure of actions under the form of an AND-OR graph G. The oriented edges of the graph represent causal links between actions. The nodes of the graph describe actions which are represented by their type (e.g. 'Pitch' or 'Throw' for a baseball game) and their agent (e.g. 'Pitcher' or 'Fielder'). The OR relation is used to model the fact that one action can lead to mutually exclusive effects (e.g. after a 'Pitch' the batter can either 'Miss' or 'Hit'). The AND link models the case where one action can cause more than one simultaneous effect (e.g. after a 'Hit' the fielder *and* the batter will 'Run'). An illustration of the approach is provided in Figure 2.13. Along with that graph, the authors also learn parameters Θ that describe the appearance of actions and their spatio-temporal relationships. The graph G and the parameters

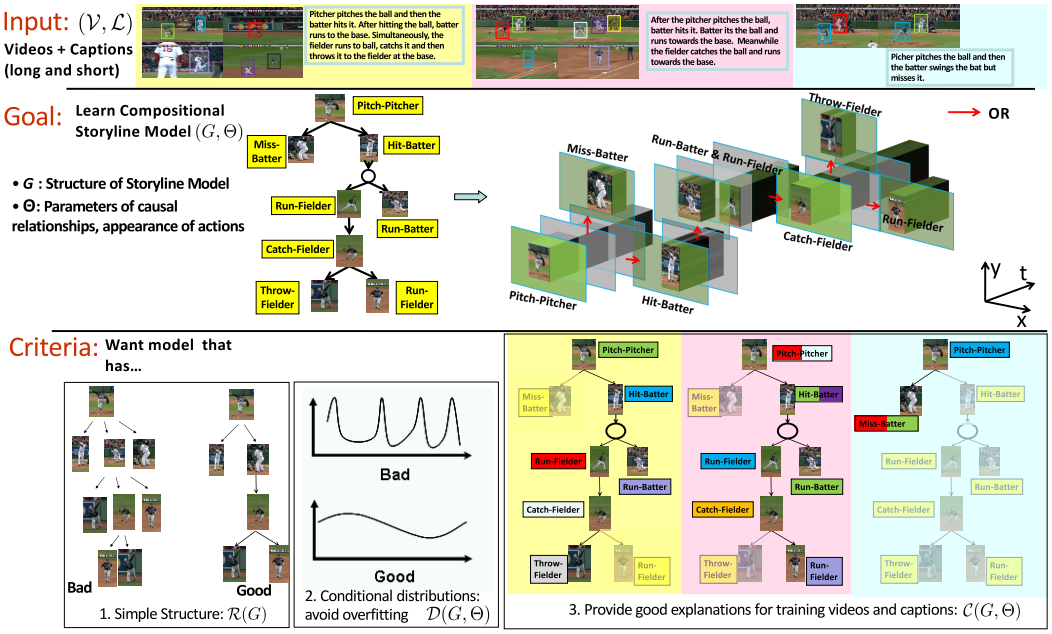


Fig. 2.13.: Illustration of the method of Gupta *et al.*, [2009b] which generates a storyline (represented by a AND-OR graph) given sport videos annotated with text.

Θ are used to match the actions to human tracks in the input videos. The learning procedure works by iterating between fixing the structure of the graph (learning G) and fitting the input videos to the graph (optimizing θ). The method relies on a good heuristic initialization. The experimental results are promising as they show that using the learned structure allows to obtain better assignment of human tracks to actions.

Later, [Rohrbach *et al.*, 2012] propose to leverage the concept of composite activities in the context of cooking videos described by scripts. In this work, the authors decide not to look at the temporal and causal structure of the composite activity (corresponding to a specific cooking recipe) but instead only look for the presence/absence of so called attributes. These attributes correspond to either atomic events (such as 'cut') or participants (such as 'carrot' or 'knife'). The authors propose to recognize which composite activity is depicted by a given video. To that end, they first train independent classifiers for each individual attribute. These classifiers are evaluated on each frame of the video to obtain attribute scores. These scores are then combined together along time (for better context coherence) and along attributes (to model co-occurrence of attributes) to compute features that are used to improve the attribute recognition by learning an 'activity attribute'

classifier. A global video feature representation is obtained through temporal max pooling of the outputs of this new classifier. This global feature is finally used to predict the composite activity. They show that script can be used to easily extract attributes and hence weight the video representation accordingly by filtering out absent attributes. Finally, they also demonstrate that this simple approach can generalize to unseen video demonstration of activity for which only a script is given. Indeed, since a list of attributes can be extracted from the script, their model is able to still make predictions for that activity. This flexibility is an important aspect of the composite activity approach.

2.5 Instructional videos

In the previous sections, we have covered various domains of computer vision, such as 'object detection', 'human pose estimation' or 'action recognition'. The recent impressive advances we have discussed are good reasons to tackle ambitious problems at the intersection of these domains. Learning from narrated instructional videos is one of such problems. In this section, we review works that have emerged over the last few years on this topic. Most of these works are contemporary or newer than the approaches presented in the following chapters.

2.5.1 Step discovery and localization

Tasks depicted by instructional videos are often decomposed into sub-actions that we refer to as steps. One of the main tasks in instructional video analysis is the task of step localization in the videos. Depending on the method, the list of steps is assumed to be known or is sometimes also discovered from the data as it is in the case with the method presented in Chapter 4. Below we review the most relevant work in that area.

Malmaud *et al.*, [2015] consider cooking narrated instructional videos accompanied with a recipe. Such recipe is composed of a list of instructions describing each step with natural language and a list of ingredients. Thanks to a smart automatic data collection process, the authors obtain 180,000 nar-

rated videos each accompanied by a recipe. Each recipe is then parsed using Natural Language Processing (NLP) standard tools (part of speech tagging, dependency parsing) in order to extract for each instruction, a list of actions (verb) with their noun entities (nouns) which are matched to an ingredient from the list whenever it is possible. For example the instruction 'Crack the eggs' would be match to the action 'crack' and to the ingredient 'egg'. Once this process is done the next step is to align the processed recipe with the transcription of the narration. Malmaud *et al.*, [2015] use transcription obtained through the YouTube Automatic Speech Recognition (ASR) system. This is challenging due to general lack of punctuations and the errors made when transcribing the spoken words. The alignment with the recipes relies on a Hidden Markov Model (HMM). The output of this alignment only gives a crude temporal localization of the steps in the video due to the imprecise temporal resolution of subtitles. The authors then propose to refine this localization with visual food detectors trained on a large corpus containing various food related labeled images as illustrated in Figure 2.14.

A contemporary work to ours [Sener *et al.*, 2015] solves a similar problem as the one tackled in Chapter 4. Given a set of narrated instructional videos describing the same task, they discover the steps composing the task and localize their appearance in the input videos. Differently from ours, their

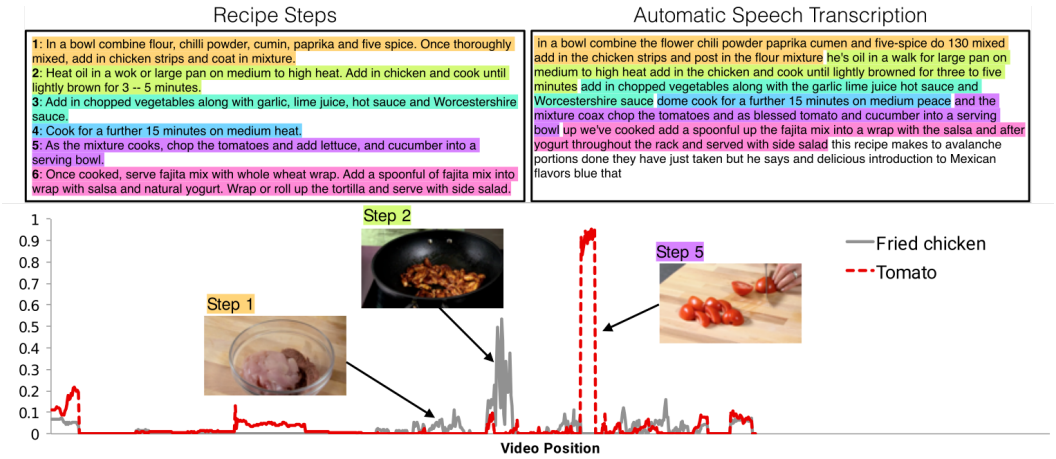


Fig. 2.14.: Illustration from Malmaud *et al.*, [2015]. **Top:** Given a cooking narrated instructional video accompanied with its recipe, the authors first align the recipe steps with the automatic speech transcription of the narration. **Bottom:** Thanks to the list of ingredients and associated visual object detectors, the authors can refine the alignment of the recipe with the video by visually grounding the steps of the recipe with frames of the video.

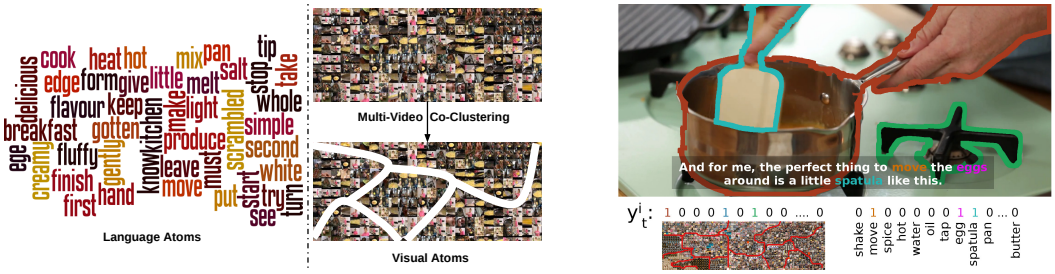


Fig. 2.15.: Illustration from [Sener *et al.*, 2015]. **Left:** The authors first learn language and visual atoms by performing clustering in order to model the multi-modal information contained in the textual and visual streams of narrated instructional videos. **Right:** This clustering is used to represent frames with subtitles by one-hot type vectors encoding the presence/absence of the visual and language atoms.

method relies on a non parametric Bayesian approach described next. First, the authors learn visual and language so called atoms in order to represent their data. In the visual domain, these atoms are constructed by clustering object proposals obtained from the frames of the videos depicting the task. In the language domain, the atoms are simply obtained by reasoning on word frequency. Once these atoms are learned, each frame of the video is represented by a bag-of-words feature translating the presence/absence of these visual and language atoms in the frame as shown in Figure 2.15. The authors then introduce a generative model based on Beta Process Hidden Markov Model where an activity step is represented as a Bernoulli distribution over the language and visual atoms. By fitting this model to their data using Markov Chain Monte Carlo methods, they are able to jointly discover activity steps and parse each video (*i.e.*, matching the frames of the videos to steps). They validate their approach on a collected dataset containing 17 tasks with 100 videos each.

Zhou *et al.*, [2018] argue that it is important for a system to work without video subtitles especially during the evaluation phase (*e.g.*, helping someone achieving a new task). In this work they introduce the task of 'procedure segmentation': training a system to segment an unconstrained video depicting a procedure (*e.g.*, an instructional video) into a coherent sequence of category-independent segments. They introduce a model named 'ProcNets' which only uses visual information and does not require the number of segments to be given beforehand. 'ProcNets' is an end-to-end trainable model which can be decomposed into three main stages. During the first phase, generic visual features computed independently for all frames of the video are combined

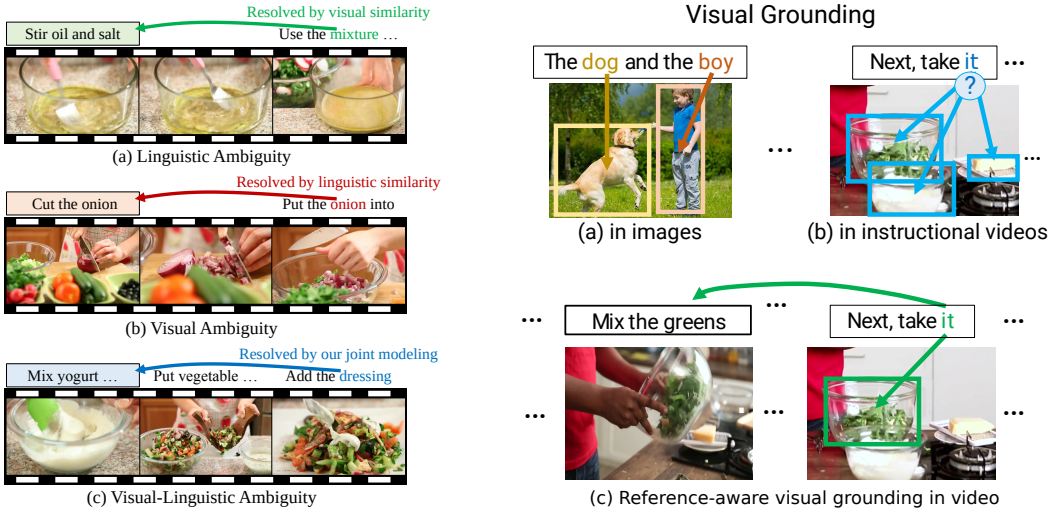
together through bi-directional LSTM to obtain frame-wise context-aware features. Second, these features are used to produce segment candidates. Finally, these candidates are fed to a sequential prediction module (LSTM) which captures the temporal structure among the candidates in order to output the final set of predictions. This model is trained in a supervised manner on a dataset collected by the authors.

Recently, Sener and Yao, [2018] also introduce a method that does not require the textual narration as input for the task of segmenting instructional videos into steps. Their method relies on a Generalized Mallows Model that was used before for a similar task in language [Frermann *et al.*, 2014]. One of the main advantages of this model is that it provides flexibility in the temporal structure governing the steps. It can indeed handle missing steps and swaps. The method works by alternating between learning the visual appearance of the steps in the videos and learning the temporal structure of the steps. They evaluate the effectiveness of their method on two challenging instructional video datasets.

In this thesis, we argue that the textual narration is a valuable input that should be used for multiple reasons in the context of instructional videos. First, the narration is easily available for the majority of online videos. In addition, thanks to the recent advances in ASR, the automatic transcription is much less error-prone than before. Second, text has the advantages of being more compact and more interpretable for humans when compared to visual input. Hence, being able to summarize a task as a sequence of textual instructions accompanied with visual illustrations is more informative than images alone. Third, as we will see in the following chapters, text helps improving visual prediction. Finally, we also believe that using text as a grounding support to train visual models does not prevent having system that can work without text during the evaluation phase.

2.5.2 Reasoning about objects

In addition to actions, objects play an important role in instructional videos. These objects are usually tools or the things the person acts on, such as a hammer or a nail. The prior work mentioned in the previous section often models objects implicitly to better predict actions (e.g. through visual



(a) From [Huang *et al.*, 2017]. **(b)** From [Huang *et al.*, 2018].

Fig. 2.16.: Approaches modeling objects in the context of narrated instructional videos. **(a)** In [Huang *et al.*, 2017], the authors tackle the problem of visual-linguistic reference resolution where the goal is to link an entity with the action that produced it. They show that jointly modeling vision and language helps improving performance. **(b)** In [Huang *et al.*, 2018], the authors introduce the task of reference-aware visual grounding in instructional videos where the goal is to link textual expressions to visual objects in the video. The authors show that it is key to also perform reference resolution (*e.g.*, finding that 'it' refers to the 'greens') for that problem when working with instructional videos.

features). In this section, we instead highlight works that focus explicitly on object modeling in the context of instructional videos. This is mostly related to the work presented in Chapter 5 which jointly models object states and actions.

Huang *et al.*, [2017] introduce the task of reference resolution in instructional videos, i.e. linking textual expressions referring to an object (e.g. 'dressing') to the action that produced it (e.g. 'mix yogurt') as illustrated in Figure 2.16a. To address that task, the authors design an unsupervised method that leverages both visual and linguistic cues in order to resolve the ambiguities inherent to the problem. They formulate the problem as a graph optimization where the nodes of the graph are the expressions (referring to objects and actions) and the edges represent the temporal links that they seek to infer during training. The experiments show that their model is superior to a linguistic model only baseline, hence demonstrating that the visual cues are crucial for solving that problem.

Huang *et al.*, [2018] build upon [Huang *et al.*, 2017] to tackle the problem of visual grounding in the context of narrated instructional videos. Visual grounding is the task of relating the objects mentioned in the narration to their appearances in the video stream (see Figure 2.16b). To address the two challenges that come with that problem, namely the heavy usage of referring expressions in instructional videos and the lack of annotations, the authors introduce a model that is 'reference-aware' and that only needs aligned transcription and video. The whole procedure is also framed as a graph optimization where the nodes are the inputs (bounding box proposals and text expressions) and the edges are the variables that represent the reference resolution and the grounding that they want to infer. A probabilistic model is introduced and optimized by alternating between the reference resolution and the grounding. The performance of the approach is demonstrated on two standard instructional video datasets for which the authors provide grounding annotations.

2.5.3 Datasets

As mentioned previously, all methods need datasets to both train and evaluate their model. In this section we quickly review the most popular datasets for instructional videos. A closer look at the dataset we introduce in this thesis is given in Chapter 4.

Malmaud *et al.*, [2015] release the '*What's cookin*' dataset[1] which contains 180,000 YouTube videos of cooking activities that have been obtained through automatic filtering. The data comes with automatic temporal predictions of objects and actions. The main advantage of this data is its scale, however the fact that it has been automatically obtained and that it does not contain human annotations prevents precise evaluation.

Sener *et al.*, [2015] introduce the '*RoboWatch*' dataset[2]. This dataset depicts 17 categories such as 'Hard Boil an Egg' or 'Make a Milkshake'. These categories have been obtained by taking the top 100 queries on the WikiHow website and retaining only the activities referring to physical tasks. For each category, the top 100 videos of YouTube are downloaded along with

[1] https://github.com/malmaud/whats_cookin
[2] http://robo.watch/

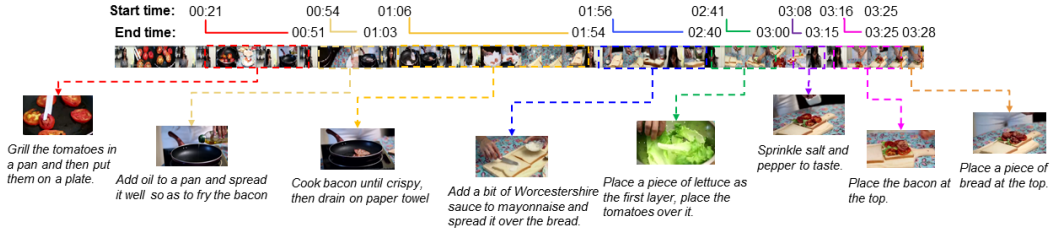


Fig. 2.17.: Illustration of a sample annotated video from the YouCook2 dataset [Zhou *et al.*, 2018]. Figure from [Zhou *et al.*, 2018].

their subtitles. 5 videos per category are annotated with precise temporal boundaries of action steps.

Zhou *et al.*, [2018] propose the 'YouCook2' dataset[3] which contains 2000 narrated instructional videos depicting 89 cooking recipes. Action steps for each video are annotated with precise temporal boundaries and described by English sentences as illustrated in Figure 2.17. This is currently one of the largest fully annotated dataset for instructional videos.

Recently, Fouhey *et al.*, [2017] introduce the 'VLOG' dataset[4] which depicts a close cousin to instructional videos: Lifestyle VLOGs. These videos are uploaded by people to describe their day-to-day lives. As they contain many interactions with standard objects (microwave, fridge...), these videos are of interest for the vision community. The dataset is large with about 114,000 videos which represents 15 days of footage.

Finally, Damen *et al.*, [2018] release the 'EPIC kitchen' dataset[5]. It is currently the largest dataset in egocentric vision, containing 55 hours of recording depicting daily activities performed in 32 different kitchens. Narration is available as the performers were asked to comment what they were doing in order to ease the annotation process. Even if not directly comparable to the instructional videos that can be found online which are often third person view, this dataset constitutes a very nice playground for designing methods that could transfer to our domain of interest.

[3] http://youcook2.eecs.umich.edu/
[4] https://people.eecs.berkeley.edu/~dfouhey/2017/VLOG/
[5] https://epic-kitchens.github.io/2018

3 Background in machine learning

In this chapter, we survey the literature that is closely related to the technical content of this thesis. We start by giving a quick overview of the different forms of learning (Section 3.1 and Section 3.2). In Section 3.3, we describe the discriminative clustering framework that is used in this thesis, and explain how it can be used to deal with weak supervision. Finally, Section 3.4 reviews the Frank-Wolfe algorithm and its variants, an optimization technique particularly suited for the kind of problems encountered in this thesis.

We place ourselves in the setting where we want to predict a label $z \in \mathcal{Z}$ given an input data point $x \in \mathcal{X}$. To that end, we use the discriminative learning paradigm which consists in learning a function $f \in \mathcal{F}$ ($f : \mathcal{X} \to \mathcal{Z}$), that maps the input data point $x \in \mathcal{X}$ to a label $f(x) = z \in \mathcal{Z}$. The space $\mathcal{F}$ is the hypothesis space, describing the type of functions that are thought to be good to predict z from x. A typical example of such setting that is used throughout this thesis is K-class classification from d-dimensional input features, where $\mathcal{X} = \mathbb{R}^d$, $Z = \{0, 1\}^K$ and f is a *classifier*.

For a more general overview on machine learning we invite the reader to refer to standard works such as [Bishop, 2006; Hastie *et al.*, 2009; Murphy, 2012].

3.1 Supervised learning

This section is devoted to provide the basic working principles of supervised learning, to highlight some of its recent successes and also to shed light on some of its limitations.

3.1.1 What is supervised learning?

In supervised learning, we dispose of a training set containing n pairs of input data points and their associated labels: $\{(x_i, z_i)\}_{i=1}^n$. Given that, a standard approach is to learn a mapping f so that the labels z_i are close to the mapped inputs $f(x_i)$. Formally, a loss function $\ell : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}^+$ is used to specify what is the cost $\ell(f(x_i), z_i)$ of predicting $f(x_i)$ when the label is actually z_i. To learn f, a standard technique is to employ the empirical risk minimization paradigm:

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \ell(f(x_i), z_i), \tag{3.1}$$

which consists in finding f so that it minimizes the average loss on the training set.

The key desired property when learning f is *generalization*, i.e. obtaining a function f that will perform well not only on the training set but also on unseen data points. To that end, a common practice is to follow the Occam's razor principle by promoting simple functions to explain the data. This process is called *regularization* and can be imposed either explicitly through the hypothesis space $\mathcal{F}$ or sometimes implicitly by adding a regularization function $\Omega : \mathcal{F} \to \mathbb{R}^+$ to the objective defined in (3.1):

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \ell(f(x_i), z_i) + \Omega(f). \tag{3.2}$$

Supervised learning is at the origin of multiple successes over the last few years. To name a few, supervised learning allows your computer to find pictures of you at the beach, your smartphone to understand oral instructions, or even your browser to translate a web page so you can read it. Despite these successes, supervised learning comes at a cost and with limitations which we discuss next.

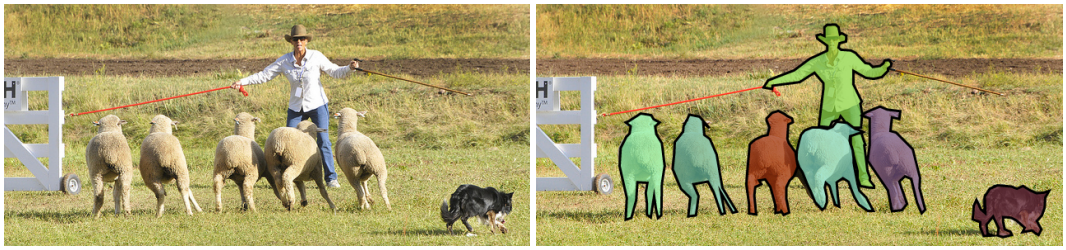
3.1.2 What are the limitations of supervised learning?

Limitations of supervised learning mainly come from the fact that it relies on manual annotations. This is worse with deep learning models that require lots of such data to work well.

First of all, these annotations are costly to get as they require an important human effort. In some cases, experts are needed for annotation (*e.g.*, for medical applications) which limits the scale of potential training datasets.

Most importantly, the annotation process may be ambiguous. Imagine the case of annotating action temporal boundaries. It is difficult to reach a consensus when annotating, because often it is difficult to agree on the start and end times of an action. For example, when does the action 'drinking' start? Is it when the glass touches the mouth, or when the person start holding it? This kind of ambiguities introduces noise and biases in the training data. This also poses the problem of granularity definition. Imagine annotating instance segmentation for an image containing a man wearing a hat as illustrated in Figure 3.1. How many classes would you define? Only one for the person? In that case, would you annotate the hat has being part of the person? Or would you decide to annotate two classes (*e.g.*, one for the man and one for the hat)?

For these reasons, it is important to find alternatives to supervised learning. This would allow finding ways to learn something meaningful about the world without always relying on defining and collecting precise ground



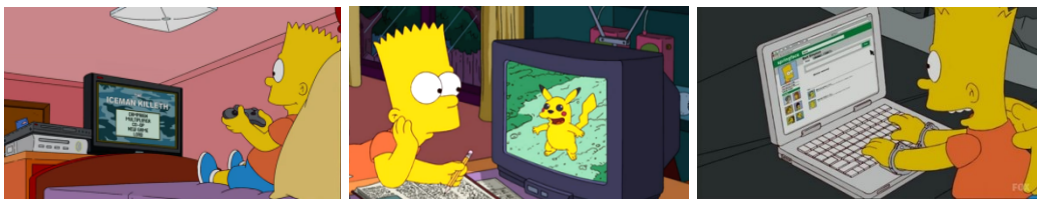
(a) Image from the COCO dataset **(b)** COCO segmentation ground truth

Fig. 3.1.: How to annotate an image for semantic segmentation? [Lin *et al.*, 2014]

truth. In the following section, we explore a few alternatives to supervised learning.

3.2 Alternatives to supervised learning

In this section, we present three standard alternatives to supervised learning with some examples: reinforcement learning, unsupervised learning and weakly supervised learning. The latter is going to be the framework used throughout the thesis. We provide an illustration of these different forms of learning in Figure 3.2.



(a) Reinforcement learning **(b)** Unsupervised learning **(c)** Weakly-supervised learning **(this thesis)**

Fig. 3.2.: Illustration of different alternatives to supervised learning. **Reinforcement learning**: learn by playing and interacting with the environment. **Unsupervised learning**: only learn by looking at the data. **Weakly Supervised Learning**: learn from readily available sources of information, such as metadata contained in the web. In this thesis, we mostly focus on this type of learning. See text for more details.

Reinforcement Learning. In reinforcement learning [Sutton and Barto, 1998], the goal is for an *agent* to learn how to act in an *environment* in order to maximize its *rewards*. Learning is typically done by trial and errors. In principle, the agent only learns by interacting with its environment and no human annotation is required. This setting is notably suited to learn how to play games such as the game of Go [Silver *et al.*, 2017].

Unsupervised Learning. In unsupervised learning, the task is to extract meaningful information about the data without relying on any annotations. Formally, only the input data $x \in \mathcal{X}$ are provided and no labels $z \in \mathcal{Z}$ are available.

Examples include generative models, where the goal is to try to approximate the probability distribution that governs the generation of the data points

$x \in \mathcal{X}$. For example, one can try to learn the generative distribution of natural images in order to be able to sample new images. A recent popular method that lead to very realistic samples is the Generative Adversarial Networks [Goodfellow *et al.*, 2014] framework. It consists of two networks, a generator whose role is to output natural looking images from a noise vector, and a discriminator whose role is to determine whether an image comes from a real world dataset or whether it was artificially generated. These two networks are competing with each other. By doing so the discriminator sends implicit information to the generator so that it starts producing better and better looking images. By learning how to generate the input data, one can hope learning meaningful representations which can help solving other tasks such as classification.

Another example which is more closely related to this thesis is *clustering*. Clustering aims at grouping the input data points $\{x_i\}_{i=1}^N$ into K separate clusters so that a global criterion is satisfied. For example, imagine you have a collection of images depicting several of your friends. A clustering algorithm could consist in separating those images according to the identity of your friends. Such an algorithm would not require any knowledge about who are your friends to proceed. Instead, it would simply regroup images containing similar faces. K-means is one the most famous clustering method when $\mathcal{X} = \mathbb{R}^d$. It consists in finding K centroids $\{c_k\}_{k=1}^K \in \mathcal{X}^K$ so that the following quantity is minimized:

$$\sum_{i=1}^{N} \min_{k \in [1..K]} \|x_i - c_k\|_2^2. \tag{3.3}$$

Each data point is therefore assigned to the cluster which has the closest centroid. Solving this problem is known to be NP-hard. A standard algorithm for finding an approximate solution is the Lloyd's algorithm [Lloyd, 1982] which consists in an iterative scheme alternating between updating the centroids $\{c_k\}_{k=1}^K \in \mathcal{X}^K$ and updating the assignments (through the minimization $\min_{k \in [1..K]} \|x_i - c_k\|_2^2$) of the data point to the centroids. In Section 3.3, we introduce another clustering algorithm based on a discriminative principle, i.e. which cluster the data so that the separation of the clusters is easily recoverable by a discriminative classifier. The applications presented in the following chapters mainly used this algorithm.

Weakly-supervised Learning. In weakly-supervised learning, we assume that for each sample x_i we have incomplete and potentially noisy information about its label.

Coming back to the example of the collection of photos depicting yours friends, imagine now that for each photo you also have tags indicating which of them are present in the picture. Given that partial information, we can try to estimate what is the identity of each face. The tags can be considered as the weak supervision. More formally, considering each face as a sample x_i, then for each of them the label z_i can take multiple values in a set $\mathcal{Z}_i \subset \mathcal{Z}$ given by the tags. One way to learn a mapping f from inputs to labels in that case would be to modify the optimization problem (3.1) as follows:

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \min_{z_i \in \mathcal{Z}_i} \ell(f(x_i), z_i), \tag{3.4}$$

where the labels z_i are now latent variables estimated during the optimization. This is the core idea used in Chapter 4 and 5.

One of the practical advantage of such learning scenario, is that this kind of partial information can often be extracted from readily available sources without requiring tedious manual annotations. For example, [Bojanowski *et al.*, 2013] use movie scripts to extract candidate clips containing actions and actors and then propose a weakly supervised learning method to refine the predictions.

3.3 Discriminative clustering

In this section, we describe the discriminative clustering framework and more specifically the DIFFRAC [Bach and Harchaoui, 2007] algorithm. This technique enables incorporating weak supervision, and is largely leveraged in this thesis.

3.3.1 Framework

Discriminative clustering uses ideas from discriminative supervised learning (presented in Section 3.1.1) in order to perform clustering [Xu *et al.*, 2004; Bach and Harchaoui, 2007]. Differently from the K-means paradigm which is based on a geometric criterion, here the goal is to separate the data so that the clusters can be easily recovered by a discriminative classifier. To do so, one can reuse the equations from the supervised case (3.2), but considering the labels as latent variables learned during optimization:

$$\min_{z_1,\cdots,z_n} \min_{f\in\mathcal{F}} \sum_{i=1}^{n} \ell(f(x_i), z_i) + \Omega(f). \tag{3.5}$$

This is usually not enough as trivial solutions emerge (*e.g.*, assigning every data points to the same cluster). To avoid this, a workaround is to add constraints on the clustering, such as a minimum size for each cluster. In some cases, additional constraints can be applied to reflect a prior on a specific problem as will be done in our applications. Next, we present a discriminative clustering method named DIFFRAC.

3.3.2 DIFFRAC

Method. Bach and Harchaoui, [2007] introduce the DIFFRAC algorithm, a special case of discriminative clustering where $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Z} = \{0,1\}^K$, $\mathcal{F}$ is the set of linear functions, ℓ is the square loss and Ω corresponds to the Tikhonov regularization. More formally, we have:

$$\min_{z_1,\cdots,z_n} \min_{W\in\mathbb{R}^{d\times K}} \frac{1}{2n} \sum_{i=1}^{n} \|W^\top x_i - z_i\|_2^2 + \frac{\lambda}{2}\|W\|_F^2, \tag{3.6}$$

where $W \in \mathbb{R}^{d\times K}$ represents the linear classifier, λ is a parameter to weight the effect of regularization, $x^\top$ stands for the transpose of x and $\|.\|_F$ is the standard Frobenius matrix norm. In other words, we apply a ridge regression [Hastie *et al.*, 2009, Chapter 3.4.1.] on the cluster identity and use its objective value as clustering cost.

Equation (3.7), can be rewritten in matrix notations by introducing the design matrix $X = (x_1^\top, \cdots, x_n^\top) \in \mathbb{R}^{n \times d}$ and regrouping the variables z_i in the same way: $Z = (z_1^\top, \cdots, z_n^\top) \in \{0,1\}^{n \times K}$. We can rewrite (3.7) as follows:

$$\min_{Z \in \{0,1\}^{n \times K}} \min_{W \in \mathbb{R}^{d \times K}} \frac{1}{2n} \|XW - Z\|_F^2 + \frac{\lambda}{2} \|W\|_F^2. \tag{3.7}$$

Note that we present a slightly different modified version of [Bach and Harchaoui, 2007] where we don't explicitly model the bias term in the linear model W. In practice, a bias term can be added by adding a constant value to the features. To avoid the bias term to be regularized, we simply set this constant value to a big number relatively to the other feature dimensions. Doing so allows to simplify equations without loss of generality.

One of the advantage of using the square loss is that the objective is jointly convex in Z and W and that the minimization in W can be done in closed form. To do so, one can simply set the gradient of the expression (3.7) with respect to W to 0 to obtain:

$$W^*(Z) = (X^\top X + n\lambda I_d)^{-1} X^\top Z, \tag{3.8}$$

where I_d is the d-dimensional identity matrix. By injecting the expression of $W^*(Z)$ and rearranging the term in (3.7), we obtain:

$$\min_{Z \in \{0,1\}^{n \times K}} \mathrm{Tr}(ZZ^\top B), \tag{3.9}$$

where Tr is the trace matrix operator and $B = I_n - X(X^\top X + n\lambda I_d)^{-1} X^\top$ is a positive definite matrix (hence the function $Z \mapsto \mathrm{Tr}(ZZ^\top B)$ is strongly convex) that only depends on the data X and on the regularization parameter λ.

In the end, the clustering cost is a quadratic function in Z, indicating how the clustering decision for a data point x_i interacts with the clustering decision for another data point x_j.

As explained before and without any additional constraints on Z, Problem (3.11) has a trivial solution consisting in setting all entries of Z to 0. To avoid this, Bach and Harchaoui, [2007] first impose all points to be assigned

to exactly one cluster, which can be imposed by a linear constraint on the rows of the matrix Z:

$$\min_{Z \in \{0,1\}^{n \times K}} \mathrm{Tr}(ZZ^\top B), \text{ such that } Z\mathbf{1}_K = \mathbf{1}_n, \tag{3.10}$$

where $\mathbf{1}_K$ is the K-th dimensional vector of all ones. Even with this constraint, another degenerate solution correspond to the case where all datapoints are assigned to the same cluster [Bach and Harchaoui, 2007]. To avoid this issue, Bach and Harchaoui, [2007] add other constraints such as a minimal size for all clusters. If one has access to enough weak supervision and as we show next, one can add other constraints to steer the solution towards a preferred direction and to avoid these degenerate cases.

Problem (3.11) is NP-hard in general due to the integer constraints on Z. A standard approach that is used throughout this thesis is to use convex relaxations to obtain approximate solutions. In our specific case, we employ the tightest convex relaxation which stays in the same space by taking the convex hull of the set of constraints on Z, for which the Frank-Wolfe algorithm (see Section 3.4) is quite suitable.

Incorporating weak supervision. Weak supervision can easily be incorporated as constraints on the variable Z in the DIFFRAC formulation:

$$\min_{Z \in \{0,1\}^{n \times K}} \mathrm{Tr}(ZZ^\top B), \text{ such that } Z \in \mathcal{Z}_{ws}. \tag{3.11}$$

3.3.3 Applications with weak supervision

In computer vision, the DIFFRAC framework has been used for several weakly supervised applications. Below, we review some of them.

In [Joulin *et al.*, 2010], the authors use DIFFRAC for cosegmenting multiple images containing the same object into foreground/background classes. In that case, the weak supervision simply consists in the fact that multiple images are known to depict the same object. Joulin *et al.*, [2012] later

extend [Joulin *et al.*, 2010] to handle the multi class scenario where one image may contain different object classes.

Joulin *et al.*, [2014c] also use DIFFRAC in order to co-localize similar objects in videos. In that case, the weak supervision consists in knowing that multiple videos contain the same object without any precise localization annotations.

In [Bojanowski *et al.*, 2013], the authors use this framework to jointly localize actors and detect actions in feature length movies only from the weak information contained in the scripts and subtitles.

In Chapter 4 and 5, we present two additional applications that use DIFFRAC in the context of instructional videos.

3.4 Frank-Wolfe algorithm

In this section we present the Frank-Wolfe algorithm (a.k.a. conditional gradient) and motivate its use for our specific applications.

3.4.1 The algorithm

The Frank-Wolfe algorithm [Frank and Wolfe, 1956] is an optimization procedure used for solving:

$$\min_{\boldsymbol{\alpha} \in \mathcal{M}} f(\boldsymbol{\alpha}), \tag{3.12}$$

with $f : \mathcal{M} \longrightarrow \mathbb{R}$ a smooth and convex function and $\mathcal{M}$ a convex and compact (closed and bounded in finite dimension) set.

Algorithm 1 describes the iterative procedure and Figure 3.3 provides an illustration of it. At each time step of the algorithm, a linear approximation of the function f is obtained by computing its gradient at the current iterate $\boldsymbol{\alpha}$. Next, a linear oracle is called in order to obtain the minimizer $\boldsymbol{s}$ of the linearization of f. Then, $\boldsymbol{s}$ is combined with $\boldsymbol{\alpha}$ using the step size γ for computing the next iterate.

Algorithm 1 Frank-Wolfe algorithm

```
# Initialization
Let α^(0) ∈ M
for t in 0 ... T do
    # Linear oracle
    s = argmin_{s∈M} ⟨s, ∇f(α^(t))⟩
    # Compute the step-size
    Set γ with γ = 2/(t+2) or by line-search
    # Update the iterate
    α^(t+1) = (1 − γ)α^(t) + γs
end for
```

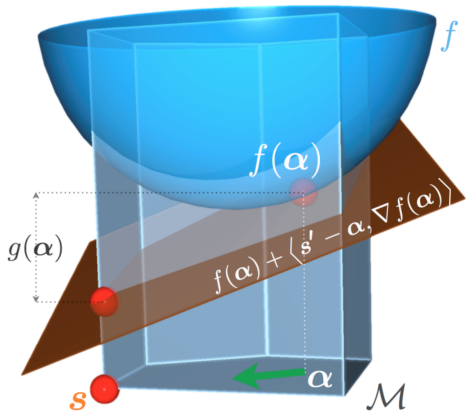


Fig. 3.3.: [Jaggi, 2013b]: Illustration of the Frank-Wolfe algorithm.

3.4.2 Properties

The Frank-Wolfe algorithm has several interesting properties that have allowed it to regain a lot of interest especially for machine learning in the recent years [Jaggi, 2013b]. These properties include:

- **Only a linear minimization oracle (LMO) is needed.** The only requirement is to be able to minimize linear functions over the set $\mathcal{M}$. Minimizing linear functions is usually simpler than minimizing quadratic ones. Hence the Frank-Wolfe algorithm can be more appealing than projected gradient descent [Nesterov, 1998, Chapter 2] in some cases as it requires to minimize quadratic functions (projection) at each iteration.

- **Convergence rate.** The algorithm enjoys a $\mathcal{O}(\frac{1}{T})$ convergence rate, *i.e.*, after T iterations the sub-optimality $f(\boldsymbol{\alpha}) - f(\boldsymbol{\alpha}^*)$ (with $\boldsymbol{\alpha}^*$ the optimum) is of order $\frac{1}{T}$. This is slower than the $\mathcal{O}(\frac{1}{T^2})$ convergence rate that can be achieved with Nesterov acceleration of the projected gradient method [Nesterov, 1998, Chapter 2]. However, when the LMO is much cheaper than the quadratic oracle then Frank-Wolfe might be much faster.

- **Iterates are sparse.** A standard situation is to use the Frank-Wolfe algorithm to minimize functions over polytopes. Minimizers of linear functions over polytopes are corners. Hence, after the T-th iteration of the algorithm, the iterate can be decomposed as a sum of at most T such corners. This allows for sparse representation of the iterate which is very appealing in high dimensional cases where explicitly storing all dimensions is not practical.

- **Dual gap certificate for free.** During the course of the algorithm, a dual gap can be obtained without performing additional computations. Indeed, because f is convex it lies above its linear approximation:

$$\forall \boldsymbol{\alpha} \in \mathcal{M}, \forall \boldsymbol{s'} \in \mathcal{M}, f(\boldsymbol{s'}) \geq f(\boldsymbol{\alpha}) + \langle \boldsymbol{s'} - \boldsymbol{\alpha}, \nabla f(\boldsymbol{\alpha}) \rangle. \tag{3.13}$$

  This implies that:

$$\begin{aligned}
& \min_{s' \in \mathcal{M}} f(\boldsymbol{s'}) \geq && \min_{s' \in \mathcal{M}} f(\boldsymbol{\alpha}) + \langle \boldsymbol{s'} - \boldsymbol{\alpha}, \nabla f(\boldsymbol{\alpha}) \rangle \\
\Rightarrow \quad & f(\boldsymbol{\alpha}^*) \geq && f(\boldsymbol{\alpha}) + \langle \boldsymbol{s} - \boldsymbol{\alpha}, \nabla f(\boldsymbol{\alpha}) \rangle \\
\Rightarrow \quad & f(\boldsymbol{\alpha}^*) - f(\boldsymbol{\alpha}) \geq && \langle \boldsymbol{s} - \boldsymbol{\alpha}, \nabla f(\boldsymbol{\alpha}) \rangle \\
\Rightarrow \quad & f(\boldsymbol{\alpha}) - f(\boldsymbol{\alpha}^*) \leq && \underbrace{\langle \boldsymbol{\alpha} - \boldsymbol{s}, \nabla f(\boldsymbol{\alpha}) \rangle}_{g(\boldsymbol{\alpha})}.
\end{aligned}$$

  The quantity $g(\boldsymbol{\alpha}) = \langle \boldsymbol{\alpha} - \boldsymbol{s}, \nabla f(\boldsymbol{\alpha}) \rangle$ (illustrated in Figure 3.3) is an upper-bound on the suboptimality $f(\boldsymbol{\alpha}) - f(\boldsymbol{\alpha}^*)$ of the problem. This quantity is in most cases computed during the linear oracle. In addition, this upper-bound can be interpreted as a Fenchel dual gap [Jaggi, 2013b] which also enjoys a convergence rate towards 0 in $\mathcal{O}(\frac{1}{T})$. In practice, this is useful in order to monitor the convergence of the algorithm.

- **Affine invariance.** The algorithm is invariant to surjective linear or affine reparametrization of the input domain $\mathcal{M}$ [Jaggi, 2013b].

- **Approximate linear solver.** Finally, the algorithm also converges in the case where the linear oracle is not exact [Jaggi, 2013b], a situation that happens in practice.

3.4.3 The variants

Notable variants of the Frank-Wolfe algorithm are the ones that allow away steps [Wolfe, 1970] (away-steps FW, pairwise FW, fully corrective FW). The core idea of away steps is to add the possibility to remove corners in the decomposition of the iterate. This can be done without any specific oracle, by simply evaluating all corners in the decomposition and deciding whether to do an away step or a standard Frank-Wolfe step based on a simple criterion. These variants all enjoy a linear convergence rate [Lacoste-Julien and Jaggi, 2015a] under a weaker condition than strong convexity on the objective function f and when $\mathcal{M}$ is polyhedral.

The Block-Coordinate Frank Wolfe (BCFW) algorithm [Lacoste-Julien *et al.*, 2013] can be used whenever the constraint set $\mathcal{M}$ can be decomposed as a Cartesian product of smaller 'blocks': $\mathcal{M} = \mathcal{M}^{(1)} \times \ldots \times \mathcal{M}^{(n)}$. BCFW consists in randomly sampling one block at each time step and perform a Frank-Wolfe step on that block. When compared to the batch Frank-Wolfe algorithm, the BCFW has the potential to be much faster by leveraging the correlations that may exist between blocks. This algorithm is detailed in Chapter 6.

Finally, Lacoste-Julien, [2016] has recently proved that the Frank-Wolfe algorithm converge to stationary points when the function f is not convex. This observation is promising in the sense that it broadens the scope of applications of the Frank-Wolfe algorithm.

3.4.4 Why is it relevant for discriminative clustering?

In Section 3.3.2, we have seen that weak supervision can be incorporated in the DIFFRAC framework by defining appropriate set of constraints $\mathcal{Z}_{ws}$:

$$\min_{Z \in \{0,1\}^{n \times K}} \mathrm{Tr}(ZZ^\top B), \text{ such that } Z \in \mathcal{Z}_{ws}. \tag{3.14}$$

Convex relaxation. $\mathcal{Z}_{ws}$ is typically defined as a subset of discrete points in $\{0,1\}^{n \times K}$ encoding all the potential labeling candidates given the weak

supervision. Due to this type of constraint, the problem is NP-hard to solve in general. We can propose a simple relaxation by replacing the discrete set $\mathcal{Z}_{ws}$ by its convex hull $\bar{\mathcal{Z}}_{ws}$:

$$\min_{Z \in [0,1]^{n \times K}} \mathrm{Tr}(ZZ^\top B), \text{ such that } Z \in \bar{\mathcal{Z}}_{ws}. \tag{3.15}$$

Frank-Wolfe is well suited for this kind opf problem because:

- $f : Z \mapsto \mathrm{Tr}(ZZ^\top B)$ is smooth and strongly convex (recall that B is strictly definite positive),
- $\bar{\mathcal{Z}}_{ws} \subset [0,1]^{n \times K}$ is convex, closed and bounded.

In addition, as explained in Chapter 4, some constraint sets used to encode prior information such as 'ordering constraints' are typical sets where it is easy to minimize linear functions (*e.g.*, through dynamic programming) but where it is hard/impossible to perform projection efficiently.

Moreover, taking the convex hull of a set of points is very easy with the Frank-Wolfe algorithm. Indeed, because the solution of the linear oracle is a corner (in that case one of the point of the discrete set), only an implicit representation of the convex hull is required as it is sufficient to only know how to solve a linear program over the set of discrete points, a situation occurring in Chapter 4 and Chapter 5.

Rounding. After solving the convex relaxation problem, one needs to come back to the discrete solution domain. This procedure is called rounding. Various solutions are proposed depending on the application (see Chapter 4 and Chapter 5).

4 Learning from narrated instructional videos

In this chapter, we address the problem of automatically learning the main steps of a task from a set of narrated instructional videos. We develop a new unsupervised learning approach that takes advantage of the complementary nature of the input video and the associated narration. The method sequentially clusters textual and visual representations of a task, where the two clustering problems are linked by joint constraints to obtain a single coherent sequence of steps in both modalities. To evaluate our method, we collect and annotate a new challenging dataset of real-world instructional videos from the Internet. The dataset contains videos for five different tasks with complex interactions between people and objects, captured in a variety of indoor and outdoor settings. We experimentally demonstrate that the proposed method can automatically discover, learn and localize the main steps of a task in input videos.

4.1 Introduction

Millions of people watch narrated instructional videos[1] to learn new tasks such as assembling IKEA furniture or changing a flat car tire. Many of such tasks have large amounts of videos available on-line. For example, querying for "how to change a tire" results in more than 300,000 hits on YouTube. Most of these videos, however, are made with the intention to teach other people to perform the task and do not provide direct supervisory signal for automatic learning algorithms. Developing unsupervised methods that could learn tasks from myriads of instructional videos on the Internet is therefore a key challenge. Such automatic cognitive ability would enable constructing virtual assistants and smart robots that learn new skills from the Internet to, for example, help people achieve new tasks in unfamiliar situations.

In this work, we consider instructional videos and develop a method that learns a sequence of steps, as well as their textual and visual representations, required to achieve a certain task. For example, given a set of narrated instructional videos demonstrating how to change a car tire, our method automatically discovers consecutive steps for this task such as *loosen the nuts of the wheel*, *jack up the car*, *remove the spare tire* and so on as illustrated in Figure 4.1. In addition, the method learns the visual and linguistic variability of these steps from natural videos.

Discovering key steps from instructional videos is a highly challenging task. First, linguistic expressions for the same step can have high variability across videos, for example: "...Loosen up the wheel nut just a little before you start jacking the car..." and "...Start to loosen the lug nuts just enough to make them easy to turn by hand...". Second, the visual appearance of each step may vary greatly because of differences in viewpoints, lighting, the poses, clothing and motion of people, types of manipulated objects and other factors. Finally, the overall structure of instructional videos may vary due to possible changes in the type and the order of steps.

To address these challenges, in this chapter we develop an unsupervised learning approach that takes advantage of the complementarity of the visual signal

[1] Some instructional videos on YouTube have tens of millions of views, e.g. www.youtube.com/watch?v=J4-GRH2nDvw.

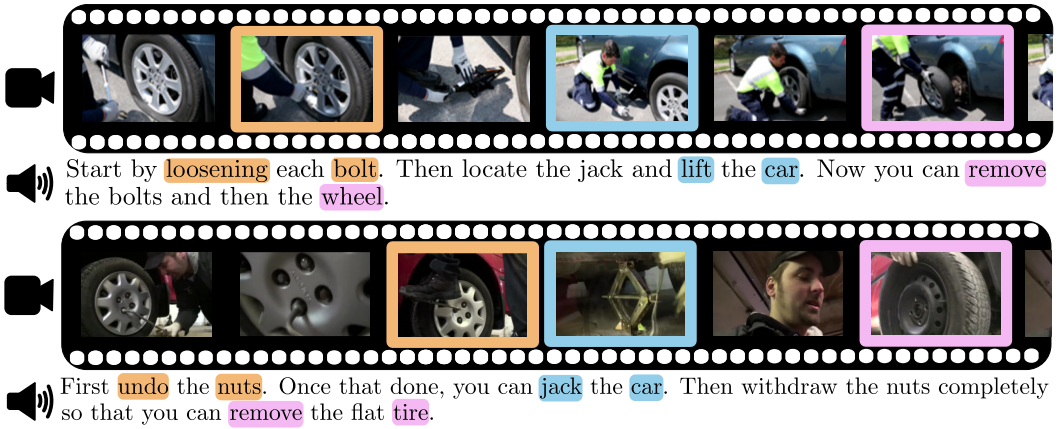


Fig. 4.1.: Given a set of narrated instructional videos demonstrating a particular task, we wish to automatically discover main steps to achieve the task and to associate each step with its corresponding narration and a temporal interval in each video. Here two videos of changing a car tire are illustrated by corresponding frames and excerpts of narrations. Steps of the same type are highlighted by the same color. Note the large variations in narrations and appearance of corresponding steps across videos.

in the video and the corresponding natural language narration to resolve their ambiguities. We assume that videos of the same task share the same sequence of ordered steps (also called script in the NLP literature [Regneri *et al.*, 2010]), however, the type and temporal locations of individual steps are unknown and should be discovered from the data. This is in contrast to other existing methods for modeling instructional videos [Malmaud *et al.*, 2015] that assume a script (recipe) is known and fixed in advance. We address the problem by first performing temporal clustering of text followed by clustering in video, where the two clustering tasks are linked by joint constraints. The complementary nature of the two clustering problems helps to resolve ambiguities in the two individual modalities. For example, two video segments with very different appearance but depicting the same step can be grouped together if they share similar narrations. Conversely, two video segments described with very different expressions, for example, "jack up the car" and "raise the vehicle" can be identified as belonging to the same instruction step because they have similar visual appearance. The output of our method is the script listing the discovered steps of the task as well as the temporal location of each step in the input videos. We validate our method on a new dataset of instructional videos composed of five different tasks[2] with a total of 150 videos and about 800,000 frames.

[2]Changing car tire, Perform cardiopulmonary resuscitation (CPR), Jump a car, Repot a plant, Make coffee

4.2 Related work

This work relates to unsupervised and weakly-supervised learning methods in computer vision and natural language processing. Particularly related to ours is the work on learning script-like knowledge from natural language descriptions [Chambers and Jurafsky, 2008; Frermann *et al.*, 2014; Regneri *et al.*, 2010]. These methods aim to discover typical events (steps) and their order for particular scenarios (tasks)[3] such as "cooking scrambled egg", "taking a bus" or "making coffee". While [Chambers and Jurafsky, 2008] uses large-scale news copora, [Regneri *et al.*, 2010] argues that many events are implicit and are not described in such general-purpose text data. Instead, [Frermann *et al.*, 2014; Regneri *et al.*, 2010] use event sequence descriptions collected for particular scenarios. Differently to this work, we learn sequences of events from narrated instructional videos on the Internet. Such data contains detailed event descriptions but is not structured and contains more noise compared to the input of [Frermann *et al.*, 2014; Regneri *et al.*, 2010].

Interpretation of narrated instructional videos has been recently addressed in [Malmaud *et al.*, 2015]. While this work analyses cooking videos at a great scale, it relies on readily-available recipes which may not be available for more general scenarios. Differently from [Malmaud *et al.*, 2015], we here aim to learn the steps of instructional videos using a discriminative clustering approach. A similar task to ours is addressed in [Naim *et al.*, 2015] using latent variable structured perceptron algorithm to align nouns in instructional sentences with objects touched by hands in instructional videos. However, similarly to [Malmaud *et al.*, 2015], [Naim *et al.*, 2015] uses laboratory experimental protocols as textual input, whereas here we consider a weaker signal in the form of the real transcribed narration of the video.

In computer vision, unsupervised action recognition has been explored in simple videos [Niebles *et al.*, 2008]. More recently, weakly supervised learning of actions in video using video scripts or event order has been addressed in [Bojanowski *et al.*, 2013; Bojanowski *et al.*, 2014; Bojanowski *et al.*, 2015; Duchenne *et al.*, 2009; Laptev *et al.*, 2008a]. Particularly related to ours

[3] We here assign the same meaning to terms "event" and "step" as well as to terms "script" and "task".

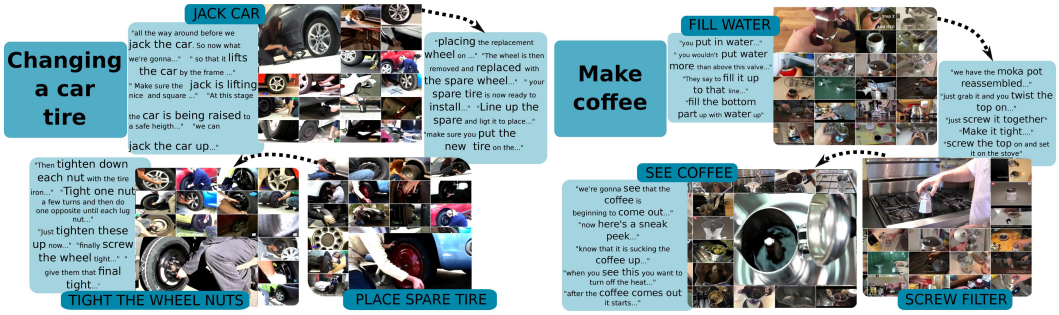


Fig. 4.2.: Illustration of our newly collected dataset of instructions videos. Examples of transcribed narrations together with still frames from the corresponding videos are shown for two (out of 5) tasks: *Changing car tire* and *Making coffee*. The dataset contains challenging real-world videos performed by many different people, captured in uncontrolled settings in a variety of outdoor and indoor environments. Note the large variability of verbal expressions and the terminology in the transcribed narrations as well as the large variability of visual appearance due to viewpoint, used objects, and actions performed in different manner. See our project webpage [Alayrac, 2016] for more examples.

is the work [Bojanowski *et al.*, 2014] which explores the known order of events to localize and learn actions in training data. While [Bojanowski *et al.*, 2014] uses manually annotated sequences of events, we here discover the sequences of main events by clustering transcribed narrations of the videos. Related is also the work of [Bojanowski *et al.*, 2015] that aligns natural text descriptions to video but in contrast to our approach does not discover automatically the common sequence of main steps. Methods in [Niebles *et al.*, 2010; Raptis and Sigal, 2013] learn in an unsupervised manner the temporal structure of actions from video but do not discover textual expressions for actions as we do in this work. The recent concurrent work [Sener *et al.*, 2015] is addressing, independently of our work, a similar problem but with a different approach based on a probabilistic generative model and considering a different set of tasks mainly focussed on cooking activities.

Our work is also related to video summarization and in particular to the recent work on category-specific video summarization [Potapov *et al.*, 2014; Sun *et al.*, 2014]. While summarization is a subjective task, we here aim to extract the key steps required to achieve a concrete task that consistently appear in the same sequence in the input set of videos. In addition, unlike video summarization [Potapov *et al.*, 2014; Sun *et al.*, 2014] we jointly exploit visual and linguistic modalities in our approach.

4.3 New dataset of instructional videos

We have collected a dataset of narrated instructional videos for five tasks: *Making coffee*, *Changing car tire*, *Performing cardiopulmonary resuscitation (CPR)*, *Jumping a car* and *Repotting a plant*. The videos were obtained by searching YouTube with relevant keywords. The five tasks were chosen so that they have a large number of available videos with English transcripts while trying to cover a wide range of activities that include complex interactions of people with objects and other people. For each task, we took the top 30 videos with English ASR returned by YouTube. We also quickly verified that each video contains a person actually performing the task (as opposed to just talking about it). The result is a total of 150 videos, 30 videos for each task. The average length of our videos is about 4,000 frames (or 2 minutes) and the entire dataset contains about 800,000 frames.

The selected videos have English transcripts obtained from YouTube's automatic speech recognition (ASR) system. To remove the dependence of results on errors of the particular ASR method, we have manually corrected misspellings and punctuations in the output transcriptions. We believe this step may soon become obsolete given rapid improvements of ASR methods. As we do not modify the content of the spoken language in videos, the transcribed verbal instructions still represent an extremely challenging example of natural language with large variability in the used expressions and terminology. Each word of the transcript is associated with a time interval in the video (usually less than 5 seconds) obtained from the closed caption timings.

Figure 4.2 illustrates two tasks of our newly collected dataset. For each task, we show a subset of 3 events that compose the task. In the following, we refer to these events as *steps* as they are units of a procedure which aims to complete the given *task*. Each step is represented by several sample frames and extracted verbal narrations. Note the large variability of verbal expressions and the terminology in the transcribed narrations as well as the large variability of visual appearance due to viewpoint, used objects, and actions performed in different manner. At the same time, note the consistency of the actions between the different videos and the underlying script of each task.

Manually annotated ground truth. For the purpose of evaluation, we have manually annotated the temporal location in each video of the main steps necessary to achieve the given task. For all tasks, we have defined the ordered sequence of ground truth steps before running our algorithm. The choice of these steps was made by an agreement of 2-3 annotators who have watched the input videos and verified the steps on instructional video websites such as http://www.howdini.com. Checking the instruction video websites helped us to validate the granularity of the steps. While some steps can be occasionally left out in some videos or the ordering slightly modified, overall we have observed a good consistency in the given sequence of instructions among the input videos and we come back to this in section 4.3.1. Given the list of steps for each task, we have manually annotated each time interval in each input video to one of the ground truth steps (or no step). The actions of the individual steps are typically separated by hundreds of frames where the narrator transitions between the steps or explains verbally what is going to happen. Note that some steps could be missing in some videos, or could be present but not described in the narration. The narrations and the actual actions in the video often have coarse temporal alignment since the actions are often described before being performed. Our dataset is available at [Alayrac, 2016]. In the following section we report and discuss the dataset statistics.

4.3.1 Dataset statistics

To illustrate and quantify different properties of our dataset, we introduce three different scores characterizing (i) the consistency of the step ordering, (ii) the frequency of missing steps and (iii) the frequency of step repetitions. We describe these scores in detail below and then measure them on our new dataset.

Task	Changing tire	Performing CPR	Repoting plant	Making coffee	Jumping cars	**Average**
Order consistency error	0.7%	11%	6%	3%	8%	**6%**
Missing steps	16%	32%	30%	28%	27%	**27%**
Repeated steps	4%	50%	7%	11%	0.4%	**14%**

Tab. 4.1.: Statistics of the newly collected instructional video dataset.

Let N be the number of videos for a given task and K the number of steps defined in the ground truth. We assume that the ground truth steps are

given in an ordered fashion, meaning the global order is defined as the sequence $\{1, \ldots, K\}$. For the n-th video, g_n denotes the total number of annotated steps, u_n denotes the number of unique annotated steps, and finally l_n denotes the length of the longest common subsequence between the annotated sequence of steps and the ground truth sequence $\{1, \ldots, K\}$. We then define the following scores. Note that, given the terms defined above, we have: $l_n \leq u_n \leq K \leq g_n$. We then define the following scores.

Order consistency error. The *order consistency error* O is the proportion of (non-repeated) steps that are not consistent with the global ordering. In other words, it measures the number of steps that do not fit the global ordering defined in the ground truth sequence divided by the total number of unique annotated steps. More formally, using the l_n and u_n notation for the n-th video defined above the order consistency error is written as:

$$O := 1 - \frac{\sum_{n=1}^{N} l_n}{\sum_{n=1}^{N} u_n}. \tag{4.1}$$

This score varies between 0 and 1. When the order consistency error is low the videos are consistent with the single ground truth sequence of steps. Note that we report numbers aggregated over all videos instead of reporting the average over all videos. This is to avoid videos with few annotations having a large effect on the resulting overall score. We use this aggregation in all the following metrics.

Missing steps. The *missing steps* score M is the proportion of steps that are visually missing in the videos when compared to the ground truth sequence common to all videos for the task. Using the u_n notation from above, the score is defined as

$$M := 1 - \frac{\sum_{n=1}^{N} u_n}{KN}. \tag{4.2}$$

The score varies between 0 and 1. When this score is 0 all steps of the ground truth sequence are depicted in all videos.

Repeated steps. The *repetition score* R is defined as the proportion of steps that are repeated:

$$R := 1 - \frac{\sum_{n=1}^{N} u_n}{\sum_{n=1}^{N} g_n}. \tag{4.3}$$

The score varies between 0 and 1. When the score is 0 none of the ground truth steps are repeated in the dataset.

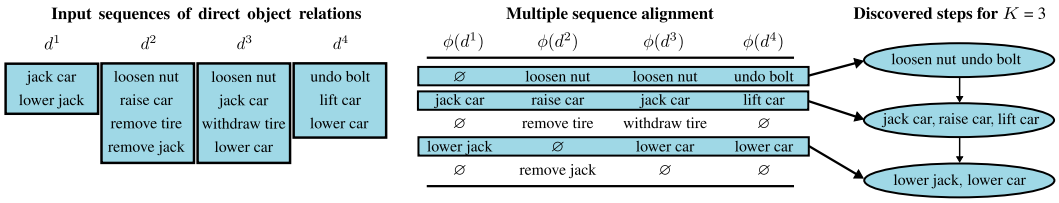


Fig. 4.3.: **Clustering transcribed verbal instructions. Left:** The input raw text for each video is converted into a sequence of direct object relations. Here, an illustration of four sequences from four different videos is shown. **Middle:** Multiple sequence alignment is used to align all sequences together. Note that different direct object relations are aligned together as long as they have the same sense, e.g. “loosen nut" and “undo bolt". **Right:** The main instruction steps are extracted as the $K = 3$ most common steps in all the sequences.

Results. Table 4.1 shows the above scores measured for the five tasks of our instructional videos dataset. Interestingly, we observe relatively low order consistency errors over the five tasks with an average error of only 6%. We believe this can be explained by the goal of instructional videos to give clear, concise and comprehensible audio-visual instructions on how to achieve a given task. On average the videos are missing 27% of the steps, which is relatively high. We believe this illustrates the difficulty of defining the right granularity of the ground truth steps for each task as some optional or implicit steps might be omitted in some videos. Finally, on average 14% of the annotated steps are repeated multiple times in the video. However, examining in detail the per-class results, we observe that this relatively high average score is mainly due to the *Performing CPR* task. CPR is indeed characterized by repetitions of the same steps, namely the alternation between compressions and giving breath. Overall, the relatively high frequency of the missing and repeated steps illustrate the difficulty of our problem. The relatively low order consistency error will be used as an advantage in our method that will be described next.

4.4 Modelling narrated instructional videos

We are given a set of N instructional videos all depicting the same task (such as “changing a tire”). The n-th input video is composed of a video stream of T_n segments of frames $(x_t^n)_{t=1}^{T_n}$ and an audio stream containing a

detailed verbal description of the depicted task. We suppose that the audio description was transcribed to raw text and then processed to a sequence of S_n text tokens $(d_s^n)_{s=1}^{S_n}$. Given this data, we want to automatically recover the sequence of K main steps that compose the given task and locate each step within each input video and text transcription.

We formulate the problem as two clustering tasks, one in text and one in video, applied one after another and linked by joint constraints on two modalities. This two-stage approach is based on the intuition that the variation in natural language describing each task is easier to capture than the visual variability of the input videos. In the first stage, we cluster the text transcripts into a sequence of K main steps to complete the given task. Empirically, we have found (see results in Section 4.5.3) that it is possible to discover the sequence of the K main steps for each task with high precision. However, the text itself gives only a poor localization of each step in each video. Therefore, in the second stage we accurately localize each step in each video by clustering the input videos using the sequence of K steps extracted from text as constraints on the video clustering.

To achieve this, we use two types of constraints between video and text. First, we assume that both the video and the text narration follow the same sequence of steps. This results in a global ordering constraint on the recovered clustering. Second, we assume that people perform the action approximately at the same time that they talk about it. This constraint temporally links the recovered clusters in text and video. The important outcome of the video clustering stage is that the K extracted steps get propagated by visual similarity to videos where the text descriptions are missing or ambiguous.

We first describe the text clustering in Section 4.4.1 and then introduce the video clustering with constraints in Section 4.4.2.

4.4.1 Clustering transcribed verbal instructions

The goal here is to cluster the transcribed verbal descriptions of each video into a sequence of *main steps* necessary to achieve the task. This stage is important as the resulting clusters will be used as constraints for jointly learning and localizing the main steps in video. We assume that the important

steps are common to many of the transcripts and that the sequence of steps is (roughly) preserved in all transcripts. Hence, following [Regneri *et al.*, 2010], we formulate the problem of clustering the input transcripts as a multiple sequence alignment problem. However, in contrast to [Regneri *et al.*, 2010] who cluster manually provided descriptions of each step, we wish to cluster transcribed verbal instructions. Hence our main challenge is to deal with the variability in spoken natural language. To overcome this challenge, we take advantage of the fact that completing a certain task usually involves interactions with objects or people and hence we can extract a more structured representation from the input text stream.

More specifically, we represent the textual data as a sequence of *direct object relations*. A direct object relation d is a pair composed of a verb and its direct object complement, such as “remove tire". This representation contains relevant information about the interaction (*verb*) between the demonstrator and the surrounding environment (*direct object complement*), while being compact. Such a direct object relation can be extracted from the dependency parser of the input transcribed narration [Marneffe *et al.*, 2006]. We denote the set of all different direct object relations extracted from all narrations as $\mathcal{D}$, with cardinality D. For the n-th video, we thus represent the text signal as a sequence of direct object relation tokens: $d^n = (d^n_1, \ldots, d^n_{S_n})$, where the length S_n of the sequence varies from one video clip to another. This step is key to the success of our method as it allows us to convert the problem of clustering raw transcribed text into an easier problem of clustering sequences of direct object relations. The goal is now to extract from the narrations the most common sequence of K main steps to achieve the given task. To achieve this, we first find a globally consistent alignment of the direct object relations that compose all text sequences by solving a multiple sequence alignment problem. Second, we pick from this alignment the K most globally consistent clusters across videos.

Multiple sequence alignment model. We formulate the first stage of finding the common alignment between the input sequences of direct object relations as a multiple sequence alignment problem with the *sum-of-pairs score* [Wang and Jiang, 1994]. In details, a global alignment can be defined by re-mapping each input sequence d^n of tokens to a global common template of L slots, for L large enough. We let $(\phi(d^n))_{1 \leq l \leq L}$ represent the (increasing) re-mapping

for sequence d^n at the new locations indexed by l: $\phi(d^n)_l$ represents the direct object relation put at location l, with $\phi(d^n)_l = \varnothing$ if a slot is left empty (denoting the insertion of a gap in the original sequence of tokens). See the middle of Figure 4.3 for an example of re-mapping. The goal is then to find a global alignment that minimizes the following sum-of-pairs cost function:

$$\sum_{(n,m)} \sum_{l=1}^{L} c(\phi(d^n)_l, \phi(d^m)_l), \tag{4.4}$$

where $c(d_1, d_2)$ denotes the cost of aligning the direct object relations d_1 and d_2 at the same common slot l in the global template. The above cost thus denotes the sum of all pairwise alignments of the individual sequences (the outer sum), where the quality of each alignment is measured by summing the cost c of matches of individual direct object relations mapped into the common template sequence. We use a negative cost when d_1 and d_2 are similar according to the distance in the WordNet tree [Fellbaum, 1998; Miller, 1995] of their verb and direct object constituents, and positive if they are dissimilar (details are given in Section 4.5). As the verbal narrations can talk about many other things than the main steps of a task, we set $c(d, d') = 0$ if either d or d' is $\varnothing$. An illustration of clustering the transcribed verbal instructions into a sequence of K steps is shown in Figure 4.3.

Multiple sequence alignment as a quadratic program. Optimizing the cost (4.4) is NP-hard [Wang and Jiang, 1994] because of the combinatorial nature of the problem. The standard solution from computational biology is to apply a heuristic algorithm that proceeds by incremental pairwise alignment using dynamic programming [Lee *et al.*, 2002]. In contrast, we reformulate the multiple sequence alignment problem given by (4.4) as an integer quadratic program (IQP) with combinatorial constraints. To the best of our knowledge, this is a new formulation of the multiple sequence alignment (MSA) problem for which we can apply the Frank-Wolfe algorithm [Jaggi, 2013c] and consistently obtain in our setting better values of the MSA objective (4.4) than the current state-of-the-art MSA heuristic algorithms. To be able to transform the MSA problem into an IQP, we first correctly encode the important quantities of our problem in a matrix form. There are three relevant quantities: (i) the cost of aligning together two different direct object relations (encoded in C_o), (ii) the content of each individual sequences (encoded in Y_n) and (iii) the mapping of each sequence

to the common template (encoded in U_n). We now give details about these different variables and how to combine them in order to get the final IQP formulation (4.6).

Cost matrix C_o. We summarize the cost of aligning non-empty direct object relations by the matrix $C_o \in \mathbb{R}^{D \times D}$. $(C_o)_{ij}$ is equal to the cost of aligning the i-th and the j-th direct object relation from the dictionary together.

Individual sequences Y_n. We encode the identity of a direct object relation with a D-dimensional indicator vector. The text sequence n can then be represented by an indicator matrix $Y_n \in \{0,1\}^{S_n \times D}$. The j-th row of Y_n indicates which direct object relations is evoked at the j-th position.

Mapping of the sequence U_n. Similarly, the token re-mapping $(\phi(d^n))_{1 \leq l \leq L}$ can be represented as a $L \times D$ indicator matrix where each row l encodes which token is appearing in slot l (and a whole row of zero is used to indicates an empty $\varnothing$ slot). This re-mapping can be constructed from the following two pieces of information: the input sequence Y_n and the mapping of each element of the sequence to the global template. We represent the latter by the decision matrix $U_n \in \{0,1\}^{S_n \times L}$ that gives which element of the sequence s is re-mapped to which global template slot l. We thus have $\phi(d^n) = U_n^\top Y_n$ (as a $L \times D$ indicator matrix).

Quadratic cost. Given this encoding, the cost matrix C_o, and the fact that the alignment of empty slots has zero cost, we can rewrite the MSA problem that minimizes the sum-of-pairs objective (4.4) as follows:

$$\begin{aligned} \underset{U_n, n \in \{1,\ldots,N\}}{\text{minimize}} \quad & \sum_{(n,m)} \mathrm{Tr}(U_n^\top Y_n C_o Y_m^\top U_m) \\ \text{subject to} \quad & U_n \in \mathcal{U}_n, \; n = 1, \ldots, N. \end{aligned} \tag{4.5}$$

In the above equation, the trace (Tr) is computing the cost of aligning sequence m with sequence n (the inner sum in (4.4)). Moreover, $\mathcal{U}_n$ is a constraint set that encodes the fact that U_n has to be a valid (increasing) re-mapping.[4] We can then eliminate the video index n by simply stacking the assignment matrices U_n in one matrix U of size $S \times L$. Similarly, we denote

[4]More formally $\mathcal{U}_n := \{U \in \{0,1\}^{S_n \times L}$ s.t. $U\mathbf{1}_L = \mathbf{1}_{S_n}$ and $\forall l, (U_{sl} = 1) \Rightarrow (\forall s' > s, l' \leq l, U_{s'l'} = 0)\}$.

Y the $S \times D$ matrix which is obtained by the concatenation of all the Y_n matrices. Finally, we can rewrite the equation (4.5) as a quadratic program over the (integer) variable U:

$$\underset{U}{\text{minimize}}\ \mathrm{Tr}(U^\top BU), \text{ subject to } U \in \mathcal{U}. \tag{4.6}$$

In this equation, the $S \times S$ matrix B is obtained from the input sequences and the cost between different direct object relations by computing $B := YC_oY^\top$. It represents the pairwise cost at the token level, i.e. the cost of aligning token s in one sequence to token s' in another sequence.

Optimization using Frank-Wolfe. Problem (4.6) is an integer quadratic program with combinatorial constraints for which the Frank-Wolfe optimization algorithm has been used recently with increasing success [Bojanowski *et al.*, 2014; Jaggi, 2013c; Joulin *et al.*, 2014a; Lacoste-Julien and Jaggi, 2015a]. We therefore also apply Frank-Wolfe here. Following [Bojanowski *et al.*, 2014], we first perform a continuous relaxation of the set of constraints $\mathcal{U}$ by replacing it with its convex hull $\bar{\mathcal{U}}$. Note that the Frank-Wolfe optimization algorithm [Jaggi, 2013c] can solve quadratic program over constraint sets for which we have access to an efficient linear minimization oracle. In the case of $\mathcal{U}$, the linear oracle can be solved exactly with a dynamic program. We note here that even with the continuous relaxation over $\bar{\mathcal{U}}$, the resulting problem is still non-convex because B is not positive semidefinite. However, the standard convergence proof for Frank-Wolfe can easily be extended to show that it converges at a rate of $O(1/\sqrt{k})$ to a stationary point on non-convex objectives [Lacoste-Julien, 2016]. Once the algorithm has converged to a (local) stationary point, we need to round the fractional solution to obtain a valid encoding U. We follow here a similar rounding strategy that was originally proposed by [Chari *et al.*, 2015] and then improved in [Joulin *et al.*, 2014a]: we pick the visited corner (which is necessarily integer) which was given as a solution to the linear minimization oracle (this is called Frank-Wolfe rounding) and gave the minimal cost during the course of optimization.

We have observed empirically (see results in Section 4.5.2) that the Frank-Wolfe algorithm was giving better solutions (in terms of objective (4.4)) than the state-of-the-art heuristic procedures for this task [Higgins and Sharp, 1988; Lee *et al.*, 2002]. Our Frank-Wolfe based solvers also offer us greater

flexibility in defining the alignment cost and scale better with the length of input sequences and the vocabulary of direct object relations.

Extracting the main steps. After a global alignment is obtained, we sort the global template l by the number of direct object relations aligned to each slot. Given K as input, the top K slots give the main instruction steps for the task, unless there are multiple steps with the same support, which go beyond K. In this case, we pick the next smaller number below K which excludes these ties, allowing the choice of an *adaptive* number of main instruction steps when there is not enough saliency for the last steps. This strategy essentially selects $k \leq K$ salient steps, while refusing to make a choice among steps with equal support that would increase the total number of steps beyond K. As we will see in our results in Section 4.5.3, our algorithm sometimes returns a much smaller number than K for the main instruction steps, giving more robustness to the exact choice of parameter K.

Encoding of the output. We post-process the output of multiple sequence alignment into an assignment matrix $R_n \in \{0,1\}^{S_n \times K}$ for each input video n, where $(R_n)_{sk} = 1$ means that the direct object token d_s^n has been assigned to step k. If a direct object has not been assigned to any step, the corresponding row of the matrix R_n will be zero.

4.4.2 Discriminative clustering of videos under text constraints

Given the output of the text clustering that identified the important K steps forming a task, we now want to find their temporal location in the video signal. We formalize this problem as looking for an assignment matrix $Z_n \in \{0,1\}^{T_n \times K}$ for each input video n, where $(Z_n)_{tk} = 1$ indicates the visual presence of step k at time interval t in video n, and T_n is the length of video n. Similarly as for the assignment matrix R_n, we allow the possibility that a whole row of Z_n is zero, indicating that no step is visually present for the corresponding time interval.

We propose to tackle this problem using a discriminative clustering approach with global ordering constraints, as was successfully used in the past for the

Changing a tire			
GT(11)	$K \leq 7$	$K \leq 10$	$K \leq 15$
put brake on			
get tools out		**get tire**	**get tire**
start loose	**loosen nut**	**loosen nut**	**loosen nut**
			lift car
	put jack	**put jack**	**put jack**
			raise vehicle
jack car	**jack car**	**jack car**	**jack car**
unscrew wheel	**remove nut**	**remove nut**	**remove nut**
remove wheel		**take wheel**	**take wheel**
put wheel	**take tire**	**take tire**	**take tire**
screw wheel		**put nut**	**put nut**
lower car	**lower jack**	**lower jack**	**lower jack**
			remove jack
tight wheel	**tighten nut**	**tighten nut**	**tighten nut**
put things back			**take tire**
Precision	0.85	0.90	0.71
Recall	0.54	0.90	0.90

Make coffee			
GT(10)	$K \leq 7$	$K \leq 10$	$K \leq 15$
grind coffee			
put filter			
add coffee		**put coffee**	**put coffee**
even surface			
		fill chamber	**fill chamber**
			make noise
fill water	**fill water**	**fill water**	**fill water**
screw top		**put filter**	**put filter**
			fill basket
		see steam	**see steam**
put stove	**take minutes**	**take minutes**	**take minutes**
	make coffee	**make coffee**	**make coffee**
see coffee	**see coffee**	**see coffee**	**see coffee**
withdraw stove			**turn heat**
pour coffee	**make cup**	**make cup**	**make cup**
			pour coffee
Precision	0.80	0.67	0.54
Recall	0.40	0.60	0.70

Repot a plant			
GT(7)	$K \leq 7$	$K \leq 10$	$K \leq 15$
cover hole			**take piece**
			keep soil
			stop soil
take plant	**take plant**	**take plant**	**take plant**
put soil	**use soil**	**use soil**	**use soil**
loosen root	**loosen soil**	**loosen soil**	**loosen soil**
place plant	**place plant**	**place plant**	**place plant**
add top	**add soil**	**add soil**	**add soil**
			fill pot
			get soil
			give drink
water plant	**water plant**	**water plant**	**water plant**
			give watering
Precision	1.00	1.00	0.54
Recall	0.86	0.86	1.00

Performing CPR			
GT(11)	$K \leq 7$	$K \leq 10$	$K \leq 15$
open airway	**open airway**	**open airway**	**open airway**
check response			
call 911			
check breathing			
check pulse		**put hand**	**put hand**
	tilt head	**tilt head**	**tilt head**
	lift chin	**lift chin**	**lift chin**
	give breath	**give breath**	**give breath**
give breath	**do compr.**	**do compr.**	**do compr.**
give compression	**open airway**	**open airway**	**open airway**
		start compr.	**start compr.**
			continue cpr
		do compr.	**do compr.**
			put hand
		give breath	**give breath**
Precision	0.50	0.40	0.33
Recall	0.43	0.57	0.57

Jumping cars			
GT(12)	$K \leq 7$	$K \leq 10$	$K \leq 15$
get cars			
open hood			
			have terminal
			attach cable
connect red A	**connect cable**	**connect cable**	**connect cable**
	charge battery	**charge battery**	**charge battery**
connect red B	**connect end**	**connect end**	**connect end**
connect black A			
connect ground			
start car A	**start car**	**start car**	**start car**
start car B			**start vehicle**
			start engine
remove ground	**remove cable**	**remove cable**	**remove cable**
remove black A	**disconnect cable**	**disconnect cable**	**disconnect cable**
remove red B			
remove red A			
Precision	0.83	0.83	0.69
Recall	0.42	0.42	0.67

Tab. 4.2.: Automatically recovered sequences of steps for the five tasks considered in this work. Each recovered step is represented by one of the aligned direct object relations (shown in bold). Note that most of the recovered steps correspond well to the ground truth steps (showed in italic). The results are shown for setting the maximum number of discovered steps, $K = \{7, 10, 15\}$. Note how our method automatically selects less than K steps in some cases. These are the automatically chosen $k \leq K$ steps that are the most salient in the aligned narrations as described in Section 4.4.1.

temporal localization of actions in videos [Bojanowski *et al.*, 2014], but with additional *weak temporal constraints*. In contrast to [Bojanowski *et al.*, 2014] where the order of actions was manually given for each video, our multiple sequence alignment approach automatically discovers the main steps. More importantly, we also use the *text caption timing* to provide a fine-grained weak temporal supervision for the visual appearance of steps, which is described next.

Temporal weak supervision from text. From the output of the multiple sequence alignment (encoded in the matrix $R_n \in \{0,1\}^{S_n \times K}$), each direct object token d_s^n has been assigned to one of the possible K steps, or to no step at all. We use the tokens that have been assigned to a step as a constraint on the visual appearance of the same step in the video (using the assumption

that people do what they say approximately when they say it). We encode the closed caption timing alignment by a binary matrix $A_n \in \{0,1\}^{S_n \times T_n}$ for each video, where $(A_n)_{st}$ is 1 if the s-th direct object is mentioned in a closed caption that overlaps with the time interval t in video. Note that this alignment is only approximate as people usually do not perform the action exactly at the same time that they talk about it, but instead with a varying delay. Second, the alignment is noisy as people typically perform the action only once, but often talk about it multiple times (e.g. in a summary at the beginning of the video). We address these issues by the following two *weak supervision* constraints. First, we consider a larger set of possible time intervals $[t - \Delta_b, t + \Delta_a]$ in the matrix A rather than the exact time interval t given by the timing of the closed caption. Δ_b and Δ_a are global parameters fixed either qualitatively, or by cross-validation if labeled data is provided. Second, we put as a constraint that the action happens at least once in the set of all possible video time intervals where the action is mentioned in the transcript (rather than every time it is mentioned). These constraints can be encoded as the following linear inequality constraint on Z_n: $A_n Z_n \geq R_n$.[5]

Ordering constraint. In addition, we also enforce that the temporal order of the steps appearing visually is consistent with the discovered script from the text, encoding our assumption that there is a common ordered script for the task across videos. We encode these sequence constraints on Z_n in a similar manner to [Bojanowski *et al.*, 2015], which was shown to work better than the encoding used in [Bojanowski *et al.*, 2014]. In particular, we only predict the *most salient* time interval in the video that describes a given step. This means that a particular step is assigned to *exactly one* time interval in each video. We denote by $\mathcal{Z}_n$ this sequence ordering constraint set.

Discriminative clustering. The main motivation behind discriminative clustering is to find a *clustering* of the data that can be easily recovered by a *linear classifier* through the minimization of an appropriate *cost function* over the assignment matrix Z_n. The approach introduced in [Bach and Harchaoui, 2007] allows to easily add prior information on the expected clustering. Such priors have been recently introduced in the context of aligning video and

[5] When $R_{sk} = 0$, then this constraint does not do anything. When $R_{sk} = 1$ (i.e. the text token s was assigned to the main action k), then the constraint enforces that $\sum_{t \in A_{s\cdot}} Z_{tk} \geq 1$, where $A_{s\cdot}$ represents which video frames are temporally close to the caption time of the text token s. It thus then enforces that at least one temporally close video frame is assigned to the main action k.

text [Bojanowski *et al.*, 2014; Bojanowski *et al.*, 2015] in the form of ordering constraints over the latent label variables. Here we use a similar approach to cluster the N input video streams (x_t) into a sequence of K steps, as follows. We represent each time interval by a d-dimensional feature vector. The feature vectors for the n-th video are stacked in a $T_n \times d$ design matrix denoted by X_n. We denote by X the $T \times d$ matrix obtained by the concatenation of all X_n matrices (and similarly, by Z, R and A the appropriate concatenation of the Z_n, R_n and A_n matrices over n). In order to obtain the temporal localization into K steps, we learn a linear classifier represented by a $d \times K$ matrix denoted by W. This model is shared among all videos.

The target assignment $\hat{Z}$ is found by minimizing the clustering cost function h under both the consistent script ordering constraints $\mathcal{Z}$ and our weak supervision constraints:

$$\underset{Z}{\text{minimize}} \quad h(Z) \quad \text{s.t.} \quad \underbrace{Z \in \mathcal{Z}}_{\text{ordered script}} \;, \quad \underbrace{AZ \geq R}_{\substack{\text{weak textual}\\ \text{constraints}}} . \tag{4.7}$$

The clustering cost $h(Z)$ is given as in DIFFRAC [Bach and Harchaoui, 2007] as:

$$h(Z) = \min_{W \in \mathbb{R}^{K \times d}} \underbrace{\tfrac{1}{2T}\|Z - XW\|_F^2}_{\text{Discriminative loss on data}} + \underbrace{\tfrac{\lambda}{2}\|W\|_F^2}_{\text{Regularizer}} . \tag{4.8}$$

The first term in (4.8) is the discriminative loss on the data that measures how easy the input data X is separable by the linear classifier W when the target classes are given by the assignments Z. For the squared loss considered in eq. (4.8), the unique optimal weights W^* minimizing (4.8) can be found in closed form, which significantly simplifies the computation:

$$W^*(Z) = (X^\top X + T\lambda I_d)^{-1} X^\top Z, \tag{4.9}$$

where I_d is the d-dimensional identity matrix. We obtain the explicit form for $h(Z)$ by substituting the expression (4.9) for $W^*(Z)$ in equation (4.8) and properly simplifying the expression:

$$h(Z) = \tfrac{1}{2T}\text{Tr}(ZZ^\top B), \tag{4.10}$$

where $B := I_T - X(X^\top X + T\lambda I_d)^{-1}X^\top$ is a strictly positive definite matrix (and so h is actually strongly convex). The clustering cost is a quadratic function in Z, encoding how the clustering decisions in one interval t interact with the clustering decisions in another interval t'. Next, we explain how we can optimize the clustering cost $h(Z)$ subject to the constraints of problem (4.7) using the Frank-Wolfe algorithm.

Frank Wolfe algorithm for minimizing $h(Z)$. The Frank Wolfe algorithm is well adapted for our problem as we know how to efficiently solve linear programs over our constraints. Recall that theses constraints encode several concepts. First, it imposes the temporal consistency between the text stream and the video stream. We recall that this constraint was written as $AZ \geq R$, where A encodes the temporal alignment constraints between video and text (type I). Second, it includes the event ordering constraints within each video input (type II). Finally, it encodes the fact that each event is assigned to exactly one time interval within each video (type III). The last two constraints are encoded in the set of constraints $\mathcal{Z}$. To summarize, let $\tilde{\mathcal{Z}}$ denote the resulting discrete feasible space for Z i.e. $\tilde{\mathcal{Z}} := \{Z \in \mathcal{Z} \,|\, AZ \geq R\}$. We are then left with a problem in Z which is still hard to solve because the set $\tilde{\mathcal{Z}}$ is not convex. To approximately optimize h over $\tilde{\mathcal{Z}}$, we again follow the strategy of [Bojanowski *et al.*, 2014; Bojanowski *et al.*, 2015] by replacing $\tilde{\mathcal{Z}}$ by its convex hull $\text{conv}(\tilde{\mathcal{Z}})$. We then use the Frank-Wolfe algorithm to get a fractional solution $Z^* \in \text{conv}(\tilde{\mathcal{Z}})$. Finally, we find a feasible candidate $\hat{Z} \in \tilde{\mathcal{Z}}$ by using a rounding procedure. We now give the details of these steps.

Linear program for our constraints. First, we note that the linear oracle of the Frank-Wolfe algorithm can be solved separately for each video n. Indeed, because we solve a linear program, there is no quadratic term that brings dependence between different videos in the objective, and moreover all the constraints are blockwise in n. Thus, in the following, we will give details for one video only by adding an index n to $\tilde{\mathcal{Z}}$, to Z and to T. Formally, the linear oracle corresponds to the following problem:

$$\min_{Z_n \in \tilde{\mathcal{Z}}_n} \text{Tr}(C_n^\top Z_n), \tag{4.11}$$

where $C_n \in \mathbb{R}^{T_n \times K}$ is a cost matrix that typically comes from the gradient computation of h with respect to Z_n at the current iterate. We now show that this problem can be solved by an efficient dynamic program.

Dynamic program. Using the formalism of [Bojanowski *et al.*, 2015], we cast problem (4.11) as a search for an optimal path inside a cost matrix $\tilde{C}$ that we can solve using dynamic programing. From the constraint of type III (unique prediction per step), we know that each column k of Z_n has exactly one 1 (to be found). From the ordering constraint (type II), we know that if $(Z_n)_{tk} = 1$, then the only possible locations for a 1 in the $(k+1)$-th column is for $t' > t$ (i.e. the pattern of 1's is going downward when traveling from left to right in Z_n). Note that there can be "jumps" in between the time assignment for two subsequent steps k and $k+1$. In order to encode this possibility using a continuous path search in a matrix, we insert dummy columns into the cost matrix C. We first subtract the minimum value from C and then insert columns filled with zeros in between every pair of columns of C. In the end, we pad C with an additional row filled with zeros at the bottom. Finally, the problem that we are interested in is subject to the additional linear constraints given by the clustering of text transcripts (constraints of type I). These constraint can be added by constraining the path in the dynamic programming algorithm. This can be done for instance by setting an infinite alignment cost outside of the constrained region. The resulting cost matrix $\tilde{C}$ is of size $(T_n + 1) \times (2K + 1)$ and is illustrated along with the corresponding update rules in Figure 4.4.

Extracting discrete temporal locations. At the end of the Frank-Wolfe optimization algorithm, we obtain a continuous solution for assignment matrix Z_n^* that encodes the (fractional) solution for the temporal location of each step in video n. By stacking matrices for all videos together, we obtain a continuous solution Z^*. From the definition of h, we can also look at the corresponding model $W^*(Z^*)$ defined in equation (4.9) which is shared among all videos. All Z_n^* have to be rounded in order to obtain a feasible point for the initial, non relaxed problem. Several rounding options were suggested in [Bojanowski *et al.*, 2015]; we found experimentally that the option using W^* gives better results in our case. More precisely, in order to get a good feasible binary matrix $\hat{Z}_n \in \tilde{Z}_n$, we solve the following problem: $\min_{Z_n \in \tilde{\mathcal{Z}}_n} \|Z_n - X_n W^*\|_F^2$. By expanding the norm, we notice that this corresponds to a simple linear program over $\tilde{\mathcal{Z}}_n$ as in equation (4.11) that can be solved using again the same dynamic program detailed above. Finally, we stack these rounded matrices $\hat{Z}_n$ to obtain our predicted assignment matrix $\hat{Z} \in \tilde{\mathcal{Z}}$.

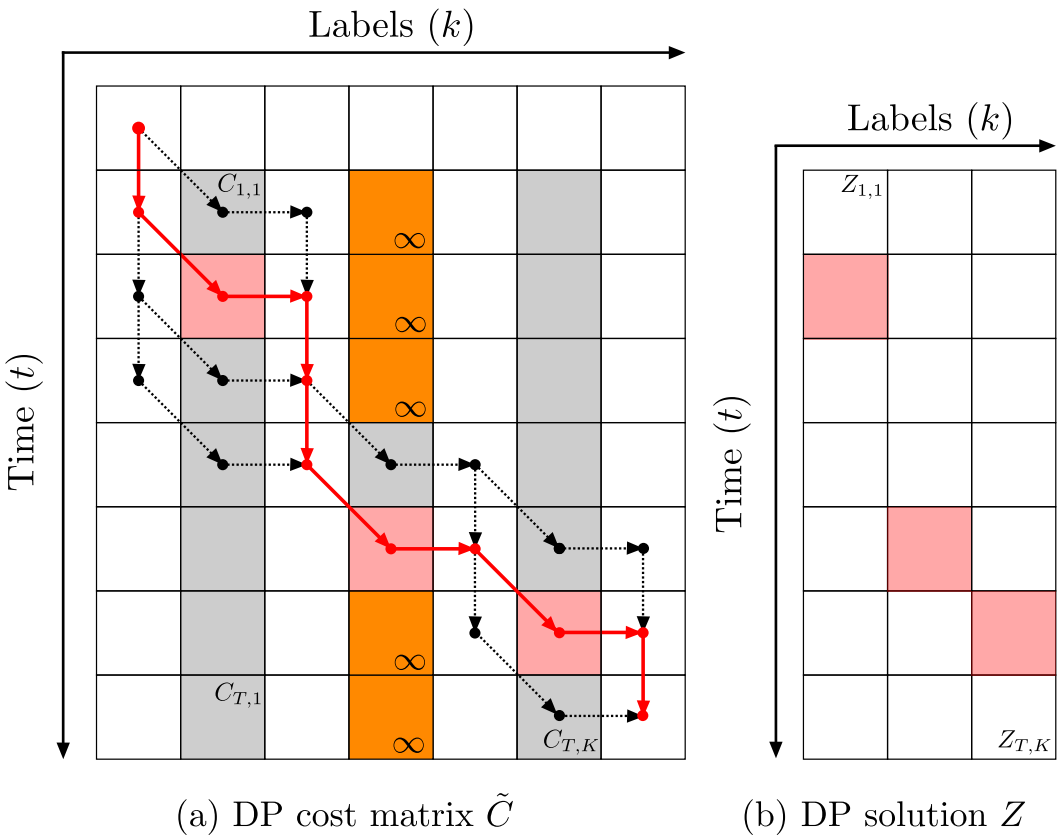


Fig. 4.4.: Illustration of the dynamic programming solution to the linear program (4.11). (a) shows an example cost matrix $\tilde{C}$ and the corresponding optimal path in red. The gray entries in the matrix $\tilde{C}$ correspond to the values from the original cost matrix C (see text). The white entries have minimal cost and are thus always preferred over any gray entry. The orange entries have maximal cost, e.g. ∞, and correspond to text constraints (type I). These constraints reduce the number of possible paths. The obtained latent variable solution Z is displayed in (b). Red and white entries respectively correspond to the value 1 and 0.

4.5 Experimental evaluation

In this section, we first give the implementation details including the description of extracted text and video features. Then we present results divided into five main experiments: (i) in Section 4.5.2, we evaluate the performance of our method for solving the multiple sequence alignment problem, (ii) in Section 4.5.3, we evaluate the quality of steps extracted from the transcribed narrations, (iii) in Section 4.5.4, we evaluate the temporal localization accuracy of the recovered steps in video using constraints derived from the transcribed narrations, (iv) in Section 4.5.5, we explore the effects of the different parameters of our method, and (v) in Section 4.5.6, we investigate the possibility of iterating the two stages of our clustering approach. All the data and code are available on our project webpage [Alayrac, 2016].

Task	Changing tire	Performing CPR	Repotting plant	Making coffee	Jumping cars
Poa [Lee *et al.*, 2002]	11.30	-3.82	1.65	-2.99	4.55
Ours using Frank-Wolfe	**-5.18**	**-4.51**	**-3.55**	**-3.86**	**-4.67**

Tab. 4.3.: Comparison of different optimization approaches for solving problem (4.6). (Objective value, lower is better).

4.5.1 Implementation details

Here we describe the extracted video and text features, followed by the details of the WordNet distance used in the multiple sequence alignment algorithm. Finally, we also discuss the running time of the proposed approach and its scalability.

Video and text features. We represent the transcribed narrations as sequences of direct object relations. For this purpose, we run a dependency parser [Marneffe *et al.*, 2006] on each transcript. We lemmatize all direct object relations and keep the ones for which the direct object corresponds to nouns. To represent a video, we use motion descriptors in order to capture actions (such as loosening, jacking-up, giving compressions) and frame appearance descriptors to capture the depicted objects (e.g., tire, jack, car). In detail, we split each video into 10-frame time intervals and represent each interval by its motion and appearance descriptors aggregated over a longer block of 30 frames. The motion representation is a histogram of local optical

flow (HOF) descriptors aggregated into a single bag-of-visual-word vector of 2,000 dimensions [Wang and Schmid, 2013]. The visual vocabulary is generated by k-means on a separate large set of training descriptors. To capture the depicted objects in the video, we apply the VGG-verydeep-16 CNN [Simonyan and Zisserman, 2015] over each frame in a sliding window manner over multiple scales. This can be done efficiently in a fully convolutional manner. The resulting 512-dimensional feature maps of conv5 responses are then aggregated into a single bag-of-visual-word vector of 1,000 dimensions, which aims to capture the presence/absence of different objects within each video block. A similar representation (aggregated into compact VLAD descriptor) was shown to work well recently for a variety of recognition tasks [Cimpoi *et al.*, 2015]. The bag-of-visual-word vectors representing the motion and the appearance are normalized using the Hellinger normalization and then concatenated into a single 3,000 dimensional vector representing each time interval.

WordNet distance. For the multiple sequence alignment presented in Section 4.4.1, we set $c(d_1, d_2) = -1$ if d_1 and d_2 have both their verbs and direct objects that match exactly in the Wordnet tree (distance equal to 0). Otherwise we set $c(d_1, d_2)$ to be 100. We found this relatively strict requirement to work well ensuring high-precision of the resulting alignment while taking advantage of the semantic information contained in the WordNet tree as it allows to align two expressions that are not identical as long as they are in the same node of the tree. While we have observed the WordNet distance to work well, we also investigate using distance based on the word2vec embedding Mikolov *et al.*, 2013. In detail, we use the average cosine similarity between the verbs and the objects composing the direct object relations. If the cosine similarity is greater than 0.7[6], we set $c(d_1, d_2) = -1$, otherwise $c(d_1, d_2) = 100$. We use word2vec embedding pre-trained on GoogleNews. We have observed that pre-training the embedding on instruction videos results in a similar performance.

Running time and scalability. For the multiple sequence alignment problem (with $N = 30$, $D = 50$), 400 iterations of our algorithm take less than 10 seconds on a single 2.40 GHz core, which is sufficient for convergence to a stationary point. For video alignment (with $N = 30$, $d = 3000$, $T =$

[6]This threshold has been chosen manually to ensure a good tradeoff between precision and recall.

20000), 600 iterations take less than 20 minutes on the same hardware configuration.

4.5.2 Results of multiple sequence alignment

In this section we evaluate the optimization performance of our relaxation of the multiple sequence alignment (MSA) problem. The (MSA) problem (4.6) is in general NP-hard [Wang and Jiang, 1994], as is typical for integer quadratic programs. However, significant amount of work has been done in computational biology to develop efficient heuristics to solve this problem, as it arises in several important applications. We briefly describe below some of the existing heuristics, and then compare the optimization performance with our Frank-Wolfe optimization approach, which gave surprisingly good empirical results.[7]

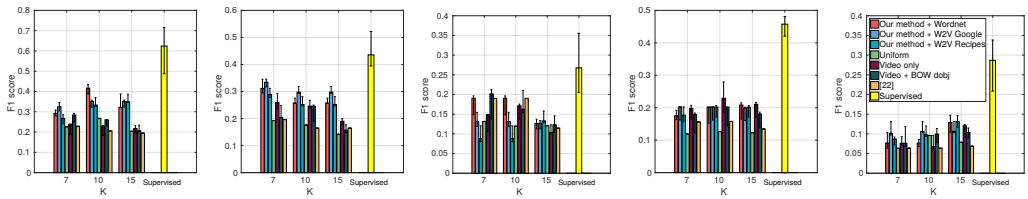
Standard methods. We compare performance to the standard state-of-the-art method for multiple sequence alignment [Lee *et al.*, 2002]. Similarly to [Higgins and Sharp, 1988], this method first aligns a pair of sequences and merges them in a common template. Then it aligns a new sequence to the template and updates the template. It continues aligning additional sequences until no sequence is left. Differently from [Higgins and Sharp, 1988], this method represents the template by a partial order graph instead of a simple linear representation, which results in a higher accuracy of the final alignment. For the experiments, we use the author's implementation [*Partial order alignment code for Multiple Sequence Alignment* n.d.].

Results. In Table 4.3, we give the value of the objective (4.6) (lower is better) for the rounded solutions obtained by the two different optimization approaches for the MSA problem on our five tasks. Interestingly, we observe that the Frank-Wolfe algorithm consistently outperforms the state-of-the-art MSA method of [Lee *et al.*, 2002] in our setting.

[7] We stress here that we do not claim that our formulation of the multiple sequence alignment (MSA) problem as a quadratic program outperforms the state-of-the-art computational biology heuristics for problems *arising in biology*. We report our observations when multiple sequence alignment is applied to our problem set-up, which might have a structure for which these heuristics are not appropriate.

4.5.3 Results of step discovery from text narrations

In the previous section we evaluated the optimization performance of our multiple sequence alignment algorithm described in Sec 4.4.1. In this section, we evaluate how well our multiple sequence alignment approach discovers the main steps of each task from the text narrations. Table 4.2 shows the automatically recovered sequences of steps for the five tasks considered in this work. The results are shown for setting the maximum number of discovered steps, $K = \{7, 10, 15\}$. Note how our method automatically selects less than K steps in some cases. These are the automatically chosen $k \leq K$ steps that are the most salient in the aligned narrations as described in Section 4.4.1. This is notably the case for the *Repotting a plant* task. Even for $K \leq 10$, the algorithm recovers only 6 steps that match very well the seven ground truth steps for this task. This saliency based task selection is important because it allows for a better precision at high K without lowering much the recall. Please note also how the steps and their ordering recovered by our method correspond well to the ground truth steps for each task. For *CPR*, our method recovers fine-grained steps e.g. *tilt head*, *lift chin*, which are not included in the main ground truth steps, but nevertheless could be helpful in some situations. For *Changing tire*, we also recover more detailed actions such as *remove jack* or *put jack*. In some cases, our method recovers repeated steps. For example, for *CPR* our method learns that one has to alternate between *giving breath* and *performing compressions* even if this alternation was not annotated in the ground truth. For *Jumping Cars* our method learns that cables need to be connected twice (to both cars). These results demonstrate that our method is able to automatically discover meaningful scripts describing very different tasks. The results also show that the constraint of a single script providing an ordering of events is a reasonable prior for a variety of different tasks.



(a) Change tire (**11**) **(b)** Perform CPR (**7**) **(c)** Repot plant (**7**) **(d)** Make coffee (**10**) **(e)** Jump car (**12**)

Fig. 4.5.: Results for temporally localizing recovered steps in the input videos. We give in **bold** the number of ground truth steps.

4.5.4 Results of localizing instruction steps in video

In the previous section, we have evaluated the quality of the sequences of steps recovered from the transcribed narrations. In this section, we evaluate how well we localize the individual instruction steps in the video by running our complete two-stage approach from Section 4.4.

Evaluation metric. To evaluate the temporal localization, we need to have a one-to-one mapping between the discovered steps in the videos and the ground truth steps. Following [Liao, 2014], we look for a one-to-one global matching (shared across all videos of a given task) that maximizes the evaluation score for a given method (using the Hungarian algorithm). Note that this mapping is used only for evaluation, the algorithm does not have access to the ground truth annotations for learning.

The goal is to evaluate whether each ground truth step has been correctly localized in all instructional videos. We thus use the *F1 score* that combines precision and recall into a single score as our evaluation measure. For a given video and a given recovered step, our video clustering method predicts exactly one video time interval t. This detection is considered correct if the time interval falls inside any of the corresponding ground truth intervals, and incorrect otherwise (resulting in a false positive for this video). We compute the recall across all steps and videos, defined as the ratio of the number of correct predictions over the total number of possible ground truth steps across videos. A recall of 1 indicates that every ground truth step has been correctly detected across all videos. The recall decreases towards 0 when we miss some ground truth steps (missed detections). This happens either

because this step was not recovered globally, or because it was detected in the video at an incorrect location. This is because the algorithm predicts exactly one occurrence of each step in each video. Similarly, precision measures the proportion of correct predictions among all $N \cdot K_{\text{pred}}$ possible predictions, where N is the number of videos and K_{pred} is the number of main steps used by the method. The F1 score is the harmonic mean of precision and recall, giving a score that ranges between 0 and 1, with the perfect score of 1 when all the steps are predicted at their correct locations in all videos.

Hyperparameters. We set the values of text constraint time interval parameters, Δ_b and Δ_a, to 0 and 10 seconds. The setting is the same for all five tasks. This models the fact that typically each step is first described verbally and then performed on the camera. We set $\lambda = 1/(NK_{\text{pred}})$ for all methods that use (4.8). See Section 4.5.5 for an analysis of the effects of these hyperparameters.

Baselines. We compare results to four baselines. To demonstrate the difficulty of our dataset, we first evaluate a "Uniform" baseline, which simply distributes instructions steps uniformly over the entire instructional video. The second baseline "Video only" [Bojanowski *et al.*, 2014] does not use the narration and performs only discriminative clustering on visual features with a global order constraint. In particular, we compare to the improved model from [Bojanowski *et al.*, 2015], which does not require a "background class" and yields a stronger baseline equivalent to our model (4.7) *without* the weak textual constraints. The third baseline "Video + BOW dobj" adds text-based features to the "Video only" baseline (by concatenating the text and video features in the discriminative clustering approach). Here the goal is to evaluate the benefits of our two-stage clustering approach, in contrast to this single-stage clustering baseline. The text features are bag-of-words histograms over a fixed vocabulary of direct object relations. An alternative set-up creating separate bag-of-words histograms for nouns and verbs gave similar results and is not considered here for brevity. The fourth baseline is our own implementation of the alignment method of [Malmaud *et al.*, 2015] (without the supervised vision refinement procedure that requires a set of pre-trained visual classifiers that are not available a-priori in our case). We use [Malmaud *et al.*, 2015] to re-align the speech transcripts to the sequence of steps discovered by our method from Section 4.4.1 (as a

proxy for the recipe assumed to be known in [Malmaud *et al.*, 2015]).[8] To assess the difficulty of the task and dataset, we also compare results with a "Supervised" approach. For that, we divide the N input videos in 5 different folds. One fold is kept for the test set while the the other 4 are used as the trainining/validation dataset. With the 4 remaining folds, we perform a 4-fold cross validation in order to choose the hyperparameter λ. Once the hyper parameter is fixed, we retrain a model on the 4 folds and evaluate it on the test set. By iterating over the five possible test folds, we report variation in performance with error bars. The supervised method learns classifiers W for all the visual steps. This is achieved by minimizing the cost defined in (4.7) under the ground truth annotation constraints on the training set. At test time, we simply apply classifiers W on the test set performing least-square prediction of Z_{test} with the ordering constraints.

Error bars for unsupervised methods. For the unsupervised methods we compute the error bars in the following manner. Recall that the Frank-Wolfe algorithm solves a continuous relaxation of the integer problem (4.7). To obtain an integer solution, we round the continuous solution using the rounding method described in Section 4.4.2. This rounding procedure is performed at each iteration of the optimization. When the stopping criterion of the Frank-Wolfe algorithm is reached (fixed number of iterations or target sub-optimality in practice), we have as many rounded solutions as the number of iterations. Our output integer solution is the rounded solution that achieves the lowest objective. Note that we are only guaranteed to lower the objective in the continuous domain whereas the objective of the rounded solution can increase (and often does), and hence there is no guarantee that the last rounded solution will have the lowest objective. In order to illustrate the variation of the performance during optimization, we define error bars as the interval determined by the minimal and maximal performance (measured by the F1 score) obtained *after* visiting the best rounded point (the integer solution with the lowest objective). This also explains why the error bars in Figure 4.5 are not necessarily symmetric. Overall, the observed variation is not very high, highlighting the stability of the optimization procedure.

Results. Results for localizing the discovered instruction steps are shown in Figure 4.5. In order to perform a fair comparison to the baseline methods

[8] Note that our method finds at the same time the sequence of steps (a recipe in [Malmaud *et al.*, 2015]) and the alignment of the transcripts.

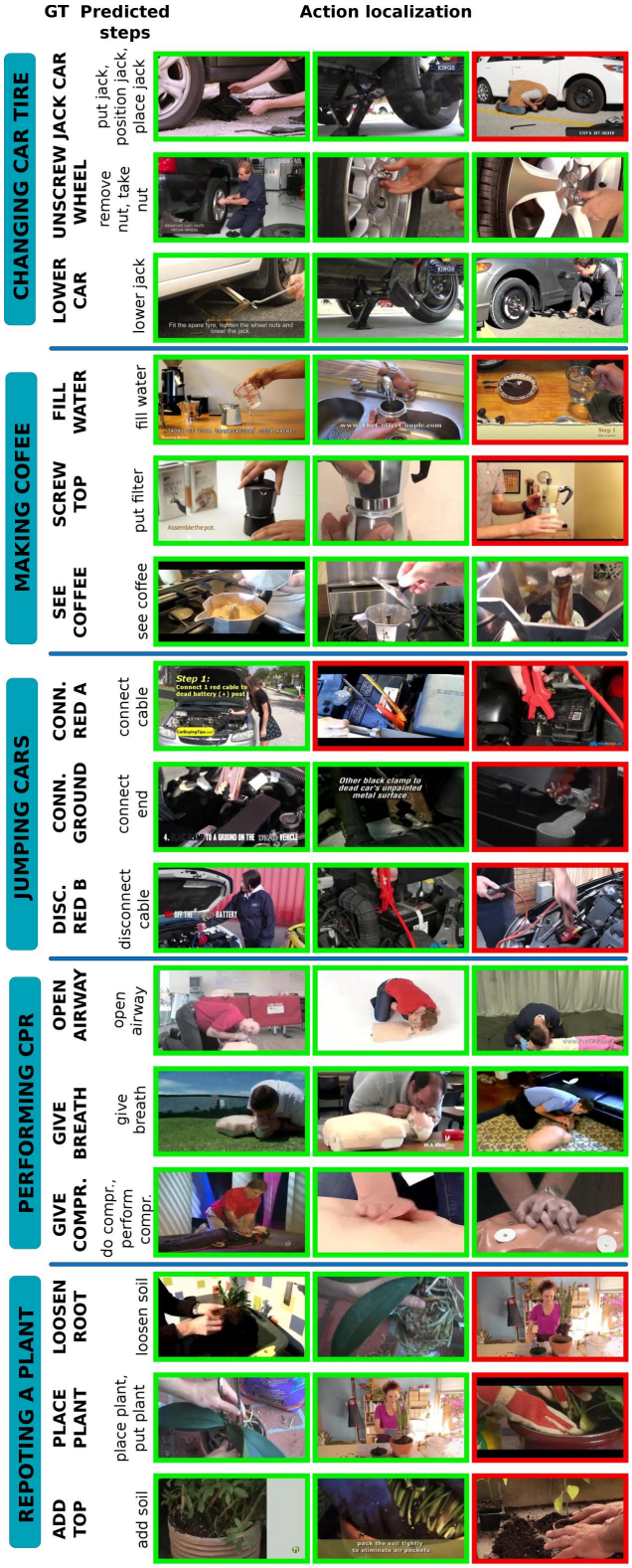


Fig. 4.6.: Examples of three recovered instruction steps for each of the five tasks in our dataset. For each step, we first show clustered direct object relations, followed by representative example frames localizing the step in the videos. Correct localizations are shown in green. Some steps are incorrectly localized in some videos (red), but often look visually very similar.

that require a known number of steps K, we report results for a range of K values. Note that in our case the actual number of automatically recovered steps can be (and often is) smaller than K. For *Change tire* and *Perform CPR*, our method consistently outperforms all baselines for all values of K demonstrating the benefits of our approach. For *Repot*, our method is comparable to text-based baselines, underlying the importance of the text signal for this problem. For *Jump car*, our method delivers the best result (for $K = 15$) but struggles for lower values of K, which we found was due to visually similar repeating steps (e.g. start car A and start car B) which are mixed-up for lower values of K. For the *Make coffee* task, the video only baseline is comparable to our method, which by inspecting the output could be attributed to large variability of narrations for this task. We also report results using the word2vec text similarity metric (see 'Our method (w2v)' in Figure 4.5) instead of the WordNet metric used in the first stage of our algorithm. We observe that both metrics result in a similar performance.

Qualitative results of the recovered steps are illustrated in Figure 4.6.

4.5.5 Parameter analysis

In this section, we present a parameter sensitivity analysis divided into two main experiments: (i) in Section 4.5.5 we evaluate the sensitivity of our method to the choice of hyperparameter λ, which controls the strength of the regularization, and hyperparameters (Δ_b, Δ_a), which control the temporal caption-video alignment; (ii) in Section 4.5.5 we analyze the two main factors that influence the performance of the method, i.e. the number of constraints that are extracted from the textual narration and their temporal localization.

Hyperparameter analysis

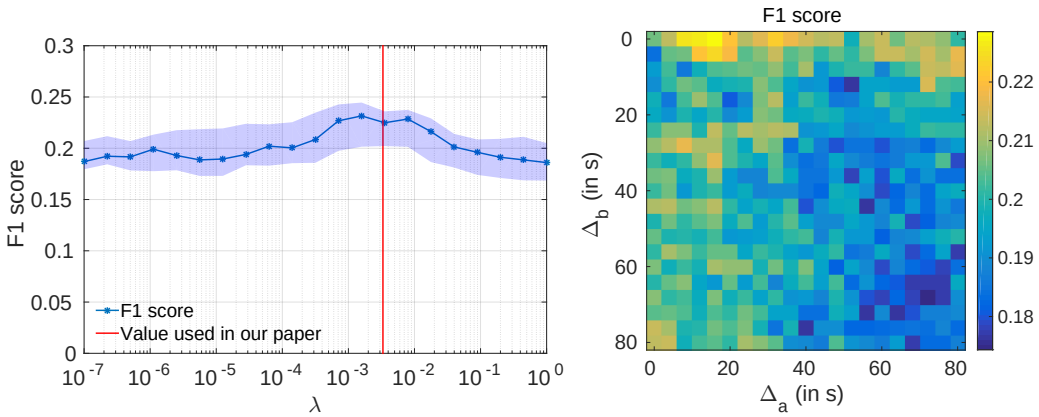
Our method has two main hyperparameters. The first one, denoted λ, controls the amount of regularization of the estimated classifier parameters W in the discriminative loss (4.8). We use a single fixed setting of λ as our method is unsupervised and has no access to ground truth annotations for cross-validation at training time. Hence in all our experiments we set $\lambda = 1/(NK_{\text{pred}})$ which is equal to the inverse of the number of predicted

latent variables. This choice is motivated by the form of the ridge regression generalization bound discussed in [Hsu *et al.*, 2014]. The second set of hyperparameters (Δ_b, Δ_a) adjusts the temporal alignment of transcribed narrations to the video. In particular, it allows us to model the fact that people usually perform the action after having talked about it. In our work we set the value of Δ_b (the 'before' delay) to 0 seconds and of Δ_a (the 'after' delay) to 12 seconds. In the following we quantify the effect of these hyperparameters on the final performance.

In Figure 4.7a, we report the F1 score averaged over the five tasks as a function of λ. As explained in Section 4.5.4, the error bars (shown by the shaded area) represent the F1 score variation during optimization. For this experiment, (Δ_b, Δ_a) is set to $(0 \text{ s}, 12 \text{ s})$ as in Section 4.5.4. We observe that, (i) a good choice of λ can result in up to 5% improvement on average across all five tasks, and (ii) the right range of values is located around $\lambda = 1/(NK_{\text{pred}})$ (see red line in Figure 4.7a). We report here the average performance but we have observed that the best setting of λ varies across different tasks. For example for the *performing cardiopulmonary resuscitation (CPR)* task, the good range of values for λ was a bit higher ($\approx 10^{-1}$) than for the other four tasks ($\approx 10^{-4} - 10^{-3}$). If a small annotated set is available, λ can be tuned for each task using cross-validation. If not, then $\lambda = 1/(NK_{\text{pred}})$ presents a reasonable choice.

In Figure 4.7b, we report the average F1 score over the five tasks as a function of (Δ_b, Δ_a). For this experiment, we set $\lambda = 1/(NK_{\text{pred}})$. For very large $(\Delta_b, \Delta_a) > (60, 60)$, the method is almost equivalent to the video only baseline (with a drop in performance of more than 5%) as the transcribed narration provides only very weak constraints on the temporal localization of the different steps. The good range for these hyperparameters is asymmetric, centered around $\Delta_a \approx 15$ s and $\Delta_b \approx 0$ s. This confirms the hypothesis that usually people first orally introduce a step and only then perform the action.

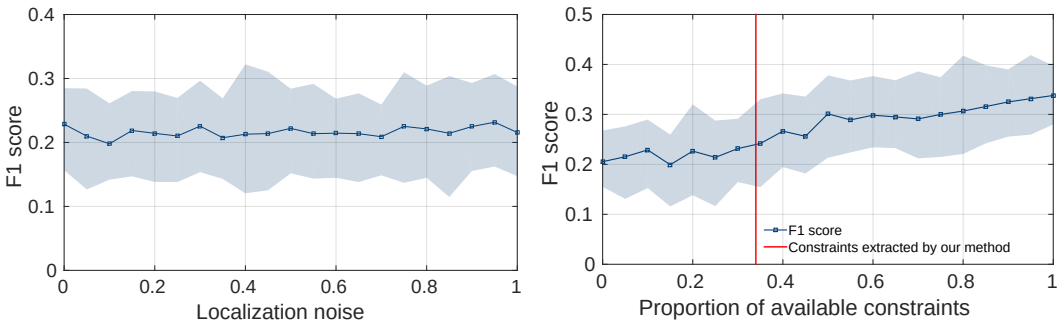
Analysis of constraints from transcribed narrations



(a) Regularization strength λ **(b)** Subtitle delays (Δ_b, Δ_a)

Fig. 4.7.: Sensitivity of the performance (measured by the F1 score) to the different hyperparameters.

The overall performance of our method relies on the amount and quality of constraints extracted from transcribed narrations. In this section we quantify this dependence experimentally. We perform two experiments.



(a) Influence of precise time localization of each constraint. **(b)** Influence of proportion of extracted constraints.

Fig. 4.8.: Analysis of constraints extracted from transcribed narrations.

First, we assess the importance of a precise time localization of each constraint. This type of mistake often occurs as the narration is usually not precisely aligned with the video, i.e. the speaker often first talks about the action before performing it. Figure 4.8a shows how the overall performance (measured by average F1 score on all tasks) varies with respect to the localization noise of extracted constraints. The set of constraints is fixed to those discovered automatically by our method. When the localization noise is 0, the temporal extent of each constraint obtained from the narration is corrected to the manually obtained temporal extent of the action performed in the video. When the localization noise is 1 constraints correspond to those extracted automatically by our method. In between values are obtained by

interpolating the temporal extent of each constraint between these two extreme values. The average performance is obtained by 5-fold cross validation. The error bars correspond to minimum and maximum performance across the five folds. Interestingly, the results do not show a drop in performance with increased localization noise in the constraints. This suggests that our method is tolerant to some amount of localization noise and the constraints extracted by our method are localized reasonably well.

Next, in Figure 4.8b we study the influence of the amount of extracted constraints. In particular, we vary the proportion of extracted constraints from 0 (equivalent to the video only setting) to 1 (extracted constraint for each step performed in video). Note that we only use constraints that respect the ground truth global ordering of steps. To remove the effect of temporal localization, we use the manually corrected temporal extent for each constraint corresponding to the zero localization noise in the first experiment above. The graph shows average performance and error bars (shaded region) obtained by 5-fold cross validation as described in the first experiment above. Our automatic method using multiple sequence alignment extracts from the transcribed narration about 35% of constraints (red vertical bar). The results suggest that increasing the number of extracted constraints can significantly improve the overall performance. Hence, designing better language models that could discover more constraints is an important direction for future research.

4.5.6 Joint clustering of video and text

Recall that the proposed approach is formulated as two clustering tasks, one in text and one in video, applied in sequence and linked by joint constraints on the two modalities. To further investigate the complementary nature of the two signals, we experiment here with performing another iteration of our algorithm with the aim of transferring information from the visual to the textual domain. We proceed as follows. Once we obtain our temporal predictions in videos (latent variable Z), we use the inferred Z to adjust the values in the cost alignment matrix C_o in multiple sequence alignment of text sequences. More precisely, we match expressions if their similarity is high (but with a slightly lower threshold than in the first iteration) *and* if they were mentioned at least two times within the predicted time interval of the

same step in different videos. We run the multiple sequence alignment with the new cost matrix C_o and use this solution to perform another iteration of the video alignment procedure. This additional iteration results in an average improvement of about 1% in the F1-score across all tasks.

4.6 Conclusion and future work

We have described a method to automatically discover the main steps of a task from a set of narrated instructional videos in an unsupervised manner. The proposed approach has been tested on a new annotated dataset of challenging real-world instructional videos containing complex person-object interactions in a variety of indoor and outdoor scenes. Our work opens up the possibility for large scale learning from instructional videos on the Internet. Our model currently assumes the existence of a common script with a fixed ordering of the main steps. While this assumption is often true, e.g. one cannot remove the wheel before jacking up the car, or make coffee before filling the water, some tasks can be performed while swapping (or even leaving out) some of the steps. Recovering more complex temporal structures is an interesting direction for future work.

5 Joint Discovery of Object States and Manipulation Actions

In the previous chapter, we have described a method to automatically discover the main action steps to achieve a complex task given a set of narrated instructional videos depicting that task. Even if person-object interactions play an important role in instructional videos, we only model these interactions implicitly by opting for generic motion and image pre-trained features.

In this chapter, we follow a different approach by explicitly modeling the link between actions and objects. More specifically, we focus on actions that are very common when dealing with instructional videos: object manipulations aiming to modify the object state. Examples of common state changes include full/empty bottle, open/closed door, and attached/detached car wheel. We seek to automatically discover the states of objects and the associated manipulation actions. Given a set of videos for a particular task, we propose a joint model that learns to identify object states and to localize state-modifying actions. Our model is formulated as a discriminative clustering cost with constraints. We assume a consistent temporal order for the changes in object states and manipulation actions, and introduce new optimization techniques to learn model parameters without additional supervision. We demonstrate successful discovery of seven manipulation actions and corresponding object states on a new dataset of videos depicting real-life object manipulations. We show that our joint formulation results in an improvement of object state discovery by action recognition and vice versa.

5.1 Introduction

Many of our activities involve changes in object states. We need to open a book to read it, to cut bread before eating it and to lighten candles before taking out a birthday cake. Transitions of object states are often coupled with

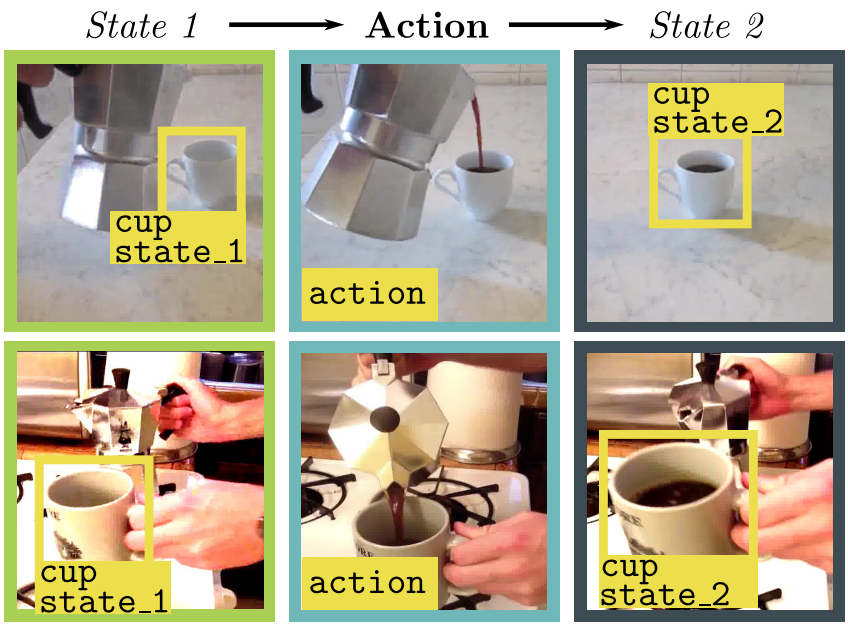


Fig. 5.1.: We automatically discover object states such as empty/full coffee cup along with their corresponding manipulation actions by observing people interacting with the objects.

particular manipulation actions (open, cut, lighten). Moreover, the success of an action is often signified by reaching the desired state of an object (whipped cream, ironed shirt) and avoiding other states (burned shirt). Recognizing object states and manipulation actions is, hence, expected to become a key component of future systems such as wearable automatic assistants or home robots helping people in their daily tasks.

Human visual system can easily distinguish different states of objects, such as open/closed bottle or full/empty coffee cup [Brady *et al.*, 2006]. Automatic recognition of object states and state changes, however, presents challenges as it requires distinguishing subtle changes in object appearance such as the presence of a cap on the bottle or screws on the car tire Despite much work on object recognition and localization, recognition of object states has received only limited attention in computer vision [Isola *et al.*, 2015].

One solution to recognizing object states would be to manually annotate states for different objects, and treat the problem as a supervised fine-grained object classification task Duan *et al.*, 2012; Farhadi *et al.*, 2009. This approach, however, presents two problems. First, we would have to decide *a priori* on the set of state labels for each object, which can be ambiguous and not

suitable for future tasks. Second, for each label we would need to collect a large number of examples, which can be very costly.

In this chapter, we propose to *discover* object states directly from videos with object manipulations. As state changes are often caused by specific actions, we attempt to jointly discover object states and corresponding manipulations. In our setup we assume that two distinct object states are temporally separated by a manipulation action. For example, the empty and full states of a coffee cup are separated by the “pouring coffee" action, as shown in Figure 5.1. Equipped with this constraint, we develop a clustering approach that jointly (i) groups object states with similar appearance and consistent temporal locations with respect to the action and (ii) finds similar manipulation actions separating those object states in the input videos. Our approach exploits the complementarity of both subproblems and finds a joint solution for states and actions. We formulate our problem by adopting a discriminative clustering loss Bach and Harchaoui, 2007 and a joint consistency cost between states and actions. We introduce an effective optimization solution in order to handle the resulting non-convex loss function and the set of spatial-temporal constraints. To evaluate our method, we collect a new video dataset depicting real-life object manipulation actions in realistic videos. Given this dataset for training, our method demonstrates successful discovery of object states and manipulation actions. We also demonstrate that our joint formulation gives an improvement of object state discovery by action recognition and vice versa.

5.2 Related work

Below we review related work on person-object interaction, recognizing object states, action recognition and discriminative clustering that we employ in our model.

Person-object interactions. Many daily activities involve person-object interactions. Modeling co-occurrences of objects and actions have shown benefits for recognizing actions in Delaitre *et al.*, 2011; Gupta *et al.*, 2009a; Kjellström *et al.*, 2011; Pirsiavash and Ramanan, 2012; Yao *et al.*, 2011. Recent work has also focused on building realistic datasets with people manipulating

objects, *e.g.* in instructional videos Alayrac *et al.*, 2016a; Malmaud *et al.*, 2015; Sener *et al.*, 2015 or while performing daily activities Sigurdsson *et al.*, 2016. We build on this work but focus on joint modeling and recognition of actions and *object states*.

Object states. Prior work has addressed recognition of object attributes Farhadi *et al.*, 2009; Parikh and Grauman, 2011; Patterson *et al.*, 2014, which can be seen as different object states in some cases. Differently from our approach, these works typically focus on classifying still images, do not consider human actions and assume an *a priori* known list of possible attributes. Closer to our setting, Isola *et al.*, [2015] discover object states and transformations between them by analyzing large collections of still images downloaded from the Internet. In contrast, our method does not require annotations of object states. Instead, we use the dynamics of consistent manipulations to discover object states in the video with minimal supervision. In [Damen *et al.*, 2014], the authors use consistent manipulations to discover task relevant objects. However, they do not consider object states, rely mostly on first person cues (such as gaze) and take advantage of the fact that videos are taken in a single controlled environment.

Action recognition. Most of the prior work on action recognition has focused on designing features to describe time intervals of a video using motion and appearance [Laptev *et al.*, 2008b; Simonyan and Zisserman, 2014; Tran *et al.*, 2015; Wang and Schmid, 2013]. This is effective for actions such as *dancing* or *jumping*, however, many of our daily activities are best distinguishable by their effect on the environment. For example, *opening door* and *closing door* can look very similar using only motion and appearance descriptors but their outcome is completely different. This observation has been used to design action models in [Fathi and Rehg, 2013; Fernando *et al.*, 2015; Wang *et al.*, 2016]. In [Wang *et al.*, 2016], for example, the authors propose to learn an embedding in which a given action acts as a transformation of features of the video. In our work we localize objects and recognize changes of their states using manipulation actions as a supervisory signal. Related to ours is also the work of Fathi and Rehg, [2013] who represent actions in egocentric videos by changes of appearance of objects (also called object states), however, their method requires manually annotated precise temporal localization of actions in training videos. In contrast, we focus on (non-egocentric) Internet

videos depicting real-life object manipulations where actions are performed by different people in a variety of challenging indoor/outdoor environments. In addition, our model jointly learns to recognize both actions and object states with only minimal supervision.

Discriminative clustering. Our model builds on unsupervised discriminative clustering methods [Bach and Harchaoui, 2007; Singh *et al.*, 2012; Xu *et al.*, 2004] that group data samples according to a simultaneously learned classifier. Such methods can incorporate (weak) supervision that helps to steer the clustering towards a preferred solution [Bojanowski *et al.*, 2013; Doersch *et al.*, 2012; Huang *et al.*, 2016; Jain *et al.*, 2013]. In particular, we build on the discriminative clustering approach of [Bach and Harchaoui, 2007] that has been shown to perform well in a variety of computer vision problems [Bojanowski *et al.*, 2013]. It leads to a quadratic optimization problem where different forms of supervision can be incorporated in the form of (typically) linear constraints. Building on this formalism, we develop a model that jointly finds object states and temporal locations of actions in the video. Part of our object state model is related to [Joulin *et al.*, 2014a], while our action model is related to [Bojanowski *et al.*, 2015]. However, we introduce new spatial-temporal constraints, as well as new effective optimization techniques. Our work is also related to [Ramanathan *et al.*, 2014]. In this work, the authors also propose a discriminative formulation for solving a joint problem (assignement of tracks of people in videos to actors names along with assignment of mentions in the corresponding script to actors names). In contrast, the nature of our two assignments (actions and object states) is different, which requires us to carefully design a novel assymetric joint cost function linking object states and actions.

Contributions. The contributions of this work are three-fold. First, we develop a new discriminative clustering model that jointly discovers object states and temporally localizes associated manipulation actions in video. Second, we introduce an effective optimization algorithm to handle the resulting non-convex constrained optimization problem. Finally, we experimentally demonstrate that our model discovers key object states and manipulation actions from input videos with minimal supervision.

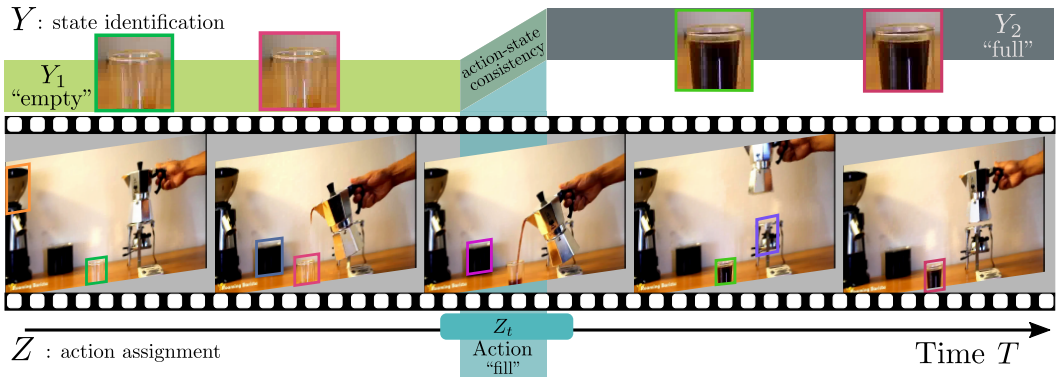


Fig. 5.2.: Given a set of clips that depict a manipulated object, we wish to automatically discover the main states that the object can take along with localizing the associated manipulation action. In this example, we show one video of someone filling a coffee cup. The video starts with an empty cup (*state 1*), which is filled with coffee (**action**) to become full (*state 2*). Given imperfect object detectors, we wish to assign to the valid object candidates either the initial state or the final state (encoded in Y). We also want to localize the manipulating action in time (encoded in Z) while maintaining a joint action-state consistency.

5.3 Modeling manipulated objects

We are given a set of N clips that contain a common **manipulation** of the same *object* (such as "**open** an *oyster*"). We also assume that we are given an *a priori* model of the corresponding object in the form of a pre-trained object detector [Girshick, 2015]. Given these inputs, our goal is twofold: (i) localize the temporal extent of the action and (ii) spatially/temporally localize the manipulated object and identify its states over time. This is achieved by jointly clustering the appearances of an object (such as an "oyster") appearing in all clips into two classes, corresponding to the two different states (such as "closed" and "open"), while at the same time temporally localizing a consistent "opening" action that separates the two states consistently in all clips. More formally, we formulate the problem as a minimization of a joint cost function that ties together the action prediction in time, encoded in the assignment variable Z, with the object state discovery in space and time, defined by the assignment variable Y:

$$\underset{\substack{Y\in\{0,1\}^{M\times 2}\\ Z\in\{0,1\}^{T}}}{\text{minimize}} \quad f(Z) + g(Y) + d(Z,Y) \tag{5.1}$$

$$\text{s.t.} \quad \underbrace{Z \in \mathcal{Z}}_{\substack{\text{saliency of action}\\ \textbf{Action localization}}} \quad \text{and} \quad \underbrace{Y \in \mathcal{Y}}_{\substack{\text{ordering + non overlap}\\ \textbf{Object state labeling}}}$$

where $f(Z)$ is a discriminative clustering cost to temporally localize the action in each clip, $g(Y)$ is a discriminative clustering cost to identify and localize the different object states and $d(Z, Y)$ is a joint cost that relates object states and actions together. T denotes the total length of all video clips and M denotes the total number of tracked object candidate boxes (tracklets). In addition, we impose constraints $\mathcal{Y}$ and $\mathcal{Z}$ that encode additional structure of the problem: we localize the action with its most salient time interval per clip ("saliency"); we assume that the ordering of object states is consistent in all clips ("ordering") and that only one object is manipulated at a time ("non overlap").

In the following, we proceed with describing different parts of the model (5.1). In Sec. 5.3.1 we describe the cost function for the discovery of object states. In Sec. 5.3.2 we detail our model for action localization. Finally, in Sec. 5.3.3 we describe and motivate the joint cost d.

5.3.1 Discovering object states

The goal here is to both (i) spatially localize the manipulated object and (ii) temporally identify its individual states. To address the first goal, we employ pre-trained object detectors. To address the second goal, we formulate the discovery of object states as a discriminative clustering task with constraints. We obtain candidate object detections using standard object detectors pre-trained on large scale existing datasets such as ImageNet [Deng *et al.*, 2009]. We assume that each clip n is accompanied with a set of M_n tracklets[1] of the object of interest.

We formalize the task of localizing the states of objects as a discriminative clustering problem where the goal is to find an assignment matrix $Y_n \in \{0, 1\}^{M_n \times 2}$, where $(Y_n)_{mk} = 1$ indicates that the m-th tracklet represents the object in state k. We also allow a complete row of Y_n to be zero to encode that no state was assigned to the corresponding tracklet. This is to model the possibility of false positive detections of an object, or that another object

[1] In this work, we use short tracks of objects (less than one second) that we call tracklet. We want to avoid long tracks that continue across a state change of objects. By using the finer granularity of tracklets, our model has the ability to correct for detection mistakes within a track as well as identify more precisely the state change.

of the same class appears in the video, but is not manipulated and thus is not undergoing any state change. In detail, we minimize the following discriminative clustering cost Bach and Harchaoui, 2007:[2]

$$g(Y) = \min_{W_s \in \mathbb{R}^{d_s \times 2}} \underbrace{\frac{1}{2M}\|Y - X_s W_s\|_F^2}_{\text{Discriminative loss on data}} + \underbrace{\frac{\mu}{2}\|W_s\|_F^2}_{\text{Regularizer}} \tag{5.2}$$

where W_s is the object state classifier that we seek to learn, μ is a regularization parameter and X_s is a $M \times d_s$ matrix of features, where each row is a d_s-dimensional (state) feature vector storing features for one particular tracklet. The minimization in W_s actually leads to a convex quadratic cost function in Y (see [Bach and Harchaoui, 2007]). The first term in (5.2) is the discriminative loss on the data that measures how easily the input data X_s is classified by the linear classifier W_s when the object state assignment is given by matrix Y. In other words, we wish to find a labeling Y for given object tracklets into two states (or no state) so that their appearance features X are easily classified by a linear classifier. To steer the cost towards the right solution, we employ the following constraints (encoded by $Y \in \mathcal{Y}$ in (5.1)).

Only one object is manipulated at a time : non overlap constraint. As it is common in instructional videos, we assume that only one object can be manipulated at a given time. However, in practice, it is common to have multiple (spatially diverse) tracklets that occur at the same time, for example, due to a false positive detection in the same frame. To overcome this issue, we impose that at most one tracklet can be labeled as belonging to *state 1* or *state 2* at any given time. We refer to this constraint as "*non overlap*" in problem (5.1).

***state 1* $\rightarrow$ *Action* $\rightarrow$ *state 2*: ordering constraints.** We assume that the manipulating action transforms the object from an initial state to a final state and that both states are present in each video. This naturally introduces two constraints. The first one is the ordering constraints on the labeling Y_n, *i.e.*the *state 1* should occur before *state 2* in each video. The second constraint imposes that we have *at least one* tracklet labeled as *state 1* and at least one tracklet labeled as *state 2*. We call this last constraint the "**at least one**" constraint in contrast to forcing "exactly one" ordered prediction as previously proposed in a discriminative clustering approach on video for

[2]We concatenate all the variables Y_n into one $M \times 2$ matrix Y.

action localization Bojanowski *et al.*, 2015. This new type of constraint brings additional optimization challenges that we address in Section 5.4.2.

5.3.2 Action localization

Our action model is equivalent to the one of [Bojanowski *et al.*, 2015] applied to only *one action*. More precisely, the goal is to find an assignment matrix $Z_n \in \{0,1\}^{T_n}$ for each clip n, where $Z_{nt} = 1$ encodes that the t-th time interval of video is assigned to an action and $Z_{nt} = 0$ encodes that no action is detected in interval t. The cost that we minimize for this problem is similar to the object states cost:

$$f(Z) = \min_{W_v \in \mathbb{R}^{d_v}} \underbrace{\tfrac{1}{2T}\|Z - X_v W_v\|_F^2}_{\text{Discriminative loss on data}} + \underbrace{\tfrac{\lambda}{2}\|W_v\|_F^2}_{\text{Regularizer}}, \tag{5.3}$$

where W_v is the action classifier, λ is a regularization parameter and X_v is a matrix of visual features. We constrain our model to predict *exactly* one time interval for an action per clip, an approach for actions that was shown to be beneficial in a weakly supervised setting [Bojanowski *et al.*, 2015] (referred to as "*action saliency*" constraint). As will be shown in experiments, this model *alone* is incomplete because the clips in our dataset can contain other actions that do not manipulate the object of interest. Our central contribution is to propose a *joint formulation* that links this action model with the object state prediction model, thereby resolving the ambiguity of actions. We detail the joint model next.

5.3.3 Linking actions and object states

Actions in our model are directly related to changes in object states. We therefore want to enforce consistency between the two problems. To do so, we design a novel joint cost function that operates on the action video labeling Z_n and the state tracklet assignment Y_n for each clip. We want to impose a constraint that the action occurs in between the presence of the two different object states. In other words, we want to penalize the fact that state 1 is detected *after* the action happens, or the fact that state 2 is triggered *before* the action occurs.

Joint cost definition. We propose the following joint symmetric cost function for each clip:

$$d(Z_n, Y_n) = \sum_{y \in \mathcal{S}_1(Y_n)} [t_y - t_{Z_n}]_+ + \sum_{y \in \mathcal{S}_2(Y_n)} [t_{Z_n} - t_y]_+, \tag{5.4}$$

where t_{Z_n} and t_y are the times when the action Z_n and the tracklet y occur in a clip n, respectively. $\mathcal{S}_1(Y_n)$ and $\mathcal{S}_2(Y_n)$ are the tracklets in the n-th clip that have been assigned to state 1 and state 2, respectively. Finally $[x]_+$ is the positive part of x. In other words, the function penalizes the inconsistent assignment of objects states Y_n by the amount of time that separates the incorrectly assigned tracklet and the manipulation action in the clip. The overall joint cost is the sum over all clips weighted by a scaling hyperparameter $\nu > 0$:

$$d(Z, Y) = \nu \frac{1}{T} \sum_{n=1}^{N} d(Z_n, Y_n). \tag{5.5}$$

5.4 Optimization

Optimizing problem (5.1) poses several challenges that need to be addressed. First, we propose a relaxation of the integer constraints and the distortion function (Section 5.4.1). Second, we optimize this relaxation using Frank-Wolfe with a new dynamic program able to handle our tracklet constraints (Section 5.4.2). Finally, we introduce a new rounding technique to obtain an integer candidate solution to our problem (Section 5.4.4).

5.4.1 Relaxation

Problem (5.1) is NP-hard in general [Loiola *et al.*, 2007] due to its specific integer constraints. Inspired by the approach of Bojanowski *et al.*, 2014 that was successful to approximate combinatorial optimization problems, we propose to use the tightest convex relaxation of the feasible subset of binary matrices by taking its convex hull. As our variables now can take values

in $[0, 1]$, we also have to propose a consistent extension for the different cost functions to handle fractional values as input. For the cost functions f and g, we can directly take their expression on the relaxed set as they are already expressed as (convex) quadratic functions. Similarly, for the joint cost function d in (5.4), we use its natural bilinear relaxation:

$$d(Z_n, Y_n) = \sum_{i=1}^{M_n} \sum_{t=1}^{T_n} \Big((Y_n)_{i1} Z_{nt} [t_{ni} - t]_+ + (Y_n)_{i2} Z_{nt} [t - t_{ni}]_+ \Big), \tag{5.6}$$

where t_{ni} denotes the video time of tracklet i in clip n. This relaxation is equal to the function (5.4) on the integer points. However, it is not jointly convex in Y and Z, thus we have to design an appropriate optimization technique to obtain good (relaxed) candidate solutions, as described next.

5.4.2 Joint optimization using Frank-Wolfe

When dealing with a constrained optimization problem for which it is easy to solve linear programs but difficult to project on the feasible set, the Frank-Wolfe algorithm is an excellent choice [Jaggi, 2013c; Lacoste-Julien and Jaggi, 2015a]. It is exactly the case for our relaxed problem, where the linear program over the convex hull of feasible integer matrices can be solved efficiently via dynamic programming as shown in Section 5.4.3. Moreover, Lacoste-Julien, [2016] recently showed that the Frank-Wolfe algorithm with line-search converges to a stationary point for non-convex objectives at a rate of $O(1/\sqrt{k})$. We thus use this algorithm for the joint optimization of (5.1). As the objective is quadratic, we can perform exact line-search analytically, which speeds up convergence in practice. Finally, in order to get a good initialization for both variables Z and Y, we first optimize separately $f(Z)$ and $g(Y)$ (without the non-convex $d(Z, Y)$), which are both convex functions.

5.4.3 Linear oracle for object state constraints

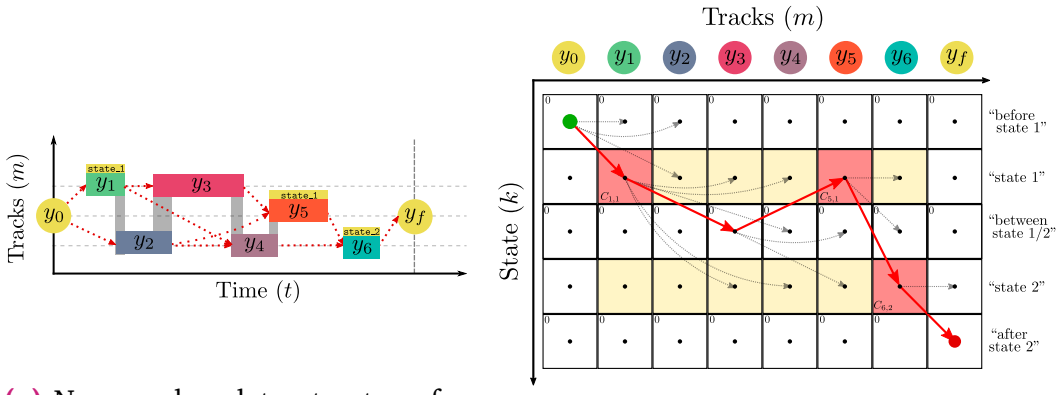
In order to apply the Frank-Wolfe algorithm, we need to solve a linear program (LP) over our set of constraints. Previous work has explored "*exact one*" ordering constraints for time localization problems [Bojanowski *et al.*,

2014]. Differently here, we have to deal with three main components in the constraints. First, we assume that only one object is manipulated at a given time. Thus *at most one* tracklet can be assigned to a state at a given time. This constraint is referred to as the **non-overlap constraint**, to remind that there is no temporal overlap between two object states. Second, we have the **ordering constraint** that imposes that *state 1* always happens *before state 2*. The last constraint imposes that we have **at least one** tracklet labeled as *state 1* and *at least one* tracklet labeled as *state 2*. We need to be able to minimize linear functions over this set of constraints in order to use the Frank-Wolfe algorithm. More precisely, as the constraints decompose over the different clips, we can solve independently for each clip n the following linear problem:

$$\begin{aligned} \underset{Y_n \in \{0,1\}^{M_n \times 2}}{\text{minimize}} \quad & \text{Tr}(C_n^\top Y_n) \\ \text{s.t.} \quad & \underbrace{Y_n \in \mathcal{Y}_n}_{\substack{\text{non-overlap + ordering} \\ \text{+ at least one}}} \end{aligned} , \tag{5.7}$$

where $C_n \in \mathbb{R}^{M_n \times 2}$ is a cost matrix that typically comes from the computation of the gradient of the cost function at the current iterate. In order to solve this problem, we use a dynamic program approach that we explain next. Recall that we are given M_n tracklets $(y_i)_{i=1}^{M_n}$ and our goal is to output the Y_n matrix that assigns to each of these tracklets either state 1, state 2 or no state at all while respecting the constraints. The whole method is illustrated in Figure 5.3 with a toy example.

Non-overlap data structure for object tracklets. We first pre-process the tracklets to build an auxiliary data-structure that is used to enforce the non-overlap constraint between the tracklets, as illustrated in Figure 5.3a. First, we sort and index each tracklet by their beginning time, and add two fictitious tracklets: y_0 as the starting tracklet and y_f as the ending tracklet. These two tracklets are used to start and terminate the dynamic program. If all the tracklets were sequentially ordered without any overlap in time, then we could simply make a decision for each of them sequentially as was done in previous work on action localization for example (one decision per time step) [Bojanowski *et al.*, 2014]. To enforce the non-overlap constraint, we force the decision process to choose *only one* possible successor among the group of overlapping valid immediate successors of a tracklet. For each



(a) Non-overlap data structure for tracklets

(b) Cost matrix $\tilde{C}_n$ for the dynamic program

Fig. 5.3.: In **(a)**, we provide an illustration of a possible situation for the tracklets. y_0 and y_f are two fictitious tracklets that encode the beginning and end of the video. Each tracklet is indexed based on its beginning time. The time overlap between tracklets is shown by the grey color. We specify for each tracklet its possible successors by the dotted red arrows (see main text). Finally an admissible labeling is illustrated by yellow tags where y_1 and y_5 have both been assigned to state 1 and y_6 to state 2. In **(b)**, we give an illustration of our approach to solve (5.7) with a dynamic program. We display the modified cost matrix $\tilde{C}_n$ (see main text). A valid path has to go from the green dot (y_0) to the red dot (y_f). The light yellow entries show parts of the C_n matrix that are inserted into $\tilde{C}_n$, whereas white entries encode the rows of 0s that are inserted to impose the **at least one** ordering constraint. The red arrows specify an example optimal path inside the matrix. The red entries display the tracklets that have been assigned to state 1 (y_1 and y_5) or state 2 (y_6) (equivalent to putting ones in the appropriate corresponding entries in Y_n). Finally, the grey arrows display the possible valid transitions that can be made for the entries *along the red path*, for clarity. We see for example that from $(2, y_1)$, there are 6 possible transitions: two column choices from the two red arrows from y_1 in (a) encoding the non-overlap constraint; and three row choices encoding the valid transition from "state 1" (corresponding to the choice "state 1", 0 or "state 2" for the next tracklet) encoding the "at least one" ordering constraint.

tracklet y_i, we thus define its (smallest) set of "*valid successors*" as the earliest tracklet y_j[3] after y_i that is also non-overlapping with y_i, as well as any other tracklet y_l for $l > j$ that is overlapping with y_j (thus giving the earliest valid group of overlapping tracklets). The valid successors are illustrated by red dotted arrows in Figure 5.3a. For example, the valid successors of y_1 are y_3 (the earliest one that is non-overlapping) as well as y_4 (which overlaps with y_3 thus forming an overlapping group). Skipping a tracklet in this decision process means that we assign it to zero (which trivially always

[3]Earliest means the smallest j.

satisfies the non-overlapping constraint); whereas once we choose a tracklet to potentially assign it to state 1 or 2, we cannot visit any overlapping tracklet by construction of the valid successors, thus maintaining the non-overlap constraint.

Dynamic program. The dynamic programming approach is used when we can solve a large problem by solving a sequence of inclusive subproblems that are linked by a simple recursive formula and that use overlapping solutions (which can be stored in a table for efficiency). In terms of implementation, Bojanowski *et al.*, [2014] encoded their dynamic program as finding an optimal path inside a cost matrix. This approach is particularly suited when the update cost rule depends *only* on the arrival entry in the cost matrix as opposed to be transition dependent. As we will show below, we can encode the solution to our problem in a way that satisfies this property. We therefore use the framework of [Bojanowski *et al.*, 2014] by casting our problem as a search for an optimal path inside a cost matrix $\tilde{C}_n$ illustrated in Figure 5.3b, and where the valid transitions encode the possible constraints.

One main difference with [Bojanowski *et al.*, 2014] is that we have to deal with the challenging **at least one** constraint in the context of ordered labels. To do so, we can filter further the set of valid decisions by using "memory states" that encode in which of the following three situations we are: **(i)** that state 1 has not yet been visited, **(ii)** that state 1 has already been visited, but state 2 has not yet been visited (and thus that we can either come back to state 1 or go to state 2) and **(iii)** that both states have been visited. These memory states can be encoded by interleaving complete rows of 0s in between columns of C_n stored as rows, to obtain the $5 \times M_n$ matrix $\tilde{C}_n$. These new rows encode the three different memory states previously described when making a prediction of 0 for a specific tracklet, and we enforce the correct memory semantic by only allowing a path to move to the same row or the row immediately below, except for state 1 which can also move directly to state 2 (two rows below), and the middle "between state 1/2" row, where one can go up one row additionally to state 1. Finally, the valid transitions between columns (tracklets) are given by the *valid successors* data structure as given in Figure 5.3a to encode the non-overlap constraints. Combining these two constraints (at least one ordering and non-overlap), we illustrate with grey arrows in Figure 5.3b the possible transitions from the states along

the path in red. To describe the dynamic program recursion below, we need to go the opposite direction from the successors, and thus we say that y_j is a *predecessor* of y_i if and only if y_i is a successor of y_j.

To perform the dynamic program, we maintain a matrix D_n of the same size as $\tilde{C}_n$ where $D_n(k,i)$ contains the minimal valid path cost of going from $(1, y_0)$ to (k, y_i) inside the cost matrix $\tilde{C}_n$. To define the cost update recursion to compute $D_n(k,i)$, let $P(k,i)$ be the set of tuples (l,j) for which it is possible to go from (l,j) to (k, y_i) according to the rules described above. The update rule is then as follows:

$$D_n(k,i) = \min_{(l,j)\in P(k,i)} D_n(l,j) + \tilde{C}_n(k, y_i). \tag{5.8}$$

As we see here, the added cost depends *only* on the arrival entry $\tilde{C}_n(k, y_i)$. We can therefore use the approach of [Bojanowski *et al.*, 2014] and only consider entry costs rather than edge costs. Thanks to our indexing property (tracklets are sorted by the beginning time), we can update the dynamic program matrix by filling each column of D_n one after the other. Once this update is finished, we back-track to get the best path by starting from the ending track (predecessors of y_f) at the last row (to be sure that both states have been visited) that has the lowest score in the D_n matrix. The total complexity of this algorithm is of order $\mathcal{O}(M_n)$.

In addition, we show in section 5.5.2 that these new constraints when compared to previous work [Bojanowski *et al.*, 2014], namely the **at least one** constraint and the **non overlap** constraint, are *key* for the success of the method, justifying the efforts invested in the design of this dynamic program.

5.4.4 Joint rounding method

Once we obtain a candidate solution of the relaxed problem, we have to round it to an integer solution in order to make predictions. Previous works (see Chapter 4 or [Bojanowski *et al.*, 2015]) have observed that using the learned W^* classifier for rounding gave better results than other possible alternatives. We extend this approach to our joint setup by proposing the following new rounding procedure. We optimize problem (5.1) but fix the

values of W in the discriminative clustering costs. Specifically, we minimize the following cost function over the integer points $Z \in \mathcal{Z}$ and $Y \in \mathcal{Y}$:

$$\frac{1}{2T}\|Z - X_v W_v^*\|_F^2 + \frac{1}{2M}\|Y - X_s W_s^*\|_F^2 + d(Z, Y), \tag{5.9}$$

where W_v^* and W_s^* are the classifier weights obtained at the end of the relaxed optimization. Because $y^2 = y$ when y is binary, (5.9) is actually a *linear* objective over the binary matrix Y_n for Z_n fixed. Thus we can optimize (5.9) *exactly* by solving a dynamic program on Y_n for each of the T_n possibilities of Z_n, yielding $O(M_n T_n)$ time complexity per clip. The joint rounding algorithm is provided in Algorithm 2.

Algorithm 2 Joint cost rounding for video n

Get W_s^* and W_a^* from the relaxed problem.
Initialize Z^*, Y^* and $\text{val}^* = +\infty$.
# Loop over all possibilities for Z_n
for t in $1 \dots T_n$ **do**
  $Z \leftarrow \texttt{zeros}(T_n, 1)$ # Set the t-th entry of Z to 1
  $Z_t \leftarrow 1$
  # Definition of the cost matrix
  $C_n \leftarrow \frac{1}{2M}\left(\texttt{ones}(M_n, 2) - 2X_s W_s\right) + \frac{\nu}{T} B_n Z \mathbf{1}_2^\top$
  # Dynamic program for the labeling of the tracks
  $Y_{\min} \leftarrow \operatorname{argmin}_{Y \in \mathcal{Y}_n} \operatorname{Tr}(C_n^T Y)$
  # Cost computation
  $\text{cost}_Z \leftarrow \frac{1}{2T}\|Z - X_a W_a\|_F^2$
  $\text{cost}_Y \leftarrow \frac{1}{2M}\|Y_{\min} - X_s W_s\|_F^2$
  $\text{cost}_{ZY} \leftarrow \frac{\nu}{T} d(Z, Y_{\min})$
  # Update solution if better
  $\text{val} \leftarrow \text{cost}_Z + \text{cost}_Y + \text{cost}_{ZY}$
  if $\text{val} < \text{val}^*$ **then**
    $Z^* \leftarrow Z$
    $Y^* \leftarrow Y_{\min}$
    $\text{val}^* \leftarrow \text{val}$
  end if
end for
return Z^*, Y^*

5.5 Experiments

In this section, we first describe our dataset, the object tracking pipeline and the feature representation for object tracklets and videos (Section 5.5.1). We consider two experimental set-ups. In the first weakly-supervised set-up (Section 5.5.2), we apply our method on a set of video clips which we know contain the manipulation action of interest but do not know its precise temporal localization. In the second, more challenging “in the wild" set-up (Section 5.5.3), the input set of weakly-supervised clips is obtained by automatic processing of text associated with the videos and hence may contain erroneous clips that do not contain the manipulation action of interest.

5.5.1 Dataset and features

Dataset of manipulation actions. We build a dataset of manipulation actions by collecting videos from different sources: the instructional video dataset introduced in Chapter 4, the Charades dataset from [Sigurdsson *et al.*, 2016], and some additional videos downloaded from YouTube. We focus on “third person" videos (rather than egocentric) as such videos depict a variety of people in different settings and can be obtained on a large scale from YouTube. We annotate the precise temporal extent of seven different actions[4] applied to five distinct objects[5]. This results in 630 annotated occurrences of ground truth manipulation action.

To evaluate object state recognition, we define a list of two states for each object. We then run automatic object detector for each involved object, track the detected object occurrences throughout the video and then subdivide the resulting long tracks into short tracklets. Finally, we label ground truth object states for tracklets within ± 40 seconds of each manipulation action. We label four possible states: *state 1*, *state 2*, *ambiguous state* or *false positive detection*. The ambiguous state covers the (not so common) in-between cases, such as cup half-full. In total, we have 19,499 fully annotated tracklets out of which: 35% cover *state 1* or *state 2*, 25% are ambiguous, and 40% are false positives.

[4] *put the wheel on the car (47 clips)*, *withdraw the wheel from the car (46)*, *place a plant inside a pot (27)*, *open an oyster (28)*, *open a refrigerator (234)*, *close a refrigerator (191)* and *pour coffee (57)*.

[5] *car wheel*, *flower pot*, *oyster*, *refrigerator* and *coffee cup*.

Objects	**Actions** (#clips)	*States*	#Tracklets
wheel	{**remove** (47), **put** (46)}	{*attached, detached*}	5447
coffee cup	{**fill** (57)}	{*full, empty*}	1819
flower pot	{**put plant** (27)}	{*full, empty*}	2463
fridge	{**open** (234), **close** (191)}	{*open, closed*}	7968
oyster	{**open** (28)}	{*open, closed*}	1802

Tab. 5.1.: Statistics of our new dataset of manipulated objects

Note that this annotation is only used for evaluation purpose, and not by any of our models. Table 5.1 provides more precise statistics for the dataset. For each object class we indicate associated action classes and the number of video clips for each action. We also provide the list of states and the number of object tracklets with state annotations.

Object detection and tracking. In order to obtain detectors for the five objects, we finetune the FastRCNN network [Girshick, 2015] with training data from ImageNet [Deng *et al.*, 2009]. We use bounding box annotations from ImageNet when available (*e.g.*the "wheel" class). For the other classes, we manually labeled more than 500 instances per class. In our set-up with only moderate amount of training data, we observed that class-agnostic object proposals combined with FastRCNN performed better than FasterRCNN [Ren *et al.*, 2015]. In detail, we use geodesic object proposals [Krähenbühl and Koltun, 2014] and set a relatively low object detection threshold (0.4) to have good recall. We track objects using a generic KLT tracker from [Bojanowski *et al.*, 2013]. The tracks are then post-processed into shorter tracklets that last about one second and thus are likely to have only one object state.

Object tracklet representation. For each detected object, represented by a set of bounding boxes over the course of the tracklet, we compute a CNN feature from each (extended) bounding box that we then average over the length of the tracklet to get the final representation. The CNN feature is extracted with a ROI pooling [Ren *et al.*, 2015] of ResNet50 [He *et al.*, 2016]. The ROI pooling notably allows to capture some context around the object which is important for some cases (*e.g.wheel* "on" or "off" the car). The resulting feature descriptor of each object tracklet is 8,192 dimensional.

Video representation for recognizing actions. Following recent approaches [Bojanowski *et al.*, 2014; Bojanowski *et al.*, 2015] and Chapter 4, each video is divided into chunks of 10 frames that are represented by a motion and ap-

pearance descriptor averaged over 30 frames. For the motion we use a 2,000 dimensional bag-of-word representation of histogram of local optical flow (HOF) obtained from Improved Dense Trajectories [Wang and Schmid, 2013]. Following Chapter 4, we add an appearance vector that is obtained from a 1,000 dimensional bag-of-word vector of conv5 features from VGG16 [Simonyan and Zisserman, 2015]. This results in a 3,000 dimensional feature vector for each chunk of 10 frames.

5.5.2 Weakly supervised object state discovery

Experimental setup. We first apply our method in a weakly supervised set-up where for each action we provide an input set of clips, where we know the action occurs somewhere in the clip but we do not provide the precise temporal localization. Each clip may contain other actions that affect other objects or actions that do not affect any object at all (e.g. walking / jumping). The input clips are about 20s long and are obtained by taking approximately $\pm$ 10s of each annotated manipulation action.

Evaluation metric: average precision. For all variants of our method, we use the rounded solution that reached the smallest objective during optimization. We evaluate these predictions with a precision score averaged over all the videos. A temporal action localization is said to be correct if it falls within the ground truth time interval. Similarly, a state prediction is correct if it matches the ground truth state.[6] Note that a "precision" metric is reasonable in our set-up as our method is forced to predict in all videos, i.e. the recall level is fixed to all videos and the method cannot produce high precision with low recall.

Hyperparameters. In all methods that involve a discriminative clustering objective, we used $\lambda = 10^{-2}$ (action localization) and $\mu = 10^{-4}$ (state discovery) for all 7 actions. For joint methods that optimize (5.1), we set the weight ν of the distortion measure (5.5) to 1.

State discovery results. Results are shown in the top part of Table 5.2. In the following, we refer to "State only" whenever we use our method without

[6] In particular, we count "*ambiguous*" labels as incorrect.

	Method	put wheel	remove wheel	fill pot	open oyster	fill coff.cup	open fridge	close fridge	**Average**
State discovery	(**a**) Chance	0.35	0.38	0.35	0.36	0.31	0.29	0.42	0.35
	(**b**) Kmeans	0.25	0.12	0.11	0.23	0.14	0.19	0.22	0.18
	(**c**) Salient state only	0.35	0.48	0.35	0.38	0.30	0.40	0.37	0.38
	(**d**) At least one state only	0.43	0.55	0.46	0.52	0.29	0.43	0.39	0.44
	(**e**) Joint model	**0.52**	0.59	**0.50**	0.45	0.39	**0.47**	**0.47**	0.48
	(**f**) Joint model + det. scores.	0.47	**0.65**	**0.50**	**0.61**	**0.44**	0.46	0.43	**0.51**
	(**g**) Joint + GT act. feat.	0.55	0.56	0.56	0.52	0.46	0.45	0.49	0.51
Action localization	(**i**) Chance	0.31	0.20	0.15	0.11	0.40	0.23	0.17	0.22
	(**ii**) [Bojanowski *et al.*, 2015]	0.24	0.13	0.11	0.14	0.26	0.29	0.23	0.20
	(**iii**) [Bojanowski *et al.*, 2015] + object cues	0.24	0.13	0.26	0.07	**0.84**	0.33	0.37	0.32
	(**iv**) Joint model	**0.67**	**0.57**	**0.48**	**0.32**	0.82	**0.57**	**0.44**	**0.55**
	(**v**) Joint + GT stat. feat.	0.72	0.66	0.44	0.46	0.86	0.55	0.44	0.59

Tab. 5.2.: State discovery (top) and action localization results (bottom).

looking at the action cost or the distortion measure (5.1). We compare to two baselines for the state discovery task. Baseline (**a**) evaluates chance performance by running our "State only" method with random features for representing tracklets, using only constraints (at least one ordering + non overlap) to make predictions. We use random features because computing analytically the chance performance on this task is difficult due to the complex spatial-temporal constraints. We believe this baseline is an important metric to measure the difficulty of the problem and the information brought by the ordering constraints. Baseline (**b**) performs K-means clustering of the tracklets with $K = 3$ (2 clusters for the states and 1 for false positives). We report performance of the best assignment for the solution with the lowest objective after 10 different initializations. The next two methods are "State only" variants. Method (**c**) corresponds to a replacement of the "at least one constraint" by an "exactly one constraint" while the method (**d**) uses our new constraint. Finally, we report three joint methods that use our new joint rounding technique (5.9) for prediction. Method (**e**) corresponds to our joint method that optimizes (5.1). Method (**f**) is a simple improvement taking into account object detection score in the objective (details below). Finally, method (**g**) is our joint method but using the action ground truth labels as video features in order to test the effect of having perfect action localization for the task of object state discovery.

We first note that method (**d**) outperforms (**c**), thus highlighting the importance of the "at least one" constraint for modeling object states. While the saliency approach (taking only the most confident detection per video) was useful for action modeling in [Bojanowski *et al.*, 2015], it is less suitable for our set-up where multiple tracklets can be in the same state. The joint approach with actions (**e**) outperforms the "State only" method (**d**) on 6

out of 7 actions and obtains better average performance, confirming the benefits of joint modeling of actions and object states. Using ground truth action locations further improves results (cf. (**g**) against (**e**)). Our weakly supervised approach (**e**) performs not much lower compared to using ground truth actions (**g**), except for the states of the coffee cup (empty/full). In this case we observe that a high number of false positive detections confuses our method. A simple way to address this issue is to add the object detection score into the objective of our method, which then prefers to assign object states to higher scoring object candidates further reducing the effect of false positives. This can be done easily by adding a linear cost reflecting the object detection score to objective (5.1). We denote this modified method "(**f**) Joint model + det. scores". This method achieves the best average performance and highlights that additional information can be easily added to our model.

Action localization results. We compare our method to three different baselines and give results in the bottom part of Table 5.2. Baseline (**i**) corresponds to chance performance, where the precision for each clip is simply the proportion of the entire clip taken by the ground truth time interval. Baseline (**ii**) is the method introduced in [Bojanowski *et al.*, 2015] used here with only one action. It also corresponds to a special case of our method where the object state part of the objective in equation (5.1) is turned off (salient action only). Interestingly, this baseline is actually worse than chance for several actions. This is because without additional information about objects, this method localizes other common actions in the clip and not the action manipulating the object of interest. This also demonstrates the difficulty of our experimental set-up where the input video clips often contain multiple different actions. To address this issue, we also evaluate baseline (**iii**), which complements [Bojanowski *et al.*, 2015] with the additional constraint that the action prediction has to be within the first and the last frame where the object of interest is detected, improving the overall performance above chance. Our joint approach (**iv**) consistently outperforms these baselines on all actions, thus showing again the strong link between object states and actions. Finally, the approach (**v**) is the analog of method (**g**) for action localization where we use ground truth state labels as tracklet features in our joint formulation showing that the action localization can be further improved with better object state descriptors.

Features	put wheel	remove wheel	fill pot	open oyster	fill coff.cup	open fridge	close fridge	Average
(1) CNN + HOF IDT	**0.65**	0.68	0.56	0.11	0.91	0.54	0.59	0.58
(2) CNN + Full IDT	**0.65**	**0.72**	**0.56**	**0.21**	**0.93**	**0.6**	**0.62**	**0.61**

Tab. 5.3.: Results of supervised baselines for action localization.

In addition, we also compare to supervised baseline methods with state-of-the-art features. To be able to compare numbers with our experiment, we used a leave-one-out technique. For each action, we train a binary classifier with SVM on all videos except one. Similarly to our setting, we then select the top scoring time interval of the left alone test video. We repeat this process for all videos and report the same metric reported in Table 5.2. For baseline **(1)**, we use the features described in Section 5.5.1. For baseline **(2)**, we complete our features with all channels of Improved Dense Trajectories (IDT) [Wang and Schmid, 2013]. Detailed results are given in Table 5.3. We observe that we obtain results that are on par with our weakly supervised baselines (0.55 versus 0.58), therefore demonstrating the potential of using the information of object states for action localization. We also note that there is not a big difference between the two different features, hence justifying our use of only 'HOF' descriptor for the improved dense trajectories which allows us to lower the dimension of the features and hence speed up our method.

Benefits of joint object-action modeling. We observe that the joint modeling of object states and actions benefits both tasks. This effect is even more pronounced for actions. Intuitively, knowing perfectly the object states greatly reduces the search space for action localization. In addition, despite the recent major progress in object recognition using CNNs, action recognition still remains a hard problem with much room for improvement. Example qualitative results are shown in Fig. 5.4.

Failure cases. We observed two main types of failures, illustrated in Figure 5.5. The first one occurs when a false positive object detection consistently satisfies the hypothesis of our model in multiple videos (the top two rows in Figure 5.5). The second typical failure mode is due to ambiguous labels (bottom row in Figure 5.5). This highlights the difficulty in annotating ground truth for long actions such as "*pouring coffee*".

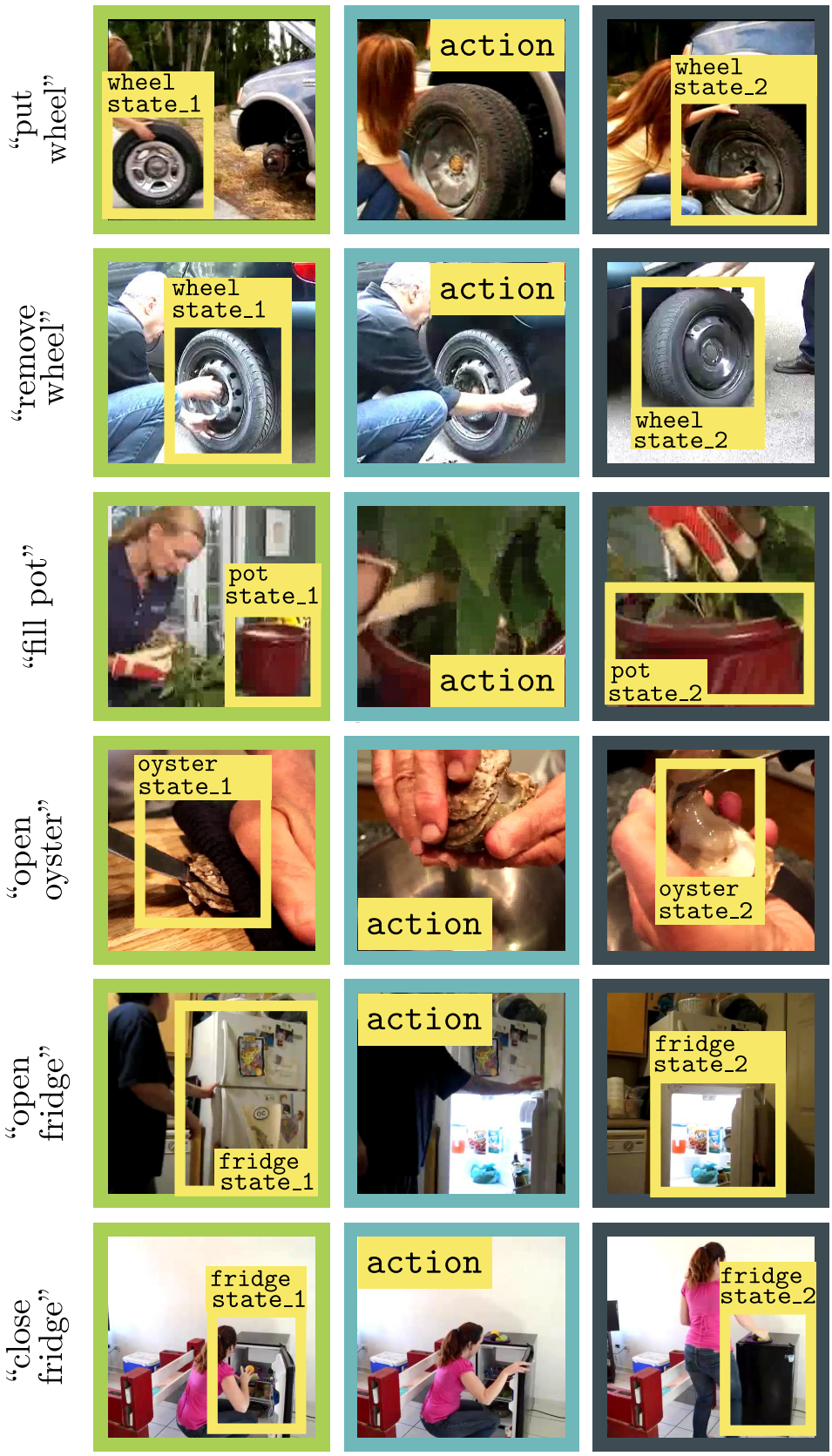


Fig. 5.4.: Qualitative results for joint action localization (middle) and state discovery (left and right) (see Fig. 5.1 for "*fill coffee cup*").

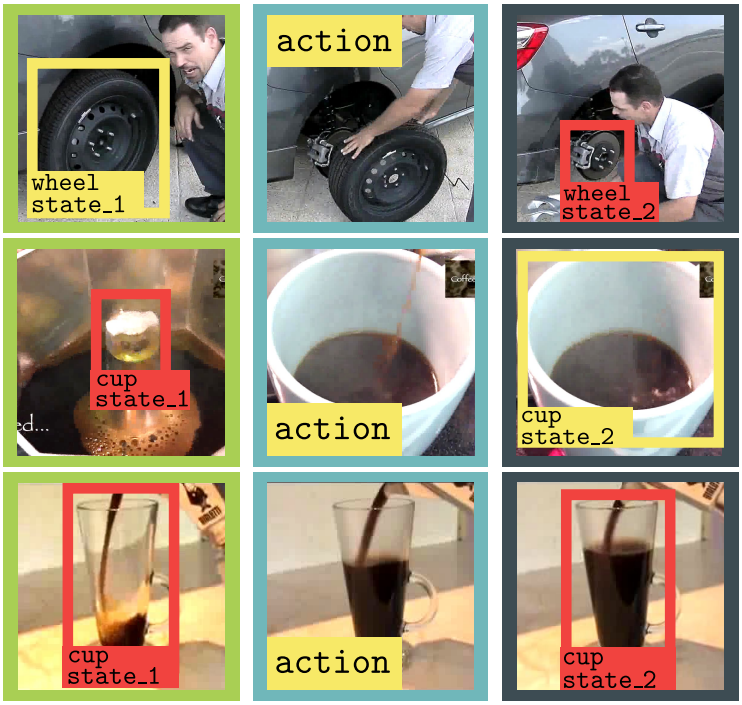


Fig. 5.5.: Typical failure cases for "*removing car wheel*" (top) and "*fill coffee cup*" (middle, bottom) actions. Yellow indicates correct predictions; red indicates mistakes. Top: the removed wheel is incorrectly localized (right). Middle: the "empty cup" is incorrectly localized (left). Bottom: In this case, both object tracklets are annotated as "*ambiguous*" in the ground truth as they occur during the pouring action and hence the predictions, while they appear reasonable, are deemed incorrect.

5.5.3 Object state discovery in the wild

Towards the discovery of a large number of manipulation actions and state changes, we next apply our method in an automatic setting, where action clips have been obtained using automatic text-based retrieval.

Clip retrieval by text. Instructional videos [Alayrac *et al.*, 2016a; Malmaud *et al.*, 2015; Sener *et al.*, 2015] usually come with a narration provided by the speaker describing the performed sequence of actions. In this experiment, we keep only such *narrated instructional videos* from our dataset. This results in the total of 140 videos that are 3 minutes long in average. We extract the narration in the form of subtitles associated with the video. These subtitles have been directly downloaded from YouTube and have been obtained either by Youtube's Automatic Speech Recognition (ASR) or provided by the users.

We use the resulting text to retrieve clip candidates that may contain the action modifying the state of an object. Obtaining the approximate temporal location of actions from the transcribed narration is still very challenging due to ambiguities in language ("undo bolt" and "loosen nut" refer to the same manipulation) and only coarse temporal localization of the action provided by the narration. Given a manipulation action such as "remove tire", we first find positive and negative sentences relevant for the action from an instruction website such as Wikihow. On average we obtain about 12 positive and 50 negative sentences per action.

Off-the-shelf methods for text parsing typically fail in the absence of punctuation. To process ASR output, which comes without punctuation, we propose to use the following simple but robust text representation. We represent every 10-word window of the narration by a TF-IDF vector based of uni-grams and bi-grams. We use the same TF-IDF representation to encode text in our Language dataset on the level of sentences.

We train binary linear SVM classifiers to identify manipulation actions using the Language dataset for training and the regularization parameter $C = 10$. The obtained classifiers are then used to score every 10-word window of text narrations. Video clips with the temporal correspondence to the top-scoring text narrations for each action are then retrieved. To deduce the temporal extent of the video clip given the top-scoring window, we trim 5 seconds before and 15 seconds after the corresponding timing. This is to account for the fact that people are usually doing the action after speaking about it.

In Figure 5.6, we provide an illustration of text based clip retrieval. This visualization demonstrates the difficulty of the addressed problem (see caption for details).

Results. As shown in Table 5.4, the pattern of results, where our joint method performs the best, is similar to the weakly supervised set-up described in Sec. 5.5.2. This highlights the robustness of our model to noisy input data – an important property for scaling-up the method to Internet scale datasets. To assess how well our joint method could do with perfect retrieval, we also report results for a "Curated" set-up where we replace the automatically

	Method	put wheel	remove wheel	fill pot	open oyster	fill coff.cup	Ave.
State disc.	**(a)** Chance	0.23	0.34	0.25	0.29	0.11	0.24
	State + det. sc.	0.33	0.48	**0.28**	0.40	0.13	0.32
	(f) Joint	**0.38**	**0.53**	0.25	**0.43**	**0.20**	**0.36**
	(f) Curated	0.63	0.68	0.63	0.63	0.53	0.62
Action local.	**(i)** Chance	0.14	0.10	0.06	0.10	0.15	0.11
	(iii) Action	0.05	0.10	0.00	0.15	**0.25**	0.11
	(iv) Joint	**0.30**	**0.30**	**0.20**	**0.20**	0.20	**0.24**
	(iv) Curated	0.53	0.35	0.32	0.40	0.59	0.44

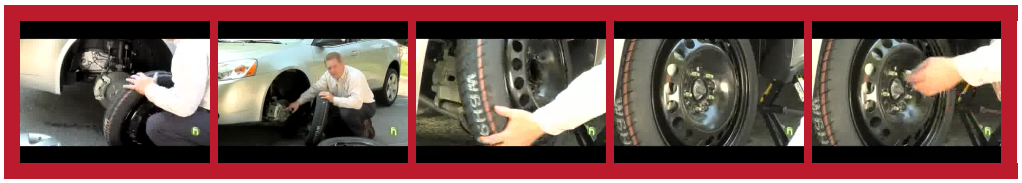
Tab. 5.4.: Results on noisy clips automatically retrieved by text.

retrieved clips with the 20s clips used in Sec. 5.5.2 for the corresponding videos.

5.6 Conclusion and future work

We have described a joint model that relates object states and manipulation actions. Given a set of input videos, our model both localizes the manipulation actions *and* discovers the corresponding object states. We have demonstrated that our joint approach improves performance of both object state recognition and action recognition. More generally, our work provides evidence that actions should be modeled in the larger context of goals and effects. Finally, our work opens up the possibility of Internet-scale learning of manipulation actions from narrated video sequences.

which we 'll just loosen those up does n't take much because they 're made out of plastic that the hubcap out of the way i have access to the lug nuts loosen the lug nuts position yourself firmly pressing counterclockwise to loosen the lug nuts not to lose just enough to crack them is a little stubborn position yourself over using your knee or your foot you can gain leverage now we 're ready to jack it up this is the common scissors jack the screen moves in and out allowing the mac mechanism to move up and down and lift your car so as you turn it to the right it will go up as you loosen it the jack will collapse allowing the car to come down now will position the jack you want to be sure to get the jack on a good part of the frame your owners manual is a good place to find where did properly jack the vehicle you can raise it up by hand until it contacts the frame it is in good position we use the jack handle to raise the vehicle insert the end of the jack handle into the jack using it as leverage it will help make the car go up easier remember turning clockwise to go up and counterclockwise to go down carefully jack the car up never stick your hands or your legs under the vehicle as the car can fall and cause damage checking can take a while so be patient and cautious as you jack ok now that the car is jacked up now we 'll take the lug nuts loose remember we already pre loosen them when the car was on the ground making it easy to notice will come right off now trying to do that while the car was jacked up would be very difficult remember to keep your lug nuts in close hand you do n't want to roll away in the grass because it 'll be hard to find and that 's what holds your tire on again we 're turning them counterclockwise to get them loose or to the left we 'll remove the flat tire set it over here out of the way now there was heavy traffic area you would n't want to leave it in the street you might want to put it to the rear of the car grab the spare tire



put it in position you 're going to center the spare tire on the wheel studs that 's what the lug nuts go so go ahead and do that lining it up you see it lines are pretty easy then install your lug nuts again clockwise is tight counterclockwise is loose so we 'll take them up by hand as far as they 'll go once the tire centered lug nuts are hand tight take the lug wrench again turning it clockwise to tighten the lugs to their firm not too tight because remember your cars up on a jack we would n't want it falling off with too much leverage of force insert the jack handle back into the jack turning it counterclockwise again we 're going to lower the vehicle again this will take some time but going down a lot easier than going up it seems to be going down easy you can just do it like this by doing it straight instead of having an angle but if it 's too tough you can angle it and get the leverage that you 'll need to lower the jack or to raise the jack turn the jack low enough you can do it by hand now that it does n't have contact with the frame and we can remove it now that the tires on the ground we 'll go ahead and give them that final type we 're going to start down here then move to the top one then back down over and back again a little star pattern making sure we have equal torque on the wheel stud ok what we have our spare tire on it set it up we 're ready to go a couple things to remember

Fig. 5.6.: Illustration of our text based clip retrieval approach for the action "put wheel on a car". We display the narration of the video that is obtained from Automatic Speech Recognition (ASR). The text is highlighted with different colors. These colors correspond to the score of the SVM that has been trained to detect sentences which refer to the action of interest. More precisely, for each word, we compute the average score over all the windows that contain that word. Red indicates high score, blue indicates low score. The shown frames correspond to the top scoring part of the narration. The highlighted narration is consistent with the demonstrated actions in the video. Note that finding the correct clips is a very difficult problem. First, the input narration is very long. Second, the text directly comes from ASR, therefore it contains mistakes and does not have any punctuation. Third, several different expressions are similar and could refer to the action of interest. Despite all these challenges, our method is able to correctly retrieve the clip that contains the action of interest.

6 Minding the Gaps for Block Frank-Wolfe Optimization

In the previous chapters, we have described two approaches in the context of instructional videos: the first one to discover and localize actions steps in videos and the second one to relate action with a change of state in the manipulated objects. Both methods are designed to work in a weakly supervised setting where only a partial information is known at training time. To that end, we employ a discriminative clustering technique, named DIFFRAC [Bach and Harchaoui, 2007], that consists in an optimization problem under constraints for which we introduce a convex relaxation. We use the Frank-Wolfe method to solve the relaxed problem, an algorithm particularly adapted to the constraints that govern our problem.

One main interest of instructional videos is that they are available at a large scale on the web which could potentially benefit the quality of the learned models. However, the Frank-Wolfe algorithm is a *batch* algorithm, i.e. it requires to see the whole dataset before doing an update on the model parameters. This is not practical for large scale scenario in which thousands of videos are available. Lacoste-Julien *et al.*, [2013] propose the block-coordinate Frank-Wolfe (BCFW) algorithm, a variant of the Frank-Wolfe algorithm that work when the set of constraints can be written as a product of smaller so-called blocks, a case which we encounter in practice as the constraints for our problem typically decompose over different videos. It alleviates the scaling problem of Frank-Wolfe through a randomized approach that consists in sampling a block at random and perform a smaller Frank-Wolfe step on that block at each iterate.

In this chapter, we propose several improvements on the BCFW algorithm. To fairly compare with [Lacoste-Julien *et al.*, 2013], we benchmark these improvements on the structured support vector machine (SSVM) objective in the context of structured prediction. However, our contributions are not specific to SSVM and can directly benefit discriminative clustering applications as recently demonstrated in our recent paper [Miech *et al.*, 2017].

The key intuition behind our improvements is that the estimates of block gaps maintained by BCFW reveal the block suboptimality that can be used as an *adaptive* criterion. First, we sample objects at each iteration of BCFW in an adaptive non-uniform way via gap-based sampling. Second, we incorporate pairwise and away-step variants of Frank-Wolfe into the block-coordinate setting. Third, we cache oracle calls with a cache-hit criterion based on the block gaps. Finally, we provide an exhaustive empirical evaluation of all our methods on four structured prediction datasets.

6.1 Introduction

One of the most popular learning objectives for structured prediction is the structured support vector machine [Taskar *et al.*, 2003; Tsochantaridis *et al.*, 2005], which generalizes the classical binary SVM to problems with structured outputs. In this chapter, we consider the ℓ_2-regularized ℓ_1-slack structured SVM, to which we will simply refer as SSVM. The SSVM method consists in the minimization of the regularized structured hinge-loss on the labeled training set. The optimization problem of SSVM is of significant complexity and, thus, hard to scale up. In the literature, multiple optimization methods have been applied to tackle this problem, including cutting-plane methods [Tsochantaridis *et al.*, 2005; Joachims *et al.*, 2009] and stochastic subgradient methods [Ratliff *et al.*, 2007], among others.

Recently, Lacoste-Julien *et al.*, [2013] proposed the block-coordinate Frank-Wolfe method (BCFW), which is currently one of the state-of-the-art algorithms for SSVM.[1] In contrast to the classical (batch) Frank-Wolfe algorithm [Frank and Wolfe, 1956], BCFW is a randomized block-coordinate method that works on *block-separable* convex compact domains. In the case of SSVM, BCFW operates in the dual domain and iteratively applies Frank-Wolfe steps on the blocks of dual variables corresponding to different objects of the training set. Distinctive features of BCFW for SSVM include optimal step size selection leading to the absence of the step-size parameter, convergence guarantees for the primal objective, and ability to compute the duality gap as a stopping criterion.

[1] Independently, Branson *et al.*, [2013] proposed their SVM-IS algorithm which is equivalent to BCFW in some scenarios.

Notably, the duality gap obtained by BCFW can be written as a sum of block gaps, where each block of dual variables corresponds to one training example. In this chapter, we exploit this property and improve the BCFW algorithm in multiple ways. First, we substitute the standard uniform sampling of objects at each iteration with an adaptive non-uniform sampling. Our procedure consists in sampling objects with probabilities proportional to the values of their block gaps, giving one of the first fully *adaptive* sampling approaches in the optimization literature that we are aware of. This choice of sampling probabilities is motivated by the intuition that objects with higher block gaps potentially can provide more improvement to the objective. We analyze the effects of the gap-based sampling on convergence and discuss the practical trade-offs.

Second, we apply pairwise [Mitchell *et al.*, 1974] and away [Wolfe, 1970] steps of Frank-Wolfe to the block-coordinate setting. This modification is motivated by the fact that batch algorithms based on these steps have linear convergence rates [Lacoste-Julien and Jaggi, 2015b] whereas convergence of standard Frank-Wolfe is sublinear.

Third, we cache oracle calls and propose a gap-based criterion for calling the oracle (cache miss vs. cache hit). Caching the oracle calls was shown do deliver significant speed-ups when the oracle is expensive, e.g., in the case of cutting-plane methods [Joachims *et al.*, 2009].

Contributions. Overall, we make the following contributions: (i) adaptive non-uniform sampling of the training objects; (ii) pairwise and away steps in the block-coordinate setting; (iii) gap-based criterion for caching the oracle calls. These contributions are general to BCFW and thus could be applied to other block-separable optimization problems where BCFW could or have been used such as for discriminative learning applications presented in Chapter 4 and Chapter 5, video co-localization [Joulin *et al.*, 2014b] or structured submodular optimization [Jegelka *et al.*, 2013], among others.

This Chapter is organized as follows. In Section 6.2, we describe the setup and review the BCFW algorithm. In Section 6.3, we describe our contributions: adaptive sampling (Section 6.3.1), pairwise and away steps (Section 6.3.2), caching (Section 6.3.3). We discuss the related work in the relevant sections

of the paper. Section 6.4 contains the experimental study of the methods. The code and datasets are available at our project webpage.[2] Finally the Appendix A contains additional content about the paper, e.g. proofs or details about the datasets used in our experimental study.

6.2 Background

6.2.1 Structured Support Vector Machine (SSVM)

In structured prediction, we are given an input $\boldsymbol{x} \in \mathcal{X}$, and the goal is to predict a structured object $\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})$ (such as a sequence of tags). In the standard setup for structured SVM (SSVM) [Taskar *et al.*, 2003; Tsochantaridis *et al.*, 2005], we assume that prediction is performed with a linear model $h_{\boldsymbol{w}}(\boldsymbol{x}) = \operatorname{argmax}_{\boldsymbol{y}\in\mathcal{Y}(\boldsymbol{x})} \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})\rangle$ parameterized by the weight vector $\boldsymbol{w}$ where the structured feature map $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^d$ encodes the relevant information for input/output pairs. We reuse below the notation and setup from Lacoste-Julien *et al.*, [2013]. Given a labeled training set $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, the parameters $\boldsymbol{w}$ are estimated by solving a convex non-smooth optimization problem

$$\min_{\boldsymbol{w}} \quad \tfrac{\lambda}{2} \|\boldsymbol{w}\|^2 + \tfrac{1}{n} \sum_{i=1}^n \tilde{H}_i(\boldsymbol{w}) \tag{6.1}$$

where λ is the regularization parameter and $\tilde{H}_i(\boldsymbol{w})$ is the structured hinge loss defined using the *loss-augmented decoding* subproblem (or *maximization oracle*):

$$\text{‘max oracle’} \qquad \tilde{H}_i(\boldsymbol{w}) := \max_{\boldsymbol{y}\in\mathcal{Y}_i} \underbrace{L_i(\boldsymbol{y}) - \langle \boldsymbol{w}, \boldsymbol{\psi}_i(\boldsymbol{y})\rangle}_{=: H_i(\boldsymbol{y};\boldsymbol{w})}. \tag{6.2}$$

Here $\boldsymbol{\psi}_i(\boldsymbol{y}) := \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}_i) - \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y})$, $\mathcal{Y}_i := \mathcal{Y}(\boldsymbol{x}_i)$, and $L_i(\boldsymbol{y}) := L(\boldsymbol{y}_i, \boldsymbol{y})$ denotes the task-dependent structured error of predicting an output $\boldsymbol{y}$ instead of the observed output $\boldsymbol{y}_i$ (e.g., a Hamming distance between two sequences).

[2] http://www.di.ens.fr/sierra/research/gapBCFW/

Dual formulation. The negative of a Fenchel dual for objective (6.1) can be written as

$$\min_{\substack{\boldsymbol{\alpha}\in\mathbb{R}^m \\ \boldsymbol{\alpha}\succcurlyeq 0}} \quad f(\boldsymbol{\alpha}) \;:=\; \tfrac{\lambda}{2}\left\|A\boldsymbol{\alpha}\right\|^2 - \boldsymbol{b}^\mathsf{T}\boldsymbol{\alpha} \qquad \text{s.t.} \quad \textstyle\sum_{\boldsymbol{y}\in\mathcal{Y}_i} \alpha_i(\boldsymbol{y}) = 1 \quad \forall i \in [n] \tag{6.3}$$

where $\alpha_i(\boldsymbol{y})$, $i \in [n] := \{1,\ldots,n\}$, $\boldsymbol{y} \in \mathcal{Y}_i$ are the dual variables. The matrix $A \in \mathbb{R}^{d\times m}$ consists of the $m := \sum_i m_i = \sum_i |\mathcal{Y}_i|$ columns

$$A := \left\{ \tfrac{1}{\lambda n}\boldsymbol{\psi}_i(\boldsymbol{y}) \in \mathbb{R}^d \;\middle|\; i \in [n], \boldsymbol{y} \in \mathcal{Y}_i \right\}, \tag{6.4}$$

and the vector $\boldsymbol{b} \in \mathbb{R}^m$ is given by $\boldsymbol{b} := \left(\frac{1}{n}L_i(\boldsymbol{y})\right)_{i\in[n],\boldsymbol{y}\in\mathcal{Y}_i}$.

In SSVM (as for the standard SVM), the Karush-Kuhn-Tucker (KKT) optimality conditions can give the primal variables $\boldsymbol{w}(\boldsymbol{\alpha}) = A\boldsymbol{\alpha} = \sum_{i,\,\boldsymbol{y}\in\mathcal{Y}_i} \alpha_i(\boldsymbol{y})\frac{\boldsymbol{\psi}_i(\boldsymbol{y})}{\lambda n}$ corresponding to the dual variables $\boldsymbol{\alpha}$ (see, e.g., [Lacoste-Julien *et al.*, 2013, App. E]). The gradient of f then takes a simple form $\nabla f(\boldsymbol{\alpha}) = \lambda A^\mathsf{T} A\boldsymbol{\alpha} - \boldsymbol{b} = \lambda A^\mathsf{T}\boldsymbol{w} - \boldsymbol{b}$; its $(i,\boldsymbol{y})$-th component equals $-\frac{1}{n}H_i(\boldsymbol{y};\boldsymbol{w})$.

6.2.2 Block Coordinate Frank-Wolfe (BCFW)

We give in Algorithm 3 the BCFW algorithm from Lacoste-Julien *et al.*, [2013] applied to problem (6.3). It exploits the block-separability of the domain $\mathcal{M} := \Delta_{|\mathcal{Y}_1|} \times \ldots \times \Delta_{|\mathcal{Y}_n|}$ for problem (6.3) and sequentially applies the Frank-Wolfe steps to the blocks of the dual variables $\boldsymbol{\alpha}_{(i)} \in \mathcal{M}^{(i)} := \Delta_{|\mathcal{Y}_i|}$.

While BCFW works on the dual (6.3) of SSVM, it only maintains explicitly the primal variables via the relationship $\boldsymbol{w}(\boldsymbol{\alpha})$. Most importantly, the Frank-Wolfe linear oracle on block i at iterate $\boldsymbol{\alpha}^{(k)}$ is equivalent to the max oracle (6.2) at the corresponding weight vector $\boldsymbol{w}^{(k)} := A\boldsymbol{\alpha}^{(k)}$ [Lacoste-Julien *et al.*, 2013, App. B.1] (see line 4 of Algorithm 3):

$$\max_{\boldsymbol{s}_{(i)}\in\mathcal{M}^{(i)}} \left\langle \boldsymbol{s}_{(i)}, -\nabla_{(i)} f(\boldsymbol{\alpha}^{(k)}) \right\rangle = \tfrac{1}{n} \max_{\boldsymbol{y}\in\mathcal{Y}_i} H_i(\boldsymbol{y};\boldsymbol{w}^{(k)}). \tag{6.5}$$

Here, the operator $\nabla_{(i)}$ denotes the partial gradient corresponding to the block i, i.e., $\nabla f = (\nabla_{(i)} f)_{i=1}^n$. Note that each argmax of the r.h.s. of (6.5),

$\boldsymbol{y}^*_{(i)}$, corresponds to a corner $\boldsymbol{s}^*_{(i)}$ of the polytope $\mathcal{M}^{(i)}$ maximizing the l.h.s. of (6.5).

As the objective (6.3) is quadratic, the optimal step size that yields the maximal improvement in the chosen direction $\boldsymbol{s}^*_{(i)} - \boldsymbol{\alpha}^{(k)}_{(i)}$ can be found analytically (Line 7 of Algorithm 3).

6.2.3 Duality gap

At each iteration, the batch Frank-Wolfe algorithm [Frank and Wolfe, 1956], [Lacoste-Julien *et al.*, 2013, Section 3] computes the following quantity, known as the *linearization duality gap* or *Frank-Wolfe gap*:

$$g(\boldsymbol{\alpha}) := \max_{\boldsymbol{s}\in\mathcal{M}} \langle \boldsymbol{\alpha} - \boldsymbol{s}, \nabla f(\boldsymbol{\alpha})\rangle = \langle \boldsymbol{\alpha} - \boldsymbol{s}^*, \nabla f(\boldsymbol{\alpha})\rangle. \tag{6.6}$$

It turns out that this Frank-Wolfe gap exactly equals the Lagrange duality gap between the dual objective (6.3) at a point $\boldsymbol{\alpha}$ and the primal objective (6.1) at the point $\boldsymbol{w}(\boldsymbol{\alpha}) = A\boldsymbol{\alpha}$ [Lacoste-Julien *et al.*, 2013, App. B.2].

Because of the separability of $\mathcal{M}$, the Frank-Wolfe gap (6.6) can be represented here as a sum of block gaps $g_i(\boldsymbol{\alpha})$, $g(\boldsymbol{\alpha}) = \sum_{i=1}^n g_i(\boldsymbol{\alpha})$, where

$$g_i(\boldsymbol{\alpha}) := \max_{\boldsymbol{s}_{(i)}\in\mathcal{M}^{(i)}} \left\langle \boldsymbol{\alpha}_{(i)} - \boldsymbol{s}_{(i)}, \nabla_{(i)} f(\boldsymbol{\alpha})\right\rangle. \tag{6.7}$$

Algorithm 3 Block-Coordinate Frank-Wolfe (BCFW) algorithm for structured SVM

1: Let $\boldsymbol{w}^{(0)} := \boldsymbol{w}_i^{(0)} := \boldsymbol{0}$; $\ell^{(0)} := \ell_i^{(0)} := 0$
2: **for** $k := 0, \dots, \infty$ **do**
3: Pick i at random in $\{1, \dots, n\}$
4: Solve $\boldsymbol{y}_i^* := \operatorname*{argmax}_{\boldsymbol{y}\in\mathcal{Y}_i} H_i(\boldsymbol{y}; \boldsymbol{w}^{(k)})$
5: Let $\boldsymbol{w}_{\boldsymbol{s}} := \frac{1}{\lambda n}\boldsymbol{\psi}_i(\boldsymbol{y}_i^*)$ and $\ell_{\boldsymbol{s}} := \frac{1}{n} L_i(\boldsymbol{y}_i^*)$
6: Let $g_i^{(k)} := \lambda(\boldsymbol{w}_i^{(k)} - \boldsymbol{w}_{\boldsymbol{s}})^\mathsf{T}\boldsymbol{w}^{(k)} - \ell_i^{(k)} + \ell_{\boldsymbol{s}}$
7: Let $\gamma := \frac{g_i^{(k)}}{\lambda\|\boldsymbol{w}_i^{(k)} - \boldsymbol{w}_{\boldsymbol{s}}\|^2}$ and clip to $[0, 1]$
8: Update $\boldsymbol{w}_i^{(k+1)} := (1-\gamma)\boldsymbol{w}_i^{(k)} + \gamma\, \boldsymbol{w}_{\boldsymbol{s}}$
9: and $\ell_i^{(k+1)} := (1-\gamma)\ell_i^{(k)} + \gamma\, \ell_{\boldsymbol{s}}$
10: Update $\boldsymbol{w}^{(k+1)} := \boldsymbol{w}^{(k)} + \boldsymbol{w}_i^{(k+1)} - \boldsymbol{w}_i^{(k)}$
11: and $\ell^{(k+1)} := \ell^{(k)} + \ell_i^{(k+1)} - \ell_i^{(k)}$
12: **end for**

Block gaps can be easily computed using the quantities maintained by Algorithm 3 (see line 6).

Finally, we can rewrite the block gap in the form

$$g_i(\boldsymbol{\alpha}) = \frac{1}{n}\left(\max_{\boldsymbol{y}\in\mathcal{Y}_i} H_i(y;\boldsymbol{w}) - \sum_{\boldsymbol{y}\in\mathcal{Y}_i} \alpha_i(\boldsymbol{y}) H_i(\boldsymbol{y};\boldsymbol{w})\right) \tag{6.8}$$

providing understandable intuition of when the block gap equals zero. This is the case when all the support vectors, i.e., labelings corresponding to $\alpha_i(\boldsymbol{y}) > 0$, are tied solutions of the max oracle (6.5).

6.2.4 Convergence of BCFW

Lacoste-Julien *et al.*, [2013] prove the convergence of the BCFW algorithm at a rate $\mathcal{O}(\frac{1}{k})$.

Theorem 1 (Lacoste-Julien *et al.*, [2013], Theorem 2)**.** *For each $k \geq 0$, the iterate[3] $\boldsymbol{\alpha}^{(k)}$ of Algorithm 3 satisfies $\mathbf{E}\left[f(\boldsymbol{\alpha}^{(k)})\right] - f(\boldsymbol{\alpha}^*) \leq \frac{2n}{k+2n}\left(C_f^{\otimes} + h_0\right)$, where $\boldsymbol{\alpha}^* \in \mathcal{M}$ is a solution of the problem (6.3), $h_0 := f(\boldsymbol{\alpha}^{(0)}) - f(\boldsymbol{\alpha}^*)$ is the suboptimality at the starting point of the algorithm, $C_f^{\otimes} := \sum_{i=1}^n C_f^{(i)}$ is the sum of the curvature constants[4] of f with respect to the domains $\mathcal{M}^{(i)}$ of individual blocks. The expectation is taken over the random choice of the block i at iterations $1, \ldots, k$ of the algorithm.*

The proof of Theorem 1 crucially depends on a standard *descent lemma* applied to a block, stating that at each iteration of BCFW, for any picked block i and any scalar $\gamma \in [0, 1]$, the following inequality holds:

$$f(\boldsymbol{\alpha}^{(k+1)}) \leq f(\boldsymbol{\alpha}^{(k)}) - \gamma g_i(\boldsymbol{\alpha}^{(k)}) + \frac{\gamma^2}{2} C_f^{(i)}. \tag{6.9}$$

We rederive inequality (6.9) as Lemma 2 in Appendix A.1. Note that $\boldsymbol{\alpha}^{(k+1)} \in \mathcal{M}$ is defined by a line search, which is why the bound (6.9) holds for any scalar $\gamma \in [0, 1]$.

[3] Note that Algorithm 3 does not maintain iterates $\boldsymbol{\alpha}^{(k)}$ explicitly. They are stored in the form of $\boldsymbol{w}^{(k)} = A\boldsymbol{\alpha}^{(k)}$.

[4] For the definition of curvature constant, see Definition 1 in App. A.1 or [Lacoste-Julien and Jaggi, 2015b, App. A]

Taking the expectation of (6.9) w.r.t. the random choice of block i (sampled uniformly on $[n]$), we get the inequality

$$\mathbf{E}\Big[f(\boldsymbol{\alpha}^{(k+1)}) \,|\, \boldsymbol{\alpha}^{(k)}\Big] \leq f(\boldsymbol{\alpha}^{(k)}) - \tfrac{\gamma}{n} g(\boldsymbol{\alpha}^{(k)}) + \tfrac{\gamma^2}{2n} C_f^{\otimes} \tag{6.10}$$

which can be used to get the convergence theorem.

6.3 Block gaps in BCFW

In this section, we propose three ways to improve the BCFW algorithm: adaptive sampling (Section 6.3.1), pairwise and away steps (Section 6.3.2) and caching (Section 6.3.3). Note that these three modifications can be straightforwardly put together in any combination. In our experimental study presented in Section 6.4, we evaluate all the possibilities.

6.3.1 Adaptive non-uniform sampling

Motivation. When optimizing finite sums such as (6.1), it is often the case that processing some summands does not lead to significant progress of the algorithm. At each iteration, the BCFW algorithm selects a training object and performs the block-coordinate step w.r.t. the corresponding dual variables. If these variables are already close to being optimal, then BCFW does not make significant progress at this iteration. Usually, it is hard to identify whether processing the summand would lead to an improvement without actually doing computations on it. The BCFW algorithm obtains at each iteration the block gap (6.7) quantifying the suboptimality on the block. In what follows, we use the block gaps to randomly choose a block (an object of the training set) at each iteration in such a way that the blocks with larger suboptimality are sampled more often (the sampling probability of a block is proportional to the value of the current gap estimate).

Convergence. Assume that at iteration k of Algorithm 3, we have the probability $p_i^{(k)}$ of sampling block i. By minimizing the descent lemma bound (6.9)

w.r.t. γ for each i independently under the assumption that $g_i(\boldsymbol{\alpha}^{(k)}) \leq C_f^{(i)}$, and then taking the conditional expectation w.r.t. i, we get

$$\mathbf{E}\Big[f(\boldsymbol{\alpha}^{(k+1)}) \,|\, \boldsymbol{\alpha}^{(k)}\Big] \leq f(\boldsymbol{\alpha}^{(k)}) - \tfrac{1}{2}\sum_{i=1}^{n} p_i^{(k)} \tfrac{g_i^2(\boldsymbol{\alpha}^{(k)})}{C_f^{(i)}}. \tag{6.11}$$

Intuitively, by adapting the probabilities $p_i^{(k)}$, we can obtain a better bound on the expected improvement of f. In the ideal scenario, one would choose deterministically the block i with the maximal value of $g_i^2(\boldsymbol{\alpha}^{(k)})/C_f^{(i)}$.

In practice, the curvature $C_f^{(i)}$ is unknown, and having access to all $g_i(\boldsymbol{\alpha}^{(k)})$'s at each step is prohibitively expensive. However, the values of the block gaps obtained at the previous iterations can serve as estimates of the block gaps at the current iteration and global updates can be done from time to time to refresh their values (see **Exploitation versus staleness trade-off** paragraph below). We use them in the following *non-uniform* gap sampling scheme: $p_i^{(k)} \propto g_i(\boldsymbol{\alpha}^{(k_i)})$. where k_i records the last iteration at which the gap i was computed. Algorithm 4 summarizes the method.

We also motivate this choice by Theorem 2 below which shows that BCFW with (exact) gap sampling converges with a better constant in the rate than BCFW with uniform sampling when the gaps are non-uniform enough (and is always better when the curvatures $C_f^{(i)}$'s are uniform).

Theorem 2. *Consider the same notation as in Theorem 1. Assume that at each iterate $\boldsymbol{\alpha}^{(k)}$, BCFW with gap sampling (Algorithm 4) has access to the exact values of the block gaps. Then, at each iteration, it holds that $\mathbf{E}\Big[f(\boldsymbol{\alpha}^{(k)})\Big] - f(\boldsymbol{\alpha}^*) \leq \frac{2n}{k+2n}\Big(C_f^{\otimes}\chi^{\otimes} + h_0\Big)$ where the constant $\chi^{\otimes}$ is an upper bound on $\mathbf{E}\Big[\frac{\chi(C_f^{(:)})}{\chi(\boldsymbol{g}_:(\boldsymbol{\alpha}^{(k)}))^3}\Big]$. The* non-uniformity measure *$\chi(\boldsymbol{x})$ of a vector $\boldsymbol{x} \in \mathbb{R}_+^n$ is defined as $\chi(\boldsymbol{x}) := \sqrt{1 + n^2 \operatorname{Var}\big[\boldsymbol{p}\big]}$ where $\boldsymbol{p} := \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_1}$ is the probability vector obtained by normalizing $\boldsymbol{x}$.*

Proof. See Appendix A.3. □

Comparison with uniform sampling. We now compare the rates obtained by Theorem 2 for BCFW with gap sampling and by Theorem 1 for BCFW

Algorithm 4 Block-coordinate Frank-Wolfe (BCFW) algorithm with gap sampling for structured SVM

Let $\boldsymbol{w}^{(0)} := \boldsymbol{w}_i^{(0)} := \mathbf{0}$; $\ell^{(0)} := \ell_i^{(0)} := 0$; $g_i^{(0)} := +\infty$;
$k_i := 0$ *// the last time g_i was computed*
for $k := 0, \ldots, \infty$ **do**
  Pick i at random with probability $\propto g_i^{(k_i)}$
  Solve $\boldsymbol{y}_i^* := \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}_i} H_i(\boldsymbol{y}; \boldsymbol{w}^{(k)})$
  Let $k_i := k$
  Let $\boldsymbol{w}_s := \frac{1}{\lambda n} \boldsymbol{\psi}_i(\boldsymbol{y}_i^*)$ and $\ell_s := \frac{1}{n} L_i(\boldsymbol{y}_i^*)$
  Let $g_i^{(k_i)} := \lambda(\boldsymbol{w}_i^{(k)} - \boldsymbol{w}_s)^\top \boldsymbol{w}^{(k)} - \ell_i^{(k)} + \ell_s$
  Let $\gamma := \frac{g_i^{(k_i)}}{\lambda \|\boldsymbol{w}_i^{(k)} - \boldsymbol{w}_s\|^2}$ and clip to $[0, 1]$
  Update $\boldsymbol{w}_i^{(k+1)} := (1-\gamma)\boldsymbol{w}_i^{(k)} + \gamma\, \boldsymbol{w}_s$
    and $\ell_i^{(k+1)} := (1-\gamma)\ell_i^{(k)} + \gamma\, \ell_s$
  Update $\boldsymbol{w}^{(k+1)} := \boldsymbol{w}^{(k)} + \boldsymbol{w}_i^{(k+1)} - \boldsymbol{w}_i^{(k)}$
    and $\ell^{(k+1)} := \ell^{(k)} + \ell_i^{(k+1)} - \ell_i^{(k)}$
  if update global gap **then**
    for $i := 1, \ldots, n$ **do**
      Let $k_i := k + 1$
      Solve $\boldsymbol{y}_i^* := \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}_i} H_i(\boldsymbol{y}; \boldsymbol{w}^{(k_i)})$
      Let $\boldsymbol{w}_s := \frac{1}{\lambda n} \boldsymbol{\psi}_i(\boldsymbol{y}_i^*)$ and $\ell_s := \frac{1}{n} L_i(\boldsymbol{y}_i^*)$
      $g_i^{(k_i)} := \lambda(\boldsymbol{w}_i^{(k_i)} - \boldsymbol{w}_s)^\top \boldsymbol{w}^{(k_i)} - \ell_i^{(k_i)} + \ell_s$
    end for
  end if
end for

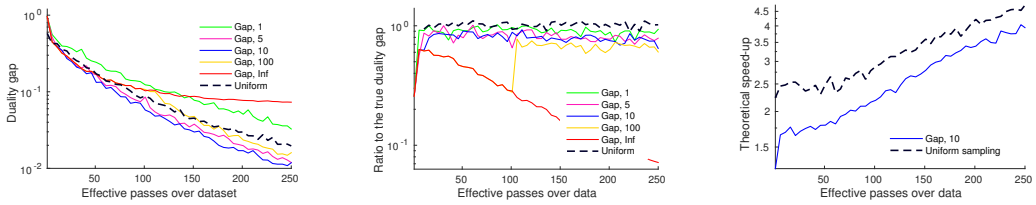
with uniform sampling. The only difference is in the constants: Theorem 2 has $C_f^{\otimes} \chi^{\otimes}$ and Theorem 1 has $C_f^{\otimes}$.

Recall that, by definition,

$$\chi^{\otimes} = \max_k \mathbf{E}\left[\frac{\chi(C_f^{(:)})}{\chi(\boldsymbol{g}_:(\boldsymbol{\alpha}^{(k)}))^3}\right].$$

In the best case for gap sampling, the curvature constants are uniform, $\chi(C_f^{(:)}) = 1$, and the gaps are nonuniform $\chi(\boldsymbol{g}_:(\boldsymbol{\alpha}^{(k)})) \approx \sqrt{n}$. Thus, $\chi^{\otimes} \approx \frac{1}{n\sqrt{n}}$.

In the worst case for gap sampling, the curvature constants are very non-uniform, $\chi(C_f^{(:)}) \approx \sqrt{n}$. The constant for gap sampling is still better if the gaps are non-uniform enough, i.e., $\chi(\boldsymbol{g}_:(\boldsymbol{\alpha}^{(k)})) \geq n^{\frac{1}{6}}$.



(a) Convergence plots **(b)** Quality of gap estimates **(c)** Theoretical improvement

Fig. 6.1.: Plot (a) shows exploitation/staleness trade-off for the gap sampling approach. We report the duality gap against the number of effective passes over the data for uniform sampling and for gap sampling with the different frequencies of batch passes updating the gap estimates (every pass over data, every 5, 10, 100 passes, no batch updates). Plot (b) shows the quality of heuristic gap estimates obtained by the same methods. We report the ratio of the heuristic gap estimate to the true gap value. Plot (c) shows the factor of improvement of exact gap sampling predicted by Theorem 2 for real gaps appearing during a run of BCFW with either uniform or gap sampling.

We note that to design a sampling scheme that always dominates uniform sampling (in terms of bounds at least), we would need to include the $C_f^{(i)}$'s in the sampling scheme (as was essentially done by Csiba *et al.*, [2015] for SDCA). Unfortunately, computing good estimates for $C_f^{(i)}$'s is too expensive for structured SVM, thus motivating our simpler yet practically efficient scheme.

Adaptive procedure. In addition note that our gap-sampling procedure is *adaptive*, meaning that the criterion for choosing an object to optimize changes during the optimization process. Our adaptive approach differs from more standard techniques that sample proportional to the Lipschitz constants, as e.g., in Nesterov, [2012]. In Appendix A.2, we illustrate the advantage of this property by constructing an example where the convergence of gap sampling can be shown *tightly* to be n times faster than when using Lipschitz sampling.

Exploitation versus staleness trade-off. In practice, having access to the exact block gaps is intractable because it requires a full pass over the dataset after every block update. However, we have access to the estimates of the block gaps computed from past oracle calls on each block. Notice that such estimates are outdated, i.e., might be quite far from the current values of the block gaps. We call this effect "staleness". One way to compensate staleness

is to refresh the block gaps by doing a full gap computation (a pass over the dataset) after several block-coordinate passes. These gap computations were often already done during the optimization process, e.g., to monitor convergence.

We demonstrate the exploitation/staleness trade-off in our exploratory experiment reported in Figure 6.1. On the OCR dataset [Taskar *et al.*, 2003], we run the gap sampling algorithm with a gap computation pass after 1, 5, 10 and 100 block-coordinate passes (Gap 1, Gap 5, Gap 10, Gap 100) and without any gap computation passes (Gap Inf). As a baseline, we use BCFW with uniform sampling (Uniform). Figure 6.1a reports the duality gap after each number of effective passes over the data (an effective pass consists in n calls to the max oracle). Figure 6.1b shows the ratio of the exact value of the duality gap to the heuristic gap estimate defined as the sum of the current gap estimates. We observe that when the gap computation is never run, the gap becomes significantly underestimated and the algorithm does not converge. On another extreme, when performing the gap computation after each pass of BCFW, the algorithm wastes too many computations and converges slowly. Between the two extremes, the method is not very sensitive to the parameter (we have tried 5, 10, 20, 50) allowing us to always use the value of 10.

Comparing adaptive methods to BCFW with uniform sampling, we observe a faster convergence. Figure 6.1c reports the improvement of gap sampling at each iteration w.r.t. uniform sampling that is predicted by Theorem 2. Specifically, we report the quantity $\chi(g_{:}(\boldsymbol{\alpha}^{(k)}))^3 / \chi(C_f^{(:)})$ with the block gaps estimated at the runs of BCFW with both uniform and gap sampling schemes. To estimate the curvature constants $C_f^{(i)}$, we use the upper bounds proposed by Lacoste-Julien *et al.*, [2013, App. A]: $\frac{4R_i^2}{\lambda n^2}$ where $R_i := \max_{\boldsymbol{y} \in \mathcal{Y}_i} \|\boldsymbol{\psi}_i(\boldsymbol{y})\|_2$. We approximate R_i by picking the largest value $\|\boldsymbol{\psi}_i(\boldsymbol{y})\|_2$ corresponding to a labeling $\boldsymbol{y}$ observed within the run of BCFW.

Related work. Non-uniform sampling schemes have been used over the last few years to improve the convergence rates of well known randomized algorithms Nesterov, 2012; Needell *et al.*, 2014; Zhao and Zhang, 2015. Most of these approaches use the Lipschitz constants of the gradients to sample more often functions for which gradient changes quickly. This approach

has two main drawbacks. First, Lipschitz constants are often unknown and heuristics are needed to estimate them. Second, such schemes are not adaptive to the current progress of the algorithm. At the time of publication, the only other approach that uses an *adaptive* sampling scheme to guide the optimization with convergence guarantees is the one from Csiba *et al.*, [2015], in the context of the stochastic dual coordinate ascent (SDCA) algorithm. Since then, several work has built on our adaptive sampling approach. Among others, [D. Perekrestenko and Jaggi, 2017; C. Dünner and Jaggi, 2017; R. Le Priol and Lacoste-Julien, 2018] have all studied gap sampling schemes in the more general context of the SDCA algorithm. Finally, a cyclic version of BCFW has been analyzed by Beck *et al.*, [2015] while Wang *et al.*, [2014] analyzed its mini-batch form.

6.3.2 Pairwise and away steps

Motivation. In the batch setting, the convergence rate of the Frank-Wolfe algorithm is known to be sublinear when the solution is on the boundary [Wolfe, 1970], as is the case for SSVM. Several modifications have been proposed in the literature to address this issue. All these methods replace (or complement) the FW step with a step of another type: pairwise step [Mitchell *et al.*, 1974], away step [Wolfe, 1970], fully-corrective step [Holloway, 1974] (see Lacoste-Julien and Jaggi, [2015b] for a recent review and the proof that all these methods have a linear rate on the objective (6.3) despite not being strongly convex). A common feature of these methods is the ability to remove elements of the active set (support vectors in the case of SSVM) in order to reach the boundary, unlike FW which oscillates while never completely reaching the boundary. As we expect the solution of SSVM to be sparse, these variants seem natural in our setting. In the rest of this section, we present in details the pairwise steps in the block-coordinate setting. The away-step version derivation relies on similar principles and is given in Algorithm 5.

Pairwise steps. A (block) pairwise step consists in *removing* mass from the *away corner* on block i and transferring it to the *FW corner* obtained by the max oracle (6.5). The away corner is the element of the active set $\mathcal{S}_i := \{\boldsymbol{y} \in \mathcal{Y}_i \mid \alpha_i(\boldsymbol{y}) > 0\} \subseteq \mathcal{Y}_i$ worst aligned with the current descent direction, which can be found by solving $\boldsymbol{y}_i^a := \operatorname{argmin}_{\boldsymbol{y} \in \mathcal{S}_i} H_i(\boldsymbol{y}; \boldsymbol{w})$. This does not require solving a combinatorial optimization problem because the

Algorithm 5 Block-coordinate away-step Frank-Wolfe (BCAFW) algorithm for structured SVM

Let $\boldsymbol{w}^{(0)} := \boldsymbol{w}_i^{(0)} := \mathbf{0}$; $\ell^{(0)} := \ell_i^{(0)} := 0$;
$\mathcal{S}_i^{(0)} := \{\boldsymbol{y}_i\}$; *// active sets*
$\alpha_i^{(0)}(\boldsymbol{y}) := 0$, $\boldsymbol{y} \neq \boldsymbol{y}_i$; $\alpha_i^{(0)}(\boldsymbol{y}_i) := 1$
for $k := 0, \dots, \infty$ **do**
  Pick i at random in $\{1, \dots, n\}$
  Solve $\boldsymbol{y}_i^* := \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}_i} H_i(\boldsymbol{y}; \boldsymbol{w}^{(k)})$ *// FW corner*
  Let $\boldsymbol{w}_{\boldsymbol{s}} := \frac{1}{\lambda n}\boldsymbol{\psi}_i(\boldsymbol{y}_i^*)$ and $\ell_{\boldsymbol{s}} := \frac{1}{n}L_i(\boldsymbol{y}_i^*)$
  Solve $\boldsymbol{y}_i^a := \operatorname*{argmin}_{\boldsymbol{y} \in \mathcal{S}_i^{(k)}} H_i(\boldsymbol{y}; \boldsymbol{w}^{(k)})$ *// away corner*
  Let $\boldsymbol{w}_{\boldsymbol{a}} := \frac{1}{\lambda n}\boldsymbol{\psi}_i(\boldsymbol{y}_i^a)$ and $\ell_{\boldsymbol{a}} := \frac{1}{n}L_i(\boldsymbol{y}_i^a)$
  Let $g_i^{FW} := \lambda(\boldsymbol{w}_i^{(k)} - \boldsymbol{w}_{\boldsymbol{s}})^\mathsf{T}\boldsymbol{w}^{(k)} - \ell_i^{(k)} + \ell_{\boldsymbol{s}}$
  Let $g_i^A := \lambda(\boldsymbol{w}_{\boldsymbol{a}} - \boldsymbol{w}_i^{(k)})^\mathsf{T}\boldsymbol{w}^{(k)} + \ell_i^{(k)} - \ell_{\boldsymbol{a}}$
  if $g_i^{FW} > g_i^A$ **then** *// FW step*
    Let $\gamma := \frac{g_i^{FW}}{\lambda\|\boldsymbol{w}_i^{(k)} - \boldsymbol{w}_{\boldsymbol{s}}\|^2}$ and clip to $[0, 1]$
    Update $\mathcal{S}_i^{(k+1)} := \mathcal{S}_i^{(k)} \cup \{\boldsymbol{y}_i^*\}$
      and $\alpha_i^{(k+1)}(\boldsymbol{y}) := (1-\gamma)\alpha_i^{(k)}(\boldsymbol{y})$
      and $\alpha_i^{(k+1)}(\boldsymbol{y}_i^*) := \alpha_i^{(k+1)}(\boldsymbol{y}_i^*) + \gamma$
    Set $\mathcal{S}_i^{(k+1)} := \{\boldsymbol{y}_i^*\}$ if $\gamma = 1$
  else *// away step*
    Let $\gamma := \frac{g_i^A}{\lambda\|\boldsymbol{w}_i^{(k)} - \boldsymbol{w}_{\boldsymbol{a}}\|^2}$ and clip to $[0, \frac{\alpha_i(\boldsymbol{y}_i^a)}{1-\alpha_i(\boldsymbol{y}_i^a)}]$
    Update $\alpha_i^{(k+1)}(\boldsymbol{y}) := (1+\gamma)\alpha_i^{(k)}(\boldsymbol{y})$
      and $\alpha_i^{(k+1)}(\boldsymbol{y}_i^a) := \alpha_i^{(k+1)}(\boldsymbol{y}_i^a) - \gamma$
      and $\mathcal{S}_i^{(k+1)} := \mathcal{S}_i^{(k)} \setminus \{\boldsymbol{y}_i^a\}$ if $\alpha_i^{(k+1)}(\boldsymbol{y}_i^a) = 0$
  end if
  Update $\boldsymbol{w}_i^{(k+1)} := \boldsymbol{w}_i^{(k)} + \gamma\,\boldsymbol{w}_{\boldsymbol{d}}$
    and $\ell_i^{(k+1)} := \ell_i^{(k)} + \gamma\,\ell_{\boldsymbol{d}}$
  Update $\boldsymbol{w}^{(k+1)} := \boldsymbol{w}^{(k)} + \boldsymbol{w}_i^{(k+1)} - \boldsymbol{w}_i^{(k)}$
    and $\ell^{(k+1)} := \ell^{(k)} + \ell_i^{(k+1)} - \ell_i^{(k)}$
end for

size of the active set is typically small, e.g., bounded by the number of iterations performed on the block i. Analogously to the case of BCFW, the optimal step size γ for the pairwise step can be computed explicitly by clipping $\frac{\lambda(\boldsymbol{w}_{\boldsymbol{a}} - \boldsymbol{w}_{\boldsymbol{s}})^\mathsf{T}\boldsymbol{w}^{(k)} + \ell_{\boldsymbol{s}} - \ell_{\boldsymbol{a}}}{\lambda\|\boldsymbol{w}_{\boldsymbol{a}} - \boldsymbol{w}_{\boldsymbol{s}}\|^2}$ to the segment $[0, \alpha_i^{(k)}(\boldsymbol{y}_i^a)]$ where the upper bound $\alpha_i^{(k)}(\boldsymbol{y}_i^a)$ corresponds to the mass of the away corner before the step and the quantities $\boldsymbol{w}_{\boldsymbol{s}} := \frac{1}{\lambda n}\boldsymbol{\psi}_i(\boldsymbol{y}_i^*)$, $\ell_{\boldsymbol{s}} := \frac{1}{n}L_i(\boldsymbol{y}_i^*)$ and $\boldsymbol{w}_{\boldsymbol{a}} := \frac{1}{\lambda n}\boldsymbol{\psi}_i(\boldsymbol{y}_i^a)$, $\ell_{\boldsymbol{a}} := \frac{1}{n}L_i(\boldsymbol{y}_i^a)$ represent the FW and away corners. Algorithm 6 summarizes the block-coordinate pairwise Frank-Wolfe (BCPFW) algorithm.

Algorithm 6 Block-coordinate pairwise Frank-Wolfe (BCPFW) algorithm for structured SVM

Let $\boldsymbol{w}^{(0)} := \boldsymbol{w}_i^{(0)} := \mathbf{0}$; $\ell^{(0)} := \ell_i^{(0)} := 0$;
$\mathcal{S}_i^{(0)} := \{\boldsymbol{y}_i\}$; *// active sets*
$\alpha_i^{(0)}(\boldsymbol{y}) := 0$, $\boldsymbol{y} \neq \boldsymbol{y}_i$; $\alpha_i^{(0)}(\boldsymbol{y}_i) := 1$
for $k := 0, \dots, \infty$ **do**
  Pick i at random in $\{1, \dots, n\}$
  Solve $\boldsymbol{y}_i^* := \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}_i} H_i(\boldsymbol{y}; \boldsymbol{w}^{(k)})$ *// FW corner*
  Let $\boldsymbol{w}_{\boldsymbol{s}} := \frac{1}{\lambda n} \boldsymbol{\psi}_i(\boldsymbol{y}_i^*)$ and $\ell_{\boldsymbol{s}} := \frac{1}{n} L_i(\boldsymbol{y}_i^*)$
  Solve $\boldsymbol{y}_i^a := \operatorname*{argmin}_{\boldsymbol{y} \in \mathcal{S}_i^{(k)}} H_i(\boldsymbol{y}; \boldsymbol{w}^{(k)})$ *// away corner*
  Let $\boldsymbol{w}_{\boldsymbol{a}} := \frac{1}{\lambda n} \boldsymbol{\psi}_i(\boldsymbol{y}_i^a)$ and $\ell_{\boldsymbol{a}} := \frac{1}{n} L_i(\boldsymbol{y}_i^a)$
  Let $\boldsymbol{w}_{\boldsymbol{d}} := \boldsymbol{w}_{\boldsymbol{s}} - \boldsymbol{w}_{\boldsymbol{a}}$ and $\ell_{\boldsymbol{d}} = \ell_{\boldsymbol{s}} - \ell_{\boldsymbol{a}}$
  Let $\gamma := \frac{-\lambda \boldsymbol{w}_{\boldsymbol{d}}^\mathsf{T} \boldsymbol{w}^{(k)} + \ell_{\boldsymbol{d}}}{\lambda \|\boldsymbol{w}_{\boldsymbol{d}}\|^2}$ and clip to $[0, \alpha_i^{(k)}(\boldsymbol{y}_i^a)]$
  Update $\boldsymbol{w}_i^{(k+1)} := \boldsymbol{w}_i^{(k)} + \gamma\, \boldsymbol{w}_{\boldsymbol{d}}$
    and $\ell_i^{(k+1)} := \ell_i^{(k)} + \gamma\, \ell_{\boldsymbol{d}}$
  Update $\boldsymbol{w}^{(k+1)} := \boldsymbol{w}^{(k)} + \boldsymbol{w}_i^{(k+1)} - \boldsymbol{w}_i^{(k)}$
    and $\ell^{(k+1)} := \ell^{(k)} + \ell_i^{(k+1)} - \ell_i^{(k)}$
  Update $\mathcal{S}_i^{(k+1)} := \mathcal{S}_i^{(k)} \cup \{\boldsymbol{y}_i^*\}$
    and $\alpha_i^{(k+1)}(\boldsymbol{y}_i^a) := \alpha_i^{(k)}(\boldsymbol{y}_i^a) - \gamma$
    and $\alpha_i^{(k+1)}(\boldsymbol{y}_i^*) := \alpha_i^{(k)}(\boldsymbol{y}_i^*) + \gamma$
  if $\gamma = \alpha_i^{(k)}(\boldsymbol{y}_i^a)$ **then**
    Set $\mathcal{S}_i^{(k+1)} := \mathcal{S}_i^{(k+1)} \setminus \{\boldsymbol{y}_i^a\}$ *// drop step*
  end if
end for

In contrast to BCFW, the steps of BCPFW cannot be expressed in terms of the primal variables $\boldsymbol{w}$ only, thus it is required to explicitly store the dual variables $\boldsymbol{\alpha}_i$. Storing the dual variables is feasible, because they are extremely sparse, but still can lead to computational overheads caused by the maintenance of the data structure.

The standard convergence analysis for pairwise and away-step FW cannot be easily extended to BCFW.

We show the geometric decrease of the objective in Theorem 3 of Appendix A.4 only when *no* block would have a drop step (a.k.a. 'bad step'); a condition that cannot be easily analyzed due to the randomization of the algorithm. We believe that novel proof techniques are required here, even though we did observe empirically a linear convergence rate when λ is big enough.

Related work. Ñanculef *et al.*, [2014, Algorithm 4] used the pairwise FW algorithm on the dual of binary SVM (in batch mode, however). It is related to classical working set algorithms, such as the SMO algorithm used to train SVMs Platt, 1999, also already applied on SSVMs in Taskar, [2004, Ch. 6]. Franc, [2014] recently proposed a version of pairwise FW for the block-coordinate setting. Their SDA-WSS2 algorithm uses a different criterion for choosing the away corner than BCPFW: instead of minimizing H_i over the active set $\mathcal{S}_i$, they compute the improvement for all possible away corners and pick the best one. Their FASOLE algorithm also contains a version of gap sampling in the form of variable shrinking: if a block gap becomes small enough, the block is not visited again, until all the counters are reset. Posterior to our work, Garber and Meshi, [2016] make further assumptions on the structure of the constraint set which allow them to propose a more efficient way to perform away-steps without the need to explicitly store the convex decomposition of the iterate, hence reducing the memory complexity of the algorithm and giving significant practical speed-ups.

6.3.3 Caching

Motivation. At each step, the BCFW and BCPFW algorithms call the max oracle to find the Frank-Wolfe corner. In cases where the max oracle is expensive, this step becomes a computational bottleneck. A natural idea to overcome this problem consists in using a "cheaper oracle" most of the time hoping that the resulting corner would be good enough. Caching the results of the max oracle implements this idea by reusing the previous calls of the max oracle to store potentially promising corners.

Caching. The main principle of caching consists in maintaining a working set $\mathcal{C}_i \subset \mathcal{Y}_i$ of labelings/corners for each block i, where $|\mathcal{C}_i| \ll |\mathcal{Y}_i|$. A *cache oracle* obtains the *cache corner* defined as a corner from the working set best aligned with the descent direction, i.e., $\boldsymbol{y}_i^c := \operatorname{argmax}_{\boldsymbol{y} \in \mathcal{C}_i} H_i(\boldsymbol{y}; \boldsymbol{w})$. If the obtained cache corner passes a *cache hit criterion*, i.e., there is a *cache hit*, we do a Frank-Wolfe (or pairwise) step based on the cache corner. A step defined this way is equivalent to the corresponding step on the convex hull of the working set, which is a subset of the block domain $\mathcal{Y}_i$. If a cache hit criterion is not satisfied, i.e., there is a *cache miss*, we call the (possibly expensive) max

Algorithm 7 Block-coordinate Frank-Wolfe (BCFW) algorithm with cache for structured SVM

Let $\boldsymbol{w}^{(0)} := \boldsymbol{w}_i^{(0)} := \mathbf{0}$; $\ell^{(0)} := \ell_i^{(0)} := 0$; $\mathcal{C}_i := \{\boldsymbol{y}_i\}$;
$g^{(0)} := g_i^{(0)} = +\infty$
$k_0 := k_i := 0$; *// the last time g / g_i was computed*
for $k := 0, \dots, \infty$ **do**
  Pick i at random in $\{1, \dots, n\}$ *// either uniform or with probability $\propto g_i^{(k_i)}$ for gap sampling*
  Solve $\boldsymbol{y}_i^c := \operatorname{argmax}_{\boldsymbol{y} \in \mathcal{C}_i} \; H_i(\boldsymbol{y}; \boldsymbol{w})$ *// cache corner*
  Let $\boldsymbol{w}_{\boldsymbol{c}} := \frac{\boldsymbol{\psi}_i(\boldsymbol{y}_i^c)}{\lambda n}$ and $\ell_{\boldsymbol{c}} := \frac{1}{n} L_i(\boldsymbol{y}_i^c)$
  Let $\hat{g}_i^{(k)} := \lambda(\boldsymbol{w}_i^{(k)} - \boldsymbol{w}_{\boldsymbol{c}})^\mathsf{T} \boldsymbol{w}^{(k)} - \ell_i^{(k)} + \ell_{\boldsymbol{c}}$
  if $\hat{g}_i^{(k)} \geq \max(F g_i^{(k_i)}, \frac{\nu}{n} g^{(k_0)})$ **then** *// cache hit*
    $\boldsymbol{w}_{\boldsymbol{s}} := \boldsymbol{w}_{\boldsymbol{c}}$, $\ell_{\boldsymbol{s}} := \ell_{\boldsymbol{c}}$, $\hat{g}_i := \hat{g}_i^{(k)}$
  else *// cache miss*
    Solve $\boldsymbol{y}_i^* := \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}_i} \; H_i(\boldsymbol{y}; \boldsymbol{w}^{(k)})$ *// FW corner*
    Let $\boldsymbol{w}_{\boldsymbol{s}} := \frac{1}{\lambda n} \boldsymbol{\psi}_i(\boldsymbol{y}_i^*)$ and $\ell_{\boldsymbol{s}} := \frac{1}{n} L_i(\boldsymbol{y}_i^*)$
    Let $g_i^{(k)} := \lambda(\boldsymbol{w}_i^{(k)} - \boldsymbol{w}_{\boldsymbol{s}})^\mathsf{T} \boldsymbol{w}^{(k)} - \ell_i^{(k)} + \ell_{\boldsymbol{s}}$
    Set $k_i := k$, $\hat{g}_i := g_i^{(k)}$
    Update $\mathcal{C}_i := \mathcal{C}_i \cup \{\boldsymbol{y}_i^*\}$
  end if
  Let $\gamma := \frac{\hat{g}_i}{\lambda \|\boldsymbol{w}_i^{(k)} - \boldsymbol{w}_{\boldsymbol{s}}\|^2}$ and clip to $[0, 1]$
  Update $\boldsymbol{w}_i^{(k+1)} := (1-\gamma) \boldsymbol{w}_i^{(k)} + \gamma \, \boldsymbol{w}_{\boldsymbol{s}}$
    and $\ell_i^{(k+1)} := (1-\gamma) \ell_i^{(k)} + \gamma \, \ell_{\boldsymbol{s}}$
  Update $\boldsymbol{w}^{(k+1)} := \boldsymbol{w}^{(k)} + \boldsymbol{w}_i^{(k+1)} - \boldsymbol{w}_i^{(k)}$
    and $\ell^{(k+1)} := \ell^{(k)} + \ell_i^{(k+1)} - \ell_i^{(k)}$
  if update global gap **then**
    Let $g^{(k_0)} := 0$, $k_0 := k + 1$
    for $i := 1, \dots, n$ **do**
      Solve $\boldsymbol{y}_i^* := \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}_i} \; H_i(\boldsymbol{y}; \boldsymbol{w}^{(k_0)})$
      Let $\boldsymbol{w}_{\boldsymbol{s}} := \frac{1}{\lambda n} \boldsymbol{\psi}_i(\boldsymbol{y}_i^*)$ and $\ell_{\boldsymbol{s}} := \frac{1}{n} L_i(\boldsymbol{y}_i^*)$
      $g^{(k_0)} \mathrel{+}= \lambda(\boldsymbol{w}_i^{(k_0)} - \boldsymbol{w}_{\boldsymbol{s}})^\mathsf{T} \boldsymbol{w}^{(k_0)} - \ell_i^{(k_0)} + \ell_{\boldsymbol{s}}$
      Set $k_i := k_0$
    end for
  end if
end for

oracle to obtain a Frank-Wolfe corner over the full domain $\mathcal{Y}_i$. Algorithm 7 summarizes the BCFW method with caching.

Note that, in the case of BCPFW, the working set $\mathcal{C}_i$ is closely related to the active set $\mathcal{S}_i$. On the implementation side, we maintain both sets in the same data structure and keep $\mathcal{S}_i \subseteq \mathcal{C}_i$.

Cache hit criterion. An important part of a caching scheme is the criterion deciding whether the cache look up is sufficient or the max oracle needs to be called. Intuitively, we want to use the cache whenever it allows optimization to make large enough progress. We use as measure of potential progress the inner product between the candidate direction and the negative gradient (which would give the block gap g_i (6.7) if the FW corner is used). For a cache step, it gives $\hat{g}_i^{(k)} := \lambda(\boldsymbol{w}_i^{(k)} - \boldsymbol{w}_{\boldsymbol{c}})^\top \boldsymbol{w}^{(k)} - \ell_i^{(k)} + \ell_{\boldsymbol{c}}$, which is defined by quantities $\boldsymbol{w}_{\boldsymbol{c}} = \frac{\psi_i(\boldsymbol{y}_i^c)}{\lambda n}$, $\ell_{\boldsymbol{c}} = \frac{1}{n} L_i(\boldsymbol{y}_i^c)$ similar to the ones defining the block gap. The quantity $\hat{g}_i^{(k)}$ is then compared to a *cache hit threshold* defined as $\max(F g_i^{(k_i)}, \frac{\nu}{n} g^{(k_0)})$ where k_i identifies the iteration when the max oracle was last called for the block i, k_0 is the index of the iteration when the full batch gap was computed, $F > 0$ and $\nu > 0$ are cache parameters.

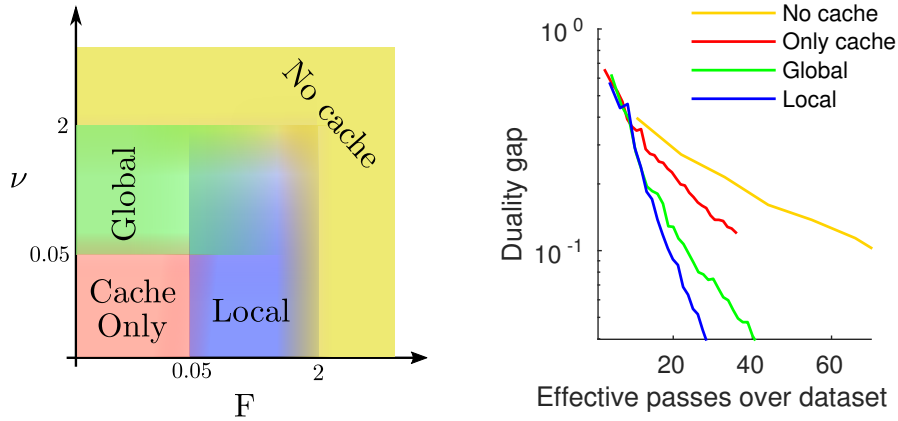
The following theorem gives a safety convergence result for BCFW with caching.

Theorem 4. *Consider the same notation as in Theorem 1. Let* $\tilde{\nu} := \frac{1}{n}\nu \leq 1$. *The iterate* $\boldsymbol{\alpha}^{(k)}$ *of Algorithm 7 satisfies* $\mathbf{E}\Big[f(\boldsymbol{\alpha}^{(k)})\Big] - f(\boldsymbol{\alpha}^*) \leq \frac{2n}{\tilde{\nu}k+2n}\Big(\frac{1}{\tilde{\nu}} C_f^{\otimes} + h_0\Big)$ *for* $k \geq 0$.

Proof. See Appendix A.5. □

Note that the convergence rate of Theorem 4 differs from the original rate of BCFW (Theorem 1) by the constant $\tilde{\nu}$. If $\tilde{\nu}$ equals one the rate is the same, but the criterion effectively prohibits cache hits. If $\tilde{\nu} < 1$ then the convergence is slower, meaning that the method with cache needs more iterations to converge, but the oracles calls might be cheaper because of the cache hits.

Effect of F and ν. The parameter ν controls the global component and acts as a safety parameter to ensure convergence (Theorem 4). The parameter F controls, instead, the local (block-dependent) component of the criterion. Figure 6.2 illustrates the effect of the parameters on OCR dataset [Taskar *et al.*, 2003] and motivates their choice. At one extreme, if either F or ν are too large the cache is almost never hit. At another extreme, if both values are small the cache is hit almost always, thus the method almost stops calling the oracle and does not converge. Between the two extremes,



(a) Regimes of the cache **(b)** Convergence plots

Fig. 6.2.: Plot (a) illustrates different regimes induced by the cache parameters F and ν. Plot (b) shows the evolution of the duality gap within BCFW with gap sampling and with cache parameters in different regimes.

one of the components usually dominates. We observe empirically that the regime with the local component dominating leads to faster convergence. Our experiments show that the method is not very sensitive to the choice of the parameters, so, in what follows, we use values $F = 0.25$ and $\nu = 0.01$.

Related work. In the context of SSVM, the idea of caching was successfully applied to the cutting plane methods by Joachims *et al.*, [2009], and, recently, to BCFW by Shah *et al.*, [2015]. In contrast to Shah *et al.*, [2015], our method chooses whether to call the oracle or to use the cache in an adaptive way by looking at the gap estimates of the current blocks. In the extreme case, when just one block is hard and requires computation and all the rest are easy, our method would be able to call an oracle on the hard block and to use the cache everywhere else. This will result to n times less oracle calls, compared to their strategy. After our work, others efforts have been made to alleviate the cost of calling the linear minimization oracle. For example, Braun *et al.*, [2017] propose to use a faster separation oracle which also uses the idea of caching. Kerdreux *et al.*, [2018] show that it is possible to solve a linear minimization oracle only over a small subset of the original domain chosen *randomly* at each iteration of the algorithm.

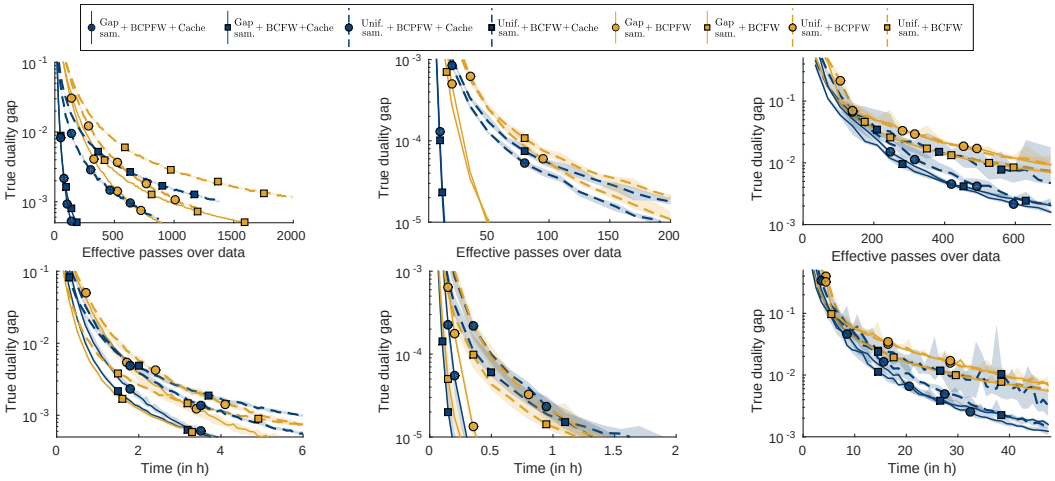
6.4 Experiments

In this section, we evaluate the three modifications of BCFW presented in Section 6.3. We compare 8 methods obtained by all the combinations of three binary dimensions: gap-based vs. uniform sampling of objects, BCFW vs. BCPFW, caching oracle calls vs. no caching.

Datasets. We evaluate our methods on four datasets for different structured prediction tasks: OCR [Taskar *et al.*, 2003] for handwritten character recognition, CoNLL [Sang and Buchholz, 2000] for text chunking, HorseSeg [Kolesnikov *et al.*, 2014] for binary image segmentation and LSP [Johnson and Everingham, 2010] for pose estimation. The models for OCR and CoNLL were provided by Lacoste-Julien *et al.*, [2013]. We build our model based on the one by Kolesnikov *et al.*, [2014] for HorseSeg, and the one by Chen and Yuille, [2014] for LSP. For OCR and CoNLL, the max oracle consists of the Viterbi algorithm [Viterbi, 1967]; for HorseSeg – in graph cut [Boykov and Kolmogorov, 2004], for LSP – in belief propagation on a tree with messages passed by a generalized distance transform [Felzenszwalb and Huttenlocher, 2005]. Note that the oracles of HorseSeg and LSP require positivity constraints on a subset of the weights in order to be tractable. The BCFW algorithm with positivity constraints is derived in Appendix A.6. We provide a detailed description of the datasets in Appendix A.7 with a summary in Table 6.1.

The problems included in our experimental study vary in the number of objects n (from 100 to $25{,}000$), in the number of features d (from 10^2 to 10^6), and in the computational cost of the max oracle (from 10^{-4} to 2 seconds).

Results. We report the results of each method on 6 datasets (including 3 sizes of HorseSeg) for three values of the regularization parameter λ: the value leading to the best test performance, a smaller and a larger value. For each setup, we report the duality gap against both number of oracle calls and elapsed time. We run each method 5 times with different random seeds influencing the order of sampled objects and report the median (bold line), minimum and maximum values (shaded region). We summarize the results in Figure 6.3 and report the rest in Figures 6.4 and 6.5.

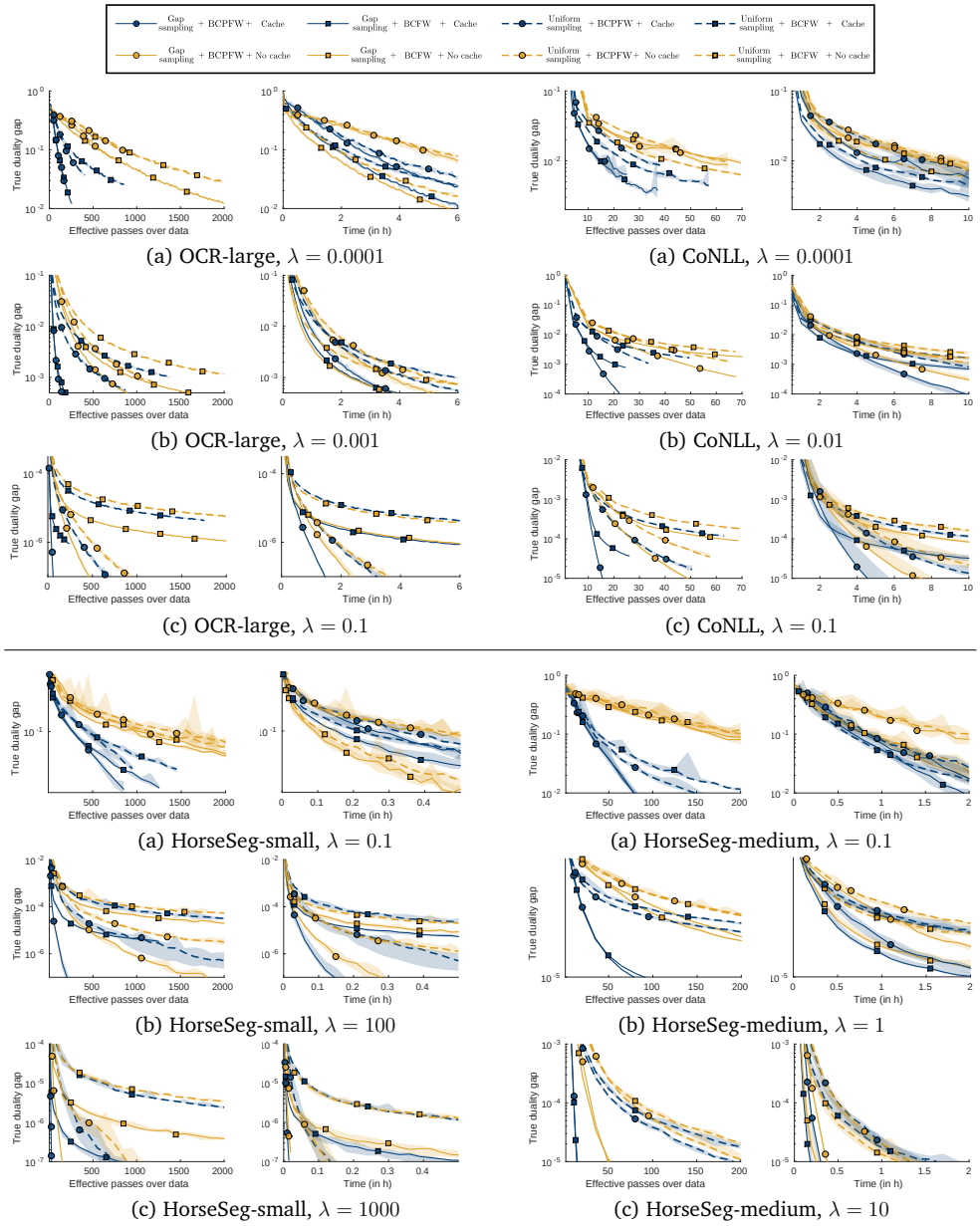


(a) OCR-large $\lambda = 0.001$ **(b)** HorseSeg-medium $\lambda = 10$ **(c)** LSP-small $\lambda = 100$

Fig. 6.3.: Summary of the results of Section 6.4: the duality gap against the number of effective passes over data (top) and time (bottom).

First, we observe that, aligned with our theoretical results, gap sampling always leads to faster convergence (both in terms of time and the number of effective passes). The effect is stronger when n is large (Figure 6.3b). Second, caching always helps in terms of number of effective passes, but an overhead caused by maintaining the cache is significant when the max oracle is fast (Figure 6.3a). In the case of expensive oracle (Figure 6.3c), the cache overhead is negligible. Third, the pairwise steps (BCPFW) lead to an improvement to get smaller values of duality gaps. The effect is stronger when the problem is more strongly convex, i.e., λ is bigger. However, maintaining the active sets results in computational overheads, which sometimes are significant. Note that the overhead of cache and active sets are shared, because they are maintained in the same data structure. Using a cache also greatly limits the memory requirements of BCPFW, because, when the cache is hit, the active set is guaranteed not to grow.

Recommendation. For off-the-shelf usage, we recommend to use the BCPFW + gap sampling + cache method when oracle calls are expensive, and the BCFW + gap sampling method when oracle calls are cheap.



(a) OCR-large, $\lambda = 0.0001$ (a) CoNLL, $\lambda = 0.0001$

(b) OCR-large, $\lambda = 0.001$ (b) CoNLL, $\lambda = 0.01$

(c) OCR-large, $\lambda = 0.1$ (c) CoNLL, $\lambda = 0.1$

(a) HorseSeg-small, $\lambda = 0.1$ (a) HorseSeg-medium, $\lambda = 0.1$

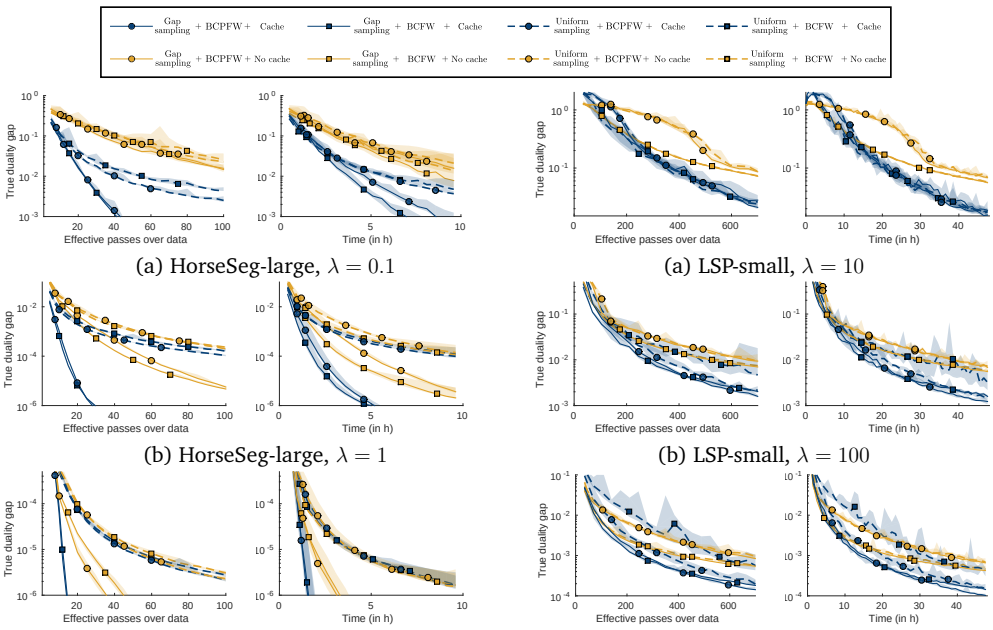
(b) HorseSeg-small, $\lambda = 100$ (b) HorseSeg-medium, $\lambda = 1$

(c) HorseSeg-small, $\lambda = 1000$ (c) HorseSeg-medium, $\lambda = 10$

Fig. 6.4.: Comparison of the variants of BCFW. We compare 8 different methods that can be represented by 3 binary dimensions: object sampling, caching, type of FW steps. We represent these dimensions in different ways: the dimension of caching (in blue) versus no caching (in orange) is represented through colors, the dimension of gap sampling (solid lines) versus uniform sampling (dashed lines) is represented through line style, the dimension of pairwise FW steps (circle markers) versus regular FW steps (square markers) is represented through markers. For each method, we report both the number of effective passes over data (n oracle calls) and the running time against obtained duality gap (computed offline). *The figure is continued in Figure 6.5.*

Dataset	Task	Sructure (Oracle)	Citation	Version	n	d	Sparsity	Box constraints	Oracle time (in s)
OCR	character recognition	chain (Viterbi)	[Taskar *et al.*, 2003]	small	626	4,082	No	No	5×10^{-4}
				large	6,251	4,082	No	No	5×10^{-4}
CoNLL	text chunking	chain (Viterbi)	[Sang and Buchholz, 2000]		8,936	1,643,026	Yes	No	2×10^{-2}
HorseSeg	binary segmentation	grid (graph cut)	[Kolesnikov *et al.*, 2014]	small	147	1,969	No	Yes	1×10^{-3}
				medium	6,121	1,969	No	Yes	1×10^{-3}
				large	25,438	1,969	No	Yes	2×10^{-2} (*)
LSP	pose estimation	tree (max sum)	[Johnson and Everingham, 2010]	small	100	2,676	No	Yes	2 (*)

Tab. 6.1.: Statistics of the datasets used in the experimental evaluation. For the oracle time, we report the sum of the average running times for both the max oracle and the joint feature map computation (the starred numbers indicate that the input features were stored on disk instead of RAM, thus slowing down the computation).



(a) HorseSeg-large, $\lambda = 0.1$ (a) LSP-small, $\lambda = 10$

(b) HorseSeg-large, $\lambda = 1$ (b) LSP-small, $\lambda = 100$

(c) HorseSeg-large, $\lambda = 10$ (c) LSP-small, $\lambda = 1000$

Fig. 6.5.: *Continuation of Figure 6.4.* Comparison of the variants of BCFW on HorseSeg-large and LSP-small.

7 Discussion

In this last chapter, we first summarize the thesis contributions before concluding with opening remarks about potential future work.

7.1 Summary of contributions

Learning from narrated instructional videos. In Chapter 4, we tackled the problem of automatically recovering the main action steps to achieve a given task, such as *changing a car tire* or *repotting a plant*, from a set of narrated instructional videos depicting the same task. We used two assumptions: (i) people perform actions steps roughly when they talk about it, *i.e.*, there is a strong temporal correlation between the language and the visual streams, (ii) tasks are composed of an ordered list of action steps. Given these assumptions, we designed a method based on discriminative clustering that consists of two stages. First, we use Multiple Sequence Alignment to cluster the textual narration into a sequence of main action steps while following the ordering assumption. Second, we localize action steps using DIFFRAC in the visual stream under ordering constraints and localization prior obtained from the language clustering. We evaluated the performance of our method on a newly collected dataset describing five different tasks.

Joint Discovery of Object States and Manipulation Actions. In Chapter 5, we explored the link between object states and manipulation actions. More precisely, we looked at actions which aim at modifying the *state* of an object, such as *opening a door* or *filling a coffee cup*. Given multiple video clips showing the same object manipulation, we proposed a model that automatically localizes the action and detects the object states before and after the action. This model is based on discriminative clustering and seeks to regroup objects and actions that look similar across videos while respecting the constraints that reflect our assumptions: (i) the second state is always after the first state, (ii) people do not manipulate more than one object at the same time, (iii) the action should be in between the two states. We evaluated our method on a

new collected dataset depicting seven different actions. We also demonstrate that our method can be applied to clips that are automatically retrieved from narrated instructional videos coming from YouTube thanks to a simple language processing technique. Our conclusion is that modeling object states improves action localization and vice versa.

Minding the Gaps for Block Frank-Wolfe Optimization. For both previous contributions, we leveraged the Frank-Wolfe algorithm in order to optimize the different objectives we were dealing with. As we have seen, the Frank-Wolfe algorithm is particularly suited to handle the complex constraints that encoded the weak supervision we had on our problems. Scaling up our optimization techniques is a key property for handling the thousands of narrated instructional videos that are available on the web. This is why, in Chapter 6, we introduced several improvements to the Block Coordinate Frank-Wolfe (BCFW) method. BCFW is a variant of the Frank-Wolfe algorithm that allows to scale to larger dataset, by replacing the *batch* update with a cheaper *block* update. The contributions we proposed are based on the fact that a measure of sub-optimality can be computed per block (*gap blocks*) and can be used as a basis for speeding-up the algorithm. First, we proposed to replace the uniform sampling of the blocks that occurs at each iteration of BCFW by an adaptive non-uniform gap-based sampling. Second, we adapted pairwise and away-step variants of Frank-Wolfe into the block-coordinate setting. Third, we also enabled to cache oracle calls with a cache-hit criterion based on the block gaps. We experimentally demonstrated the effectiveness of our improvements with an exhaustive empirical evaluation conducted on four structured prediction datasets.

7.2 Perspectives

In this section, we provide an overview of the potential perspectives offered by this thesis and outline possible directions for future work.

7.2.1 Action step discovery

Automatic Speech Recognition. Our method presented in Chapter 4 relied on manually corrected transcription. This manual correction was necessary in order to remove errors and also add punctuation in order to ease the use of natural language processing tools such as the dependency parser. However, such manual correction is tedious and prevents scaling to many videos. With recent advances in automatic speech recognition that have significantly reduced the word error rate, we believe that efforts should be made to directly exploit automatically generated transcripts.

Towards a large scale dataset of instructional videos. The dataset introduced in Chapter 4 contains over 150 videos for 5 different tasks. While this constitutes an important stepping stone, it remains several orders of magnitude bellow what is actually available online. Among the things that limited the scale of our dataset was the fact that we had to manually filter the videos. Therefore, a significant effort could be devoted to collecting a larger dataset of instructional videos in a more automated manner as was recently done in the context of Vlog style videos [Fouhey *et al.*, 2017]. Among the potential leads, we can mention here the WikiHow[1] database that is a very nice structured source of information for complex human tasks that one could leverage in order to build a large scale dataset. Building such a dataset is promising in order to improve the quality of action recognition models but also for better understanding the variability and complexity of goal oriented human tasks.

Going beyond simple action step structure. One of the main assumption of the method presented in Chapter 4 is that a task is composed of an *ordered* list of steps. This was sufficient for the tasks considered in this thesis and was also important in order to reduce the search space of action steps in the videos. However, even if reasonable, this is a strong assumption which is not always satisfied. It is indeed common to swap or skip some action steps. Therefore, it would be interesting to study more complex task structure. It would be particularly worthwhile to explore ways to learn this structure directly from data.

[1] https://www.wikihow.com

7.2.2 Reasoning about objects and actions

Explicit modeling of human-object interaction. In Chapter 5, we studied the link between action steps and manipulation of objects. For that, we used object detection techniques and employ motion and appearance features, hence only implicitly representing the human interaction. As explained in the related work chapter, much progress has been made over the past few years to estimate human cues, such as person detection but also pose estimation or hand detection. For these reasons, one interesting direction for research is to model more closely the way people interact with objects in the context of instructional videos. We believe that this could be a strong signal to improve performance in object and action recognition.

Going beyond a single change of state. In Chapter 5, we only studied the case of one state change happening on a single object. During instructional videos, *e.g.*, for assembling an ikea furniture, multiple objects are assembled together and undergo various state changes. One direction of future work is to design a model that can handle multiple state changes. It would be notably interesting to combine this with the work conducted in Chapter 4, *i.e.*using narration to discover the action step structure and which action step corresponds to a change of state in objects. Using this, one could then detect these actions and object states in the videos.

Towards object discovery. In Chapter 4, we assumed that we had at our disposal an object detector of the class of interest. The next step may be to automatically discover which objects are important in the videos, either from the narration or directly from the visual signal.

7.2.3 Learning and Optimization

Beyond individual tasks. In Chapter 4 and Chapter 5, we always assume that we deal with videos from a single task or a single manipulation action. As humans, we never really start from scratch when we learn a new task. On the contrary, we reuse knowledge from past experience in order to learn faster and better. For example, even if one does not know *how to make pancakes*, they might know *how to crack an egg* because they have already done it by

learning other recipes. Starting from this observation, we could imagine sharing an action step representation between different tasks. Designing a model that would be able to discover this sharing pattern by looking at multiple videos depicting different tasks is an open challenge. Intuitively, this could potentially collect more examples for each action steps and thus improve the quality of the learned models.

Beyond linear models. All the models presented in this thesis are based on linear classifiers and the square loss. Implicitly, it means that we assume that we already have a good representation of the data that is easily separable according to what we expect. This might not always be the case, especially in the context of narrated instructional videos which show action steps that are not always depicted in standard action recognition datasets and thus might suffer from less tuned representation. Moreover, due to the large number of available instructional videos, there is a hope that we can learn more complex and accurate models. In particular, we could consider enriching the class of classification functions by using the advances of deep learning techniques. Combining such techniques with weak supervision is still an open challenge and constitutes a very nice avenue for future work.

Appendix of Chapter 6

A

A.1 Block descent lemma for BCFW

Definition 1 (Block curvature constant)**.** *Consider a convex function f defined on a separable domain $\mathcal{M} = \mathcal{M}^{(1)} \times \cdots \times \mathcal{M}^{(n)}$. The curvature constant $C_f^{(i)}$ of the function f w.r.t. the individual block of coordinates $\mathcal{M}^{(i)}$ is defined by*

$$\begin{aligned} C_f^{(i)} := \sup \tfrac{2}{\gamma^2} \Big(f(\boldsymbol{\beta}) - f(\boldsymbol{\alpha}) - \langle \boldsymbol{\beta}_{(i)} - \boldsymbol{\alpha}_{(i)}, \nabla_{(i)} f(\boldsymbol{\alpha}) \rangle \Big) \\ \text{s.t. } \boldsymbol{\alpha} \in \mathcal{M},\ \boldsymbol{s}_{(i)} \in \mathcal{M}^{(i)}, \gamma \in [0,1], \\ \boldsymbol{\beta} = \boldsymbol{\alpha} + \gamma(\boldsymbol{s}_{[i]} - \boldsymbol{\alpha}_{[i]}). \end{aligned} \tag{A.1}$$

Here $\boldsymbol{s}_{[i]} \in \mathbb{R}^m$ and $\boldsymbol{\alpha}_{[i]} \in \mathbb{R}^m$ are the zero-padded versions of $\boldsymbol{s}_{(i)} \in \mathcal{M}^{(i)}$ and $\boldsymbol{\alpha}_{(i)} \in \mathcal{M}^{(i)}$, respectively. Note that, although $\boldsymbol{s}_{[i]} \notin \mathcal{M}$ and $\boldsymbol{\alpha}_{[i]} \notin \mathcal{M}$, we have that $\boldsymbol{\beta} := \boldsymbol{\alpha} + \gamma(\boldsymbol{s}_{[i]} - \boldsymbol{\alpha}_{[i]}) \in \mathcal{M}$.

In the case of SSVM, the curvature constant $C_f^{(i)}$ can be upper bounded (tightly in the worst case) with $\frac{4R_i^2}{\lambda n^2}$ where $R_i := \max_{\boldsymbol{y} \in \mathcal{Y}_i} \|\boldsymbol{\psi}_i(\boldsymbol{y})\|_2$ [Lacoste-Julien *et al.*, 2013, Appendix A]. More generally, let $\|\cdot\|_i$ be some norm defined on $\mathcal{M}^{(i)}$. Then suppose that L_i is the Lipschitz-continuity constant with respect to this norm for $\nabla_{(i)} f(\boldsymbol{\alpha})$ when only $\boldsymbol{\alpha}_{(i)}$ varies, i.e., $\|\nabla_{(i)} f(\boldsymbol{\alpha}) - \nabla_{(i)} f(\boldsymbol{\alpha} + \boldsymbol{s}_{[i]} - \boldsymbol{\alpha}_{[i]})\|_i^* \leq L_i \|\boldsymbol{s}_{(i)} - \boldsymbol{\alpha}_{(i)}\|_i$ for all $\boldsymbol{\alpha} \in \mathcal{M}$, $\boldsymbol{s}_{(i)} \in \mathcal{M}^{(i)}$, where $\|\cdot\|_i^*$ is the dual norm of $\|\cdot\|_i$. Then similarly to Lemma 7 in Jaggi, [2013a], we have $C_f^{(i)} \leq L_i \Big(\operatorname{diam}_{\|\cdot\|_i} \mathcal{M}^{(i)} \Big)^2$.

For reference, we restate below the key descent lemma used for the proof of convergence of BCFW and its variants. We note that this is an affine invariant analog of the standard descent lemmas that use the Lipschitz continuity of the gradient function to show progress during first order optimization algorithms.

Lemma 2 (Block descent lemma)**.** *For any $\boldsymbol{\alpha} \in \mathcal{M}$ and block i, let $\boldsymbol{s}_{(i)} \in \mathcal{M}^{(i)}$ be the Frank-Wolfe corner selected by the max oracle of block i at $\boldsymbol{\alpha}$. Let $\boldsymbol{\alpha}_{LS}$*

be obtained by the line search between $\boldsymbol{\alpha}_{(i)} \in \mathcal{M}^{(i)}$ *and* $\boldsymbol{s}_{(i)}$*, i.e.,* $f(\boldsymbol{\alpha}_{LS}) = \min_{\gamma \in [0,1]} f(\boldsymbol{\alpha}_\gamma)$ *where* $\boldsymbol{\alpha}_\gamma := \boldsymbol{\alpha} + \gamma(\boldsymbol{s}_{[i]} - \boldsymbol{\alpha}_{[i]})$*. Then, it holds that for each* $\gamma \in [0, 1]$*:*

$$f(\boldsymbol{\alpha}_{LS}) \leq f(\boldsymbol{\alpha}) - \gamma g_i(\boldsymbol{\alpha}) + \tfrac{\gamma^2}{2} C_f^{(i)} \tag{A.2}$$

where $C_f^{(i)}$ *is the curvature constant of the function* f *over the factor* $\mathcal{M}^{(i)}$ *and* $g_i(\boldsymbol{\alpha})$ *is the block gap at the point* $\boldsymbol{\alpha}$ *w.r.t. the block* i*.*

Proof. From Definition 1 of the curvature constant and the expression (6.7) for the block gap, we have

$$\begin{aligned} f(\boldsymbol{\alpha}_\gamma) =& f(\boldsymbol{\alpha} + \gamma(\boldsymbol{s}_{[i]} - \boldsymbol{\alpha}_{[i]})) \\ \leq& f(\boldsymbol{\alpha}) + \gamma \langle \boldsymbol{s}_{(i)} - \boldsymbol{\alpha}_{(i)}, \nabla_{(i)} f(\boldsymbol{\alpha}) \rangle + \tfrac{\gamma^2}{2} C_f^{(i)} \\ =& f(\boldsymbol{\alpha}) - \gamma g_i(\boldsymbol{\alpha}) + \tfrac{\gamma^2}{2} C_f^{(i)} \ . \end{aligned}$$

The inequality $f(\boldsymbol{\alpha}_{LS}) \leq f(\boldsymbol{\alpha}_\gamma)$ completes the proof. □

A.2 Toy example for gap sampling

In this section, we construct a toy example of the structured SVM problem where the adaptive gap-based sampling is n times faster than non-adaptive sampling schemes such as uniform sampling or curvature-based sampling (the latter being the affine invariant analog of Lipschitz-based sampling).

A.2.1 General idea

The main idea is to consider a training set where there are $n - 1$ "easy" objects that need to be visited only once to learn to classify them, and one "hard" object that requires at least $K \gg 1$ visits in order to get the optimal parameter. We can design the example in such a way that the curvature or Lipschitz constants are non-informative about which example is hard, and which is easy. The non-adaptive sampling schemes will thus have to visit the easy objects as often as the hard object, whereas the gap sampling technique can adapt to focus only on the single hard object after having visited the easy objects once, thus yielding an overall $\min\{n, K\}$-times speedup. Note

that large-scale learning datasets could have analogous features as this toy example: a subgroup of objects might be easier to learn than another, and moreover, they might share similar information, so that after visiting a subset, we do not need to linger on the other ones from the same subset as all the information has already been extracted. We cannot know in advance which subgroups are these subsets, and thus an adaptive scheme is needed.

A.2.2 Explicit construction

For simplicity, we set the weight of the regularizer λ to $1/n$ so that the scaling factor defining A in Problem (6.3) is $1/\lambda n = 1$. The matrix A thus consists of the difference feature maps, i.e.,

$$A := \left\{ \boldsymbol{\psi}_i(\boldsymbol{y}) := \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}_i) - \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}) \in \mathbb{R}^d \;\middle|\; i \in [n], \boldsymbol{y} \in \mathcal{Y}_i \right\}.$$

In our example, we use feature maps of dimensionality $d := K + 1 := |\mathcal{Y}_i|$. Let $\mathcal{Y}_i := \{0, 1, \ldots, K\}$ be the set of labels for the object i and the label 0 be the correct label. We consider the zero-one loss, i.e., $L_i(0) = 0$ and $L_i(k) = 1$ for $k \geq 1$. In the following, let $\{\boldsymbol{e}_j\}_{j=1}^d$ be the standard basis vectors for $\mathbb{R}^d$.

Hard and easy objects. We construct the feature map $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ so that only the last coordinate of the parameter vector is needed to classify correctly the easy object, whereas all the other coordinates are needed to classify correctly the hard object. By using a different set of coordinates between the easy and the hard objects, we simplify the analysis as the optimization for both block types decouples (become independent). Specifically, we set the feature map for the correct label to be $\boldsymbol{\phi}(\boldsymbol{x}_i, 0) := \mathbf{0}$ for all objects i. We let $i = 1$ be the hard object and we set $\boldsymbol{\phi}(\boldsymbol{x}_i, k) := -\frac{1}{\sqrt{2}}\boldsymbol{e}_k$ for $k = 1, \ldots, K$. For the easy object, $i \in \{2, \ldots, n\}$, we use the constant $\boldsymbol{\phi}(\boldsymbol{x}_i, k) := -\boldsymbol{e}_{K+1}$ for all $k \geq 1$. The normalization of the feature maps is made so that the curvature constants for all the objects are equal (see below). Note also here that $\boldsymbol{\psi}_1(k) \perp \boldsymbol{\psi}_i(l)$ for any labels k, l, and thus the optimization over block 1 decouples with the

one for the other blocks $i = 2, \dots, n$. The SSVM dual (6.3) takes here the following simple form:

$$\begin{aligned} \min_{\substack{\boldsymbol{\alpha}\in\mathbb{R}^m \\ \boldsymbol{\alpha}\succcurlyeq 0}} \quad & \tfrac{1}{n}\sum_{k=1}^{K}\Big(\tfrac{1}{4}\alpha_1(k)^2 - \alpha_1(k)\Big) + \tfrac{1}{n}(\tfrac{1}{2}u^2 - u) \\ \text{s.t.} \quad & \sum_{k=0}^{K}\alpha_i(k) = 1 \quad \forall i \in [n]\,, \quad u = \sum_{i=2}^{n}\sum_{k=1}^{K}\alpha_i(k)\,, \end{aligned} \tag{A.3}$$

where we have introduced the auxiliary variable u to highlight the simple structure for the optimization over the easy blocks. The unique[1] solution for the first (hard) block is easily seen to be $\alpha_1^*(k) = \frac{1}{K}$ for $k \geq 1$ and $\alpha_1^*(0) = 0$. For the easy blocks, any feasible combination of dual variables that gives $u^* = 1$ is a solution. This gives the optimal parameter $\boldsymbol{w}^* = A\boldsymbol{\alpha}^* = \boldsymbol{e}_{K+1} + \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{e}_k$.

Optimization on the hard object. The objective for the hard object (block 1) in (A.3) is similar to the one used to show a lower bound of $\Omega(1/t)$ suboptimality error after t iterations for the Frank-Wolfe algorithm for t smaller than the dimensionality (e.g., see Lemma 3 and 4 in Jaggi, [2013a]), hence showing that the optimization is difficult on this block. The BCFW algorithm is initialized with $\boldsymbol{w} = \mathbf{0}$, which corresponds to putting all the mass on the correct label, i.e., $\alpha_i(0) = 1$ and $\alpha_i(k) = 0$, $k \geq 1$. At each iteration of BCFW, the mass can be moved only towards one corner, and all the corners (of the simplex) have exactly one non-zero coordinate. This means that after t iterations of BCFW on the first block, at most t non-ground truth dual variables can be non-zero. Minimizing the objective (A.3) over the first block with the constraint that at most t of these variables are non-zero give the similar solution $\alpha_1(k) = 1/t$ for $k = 1, \dots, t$, which gives a suboptimality of $\frac{1}{4n}(\frac{1}{t} - \frac{1}{K})$ for $t \leq K$. Similarly, this also yields the smallest FW gap[2] possible for this block after t iterations, which is $\frac{1}{n}\frac{1}{2t}$. This means that in order to get a suboptimality error smaller than ε, one needs at least

$$t \geq \Omega(\min\{K, \tfrac{1}{n\varepsilon}\}) \tag{A.4}$$

[1] Uniqueness can be proved by noticing that the objective is strongly convex in $\boldsymbol{\alpha}_1$ after removing $\alpha_1(0)$ and replacing the equality constraint with an inequality.

[2] Recall that the FW gap here is the same as the Lagrangian duality gap (see Section 6.2.3), and so if one cares about the SSVM primal suboptimality, one needs a small FW gap.

BCFW iterations on the first block.[3]

Optimization on easy objects. Finally, we now show that after one iteration on *any* easy object, the gaps g_i on all easy objects become zero (i.e., they are all optimal and then stay optimal as the optimization is decoupled with the first block). After this iteration, BCFW with gap sampling visits all the easy objects exactly once and sets their gap estimates to zero, thus never revisiting them again.

Note that before visiting any easy object i, we have $\langle \boldsymbol{w}, \boldsymbol{\psi}_i(k) \rangle = 0$ for all k as the features for the hard object are orthogonal and $\boldsymbol{w}$ is initialized to zero. Thus, at the first visit of an easy object $i \in \{2, \dots, n\}$, we have $H_i(0; \boldsymbol{w}) = 0$ and $H_i(k; \boldsymbol{w}) = 1$, $k \geq 1$, and the max oracle returns some (any) label $k \in \{1, \dots, K\}$. Following the steps of Algorithm 3, we have $\boldsymbol{w}_s := \frac{1}{\lambda n} \boldsymbol{\psi}(k) = \boldsymbol{e}_{K+1}$ and $\ell_s = \frac{1}{n} L_i(k) = \frac{1}{n}$. Then $g_i = \frac{1}{n}$ and $\gamma = 1$ as $\boldsymbol{w}_i = \boldsymbol{0}$. The assignment $\boldsymbol{w}_i = \boldsymbol{e}_{K+1}$ implies the update $\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{e}_{K+1}$ of the parameter vector. After such an update, at all iterations, for all easy objects $i \in \{2, \dots, n\}$ and for all labels $k \in \{1, \dots, K\}$, we have

$$H_i(k; \boldsymbol{w}) = L_i(k) - \langle \boldsymbol{w}, \boldsymbol{e}_{K+1} \rangle = 0 \tag{A.5}$$

because the coordinate w_{K+1} is never updated again. According to (6.8), the equalities (A.5) imply that the block gaps g_i equal zero for all the easy objects.

Curvature constants. The simple structure of the matrix A allows us to explicitly compute the curvature constants $C_f^{(i)}$ corresponding to both easy and hard objects.

[3] In fact, BCFW also has a $\mathcal{O}(\frac{1}{nt})$ gap after t iterations on the first block by the standard FW convergence theorem, as $C_f^{(1)} = \frac{1}{n}$ as we show in (A.6).

The SSVM dual (6.3) is a quadratic function with a constant Hessian $H := \lambda A^\mathsf{T} A$, so the second-order Taylor expansion of f at a point $\boldsymbol{\alpha}$ allows us to rewrite the definition A.1 as

$$\begin{aligned} C_f^{(i)} &= \sup_{\substack{\boldsymbol{\alpha}\in\mathcal{M} \\ \boldsymbol{s}_{(i)}\in\mathcal{M}^{(i)}}} (\boldsymbol{s}_{[i]} - \boldsymbol{\alpha}_{[i]})^\mathsf{T} H (\boldsymbol{s}_{[i]} - \boldsymbol{\alpha}_{[i]}) \\ &= \lambda \sup_{\substack{\boldsymbol{\alpha}\in\mathcal{M} \\ \boldsymbol{s}_{(i)}\in\mathcal{M}^{(i)}}} \|A(\boldsymbol{s}_{[i]} - \boldsymbol{\alpha}_{[i]})\|_2^2 \\ &= \lambda \max_{k,l} \|\boldsymbol{\phi}(\boldsymbol{x}_i, k) - \boldsymbol{\phi}(\boldsymbol{x}_i, l)\|_2^2. \end{aligned}$$

The last line uses the property that the maximum of a convex function over a convex set is obtained at a vertex.

In the case of the hard object, we can get

$$C_f^{(1)} = \lambda \|\boldsymbol{\phi}(\boldsymbol{x}_1, 1) - \boldsymbol{\phi}(\boldsymbol{x}_1, 2)\|_2^2 = \tfrac{1}{n}. \tag{A.6}$$

In the case of an easy object, we can get

$$C_f^{(i)} = \lambda \|\boldsymbol{\phi}(\boldsymbol{x}_i, 1) - \boldsymbol{\phi}(\boldsymbol{x}_i, 0)\|_2^2 = \tfrac{1}{n}. \tag{A.7}$$

Adaptive and non-adaptive sampling. Let t be the number of steps needed on the hard block. By (A.4), we need $t \geq \Omega(\min\{K, \frac{1}{n\varepsilon}\})$ to get a suboptimality smaller than ε. The uniform sampling scheme visits all the objects with the same probability. In the setting constructed above, it makes, on average, t visits to each easy object prior to visiting the hard object t times. Thus, the overall scheme will call the max-oracle $O(nt)$ times. All the curvature constants $C_f^{(i)}$ are equal, so the sampling proportional to the curvature constants is equivalent to uniform sampling.

The adaptive sampling scheme visits each easy object only once after the first visit to any of them. After such a visit to any easy object, its local gap estimate equals zero and this object is never visited again. The gap sampling scheme thus makes an overall $O(n + t)$ oracle calls. The adaptive scheme is thus approximately $\min\{n, t\} = \min\{n, K, \frac{1}{n\varepsilon}\}$ times faster than the non-adaptive ones. The speed-up can be made arbitrary large by setting both n and K large enough, and ε small enough.

Lipschitz and curvature constants. The non-uniform sampling techniques used in the work of Nesterov, [2012], Needell *et al.*, [2014], and Zhao and Zhang, [2015] use Lipschitz constants of partial derivatives to obtain the sampling probabilities. In our discussion above, we use the curvature constants. Lacoste-Julien and Jaggi, [2015b, Appendix C] note that the curvature constants are affine invariant quantities and, thus, are more suited for the analysis of Frank-Wolfe methods compared to Lipschitz constants (which depend on a choice of norm). We illustrate this point on our toy example by explicitly computing the Lipschitz constants over blocks for the ℓ_2 and ℓ_1 norm. For both easy and hard blocks, the Lipschitz constant of the gradient with respect to the ℓ_2 norm equals the largest eigenvalues of the corresponding block Hessians. For an easy object, the block Hessian is a rank one matrix with the only non-zero eigenvalue equal to $\lambda(|\mathcal{Y}_i| - 1) = \frac{K}{n}$. For the hard object, the block Hessian is a diagonal matrix with non-zero entries equal to $\frac{\lambda}{2} = \frac{1}{2n}$. Here the Lipschitz constant for the easy block is about K times bigger than the one for the hard block, and thus for a large number of labels K, sampling according to Lipschitz constants can be much slower than sampling according to the curvature constants, which was itself slower than the adaptive sampling scheme.

This poor scaling of the Lipschitz constants is partly due to the bad choice of norm in relationship to the optimization domain. d'Aspremont *et al.*, [2013] suggests to use the atomic norm of the domain $\mathcal{M}$ for the analysis. In the case of the simplex, we get the ℓ_1 norm to measure the diameter of the domain, and its dual norm (ℓ_∞) to measure the Lipschitz constant of the gradient. With this norm, the Lipschitz constant stays as $\frac{1}{2n}$ for the hard block, but decreases to the more reasonable $\frac{1}{n}$ for the easy blocks. As explained before, we can use the bound $C_f^{(i)} \leq L_i \left(\operatorname{diam}_{\|\cdot\|_i} \mathcal{M}^{(i)} \right)^2$ for the curvature constant. As the diameter for the simplex measured with the ℓ_1-norm is 2, we get the bound $C_f^{(1)} \leq \frac{2}{n}$ for the hard block, very close to its exact value of $\frac{1}{n}$ as derived in (A.6). The ℓ_1 norm thus appears as a more appropriate choice for this problem.

A.3 Proof of Theorem 2 (convergence of BCFW with gap sampling)

Lemma 3 (Expected block descent lemma)**.** *Let $g_j(\boldsymbol{\alpha}^{(k)})$ be the block gap for block j for the iterate $\boldsymbol{\alpha}^{(k)}$. Let $\boldsymbol{\alpha}^{(k+1)}$ be obtained by sampling a block i with probability p_i and then doing a (block) FW step with line-search on this block, starting from $\boldsymbol{\alpha}^{(k)}$. Consider* any *set of scalars $\gamma_j \in [0,1]$, $j = 1, \dots, n$, which do not depend on the chosen block i. Then in conditional expectation over the random choice of block i with probabilities p_i, it holds:*

$$\mathbf{E}\Big[f(\boldsymbol{\alpha}^{(k+1)}) \,|\, \boldsymbol{\alpha}^{(k)}\Big] \leq f(\boldsymbol{\alpha}^{(k)}) - \sum_{i=1}^{n} \gamma_i p_i g_i(\boldsymbol{\alpha}^{(k)}) + \tfrac{1}{2} \sum_{i=1}^{n} \gamma_i^2 p_i C_f^{(i)}. \tag{A.8}$$

Proof. The proof is analogous to the proof of Lemma 2, but being careful with the expectation. Let block i be the chosen one that defined $\boldsymbol{\alpha}^{(k+1)}$ and let $\boldsymbol{\alpha}_\gamma := \boldsymbol{\alpha} + \gamma(\boldsymbol{s}_{[i]} - \boldsymbol{\alpha}_{[i]})$, where $\mathbf{s}_{(i)} \in \mathcal{M}^{(i)}$ is the FW corner on block i and $\mathbf{s}_{[i]} \in \mathbb{R}^m$ is its zero-padded version. By the line-search, we have $f(\boldsymbol{\alpha}^{(k+1)}) \leq f(\boldsymbol{\alpha}_\gamma)$ for any $\gamma \in [0,1]$. By using $\gamma = \gamma_i$ in the bound (A.1) provided in the curvature Definition 1, and by the definition of the Frank-Wolfe gap, we get:

$$\begin{aligned} f(\boldsymbol{\alpha}^{(k+1)}) \leq f(\boldsymbol{\alpha}_{\gamma_i}) &= f(\boldsymbol{\alpha}^{(k)} + \gamma_i(\mathbf{s}_{[i]} - \boldsymbol{\alpha}_{[i]}^{(k)})) \\ &\leq f(\boldsymbol{\alpha}^{(k)}) + \gamma_i g_i(\boldsymbol{\alpha}^{(k)}) + \tfrac{\gamma_i^2}{2} C_f^{(i)}. \end{aligned}$$

Taking the expectation of the bound with respect to i, conditioned on $\boldsymbol{\alpha}^{(k)}$, proves the lemma. □

Definition 4. *The* nonuniformity measure *$\chi(\boldsymbol{x})$ of a vector $\boldsymbol{x} \in \mathbb{R}_+^n$ is defined as:*

$$\chi(\boldsymbol{x}) := \sqrt{1 + n^2 \operatorname{Var}\big[\boldsymbol{p}\big]}$$

where $\boldsymbol{p} := \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_1}$ is the probability vector obtained by normalizing $\boldsymbol{x}$.

Lemma 5. *Let $\boldsymbol{x} \in \mathbb{R}_+^n$. The following relation between its ℓ_1-norm and ℓ_2-norm holds:*

$$\|\boldsymbol{x}\|_2 = \tfrac{\chi(\boldsymbol{x})}{\sqrt{n}} \|\boldsymbol{x}\|_1 .$$

Proof. We have that

$$\mathrm{Var}\left[\boldsymbol{p}\right] = \mathbf{E}\left[\boldsymbol{p}^2\right] - \mathbf{E}\left[\boldsymbol{p}\right]^2 = \tfrac{1}{n}\|\boldsymbol{p}\|_2^2 - \tfrac{1}{n^2}. \tag{A.9}$$

Combining (A.9) and Definition 4 we prove the lemma. □

Remark. For any $\boldsymbol{x} \in \mathbb{R}_+^n$, the quantity $\chi(\boldsymbol{x})$ always belongs to the segment $[1, \sqrt{n}]$. We have $\chi(\boldsymbol{x}) = 1$ when all the elements of $\boldsymbol{x}$ are equal and $\chi(\boldsymbol{x}) = \sqrt{n}$ when all the elements, except one, equal zero.

Theorem 2. *Assume that at each iterate $\boldsymbol{\alpha}^{(k)}$, $k \geq 0$, BCFW with gap sampling (Algorithm **??**) has access to the exact values of the block gaps. Then, at each iteration, it holds that $\mathbf{E}\left[f(\boldsymbol{\alpha}^{(k)})\right] - f(\boldsymbol{\alpha}^*) \leq \frac{2n}{k+2n}\left(C_f^{\otimes}\chi^{\otimes} + h_0\right)$ where $\boldsymbol{\alpha}^* \in \mathcal{M}$ is a solution of problem (6.3), $h_0 := f(\boldsymbol{\alpha}^{(0)}) - f(\boldsymbol{\alpha}^*)$ is the suboptimality at the starting point of the algorithm, the constant $C_f^{\otimes} := \sum_{i=1}^n C_f^{(i)}$ is the sum of the curvature constants, and the constant $\chi^{\otimes}$ is an upper bound on $\mathbf{E}\left[\frac{\chi(C_f^{(:)})}{\chi(\boldsymbol{g}_:(\boldsymbol{\alpha}^{(k)}))^3}\right]$, which quantities the amount of non-uniformity of the $C_f^{(i)}$'s in relationship to the non-uniformity of the gaps obtained during the algorithm. The expectations are taken over the random choice of the sampled block at iterations $1, \ldots, k$ of the algorithm.*

Proof. Starting from Lemma 3 with $\gamma_i := \gamma$ for some γ to be determined later and $p_i := \frac{g_i}{g}$ where $g_i := g_i(\boldsymbol{\alpha}^{(k)})$ and $g := g(\boldsymbol{\alpha}^{(k)})$, we get

$$\begin{aligned}\mathbf{E}\left[f(\boldsymbol{\alpha}^{(k+1)}) \,|\, \boldsymbol{\alpha}^{(k)}\right] \leq f(\boldsymbol{\alpha}^{(k)}) - \gamma \sum_{i=1}^n \frac{g_i^2}{g} \\ + \frac{\gamma^2}{2} \sum_{i=1}^n C_f^{(i)} \frac{g_i}{g}.\end{aligned} \tag{A.10}$$

The Cauchy-Schwarz inequality bounds the dot product between the vectors of curvature constants $\boldsymbol{c} := C_f^{(:)} := (C_f^{(i)})_{i=1}^n$ and block gaps $\boldsymbol{g} := \boldsymbol{g}_:(\boldsymbol{\alpha}^{(k)}) := (g_i)_{i=1}^n$

$$\sum_{i=1}^n C_f^{(i)} g_i \leq \|\boldsymbol{c}\|_2 \, \|\boldsymbol{g}\|_2 \,. \tag{A.11}$$

Combining (A.11) and the result of Lemma 5 for the vectors of curvature

constants and block gaps (with $\|\boldsymbol{g}\|_1 = g$ and $\|\boldsymbol{c}\|_1 = C_f^{\otimes}$), we can further bound (A.10):

$$\mathbf{E}\Big[f(\boldsymbol{\alpha}^{(k+1)}) \,|\, \boldsymbol{\alpha}^{(k)}\Big] \leq f(\boldsymbol{\alpha}^{(k)}) - \tfrac{\gamma g}{n}\chi(\boldsymbol{g})^2 + \tfrac{\gamma^2}{2n}\chi(\boldsymbol{g})\,\chi(\boldsymbol{c})\,C_f^{\otimes}. \tag{A.12}$$

Subtracting the minimal function value $f(\boldsymbol{\alpha}^*)$ from both sides of (A.12) and by using $h(\boldsymbol{\alpha}^{(k)}) := f(\boldsymbol{\alpha}^{(k)}) - f(\boldsymbol{\alpha}^*) \leq g$, we bound the conditional expectation of the suboptimality h with

$$\mathbf{E}[h(\boldsymbol{\alpha}^{(k+1)}) \mid \boldsymbol{\alpha}^{(k)}] \leq h(\boldsymbol{\alpha}^{(k)}) - \tfrac{\gamma}{n}\chi(\boldsymbol{g})^2\, h(\boldsymbol{\alpha}^{(k)}) + \tfrac{\gamma^2}{2n}\chi(\boldsymbol{g})\,\chi(\boldsymbol{c})\,C_f^{\otimes} \tag{A.13}$$

which is analogous to [Lacoste-Julien *et al.*, 2013, Eq. (20)]. In what follows, we use the modified induction technique of [Lacoste-Julien *et al.*, 2013, Proof of Theorem C.1].

By induction, we are going to prove the following upper bound on the unconditional expectation of the suboptimality h:

$$\mathbf{E}\Big[h(\boldsymbol{\alpha}^{(k)})\Big] \leq \tfrac{2nC}{k+2n}, \quad \text{for } k \geq 0, \tag{A.14}$$

that corresponds to the statement of the theorem with $C := C_f^{\otimes}\chi^{\otimes} + h_0$.

The basis of the induction $k = 0$ follows immediately from the definition of C, given that $C_f^{\otimes} \geq 0$ and $\chi^{\otimes} > 0$.

Consider the induction step. Assume that (A.14) is satisfied for $k \geq 0$. With a particular choice of step size $\gamma := \frac{2n}{\chi(\boldsymbol{g})^2(k+2n)} \in [0, 1]$ (which does not depend on the picked i), we rewrite the bound (A.13) on the conditional expectation as

$$\mathbf{E}[h(\boldsymbol{\alpha}^{(k+1)}) \mid \boldsymbol{\alpha}^{(k)}] \leq \left(1 - \tfrac{2}{k+2n}\right) h(\boldsymbol{\alpha}^{(k)}) + \tfrac{2n}{(k+2n)^2}\,\tfrac{\chi(\boldsymbol{c})\,C_f^{\otimes}}{\chi(\boldsymbol{g})^3}. \tag{A.15}$$

Taking the unconditional expectation of (A.15), then the induction assumption (A.14) and the definition of $\chi^{\otimes}$ give us the deterministic inequality

$$\mathbf{E}[h(\boldsymbol{\alpha}^{(k+1)})] \leq \left(1 - \tfrac{2}{k+2n}\right) \tfrac{2nC}{k+2n} + \tfrac{2n}{(k+2n)^2} \chi^{\otimes} C_f^{\otimes}. \tag{A.16}$$

Bounding $\chi^{\otimes} C_f^{\otimes}$ by C and rearranging the terms gives

$$\begin{aligned}\mathbf{E}[h(\boldsymbol{\alpha}^{(k+1)})] &\leq \tfrac{2nC}{k+2n}\left(1 - \tfrac{2}{k+2n} + \tfrac{1}{k+2n}\right)\\ &= \tfrac{2nC}{k+2n}\tfrac{k+2n-1}{k+2n}\\ &\leq \tfrac{2nC}{k+2n}\tfrac{k+2n}{k+2n+1}\\ &= \tfrac{2nC}{(k+1)+2n}\,,\end{aligned}$$

which completes the induction proof. □

A.4 Proof of Theorem 3 (convergence of BCPFW and BCAFW)

In this section, we prove Theorem 3 that states that the suboptimality error on (6.3) decreases geometrically in expectation for BCPFW and BCAFW for the iterates at which *no* block would have a drop step, i.e., when no atom would be removed from the active sets. We follow closely the notation and the results from Lacoste-Julien and Jaggi, [2015b] where the global linear convergence of the (batch) pairwise FW (PFW) and away-step FW (AFW) algorithms was shown. The main insight to get our result is that the "pairwise FW gap" decomposes also as a sum of block gaps. We give our result for the following more general setting (the block-separable analog of the setup in Appendix F of Lacoste-Julien and Jaggi, [2015b]):

$$\begin{aligned}\min_{\boldsymbol{\alpha}\in\mathcal{M}} f(\boldsymbol{\alpha}) \quad &\text{with} \quad f(\boldsymbol{\alpha}) := q(A\boldsymbol{\alpha}) + \boldsymbol{b}^\top\boldsymbol{\alpha}\\ &\text{and} \quad \mathcal{M} = \mathcal{M}^{(1)} \times \cdots \times \mathcal{M}^{(n)},\end{aligned} \tag{A.17}$$

where q is a strongly convex function, and $\mathcal{M}^{(i)} := \operatorname{conv}(\mathcal{A}^{(i)})$ for each i, where $\mathcal{A}^{(i)} \subseteq \mathbb{R}^{m_i}$ is a *finite* set of vectors (called *atoms*). In other words, each $\mathcal{M}^{(i)}$ is a polytope. For the example of the dual SSVM objective (6.3),

$q(\cdot) := \frac{\lambda}{2}\|\cdot\|^2$ and $\mathcal{A}^{(i)}$ are the corners of a probability simplex in $m_i := |\mathcal{Y}_i|$ dimensions.

Suppose that we maintain an active set $\mathcal{S}_i$ for each block (as in the BCPFW algorithm). We first relate the batch PFW direction with the block PFW directions, as well as their respective batch and blockwise *PFW gaps* (the PFW gap is replacing the FW gap (6.6) in the analysis of PFW).

Definition 6 (Block PFW gaps)**.** *Consider the problem* (A.17) *and suppose that the point $\boldsymbol{\alpha}$ has each of its block $\boldsymbol{\alpha}_{(i)}$ with current active set $\mathcal{S}_i \subseteq \mathcal{A}^{(i)}$.*[4] *We define the corresponding* batch PFW gap *at $\boldsymbol{\alpha}$ with active set $\mathcal{S} := \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ as:*

$$
\begin{aligned}
g^{\text{PFW}}(\boldsymbol{\alpha}; \mathcal{S}) &:= \max_{\boldsymbol{s}\in\mathcal{M}, \boldsymbol{v}\in\mathcal{S}} \langle -\nabla f(\boldsymbol{\alpha})\,,\, \boldsymbol{s} - \boldsymbol{v}\rangle && \text{(A.18)}\\
&= \max_{\boldsymbol{s}\in\mathcal{M}, \boldsymbol{v}\in\mathcal{S}} \sum_i \langle -\nabla_{(i)} f(\boldsymbol{\alpha})\,,\, \boldsymbol{s}_{(i)} - \boldsymbol{v}_{(i)}\rangle \\
&= \sum_i \underbrace{\max_{\substack{\boldsymbol{s}_{(i)}\in\mathcal{M}^{(i)}\\ \boldsymbol{v}_{(i)}\in\mathcal{S}_i}} \langle -\nabla_{(i)} f(\boldsymbol{\alpha})\,,\, \boldsymbol{s}_{(i)} - \boldsymbol{v}_{(i)}\rangle}_{} \\
&=: \sum_i \quad g_i^{\text{PFW}}(\boldsymbol{\alpha}\,;\, \mathcal{S}_i), && \text{(A.19)}
\end{aligned}
$$

where g_i^{PFW} is the PFW gap for block i. We recognize that the maximizing arguments for g_i^{PFW} are the FW corner $\boldsymbol{s}_{(i)}$ and the away corner $\boldsymbol{v}_{(i)}$ for block i that one would obtain when running BCPFW on this block.

We note that by maintaining independent active sets $\mathcal{S}_i$ for each block, the number of potential away corner combinations is exponential in the number of blocks, yielding many more possible directions of movement than in the batch PFW algorithm where the number of away corners is bounded by the number of iterations. Moreover, suppose that we have an explicit expansion for each block $\boldsymbol{\alpha}_{(i)}$ as a convex combination of atoms in the active set: $\boldsymbol{\alpha}_{(i)} = \sum_{\boldsymbol{v}_{(i)}\in\mathcal{S}_i} \beta_i(\boldsymbol{v}_{(i)})\, \boldsymbol{v}_{(i)}$, where $\beta_i(\boldsymbol{v}_{(i)}) > 0$ is the convex combination coefficient associated with atom $\boldsymbol{v}_{(i)}$. Then we can also express $\boldsymbol{\alpha}$ as an explicit convex combination of the (exponential size) active set $\mathcal{S}$ as follows: $\boldsymbol{\alpha} = \sum_{\boldsymbol{v}\in\mathcal{S}} \beta(\boldsymbol{v})\, \boldsymbol{v}$, where $\beta(\boldsymbol{v}) := \prod_{i=1}^n \beta_i(\boldsymbol{v}_{(i)})$.

[4] That is, $\boldsymbol{\alpha}_{(i)}$ is a convex combination of *all* the elements of $\mathcal{S}^{(i)}$ with non-zero coefficients.

We can now prove an analog of the expected block descent lemma (Lemma 3 for BCFW) in the case of BCPFW and BCAFW. For technical reasons, we need a slightly different block curvature constant $C_f^{\mathrm{A}\,(i)}$ (cf. Eq. (26) in Lacoste-Julien and Jaggi, [2015b]).

Lemma 7 (Expected BCPFW descent lemma)**.** *Consider running the BCPFW algorithm on problem* (A.17)*. Let* $\boldsymbol{\alpha}^{(k)}$ *be the current iterate, and suppose that* $\mathcal{S}_j$ *is the current active set for each block* $\boldsymbol{\alpha}_{(j)}^{(k)}$*. Let* $\mathcal{S}^{(k)} := \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ *be the current (implicit) active set for* $\boldsymbol{\alpha}^{(k)}$*. Suppose that* there is no drop set at $\boldsymbol{\alpha}^{(k)}$*, that is, that for each possible block* i *that could be picked at this stage, the PFW step with line-search on block* i *will not have its step size truncated (we say that the line-search* will succeed*). Then, conditioned on the current state, in expectation over the random choice of block* i *with uniform probability and for any* $\gamma \in [0, 1]$*, it holds for the next iterate* $\boldsymbol{\alpha}^{(k+1)}$ *of BCPFW:*

$$\mathbf{E}\Big[f(\boldsymbol{\alpha}^{(k+1)}) \,|\, \boldsymbol{\alpha}^{(k)}, \mathcal{S}^{(k)}\Big] \leq f(\boldsymbol{\alpha}^{(k)}) - \tfrac{\gamma}{n} g^{\mathrm{PFW}}(\boldsymbol{\alpha}^{(k)}\,;\, \mathcal{S}^{(k)}) + \tfrac{\gamma^2}{2n} C_f^{\mathrm{A}\otimes}, \quad \text{(A.20)}$$

where $C_f^{\mathrm{A}\otimes} := \sum_{i=1}^n C_f^{\mathrm{A}\,(i)}$ *is the total (away) curvature constant, and where* $C_f^{\mathrm{A}\,(i)}$ *is defined as in Definition 1, but allowing the reference point* $\boldsymbol{\alpha}_{(i)}$ *in* (A.1) *to be any point* $\boldsymbol{v}_{(i)} \in \mathcal{M}^{(i)}$ *instead, thus allowing a pairwise FW direction* $\boldsymbol{s}_{[i]} - \boldsymbol{v}_{[i]}$ *to be used in its definition.*

Moreover, (A.20) *also holds for BCAFW (again under the assumption of* no drop step*), but with an extra* $1/2$ *factor in front of* $g^{\mathrm{PFW}}(\boldsymbol{\alpha}^{(k)}\,;\, \mathcal{S}^{(k)})$ *in the bound.*

Proof. Let block i be the chosen one that defined $\boldsymbol{\alpha}^{(k+1)}$ and let $\boldsymbol{\alpha}_\gamma := \boldsymbol{\alpha}^{(k)} + \gamma(\boldsymbol{s}_{[i]} - \boldsymbol{v}_{[i]})$, where $\mathrm{s}_{(i)} \in \mathcal{M}^{(i)}$ is the FW corner on block i with $\mathrm{s}_{[i]} \in \mathbb{R}^m$ its zero-padded version, and similarly $\boldsymbol{v}_{(i)} \in \mathcal{S}_i$ is the chosen away corner on block i. By assumption, we have that the line-search succeeds, i.e., the minimum of $\min_{\gamma \in [0, \gamma_{\max}]} f(\boldsymbol{\alpha}_\gamma)$ is achieved for $\gamma^* < \gamma_{\max}$, where $\gamma_{\max}$ is the maximum step size for this block for the PFW direction (this is because the optimal step size for the line-search cannot be truncated at $\gamma_{\max}$, as otherwise it would be a drop step). As f is a convex function, this means that $f(\boldsymbol{\alpha}^{(k+1)}) = \min_{\gamma \in [0, \gamma_{\max}]} f(\boldsymbol{\alpha}_\gamma) = \min_{\gamma \geq 0} f(\boldsymbol{\alpha}_\gamma)$ (removing inactive con-

straints does not change its minimum). By definition of $C_f^{\mathrm{A}\,(i)}$, we thus have for any $\gamma \in [0,1]$:

$$\begin{aligned} f(\boldsymbol{\alpha}^{(k+1)}) &\leq f(\boldsymbol{\alpha}_\gamma) \\ &= f(\boldsymbol{\alpha}^{(k)} + \gamma(\boldsymbol{s}_{[i]} - \boldsymbol{v}_{[i]})) \\ &\leq f(\boldsymbol{\alpha}^{(k)}) + \gamma\langle \nabla_{(i)} f(\boldsymbol{\alpha}),\, \boldsymbol{s}_{(i)} - \boldsymbol{v}_{(i)}\rangle + \tfrac{\gamma^2}{2} C_f^{\mathrm{A}\,(i)} \\ &= f(\boldsymbol{\alpha}^{(k)}) - \gamma\, g_i^{\mathrm{PFW}}(\boldsymbol{\alpha}^{(k)}; \mathcal{S}_i) + \tfrac{\gamma^2}{2} C_f^{\mathrm{A}\,(i)}. \end{aligned} \tag{A.21}$$

Taking the expectation of the bound with respect to i, conditioned on $\boldsymbol{\alpha}^{(k)}$ and $\mathcal{S}^{(k)}$, yields (A.20) by using the block-decomposition relationship (A.19) in the definition of $g^{\mathrm{PFW}}(\boldsymbol{\alpha}^{(k)}\,;\,\mathcal{S}^{(k)})$. This completes the proof for BCPFW.

In the case of BCAFW, let $\boldsymbol{d}_i$ be the chosen direction for block i (either a FW direction or an away direction). Then since $\boldsymbol{d}_i$ is chosen to maximize the inner product with $-\nabla_{(i)} f(\boldsymbol{\alpha}^{(k)})$, we have $\langle -\nabla_{(i)} f(\boldsymbol{\alpha}^{(k)}), \boldsymbol{d}_i\rangle \geq \frac{1}{2} g_i^{\mathrm{PFW}}(\boldsymbol{\alpha}^{(k)}\,;\,\mathcal{S}_i)$ (with a similar argument as used to get Eq. (6) in Lacoste-Julien and Jaggi, [2015b] for AFW). We then follow the same argument to derive (A.21), but using $\boldsymbol{d}_i$ instead of $(\boldsymbol{s}_{(i)} - \boldsymbol{v}_{(i)})$, which gives an extra $1/2$ factor as $\langle -\nabla_{(i)} f(\boldsymbol{\alpha}^{(k)}), \boldsymbol{d}_i\rangle$ is potentially only half of $g_i^{\mathrm{PFW}}(\boldsymbol{\alpha}^{(k)}\,;\,\mathcal{S}_i)$. Taking again the expectation of (A.21) completes the proof. □

Remark 8. The important condition that there is *no drop step* at $\boldsymbol{\alpha}^{(k)}$ in the BCPFW descent lemma 7 is to allow the bound (A.21) to hold for *any* $\gamma \in [0,1]$. Otherwise, let I be the (non-empty) set of blocks for which there would be a drop step at $\boldsymbol{\alpha}^{(k)}$ and let $\gamma_I := \min_{i \in I} \gamma_{\max}^{(i)}$, where $\gamma_{\max}^{(i)}$ is the maximum step size for block i. Then in this case we could only show the bound (A.21) for $\gamma \leq \gamma_I$. But γ_I could be arbitrarily small,[5] and so no significant progress is guaranteed in expectation in this case.

We also note that $C_f^{\mathrm{A}\,(i)}$ is used instead of $C_f^{(i)}$ in the lemma because $C_f^{(i)}$ can only be used with a feasible step from $\boldsymbol{\alpha}^{(k)}$, and thus again, the bound would only be valid for $\gamma \leq \gamma_{\max}$ (as bigger step sizes can take you outside of $\mathcal{M}^{(i)}$). If the gradient of f is Lipschitz continuous, one can bound $C_f^{\mathrm{A}\,(i)} \leq \tilde{L}_i \big(\operatorname{diam}_{\|\cdot\|_i} \mathcal{M}^{(i)}\big)^2$, which is almost the same bound as for $C_f^{(i)}$ used before, but with $\tilde{L}_i$ being the Lipschitz constant of $\nabla_{(i)} f$ for variations in the slightly

[5] Small maximum step sizes happen when the current coordinate value for an away corner is small (perhaps because a small step size was used by the line-search when they were added as a FW corner previously).

extended domain $\mathcal{M}^{(i)} + (\mathcal{M}^{(i)} - \mathcal{M}^{(i)})$ (with set addition in the Minkowski sense).

Theorem 3 (Geometric convergence of BCPFW)**.** *Consider running BCPFW (or BCAFW) on problem* (A.17) *where q is a strongly convex function and $\mathcal{M}$ is a block-separable polytope. Let $h_k := f(\boldsymbol{\alpha}^{(k)}) - f(\boldsymbol{\alpha}^*)$ be the suboptimality of the iterate k, where $\boldsymbol{\alpha}^*$ is any optimal solution to* (A.17)*. Conditioned on any iterate $\boldsymbol{\alpha}^{(k)}$ with active set $\mathcal{S}^{(k)}$ such that* no block could give a drop set *(as defined in the conditions for Lemma 7), then the expected new suboptimality decreases geometrically, that is:*

$$\mathbf{E}\Big[h_{k+1} \,|\, \boldsymbol{\alpha}^{(k)}, \mathcal{S}^{(k)}\Big] \leq (1-\rho) h_k, \tag{A.22}$$

with rate:

$$\rho := \tfrac{1}{2n} \min\{1, 2\tfrac{\tilde{\mu}_f}{C_f^{\mathrm{A}\otimes}}\} \text{ for the BCPFW algorithm}, \tag{A.23}$$

$$\rho := \tfrac{1}{4n} \min\{1, \tfrac{\tilde{\mu}_f}{C_f^{\mathrm{A}\otimes}}\} \text{ for the BCAFW algorithm}, \tag{A.24}$$

where $C_f^{\mathrm{A}\otimes} := \sum_{i=1}^n C_f^{\mathrm{A}\,(i)}$ is the total (away) curvature constant for problem (A.17) *as defined in Lemma 7, and $\tilde{\mu}_f$ is the generalized strong convexity constant for problem* (A.17) *as defined in Eq. (39) of Lacoste-Julien and Jaggi, [2015b] ($\tilde{\mu}_f$ is strictly greater than zero when q is strongly convex and $\mathcal{M}$ is a polytope).*

Proof. We first do the argument for BCPFW. Let $g_k := g^{\mathrm{PFW}}(\boldsymbol{\alpha}^{(k)}\,;\,\mathcal{S}^{(k)})$, and notice that $g_k \geq h_k$ always. Because we assume that there is no drop step at $\boldsymbol{\alpha}^{(k)}$, we can use the expected BCPFW descent lemma 7. By subtracting $f(\boldsymbol{\alpha}^*)$ on both side of the descent inequality (A.20), we get (for any $\gamma \in [0,1]$):

$$\mathbf{E}\Big[h_{k+1} \,|\, \boldsymbol{\alpha}^{(k)}, \mathcal{S}^{(k)}\Big] \leq h_k - \tfrac{\gamma}{n} g_k + \tfrac{\gamma^2}{2n} C_f^{\mathrm{A}\otimes}. \tag{A.25}$$

We can minimize the RHS of (A.25) with $\gamma^* = \frac{g_k}{C_f^{\mathrm{A}\otimes}}$. If $g_k > C_f^{\mathrm{A}\otimes}$ (i.e. $\gamma^* > 1$), then use $\gamma = 1$ in (A.25) to get:

$$\mathbf{E}\Big[h_{k+1} \,|\, \boldsymbol{\alpha}^{(k)}, \mathcal{S}^{(k)}\Big] \leq h_k - \tfrac{1}{2n} g_k \leq (1 - \tfrac{1}{2n}) h_k. \tag{A.26}$$

This gives a geometric rate of $\rho = \frac{1}{2n}$. So now suppose that $g_k \leq C_f^{\mathrm{A}\otimes}$ (so that $\gamma^* \leq 1$); putting $\gamma = \gamma^*$ in (A.25), we get:

$$\mathbf{E}\Big[h_{k+1} \,|\, \boldsymbol{\alpha}^{(k)}, \mathcal{S}^{(k)}\Big] \leq h_k - \tfrac{1}{2nC_f^{\mathrm{A}\otimes}}\, {g_k}^2. \tag{A.27}$$

We now use the key relationship between the suboptimality h_k and the PFW gap g_k derived in inequality (43) of Lacoste-Julien and Jaggi, [2015b] (which is true for any function f by definition of $\tilde{\mu}_f$ if we allow it to be zero):

$$h_k \leq \tfrac{{g_k}^2}{2\tilde{\mu}_f}. \tag{A.28}$$

Substituting (A.28) into (A.27), we get:

$$\mathbf{E}\Big[h_{k+1} \,|\, \boldsymbol{\alpha}^{(k)}, \mathcal{S}^{(k)}\Big] \leq (1 - \tfrac{\tilde{\mu}_f}{nC_f^{\mathrm{A}\otimes}})h_k, \tag{A.29}$$

which gives the $\rho = \tilde{\mu}_f / nC_f^{\mathrm{A}\otimes}$ rate. Taking the worst rate of (A.26) and (A.29) gives the rate (A.23), completing the proof for BCPFW.

In the case of BCAFW, Lemma 7 yields the inequality (A.25) but with an extra $1/2$ factor in front of g_k. Re-using the same argument as above, we get a rate of $\rho = 1/4n$ when $\gamma^* > 1$, and $\rho = \tilde{\mu}_f / 4nC_f^{\mathrm{A}\otimes}$ when $\gamma^* \leq 1$, showing (A.24) as required.

Finally, the fact that $\tilde{\mu}_f > 0$ when q is μ-strongly convex and $\mathcal{M}$ is a polytope comes from the lower bound given in Theorem 10 of Lacoste-Julien and Jaggi, [2015b] in terms of the *pyramidal width* of $\mathcal{M}$ (a strictly positive geometric quantity for polytopes), and the *generalized strong convexity* of f as defined in Lemma 9 of Lacoste-Julien and Jaggi, [2015b]. The generalized strong convexity of f is simply μ if f is μ-strongly convex. In the more general case of problem (A.17) where only q is μ-strongly convex, the generalized strong convexity depends both on μ and the *Hoffman constant* [Hoffman, 1952] associated with the linear system of problem (A.17). See Lacoste-Julien and Jaggi, [2015b] for more details, as well as Lemma 2.2 of Beck and Shtern, [2015]. □

Interpretation. Theorem 3 only guarantees progress of BCPFW or BCAFW when there would not be any drop step for *any* block i for the current iterate. For the batch AFW algorithm, one can easily lower bound the number of

times that these "good steps" can happen as a drop step reduces the size of the active set and thus cannot happen more than half of the time. On the other hand, in the block coordinate setting, we can be unlucky and always have one block that could give a drop step (while we pick other blocks during the algorithm, this bad block affects the expectation). This means that without a refined analysis of the drop step possibility, we cannot guarantee any progress in the worst case for BCPFW or BCAFW. As a safeguard, one can modify BCPFW or BCAFW so that it also has the option to do a standard BCFW step on a block if it yields better progress on f – this way, the algorithm inherits at least the (sublinear) convergence guarantees of BCFW.

Empirical linear convergence. In our experiments, we note that BCPFW always converged empirically, and had an empirical linear convergence rate for the SSVM objective when λ was big enough ($q(\cdot) = \frac{\lambda}{2}\|\cdot\|^2$ for the SSVM objective (6.3)). See Figure 6.4 for OCR-large (c) for example. We also tried the modified BCPFW algorithm where a choice is made between a FW step, a pairwise FW step or an away step on a block by picking the one which gives the biggest progress. We did not notice any significant speed-up for this modified method.

On the dimension of SSVM. Finally, we note that the rate constant ρ in Theorem 3 has an implicit dependence on the dimensionality (in particular, through the pyramidal width of $\mathcal{M}$). Lacoste-Julien and Jaggi, [2015b] showed that the largest possible pyramidal width of a polytope in dimension m (for a fixed diameter) is achieved by the probability simplex and is $\Theta(1/\sqrt{m})$. For the SSVM in the general form (6.3), the dimensionality of $\mathcal{M}^{(i)}$ is the number of possible structured outputs for input i, which is typically an exponentially large number, and thus the pyramidal width lower bound would be useless in this case. Fortunately, the matrix A (feature map) and vector $\boldsymbol{b}$ (loss function) are highly structured, and thus many $\boldsymbol{\alpha}$'s are mapped to the same objective value. For a feature mapping $\boldsymbol{\psi}_i(\boldsymbol{y})$ representing the sufficient statistics for an energy function associated with a graphical model (as for a conditional random field [Lafferty *et al.*, 2001]), then the SSVM objective is implicitly optimizing over the *marginal polytope* for the graphical model [Wainwright and Jordan, 2008]. More specifically, let A_i be the $d \times m_i$ submatrix of A associated with example i. Then we can write $A_i = B_i M_i$ where M_i is a $p \times m_i$ *marginalization* matrix, that is, $\boldsymbol{\mu} = M_i \boldsymbol{\alpha}$ is an element

of the marginal polytope for the graphical model, where p is the dimensionality of the marginal polytope – which is a polynomial number in the size of the graph, rather than exponential. By the affine invariance property of the FW-type algorithms, we can thus instead use the pyramidal width of the marginal polytope for the convergence analysis (and similarly for the Hoffman constant). Lacoste-Julien and Jaggi, [2015b] conjectured that the pyramidal width of a marginal polytope in dimension p was also $\Theta(1/\sqrt{p})$, thus giving a more reasonable bound for the convergence rate of BCPFW for SSVM.

A.5 Proof of Theorem 4 (convergence of BCFW with caching)

Theorem 4. *Let $\tilde{\nu} := \frac{1}{n}\nu \leq 1$. Then, for each $k \geq 0$, the iterate $\boldsymbol{\alpha}^{(k)}$ of Algorithm* **??** *satisfies $\mathbf{E}\Big[f(\boldsymbol{\alpha}^{(k)})\Big] - f(\boldsymbol{\alpha}^*) \leq \frac{2n}{\tilde{\nu}k+2n}\Big(\frac{1}{\tilde{\nu}}C_f^{\otimes} + h_0\Big)$ where $\boldsymbol{\alpha}^* \in \mathcal{M}$ is a solution of problem (6.3), $h_0 := f(\boldsymbol{\alpha}^{(0)}) - f(\boldsymbol{\alpha}^*)$ is the suboptimality at the starting point of the algorithm, $C_f^{\otimes} := \sum_{i=1}^n C_f^{(i)}$ is the sum of the curvature constants (see Definition 1) of f with respect to the domains $\mathcal{M}^{(i)}$ of individual blocks. The expectation is taken over the random choice of the sampled blocks at iterations $1, \ldots, k$ of the algorithm.*

Proof. The key observation of the proof consists in the fact that the combined oracle (the cache oracle in the case of a cache hit and the max oracle in the case of a cache miss) closely resembles an oracle with multiplicative approximation error [Lacoste-Julien *et al.*, 2013, Eq. (12) of Appendix C].

In the case of a cache hit, Definition 1 of curvature constant for any step size $\gamma \in [0, 1]$ gives us

$$\begin{aligned}
& f(\boldsymbol{\alpha}_\gamma^{(k+1)}) := f(\boldsymbol{\alpha}^{(k)} + \gamma(\boldsymbol{c}_{[i]} - \boldsymbol{\alpha}_{[i]}^{(k)})) \\
\leq\ & f(\boldsymbol{\alpha}^{(k)}) + \gamma\langle \boldsymbol{c}_{(i)} - \boldsymbol{\alpha}_{(i)}^{(k)}, \nabla_{(i)} f(\boldsymbol{\alpha}^{(k)})\rangle + \tfrac{\gamma^2}{2} C_f^{(i)} \\
=\ & f(\boldsymbol{\alpha}^{(k)}) - \gamma \hat{g}_i^{(k)} + \tfrac{\gamma^2}{2} C_f^{(i)} \\
\leq\ & f(\boldsymbol{\alpha}^{(k)}) - \gamma \tilde{\nu} g^{(k_0)} + \tfrac{\gamma^2}{2} C_f^{(i)}
\end{aligned}$$

where the corner $\boldsymbol{c}_{(i)} \in \mathcal{M}^{(i)}$ and its zero-padded version $\boldsymbol{c}_{[i]} \in \mathbb{R}^m$ are provided by the cache oracle, and $\tilde{\nu} = \frac{1}{n}\nu$ is the constant controlling the global part of the cache-hit criterion. In the case of a cache miss, similarly to Lemma 2, we get

$$f(\boldsymbol{\alpha}_\gamma^{(k+1)}) \leq f(\boldsymbol{\alpha}^{(k)}) - \gamma g_i^{(k)} + \tfrac{\gamma^2}{2} C_f^{(i)} \ .$$

Combining the two cases we get

$$f(\boldsymbol{\alpha}_\gamma^{(k+1)}) \leq f(\boldsymbol{\alpha}^{(k)}) - \gamma \tilde{g}_i^{(k)} + \tfrac{\gamma^2}{2} C_f^{(i)} \tag{A.30}$$

where

$$\tilde{g}_i^{(k)} := [i \text{ is a cache miss}] g_i^{(k)} + [i \text{ is a cache hit}] \tilde{\nu} g^{(k_0)}.$$

Subtracting $f(\boldsymbol{\alpha}^*)$ from both sides of (A.30) and taking the expectation of (A.30) w.r.t. the block index i we get

$$\mathbf{E}\Big[h(\boldsymbol{\alpha}_\gamma^{(k+1)}) \,|\, \boldsymbol{\alpha}^{(k)}\Big] \leq h(\boldsymbol{\alpha}^{(k)}) - \tfrac{\gamma}{n} \tilde{g}^{(k)} + \tfrac{\gamma^2}{2n} C_f^{\otimes} \tag{A.31}$$

where $h(\boldsymbol{\alpha}) := f(\boldsymbol{\alpha}) - f(\boldsymbol{\alpha}^*)$ is the suboptimality of the function f and $\tilde{g}^{(k)} := \sum_{i=1}^n \tilde{g}_i^{(k)}$. We know that the duality gap upper-bounds the suboptimality, i.e., $g(\boldsymbol{\alpha}) \geq h(\boldsymbol{\alpha})$, and that cache miss steps, as well as cache hit steps, always decrease suboptimality, i.e., $h(\boldsymbol{\alpha}^{(k)}) \leq h(\boldsymbol{\alpha}^{(k_0)})$.

If at iteration k there is at least one cache hit, then we can bound the quantity $\tilde{g}^{(k)}$ from below:

$$\tilde{g}^{(k)} \geq \tilde{\nu} g(\boldsymbol{\alpha}^{(k_0)}) \geq \tilde{\nu} h(\boldsymbol{\alpha}^{(k_0)}) \geq \tilde{\nu} h(\boldsymbol{\alpha}^{(k)}). \tag{A.32}$$

In the case of no cache hits, we have

$$\tilde{g}^{(k)} = g(\boldsymbol{\alpha}^{(k)}) \geq h(\boldsymbol{\alpha}^{(k)}) \geq \tilde{\nu} h(\boldsymbol{\alpha}^{(k)})$$

where the last inequality holds because $\tilde{\nu} \leq 1$. Applying the lower bound on $\tilde{g}^{(k)}$ to (A.31), we get

$$\begin{aligned} \mathbf{E}\Big[h(\boldsymbol{\alpha}_\gamma^{(k+1)}) \,|\, \boldsymbol{\alpha}^{(k)}\Big] \leq h(\boldsymbol{\alpha}^{(k)}) &- \tfrac{\gamma\tilde{\nu}}{n} h(\boldsymbol{\alpha}^{(k)}) \\ &+ \tfrac{\gamma^2}{2n} C_f^{\otimes}. \end{aligned} \tag{A.33}$$

Inequality (A.33) is identical to the inequality [Lacoste-Julien *et al.*, 2013, Eq. (20)] in the proof of convergence of BCFW with a multiplicative approximation error in the oracle. We recopy their argument below for reference to finish the proof. First, we take the expectation of (A.33) w.r.t. the choice of previous blocks:

$$\mathbf{E}\Big[h(\boldsymbol{\alpha}_{\gamma}^{(k+1)})\Big] \leq (1 - \tfrac{\gamma\tilde{\nu}}{n})\mathbf{E}\Big[h(\boldsymbol{\alpha}^{(k)})\Big] + \tfrac{\gamma^2}{2n}C_f^{\otimes}. \tag{A.34}$$

Following the proof of Theorem C.1 in Lacoste-Julien *et al.*, [2013], we prove the bound of Theorem 4 by induction. The induction hypothesis consists in inequality

$$\mathbf{E}\Big[h(\boldsymbol{\alpha}^{(k)})\Big] \leq \tfrac{2nC}{\tilde{\nu}k+2n} \text{ for } k \geq 0$$

where $C := \Big(\frac{1}{\tilde{\nu}}C_f^{\otimes} + h_0\Big)$.

The base-case $k = 0$ follows directly from $C \geq h_0$. We now prove the induction step. Assume that the hypothesis is true for a given $k \geq 0$. Let us now prove that the hypothesis is true for $k+1$. We use inequality (A.33) with the step size $\gamma_k := \frac{2n}{\tilde{\nu}k+2n} \in [0,1]$:

$$\begin{aligned}
\mathbf{E}\Big[h(\boldsymbol{\alpha}_{\gamma_k}^{(k+1)})\Big] &\leq (1 - \tfrac{\gamma_k\tilde{\nu}}{n})\mathbf{E}\Big[h(\boldsymbol{\alpha}^{(k)})\Big] + (\gamma_k)^2\tfrac{C\tilde{\nu}}{2n} \\
&= (1 - \tfrac{2\tilde{\nu}}{\tilde{\nu}k+2n})\mathbf{E}\Big[h(\boldsymbol{\alpha}^{(k)})\Big] + (\tfrac{2n}{\tilde{\nu}k+2n})^2\tfrac{C\tilde{\nu}}{2n} \\
&\leq (1 - \tfrac{2\tilde{\nu}}{\tilde{\nu}k+2n})\tfrac{2nC}{\tilde{\nu}k+2n} + (\tfrac{1}{\tilde{\nu}k+2n})^2 2nC\tilde{\nu}
\end{aligned}$$

where, in the first line, we use inequality $C_f^{\otimes} \leq C\tilde{\nu}$, and, in the last line, we use the induction hypothesis for $\mathbf{E}\Big[h(\boldsymbol{\alpha}^{(k)})\Big]$.

By rearranging the terms, we have

$$\begin{aligned}
\mathbf{E}\Big[h(\boldsymbol{\alpha}^{(k+1)})\Big] &\leq \tfrac{2nC}{\tilde{\nu}k+2n}\Big(1 - \tfrac{2\tilde{\nu}}{\tilde{\nu}k+2n} + \tfrac{\tilde{\nu}}{\tilde{\nu}k+2n}\Big) \\
&= \tfrac{2nC}{\tilde{\nu}k+2n}\tfrac{\tilde{\nu}k+2n-\tilde{\nu}}{\tilde{\nu}k+2n} \\
&\leq \tfrac{2nC}{\tilde{\nu}k+2n}\tfrac{\tilde{\nu}k+2n}{\tilde{\nu}k+2n+\tilde{\nu}} \\
&= \tfrac{2nC}{\tilde{\nu}(k+1)+2n},
\end{aligned}$$

which finishes the proof. □

A.6 BCFW for SSVM with box constraints

A.6.1 Problem with box constraints

Problem (6.1) can be equivalently rewritten as a quadratic program (QP) with an exponential number of constraints:

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{\xi}} \quad & \tfrac{\lambda}{2}\|\boldsymbol{w}\|^2 + \tfrac{1}{n}\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & \langle \boldsymbol{w}, \boldsymbol{\psi}_i(\boldsymbol{y})\rangle \geq L(\boldsymbol{y}_i, \boldsymbol{y}) - \xi_i \quad \forall i,\, \forall \boldsymbol{y} \in \mathcal{Y}_i \end{aligned} \tag{A.35}$$

where the slack variable ξ_i measures the surrogate loss for the i-th datapoint. Problem (A.35) is often referred to as the n-slack structured SVM with margin-rescaling [Joachims *et al.*, 2009, Optimization Problem 2].

In this section, we consider the problem (A.35) with additional box constraints on the parameter vector $\boldsymbol{w}$:

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{\xi}} \quad & \tfrac{\lambda}{2}\|\boldsymbol{w}\|^2 + \tfrac{1}{n}\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & \langle \boldsymbol{w}, \boldsymbol{\psi}_i(\boldsymbol{y})\rangle \geq L(\boldsymbol{y}_i, \boldsymbol{y}) - \xi_i \quad \forall i,\, \forall \boldsymbol{y} \in \mathcal{Y}_i, \\ & \boldsymbol{l} \preccurlyeq \boldsymbol{w} \preccurlyeq \boldsymbol{u}, \end{aligned} \tag{A.36}$$

where $\boldsymbol{l} \in \mathbb{R}^d$ and $\boldsymbol{u} \in \mathbb{R}^d$ denote the lower and upper bounds, respectively, and the symbol "$\preccurlyeq$" is the element-wise "less or equal to" sign. In the following, we assume that the box constraints are feasible, i.e., $\boldsymbol{l} \preccurlyeq \boldsymbol{u}$. Note that the following discussion can be directly extended to the case where only some dimension of the weight vector have to respect the box constraints. The Lagrangian of problem (A.36) can be written as

$$\begin{aligned} L(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}_l, \boldsymbol{\beta}_u) = \tfrac{\lambda}{2}\langle \boldsymbol{w}, \boldsymbol{w}\rangle + \tfrac{1}{n}\sum_{i=1}^{n}\xi_i \\ + \sum_{i\in[n],\, \boldsymbol{y}\in\mathcal{Y}_i} \tfrac{1}{n}\alpha_i(\boldsymbol{y})\left(-\xi_i + \langle \boldsymbol{w}, -\boldsymbol{\psi}_i(\boldsymbol{y})\rangle + L_i(\boldsymbol{y})\right) \\ + \lambda\langle \boldsymbol{\beta}_u, \boldsymbol{w} - \boldsymbol{u}\rangle + \lambda\langle \boldsymbol{\beta}_l, -\boldsymbol{w} + \boldsymbol{l}\rangle \end{aligned} \tag{A.37}$$

where $\boldsymbol{\beta}_l \in \mathbb{R}^d$ and $\boldsymbol{\beta}_u \in \mathbb{R}^d$ are the dual variables associated with the lower and upper bound constraints, respectively. From the KKT conditions, we obtain

$$\boldsymbol{w} = A\boldsymbol{\alpha} - (\boldsymbol{\beta}_u - \boldsymbol{\beta}_l), \tag{A.38}$$

$$\sum_{\boldsymbol{y}\in\mathcal{Y}_i} \alpha_i(\boldsymbol{y}) = 1 \quad \forall i \in [n]. \tag{A.39}$$

Finally, the dual of problem (A.36) (here written in a minimization form) can be written as follows:

$$\begin{aligned} \min_{\substack{\boldsymbol{\alpha}\in\mathbb{R}^m \\ \boldsymbol{\alpha}\succcurlyeq 0}} f(\boldsymbol{\alpha}, \boldsymbol{\beta}_l, \boldsymbol{\beta}_u) := \tfrac{\lambda}{2}\left\|A\boldsymbol{\alpha} - (\boldsymbol{\beta}_u - \boldsymbol{\beta}_l)\right\|^2 - \boldsymbol{b}^\mathsf{T}\boldsymbol{\alpha} \\ + \lambda(\boldsymbol{\beta}_u^\mathsf{T}\boldsymbol{u} - \boldsymbol{\beta}_l^\mathsf{T}\boldsymbol{l}) \\ \text{s.t.} \sum_{\boldsymbol{y}\in\mathcal{Y}} \alpha_i(\boldsymbol{y}) = 1 \quad \forall i \in [n], \\ \text{and } \boldsymbol{\beta}_u \succcurlyeq 0, \boldsymbol{\beta}_l \succcurlyeq 0. \end{aligned} \tag{A.40}$$

A modified block optimization method. Ideally, we should optimize $f(\boldsymbol{\alpha}, \boldsymbol{\beta}_l, \boldsymbol{\beta}_u)$ jointly w.r.t. all the dual variables. This task is not directly suitable for the Frank-Wolfe approach as the domain for $\boldsymbol{\beta}_l$ and $\boldsymbol{\beta}_u$ is unbounded. However, joint optimization w.r.t. $\boldsymbol{\beta}_l$ and $\boldsymbol{\beta}_u$ with $\boldsymbol{\alpha}$ kept fixed can be done in closed form. After that, optimization w.r.t. $\boldsymbol{\alpha}$ can be performed using the Frank-Wolfe blockwise approach. Therefore, we resort to optimizing in a blockwise fashion: we iterate either a batch FW or a BCFW step on $\boldsymbol{\alpha}$ with an exact block-update on $(\boldsymbol{\beta}_u, \boldsymbol{\beta}_l)$. As we will see below, this principled approach is similar to a commonly used heuristic of truncating the value of $\boldsymbol{w}$ to make it feasible during an algorithm which works on the dual. In fact, our approach will be equivalent to run FW or BCFW with a truncation making $\boldsymbol{w}(\boldsymbol{\alpha})$ feasible after each FW step, but with a *change in the optimal step-size computation* (line 8 in Algorithm 8 for FW; line 7 in Algorithm 9 for BCFW) due to the different nature of the optimization problem.

Algorithm 8 Batch Frank-Wolfe algorithm for structured SVM with box constraints

Let $\boldsymbol{v}^{(0)} := \mathbf{0}$; $\ell^{(0)} := 0$
$\boldsymbol{w}^{(0)} := [\boldsymbol{v}^{(0)}]_{\boldsymbol{l}}^{\boldsymbol{u}}$ *// truncation to the feasible set*
for $k := 0, \ldots, \infty$ **do**
  for $i := 1, \ldots, n$ **do**
    Solve $\boldsymbol{y}_i^* := \underset{\boldsymbol{y} \in \mathcal{Y}_i}{\operatorname{argmax}} \; H_i(\boldsymbol{y}; \boldsymbol{w}^{(k)})$
  end for
  Let $\boldsymbol{v}_{\boldsymbol{s}} := \sum_{i=1}^{n} \frac{1}{\lambda n} \boldsymbol{\psi}_i(\boldsymbol{y}_i^*)$ and $\ell_{\boldsymbol{s}} := \frac{1}{n} \sum_{i=1}^{n} L_i(\boldsymbol{y}_i^*)$
  Let $\gamma := \frac{\lambda(\boldsymbol{v}^{(k)} - \boldsymbol{v}_{\boldsymbol{s}})^\top \boldsymbol{w}^{(k)} - \ell^{(k)} + \ell_{\boldsymbol{s}}}{\lambda \|\boldsymbol{v}^{(k)} - \boldsymbol{v}_{\boldsymbol{s}}\|^2}$ and clip to $[0, 1]$
  Update $\boldsymbol{v}^{(k+1)} := (1 - \gamma)\boldsymbol{v}^{(k)} + \gamma \, \boldsymbol{v}_{\boldsymbol{s}}$
    and $\ell^{(k+1)} := (1 - \gamma)\ell^{(k)} + \gamma \, \ell_{\boldsymbol{s}}$
    and $\boldsymbol{w}^{(k+1)} := [\boldsymbol{v}^{(k+1)}]_{\boldsymbol{l}}^{\boldsymbol{u}}$
end for

A.6.2 Optimizing w.r.t slack variables

The optimization w.r.t. $\boldsymbol{\beta}_u$ with $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}_l$ fixed can be easily solved in closed form via a simple thresholding operation:

$$\boldsymbol{\beta}_u^* = [A\boldsymbol{\alpha} + \boldsymbol{\beta}_l - \boldsymbol{u}]_+ \, . \tag{A.41}$$

The optimization w.r.t. $\boldsymbol{\beta}_l$ with $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}_u$ fixed is analogous:

$$\boldsymbol{\beta}_l^* = [-A\boldsymbol{\alpha} + \boldsymbol{\beta}_u + \boldsymbol{l}]_+ \, . \tag{A.42}$$

Denote the p-th variable of $\boldsymbol{\beta}_u$ and $\boldsymbol{\beta}_l$ with $\boldsymbol{\beta}_u(p)$ and $\boldsymbol{\beta}_l(p)$, respectively. For any index p, both $\boldsymbol{\beta}_u(p)$ and $\boldsymbol{\beta}_l(p)$ cannot be nonzero simultaneously, because if one of the constraints is violated (either the upper or the lower bound), then the other constraint must be satisfied. Hence, $\boldsymbol{\beta}_u(p) \neq 0$ implies $\boldsymbol{\beta}_l(p) = 0$ and vice versa. Therefore, the final update equations can equivalently be written as

$$\boldsymbol{\beta}_u^*(\boldsymbol{\alpha}) = [A\boldsymbol{\alpha} - \boldsymbol{u}]_+ \, , \tag{A.43}$$

$$\boldsymbol{\beta}_l^*(\boldsymbol{\alpha}) = [-A\boldsymbol{\alpha} + \boldsymbol{l}]_+ \, . \tag{A.44}$$

Algorithm 9 Block-coordinate Frank-Wolfe algorithm for structured SVM with box constraints

Let $\boldsymbol{v}^{(0)} := \boldsymbol{v}_i^{(0)} := \mathbf{0}$; $\ell^{(0)} := \ell_i^{(0)} := 0$;
$\boldsymbol{w}^{(0)} := [\boldsymbol{v}^{(0)}]_{\boldsymbol{l}}^{\boldsymbol{u}}$ *// truncation to the feasible set*
for $k := 0, \dots, \infty$ **do**
  Pick i at random in $\{1, \dots, n\}$
  Solve $\boldsymbol{y}_i^* := \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}_i} H_i(\boldsymbol{y}; \boldsymbol{w}^{(k)})$
  Let $\boldsymbol{v}_{\boldsymbol{s}} := \frac{1}{\lambda n} \boldsymbol{\psi}_i(\boldsymbol{y}_i^*)$ and $\ell_{\boldsymbol{s}} := \frac{1}{n} L_i(\boldsymbol{y}_i^*)$
  Let $\gamma := \frac{\lambda(\boldsymbol{v}_i^{(k)} - \boldsymbol{v}_{\boldsymbol{s}})^\top \boldsymbol{w}^{(k)} - \ell_i^{(k)} + \ell_{\boldsymbol{s}}}{\lambda \|\boldsymbol{v}_i^{(k)} - \boldsymbol{v}_{\boldsymbol{s}}\|^2}$ and clip to $[0, 1]$
  Update $\boldsymbol{v}_i^{(k+1)} := (1-\gamma)\boldsymbol{v}_i^{(k)} + \gamma\, \boldsymbol{v}_{\boldsymbol{s}}$
    and $\ell_i^{(k+1)} := (1-\gamma)\ell_i^{(k)} + \gamma\, \ell_{\boldsymbol{s}}$
  Update $\boldsymbol{v}^{(k+1)} := \boldsymbol{v}^{(k)} + \boldsymbol{v}_i^{(k+1)} - \boldsymbol{v}_i^{(k)}$
    and $\ell^{(k+1)} := \ell^{(k)} + \ell_i^{(k+1)} - \ell_i^{(k)}$
  Let $\boldsymbol{w}^{(k+1)} := [\boldsymbol{v}^{(k+1)}]_{\boldsymbol{l}}^{\boldsymbol{u}}$
end for

Introducing $\boldsymbol{v}(\boldsymbol{\alpha}) := A\boldsymbol{\alpha}$, we get $\boldsymbol{\beta}_u^* = [\boldsymbol{v} - \boldsymbol{u}]_+$ and $\boldsymbol{\beta}_l^* = [-\boldsymbol{v} + \boldsymbol{l}]_+$. Hence, the operation $\boldsymbol{w} = \boldsymbol{v} - (\boldsymbol{\beta}_u^* - \boldsymbol{\beta}_l^*)$ is simply the projection (truncation) of $\boldsymbol{v}$ on the feasible set defined by the upper and lower bounds. In the final algorithm, we maintain $\boldsymbol{v}(\boldsymbol{\alpha})$ and directly update the primal variables $\boldsymbol{w}$ without updating $\boldsymbol{\beta}_u$ and $\boldsymbol{\beta}_l$.

A.6.3 Batch setting: optimization w.r.t iterate

When the variables $\boldsymbol{\beta}_u$ and $\boldsymbol{\beta}_l$ are fixed, the convex problem (A.40) has a compact domain and so we can use the Frank-Wolfe algorithm on it. In the following, we highlight the differences with the setting without box constraints. We denote by $\boldsymbol{w}(\boldsymbol{\alpha})$ the truncation of $\boldsymbol{v}(\boldsymbol{\alpha})$ on the box constraints, i.e.,

$$\boldsymbol{w}(\boldsymbol{\alpha}) := \boldsymbol{v}(\boldsymbol{\alpha}) - (\boldsymbol{\beta}_u^*(\boldsymbol{\alpha}) - \boldsymbol{\beta}_l^*(\boldsymbol{\alpha}))\,. \tag{A.45}$$

The derivations below assume that $\boldsymbol{\beta}_u$ and $\boldsymbol{\beta}_l$ are fixed to their optimal values $\boldsymbol{\beta}_u^*(\boldsymbol{\alpha})$ and $\boldsymbol{\beta}_l^*(\boldsymbol{\alpha})$ for a specific $\boldsymbol{\alpha}$.

Linear subproblem. The Frank-Wolfe linear subproblem can be written as

$$\boldsymbol{s} = \operatorname*{argmin}_{\boldsymbol{s}' \in \mathcal{M}} \langle \boldsymbol{s}', \nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}, \boldsymbol{\beta}_l, \boldsymbol{\beta}_u) \rangle \tag{A.46}$$

where $\nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}, \boldsymbol{\beta}_l, \boldsymbol{\beta}_u)$ can be easily computed:

$$\begin{aligned}\nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}, \boldsymbol{\beta}_l, \boldsymbol{\beta}_u) &= \lambda A^\mathsf{T}(A\boldsymbol{\alpha} - (\boldsymbol{\beta}_u - \boldsymbol{\beta}_l)) - \boldsymbol{b} \\ &= \lambda A^\mathsf{T}\boldsymbol{w} - \boldsymbol{b}. \end{aligned} \tag{A.47}$$

Analogously to the problem without box constraints, the linear subproblem used by the Frank-Wolfe algorithm is equivalent to the loss-augmented decoding subproblem (6.2). The update of $\boldsymbol{\alpha}$ can be made using the corner $\boldsymbol{s}$. In what follows, we show that this update can be performed without explicitly keeping the dual variables at the cost of storing the extra vector $\boldsymbol{v}$.

The duality gap. The Frank-Wolfe gap for problem (A.40) can be written as

$$\begin{aligned} g(\boldsymbol{\alpha}) := \quad & \max_{\boldsymbol{s}' \in \mathcal{M}} \langle \boldsymbol{\alpha} - \boldsymbol{s}', \nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}, \boldsymbol{\beta}_l, \boldsymbol{\beta}_u) \rangle \\ = \quad & (\boldsymbol{\alpha} - \boldsymbol{s})^\mathsf{T}(\lambda A^\mathsf{T}(A\boldsymbol{\alpha} - (\boldsymbol{\beta}_u - \boldsymbol{\beta}_l)) - \boldsymbol{b}) \\ = \quad & \lambda(\boldsymbol{v} - \boldsymbol{v}_{\boldsymbol{s}})^\mathsf{T}\boldsymbol{w} - \boldsymbol{b}^\mathsf{T}\boldsymbol{\alpha} + \boldsymbol{b}^\mathsf{T}\boldsymbol{s} \end{aligned}$$

where $\boldsymbol{v}_{\boldsymbol{s}} := A\boldsymbol{s}$. Below, we prove that the Frank-Wolfe duality gap $g(\boldsymbol{\alpha})$ for the problem (A.40) when $\boldsymbol{\beta}_u$ and $\boldsymbol{\beta}_l$ are fixed at their current optimal value for the current $\boldsymbol{\alpha}$ equals to a Lagrange duality gap, analogously to the case without box constraints [Lacoste-Julien *et al.*, 2013, Appendix B.2].[6]

Proof. Consider the difference between the primal objective of (A.36) at $\boldsymbol{w} := A\boldsymbol{\alpha} - (\boldsymbol{\beta}_u - \boldsymbol{\beta}_l)$ with the optimal slack variables $\boldsymbol{\xi}$ and the dual objective of (A.40) at $\boldsymbol{\alpha}$ (in the maximization form). We get

$$\begin{aligned} g_{\text{Lag.}}(\boldsymbol{w}, \boldsymbol{\alpha}) &= \tfrac{\lambda}{2}\boldsymbol{w}^\mathsf{T}\boldsymbol{w} + \tfrac{1}{n}\sum_{i=1}^{n} \tilde{H}_i(\boldsymbol{w}) \\ &\quad - \left(\boldsymbol{b}^\mathsf{T}\boldsymbol{\alpha} - \tfrac{\lambda}{2}\boldsymbol{w}^\mathsf{T}\boldsymbol{w} - \lambda(\boldsymbol{\beta}_u^\mathsf{T}\boldsymbol{u} - \boldsymbol{\beta}_l^\mathsf{T}\boldsymbol{l})\right) \\ &= \lambda\boldsymbol{w}^\mathsf{T}\boldsymbol{w} - \boldsymbol{b}^\mathsf{T}\boldsymbol{\alpha} + \tfrac{1}{n}\sum_{i=1}^{n} \max_{\boldsymbol{y} \in \mathcal{Y}_i} H_i(\boldsymbol{y}; \boldsymbol{w}) \\ &\quad + \lambda(\boldsymbol{\beta}_u^\mathsf{T}\boldsymbol{u} - \boldsymbol{\beta}_l^\mathsf{T}\boldsymbol{l}) \ . \end{aligned}$$

[6] We stress that this relationship is only valid for the pair $\boldsymbol{w} = \boldsymbol{w}(\boldsymbol{\alpha})$ in the primal, and $\boldsymbol{\beta}_u = \boldsymbol{\beta}_u^*(\boldsymbol{\alpha}), \boldsymbol{\beta}_l = \boldsymbol{\beta}_l^*(\boldsymbol{\alpha})$ in the dual.

Recalling

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\max_{\boldsymbol{y}\in\mathcal{Y}_i} H_i(\boldsymbol{y};\boldsymbol{w}) &= \max_{\boldsymbol{s}'\in\mathcal{M}} -\boldsymbol{s}'^{\mathsf{T}}\nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha},\boldsymbol{\beta}_l,\boldsymbol{\beta}_u)\\ &= -\boldsymbol{s}^{\mathsf{T}}\nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha},\boldsymbol{\beta}_l,\boldsymbol{\beta}_u),\end{aligned}$$

we can write

$$\begin{aligned}g_{\text{Lag.}}(\boldsymbol{w},\boldsymbol{\alpha}) &= \lambda\boldsymbol{w}^{\mathsf{T}}(A\boldsymbol{\alpha}-(\boldsymbol{\beta}_u-\boldsymbol{\beta}_l)) - \boldsymbol{b}^{\mathsf{T}}\boldsymbol{\alpha}\\ &\qquad - \boldsymbol{s}^{\mathsf{T}}\nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha},\boldsymbol{\beta}_l,\boldsymbol{\beta}_u) + \lambda(\boldsymbol{\beta}_u^{\mathsf{T}}\boldsymbol{u}-\boldsymbol{\beta}_l^{\mathsf{T}}\boldsymbol{l})\\ &= (\lambda\boldsymbol{w}^{\mathsf{T}}A-\boldsymbol{b}^{\mathsf{T}})\boldsymbol{\alpha} - \boldsymbol{s}^{\mathsf{T}}\nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha},\boldsymbol{\beta}_l,\boldsymbol{\beta}_u)\\ &\qquad - \lambda(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\beta}_u-\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\beta}_l) + \lambda(\boldsymbol{\beta}_u^{\mathsf{T}}\boldsymbol{u}-\boldsymbol{\beta}_l^{\mathsf{T}}\boldsymbol{l})\\ &= (\boldsymbol{\alpha}-\boldsymbol{s})^{\mathsf{T}}\nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha},\boldsymbol{\beta}_l,\boldsymbol{\beta}_u)\\ &\qquad + \lambda(\boldsymbol{\beta}_u^{\mathsf{T}}(\boldsymbol{u}-\boldsymbol{w}) - \boldsymbol{\beta}_l^{\mathsf{T}}(\boldsymbol{l}-\boldsymbol{w})).\end{aligned}$$

As we assumed that $\boldsymbol{\beta}_u = \boldsymbol{\beta}_u^*(\boldsymbol{\alpha})$ and $\boldsymbol{\beta}_l = \boldsymbol{\beta}_l^*(\boldsymbol{\alpha})$, we have $\boldsymbol{\beta}_u^{*\mathsf{T}}(\boldsymbol{u}-\boldsymbol{w}) = 0$ and $\boldsymbol{\beta}_l^{*\mathsf{T}}(\boldsymbol{l}-\boldsymbol{w}) = 0$, and thus

$$g_{\text{Lag}}(\boldsymbol{w},\boldsymbol{\alpha},\boldsymbol{\beta}_u^*,\boldsymbol{\beta}_l^*) = g(\boldsymbol{\alpha}).$$

□

Line-Search. Line search can be performed efficiently using $\gamma_{\text{opt}\boldsymbol{s}} := \frac{\langle\boldsymbol{\alpha}-\boldsymbol{s},\nabla f(\boldsymbol{\alpha})\rangle}{\lambda\|A(\boldsymbol{\alpha}-\boldsymbol{s})\|^2} = \frac{g(\boldsymbol{\alpha})}{\lambda\|\boldsymbol{v}-\boldsymbol{v}_{\boldsymbol{s}}\|^2}$.

Algorithm. Algorithm 8 contains the batch Frank-Wolfe algorithm with box constraints. The main idea consists in maintaining the vector $\boldsymbol{v} := A\boldsymbol{\alpha}$ in order to perform all the updates of $\boldsymbol{\alpha}$ using only the primal variables. Optimization w.r.t. $\boldsymbol{\beta}_l$ and $\boldsymbol{\beta}_u$ corresponds to the truncation of $\boldsymbol{v}$. Given these variables, the gap and the optimal step size are easy to compute.

A.6.4 The block-coordinate setting

Algorithm 9 describes the block-coordinate version of the Frank-Wolfe algorithm with box constraints. The algorithm is obtained from the batch version (Algorithm 8) in exactly the same way as BCFW (Algorithm 3) is

obtained from the batch Frank-Wolfe method [Lacoste-Julien *et al.*, 2013, Algorithm 2].

A.7 Dataset description

In our experiments, we use four structured prediction datasets: OCR Taskar *et al.*, 2003 for character recognition; CoNLL Sang and Buchholz, 2000 for text chunking; HorseSeg Kolesnikov *et al.*, 2014 for binary image segmentation; LSP Johnson and Everingham, 2010 for pose estimation. In this section, we provide a detailled description of the datasets and the corresponding models that were summarized in Table 6.1. For the OCR and CoNLL datasets, the features and models described below are exactly the same as used by Lacoste-Julien *et al.*, [2013]; we give a detailed description for reference. For HorseSeg and LSP, we had to build the models ourselves from previous work referenced in the relevant section.

A.7.1 OCR

The Optical Character Recognition (OCR) dataset collected by Taskar *et al.*, [2003] poses the task of recognizing English words from sequences of handwritten symbols represented by binary images. The average length of sequences equals 7.6 symbols. For a sequence of length T, the input feature representation $\boldsymbol{x}$ consists of T binary images of size 16×8. The output object $\boldsymbol{y}$ is a sequence of length T with each symbol taking 26 possible values.

The OCR dataset contains $6,877$ words. In the small version, 626 words are used for training and the rest for testing. In the large version, $6,251$ words are used for training and the rest for testing.

The prediction model is a chain. The feature map $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ contains features of three types: emission, transition and bias. The $16 \times 8 \times 26$ emission features count the number of times along the chain a specific position of the 16×8 binary image equals 1 when associated with a specific output symbol. The 26×26 transition features count the number of times one symbol follows

another. The 26×3 bias features represent three biases for each element of the output alphabet: one model bias, and a bias for when the letter appears at the beginning or at the end of the sequence.

As the structured error between output vectors $L(\boldsymbol{y}_i, \boldsymbol{y})$, the OCR dataset uses the Hamming distance normalized by the length of the sequences. The loss-augmented structured score $H_i(\boldsymbol{y}; \boldsymbol{w})$ is a function with unary and pairwise potentials defined on a chain and is exactly maximized with the dynamic programming algorithm of Viterbi, [1967].

A.7.2 CoNLL

The CoNLL dataset released by Sang and Buchholz, [2000] poses the task of text chunking. Text chunking, also known as shallow parsing Sha and Pereira, [2003], consists in dividing the input text into syntactically related non-overlapping groups of words, called phrase or chunks. The task of text chunking can be cast as a sequence labeling where a sequence of labels $\boldsymbol{y}$ is predicted from an input sequence of tokens $\boldsymbol{x}$. For a given token x_t (a word with its corresponding part-of-speech tag), the associated label y_t gives the type of phrase the token belongs to, i.e., says whether or not it corresponds to the beginning of a chunk, or encodes the fact that the token does not belong to a chunk.

The CoNLL dataset contains $8,936$ training English sentences extracted from the Wall Street Journal part of the Penn Treebank II Marcus *et al.*, [1993]. Each output label y_t can take up to 22 different values.

We use the feature map $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})$ proposed by Sha and Pereira, [2003]. First, for each position t of the input sequence $\boldsymbol{x}$, we construct a unary feature representation, containing the local information. We start with extracting several attributes representing the words and the part-of-speech tags at the positions neighboring to t.[7] Each attribute is encoded with an indicator vector of length equal to either the dictionary size or the number of part-of-speech tags. We concatenate these vectors to get a unary feature representation,

[7] We extract the attributes with the CRFsuite library Okazaki, [2007] and refer to its documentation for the exact list of attributes: `http://www.chokkan.org/software/crfsuite/tutorial.html`.

which is a sparse binary vector of dimensionality $74,658$. Note that these representation can be precomputed outside of the training process.

Given a labeling $\boldsymbol{y}$ and the unary representations, the feature map $\boldsymbol{\phi}$ is constructed by concatenating features of three types (as in the chain model for OCR): emission, transition and bias. The $74,658 \times 22$ emission features count the number of times each coordinate of the unary representation of token x_t is nonzero and the corresponding output variable y_t is assigned a particular value. The transition map of size 22×22 encodes the number of times one label follows another in the output $\boldsymbol{y}$. The 22×3 bias features encode biases for all the possible values of the output variables and, specifically, biases for the first and the last variables.

As in the OCR task, the structured error $L(\boldsymbol{y}_i, \boldsymbol{y})$ is the normalized Hamming distance and thus the max-oracle can be efficiently implemented using the dynamic programming algorithm of Viterbi, [1967].

A.7.3 HorseSeg

The HorseSeg dataset[8] released by Kolesnikov *et al.*, [2014] poses the task of object/background segmentation of images containing horses, i.e., assigning a label "horse" or "background" to each pixel of the image. HorseSeg contains $25,438$ training images, 147 of which are manually annotated, $5,974$ annotations are constructed from object bounding boxes by the automatic method of Guillaumin *et al.*, [2014], while the remaining $19,317$ annotations were constructed by the same method but without any human supervision. The test set of HorseSeg consists of 241 images with manual annotations. In our experiments, we use training sets of three different sizes: 147 images for HorseSeg-small, $6,121$ images for HorseSeg-medium and $25,438$ for HorseSeg-large.

In addition to images and their pixel-level annotations, Kolesnikov *et al.*, [2014] released[9] oversegmentations (superpixels) of the images precomputed with the SLIC algorithm Achanta *et al.*, [2012] and the unary features of each superpixel computed similarly to the work of Lempitsky *et al.*, [2011].

[8] https://pub.ist.ac.at/~akolesnikov/HDSeg/HDSeg.tar
[9] https://pub.ist.ac.at/~akolesnikov/HDSeg/data.tar

On average, each image contains 147 superpixels. The $1,969$ unary features include 1 constant feature, 512-bin histograms of densely sampled visual SIFT words Lowe, [2004], 128-bin histograms of RGB colors, 16-bin histograms of locations (each pixel of a region of interest is matched to a cell of the 4×4 uniform grid). The three aforementioned histograms are computed on the superpixels themselves, on the superpixels together with their neighboring superpixels, and on the second-order neighborhoods.

For each pair of adjacent superpixels, we construct 100 pairwise features: a constant feature; and quantities $\exp(-\eta d_{pq})$ where d_{pq} is a χ^2-distance between 9 pairs of the corresponding histograms of each type for neighbors p and q, and η is a parameter taking 11 values from the set $2^{-5}, 2^{-4}, \ldots, 2^5$.

The structured feature map is defined in such a way that the corresponding structured score function contains unary and pairwise Potts potentials

$$\begin{aligned}\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y})\rangle = &\sum_{p \in \mathcal{V}_i} \langle \boldsymbol{w}_U, \boldsymbol{x}^U_{i,p}\rangle([y_p = 1] - [y_p = 0]) \\ &+ \sum_{\{p,q\} \in \mathcal{E}_i} \langle \boldsymbol{w}_P, \boldsymbol{x}^P_{i,pq}\rangle [y_p \neq y_p]\end{aligned}$$

where the vector $\boldsymbol{x}_i = ((\boldsymbol{x}^U_{i,p})_{p \in \mathcal{V}_i}, (\boldsymbol{x}^P_{i,pq})_{\{p,q\} \in \mathcal{E}_i})$ denotes all the features of image i, the vector $\boldsymbol{y} = (y_p)_{p \in \mathcal{V}_i} \in \{0,1\}^{\mathcal{V}_i}$ is a feasible labeling, the set $\mathcal{V}_i$ is the set of the superpixels of the image i, and the set $\mathcal{E}_i$ represents the adjacency graph.

The structured error is measured with a Hamming loss with class-normalized penalties

$$L(\boldsymbol{y}_i, \boldsymbol{y}) = \sum_{p \in \mathcal{V}_i} \omega_{y_{i,p}} [y_{i,p} \neq y_p]$$

where $\boldsymbol{y}_i = (y_{i,p})_{p \in \mathcal{V}_i}$ is the labeling of superpixels closest to the ground-truth annotation and the weights ω_0 and ω_1 are proportional to the ground-truth area of each class.

The loss-augmented score function

$$L(\boldsymbol{y}_i, \boldsymbol{y}) - \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y})\rangle$$

is a discrete function defined w.r.t. a cyclic graph and can be maximized in polynomial time when it is supermodular. By construction, all our pairwise features are nonnegative, so we can ensure supermodularity by adding positivity constraints on the weights corresponding to the pairwise features $\boldsymbol{w}_P \succcurlyeq 0$. The version of BCFW with positivity constraints is described in Appendix A.6.

The discrete optimization problem arising in the max-oracle is solved by the min-cut/max-flow algorithm of Boykov and Kolmogorov, [2004]. The running time or the max-flow is small compared to the operations with the features required to compute the potentials.

A.7.4 LSP

The Leeds Sports Pose (LSP) dataset introduced by Johnson and Everingham, 2010 poses the tasks of full body pose estimation from still images containing sports activities. Based on the input image with a centered prominent person, the task is to predict the locations of 14 body-parts (joints), e.g., "left knee" or "right ankle".

We cast the task of pose estimation as a structured prediction problem and build our model based on the work of Chen and Yuille, [2014], which is one of the state-of-the-art methods for pose estimation. First, we construct an acyclic graph where the nodes $p \in \mathcal{V}$ correspond to the different body-parts. The set of body parts is extended from the original 14 parts of interest by the midway points to get the 26 nodes of the graph. Second, the graph is converted into the directed one by utilizing the arcs of both orientations for each original edge. We denote the resulting graph by $\mathcal{G} = (\mathcal{V}, \vec{\mathcal{E}})$.

In the model of Chen and Yuille, [2014], each node $p \in \mathcal{V}$ has a variable $\boldsymbol{l}_p \in \mathcal{P} \subset \mathbb{R}^2$ denoting the spatial position of the corresponding joint that belongs to a finite set of possibilities $\mathcal{P}$; each arc $(p, q) \in \vec{\mathcal{E}}$ has a variable $t_{pq} \in \mathcal{T} = \{1, \ldots, 13\}$ representing the type of spacial relationship between the two nodes. The output variable $\boldsymbol{y}$ is constructed by concatenating the unary and pairwise variables $\boldsymbol{y} = \Big((\boldsymbol{l}_p)_{p\in\mathcal{V}},\ (t_{pq})_{(p,q)\in\vec{\mathcal{E}}}\Big)$.

The structure score function is a function of discrete variables $\boldsymbol{l}_p$ and t_{pq} that is defined w.r.t. the graph $\mathcal{G}$:

$$\begin{aligned}\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y})\rangle = &\sum_{p\in\mathcal{V}} w_{U,p}\phi_p^U(\boldsymbol{I}_i, \boldsymbol{l}_p) \\ &+ \sum_{(p,q)\in\mathcal{E}} w_{T,pq}\phi_{pq}^P(\boldsymbol{I}_i, \boldsymbol{l}_p, t_{pq}) \\ &+ \sum_{(p,q)\in\mathcal{E}} \langle \boldsymbol{w}_{P,pq,t_{pq}}, \Delta(\boldsymbol{l}_p - \boldsymbol{l}_q - \boldsymbol{r}_{pq}^{t_{pq}})\rangle.\end{aligned}$$

Here, the input $\boldsymbol{x}_i = \left(\boldsymbol{I}_i, (\boldsymbol{r}_{pq}^t)_{(p,q)\in\vec{\mathcal{E}}}^{t\in\mathcal{T}}\right)$ consists of the original image $\boldsymbol{I}_i$ and the mean relative positions $\boldsymbol{r}_{pq}^t \in \mathbb{R}^2$ of each type of spatial relationship corresponding to each arc $(p,q) \in \vec{\mathcal{E}}$. Functions $\phi_p^U(\boldsymbol{I}_i, \cdot)$ and $\phi_{pq}^P(\boldsymbol{I}_i, \cdot, \cdot)$ compute the scores for each possible value of the discrete variables $\boldsymbol{l}_p$ and $(\boldsymbol{l}_p, t_{pq})$, respectively. The vector-valued function $\Delta(\boldsymbol{l}) = (l_1, l_1^2, l_2, l_2^2)$, $\boldsymbol{l} \in \mathbb{R}^2$, measures different types of mismatch between the preferred relative displacement $\boldsymbol{r}_{pq}^{t_{pq}}$ and the displacement $\boldsymbol{l}_p - \boldsymbol{l}_q$ coming from the labeling $\boldsymbol{y}$. The vector $\boldsymbol{w} = \left((w_{U,p})_{p\in\mathcal{V}}, (w_{T,pq})_{(p,q)\in\vec{\mathcal{E}}}, (\boldsymbol{w}_{P,pq,t})_{(p,q)\in\vec{\mathcal{E}}}^{t\in\mathcal{T}}\right)$ is the joint vector of parameters learned by structured SVM. Overall, this setup has $2,676$ parameters.

Displacements $\boldsymbol{r}_{pq}^t$ and functions ϕ_p^U, ϕ^P are computed at the preprocessing stage of the structured SVM. We follow Chen and Yuille, [2014] and obtain the displacements $\boldsymbol{r}_{pq}^t$ with the K-means clustering of the displacements of the training set. Functions ϕ_p^U, ϕ^P consists in a Convolutional Neural Network (CNN) and are also learned from the training set. We refer the reader to the work of Chen and Yuille, [2014] and their project page[10] for further details. Note that the last training stage of Chen and Yuille, [2014] is different from ours, i.e., it consists in binary SVM on the carefully sampled sets of positive and negative examples.

To run SSVM, we define the structured error $L(\boldsymbol{y}_i, \boldsymbol{y})$ as a decomposable function w.r.t. the positions of the joints

$$L(\boldsymbol{y}_i, \boldsymbol{y}) = \frac{1}{|\mathcal{V}|}\sum_{p\in\mathcal{V}} \max\left(1, \frac{\|\boldsymbol{l}_{i,p} - \boldsymbol{l}_p\|_2^2}{s_i^2}\right)$$

[10] http://www.stat.ucla.edu/~xianjie.chen/projects/pose_estimation/pose_estimation.html

where $\boldsymbol{l}_{i,p}$ belongs to the ground-truth labeling $\boldsymbol{y}_i$ and $\boldsymbol{l}_p$ belongs to the labeling $\boldsymbol{y}$. The quantity $s_i \in \mathbb{R}$ is the scaling factor and is defined by the distance between the left shoulder and the right hip in the ground-truth labeling $\boldsymbol{y}_i$. Similarly to Osokin and Kohli, [2014], the complex dependence of the loss on the ground-truth labeling does not influence the complexity of the max oracle.

For the defined structured score and loss, the optimization problem of the max oracle can be exactly solved by the max-sum belief propagation algorithm on an acyclic graph with messages computed with the generalized distance transform (GDT) Felzenszwalb and Huttenlocher, 2005. The usage of GDTs allows to significantly reduce the oracle running time, but requires positivity constraints on the connection weights $\boldsymbol{w}_{P,pq,t}$. BCFW with positivity constraints is described in Appendix A.6.

The resulting max oracle is quite slow (2 seconds per image) even when the belief propagation algorithm is optimized with GDTs. Slow running time made it intractable for us to run the experiments on the full original training set consisting of $1,000$ images. We use only the first 100 images and refer to this dataset as LSP-small. However, both the CNN training and clustering of the displacements were done on the original training set.

Bibliography

[Achanta *et al.*, 2012] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, 2012. "SLIC Superpixels Compared to State-of-the-art Superpixel Methods". In: *PAMI* (cit. on p. 169).

[Alayrac, 2016] J.-B. Alayrac, 2016. *Project webpage (code/dataset)*. http://www.di.ens.fr/willow/research/instructionvideos/ (cit. on pp. 56, 58, 73).

[Alayrac *et al.*, 2016a] J.-B. Alayrac, P. Bojanowski, N. Agrawal, I. Laptev, J. Sivic, and S. L. Julien, 2016a. "Unsupervised learning from Narrated Instruction Videos". In: *CVPR* (cit. on pp. 10, 89, 109).

[Alayrac *et al.*, 2017a] J.-B. Alayrac, P. Bojanowski, N. Agrawal, J. Sivic, I. Laptev, and S. Lacoste-Julien, 2017a. "Learning from narrated instruction videos". In: *PAMI* (cit. on p. 10).

[Alayrac *et al.*, 2016b] J.-B. Alayrac, A. Osokin, I. Lukasewitz, P. K. Dokania, and S. Lacoste-Julien, 2016b. "Minding the Gaps for Block Frank-Wolfe Optimization of Structured SVMs". In: *ICML* (cit. on p. 10).

[Alayrac *et al.*, 2017b] J.-B. Alayrac, J. Sivic, I. Laptev, and S. Lacoste-Julien, 2017b. "Joint Discovery of Object States and Manipulation Actions". In: *ICCV* (cit. on p. 10).

[Andriluka *et al.*, 2009] M. Andriluka, S. Roth, and B. Schiele, 2009. "Pictorial structures revisited: People detection and articulated pose estimation". In: *CVPR* (cit. on p. 14).

[Bach and Harchaoui, 2007] F. Bach and Z. Harchaoui, 2007. "DIFFRAC: A discriminative and flexible framework for clustering". In: *NIPS* (cit. on pp. 26, 43–46, 68, 69, 88, 90, 93, 113).

[Beck *et al.*, 2015] A. Beck, E. Pauwels, and S. Sabach, 2015. "The Cyclic Block Conditional Gradient Method for Convex Optimization Problems". In: *SIOPT* (cit. on p. 125).

[Beck and Shtern, 2015] A. Beck and S. Shtern, 2015. "Linearly convergent away-step conditional gradient for non-strongly convex functions". In: *arXiv* (cit. on p. 156).

[Bishop, 2006] C. M. Bishop, 2006. *Pattern recognition and machine learning*. Springer (cit. on p. 38).

[Bojanowski *et al.*, 2013] P. Bojanowski, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic, 2013. "Finding Actors and Actions in Movies". In: *ICCV* (cit. on pp. 23, 24, 43, 47, 55, 90, 103).

[Bojanowski *et al.*, 2014] P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic, 2014. "Weakly Supervised Action Labeling in Videos Under Ordering Constraints". In: *ECCV* (cit. on pp. 26, 27, 55, 56, 65, 67–70, 78, 95–97, 99, 100, 103).

[Bojanowski *et al.*, 2015] P. Bojanowski, R. Lajugie, E. Grave, F. Bach, I. Laptev, J. Ponce, and C. Schmid, 2015. "Weakly-supervised alignment of video with text". In: *ICCV* (cit. on pp. 26, 55, 56, 68–71, 78, 90, 94, 100, 103, 105, 106).

[Boykov and Kolmogorov, 2004] Y. Boykov and V. Kolmogorov, 2004. "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision". In: *PAMI* (cit. on pp. 132, 171).

[Brady *et al.*, 2006] T. F. Brady, T. Konkle, A. Oliva, and G. A. Alvarez, 2006. "Detecting changes in real-world objects: The relationship between visual long-term memory and change blindness". In: *Communicative and Integrative Biology* (cit. on p. 87).

[Branson *et al.*, 2013] S. Branson, O. Beijbom, and S. Belongie, 2013. "Efficient Large-Scale Structured Learning". In: *CVPR* (cit. on p. 114).

[Braun *et al.*, 2017] G. Braun, S. Pokutta, and D. Zink, 2017. "Lazifying Conditional Gradient Algorithms". In: *ICML* (cit. on p. 131).

[C. Dünner and Jaggi, 2017] T. P. C. Dünner and M. Jaggi, 2017. "Efficient use of limited-memory accelerators for linear learning on heterogeneous systems". In: *NIPS* (cit. on p. 125).

[Carreira and Zisserman, 2017] J. Carreira and A. Zisserman, 2017. "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset". In: *CVPR* (cit. on pp. 6, 21, 22).

[Chambers and Jurafsky, 2008] N. Chambers and D. Jurafsky, 2008. "Unsupervised Learning of Narrative Event Chains." In: *ACL* (cit. on pp. 27, 28, 55).

[Chari *et al.*, 2015] V. Chari, S. Lacoste-Julien, I. Laptev, and J. Sivic, 2015. "On Pairwise Costs for Network Flow Multi-Object Tracking". In: *CVPR* (cit. on p. 65).

[Chen and Yuille, 2014] X. Chen and A. Yuille, 2014. "Articulated Pose Estimation by a Graphical Model with Image Dependent Pairwise Relations". In: *NIPS* (cit. on pp. 14, 132, 171, 172).

[Chéron *et al.*, 2018] G. Chéron, J.-B. Alayrac, I. Laptev, and C. Schmid, 2018. "A Flexible Model for Training Action Localization with Varying Levels of Supervision". In: *arXiv* (cit. on p. 11).

[Cimpoi *et al.*, 2015] M. Cimpoi, S. Maji, and A. Vedaldi, 2015. "Deep Filter Banks for Texture Recognition and Segmentation". In: *CVPR* (cit. on p. 74).

[Cour *et al.*, 2009] T. Cour, B. Sapp, C. Jordan, and B. Taskar, 2009. "Learning from ambiguously labeled images". In: *CVPR* (cit. on p. 23).

[Csiba *et al.*, 2015] D. Csiba, Z. Qu, and P. Richtárik, 2015. "Stochastic Dual Coordinate Ascent with Adaptive Probabilities". In: *ICML* (cit. on pp. 123, 125).

[D. Perekrestenko and Jaggi, 2017] V. C. D. Perekrestenko and M. Jaggi, 2017. "Faster coordinate descent via adaptive importance sampling". In: *AISTATS* (cit. on p. 125).

[Dalal and Triggs, 2005] N. Dalal and B. Triggs, 2005. "Histograms of Oriented Gradients for Human Detection". In: *CVPR* (cit. on pp. 13, 16).

[Damen *et al.*, 2018] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray, 2018. "Scaling Egocentric Vision: The EPIC-KITCHENS Dataset". In: (cit. on p. 37).

[Damen *et al.*, 2014] D. Damen, T. Leelasawassuk, O. Haines, A. Calway, and W. Mayol-Cuevas, 2014. "You-Do, I-Learn: Discovering Task Relevant Objects and their Modes of Interaction from Multi-User Egocentric Video". In: *BMVA* (cit. on pp. 18, 89).

[d'Aspremont *et al.*, 2013] A. d'Aspremont, C. Guzmán, and M. Jaggi, 2013. "An optimal affine invariant smooth minimization algorithm". In: *arXiv* (cit. on p. 147).

[Delaitre *et al.*, 2012] V. Delaitre, D. F. Fouhey, I. Laptev, J. Sivic, A. Gupta, and A. A. Efros, 2012. "Scene semantics from long-term observation of people". In: *ECCV* (cit. on p. 18).

[Delaitre *et al.*, 2011] V. Delaitre, J. Sivic, and I. Laptev, 2011. "Learning person-object interactions for action recognition in still images". In: *NIPS* (cit. on pp. 18, 88).

[Deng *et al.*, 2009] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, 2009. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR* (cit. on pp. 92, 103).

[Desai *et al.*, 2010] C. Desai, D. Ramanan, and C. Fowlkes, 2010. "Discriminative models for static human-object interactions". In: *CVPR Workshops* (cit. on p. 19).

[Doersch *et al.*, 2012] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. A. Efros, 2012. "What Makes Paris Look like Paris?" In: *SIGGRAPH* (cit. on p. 90).

[Duan *et al.*, 2012] K. Duan, D. Parikh, D. Crandall, and K. Grauman, 2012. "Discovering localized attributes for fine-grained recognition". In: *CVPR* (cit. on p. 87).

[Duchenne *et al.*, 2009] O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce, 2009. "Automatic annotation of human actions in video". In: *ICCV* (cit. on pp. 23, 55).

[Everingham *et al.*, 2010] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, 2010. "The Pascal Visual Object Classes (VOC) Challenge". In: *IJCV* (cit. on p. 16).

[Everingham *et al.*, 2006] M. Everingham, J. Sivic, and A. Zisserman, 2006. ""Hello! My name is... Buffy" – Automatic Naming of Characters in TV Video". In: *BMVC* (cit. on p. 23).

[Farhadi *et al.*, 2009] A. Farhadi, I. E. Lim, D. Hoiem, and D. Forsyth, 2009. "Describing Objects by their Attributes". In: *CVPR* (cit. on pp. 17, 87, 89).

[Fathi and Rehg, 2013] A. Fathi and J. M. Rehg, 2013. "Modeling Actions through State Changes". In: *CVPR* (cit. on pp. 18, 89).

[Fellbaum, 1998] C. Fellbaum, 1998. "WordNet: An Electronic Lexical Database." In: *Cambridge, MA: MIT Press.* (Cit. on p. 63).

[Felzenszwalb *et al.*, 2010] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, 2010. "Object Detection with Discriminatively Trained Part-Based Models". In: *PAMI* (cit. on pp. 14–16).

[Felzenszwalb and Huttenlocher, 2000] P. F. Felzenszwalb and D. P. Huttenlocher, 2000. "Efficient matching of pictorial structures". In: *CVPR* (cit. on p. 16).

[Felzenszwalb and Huttenlocher, 2005] P. F. Felzenszwalb and D. P. Huttenlocher, 2005. "Pictorial structures for object recognition". In: *IJCV* (cit. on pp. 132, 173).

[Fernando *et al.*, 2015] B. Fernando, E. Gavves, M. J. Oramas, A. Ghodrati, and T. Tuytelaars, 2015. "Modeling Video Evolution For Action Recognition". In: *CVPR* (cit. on p. 89).

[Fischler and Elschlager, 1973] M. A. Fischler and R. A. Elschlager, 1973. "The Representation and Matching of Pictorial Structures". In: *IEEE Transactions on Computers* (cit. on p. 16).

[Förstner and Gülsch, 1987] W. Förstner and E. Gülsch, 1987. "A fast operator for detection and precise location of distinct points, corners and centres of circular features". In: *ISPRS* (cit. on p. 19).

[Fouhey *et al.*, 2017] D. F. Fouhey, W.-C. Kuo, A. A. Efros, and J. Malik, 2017. "From Lifestyle Vlogs to Everyday Interactions". In: *arXiv* (cit. on pp. 37, 138).

[Fouhey *et al.*, 2014] D. F. Fouhey, V. Delaitre, A. Gupta, A. A. Efros, I. Laptev, and J. Sivic, 2014. "People Watching: Human Actions as a Cue for Single View Geometry". In: *IJCV* (cit. on p. 18).

[Franc, 2014] V. Franc, 2014. "FASOLE: Fast Algorithm for Structured Output LEarning". In: *ECML PKDD* (cit. on p. 128).

[Frank and Wolfe, 1956] M. Frank and P. Wolfe, 1956. "An algorithm for quadratic programming". In: *Naval Research Logistics Quarterly* (cit. on pp. 47, 114, 118).

[Frermann *et al.*, 2014] L. Frermann, I. Titov, and M. Pinkal, 2014. "A hierarchical Bayesian model for unsupervised induction of script knowledge". In: *EACL* (cit. on pp. 28, 29, 34, 55).

[Garber and Meshi, 2016] D. Garber and O. Meshi, 2016. "Linear-Memory and Decomposition-Invariant Linearly Convergent Conditional Gradient Algorithm for Structured Polytopes". In: *NIPS* (cit. on p. 128).

[Girshick, 2015] R. Girshick, 2015. "Fast R-CNN". In: *ICCV* (cit. on pp. 16, 91, 103).

[Girshick *et al.*, 2014] R. Girshick, J. Donahue, T. Darrell, and J. Malik, 2014. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *CVPR* (cit. on p. 16).

[Girshick, 2013] R. Girshick, 2013. *Deformable part models*. http://vision.stanford.edu/teaching/cs231b_spring1213/slides/dpm-slides-ross-girshick.pdf (cit. on p. 13).

[Goodfellow *et al.*, 2014] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, 2014. "Generative Adversarial Networks". In: *NIPS* (cit. on p. 42).

[Goyal *et al.*, 2017] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fründ, P. Yianilos, M. Mueller-Freitag, F. Hoppe, C. Thurau, I. Bax, and R. Memisevic, 2017. "The something something video database for learning and evaluating visual common sense". In: *arXiv* (cit. on p. 19).

[Guillaumin *et al.*, 2014] M. Guillaumin, D. Küttel, and V. Ferrari, 2014. "ImageNet Auto-Annotation with Segmentation Propagation". In: *IJCV* (cit. on p. 169).

[Gupta *et al.*, 2009a] A. Gupta, A. Kembhavi, and L. S. Davis, 2009a. "Observing human-object interactions: Using spatial and functional compatibility for recognition". In: *PAMI* (cit. on pp. 18, 88).

[Gupta *et al.*, 2009b] A. Gupta, P. Srinivasan, J. Shi, and L. S. Davis, 2009b. "Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos". In: *CVPR* (cit. on pp. 29, 30).

[Harris and Stephens, 1988] C. Harris and M. Stephens, 1988. "A combined corner and edge detector". In: *BMVC* (cit. on p. 19).

[Hastie *et al.*, 2009] T. Hastie, R. Tibshirani, and J. Friedman, 2009. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer (cit. on pp. 38, 44).

[He *et al.*, 2017] K. He, G. Gkioxari, P. Dollár, and R. Girshick, 2017. "Mask R-CNN". In: *ICCV* (cit. on p. 17).

[He *et al.*, 2016] K. He, X. Zhang, S. Ren, and J. Sun, 2016. "Deep Residual Learning for Image Recognition". In: *CVPR* (cit. on p. 103).

[Higgins and Sharp, 1988] D. G. Higgins and P. M. Sharp, 1988. "Clustal: A package for performing multiple sequence alignment on a microcomputer". In: *Gene* (cit. on pp. 65, 75).

[Hoffman, 1952] A. J. Hoffman, 1952. "On Approximate Solutions of Systems of Linear Inequalities". In: *Journal of Research of the National Bureau of Standards* (cit. on p. 156).

[Holloway, 1974] C. A. Holloway, 1974. “An extension of the Frank and Wolfe method of feasible directions”. In: *Mathematical Programming* (cit. on p. 125).

[Hsu *et al.*, 2014] D. Hsu, S. M. Kakade, and T. Zhang, 2014. “Random Design Analysis of Ridge Regression”. In: *Foundations of Computational Mathematics* (cit. on p. 82).

[Huang *et al.*, 2016] D.-A. Huang, L. Fei-Fei, and J. C. Niebles, 2016. “Connectionist Temporal Modeling for Weakly Supervised Action Labeling”. In: *ECCV* (cit. on pp. 27, 90).

[Huang *et al.*, 2017] D.-A. Huang, J. J. Lim, L. Fei-Fei, and J. C. Niebles, 2017. “Unsupervised Visual-Linguistic Reference Resolution in Instructional Videos”. In: *CVPR* (cit. on pp. 35, 36).

[Huang *et al.*, 2018] D.-A. Huang, V. Ramanathan, D. Mahajan, L. Torresani, M. Paluri, L. Fei-Fei, and J. C. Niebles, 2018. “Finding "It": Weakly-Supervised Reference-Aware Visual Grounding in Instructional Video”. In: *CVPR* (cit. on pp. 35, 36).

[Isola *et al.*, 2015] P. Isola, J. J. Lim, and E. H. Adelson, 2015. “Discovering States and Transformations in Image Collections”. In: *CVPR* (cit. on pp. 17, 87, 89).

[Jaggi, 2013a] M. Jaggi, 2013a. “Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization”. In: *ICML* (cit. on pp. 141, 144).

[Jaggi, 2013b] M. Jaggi, 2013b. “Revisiting Frank-Wolfe: Projection-free sparse convex optimization”. In: *ICML* (cit. on pp. 48, 49).

[Jaggi, 2013c] M. Jaggi, 2013c. “Revisiting Frank-Wolfe: Projection-free sparse convex optimization”. In: *ICML* (cit. on pp. 63, 65, 96).

[Jain *et al.*, 2013] A. Jain, A. Gupta, M. Rodriguez, and L. Davis, 2013. “Representing videos using mid-level discriminative patches”. In: *CVPR* (cit. on p. 90).

[Jegelka *et al.*, 2013] S. Jegelka, F. Bach, and S. Sra, 2013. “Reflection methods for user-friendly submodular optimization”. In: *NIPS* (cit. on p. 115).

[Joachims *et al.*, 2009] T. Joachims, T. Finley, and C. Yu, 2009. “Cutting-plane training of structural SVMs”. In: *Machine Learning* (cit. on pp. 114, 115, 131, 161).

[Johnson and Everingham, 2010] S. Johnson and M. Everingham, 2010. “Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation”. In: *BMVC* (cit. on pp. 132, 135, 167, 171).

[Joo *et al.*, 2015] H. Joo, H. Liu, L. Tan, L. Gui, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh, 2015. "Panoptic Studio: A Massively Multiview System for Social Motion Capture *". In: *ICCV* (cit. on p. 15).

[Joulin *et al.*, 2014a] A. Joulin, K. Tang, and L. Fei-Fei, 2014a. "Efficient Image and Video Co-localization with Frank-Wolfe Algorithm". In: *ECCV* (cit. on pp. 65, 90).

[Joulin *et al.*, 2014b] A. Joulin, K. Tang, and L. Fei-Fei, 2014b. "Efficient image and video co-localization with Frank-Wolfe algorithm". In: *ECCV* (cit. on p. 115).

[Joulin *et al.*, 2010] A. Joulin, F. Bach, and J. Ponce, 2010. "Discriminative Clustering for Image Co-segmentation". In: *CVPR* (cit. on pp. 46, 47).

[Joulin *et al.*, 2012] A. Joulin, F. Bach, and J. Ponce, 2012. "Multi-class cosegmentation". In: *CVPR* (cit. on p. 46).

[Joulin *et al.*, 2014c] A. Joulin, K. Tang, and L. Fei-Fei, 2014c. "Efficient Image and Video Co-localization with Frank-Wolfe Algorithm". In: *ECCV* (cit. on p. 47).

[Kerdreux *et al.*, 2018] T. Kerdreux, F. Pedregosa, and A. d'Aspremont, 2018. "Frank-Wolfe with Subsampling Oracle". In: *ICML* (cit. on p. 131).

[Kjellström *et al.*, 2011] H. Kjellström, J. Romero, and D. Kragić, 2011. "Visual object-action recognition: Inferring object affordances from human demonstration". In: *CVIU* (cit. on p. 88).

[Kolesnikov *et al.*, 2014] A. Kolesnikov, M. Guillaumin, V. Ferrari, and C. H. Lampert, 2014. "Closed-form approximate CRF training for scalable image segmentation". In: *ECCV* (cit. on pp. 132, 135, 167, 169).

[Krähenbühl and Koltun, 2014] P. Krähenbühl and V. Koltun, 2014. "Geodesic Object Proposals". In: *ECCV* (cit. on pp. 16, 103).

[Kuehne *et al.*, 2011] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, 2011. "HMDB: a large video database for human motion recognition". In: *ICCV* (cit. on pp. 21, 22).

[Lacoste-Julien and Jaggi, 2015a] S. Lacoste-Julien and M. Jaggi, 2015a. "On the Global Linear Convergence of Frank-Wolfe Optimization Variants". In: *NIPS* (cit. on pp. 50, 65, 96).

[Lacoste-Julien and Jaggi, 2015b] S. Lacoste-Julien and M. Jaggi, 2015b. "On the Global Linear Convergence of Frank-Wolfe Optimization Variants". In: *NIPS* (cit. on pp. 115, 119, 125, 147, 151, 153–158).

[Lacoste-Julien *et al.*, 2013] S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher, 2013. "Block-Coordinate Frank-Wolfe Optimization for Structural SVMs". In: *ICML* (cit. on pp. 7, 9, 10, 50, 113, 114, 116–119, 124, 132, 141, 150, 158, 160, 165, 167).

[Lacoste-Julien, 2016] S. Lacoste-Julien, 2016. "Convergence Rate of Frank-Wolfe for Non-Convex Objectives". In: *arXiv* (cit. on pp. 50, 65, 96).

[Lafferty *et al.*, 2001] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, 2001. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *ICML* (cit. on p. 157).

[Laptev *et al.*, 2008a] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, 2008a. "Learning realistic human actions from movies". In: *CVPR* (cit. on pp. 23, 55).

[Laptev, 2005] I. Laptev, 2005. "On space-time interest points". In: *IJCV* (cit. on p. 19).

[Laptev *et al.*, 2008b] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, 2008b. "Learning Realistic Human Actions from Movies". In: *CVPR* (cit. on p. 89).

[Leblond *et al.*, 2018] R. Leblond, J.-B. Alayrac, A. Osokin, and S. Lacoste-Julien, 2018. "SEARNN: Training RNNs with Global-Local Losses". In: *International Conference on Learning Representations (ICLR)* (cit. on p. 11).

[Lee *et al.*, 2002] C. Lee, C. Grasso, and M. Sharlow, 2002. "Multiple sequence alignment using partial order graphs". In: *Bioinformatics* (cit. on pp. 63, 65, 73, 75).

[Lempitsky *et al.*, 2011] V. Lempitsky, A. Vedaldi, and A. Zisserman, 2011. "A Pylon Model for Semantic Segmentation". In: *NIPS* (cit. on p. 169).

[Liao, 2014] T. Liao, 2014. "Clustering of time series data, a survey". In: *Pattern recognition* (cit. on p. 77).

[Lin *et al.*, 2014] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, 2014. "Microsoft coco: Common objects in context". In: *ECCV* (cit. on pp. 16, 40).

[Lloyd, 1982] S. P. Lloyd, 1982. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* (cit. on p. 42).

[Loiola *et al.*, 2007] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, 2007. "A Survey for the Quadratic Assignment Problem". In: *EJOR* (cit. on p. 95).

[Lowe, 2004] D. G. Lowe, 2004. "Distinctive Image Features from Scale-Invariant Keypoint". In: *IJCV* (cit. on p. 170).

[Malmaud *et al.*, 2015] J. Malmaud, J. Huang, V. Rathod, N. Johnston, A. Rabinovich, and K. Murphy, 2015. "What's Cookin'? Interpreting Cooking Videos using Text, Speech and Vision". In: *NAACL* (cit. on pp. 31, 32, 36, 54, 55, 78, 79, 89, 109).

[Marcus *et al.*, 1993] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, 1993. "Building a large annotated corpus of English: The Penn Treebank". In: *Computational linguistics* (cit. on p. 168).

[Marneffe *et al.*, 2006] M.-C. de Marneffe, B. MacCartney, and C. D. Manning, 2006. "Generating typed dependency parses from phrase structure parses". In: *LREC* (cit. on pp. 62, 73).

[Marszalek *et al.*, 2009] M. Marszalek, I. Laptev, and C. Schmid, 2009. "Actions in context". In: *CVPR* (cit. on pp. 22, 23).

[Miech *et al.*, 2017] A. Miech, J.-B. Alayrac, P. Bojanowski, I. Laptev, and J. Sivic, 2017. "Learning from Video and Text via Large-Scale Discriminative Clustering". In: *ICCV* (cit. on pp. 11, 113).

[Mikolov *et al.*, 2013] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, 2013. "Distributed Representations of Words and Phrases and their Compositionality". In: *NIPS* (cit. on p. 74).

[Miller, 1995] G. A. Miller, 1995. "WordNet: A Lexical Database for English." In: *Communications of the ACM* (cit. on p. 63).

[Mitchell *et al.*, 1974] B. Mitchell, V. F. Demyanov, and V. Malozemov, 1974. "Finding the Point of a Polyhedron Closest to the Origin". In: *SIAM Journal on Control* (cit. on pp. 115, 125).

[Mittal *et al.*, 2011] A. Mittal, A. Zisserman, and P. Torr, 2011. "Hand detection using multiple proposals". In: *BMVC* (cit. on p. 15).

[Murphy, 2012] K. P. Murphy, 2012. *Machine learning : a probabilistic perspective*. MIT Press (cit. on p. 38).

[Naim *et al.*, 2015] I. Naim, Y. C. Song, Q. Liu, L. Huang, H. Kautz, J. Luo, and D. Gildea, 2015. "Discriminative Unsupervised Alignment of Natural Language Instructions with Corresponding Video Segments". In: *NAACL* (cit. on p. 55).

[Ñanculef *et al.*, 2014] R. Ñanculef, E. Frandi, C. Sartori, and H. Allende, 2014. "A novel Frank-Wolfe algorithm. Analysis and applications to large-scale SVM training". In: *Information Sciences* (cit. on p. 128).

[Needell *et al.*, 2014] D. Needell, R. Ward, and N. Srebro, 2014. "Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm". In: *NIPS* (cit. on pp. 124, 147).

[Nesterov, 2012] Y. Nesterov, 2012. "Efficiency of Coordinate Descent Methods on Huge-Scale Optimization Problems". In: *SIAM Journal on Optimization* (cit. on pp. 123, 124, 147).

[Nesterov, 1998] Y. Nesterov, 1998. *Introductory Lectures on Convex Programming Volume I: Basic course* (cit. on p. 48).

[Nguyen *et al.*, 2016] D. T. Nguyen, W. Li, and P. O. Ogunbona, 2016. "Human detection from images and videos: A survey". In: *Pattern Recognition* (cit. on p. 14).

[Niebles *et al.*, 2010] J. C. Niebles, C.-W. Chen, and L. Fei-Fei, 2010. "Modeling Temporal Structure of Decomposable Motion Segments for Activity Classification". In: *ECCV* (cit. on p. 56).

[Niebles *et al.*, 2008] J. C. Niebles, H. Wang, and L. Fei-Fei, 2008. "Unsupervised learning of human action categories using spatial-temporal words". In: *IJCV* (cit. on p. 55).

[Okazaki, 2007] N. Okazaki, 2007. *CRFsuite: a fast implementation of Conditional Random Fields (CRFs)* (cit. on p. 168).

[Osokin and Kohli, 2014] A. Osokin and P. Kohli, 2014. "Perceptually Inspired Layout-aware Losses for Image Segmentation". In: *ECCV* (cit. on p. 173).

[Papandreou *et al.*, 2017] G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, and K. Murphy, 2017. "Towards Accurate Multi-person Pose Estimation in the Wild". In: *CVPR* (cit. on p. 15).

[Parikh and Grauman, 2011] D. Parikh and K. Grauman, 2011. "Relative Attributes". In: *ICCV* (cit. on p. 89).

[Parkhi *et al.*, 2015] O. M. Parkhi, A. Vedaldi, and A. Zisserman, 2015. "Deep Face Recognition". In: *BMVC* (cit. on p. 13).

[N.d.], *Partial order alignment code for Multiple Sequence Alignment*. `http://sourceforge.net/projects/poamsa/` (cit. on p. 75).

[Patterson *et al.*, 2014] G. Patterson, C. Xu, H. Su, and J. Hays, 2014. "The SUN Attribute Database: Beyond Categories for Deeper Scene Understanding". In: *IJCV* (cit. on p. 89).

[Peyre *et al.*, 2017] J. Peyre, I. Laptev, C. Schmid, and J. Sivic, 2017. "Weakly-supervised learning of visual relations". In: *ICCV* (cit. on p. 19).

[Pirsiavash and Ramanan, 2012] H. Pirsiavash and D. Ramanan, 2012. "Detecting activities of daily living in first-person camera views". In: *CVPR* (cit. on p. 88).

[Platt, 1999] J. C. Platt, 1999. "Fast Training of Support Vector Machines using Sequential Minimal Optimization". In: *Advances in Kernel Methods - Support Vector Learning*. Ed. by B. Schölkopf, C. Burges, and A. Smola. MIT Press (cit. on p. 128).

[Potapov *et al.*, 2014] D. Potapov, M. Douze, Z. Harchaoui, and C. Schmid, 2014. "Category-specific video summarization". In: *ECCV* (cit. on p. 56).

[Prest *et al.*, 2012] A. Prest, C. Schmid, and V. Ferrari, 2012. "Weakly Supervised Learning of Interactions between Humans and Objects". In: *PAMI* (cit. on p. 19).

[R. Le Priol and Lacoste-Julien, 2018] A. P. R. Le Priol and S. Lacoste-Julien, 2018. "Adaptive Stochastic Dual Coordinate Ascent for Conditional Random Fields". In: *UAI* (cit. on p. 125).

[Ramanathan *et al.*, 2014] V. Ramanathan, A. Joulin, P. Liang, and L. Fei-Fei, 2014. "Linking people with "their" names using coreference resolution". In: *ECCV* (cit. on p. 90).

[Raptis and Sigal, 2013] M. Raptis and L. Sigal, 2013. "Poselet Key-framing: A Model for Human Activity Recognition". In: *CVPR* (cit. on p. 56).

[Ratliff *et al.*, 2007] N. Ratliff, J. A. Bagnell, and M. Zinkevich, 2007. "(Online) Subgradient Methods for Structured Prediction". In: *AISTATS* (cit. on p. 114).

[Regneri *et al.*, 2010] M. Regneri, A. Koller, and M. Pinkal, 2010. "Learning Script Knowledge with Web Experiments". In: *ACL* (cit. on pp. 28, 29, 54, 55, 62).

[Regneri *et al.*, 2013] M. Regneri, M. Rohrbach, D. Wetzel, S. Thater, B. Schiele, and M. Pinkal, 2013. "Grounding Action Descriptions in Videos". In: *TACL* (cit. on p. 26).

[Ren *et al.*, 2015] S. Ren, K. He, R. Girshick, and J. Sun, 2015. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *NIPS* (cit. on pp. 16, 103).

[Richard *et al.*, 2017] A. Richard, H. Kuehne, and J. Gall, 2017. "Weakly Supervised Action Learning with RNN based Fine-to-coarse Modeling". In: *CVPR* (cit. on p. 27).

[Rohrbach *et al.*, 2015] A. Rohrbach, M. Rohrbach, N. Tandon, and B. Schiele, 2015. "A dataset for movie description". In: *CVPR* (cit. on p. 25).

[Rohrbach *et al.*, 2012] M. Rohrbach, M. Regneri, M. Andriluka, S. Amin, M. Pinkal, and B. Schiele, 2012. "Script Data for Attribute-Based Recognition of Composite Activities". In: *ECCV* (cit. on p. 30).

[Sang and Buchholz, 2000] E. F. T. K. Sang and S. Buchholz, 2000. *Introduction to the CoNLL-2000 shared task: Chunking* (cit. on pp. 132, 135, 167, 168).

[Sener and Yao, 2018] F. Sener and A. Yao, 2018. "Unsupervised Learning and Segmentation of Complex Activities from Video". In: *CVPR* (cit. on p. 34).

[Sener *et al.*, 2015] O. Sener, A. Zamir, S. Savarese, and A. Saxena, 2015. "Unsupervised Semantic Parsing of Video Collections". In: *ICCV* (cit. on pp. 32, 33, 36, 56, 89, 109).

[Sha and Pereira, 2003] F. Sha and F. Pereira, 2003. "Shallow Parsing with Conditional Random Fields". In: *NAACL* (cit. on p. 168).

[Shah *et al.*, 2015] N. Shah, V. Kolmogorov, and C. H. Lampert, 2015. "A Multi-Plane Block-Coordinate Frank-Wolfe Algorithm for Training Structural SVMs with a Costly max-Oracle". In: *CVPR* (cit. on p. 131).

[Sigurdsson *et al.*, 2016] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta, 2016. "Hollywood in Homes: Crowdsourcing Data Collection for Activity Understanding". In: *ECCV* (cit. on pp. 89, 102).

[Silver *et al.*, 2017] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, 2017. "Mastering the game of Go without human knowledge". In: *Nature* (cit. on p. 41).

[Simon *et al.*, 2017] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, 2017. "Hand Keypoint Detection in Single Images using Multiview Bootstrapping". In: *CVPR* (cit. on p. 15).

[Simonyan and Zisserman, 2015] K. Simonyan and A. Zisserman, 2015. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *ICLR* (cit. on pp. 74, 104).

[Simonyan and Zisserman, 2014] K. Simonyan and A. Zisserman, 2014. “Two-stream convolutional networks for action recognition in videos”. In: *NIPS* (cit. on pp. 20, 89).

[Singh *et al.*, 2012] S. Singh, A. Gupta, and A. A. Efros, 2012. “Unsupervised Discovery of Mid-level Discriminative Patches”. In: *ECCV* (cit. on p. 90).

[Sivic *et al.*, 2009] J. Sivic, M. Everingham, and A. Zisserman, 2009. “"Who are you?" - Learning person specific classifiers from video”. In: *CVPR* (cit. on p. 23).

[Soomro *et al.*, 2012] K. Soomro, A. R. Zamir, and M. Shah, 2012. “UCF101: A dataset of 101 human actions classes from videos in the wild”. In: *arXiv* (cit. on pp. 21, 22).

[Sun *et al.*, 2014] M. Sun, A. Farhadi, and S. Seitz, 2014. “Ranking domain-specific highlights by analyzing edited videos”. In: *ECCV* (cit. on p. 56).

[Sutton and Barto, 1998] R. S. Sutton and A. G. Barto, 1998. *Introduction to Reinforcement Learning*. 1st. Cambridge, MA, USA: MIT Press (cit. on p. 41).

[Tapaswi *et al.*, 2016] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler, 2016. “MovieQA: Understanding Stories in Movies through Question-Answering”. In: *CVPR* (cit. on p. 25).

[Taskar, 2004] B. Taskar, 2004. “Learning structured prediction models: A large margin approach”. Doctoral dissertation. Stanford University (cit. on p. 128).

[Taskar *et al.*, 2003] B. Taskar, C. Guestrin, and D. Koller, 2003. “Max-Margin Markov Networks”. In: *NIPS* (cit. on pp. 114, 116, 124, 130, 132, 135, 167).

[Taylor *et al.*, 2016] J. Taylor, L. Bordeaux, T. Cashman, B. Corish, C. Keskin, E. Soto, D. Sweeney, J. Valentin, B. Luff, A. Topalian, E. Wood, S. Khamis, P. Kohli, T. Sharp, S. Izadi, R. Banks, A. Fitzgibbon, and J. Shotton, 2016. “Efficient and Precise Interactive Hand Tracking through Joint, Continuous Optimization of Pose and Correspondences”. In: *SIGGRAPH* (cit. on p. 15).

[Toshev and Szegedy, 2014] A. Toshev and G. C. Szegedy, 2014. “DeepPose: Human Pose Estimation via Deep Neural Networks”. In: *CVPR* (cit. on p. 14).

[Tran *et al.*, 2015] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, 2015. “Learning spatiotemporal features with 3D convolutional networks”. In: *ICCV* (cit. on pp. 21, 89).

[Tsochantaridis *et al.*, 2005] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, 2005. "Large margin methods for structured and interdependent output variables". In: *JMLR* (cit. on pp. 114, 116).

[Venugopalan *et al.*, 2015a] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, 2015a. "Sequence to sequence-video to text". In: *ICCV* (cit. on p. 25).

[Venugopalan *et al.*, 2015b] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko, 2015b. "Translating Videos to Natural Language Using Deep Recurrent Neural Networks". In: *NAACL* (cit. on p. 25).

[Vinyals *et al.*, 2014] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, 2014. "Show and tell: A neural image caption generator". In: *arXiv* (cit. on p. 24).

[Viola and Jones, 2001] P. Viola and M. Jones, 2001. "Rapid Object Detection using a Boosted Cascade of Simple Features". In: *CVPR* (cit. on p. 13).

[Viterbi, 1967] A. J. Viterbi, 1967. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". In: *IEEE Transactions on Information Theory* 13.2, pp. 260–269 (cit. on pp. 132, 168, 169).

[Wainwright and Jordan, 2008] M. J. Wainwright and M. I. Jordan, 2008. "Graphical models, exponential families, and variational inference". In: *Foundations and Trends in Machine Learning* (cit. on p. 157).

[Wang and Schmid, 2013] H. Wang and C. Schmid, 2013. "Action Recognition with Improved Trajectories". In: *ICCV* (cit. on pp. 20, 21, 74, 89, 104, 107).

[Wang *et al.*, 2011] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, 2011. "Action recognition by dense trajectories". In: *CVPR* (cit. on p. 20).

[Wang and Jiang, 1994] L. Wang and T. Jiang, 1994. "On the complexity of multiple sequence alignment". In: *Journal of computational biology* (cit. on pp. 62, 63, 75).

[Wang *et al.*, 2016] X. Wang, A. Farhadi, and A. Gupta, 2016. "Actions Transformations". In: *CVPR* (cit. on pp. 17, 89).

[Wang *et al.*, 2014] Y.-X. Wang, V. Sadhanala, W. Dai, W. Neiswanger, S. Sra, and E. P. Xing, 2014. "Parallel and Distributed Block-Coordinate Frank-Wolfe Algorithms". In: *arXiv* (cit. on p. 125).

[Wei *et al.*, 2016] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, 2016. "Convolutional Pose Machines". In: *CVPR* (cit. on p. 14).

[Wolfe, 1970] P. Wolfe, 1970. "Convergence Theory in Nonlinear Programming". In: *Integer and Nonlinear Programming*. Ed. by J. Abadie. North-Holland (cit. on pp. 50, 115, 125).

[Xu *et al.*, 2004] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans, 2004. "Maximum Margin Clustering". In: *NIPS* (cit. on pp. 44, 90).

[Yang and Toderici, 2011] W. Yang and G. Toderici, 2011. "Discriminative tag learning on YouTube videos with latent sub-tags". In: *CVPR* (cit. on p. 23).

[Yang and Ramanan, 2011] Y. Yang and D. Ramanan, 2011. "Articulated pose estimation with flexible mixtures-of-parts". In: *CVPR* (cit. on p. 14).

[Yao and Fei-Fei, 2010] B. Yao and L. Fei-Fei, 2010. "Grouplet: A structured image representation for recognizing human and object interactions". In: *CVPR* (cit. on p. 19).

[Yao *et al.*, 2011] B. Yao, X. Jiang, A. Khosla, A. L. Lin, L. Guibas, and L. Fei-Fei, 2011. "Human action recognition by learning bases of action attributes and parts". In: *ICCV* (cit. on p. 88).

[Yao *et al.*, 2015] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville, 2015. "Describing videos by exploiting temporal structure". In: *ICCV* (cit. on p. 25).

[Zhang and Zhang, 2010] C. Zhang and Z. Zhang, 2010. *A Survey of Recent Advances in Face Detection*. Tech. rep. (cit. on p. 13).

[Zhao and Zhang, 2015] P. Zhao and T. Zhang, 2015. "Stochastic Optimization with Importance Sampling for Regularized Loss Minimization". In: *ICML* (cit. on pp. 124, 147).

[Zhou *et al.*, 2018] L. Zhou, X. Chenliang, and J. J. Corso, 2018. "Towards Automatic Learning of Procedures from Web Instructional Videos". In: *AAAI* (cit. on pp. 33, 37).

Résumé

Le but de cette thèse est de développer des modèles, des représentations adaptées et des algorithmes de prédiction structurée afin de pouvoir analyser de manière automatique des activités humaines complexes commentées par du langage naturel.

Dans un premier temps, nous présentons un modèle capable de découvrir quelle est la liste d'actions nécessaires à l'accomplissement de la tâche ainsi que de localiser ces actions dans le flux vidéo et dans la narration textuelle à partir de plusieurs vidéos tutorielles. La première hypothèse est que les gens réalisent les actions au moment où ils les décrivent. La seconde hypothèse est que ces tâches complexes sont réalisées en suivant un ordre précis d'actions.. Notre modèle est évalué sur un nouveau jeu de données de vidéos tutorielles qui décrit 5 tâches complexes.

Nous proposons ensuite de relier les actions avec les objets manipulés. Plus précisement, on se concentre sur un type d'action particulière qui vise à modifier l'état d'un objet. Par exemple, cela arrive lorsqu'on *sert une tasse de café* ou bien lorsqu'on *ouvre une porte*. Ce type d'action est particulièrement important dans le contexte des vidéos tutorielles. Notre méthode consiste à minimiser un objectif commun entre les actions et les objets. Nous démontrons via des expériences numériques que localiser les actions aident à mieux reconnaitre l'état des objets et inversement que modéliser le changement d'état des objets permet de mieux déterminer le moment où les actions se déroulent.

Tous nos modèles sont basés sur du partionnement discriminatif, une méthode qui permet d'exploiter la faible supervision contenue dans ce type de vidéos. Cela se résume à formuler un problème d'optimisation qui peut se résoudre aisément en utilisant l'algorithme de Frank-Wolfe qui est particulièrement adapté aux contraintes envisagées. Motivé par le fait qu'il est très important d'être en mesure d'exploiter les quelques milliers de vidéos qui sont disponibles en ligne, nous portons enfin notre effort à rendre l'algorithme de Frank-Wolfe plus rapide et plus efficace lorsque confronté à beaucoup de données. En particulier, nous proposons trois modifications à l'algorithme Block-Coordinate Frank-Wolfe : un échantillonage adaptatif des exemples d'entrainement, une version bloc des 'away steps' et des 'pairwise steps' initialement prévu dans l'algorithme original et enfin une manière de mettre en cache les appels à l'oracle linéaire.

Mots Clés

Vision par ordinateur, Reconnaissance d'actions, Texte et vidéo, Apprentissage faiblement supervisée

Abstract

The goal of this thesis is to develop models, representations and structured learning algorithms for the automatic understanding of complex human activities from instructional videos narrated with natural language.

We first introduce a model that, given a set of narrated instructional videos describing a task, is able to generate a list of action steps needed to complete the task and locate them in the visual and textual streams. To that end, we formulate two assumptions. First, people perform actions when they mention them. Second, we assume that complex tasks are composed of an ordered sequence of action steps. Equipped with these two hypotheses, our model first clusters the textual inputs and then uses this output to refine the location of the action steps in the video. We evaluate our model on a newly collected dataset of instructional videos depicting 5 different complex goal oriented tasks.

We then present an approach to link action and the manipulated objects. More precisely, we focus on actions that aim at modifying the *state* of a specific object, such as *pouring a coffee cup* or *opening a door*. Such actions are an inherent part of instructional videos. Our method is based on the optimization of a joint cost between actions and object states under constraints. The constraints are reflecting our assumption that there is a consistent temporal order for the changes in object states and manipulation actions. We demonstrate experimentally that object states help localizing actions and conversely that action localization improves object state recognition.

All our models are based on discriminative clustering, a technique which allows to leverage the readily available *weak supervision* contained in instructional videos. In order to deal with the resulting optimization problems, we take advantage of a highly adapted optimization technique: the Frank-Wolfe algorithm. Motivated by the fact that scaling our approaches to thousands of videos is essential in the context of narrated instructional videos, we also present several improvements to make the Frank-Wolfe algorithm faster and more computationally efficient. In particular, we propose three main modifications to the Block-Coordinate Frank-Wolfe algorithm: gap-based sampling, away and pairwise Block Frank-Wolfe steps and a solution to cache the oracle calls. We show the effectiveness of our improvements on four challenging structured prediction tasks.

Keywords

Computer vision, Action recognition, Video and text, Weakly-supervised learning, Instructional Videos

Numéro national de thèse :
xxxxx