Discovering and Leveraging Visual Structure for Large-scale Recognition

Abhinav Shrivastava

CMU-RI-TR-17-63

Submitted in partial fulfillment
of the requirements of the degree of
Doctor of Philosophy in Robotics

The Robotics Institute
School of Computer Science
Carnegie Mellon University

August 2017

Thesis Committee:

Abhinav Gupta, Chair
Martial Hebert
Deva Ramanan
Alexei A. Efros, University of California, Berkeley
Jitendra Malik, University of California, Berkeley

Abstract

Our visual world is extraordinarily varied and complex, but despite its richness, the space of visual data may not be that astronomically large. We live in a well-structured, predictable world, where cars almost always drive on roads, sky is always above the ground, and so on. As humans, the ability to learn this structure from prior experiences is essential to our visual perception. In fact, we effortlessly (and often unconsciously) employ this structure for perceiving and responding to our surroundings; a feat that still eludes our computational systems. In this dissertation, we propose to discover and harness this structure to improve large-scale visual recognition systems.

In Part I, we present supervised recognition algorithms that can leverage these underlying regularities in our visual world. We propose effective models for object recognition that incorporate top-down contextual feedback and models that can leverage geometric-structure of objects. We also develop supervised learning and inference methods that exploit the structure offered by visual data and by a wide range of recognition tasks.

These supervised systems, limited by our ability to collect annotations, are confined to curated datasets. Therefore, in Part II, we propose to overcome this limitation by automatically discovering structure in large amounts of visual data and incorporating it as constraints in large-scale semi-supervised learning algorithms to improve visual recognition systems.

Acknowledgments

There are many people, including advisors, collaborators, friends, and family, who have supported and guided me throughout, and to whom I owe deep gratitude.

First and foremost, I thank my advisor, Abhinav Gupta, for always providing me with the right balance of guidance and freedom, at the right time. I appreciate his consistent support, the time and energy he put in, and above all his patience, during both successes and failures. Though it is impossible to justify his impact on my growth in a few lines without being reductive; put simply, he has been a close friend and a fantastic advisor over the last 7 years, and for that, I am immensely and sincerely grateful.

I'd also like to thank my Masters' advisors, Alyosha Efros and Martial Hebert, for taking a chance on a young and inexperienced student. They have, generously and selflessly, continued to serve as shadow advisors during my Ph.D. I cannot summarize all that they have done for me; but I especially thank Alyosha for giving me my first break in the field, teaching me never to give up, always pushing me and insisting on excellence, spending long hours late at night helping me with talks, coining my alias 'A2', and introducing me to fine spirits; and Martial for teaching me the importance of putting ideas in broader research perspective, exposing me to hands-on real-world applications, trusting me to give demos and presentations to senior leadership, and making sure that my scotch is without ice!

I cannot overstate the continued impact of Abhinav, Alyosha, and Martial in shaping my research career and outlook since the day I stepped into Carnegie Mellon. I cannot imagine my academic fate had I not been part of their labs, for which I consider myself undeservingly fortunate.

Thank you to my thesis committee members, Deva Ramanan and Jitendra Malik, for enlightening discussions, valuable feedback, kind words of encouragement, and flexibility throughout the entire process. I thank David Forsyth for his support, critical insights into my work, and thoughtful suggestions. I am deeply grateful to Rahul Sukthankar for his support and guidance all these years which have had a remarkable influence on my career.

It was a pleasure learning from a diverse group of faculty in Smith Hall. I'd especially like to thank Kayvon Fatahalian, Takeo Kanade, Kris Kitani, Srinivas Narasimhan, Yaser Sheikh, and Fernando De la Torre for their time, guidance and encouragement. Thanks to Microsoft Research and Google Research for excellent internship opportunities; especially Rahul Sukthankar, Mark Segal, Ross Girshick, and Larry Zitnick, for their incredible mentorship.

I owe much to the awesome atmosphere of Smith Hall and the vision and graphics group members, for whom I have great respect and admiration. Thanks to Aayush Bansal, Nadine Chang, Xinlei Chen, Alvaro Collet, Shreyansh Daftry, Carl Doersch, Santosh Divvala, Ali Farhadi, David Fouhey, Dhiraj Gandhi, Rohit Girdhar, Ed Hsiao, Eakta Jain, Hongwen Kang, Abhijeet Khanna, Natasha Kholgade, Jean-François Lalonde, Yong-Jae Lee, Aravindh Mahendran, Tomasz Malisiewicz, Kenneth Marino, Narek Melik-Barkhudarov, Ishan Misra, Lekha Mohan, Yair Movshovitz-Attias, Dan Munoz, Adithya Murali, Ben Newman, Devi Parikh, Lerrel Pinto, Sentil Purushwalkam, Varun Ramakrishna, Olga Russakovsky, Scott Satkin, Gunnar Sigurdsson, Krishna Kumar Singh, Saurabh Singh, Anish Sinha, Ekaterina Taralova, Yuandong Tian, Jack Valmadre, Jacob Walker, Jiuguang Wang, Xiaolong Wang, Yuxiong Wang, Andreas Wendel, Tinghui Zhou, and Jun-Yan Zhu, for discussions, feedback, and friendship. Sincere thanks to my co-authors Ishan, Saurabh, Aayush, Chen, Xiaolong, Tomasz, Elissa, Xinlei, and Carl for their efforts and all those fun all-nighters. I've learned a lot from each of you.

Special thanks to: Jean-François, Tomasz, and Alvaro for their instruction during my early days; Derek, Jean-François, and Yuandong for their advice during fellowship and job applications; Chen, Ishan, Saurabh, and Sean for a great internship experience; Abhinav, David, Saurabh, and Ishan for making conferences memorable; David and Ishan for helping with talks and proof-reading; Abhinav, Bhavna, Dan, David, Dev, Dey, Ermine, Govind, Hatem, Ishan, Jack, Natasha, Nandita, Ravi, Saloni, Saurabh, Shannon, Shaurya, Swati, Varun, Varuni, and Zeel for some unforgettable parties. Shout-out to the coffee gang (Abhinav, David, and Ishan) and movie knights (Jiuguang, Dey, Natasha, Saurabh) for a great time!

The Robotics Institute and CMU provides an incredibly nurturing environment for a graduate student. Particular thanks to Rachel Burchin for being a friend and advisor and helping navigate the immigration maze, Suzanne Lyons Muth for keeping me on track, Lynnetta Miller, Christine Downey, and Jessica Butterbaugh for all their help and patience, Byron Spice for helping with media outreach, SCS Computing Facilities (especially Ed Walter and Bill Love) for maintaining the servers and keeping up with my untimely requests and unannounced meetings.

I am also indebted to my late uncle, Abhay Shrivastava, without whose guidance I would not have pursued a research career, and to my undergraduate mentor, Sanjay Goel, for motivating me to pursue my passion. Thanks to all my teachers for their kind words, encouragement, and support.

I've been incredibly lucky to be surrounded by many supportive friends, who have kept me sane during some insane times. For being my family away from home, I'd like to thank: Ashima, Shannon, & Dey; Samhita, Harshita, & Saurabh; Saloni & Ishan; Skip, Swati, & Abhinav; Zeel & Ravi; Manali, Anubha, Akshat, Anurag, & Varun Saxena. Thanks to: Saurabh and Harshita for hosting and feeding me during my Bay Area visits; David for hosting me at Berkeley; Virag Mama for Laphroaig 30, and much more, which made Bay Area visits particularly fun; friends and colleagues in Bay Area (especially Varun Somani, Ridhima, Rashmi, Ashwath, Rishabh, Amrita, Sean, and Himanshu) for always finding time for me; dear friends elsewhere (Anubhav, Neha Kumar, Aparna, Kripi, Robin, Shalini, Adit, Ashmita, Saurabh Aswani, Sonal, and everyone else), for always being a phone call away, no matter for how long we haven't talked. Thanks to everyone who has ever served me caffeine, wine, and scotch, for this dissertation was written in between sips. Special thanks to Ishan, Dey, Saurabh, and Anubhav for always being there.

Finally, I'd like to thank my family and extended family, without whose support I could not have finished this thesis. I thank my parents and sister for their never-ending love and support throughout all these years. Thank you, Mom and Dad, for your sacrifices and encouragement which enabled me to follow my dreams. And last, but certainly not the least, I thank my wife, Varuni, for selfless sacrifice, quiet patience, unwavering support, and unconditional love. Thank you, Varuni, for being my partner in crime, bearing with my erratic schedules, sticking with me through thick and thin, and constantly motivating me to achieve more.

Thank you all for believing in me, even when I did not.

Credits. This work has been partially supported by a Microsoft Research PhD Fellowship, ONR grants: N000141010766, MURI N000141612007, MURI N000141010934, and MURI N000141612007, NSF grants IIS-1320083 and IIS-1065336, ARL grant CTA W911NF-10-2-0016, and gifts from Google and Bosch. We'd also like to thank Yahoo! and NVIDIA for hardware donations. Image credits (Chapter 7): Charalampos Laskaris, Carol Williams, Claudio Conforti, Eddie Wong, Edson Campos, Prof. Hall Groat II, Kathleen Brodeur, Moira Munro, Matt Wyatt, Keith Hornblower, Don Amadio (Scrambled Eggz Productions), The Stephen Wilthsire Gallery, www.daydaypaint.com, The Art Renewal Center and Bundesarchiv. We thank the Flickr users who placed their work under Creative Commons License, researchers who collected and released datasets and made their code available online.

Traveler, there is no path.
The path is made by walking.
– Antonio Machado

For my friends and family,
who offered their unconditional love and support
whatever path I took.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Problems worthy
of attack
prove their worth
by fighting back.

Piet Hein

We humans have the remarkable ability to perceive and operate in the visual world around us. For example, recognizing our favorite cookie on a heavily cluttered dinner table is hardly a challenge for us. However, the same task is herculean for a robot, despite tremendous advancement in artificial intelligence over the years. This dissertation is about bringing computational systems closer to operating in such real-world scenarios. Our primary focus is developing computation models and algorithms for **Visual Recognition**, the problem of identifying and reasoning about concepts within an image, which is one of the key challenges in computer vision and artificial intelligence.

Our visual world is extraordinarily varied and complex, but despite its richness, the space of visual data may not be that astronomically large. We live in a well-structured, predictable visual world, where cars almost always drive on roads, plates are kept on dining-tables, sky is always above the ground, televisions and paintings usually have rectangular fronts, beds and tables have horizontal surfaces, and so on. As humans, the ability to learn these regularities, or structure, from prior experiences is essential to our visual perception [294]. This structure not only acts as top-down contextual feedback in our visual system, but also facilitates reasoning when visual information is insufficient. In fact, we effortlessly (and often unconsciously [294]) employ this structure for perceiving and responding to our surroundings; a feat that still eludes our artificial systems. *I firmly believe the better a system is at discovering, learning, and exploiting this inherent structure, the better*

it will be at understanding and reacting to the visual world.

Driven by this hypothesis, this dissertation proposes to **discover and harness this structure in the visual data to enable large-scale visual recognition**.

The last decade has seen tremendous advances in visual recognition algorithms. This progress has been primarily fueled by massive efforts by researchers in the field to collect human-annotations (in terms of labeled instances of scenes, objects, actions, attributes *etc.*) for large amounts of visual data available online (images, videos, *etc.*). This supervised learning paradigm is the backbone of today's visual recognition algorithms. Therefore, the first part of this dissertation is on:

I. **Supervised Visual Recognition**, where we propose computational models and learning algorithms that enable these supervised recognition systems to leverage the underlying regularities in our visual world. This includes:

   (a) **Models for object recognition**; such as models that can incorporate the visual structure using top-down contextual feedback and achieve state-of-the-art results on challenging benchmarks (Chapters 2 and 3), and models that leverage the 3D structure and provide a geometric understanding of objects from 2D images (Chapter 5).

   (b) **Optimization and inference methodologies** that utilize the structure in visual data; such as algorithms that can effectively leverage task-specific problem structure (Chapter 4), underlying 3D structure of objects (Chapter 5), shared structure between multiple tasks (Chapter 6).

The two ingredients required by these supervised learning algorithms, visual data and human-annotations, vary greatly in their availability and cost. While visual data is abundant and cheap (due to increasingly inexpensive sensors, computation, and storage), human-provided labels, in comparison, are scant and expensive. For example, annotating one of the largest vision dataset (ImageNet), which contains $\sim$1 million images with bounding-box labels, took over 5 years (using 19 man-years). This might seem impressive if it were not for the fact that $\sim$350 million new images are uploaded to Facebook daily and 300 hours of video are uploaded to YouTube every minute. In all likelihood, manual labeling cannot possibly scale to the ever increasing amounts of visual data. Therefore, limited by our ability to collect annotations, current systems are confined to curated datasets.

An obvious solution is to circumvent this labeling bottleneck and do the following: a) train supervised models using the already labeled data, b) use these models to find similar concepts in unlabeled data and label it, and c) continue doing this until everything is labeled. This is the classic semi-supervised learning paradigm, which

has been quite promising in several domains. However, semi-supervised approaches are often not reliable when applied to visual data. The primary reason is that our supervised visual models are not perfect. Hence, the notion of similarity as captured by these visual models (*i.e.*, deciding if two images depict visually similar information or concept), which is a critical requirement for any semi-supervised method, is not very reliable for visual data. This often leads to newly labeled examples straying away from the original meaning of the concept (semantic drift).

So, the key question is: how do we reduce the reliance of our recognition systems on carefully annotated datasets and enable them to harness the sea of unlabeled visual data? To answer this, the second part of this dissertation focuses on:

II. **Recognition beyond Extensive Supervision**. We propose systems that leverage the underlying structure in our visual data to overcome the limitations highlighted above and capitalize on large-scale unlabeled visual data. This includes learning frameworks (Chapters 7 and 8) that incorporate this structure as constraints, and systems that discover and learn this structure continuously from millions of images and videos (Chapter 9).

1.1 Overview

> Most of the complex structures found in the world are enormously redundant, and we can use this redundancy to simplify their description. But to use it, to achieve the simplification, we must find the right representation.
>
> Herbert A. Simon

The organization of this thesis follows the two-pronged strategy for discovering and leveraging the regularities in our visual data. In Part I, we present supervised recognition models and learning algorithms, with various flavors of labeled data and target tasks. In Part II, we show how to reduce the reliance on extensive human-provided annotations and leverage large amounts of unlabeled data to improve visual recognition systems. We describe the problem setup for each Chapter in both Parts and their key insights below.

Identifying and localizing objects in a scene, or Object Recognition, is at the heart of visual perception; and representations from *bottom-up*, *feedforward* Convolutional Networks (ConvNets) are the backbone of recent object recognition systems. However, studies in human perception suggest the importance of top-down information, context, and feedback for object recognition [154]. Drawing inspiration from this, in Chapters 2 and 3, we propose novel ConvNet representations with *top-down*

Figure 1.1 – Qualitative results from Chapter 2 (top row) and Chapter 3 (bottom row).

feedback that enable recognition systems to leverage contextual structure in visual data. These models, originally introduced in [250, 254], are one of the first to report significant quantitative improvements on various recognition tasks by incorporating top-down evidence.

In Chapter 2, we show how semantic segmentation outputs (per-pixel likelihood of objects) can be used as a proxy for top-down information. We argue that a segmentation output captures contextual relationships between objects (such as relative likelihood, location, and size) and use it for contextual priming and providing feedback. Our results indicate that such top-down priming improves the performance on object detection, semantic segmentation, and region proposal generation. Qualitative results on semantic segmentation are shown in Figure 1.1 (top row). Current recognition systems, including Chapter 2, rely on the high-level semantic representations from ConvNets, which, by design, are invariant to low-level details. These fine details are often essential to recognize many objects; *e.g.*, finding the 'remote' in cluttered 'livingroom' in Figure 1.1 (bottom row). To address this, we introduce top-down modulation network in Chapter 3, which utilizes top-down contextual structure to modulate and select low-level finer details, and integrates them with the high-level semantic representation for object recognition. The proposed network provides substantial gains in recognition rates for various ConvNet architectures, yielding state-of-the-art results on the challenging COCO object detection dataset (Qualitative detection results are shown in Figure 1.1 (bottom row)).

The object detection frameworks discussed in Chapters 2 and 3 are trained using techniques developed for object classification, where the goal is to identify the presence of objects in an image, and not localize them. To adapt these techniques for localization, several heuristics and hyperparameters are used which are suboptimal and costly to tune. These heuristics are primarily used to address the issue that detection datasets contain an overwhelming number of easy examples and a small number of hard examples. Instead of using costly heuristics, in Chapter 4, we

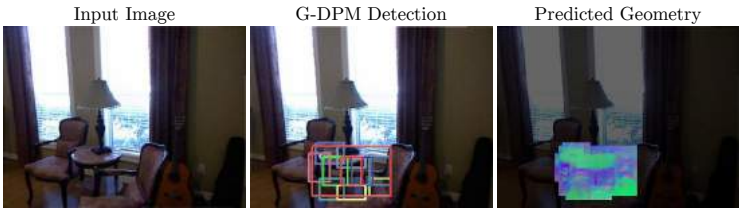


Figure 1.2 – Qualitative result from Chapter 5.

present a simple and intuitive online hard example mining algorithm that leverages the detection-specific problem structure when training ConvNet-based models in a principled way and enables automatic selection of these hard examples. This not only simplifies the training procedure, but also makes it more effective and efficient. More importantly, it yields consistent and significant boosts in detection performance on benchmarks like PASCAL VOC 2007, 2012 and MS COCO. This work, originally introduced in [253], is being used by the community for training state-of-the-art systems for object detection, semantic and instance segmentation.

Next, we look at how we can use the underlying geometry of objects to impose structure while training detection models in Chapter 5. The standard output of object recognition systems discussed so far is either a box around the localized object (detection) or per-pixel labels (segmentation). Though critical building blocks, these outputs offer a rather shallow understanding of the recognized object. In Chapter 5, we propose a system to infer 3D properties of objects from 2D images. Our model first automatically discovers the geometric structure of an object and its parts using multi-model input (images + depth). During training, we enforce that the model follows this geometric structure. The model is trained only for 2D images, and depth is only used to provide geometric constraints. Therefore, during inference, the model only needs images (and no depth). These geometry-constrained deformable part-based models (G-DPM), originally introduced in [249], provide state-of-the-art performance on the NYUv2 dataset (Figure 1.2).

In Chapter 5, we proposed a model to better utilize images and depth data during training. Next, we look at a more generic setup of multi-task learning, where we jointly utilize multiple supervisory labels for training recognition models (*e.g.*, labels for scenes, objects, attributes, depth, *etc.*). ConvNets trained using multi-task learning have been widely successful in the field of recognition, primarily because having multiple tasks forces the model to learn shared representation suitable for both tasks. However, existing multi-task approaches rely on enumerating multiple ConvNet architectures, which are specific to the tasks at hand and do not generalize. In Chapter 6, we propose a principled approach to learn such shared representation, which automatically discovers optimal combination of shared and

task-specific representations via "cross-stitch" units. Our method generalizes across multiple tasks and shows dramatically improved performance for categories with few training examples. This work, originally presented in [193], highlights the importance of leveraging structure between tasks for learning shared representations.

So far, in Part I, we have seen how to improve supervised recognition models by leveraging the structure in visual data. In Part II, we present methods for discovering this structure automatically from large-amounts of visual data, and leveraging it to improve recognition algorithms. Towards this, we develop systems that utilize data with varying granularity of labeling, such as weak and noisy labels (Chapter 9), sparse and partial labels (Chapters 8 and 9), or no labels at all (Chapter 7).

In Chapter 7, we start with the simplest setting, where we are given only a single labeled instance of a concept, without any explicit negatives (or images without any concepts labeled), and the goal is to learn a good visual similarity metric. A critical component of finding recurring patterns in 'big visual data' is matching images, or parts thereof, with each other. However, this is surprisingly challenging because the notion of similarity, required for matching, is ill-defined. We present a simple visual similarity metric based on notion of "data-driven uniqueness," which estimates the relative importance of different features of an image based on what best distinguishes it from the statistical structure of millions of images. This visual similarity shows good performance on a number difficult cross-domain visual tasks, *e.g.*, matching paintings or sketches to real photographs.

In Section 7.4, we briefly discuss applications of this improved visual similarity, such as Internet Re-photography and Painting2GPS in Section A1, organization and exploration of unordered large-scale visual data in Section A2, and finally discovering object instances and their segmentation masks from weakly-supervised and noisy web data in Section A3. These works were originally presented in [42, 186, 251].

Finally, in Chapters 8 and 9, we tackle the problem of effectively utilizing large amounts of unlabeled data along with a small amount of labeled data, i.e., the semi-supervised learning (SSL) paradigm. We present recognition algorithms that discover the visual structure from labeled data and use it as constraints in an SSL framework, to capitalize on large amounts of unlabeled visual data. These works were originally presented in [41, 192, 252].

In Chapter 8, we begin with a case study of a proof-of-concept system, where we study how to incorporate different types of constraints, and their importance, in an SSL framework. We start with a small list of scene categories and manually provide annotated instances and associated constraints. The constraints are provided in the form of attribute and comparative attribute labels for the scenes. To incor-

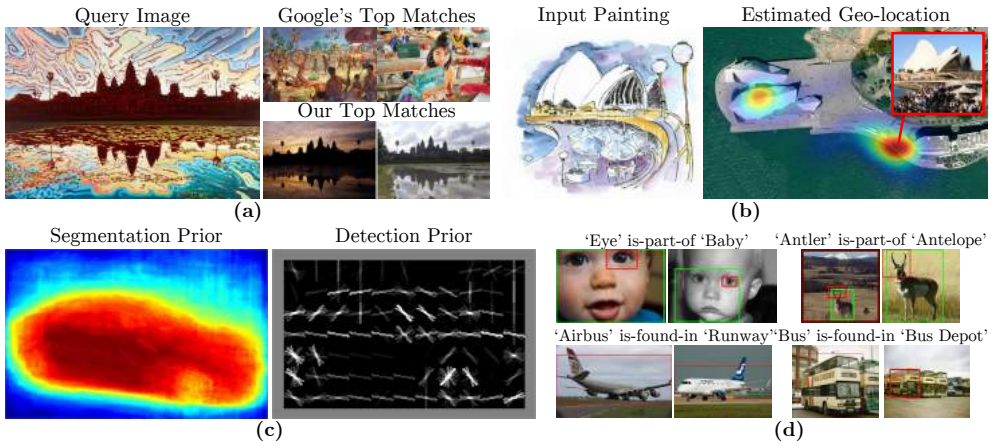


Figure 1.3 – Qualitative results from Part II. (a) Image retrieval results using similarity metric (Chapter 7); (b) `Painting2GPS` (Chapter 7 Section A1); (c) Learned object recognition priors from weakly-supervised and noisy web data (Chapter 7 Section A3); (d) Learned visual relationships (Chapter 9).

porate these constraints, we propose a mathematical framework 'constrained-SSL,' which can train good recognition models even when starting from just two labeled instances; while the standard SSL approaches suffer severe semantic drift. This work, presented in [252], was one of the first to reliably utilize SSL for large-scale recognition (with an unlabeled dataset of millions of images).

In Chapter 9, we propose to apply the ideas from this case study to real world scenarios. In Section 9.1, we propose a system that can learn these constraints from weakly-supervised, noisy web-data. We extend the list of concepts to scenes, objects and attributes, and type of constraints to scene-object, object-object, scene-attribute and object-attribute relationships. We show that these automatically discovered relationships are good for constrained SSL. We only provided relevant details in Section 9.1, other details can be found in [41]. In Section 9.2, we demonstrate how constraints can be discovered and harnessed in large-scale videos, where only a handful of frames are sparsely labeled with concepts. The proposed technique handles detection of multiple objects without assuming exhaustive labeling of object instances on any input frame; and starting with a handful of labeled examples, it can label hundreds of thousands of new examples. The models trained with these discovered examples result in much better recognition rates, across multiple video datasets. Again, we only provided relevant details in Section 9.2, other details can be found in [192].

Finally, we summarize the contributions of this thesis, include a discussion on future directions it enables, and put this dissertation in the broader context of the fast-paced field of Visual Recognition.

Part I

Supervised Visual Recognition

Chapter 2

Contextual Priming & Feedback

> The situation has provided a cue; this cue has given the expert access to information stored in memory, and the information provides the answer. Intuition is nothing more and nothing less than recognition.
>
> Herbert A. Simon

The field of object recognition has changed drastically over the past few years. We have moved from manually designed features [54, 79] to learned ConvNet features [96, 119, 155, 258]; from the original sliding window approaches [79, 292] to region proposals [95, 96, 103, 223, 298]; and from pipeline based frameworks such as Region-based CNN (R-CNN) [96] to more end-to-end learning frameworks such as Fast [95] and Faster R-CNN [223]. The performance has continued to soar higher, and things have never looked better. There seems to be a growing consensus – powerful representations learned by ConvNets are well suited for this task, and designing and learning deeper networks lead to better performance.

Most recent gains in the field have come from bottom-up, feedforward framework of ConvNets. On the other hand, in the case of human visual system, the number of feedback connections significantly outnumber the feedforward connections. In fact, many behavioral studies have shown the importance of context and top-down information for the task of object detection. This raises a few important questions – Are we on the right path as we try to develop deeper and deeper, but only feedforward networks? Is there a way we can bridge the gap between empirical results and theory, when it comes to incorporating top-down information, feedback and/or contextual reasoning in object detection?

This Chapter investigates how we can break the feedforward mold in current detection pipelines and incorporate context, feedback and top-down information.

Current detection frameworks have two components: the first component generates region proposals and the second classifies them as an object category or background. These region proposals seem to be beneficial because (a) they reduce the search space; and (b) they reduce false positives by focusing the 'attention' in right areas. In fact, this is in line with the psychological experiments that support the idea of priming (although note that while region proposals mostly use bottom-up segmentation [7, 103], top-down context provides the priming in humans [190, 287, 304]). So, as a first attempt, we propose to use top-down information in generating region proposals. Specifically, we add segmentation as a complementary task and use it to provide top-down information to guide region proposal generation and object detection. The intuition is that semantic segmentation captures contextual relationships between objects (*e.g.*, support, likelihood, size *etc.* [19]), and can guide the region proposal module to focus attention in the right areas and learn detectors from them.

But contextual priming using top-down attention mechanism is only part of the story. In case of humans, the top-down information provides feedback to the whole visual pathway (as early as V1 [130, 154]). Therefore, we further explore providing top-down feedback to the entire network in order to modulate feature extraction in all layers. This is accomplished by providing the semantic segmentation output as input to different parts of the network and training another stage of our model. The hypothesis is that equipping the network with this top-down semantic feedback would guide the visual attention of feature extractors to the regions relevant for the task at hand.

To summarize, we propose to revisit the architecture of a current state-of-the-art detector (Faster R-CNN [223]) to incorporate top-down information, feedback and contextual information. Our new architecture includes:

- **Semantic Segmentation Network:** We augment Faster R-CNN with a semantic segmentation network. We believe this segmentation can be used to provide top-down feedback to Faster R-CNN (as discussed below).
- **Contextual Priming via Semantic Segmentation:** In Faster R-CNN, both region proposal and object detection modules are feedforward. We propose to use semantic segmentation to provide top-down feedback to these modules. This is analogous to contextual priming; in this case top-down semantic feedback helps propose better regions and learn better detectors.
- **Iterative Top-Down Feedback:** We also propose to use semantic segmentation to provide top-down feedback to low-level filters, so that they become better suited for the detection problem. In particular, we use segmentation as an additional input to lower layers of a second round of Faster R-CNN.

2.1 Related Work

Object detection was once dominated by the sliding window search paradigm [79, 292]. Soon after the resurgence of ConvNets for image classification [58, 155, 167], there were attempts at using this sliding window machinery with ConvNets [71, 243, 270]; but a key limitation was the computational complexity of brute-force search.

As a consequence, there was major paradigm shift in detection which completely bypassed the exhaustive search in favor of region-based methods and object proposals [4, 4, 7, 32, 69, 103, 288, 329]. By reducing the search space, it allowed us to use sophisticated (both manually designed [48, 83, 298] and learned ConvNet [16, 96, 115, 116, 120, 180, 223]) features. Moreover, this also helped focus the attention of detectors to regions well supported by perceptual structures in the image. However, recently, Faster R-CNN [223] showed that even these region proposals can be generated by using ConvNet features. It removed segmentation from proposal pipeline by training a small network on top of ConvNet features that proposes a few object candidates. This raises an important question: Do ConvNet features already capture the structure that was earlier given by segmentation or does segmentation provide complementary information?

To answer this, we study the impact of using semantic segmentation in the region proposal and object detection modules of Faster R-CNN [223]. In fact, there has been a lot of interest in using segmentation in tandem with detection [42, 48, 64, 83]; *e.g.*, Fidler *et al.* [83] proposed to use segmentation proposals as additional features for DPM detection hypothesis. In contrast, we propose to use semantic segmentation to guide/prime the region proposal generation itself. There is ample evidence of the importance of similar top-down contextual priming in the human visual system [57, 190], and its utility in reducing areas to focus our attention on for recognizing objects [287, 304].

This prevalence and success of region proposals is only part of the story. Another key ingredient is the powerful ConvNet features [119, 155, 258]. ConvNets are multi-layered hierarchical feature extractors, inspired by visual pathways in humans [78, 154]. But so far, our focus has been on designing deeper [119, 258] feedforward architectures, even when there is a broad agreement on the importance of feedback connections [47, 94, 130] and limitations of purely feedforward recognition [161, 307] in human visual systems. Inspired by this, we investigate how can we start incorporating top-down feedback in our current object detection architectures. There have been attempts earlier at exploiting feedback mechanisms; some well known examples are auto-context [286] and inference machines [228]. These iteratively use predictions from a previous iteration to provide contextual features

to the next round of processing; however they do not trivially extend to ConvNet architectures. Closest to our goal are the contemporary works on using feedback to learn selective attention [194, 265] and using top-down iterative feedback to improve at a task at hand [33, 89, 170]. In this work, we additionally explore using top-down feedback from one task to another.

The discussion on using global top-down feedback to contextually prime object recognition is incomplete without relating it to 'context' in general, which has a long history in cognitive neuroscience [19, 124, 127, 190, 203, 204, 287, 304] and computer vision [59, 88, 197, 219, 280, 281, 282, 314]. It is widely accepted that human visual inference of objects is heavily influenced by 'context', be it contextual relationships [19, 124], priming for focusing attention [190, 287, 304] or importance of scene context [57, 127, 203, 204]. These ideas have inspired lot of computer vision research (see [59, 88] for survey). However, these approaches seldom lead to strong empirical gains. Moreover, they are mostly confined to weaker visual features (*e.g.*, [54]) and have not been explored much in ConvNet-based object detectors.

For region-based ConvNet object detectors, simple contextual features are slowly becoming popular; *e.g.*, computing local context features by expanding the region [93, 195, 196, 328], using other objects (*e.g.*, people) as context [110] and using other regions [98]. In comparison, the use of context has been much more popular for semantic segmentation. *E.g.*, CRFs are commonly used to incorporate context and post-process segmentation outputs [39, 241, 323] or to jointly reason about regions, segmentation and detection [158, 328]. More recently, RNNs have also been employed to either integrate intuitions from CRFs [174, 212, 323] in end-to-end learning systems or to capture context outside the region [16]. But empirically, at least for detection, such uses of context have mostly given feeble gains.

2.2 Preliminaries: Faster R-CNN

We first describe the two core modules of the Faster R-CNN [223] framework (Figure 2.1). The first module takes an image as input and proposes rectangular regions of interest (RoIs). The second module is the Fast R-CNN [95] (FRCN) detector that classifies these proposed regions. In this Chapter, both modules use the VGG16 [258] network, which has 13 convolutional (`conv`) and 2 fully connected (`fc`) layers. Both modules share all `conv` layers and branch out at `conv5_3`. Given an arbitrary sized image, the last `conv` feature map (`conv5_3`) is used as input to both the modules as described below.

Region Proposal Network (RPN). The region proposal module (Figure 2.1(left) in green) is a small fully convolutional network that operates on the last feature

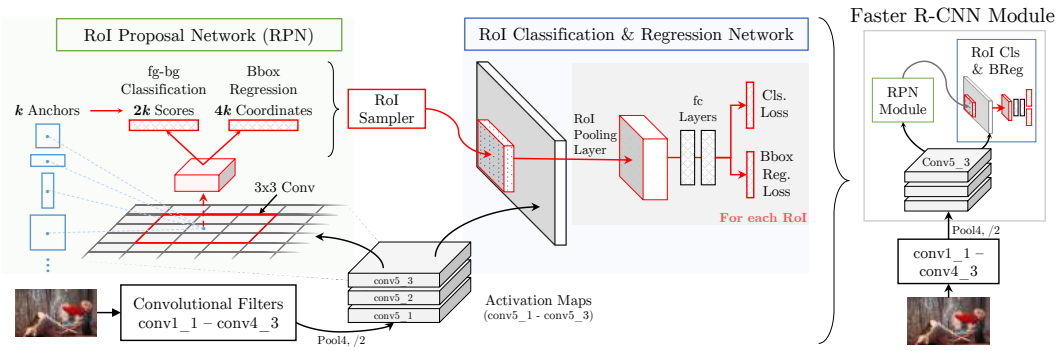


Figure 2.1 – Faster R-CNN. (left) Overview of Region Proposal Network (RPN) and RoI classification and box regression. (right) Shorthand diagram of Faster R-CNN.

map and outputs a set of rectangular object proposals, each with a score. RPN is composed of a `conv` layer and 2 sibling `fc` layers. The `conv` layer operates on the input feature map to produce a `D`-dim. output at every spatial location; which is then fed to two `fc` layers – classification (`cls`) and box-regression (`breg`). At each spatial location, RPN considers k candidate boxes (anchors) and learns to classify them as either foreground or background based on their IOU overlap with the ground-truth boxes. For foreground boxes, `breg` layer learns to regress to the closest ground-truth box. A typical setting is $\texttt{D} = 512$ and $k = 9$ (3 scales, 3 aspect-ratios) (details in [223] for details).

Using RPN regions in FRCN. For training the Fast R-CNN (FRCN) module, a mini-batch is constructed using the regions from RPN. Each region in the mini-batch is projected onto the last `conv` feature map and a fixed-length feature vector is extracted using RoI-pooling [95, 120]. Each feature is then fed to two `fc` layers, which finally give two outputs: (1) a probability distribution over object classes and background; and (2) regressed coordinates for box re-localization. An illustration is shown in Figure 2.1(left) in blue.

Training Faster R-CNN. Both RPN and FRCN modules of Faster R-CNN are trained by minimizing the multi-task loss (for classification and box-regression) from [95, 223] using mini-batch SGD. To construct a mini-batch for RPN, 256 anchors are randomly sampled with $1:1$ foreground to background ratio; and for FRCN, 128 proposals are sampled with $1:3$ ratio. We train both modules jointly using the 'approximate joint training'. For more details, refer to [95, 96, 223, 253].

Given an image during training, a forward pass through all the `conv` layers produces `conv5_3` feature map. RPN operates on this feature to propose two sets of regions, one each for training RPN and FRCN. Independent forward-backward

passes are computed for RPN and FRCN using their region sets, gradients are accumulated at `conv5_3` and back-propagated through the `conv` layers.

Why Faster R-CNN? Apart from being the current state-of-the-art object detector, Faster R-CNN is also the first framework that *learns* where to guide the 'attention' of an object detector along with the detector itself. This end-to-end learning of proposal generation and object detection provides a principled testbed for studying the proposed top-down contextual feedback mechanisms.

In the following Sections, we first describe how we add a segmentation module to Faster R-CNN (Section 2.3.1) and then present how we use segmentation for top-down contextual priming (Section 2.3.2) and iterative feedback (Section 2.3.3).

2.3 Our Approach

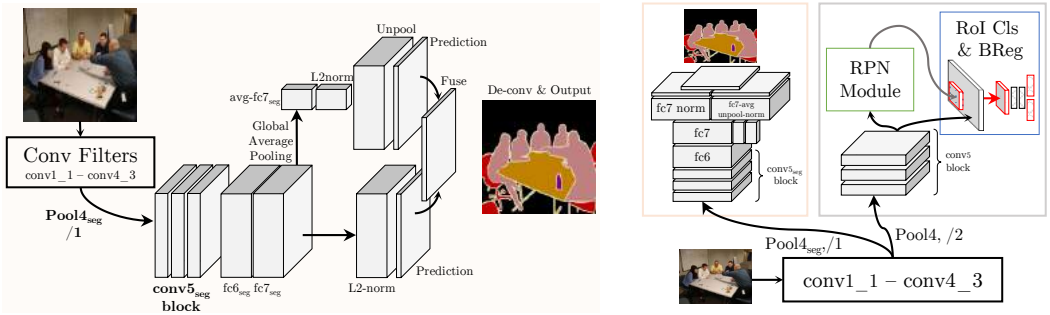
We propose to use semantic segmentation as a top-down feedback to the RPN and FRCN modules in Faster R-CNN, and iteratively to the entire network. We argue that a raw segmentation output is a compact signal that captures the desired contextual information, such as relationships between objects, along with global structures in the image; and hence is a good representation for top-down feedback.

2.3.1 Augmenting Faster R-CNN with Segmentation

The first step is to augment Faster R-CNN framework with an additional segmentation module. This module should ideally: 1) be fast, so that we do not give up the speed advantages of [95, 223]; 2) closely follow the network used by Faster R-CNN (VGG16 in this Chapter), for easy integration; and 3) use minimal (preferably no) post-processing, so that we can train it jointly with Faster R-CNN. Out of several possible architectures [12, 39, 178, 180, 323], we choose the ParseNet architecture [178] because of the simplicity.

ParseNet [178] is a fully convolutional network [180] for segmentation. It is fast because it uses filter rarefication technique (a-trous algorithm) from [39]. Its architecture is similar to VGG16. Moreover, it uses no post-processing; and instead adds an average pooling layer to incorporate global context; which is shown to have similar benefits to using CRFs [39, 174].

Architecture details. An overview is shown in Figure 2.2(a). The key difference from standard VGG16 is that the pooling after `conv4_3` ($\texttt{pool4}_{\text{seg}}$) does no down-sampling, as opposed to the standard `pool4` which down-samples by a factor of 2. After the `conv5` block, it has two 1×1 `conv` layers with 1024 channels applied with a filter stride [39, 178]. Finally, it has a global average pooling step which given



(a) ParseNet Segmentation Framework (b) Faster R-CNN with Segmentation

Figure 2.2 – (a) Overview of ParseNet. (b) Shorthand diagram of our multi-task setup (Faster R-CNN + Segmentation). Refer to Section 2.3.1 and 2.4.2 for details.

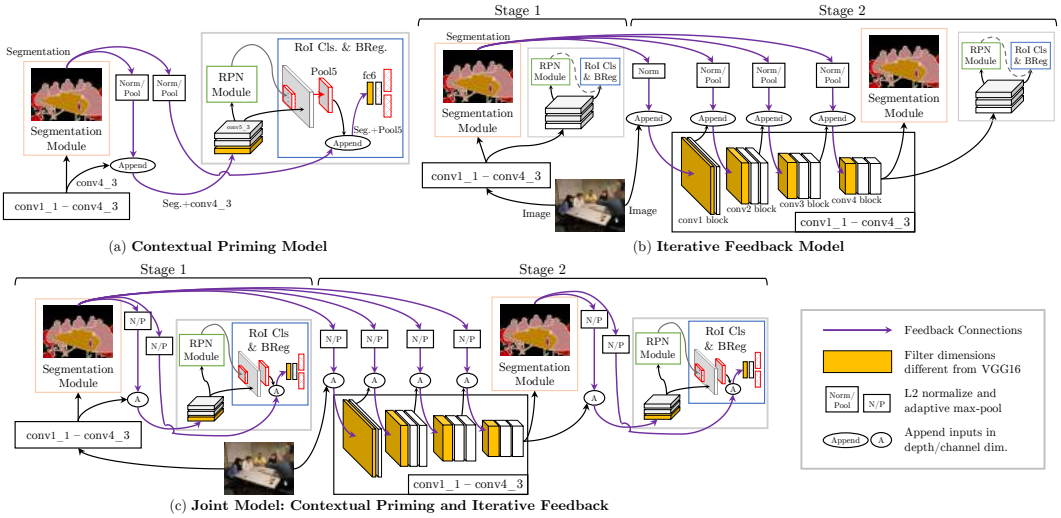
the feature map of after any layer (H×W×D) computes its spatial average (1×1×D) and 'unpools' the features. Both source and its average feature maps are normalized and used to predict per-pixel labels. These outputs are then fused and a 8× `deconv` layer is used to produce the final output.

Faster R-CNN with Segmentation – A Multi-task setup

In the joint network (Figure 2.2(b)), both the Faster R-CNN modules and the segmentation module share the first 10 `conv` layers (`conv1_1` - `conv4_3`) and differ `pool4` onwards. For the segmentation module, we branch out $\texttt{pool4}_{\text{seg}}$ layer with stride of 1 and add the remaining ParseNet layers (`conv5_1` to `deconv`)(Figure 2.2). The final architecture is a multi-task setup [193], which produces both semantic segmentation and object detection outputs simultaneously.

Training details. Now that we have a joint architecture, we can train segmentation, RPN and detection modules by minimizing a multi-task loss. However, there are some key issues: 1) Faster R-CNN can operate on an arbitrary sized input image, whereas ParseNet requires a fixed 500×500 image. In this joint framework, our segmentation module is adapted to handle arbitrary sized images; 2) Faster R-CNN and ParseNet are trained using very different set of hyperparameters (*e.g.*, learning rate schedule, batch-size *etc.*); and neither set of parameters is optimal for the other. So for joint training, we modify the hyperparameters of segmentation module and shared layers. Details on these design decisions and analysis of their impact will be presented in Section 2.4.2.

This Faster R-CNN + Segmentation framework serves as the base model on top of which we add top-down contextual feedback. We will also use this multi-task model as our primary baseline (Base-MT) as it is trained using both segmentation



(a) **Contextual Priming Model** (b) **Iterative Feedback Model** (c) **Joint Model: Contextual Priming and Iterative Feedback**

Figure 2.3 – Overview of the proposed models. (a) **Contextual Priming via Segmentation** (Section 2.3.2) uses segmentation as top-down feedback signal to guide the RPN and FRCN modules of Faster R-CNN. (b) **Iterative Feedback** (Section 2.3.3) is a 2-unit model, where the Stage-1 provides top-down feedback for Stage-2 filters. (c) **Joint Model** (Section 2.3.4) uses (a) as the base unit in (b).

and detection labels but does not have contextual feedback.

2.3.2 Contextual Priming via Segmentation

We propose to use semantic segmentation as top-down feedback to the region proposal and object detection modules of our base model. We argue that segmentation captures contextual information which will 'prime' the region proposal and object detection modules to propose better regions and learn better detectors.

In our base multi-task model, the Faster R-CNN modules operate on the `conv` feature map from the shared network. To contextually prime these modules, their input is modified to be a combination of aforementioned `conv` features and the segmentation output. Both modules can now learn to guide their operations based on the semantic segmentation of an image – it can learn to ignore background regions, find smaller objects or find large occluded objects (*e.g.*, tables) *etc.* Specifically, we take the raw segmentation output and append it to the `conv4_3` feature. The `conv5` block of filters operate on this new input ('`seg+conv4_3`') and their output is input to the individual Faster R-CNN modules. Hence, a top-down feedback signal from segmentation 'primes' both Faster R-CNN modules. However, because of the RoI-pooling operation, the detection module only sees the segmentation signal local to a particular region. To provide a global context to each region, we also append segmentation to the fixed-length feature vector ('`seg+pool5`') before feeding it to

fc6. Overview in Figure 2.3(a).

This entire system (three modules with connections between them) is trained jointly. After a forward pass through the shared conv layers and the segmentation module, their outputs are used as input to both Faster R-CNN modules. A forward-backward pass is performed for both RPN and FRCN. Next, the segmentation module does a backward pass using the gradients from its loss and from the other modules. Finally, gradients are accumulated at conv4_3 from all three modules and backward pass is performed for the shared conv layers.

Architecture details. Given an $(\mathtt{H_I} \times \mathtt{W_I} \times 3)$ input, the conv4_3 produces a $(\mathtt{H_c} \times \mathtt{W_c} \times 512)$ feature map, where $(\mathtt{H_c}, \mathtt{W_c}) \approx (\mathtt{H_I}/8, \mathtt{W_I}/8)$. Using this feature map, the segmentation module produces a $(\mathtt{H_I} \times \mathtt{W_I} \times (\mathtt{K}+1))$ output, which is a pixel-wise probability distribution over $\mathtt{K}+1$ classes. We ignore the background class and only use $(\mathtt{H_I} \times \mathtt{W_I} \times \mathtt{K})$ output, which we refer to as S. Now, S needs to be combined with conv4_3 feature for the Faster R-CNN modules and each region's $(7 \times 7 \times \mathtt{K})$-dim. pool5 feature map for FRCN, but there are 2 issues: 1) spatial dimensions of S does not match either, and 2) feature values from different layers are at drastically different scales [178]. To deal with the spatial dimension mismatch, we utilize the RoI/spatial-pooling layer from [95, 120]: We maxpool S using an adaptive grid to produce two outputs $\mathtt{S_c}$ and $\mathtt{S_p}$, which have the same spatial dimensions as conv4_3 and pool5 respectively. Now, we normalize and scale $\mathtt{S_c}$ to $\mathtt{S_{cN}}$ and $\mathtt{S_p}$ to $\mathtt{S_{pN}}$, such that their L2-norm [178] is of the same scale as the per-channel L2-norm of their corresponding features (conv4_3 and pool5 respectively). Now, we append $\mathtt{S_{cN}}$ to conv4_3 and the resulting $(\mathtt{H_c} \times \mathtt{W_c} \times (512 + \mathtt{K}))$ feature is the input for Faster R-CNN. Finally, we append $\mathtt{S_{pN}}$ with each region's pool5 and the resulting $(7 \times 7 \times (512 + \mathtt{K}))$ feature is the input for fc6 of FRCN. This network architecture is trained from a VGG16 initialized base model; and the additional K channels in conv5_3 and fc6 are initialized randomly using [100, 258]. Refer to Figure 2.3(a) for an overview.

2.3.3 Iterative Feedback via Segmentation

The architecture proposed in the previous Section provides top-down semantic feedback and modulates only the Faster R-CNN module. We also propose to provide top-down information to the whole network, especially the shared conv layers, to modulate low-level filters. The hypothesis is that this feedback will help the earlier conv layers to focus on areas likely to have objects. We again build from the Base-MT model (Section 2.3.1).

This top-down feedback is iterative in nature and will pass from one instanti-

ation of our base model to another. To provide this top-down feedback, we take the raw segmentation output of our base model (Stage-1) and append it to the input of the `conv` layer to be modulated in the second model instance (Stage-2) (see Figure 2.3(b)). *E.g.*, to modulate the first `conv` layer of Stage-2, we append the Stage-1 segmentation signal to the input image, and use this combination as the new input to `conv1_1`. This feedback mechanism is trained stage-wise: the Stage-1 model (Base-MT) is trained first; and then it is frozen and only the Stage-2 model is trained. This iterative feedback is similar to [33, 170]; the key difference being that they only focus on iteratively improving the same task, whereas in this work, we also use feedback from one task to improve another.

Architecture details. Given the pixel-wise probability output of the Stage-1 segmentation module, the background class is ignored and the remaining output (`S`) is used as the semantic feedback signal. Again, `S` needs to be resized, rescaled and/or normalized to match the spatial dimensions and the feature values scale of the input to various `conv` layers. To append with the input image, `S` is re-scaled and centered element-wise to lie in $[-127, 128]$. This results in a new $(\mathtt{H_I} \times \mathtt{W_I} \times (3 + \mathtt{K}))$ input for `conv1_1`. To modulate `conv2_1`, `conv3_1` and `conv4_1`, we `maxpool` and L2-normalize `S` to match the spatial dimensions and the feature value scales of `pool1`, `pool2` and `pool3` features respectively (similar to Section 2.3.2). The filters corresponding to additional `K` channels in `conv1_1`, `conv2_1`, `conv3_1` and `conv4_1` are initialized using [100].

2.3.4 Joint Model

So far, given our multi-task base model, we have proposed a top-down feedback for contextual priming of region proposal and object detection modules and an iterative top-down feedback mechanism to the entire architecture. Next, we put these two pieces together in a single joint framework. Our final model is a 2-unit model: each individual unit being the contextual priming model (from Section 2.3.2), and both units being connected for iterative top-down feedback (Section 2.3.3). We train this 2-unit model stage-wise (Section 2.3.3). Architecture details of the joint model follow from Section 2.3.2 and 2.3.3 (see Figure 2.3(c)).

Through extensive evaluation, presented in the following Sections, we show that: 1) individually, both contextual priming and iterative feedback models are effective and improve performance; and 2) the joint model is better than both individual models, indicating their complementary nature. We would like to highlight that our method is fairly general – both segmentation and detection modules can easily utilize newer network architectures (*e.g.*, [12, 119]).

Table 2.1 – Ablation analysis of modifying ParseNet training methodology (Section 2.4.2)

Notes	Input dim.	Learning Rates (LR) Base LR	Layer LR	LR Policy	Batch-size	#iter	Normalize Loss?	mIOU (12S val)
1) [178] (Original ParseNet)	500×500	10^{-8}	1	poly	8	20k	N	69.6
2) Reproducing [178]‡ (ParseNet)	500×500	10^{-8}	1	poly	8	20k	N	68.2
3) Faster R-CNN LR-policy, Norm. Loss	500×500	10^{-3}	1	step	8	20k	Y	68.5
4) Faster R-CNN batch-size, new LR	500×500	2.5×10^{-4}	1	step	2	80k	Y	67.8
5) Faster R-CNN Base-LR	500×500	10^{-3}	0.25	step	2	80k	Y	67.8
6) Faster R-CNN input dim. (ParseNet*)	[600×1000]†	10^{-3}	0.25	step	2	80k	Y	66

†min dim. is 600, max dim. capped at 1000.‡https://github.com/weiliu89/caffe/tree/fcn

2.4 Design and Ablation Analysis

We conduct experiments to better understand the impact of contextual priming and iterative feedback; and provide ablation analysis of various design decisions. Our implementation uses the Caffe [135] library.

2.4.1 Experimental setup

For ablation studies, we use the multi-task setup from Section 2.3.1 as our baseline (Base-MT). We also compare our method to Faster R-CNN [223] and ParseNet [178] frameworks. For quantitative evaluation, we use the standard mean average precision (mAP) [72] metric for object detection and mean intersection-over-union metric (mIOU) [72, 95] for segmentation.

Datasets. All models in this Section are trained on the PASCAL VOC12 [72] segmentation set (12S), augmented with the extra annotations (A) from [113] as is standard practice. Results are analyzed on VOC12 segmentation val set. For analysis, we chose the segmentation set, and not detection, because *all* images have *both* segmentation *and* bounding-box annotations; this helps us isolate the effects of using segmentation as top-down semantic feedback without worrying about missing segmentation labels in the standard detection split. Results on the standard splits will be presented in Section 2.5.

2.4.2 Base Model – Augmenting Faster R-CNN with Segmentation

Faster R-CNN and ParseNet both use mini-batch SGD for training, however, they follow different training methodologies. We first describe the implementation details and design decisions adopted to augment the segmentation module to Faster R-CNN and report baseline performances.

ParseNet Optimization. ParseNet is trained for 20k SGD iterations using an effective mini-batch of 8 images, an initial learning rate (LR) of 10^{-8} and polynomial LR decay policy. Compare this to Faster R-CNN, which is trained for 70k SGD

Table 2.2 – Detection results on VOC 2012 segmentation val set. All methods use VOC12S+A training set (Section 2.4.1). Legend: **S**: uses segmentation labels (Section 2.3.1), **P**: contextual priming (Section 2.3.2), **F**: iterative feedback (Section 2.3.3)

method	S	P	F	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
Fast R-CNN [95]				71.6	88.2	79.7	83.6	62.8	42.3	84.0	69.4	87.5	41.5	73.7	57.4	84.7	77.7	85.8	75.8	35.3	73.1	67.7	85.0	76.3
Faster R-CNN [223]				75.3	92.3	80.9	86.7	65.4	49.3	87.1	78.2	89.7	42.7	79.8	61.4	87.4	82.8	89.4	82.2	46.1	78.2	64.6	86.8	75.6
Base-MT (sec. 2.3.1)	✓			75.6	93.0	82.5	88.1	70.2	47.2	86.5	76.5	89.3	47.7	78.3	56.4	88.0	80.2	88.9	80.7	43.6	81.5	67.9	89.4	75.2
Ours (priming, sec. 2.3.2)	✓	✓		77.0	91.1	82.3	85.3	70.8	47.5	90.3	75.2	90.9	46.0	82.3	65.6	88.0	83.3	91.2	81.0	49.6	81.0	69.8	92.1	76.0
Ours (feedback, sec. 2.3.3)	✓		✓	77.3	90.7	82.9	90.4	70.3	51.2	89.7	77.0	91.7	49.9	81.4	66.9	87.8	81.1	90.3	82.2	50.4	79.2	70.2	85.9	76.9
Ours (joint, sec. 2.3.4)	✓	✓	✓	**77.8**	89.8	83.8	84.0	72.1	54.2	92.0	75.5	91.2	53.6	82.1	69.8	85.7	81.7	92.4	82.5	49.9	76.2	72.5	89.3	78.4

Table 2.3 – Segmentation results on VOC 2012 segmentation val set. All methods use VOC12S+A training set (Section 2.4.1). Legend: **S**: uses segmentation labels, **P**: contextual priming, **F**: iterative feedback

method	S	P	F	mIOU	bg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
ParseNet (Table 2.1(2))	✓			68.2	92.3	86.9	38.4	77.1	66.4	66.5	83.0	80.9	82.5	31.0	72.9	49.5	71.4	73.9	76.7	79.3	47.9	73.3	40.3	78.3	63.6
ParseNet* (Table 2.1(6))	✓			66.0	91.7	85.2	36.8	73.2	64.0	60.8	82.4	76.9	81.8	30.4	65.4	51.3	69.6	73.5	75.4	78.2	43.9	71.3	38.9	79.3	56.2
Base-MT (sec. 2.3.1)	✓			65.8	91.6	84.3	37.1	71.5	63.8	60.8	82.3	74.8	80.3	30.8	68.7	48.8	71.4	75.7	73.8	77.7	42.8	70.1	39.1	79.9	56.4
Ours (priming, sec. 2.3.2)	✓	✓		65.3	91.5	85.1	36.4	73.3	64.0	60.4	81.4	75.1	81.8	31.7	64.8	48.8	69.0	73.7	73.4	77.1	41.6	69.9	38.4	78.1	55.5
Ours (feedback, sec. 2.3.3)	✓		✓	69.5	92.8	87.3	39.4	76.9	66.7	68.1	86.9	80.6	86.4	33.4	68.1	50.9	71.8	80.1	77.3	81.3	48.6	73.3	42.0	82.8	65.5
Ours (joint, sec. 2.3.4)	✓	✓	✓	**69.6**	92.9	88.5	39.4	78.1	66.9	69.1	84.5	79.8	84.9	37.8	69.2	50.5	71.4	79.7	77.5	81.3	47.1	74.2	43.4	80.1	65.0

iterations with a mini-batch size of 2, 10^{-3} initial LR and step LR decay policy (step at 50k). Since we are augmenting Faster R-CNN, we try to adapt ParseNet's optimization. On the 12S val set, [178] reports 69.6% (we achieved 68.2% using the released code, Table 2.1(1-2)). We will refer to the latter as ParseNet throughout. Similar to [180], ParseNet does not normalize the Softmax loss by number of valid pixels. But to train with Faster R-CNN in a multi-task setup, all losses need to have similar magnitude; so, we normalize the loss of ParseNet and modify the LR accordingly. Next, we change the LR decay policy from polynomial to step (step at 12.5k) to match that of Faster R-CNN. These changes result in similar performance (+0.3 points, Table 2.1(2-3)). We now reduce the batch size to 2 and adjust the LR appropriately (Table 2.1(4)). To keep the base LR of Faster R-CNN and ParseNet same, we change it to 10^{-3} and modify the LR associated with each ParseNet layer to 0.25, thus keeping the same effective LR for ParseNet (Table 2.1(4-5)).

Training data. ParseNet re-scales the input images and their segmentation labels to a fixed size (500×500), thus ignoring the aspect-ratio. On the other hand, Faster R-CNN maintains the aspect-ratio and re-scales the input images such that their shorter side is 600 pixels (and the max dim. is capped at 1000). We found that ignoring the aspect-ratio drops Faster R-CNN performance and maintaining it drops the performance of ParseNet (−1.8 points, Table 2.1(5-6)). Because our main task is detection, we opted to use Faster R-CNN strategy, and treat the new ParseNet (ParseNet*) as the baseline for our base model.

Table 2.4 – Ablation analysis of Contextual Priming and Iterative Feedback on VOC 12S val set. All methods use VOC 12S+A train set for training. (left) Evaluating Priming different layers; (right) Evaluating Iterative Feedback design decisions

	mAP	mIOU
Base-MT	75.6	**65.8**
Priming `conv5_1`	76.6	**65.8**
Priming `conv5_1`, each `fc6`	**77.0**	65.3

	Stage-2 Init.	mAP	mIOU
Base-MT	-	75.6	65.8
Iterative Feedback to `conv1_1`	ImageNet	76.5	69.3
	Stage-1	76.3	69.3
Iterative Feedback to `conv{1,2,3,4}_1`	ImageNet	76.3	69.1
	Stage-1	**77.3**	**69.5**

Base Model Optimization. Following the changes mentioned above, our base model uses these standardized parameters: batch size of 2, 10^{-3} base LR, step decay policy (step at 50k), LR of 0.25 for segmentation and shared `conv` layers, and 80k SGD iterations. This model serves as our multi-task baseline (Base-MT).

Baselines. For comparison, we re-train Fast [95] and Faster R-CNN [223] on VOC 12S+A training set. We use 'approximate joint training' for Faster R-CNN, same as our method [223]. Results of the Base-MT model for detection and segmentation are reported in Table 2.2 and 2.3 respectively. Performance increases by 0.3 mAP on detection and drops by 0.1 mIOU on segmentation. These results show that just adding another task for training does not effect either modules by much. This will be our primary baseline, on top of which we will add contextual and feedback.

2.4.3 Contextual Priming

We evaluate the effects of using segmentation as top-down semantic feedback to the region proposal generation and object detection modules. We follow the same optimization hyperparameters as the Base-MT model, and report the results in Table 2.2 and 2.3. Table 2.2 shows that providing top-down feedback via priming to the Faster R-CNN modules improves its detection performance by **1.4** points over the Base-MT model and **1.7** points over Faster R-CNN. Results in Table 2.3 show that performance of segmentation drops slightly when it is used for priming.

Design Evaluation. In Table 2.4(left), we report the impact of providing segmentation signal to different modules. We see that just priming `conv5_1` gives a 1 point boost over Bast-MT and adding the segmentation signal to each individual region ('`seg+pool5`' to `fc6`) gives another 0.4 points boost. It is interesting that the segmentation performance is not affected when priming `conv5_1`, but it drops by 0.5 mIOU when we prime each region. Our hypothesis is that gradients accumulated from all regions in the mini-batch start overpowering the gradients from

Table 2.5 – Detection results on VOC 2007 detection test set. All methods are trained on union of VOC07 trainval and VOC12 trainval

method	S	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
Fast R-CNN [95]		70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
Faster R-CNN [223]		73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
Base-MT	✓	74.7	78.4	79.3	75.9	63.2	56.8	85.9	85.4	88.4	54.9	83.9	68.6	84.6	85.6	78.5	78.1	41.3	74.6	74.8	84.0	72.4
Ours (joint)	✓	**76.4**	79.3	80.5	76.8	72.0	58.2	85.1	86.5	89.3	60.6	82.2	69.2	87.0	87.2	81.6	78.2	44.6	77.9	76.7	82.4	71.9

Table 2.6 – Detection results on VOC 2012 detection test set. All methods are trained on union of VOC07 trainval, VOC07 test and VOC12 trainval

method	S	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
Fast R-CNN [95]		68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Faster R-CNN [223]		70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
Base MT	✓	71.1^	84.2	80.9	73.1	55.1	50.6	78.2	75.6	89.0	48.6	76.7	54.8	87.6	82.5	83.0	80.0	41.7	74.2	60.7	81.4	63.1
Ours (joint)	✓	**72.6**◇	84.0	81.2	75.9	60.4	51.8	81.2	77.4	90.9	50.2	77.6	58.7	88.4	83.6	82.0	80.4	41.5	75.0	64.2	82.9	65.1

^ http://host.robots.ox.ac.uk:8080/anonymous/RUZFQC.html, ◇ http://host.robots.ox.ac.uk:8080/anonymous/YFSUQA.html

segmentation. To deal with this, methods like [193] can be used in the future.

2.4.4 Iterative Feedback

Next we study the impact of giving iterative top-down semantic feedback to the entire network. In this 2-unit setup, the first unit (Stage-1) is a trained Base-MT model and the second unit (Stage-2) is a Stage-1 initialized Base-MT model (new filters are initialized randomly, see Section 2.4.4). During inference, we have the option of using the outputs from both units or just the Stage-2 unit. Given that segmentation is used as feedback, it is supposed to self-improve across units, therefore we use the Stage-2 output as our final output (similar to [33, 170]). For detection, we combine the outputs from both units; because the Stage-2 unit is modulated by segmentation, and the first unit is not, hence both might focus on different regions.

This iterative feedback improves the segmentation performance (Table 2.3) by **3.7** points over Base-MT (**3.5** points over ParseNet*). For detection, it improves over the Base-MT model by **1.7** points (**2** points over Faster R-CNN) (Table 2.2).

Design Evaluation. We study the impact of: (1) varying the degree of feedback to the Stage-2 unit, and (2) different Stage-2 initializations. In Table 2.4(right), we see that when initializing the Stage-2 unit with an ImageNet trained network, varying iterative feedback does not have much impact; however, when initializing with a Stage-1 model, providing more feedback leads to better performance. Specifically, iterative feedback to all shared `conv` layers improves both detection and segmenta-

Table 2.7 – Segmentation results on VOC 2012 segmentation test set. All methods are trained on union of VOC07 trainval, VOC07 test and VOC12 trainval

method	S	mIOU	bg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
Base MT	✓	66.4^	91.3	82.0	37.7	77.6	58.8	58.8	84.0	75.6	83.1	25.1	70.9	57.8	74.0	74.6	76.4	75.0	48.8	73.7	45.6	72.3	52.0
Ours (joint)	✓	**71.4**◇	93.0	89.3	41.4	84.1	63.8	65.2	88.1	80.9	88.6	28.4	75.4	60.6	80.3	80.9	83.1	79.7	55.4	77.9	48.2	75.8	58.8

^http://host.robots.ox.ac.uk:8080/anonymous/RUZFQC.html, ◇http://host.robots.ox.ac.uk:8080/anonymous/YFSUQA.html

tion by 1.7 mAP and 3.7 mIOU respectively, as opposed to feedback to just `conv1_1` (as in [33, 170]) which results in lower gains (Table 2.4,right). Our hypothesis is that iterative feedback to a Stage-1 initialized unit allows the network to correct its mistakes and/or refine its predictions; therefore, providing more feedback leads to better performance.

2.4.5 Joint Model

Finally, we evaluate our joint 2-unit model, where each unit is a model with contextual priming, and both units are connected via segmentation feedback. In this setup, a trained contextual priming model is used as the Stage-1 unit as well as the initialization for the Stage-2 unit. We remove the dropout layers from Stage-2 unit. Inference follows the procedure described in Section 2.4.4.

As shown in Table 2.2, for detection, the joint model achieves **77.8**% mAP (**+2.2** points over Base-MT and **+2.5** points over Faster R-CNN), which is better than both priming only and feedback only models. This suggests that both forms of top-down feedback are complementary for object detection. The segmentation performance (Table 2.3) is similar to the feedback only model, which is expected since in both cases, the segmentation module receives similar feedback.

2.5 Results

We now report results on the PASCAL VOC and MS COCO [175] datasets. We also evaluate the region proposal generation on the proxy metric of average recall.

Experimental Setup. When training on the VOC datasets with extra data (Table 2.5, 2.6 and 2.7), we use 100k SGD iterations (other hyperparameters follow Section 2.4); and for MS COCO, we use 490k SGD iterations with an initial LR of 10^{-3} and decay step size of 200k, owing to a larger epoch size.

VOC07 and VOC12 Results. Table 2.5 shows that on VOC07, our joint priming and feedback model improves the detection mAP by **1.7** points over Base-MT and **3.2** points over Faster R-CNN. Similarly, on VOC12 (Table 2.6), priming and feedback lead to **1.5** points boost over Bast-MT (**2.2** over Faster R-CNN). For

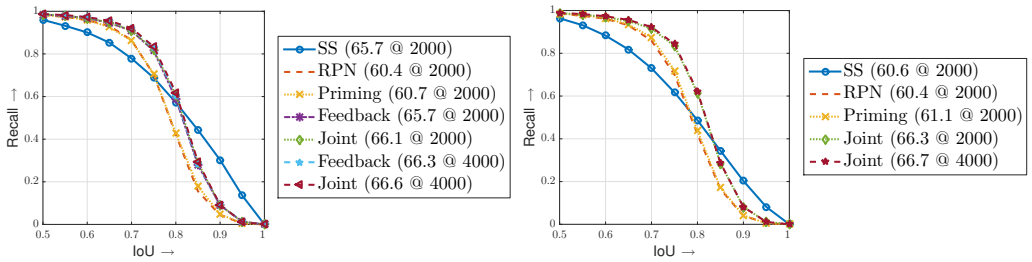


Figure 2.4 – Recall-to-IoU on VOC12 Segmentation val set (left) and VOC07 test set (right) (best viewed digitally).

Table 2.8 – Detection results on MS COCO 2015 test-dev set. All methods use COCO trainval35k for training and results were obtained from the online 2015 test-dev server. Legend: **F**: using iterative feedback, **P**: using contextual priming, **S**: uses segmentation labels

Method	S	P	F	AP, IoU:			AP, Area:			AR, # Dets:			AR, Area:		
				0.5:0.95	0.50	0.75	Small	Med.	Large	1	10	100	Small	Med.	Large
Faster R-CNN				24.5	46.0	23.7	8.2	26.4	36.9	24.0	34.8	35.5	13.4	39.2	54.3
Base-MT	✓			25.0	47.0	24.2	8.1	27.1	38.1	24.3	35.1	35.8	13.2	39.8	55.0
Ours (priming)	✓	✓		25.8	48.2	25.3	8.3	27.8	38.6	24.5	35.7	36.5	13.6	40.6	54.7
Ours (joint)	✓	✓	✓	27.5	49.2	27.8	8.9	29.5	41.5	25.5	37.4	38.3	14.6	42.5	57.4

segmentation on VOC12 (Table 2.7), we see a huge **5** point boost in mIOU over Base-MT. We would like highlight that both Base-MT and our joint model use exactly the same annotations and hyperparameters; therefore the performance boosts are because of contextual priming and iterative feedback in our model.

Recall-to-IOU. Since our hypothesis is that priming and feedback lead to better proposal generation, we also evaluate the recall of region proposals by the RPN module from various models, at different IOU thresholds. In Figure 2.4, we show the results of using 2000 proposal per RPN module. Since feedback models have 2 units, we report their number with both 4000 and top 2000 proposals (sorted by `cls` score). As can be seen priming, feedback and joint models all lead to higher average recall (shown in legend) over the baseline RPN module.

MS COCO Results. We also perform additional analysis of contextual priming on the COCO [175] dataset and report AP (averaged over classes, recall, and IoU levels) in Table 2.8. Our priming model results in +1.2 AP points (+2.1 AP50) over Faster R-CNN and +0.8 AP points (+1.1 AP50) over Base-MT on the COCO test-dev set [175]. On further analysis, we notice that the most performance gains are for objects where context should intuitively help; *e.g.*, +12.4 for 'parking-meter', +8.7 for 'suitcase', +8.3 for 'umbrella' *etc.* on AP50 wrt. to Faster R-CNN. Finally, our joint model achieves **27.5** AP points (+**3** AP points over Faster R-CNN and

+**2.5** over Base-MT), further highlighting effectiveness of the proposed method.

Conclusion. We presented and investigated how we can incorporate top-down semantic feedback in the state-of-the-art Faster R-CNN framework. We proposed to augment a segmentation network to Faster R-CNN, which is then used to provide top-down contextual feedback to the region proposal generation and object detection modules. We also use this segmentation network to provide top-down feedback to the entire Faster R-CNN network iteratively. Our results demonstrate the effectiveness of these top-down feedback mechanisms for the tasks of region proposal generation, object detection and semantic segmentation.

Chapter 3

Top-Down Modulation

> Details are confusing. It is only by selection, by elimination, by emphasis, that we get at the real meaning of things.
>
> Georgia O'Keeffe

Figure 3.1 – Detecting objects such as the bottle or remote shown above requires low-level finer details as well as high-level contextual information. In this Chapter, we propose a top-down modulation (TDM) network, which can be used with any bottom-up, feedforward ConvNet. We show that the features learnt by our approach lead to significantly improved object detection.

As we saw in the previous Chapter, ConvNet representations have revolutionized the field of object detection [16, 93, 95, 96, 223, 253, 270]. Most of these ConvNets are bottom-up, feedforward architectures constructed using repeated convolutional layers (with non-linearities) and pooling operations [119, 155, 258, 271, 272]. These convolutional layers learn invariances and the spatial pooling increases the receptive field of subsequent layers; thus resulting in a coarse, highly semantic representation at the final layer.

However, consider the images shown in Figure 3.1. Detecting and recognizing

an object like the bottle in the left image or remote in the right image requires extraction of very fine details such as the vertical and horizontal parallel edges. But these are exactly the type of edges ConvNets try to gain invariance against in early convolutional layers. One can argue that ConvNets can learn not to ignore such edges when in context of other objects like table. However, objects such as table do not emerge until very late in feed-forward architecture. So, how can we incorporate these fine details in object detection?

A popular solution is to use variants of 'skip' connections [16, 75, 116, 180, 242, 310], that capture these finer details from lower convolutional layers with *local* receptive fields. But simply incorporating high-dimensional skip features into detection does not yield significant improvements due to overfitting caused by curse of dimensionality. What we need is a selection/attention mechanism that selects the relevant features from lower convolutional layers.

We believe the answer lies in the process of *top-down modulation*. In the human visual pathway, once receptive field properties are tuned using feedforward processing, *top-down modulations* are evoked by feedback and horizontal connections [154, 162]. These connections modulate representations at multiple levels [90, 94, 211, 317, 318] and are responsible for their selective combination [47, 128]. We argue that the use of skip connections is a special case of this process, where the *modulation* is relegated to the final classifier, which directly tries to influence lower layer features and/or learn how to combine them.

In this Chapter, we propose to incorporate the *top-down modulation* process in the ConvNet itself. Our approach supplements the standard bottom-up, feedforward ConvNet with a top-down network, connected using lateral connections. These connections are responsible for the *modulation* and *selection* of the lower layer filters, and the top-down network handles the *integration* of features.

Specifically, after a bottom-up ConvNet pass, the final high-level semantic features are transmitted back by the top-down network. Bottom-up features at intermediate depths, after lateral processing, are combined with the top-down features, and this combination is further transmitted down by the top-down network. Capacity of the new representation is determined by lateral and top-down connections, and optionally, the top-down connections can increase the spatial resolution of features. These final, possibly high-res, top-down features inherently have a combination of *local* and *larger* receptive fields.

The proposed Top-Down Modulation (TDM) network is trained end-to-end and can be readily applied to any base ConvNet architecture (*e.g.*, VGG [258], ResNet [119], Inception-Resnet [272] *etc.*). To demonstrate its effectiveness, we use

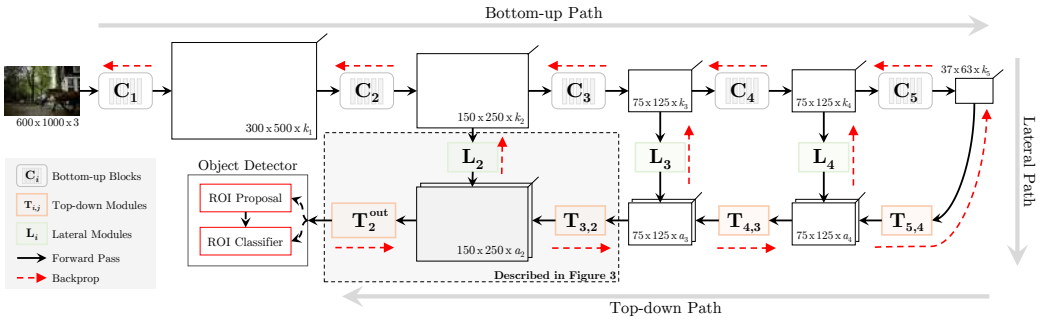


Figure 3.2 – The illustration shows an example of **Top-Down Modulation** (TDM) Network, which is integrated with the bottom-up network with lateral connections. $\mathbf{C}_i$ are bottom-up, feedforward feature blocks, $\mathbf{L}_i$ are the lateral modules which transform low level features for the top-down contextual pathway. Finally, $\mathbf{T}_{j,i}$, which represent flow of top-down information from index j to i. Individual components are explained in Figure 3.3 and 3.4.

the proposed network in the standard Faster R-CNN detection method [223] and evaluate on the challenging COCO benchmark [175]. We report a consistent and significant boost in performance on all metrics across network architectures. TDM network increases the performance of vanilla Faster R-CNN with: (a) VGG16 from 23.3 AP to **28.6** AP, (b) ResNet101 from 31.5 AP to **35.2** AP, and (c) InceptionResNetv2 from 34.7 AP to **37.2** AP. These are the best performances reported to-date for these architectures without any bells and whistles (*e.g.*, multi-scale features, iterative box-refinement). Furthermore, we see drastic improvements in small objects (*e.g.*, +**4.5** AP) and in objects where selection of fine details using top-down context is important.

Apart from how the architecture is designed, a difference from the previous Chapter is that in TDM, we do not rely of semantic segmentation of an image to act as a proxy that provides this top-down contextual feedback.

3.1 Related Work

After the resurgence of ConvNets for image classification [58, 155], they have been successfully adopted for a variety of computer vision tasks such as object detection [95, 96, 223, 270], semantic segmentation [12, 39, 178, 180], instance segmentation [115, 116, 212], pose estimation [279, 285], depth estimation [67, 299], edge detection [310], optical flow predictions [84, 220] *etc.* However, by construction, final ConvNet features lack the finer details that are captured by lower convolutional layers. These finer details are considered necessary for a variety of recognition tasks, such as accurate object localization and segmentation.

To counter this, 'skip' connections have been widely used with ConvNets. Though the specifics of methods vary widely, the underlying principle is same: using or combining finer features from lower layers and coarse semantic features for higher layers. For example, [16, 75, 116, 242] combine features from multiple layers for the final classifier; while [16, 242] use subsampled features from finer scales, [75, 116] upsample the features to the finest scale and use their combination. Instead of combining features, [178, 180, 310] do independent predictions at multiple layers and average the results. In our proposed framework, such upsampling, subsampling and fusion operations can be easily controlled by the lateral and top-down connections.

The proposed TDM network is conceptually similar to the strategies explored in other contemporary works [12, 176, 213, 220, 226]. All methods, including ours, propose architectures that go beyond the standard skip-connection paradigm and/or follow a coarse-to-fine strategy when using features from multiple levels of the bottom-up feature hierarchy. However, different methods focus on different tasks which guide their architectural design and training methodology.

Conv-deconv [198] and encoder-decoder style networks [12, 226] have been used for image segmentation to utilize finer features via lateral connections. These connections generally use the 'unpool' operation [319], which merely inverts the spatial pooling operation. Moreover, there is no modulation of bottom-up network. In comparison, our formulation is more generic and is responsible for the flow of high-level context features [211].

Pinheiro *et al.* [213] focus on refining class-agnostic object proposals by first selecting proposals *purely* based on bottom-up feedforward features [212], and then *post-hoc* learning how to refine each proposal independently using top-down and lateral connections (due to the computational complexity, only a few proposals can be selected to be refined). We argue that this use of top-down and lateral connections for refinement is sub-optimal for detection because it relies on the proposals selected based on feedforward features, which are insufficient to represent small and difficult objects. This training methodology also limits the ability to update the feedforward network through lateral connections. In contrast to this, we propose to learn better features for recognition tasks in an end-to-end trained system, and these features are used for both proposal generation and object detection. Similar to the idea of coarse-to-fine refinement, Ranjan and Black [220] propose a coarse-to-fine spatial pyramid network, which computes a low resolution residual optical flow and iteratively improves predictions with finer pyramid levels. This is akin to *only* using a specialized top-down network, which is suited for low-level tasks (like optical flow) but not for recognition tasks. Moreover, such purely coarse-to-fine networks cannot

utilize models pre-trained on large-scale datasets [58], which is important for recognition tasks [96]. Therefore, our approach learns representation using bottom-up (fine-to-coarse), top-down (coarse-to-fine) and lateral networks simultaneously, and can use different pre-trained modules.

The proposed top-down network is closest to the recent work of Lin *et al.* [176], developed concurrently to ours, on feature pyramid network for object detection. Lin *et al.* [176] use bottom-up, top-down and lateral connections to learn a feature pyramid, and require multiple proposal generators and region classifiers on each level of the pyramid. In comparison, the proposed top-down modulation focuses on using these connections to learn a single final feature map that is used by a single proposal generator and region classifier.

There is strong evidence of such top-down context, feedback and lateral processing in the human visual pathway [47, 90, 94, 128, 154, 161, 162, 211, 317, 318]; wherein, the top-down signals are responsible for modulating low-level features [90, 94, 211, 317, 318] as well as act as attentional mechanism for selection of features [47, 128]. In this Chapter, we propose a computation model that captures some of these intuitions and incorporates them in a standard ConvNets, giving substantial performance improvements.

Our top-down framework is also related to the process of contextual feedback [11]. To incorporate top-down feedback loop in ConvNets, contemporary works [33, 89, 170, 250], have used 'unrolled' networks (trained stage-wise). Even in Chapter 6, we used semantic segmentation as a proxy for top-down feedback. The proposed top-down network with lateral connections explores a complementary paradigm and can be readily combined with them. Contextual features have also been used for ConvNets based object detectors; *e.g.*, using other objects [110] or regions [98] as context. We believe the proposed top-down path can naturally transmit these contextual features.

3.2 Top-Down Modulation (TDM)

Our goal is to incorporate *top-down modulation* into current object detection frameworks. The key idea is to select/attend to fine details from lower level feature maps based on top-down contextual features and select top-down contextual features based on the fine low-level details. We formalize this by proposing a simple top-down modulation (TDM) network as shown in Figure 3.2.

The TDM network starts from the last layer of bottom-up feedforward network. For example, in the case of VGG16, the input to the first layer of the TDM

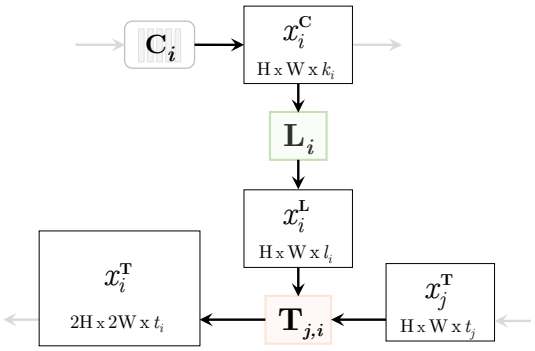


Figure 3.3 – The basic building blocks of Top-Down Modulation Network (detailed Section 3.2.1).

network is the `conv5_3` output. Every layer of TDM network also gets the bottom-up features as inputs via lateral connections. Thus, the TDM network learns to: (a) transmit high-level contextual features that guide the learning and selection of relevant low-level features, and (b) use the bottom-up features to select the contextual information to transmit. The output of the proposed network captures both pertinent finer details and high-level information.

3.2.1 Proposed Architecture

An overview of the proposed framework is illustrated in Figure 3.2. The standard bottom-up network is represented by blocks of layers, where each block $\mathbf{C}_i$ has multiple operations. The TDM network hinges on two key components: a lateral module $\mathbf{L}$, and a top-down module $\mathbf{T}$ (see Figure 3.3). Each lateral module $\mathbf{L}_i$ takes in a bottom-up feature $x_i^{\mathbf{C}}$ (output of $\mathbf{C}_i$) and produces the corresponding lateral feature $x_i^{\mathbf{L}}$. These lateral features $x_i^{\mathbf{L}}$ and top-down features $x_j^{\mathbf{T}}$ are combined, and optionally upsampled, by the $\mathbf{T}_{j,i}$ module to produce the top-down features $x_i^{\mathbf{T}}$. These modules, $\mathbf{T}_{j,i}$ and $\mathbf{L}_i$, control the capacity of the modulation network by changing their output feature dimensions.

The feature from the last top-down module $\mathbf{T}_i^{\text{out}}$ is used for the task of object detection. For example, in Figure 3.2, instead of $x_5^{\mathbf{C}}$, we use $\mathbf{T}_2^{\text{out}}$ as input to ROI proposal and ROI classifier networks of the Faster R-CNN [223] detection system (discussed in Section 3.3.1). During training, gradient updates from the object detector backpropagate via top-down and lateral modules to the $\mathbf{C}_i$ blocks. The lateral modules $\mathbf{L}_\bullet$ learn how to transform low-level features and the top-down modules $\mathbf{T}_\bullet$ learn what semantic or context information to preserve in the top-down feature transmission as well as the selection of relevant low-level lateral features. Ultimately, the bottom-up features are modulated to adapt for this new representation.

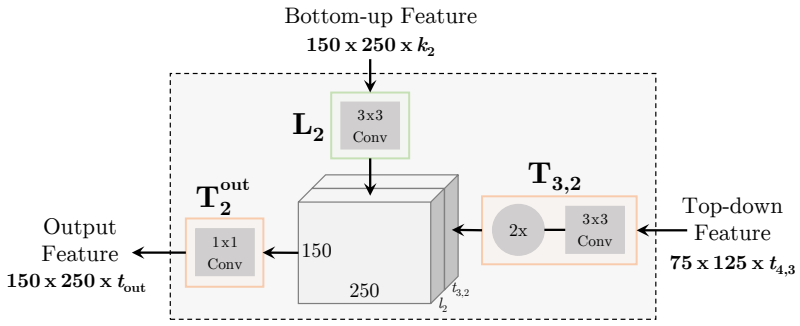


Figure 3.4 – An example with details of top-down modules and lateral connections. Please see Section 3.2.1 for details of the architecture.

Architecture details. The top-down and lateral modules described above are essentially small ConvNets, which can vary from a single or a hierarchy of convolutional layers to more involved blocks with Residual [119] or Inception [272] units. In this Chapter, we limit our study by using modules with a single convolutional layer and non-linearity to analyze the impact of top-down modulation.

A detailed example of lateral and top-down modules is illustrated in Figure 3.4. The lateral module $\mathbf{L}_i$ is a 3×3 convolutional layer with ReLU non-linearity, which transforms an $(\mathtt{H}_i \times \mathtt{W}_i \times k_i)$ input $x_i^{\mathbf{C}}$, to $(\mathtt{H}_i \times \mathtt{W}_i \times l_i)$ lateral feature $x_i^{\mathbf{L}}$. The top-down module $\mathbf{T}_{j,i}$ is also a 3×3 convolutional layer with ReLU, that combines this lateral feature with $(\mathtt{H}_i \times \mathtt{W}_i \times t_j)$ top-down feature $x_j^{\mathbf{T}}$, to produce an intermediate output $(\mathtt{H}_i \times \mathtt{W}_i \times t_i)$. If the resolution of next lateral feature $x_{i-1}^{\mathbf{L}}$ is higher than the previous lateral feature (*e.g.*, $\mathtt{H}_{i-1} = 2 \times \mathtt{H}_i$), then $\mathbf{T}_{j,i}$ also upsamples the intermediate output to produce $(\mathtt{H}_{i-1} \times \mathtt{W}_{i-1} \times t_i)$ top-down feature $x_i^{\mathbf{T}}$. In Figure 3.2, we denote $a_i = t_j + l_i$ for simplicity. The final $\mathbf{T}_i^{\text{out}}$ module can additionally have a 1×1 convolutional layer with ReLU to output a $(\mathtt{H}_i \times \mathtt{W}_i \times k_{\text{out}})$ feature, which is used by the detection system.

Varying l_i, t_i and k_{out} controls the capacity of the top-down modulation system and dimension of the output features. These hyperparameters are governed by the base network design, detection system and hardware constraints (discussed in Section 3.3.3). Notice that the upsampling step is optional and depends on the content and arrangement of $\mathbf{C}$ blocks (*e.g.*, no upsampling by $\mathbf{T}_4$ in Figure 3.2). Also, the first top-down module ($\mathbf{T}_5$ in the illustration) only operates on $x_5^{\mathbf{C}}$ (the final output of the bottom-up network).

Training methodology. Integrating top-down modulation framework into a bottom-up ConvNet is only meaningful when the latter can represent high-level concepts in higher layers. Thus, we typically start with a pre-trained bottom-up network (see

Table 3.1 – Base networks architecture details for VGG16, ResNet101 and InceptionResNetv2. Legend: $\mathbf{C}_i$: bottom-up block id, N: number of convolutional filters, NR: number of residual units, NI: number of inception-resnet units, $\texttt{dim}\left(x_i^{\mathbf{C}}\right)$: Resolution and dimensions of the output feature. Refer to [119, 129, 258, 272] for details

VGG16			
name	$\mathbf{C}_i$	N	$\texttt{dim}\left(x_i^{\mathbf{C}}\right)$
conv1_x	$\mathbf{C}_1$	2	$(300, 500, 64)$
conv2_x	$\mathbf{C}_2$	2	$(150, 250, 128)$
conv3_x	$\mathbf{C}_3$	3	$(75, 125, 256)$
conv4_x	$\mathbf{C}_4$	3	$(37, 63, 512)$
conv5_x	$\mathbf{C}_5$	3	$(37, 63, 512)$

ResNet101				
name	$\mathbf{C}_i$	NB	N	$\texttt{dim}\left(x_i^{\mathbf{C}}\right)$
conv1	$\mathbf{C}_1$	1	1	$(300, 500, 64)$
conv2_x	$\mathbf{C}_2$	3	9	$(150, 250, 256)$
conv3_x	$\mathbf{C}_3$	4	12	$(75, 125, 512)$
conv4_x	$\mathbf{C}_4$	23	69	$(75, 125, 1024)$

InceptionResNetv2				
name	$\mathbf{C}_i$	NI	N	$\texttt{dim}\left(x_i^{\mathbf{C}}\right)$
conv_x	$\mathbf{C}_x$	-	5	$(71, 246, 192)$
Mixed_5b	$\mathbf{M}_5$	1	7	$(35, 122, 320)$
Block_10x	$\mathbf{B}_{10}$	10	70	$(35, 122, 320)$
Mixed_6a	$\mathbf{M}_{6a}$	1	3	$(33, 120, 1088)$
Block_20x	$\mathbf{B}_{20}$	20	100	$(33, 120, 1088)$

Section 3.5.4 for discussion). Starting with this pre-trained network, we find that progressively building the top-down network performs better in general. Therefore, we add one new pair of lateral and top-down modules at a time. For example, for the illustration in Figure 3.2, we will begin by adding $(\mathbf{L}_4, \mathbf{T}_{5,4})$ and use $\mathbf{T}_4^{\text{out}}$ to get features for object detection. After training $(\mathbf{L}_4, \mathbf{T}_{5,4})$ modules, we will add the next pair $(\mathbf{L}_3, \mathbf{T}_{4,3})$ and use a new $\mathbf{T}_3^{\text{out}}$ module to get features for detection; and we will repeat this process. With each new pair, the entire network, top-down and bottom-up along with lateral connections, is trained end-to-end. Implementation details of this training methodology depends on the base network architecture, and will be discussed in Section 3.3.3.

To better understand the impact of the proposed TDM network, we conduct extensive experimental evaluation; and provide ablation analysis of various design decisions. We describe our approach in detail (including preliminaries and implementation details) in Section 3.3 and present our results in Section 3.4. We also report ablation analysis in Section 3.5. We would like to highlight that the proposed framework leads to substantial performance gains across different base network architectures, indicating its wide applicability.

3.3 Approach Details

In this Section, we describe the preliminaries and provide implementation details of our top-down modulation (TDM) network under various settings. We first give a a brief overview of the object detection system and the ConvNet architectures used throughout this Chapter.

3.3.1 Preliminaries: Faster R-CNN

We use the Faster R-CNN [223] framework as our base object detection system (we only recap relevant terminology here; please refer to Section 2.2 for details). Faster R-CNN consists of two core modules: 1) ROI Proposal Network (RPN),

which takes an image as input and proposes rectangular regions of interests (ROIs); and 2) ROI Classifier Network (RCN), which is a Fast R-CNN [95] detector that classifies these proposed regions and learns to refine ROI coordinates. Given an image, Faster R-CNN first uses a ConvNet to extract features that are shared by both RPN and RCN. RPN uses these features to propose candidate ROIs, which are then classified by RCN. The RCN network projects each ROI onto the shared feature map and performs the 'ROI Pooling' [95, 120] operation to extract a fixed length representation. Finally, this feature is used for classification and box regression. See [95, 120, 129, 223] for details.

Due to lack of support for recent ConvNet architectures [119, 272] in the Faster R-CNN framework, we use our implementation in Tensorflow [1]. We follow the design choices outlined in [129]. In Section 3.4, we will provide performance numbers using both the released code [223] as well as our implementation (which tends to generally perform better). We use the end-to-end training paradigm for Faster R-CNN for all experiments [223]. Unless specified otherwise, all methods start with models that were pre-trained on ImageNet classification [58, 96].

3.3.2 Preliminaries: Base Network Architectures

In this Chapter, we use three standard ConvNet architectures: VGG16 [258], ResNet101 [119] and InceptionResNetv2 [272]. We briefly explain how they are incorporated in the Faster R-CNN framework (see [119, 129, 223] for details), and give a quick overview of these architectures with reference to the bottom-up blocks $\mathbf{C}$ from Section 3.2.1.

We use the term 'Base Network' to refer to the part of ConvNet that is shared by both RPN and RCN; and 'Classifier Network' to refer to the part that is used as RCN. For VGG16 [223, 258], ConvNet till `conv5_3` is used as the base network, and the following two `fc` layers are used as the classifier network. Similarly, for ResNet101 [119, 223], base network is the ConvNet till `conv4_x`, and classifier network is the `conv5_x` block (with 3 residual units or 9 convolutional layers). For InceptionResNet101v2 [129, 272], ConvNet till the 'Block_20x' is used as the base network, and the remaining layers ('Mixed_7a' and 'Block_9x', with a total of 11 inception-resnet units or 48 convolutional layers) are used as the classifier network. Following [129], we change the pooling stride of the penultimate convolutional block in ResNet101 and InceptionResNetv2 to 1 to maintain spatial resolution, and use atrous [39, 178] convolution to recover the original field-of-view. Properties of bottom-up blocks $\mathbf{C}_i$, including number of layers, the output feature resolution and feature dimension *etc.* are given in Table 3.1.

Table 3.2 – Top-Down Modulation network design for VGG16, ResNet101 and InceptionResNetv2. Notice that t_{out} VGG16 is much smaller than 512, thus requiring fewer parameters in RPN and RCN modules. Also note that it is important to keep t_{out} fixed for ResNet101 and InceptionResNetv2 in order to utilize pre-trained RPN and RCN modules

VGG16					ResNet101					InceptionResNetv2				
$\mathbf{T}_{i,j}$	$\mathbf{L}_j$	$t_{i,j}$	l_j	t_{out}	$\mathbf{T}_{i,j}$	$\mathbf{L}_j$	$t_{i,j}$	l_j	t_{out}	$\mathbf{T}_{i,j}$	$\mathbf{L}_j$	$t_{i,j}$	l_j	t_{out}
$\mathbf{T}_{5,4}$	$\mathbf{L}_4$	128	128	256	$\mathbf{T}_{4,3}$	$\mathbf{L}_3$	128	128	1024	$\mathbf{T}_{\text{B}_{20},\text{6a}}$	$\mathbf{L}_{\text{6a}}$	576	512	1088
$\mathbf{T}_{4,3}$	$\mathbf{L}_3$	64	64	128	$\mathbf{T}_{3,2}$	$\mathbf{L}_2$	128	128	1024	$\mathbf{T}_{\text{6a},\text{5b}}$	$\mathbf{L}_{\text{5b}}$	512	256	1088
$\mathbf{T}_{3,2}$	$\mathbf{L}_2$	64	64	128	$\mathbf{T}_{2,1}$	$\mathbf{L}_1$	32	32	1024					
$\mathbf{T}_{2,1}$	$\mathbf{L}_1$	64	64	128										

Table 3.3 – Object detection results on the COCO benchmark. Different methods use different networks for region proposal generation (ROINet) and for region classification (ClsNet). Results for the top block (except Faster R-CNN⋆) were directly obtained from their respective publications [119, 213, 223, 250]. Faster R-CNN⋆ was reproduced by us. Middle block shows our implementation of Faster R-CNN framework, which is our primary baseline. Bottom block presents the main results of TDM network, with the **state-of-the-art** single-model performance (when [254] was published) highlighted

Method	**train**	**test**	**ROINet**	**ClsNet**	AP	AP^{50}	AP^{75}	AP^{S}	AP^{M}	AP^{L}	AR^{1}	AR^{10}	AR^{100}	AR^{S}	AR^{M}	AR^{L}
Faster R-CNN [223]	train	val	VGG16	VGG16	21.2	41.5	-	-	-	-	-	-	-	-	-	-
Faster R-CNN [119]	train	val	ResNet101	ResNet101	27.2	48.4	-	-	-	-	-	-	-	-	-	-
SharpMask [213]	train	testdev	ResNet50	VGG16	25.2	43.4	-	-	-	-	-	-	-	-	-	-
Faster R-CNN [223]⋆	trainval	testdev	VGG16	VGG16	24.5	46.0	23.7	8.2	26.4	36.9	24.0	34.8	35.5	13.4	39.2	54.3
Chapter 2 [250]	trainval	testdev	VGG16++	VGG16++	27.5	49.2	27.8	8.9	29.5	41.5	25.5	37.4	38.3	14.6	42.5	57.4
Faster R-CNN	trainval⋆	testdev	VGG16	VGG16	23.3	44.7	21.5	9.4	27.1	32.0	22.7	36.8	39.4	18.3	44.0	56.2
Faster R-CNN	trainval⋆	testdev	ResNet101	ResNet101	31.5	52.8	33.3	13.6	35.4	44.5	28.0	43.6	45.8	22.7	51.2	64.1
Faster R-CNN	trainval⋆	testdev	IRNv2	IRNv2	34.7	55.5	36.7	13.5	38.1	52.0	29.8	46.2	48.9	23.2	54.3	70.8
TDM [ours]	trainval⋆	testdev	VGG16 + TDM	VGG16 + TDM	28.6	48.1	30.4	14.2	31.8	36.9	26.2	42.2	44.2	23.7	48.3	59.3
TDM [ours]	trainval⋆	testdev	ResNet101 + TDM	ResNet101 + TDM	35.2	55.3	38.1	16.6	38.4	47.9	30.4	47.8	50.3	27.8	54.9	67.6
TDM [ours]	trainval⋆	testdev	IRNv2 + TDM	IRNv2 + TDM	**37.3**	**57.8**	**39.8**	**17.1**	**40.3**	**52.1**	**31.6**	**49.3**	**51.9**	**28.1**	**56.6**	**71.1**

3.3.3 Top-Down Modulation

To add the proposed TDM network to the ConvNet architectures described above, we need to decide the extent of top-down modulation, the frequency of lateral connections and the capacity of $\mathbf{T}$, $\mathbf{L}$ and $\mathbf{T}^{\text{out}}$ modules. We try to follow these principles when making design decisions: (a) coarse semantic modules need larger capacity; (b) lateral and top-down connections should reduce feature dimensionality in order to force selection; and (c) the capacity of $\mathbf{T}^{\text{out}}$ should be informed by the Proposal (RPN) and Classifier (RCN) Network design. Finally, the hardware constraint that a TDM augmented ConvNet should fit on a standard GPU.

To build a TDM network, we start with a standard bottom-up model trained on the detection task, and add $(\mathbf{T}_{i,j}, \mathbf{L}_j)$ progressively. The capacity for different $\mathbf{T}$, $\mathbf{L}$, and $\mathbf{T}^{\text{out}}$ modules is given in Table 3.2. For the VGG16 network, we add top-down and lateral modules all the way to the `conv1_x` feature. Notice that the input

feature dimension to RPN and RCN networks changes from 512 (for `conv5_x`) to 256 (for $\mathbf{T}_4^{\text{out}}$), therefore we initialize the `fc` layers in RPN and RCN randomly [100]. However, since t_{out} is same for the last three $\mathbf{T}^{\text{out}}$ modules, we re-use the RPN and RCN layers for these modules.

For the ResNet101 and InceptionResNetv2, we add top-down and lateral modules for till `conv1` and 'Mixed_5b' respectively. Similar to VGG16, their base networks are initialized with a model pre-trained on the detection task. However, as opposed to VGG16, where the RCN has just 2 `fc` layers, ResNet101 and InceptionResNetv2 models have an RCN with 9 and 48 convolutional layers respectively. This makes training RCN from random initialization difficult. To counter this, we ensure that all $\mathbf{T}^{\text{out}}$ output feature dimensions (t_{out}) are same, so that we can be readily use pre-trained RPN and RCN. This is implemented using an additional 1×1 convolutional layer wherever $(t_{i,j} + l_j)$ differs from t_{out} (*e.g.*, all $\mathbf{T}^{\text{out}}$ modules in ResNet101, and the final $\mathbf{T}^{\text{out}}$ module in InceptionResNetv2).

We would like to highlight an issue with training of RPN at high-resolution feature maps. RPN is a fully convolutional module of Faster R-CNN, that generates an intermediate 512 dimension representation which is of the same resolution as input; and losses are computed at all pixel locations. This is efficient for coarse features (*e.g.*, last row in Table 3.1), but the training becomes prohibitively slow for finer resolution features. To counter this, we apply RPN at a stride which ensures that computation remains exactly the same (*e.g.*, using stride of 8 for $\mathbf{T}_1^{\text{out}}$ in VGG16). Because of 'ROI Pooling' operations, RCN module still efficiently utilizes the finer resolution features.

3.4 Results

In this Section, we evaluate our method on the task of object detection, and demonstrate consistent and significant improvement in performance when using features from the proposed TDM network.

Dataset and metrics. All experiments and analysis in this Chapter are performed on the COCO dataset [175]. All models were trained on 40k train and 32k val images (which we refer to as 'trainval*' set). All ablation evaluations were performed on 8k val images ('minival*' set) held out from the val set. We also report quantitative results on the standard testdev2015 split. For quantitative evaluation, we use the AP averaged over classes, recall, and IoU levels.

Experimental Setup. We conduct experiments with three standard ConvNet architectures: VGG16 [258], ResNet101 [119] and InceptionResNetv2 [272]. All models ('Baseline' Faster R-CNN and ours) were trained with SGD for 1.5M mini-batch iterations, with batch size of 256 and 128 ROIs for RPN and RCN respectively. We start with an initial learning rate of 0.001 and decay it by 0.1 at 800k and 900k iterations.

Baselines. Our primary baseline is using vanilla VGG16, ResNet101 and InceptionResNetv2 features in the Faster R-CNN framework. However, due to lack of implementations supporting all three ConvNets, we opted to re-implement Faster R-CNN in Tensorflow [1]. The baseline numbers reported in Table 3.3(middle) are using our implementation and training schedule and are generally higher than the ones reported in [223, 250]. Faster R-CNN⋆ was reproduced by us using the official implementation [223]. All other results in Table 3.3(top) were obtained from the original papers.

We also compare against models which use a single region proposal generator and a single region classifier network. In particular, we compare with SharpMask [213], because of its refinement modules with top-down and lateral connections, and [250] because they also augment the standard VGG16 network with top-down information. Note that different methods use different networks and train/test splits (see Table 3.3), making it difficult to do a comprehensive comparison. Therefore, for discussion, we will directly compare against our Faster R-CNN baseline (Table 3.3(middle)), and highlight that the improvements obtained by our approach are much bigger than the boosts by other methods.

3.4.1 COCO Results

In Table 3.3(bottom), we report results of the proposed TDM network on the testdev2015 split of the COCO dataset. We see that the TDM network leads to a **5.3** AP point boost over the vanilla VGG16 network (28.6 AP *vs.* 23.3 AP), indicating that TDM learns much better features for object detection. Note that even though our algorithm is trained on less data (trainval∗), we outperform all methods with VGG16 architecture. For ResNet101, we improve the performance by **3.7** points to 35.2 AP. InceptionResNetv2 [272] architecture was the cornerstone of the winning entry to COCO 2016 detection challenge [129]. The best single model performance used by this entry achieves 34.7 AP on the testdev split ([129]). Using InceptionResNetv2 as base, our TDM network achieves **37.3** AP, which is currently the **state-of-the-art** single model performance on the testdev split without any bells and whistles (*e.g.*, multi-scale, iterative box refinement, *etc.*). In fact, the

TDM network outperforms the baselines (with same base networks) on almost all AP and AR metrics. Similarly, in Table 3.5, we observe that TDM achieves similar boosts across all network architectures on the minival∗ split as well.

Figure 3.5 shows change in AP from Faster R-CNN baseline to the TDM network (for the testdev split). When using VGG16, for all but one category, TDM features improve the performance on object detection. In fact, more than 50% of the categories improve by 5 AP points or more, highlighting that the features are good for small and big objects alike. Similar trends hold for ResNet101 and InceptionResNetv2.

In Table 3.4, we report AP results on the COCO testdev split for all classes. Results are reported for baseline Faster R-CNN and our TDM network with different base network architectures. The proposed TDM network results in **substantial** improvement in AP across all network architectures and object categories. In particular, we see significant improvements for small objects and on stricter AP^{75} metric.

Improved localization. In Table 3.3, we also notice that for VGG16, our method performs exceptionally well on the AP^{75} metric, improving the baseline Faster R-CNN by **8.9** AP^{75} points, which is much higher than the **3.5** point AP^{50} boost. We believe that using contextual features to select and integrate low-level finer details is they key reason for this improvement. Similarly for ResNet101 and InceptionResNetv2, we see **4.8** AP^{75} and **3.1** AP^{75} boost respectively.

Improvement for small objects. In Table 3.3, for VGG16, ResNet101 and InceptionResNetv2), we see **4.8**, **3** and **3.6** point boost respectively for small objects (AP^S) highlighting the effectiveness of features with TDM. Moreover, small objects are often on top of the list in Figure 3.5 (*e.g.*, sportsball +13 AP point, mouse +10 AP for VGG16). This is in line with other studies [110], which show that context is particularly helpful for some objects. Similar trends hold for the minival∗ split as well: **5.6**, **7.4** and **8.5** AP^S boost for VGG16, ResNet101 and InceptionResNetv2 respectively. Also refer to Table 3.4 for per-category results for all network architectures.

Qualitative Results. In Figure 3.6, we display qualitative results of the Top-down Modulation Network. Notice that the network can find small objects like remote (second row, last column; fourth row, first column) and sportsball (second row, third and fourth column). Also notice the detection results in the presence of heavy clutter.

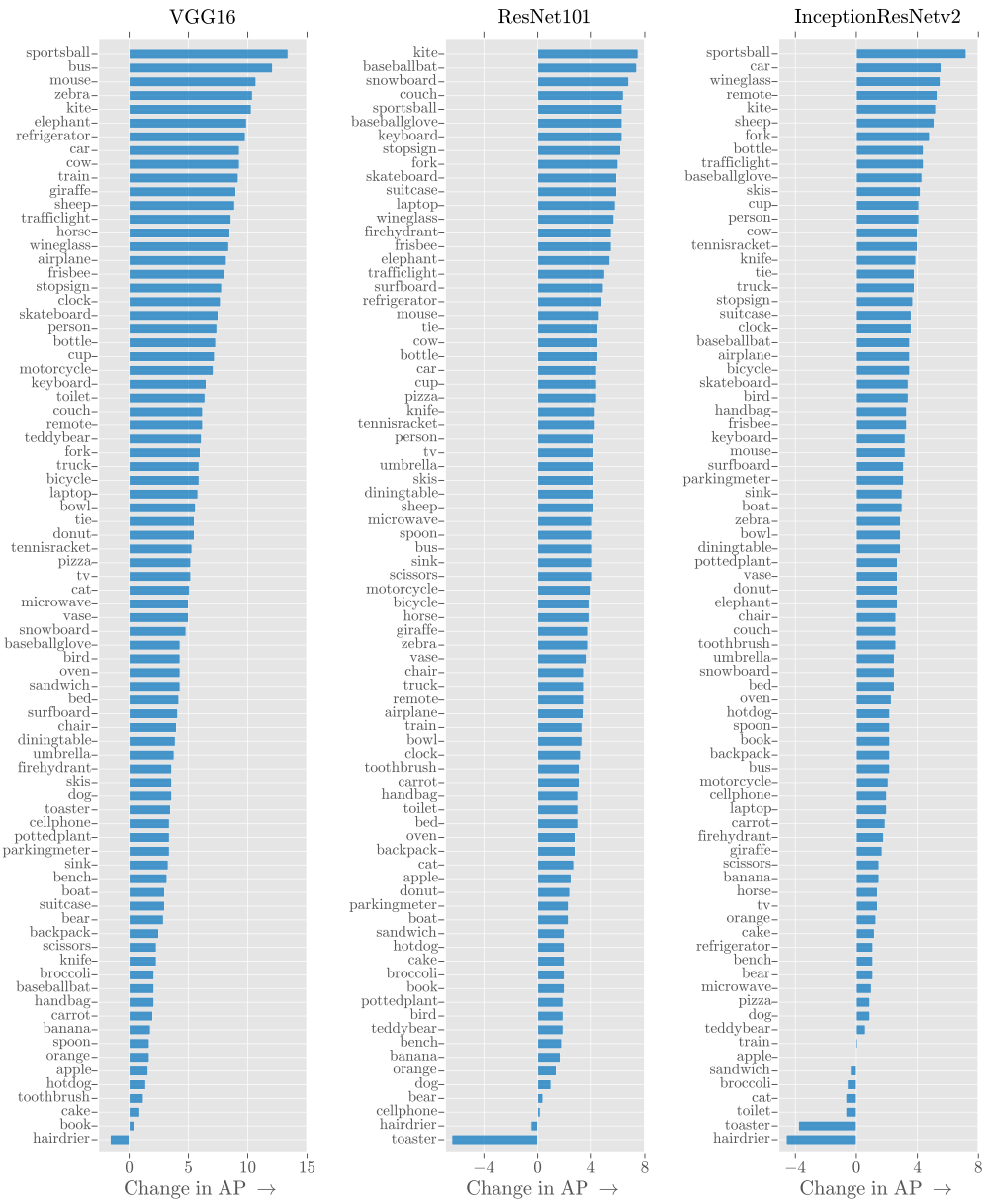


Figure 3.5 – Improvement in AP over Faster R-CNN baseline. Base Networks: (left) VGG16, (middle) ResNet101, and (right) InceptionResNetv2. Improved performance for almost all categories emphasize the effectiveness of Top-Down Modulation for object detection. (best viewed digitally)

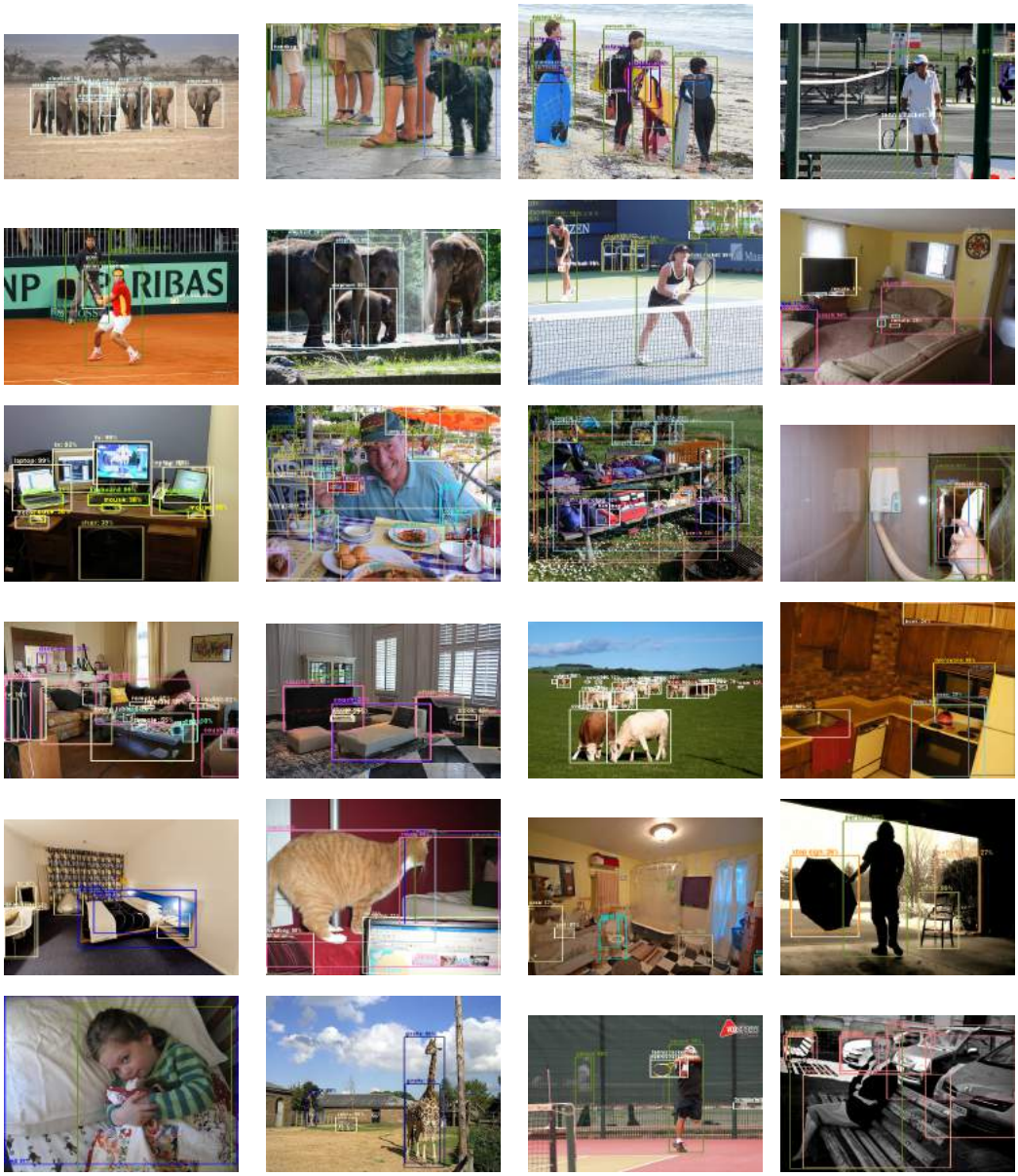


Figure 3.6 – **Qualitative results** of the proposed TDM network on randomly selected images from the minival∗ set (best viewed digitally).

Table 3.4 – COCO testdev detection AP results for all classes using different network architectures. Results are reported for baseline Faster R-CNN (**B**) and our method (**+TDM**). All methods use trainval∗ for training

Network →	VGG16		Resnet101		InceptionResNet	
	B	**+TDM**	**B**	**+TDM**	**B**	**+TDM**
AP	23.3	28.6	31.5	35.2	34.7	37.3
AP^{50}	44.7	48.1	52.8	55.3	55.5	57.8
AP^{75}	21.6	30.4	33.3	38.1	36.7	39.8
AP^{S}	9.4	14.2	13.6	16.6	13.5	17.1
AP^{M}	27.1	31.8	35.4	38.4	38.1	40.3
AP^{L}	32.0	36.9	44.5	47.9	52.0	52.1
person	35.7	43.1	44.3	48.5	44.7	48.8
bicycle	16.6	22.5	23.9	27.8	25.6	29.1
car	23.5	32.8	30.7	35.1	29.5	35.1
motorcycle	25.1	32.2	33.9	37.9	37.4	39.5
airplane	34.5	42.7	48.2	51.6	51.3	54.8
bus	42.1	54.2	58.1	62.2	61.6	63.8
train	38.7	47.9	55.2	58.5	62.2	62.3
truck	19.2	25.1	27.9	31.4	29.5	33.3
boat	11.9	14.9	17.4	19.7	18.9	21.9
traffic light	13.3	21.9	19.7	24.7	19.9	24.3
fire hydrant	43.0	46.6	53.3	58.8	57.8	59.6
stop sign	51.3	59.1	58.1	64.3	60.5	64.2
parking meter	26.0	29.4	33.4	35.7	33.7	36.8
bench	11.0	14.2	17.3	19.1	20.1	21.2
bird	21.0	25.3	28.7	30.6	30.3	33.7
cat	40.0	45.1	55.3	58.0	63.0	62.3
dog	36.2	39.8	51.7	52.7	57.7	58.6
horse	36.9	45.4	49.9	53.8	54.5	55.9
sheep	31.5	40.4	40.1	44.3	42.9	48.0
cow	30.1	39.4	41.6	46.1	45.0	49.0
elephant	46.5	56.4	59.7	65.1	64.4	67.1
bear	49.5	52.4	66.3	66.7	71.5	72.6
zebra	44.6	55.0	56.7	60.5	59.0	61.9
giraffe	50.7	59.7	63.9	67.7	67.9	69.6
backpack	8.0	10.5	11.4	14.2	12.4	14.6
umbrella	20.8	24.6	26.3	30.5	30.0	32.5
handbag	04.3	06.4	07.1	10.1	08.2	11.5
tie	13.6	19.1	22.0	26.5	23.5	27.3
suitcase	16.1	19.1	22.5	28.4	30.1	33.7
frisbee	31.8	39.8	42.8	48.3	44.1	47.4
skis	9.5	13.1	15.0	19.2	17.0	21.2
snowboard	16.2	21.0	23.4	30.2	28.4	30.9
sports ball	20.8	34.2	31.5	37.8	29.7	36.9
kite	23.5	33.8	30.6	38.1	34.2	39.4
baseball bat	14.7	16.8	17.2	24.6	23.8	27.3
baseball glove	20.3	24.6	25.1	31.4	26.1	30.4
skateboard	27.0	34.5	36.2	42.1	39.3	42.7

Network →	VGG16		Resnet101		InceptionResNet	
	B	**+TDM**	**B**	**+TDM**	**B**	**+TDM**
surfboard	18.5	22.6	26.6	31.5	30.7	33.8
tennis racket	30.4	35.7	39.2	43.5	41.1	45.1
bottle	19.6	26.9	26.1	30.6	25.9	30.3
wine glass	20.6	29.0	26.9	32.6	28.2	33.7
cup	23.8	31.0	30.1	34.5	31.3	35.4
fork	09.7	15.7	19.3	25.3	23.1	27.9
knife	04.8	07.1	10.9	15.2	13.4	17.3
spoon	04.4	06.1	08.2	12.3	11.7	13.9
bowl	25.3	30.9	30.6	33.9	32.1	35.0
banana	12.6	14.4	17.7	19.4	20.0	21.5
apple	13.6	15.2	17.5	20.0	19.6	19.6
sandwich	18.6	22.9	27.5	29.5	33.3	32.9
orange	17.8	19.5	23.3	24.7	26.0	27.3
broccoli	16.7	18.8	22.2	24.2	26.0	25.4
carrot	07.8	09.8	12.5	15.6	14.9	16.8
hot dog	15.6	17.0	22.0	24.0	24.4	26.6
pizza	39.1	44.3	47.7	52.1	51.6	52.5
donut	31.1	36.6	40.3	42.7	44.1	46.8
cake	18.8	19.7	24.6	26.6	28.2	29.4
chair	14.1	18.1	19.0	22.5	21.9	24.5
couch	21.4	27.6	32.0	38.4	38.4	41.0
potted plant	13.5	16.9	18.9	20.8	19.9	22.6
bed	27.8	32.0	37.0	40.0	44.0	46.5
dining table	18.0	21.9	23.2	27.4	26.0	28.9
toilet	40.8	47.2	52.2	55.2	57.5	56.8
tv	37.7	42.9	46.3	50.5	51.4	52.8
laptop	38.2	44.0	49.4	55.2	56.0	58.0
mouse	30.9	41.6	41.4	46.0	43.1	46.3
remote	11.1	17.3	19.9	23.4	19.3	24.6
keyboard	30.4	36.9	38.7	45.0	45.4	48.6
cell phone	15.7	19.1	23.4	23.6	23.8	25.8
microwave	36.5	41.5	46.0	50.1	51.8	52.8
oven	20.6	24.9	29.0	31.8	33.8	36.1
toaster	08.5	12.0	19.8	13.4	20.8	17.0
sink	24.6	27.9	28.5	32.6	31.7	34.7
refrigerator	19.7	29.5	37.6	42.4	46.9	48.0
book	04.7	05.2	06.5	08.5	07.0	09.2
clock	34.3	42.0	42.1	45.3	42.0	45.6
vase	22.5	27.5	30.3	34.0	34.3	37.0
scissors	09.1	11.4	17.6	21.7	25.5	27.0
teddy bear	22.3	28.4	34.0	35.9	39.4	40.0
hair drier	01.7	00.1	01.3	00.8	06.2	01.6
toothbrush	04.1	05.3	10.1	13.2	12.3	14.9

Table 3.5 – Ablation analysis on the COCO benchmark using the Faster R-CNN detection framework. All methods are trained on trainval∗ and evaluated on minival∗ set (Section 3.4). Methods are grouped based on their base network, **best** results are highlighted in each group.

Method	Net	Features from:	AP	AP^{50}	AP^{75}	AP^{S}	AP^{M}	AP^{L}	AR^{1}	AR^{10}	AR^{100}	AR^{S}	AR^{M}	AR^{L}
Baseline	VGG16	$\mathbf{C}_5$	25.5	46.7	24.6	6.1	23.3	37.0	23.9	38.2	40.7	14.1	39.5	55.3
Skip-pool	VGG16	$\mathbf{C}_2, \mathbf{C}_3, \mathbf{C}_4, \mathbf{C}_5$	25.3	46.3	25.9	9.1	24.0	36.0	24.6	40.0	42.4	18.6	41.8	54.1
TDM [ours]	VGG16 + TDM	$\mathbf{T}_4^{\text{out}}$	26.2	45.7	27.2	9.4	25.1	34.8	25.0	40.7	43.0	18.7	43.1	54.7
TDM [ours]	VGG16 + TDM	$\mathbf{T}_3^{\text{out}}$	28.8	48.6	30.7	11.0	27.1	37.3	26.5	42.7	45.0	21.1	44.2	56.4
TDM [ours]	VGG16 + TDM	$\mathbf{T}_2^{\text{out}}$	**29.9**	**50.3**	31.6	11.4	**28.1**	38.6	**27.3**	**43.7**	**46.0**	22.8	44.7	**57.1**
TDM [ours]	VGG16 + TDM	$\mathbf{T}_1^{\text{out}}$	29.8	49.9	**31.7**	**11.7**	28.0	**39.3**	27.1	43.5	45.9	**23.9**	**45.4**	56.8
Baseline	ResNet101	$\mathbf{C}_5$	32.1	53.2	33.8	9.4	29.7	45.7	28.3	44.3	46.7	19.3	46.3	60.9
TDM [ours]	ResNet101 + TDM	$\mathbf{T}_3^{\text{out}}$	34.4	54.4	37.1	10.9	31.8	48.2	30.1	47.5	49.8	21.7	49.1	64.0
TDM [ours]	ResNet101 + TDM	$\mathbf{T}_2^{\text{out}}$	35.3	55.1	38.3	11.2	33.0	48.2	30.7	48.0	50.5	22.5	50.1	63.6
TDM [ours]	ResNet101 + TDM	$\mathbf{T}_1^{\text{out}}$	**35.7**	**56.0**	**38.5**	**16.8**	**39.2**	**49.0**	**30.9**	**48.5**	**50.9**	**28.1**	**55.6**	**68.5**
Baseline	IRNv2	$\mathbf{B}_{20}$	35.7	56.5	38.0	8.9	32.0	52.5	30.8	47.8	50.3	19.6	49.9	66.9
TDM [ours]	IRNv2 + TDM	$\mathbf{T}_{6a}^{\text{out}}$	37.3	57.9	39.5	11.4	33.3	53.3	**32.8**	49.1	51.5	22.7	50.6	67.5
TDM [ours]	IRNv2 + TDM	$\mathbf{T}_{5a}^{\text{out}}$	**38.1**	**58.6**	**40.7**	**17.4**	**41.1**	**54.7**	32.4	**50.1**	**52.6**	**28.9**	**57.2**	**72.3**

3.5 Design and Ablation Analysis

In this Section, we perform control and ablative experiments to study the importance of top-down and lateral modules in the proposed TDM network.

3.5.1 How low should the Top-Down Modulation go?

In Section 3.3.3, we discussed the principles that we follow to add top-down and lateral modules. We connected these modules till the lowest layer choosing design decisions that hardware constraints would permit. However, is that overkill? Is there an optimal layer, after which this modulation does not help or starts hurting? To answer these questions, we limit the layer till which the modulation process happens, and use those features for Faster R-CNN.

Recall that the TDN network is built progressively, *i.e.*, we add one pair of lateral and top-down module at a time. So for this control study, we simply let each subsequent pair train for the entire learning schedule, treating the $\mathbf{T}^{\text{out}}$ as the final output. We report the results on minival∗ set in Table 3.5. As we can see, adding more top-down modulation helps in general. However, for VGG16, we see that the performance saturates at $\mathbf{T}_2^{\text{out}}$, and adding modules till $\mathbf{T}_1^{\text{out}}$ do not seem to help much. Deciding the endpoint criteria for top-down modulation is an interesting future direction.

3.5.2 No lateral modules

To analyze the importance of lateral modules, and to control for the extra parameters added by the TDM network (Table 3.2), we train additional baselines with

Table 3.6 – Importance of lateral modules. (top block) $\sim \mathbf{T}_{i,j}$ is a modified top-down module to account for more parameters in the TDM network. (bottom block) $(\mathbf{T}_{i,j}, \mathbf{L}_j)$ is the proposed top-down modulation network. All methods use the VGG16 network

	with $\mathbf{L}$	AP	AP^{50}	AP^{75}	AP^S	AP^M	AP^L	AR^1	AR^{10}	AR^{100}	AR^S	AR^M	AR^L
$\sim \mathbf{T}_{5,4}$		24.8	44.3	24.9	6.2	23.3	34.8	23.7	38.3	40.5	14.3	40.1	53.7
$\sim \mathbf{T}_{4,3}$		25.1	43.8	25.8	6.8	24.0	34.5	24.3	38.9	41.0	15.2	40.5	53.9
$\sim \mathbf{T}_{3,2}$		26.5	46.1	27.4	8.0	25.2	35.8	24.6	39.9	42.2	16.4	41.6	54.9
$\sim \mathbf{T}_{2,1}$		21.4	38.3	21.6	5.6	20.3	28.5	22.0	35.9	37.7	13.4	37.2	49.0
$(\mathbf{T}_{5,4}, \mathbf{L}_4)$	✓	26.2	45.7	27.2	9.4	25.1	34.8	25.0	40.7	43.0	18.7	43.1	54.7
$(\mathbf{T}_{4,3}, \mathbf{L}_3)$	✓	28.8	48.6	30.7	11.0	27.1	37.3	26.5	42.7	45.0	21.1	44.2	56.4
$(\mathbf{T}_{3,2}, \mathbf{L}_2)$	✓	**29.9**	50.3	31.6	11.4	28.1	38.6	27.3	43.7	46.0	22.8	44.7	57.1
$(\mathbf{T}_{2,1}, \mathbf{L}_1)$	✓	29.6	49.8	31.1	11.8	27.2	38.8	27.1	43.2	45.5	22.3	44.3	56.5

variants of VGG16 + TDM network. In particular, we remove the lateral modules and use convolutional and upsampling operations from the top-down modules $\mathbf{T}$ to train 'deeper' variants of VGG16 as baseline. To control for the extra parameters from lateral modules, we also increase the parameters in the convolutional layers. Note that for this baseline, we follow training methodology and design decisions used for training TDM networks.

As shown in Table 3.6, even though using more depth increases the performance slightly, the performance boost due to lateral modules is much higher. This highlights the importance of dividing the capacity of TDM network amongst lateral and top-down modules.

3.5.3 No top-down modules

Next we want to study the importance of the top-down path introduced by our TDM network. We believe that this path is responsible for transmitting contextual features and for selection of relevant finer details. Removing the top-down path exposes the 'skip'-connections from bottom-up features, which can be used for object detection. We follow the strategy from [16], where they ROI-pool features from different layers, L2-normalize and concatenate these features and finally scale them back to the original `conv5_3` magnitude and dimension.

We tried many variants of the Skip-pooling baseline, and report the best results in Table 3.5 (Skip-pool). We see that performance for small objects (AP^S) increases slightly, but overall the AP does not change much. This highlights the importance of using high-level contextual features in the top-down path for the selection of low-level features.

Table 3.7 – Impact of Pre-training. All methods use Resnet101 as the base network

	pre-trained on:	AP	AP^{50}	AP^{75}	AP^S	AP^M	AP^L	AR^1	AR^{10}	AR^{100}	AR^S	AR^M	AR^L
$\mathbf{T}_{4,3}$	Imagenet	34.0	54.9	36.2	10.3	31.9	47.2	29.4	46.1	48.8	21.3	48.4	62.4
$\mathbf{T}_{4,3}$	+COCO	34.4	54.4	37.1	10.9	31.8	48.2	30.1	47.5	49.8	21.7	49.1	64.0
$\mathbf{T}_{3,2}$	Imagenet	34.1	55.8	36.3	10.4	32.3	47.2	29.4	45.9	48.6	21.2	48.1	62.2
$\mathbf{T}_{3,2}$	+COCO	35.3	55.1	38.3	11.2	33.0	48.2	30.7	48.0	50.5	22.5	50.1	63.6

3.5.4 Impact of Pre-training

Finally, we study the impact of using a model pre-trained on the detection task to initialize our base networks and ResNet101/InceptionResNetv2's RPN and RCN networks *vs.* using only an image classification [58] pre-trained model. In Table 3.7, we see that initialization does not impact the performance by a huge margin. However, pre-training on the detection task is consistently better than using the classification initialization.

Conclusion. This Chapter introduces the Top-Down Modulation (TDM) network, which leverages top-down contextual features and lateral connections to bottom-up features for object detection. The TDM network uses top-down context to select low-level finer details, and learns to integrate them together. Through extensive experiments on the challenging COCO dataset, we demonstrate the effectiveness and importance of features from the TDM network. We show empirically that the proposed representation benefits all objects, big and small, and is helpful for accurate localization. Even though we focused on the object detection, we believe these top-down modulated features will be helpful in a wide variety of computer vision tasks.

Chapter 4

Online Hard Example Mining

> If you would only recognize that life is hard, things would be so much easier for you.
>
> Louis Brandeis

The Object detection systems, including the ones discussed in the last two Chapters, are trained through a *reduction* that converts object detection into an image classification problem. This reduction introduces a new challenge that is not found in natural image classification tasks: the training set is distinguished by a large imbalance between the number of annotated objects and the number of *background* examples (image regions not belonging to any object class of interest). In the case of sliding-window object detectors, such as the deformable parts model (DPM) [79], this imbalance may be as extreme as 100,000 background examples to every one object. The recent trend towards object-proposal-based detectors [96, 288] mitigates this issue to an extent, but the imbalance ratio may still be high (*e.g.*, 70:1). This challenge opens space for learning techniques that cope with imbalance and yield faster training, higher accuracy, or both.

Unsurprisingly, this is not a new challenge and a standard solution, originally called *bootstrapping* (and now often called *hard negative mining*), has existed for at least 20 years. Bootstrapping was introduced in the work of Sung and Poggio [268] in the mid-1990's (if not earlier) for training face detection models. Their key idea was to gradually grow, or *bootstrap*, the set of background examples by selecting those examples for which the detector triggers a false alarm. This strategy leads to an iterative training algorithm that alternates between updating the detection model given the current set of examples, and then using the updated model to find new false positives to add to the bootstrapped training set. The process typically commences with a training set consisting of all object examples and a small, random

set of background examples.

Bootstrapping has seen widespread use in the intervening decades of object detection research. Dalal and Triggs [54] used it when training SVMs for pedestrian detection. Felzenszwalb *et al.* [79] later proved that a form of bootstrapping for SVMs converges to the global optimal solution defined on the entire dataset. Their algorithm is often referred to as *hard negative mining* and is frequently used when training SVMs for object detection [96, 120, 288]. Bootstrapping was also successfully applied to a variety of other learning models, including shallow neural networks [230] and boosted decision trees [62]. Even modern detection methods based on deep convolutional neural networks (ConvNets) [155, 166], such as R-CNN [96] and sppnet [120], still employ SVMs trained with hard negative mining.

It may seem odd then that the current state-of-the-art object detectors, embodied by Fast R-CNN [95] and its descendants [223], do not use bootstrapping. The underlying reason is a technical difficulty brought on by the shift towards purely online learning algorithms, particularly in the context of deep ConvNets trained with stochastic gradient descent (SGD) on millions of examples. Bootstrapping, and its variants in the literature, rely on the aforementioned alternation template: (a) for some period of time a *fixed* model is used to find new examples to add to the active training set; (b) then, for some period of time the model is trained on the *fixed* active training set. Training deep ConvNet detectors with SGD typically requires hundreds of thousands of SGD steps and freezing the model for even a few iterations at a time would dramatically slow progress. What is needed, instead, is a purely online form of hard example selection.

In this Chapter, we propose a novel bootstrapping technique called *online hard example mining*[1] (OHEM) for training state-of-the-art detection models based on deep ConvNets. The algorithm is a simple modification to SGD in which training examples are sampled according to a non-uniform, non-stationary distribution that depends on the current loss of each example under consideration. The method takes advantage of detection-specific problem structure in which each SGD mini-batch consists of only one or two images, but *thousands* of candidate examples. The candidate examples are subsampled according to a distribution that favors diverse, high loss instances. Gradient computation (backpropagation) is still efficient because it only uses a small subset of all candidates. We apply OHEM to the standard Fast R-CNN detection method and show three benefits compared to the baseline training algorithm:

[1] We use the term hard *example* mining, rather than hard *negative* mining, because our method is applied in a multi-class setting to all classes, not just a "negative" class.

- It removes the need for several heuristics and hyperparameters commonly used in region-based ConvNets.
- It yields a consistent and significant boosts in mean average precision.
- Its effectiveness increases as the training set becomes larger and more difficult, as demonstrated by results on the MS COCO dataset.

Moreover, the gains from OHEM are complementary to recent improvements in object detection, such as multi-scale testing [120] and iterative bounding-box regression [93]. Combined with these tricks, OHEM gives state-of-the-art results of **78.9%** and **76.3%** mAP on PASCAL VOC 2007 and 2012, respectively.

4.1 Related Work

Object detection is one of the oldest and most fundamental problems in computer vision. The idea of dataset *bootstrapping* [230, 268], typically called *hard negative mining* in recent work [79], appears in the training of most successful object detectors [54, 62, 79, 93, 96, 120, 185, 230, 260]. Many of these approaches use SVMs as the detection scoring function, even after training a deep convolutional neural network (ConvNet) [155, 166] for feature extraction. One notable exception is the Fast R-CNN detector [95] and its descendants, such as Faster R-CNN [223]. Since these models do not use SVMs, and are trained purely online with SGD, existing hard example mining techniques cannot be immediately applied. This work addresses that problem by introducing an online hard example mining algorithm that improves optimization and detection accuracy. We briefly review hard example mining, modern ConvNet-based object detection, and relationships to concurrent works using hard example selection for training deep networks.

Hard example mining. There are two hard example mining algorithms in common use. The first is used when optimizing SVMs. In this case, the training algorithm maintains a working set of examples and alternates between training an SVM to convergence on the working set, and updating the working set by removing some examples and adding others according to a specific rule [79]. The rule removes examples that are "easy" in the sense that they are correctly classified beyond the current model's margin. Conversely, the rule adds new examples that are hard in the sense that they violate the current model's margin. Applying this rule leads to the global SVM solution. Importantly, the working set is usually a small subset of the entire training set.

The second method is used for non-SVMs and has been applied to a variety of

models including shallow neural networks [230] and boosted decision trees [62]. This algorithm usually starts with a dataset of positive examples and a random set of negative examples. The machine learning model is then trained to convergence on that dataset and subsequently applied to a larger dataset to harvest false positives. The false positives are then added to the training set and then the model is trained again. This process is usually iterated only once and does not have any convergence proofs.

ConvNet-based object detection. In the last three years significant gains have been made in object detection. These improvements were made possible by the successful application of deep ConvNets [155] to ImageNet classification [58]. The R-CNN [96] and OverFeat [243] detectors lead this wave with impressive results on PASCAL VOC [72] and ImageNet detection. OverFeat is based on the sliding-window detection method, which is perhaps the most intuitive and oldest search method for detection. R-CNN, in contrast, uses region proposals [4, 7, 32, 43, 69, 153, 288, 329], a method that was made popular by the selective search algorithm [288]. Since R-CNN, there has been rapid progress in region-based ConvNets, including sppnet [120], MR-CNN [93], and Fast R-CNN [95], which our work builds on.

Hard example selection in deep learning. There is recent work [181, 257, 297] concurrent to our own that selects hard examples for training deep networks. Similar to our approach, all these methods base their selection on the current loss for each datapoint. [257] independently selects hard positive and negative example from a larger set of random examples based on their loss to learn image descriptors. Given a positive pair of patches, [297] finds hard negative patches from a large set using triplet loss. Akin to our approach, [181] investigates online selection of hard examples for mini-batch SGD methods. Their selection is also based on loss, but the focus is on ConvNets for image classification. Complementary to [181], we focus on online hard example selection strategy for region-based object detectors.

4.2 Preliminaries: Fast R-CNN

We first summarize the Fast R-CNN [95] (FRCN) framework. FRCN takes as input an image and a set of object proposal regions of interest (RoIs). The FRCN network itself can be divided into two sequential parts: a convolutional (*conv*) network with several convolution and max-pooling layers (Figure 4.1, "Convolutional Network"); and an RoI network with an RoI-pooling layer, several fully-connected (*fc*) layers and two loss layers (Figure 4.1, "RoI Network").

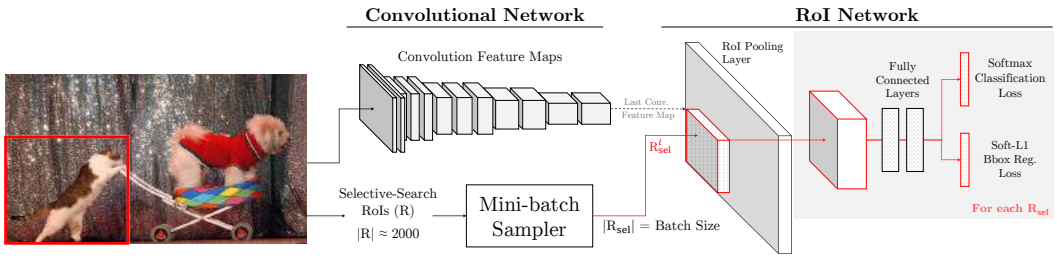


Figure 4.1 – Architecture of the Fast R-CNN approach (see Section 4.2 for details).

During inference, the conv network is applied to the given image to produce a conv feature map, size of which depends on the input image dimensions. Then, for each object proposal, the RoI-pooling layer projects the proposal onto the conv feature map and extracts a fixed-length feature vector. Each feature vector is fed into the fc layers, which finally give two outputs: (1) a softmax probability distribution over the object classes and background; and (2) regressed coordinates for bounding-box relocalization.

There are several reasons for choosing FRCN as our base object detector, apart from it being a fast end-to-end system. Firstly, the basic two network setup (conv and RoI) is also used by other recent detectors like sppnet and MR-CNN; therefore, our proposed algorithm is more broadly applicable. Secondly, though the basic setup is similar, FRCN also allows for training the entire conv network, as opposed to both sppnet and MR-CNN which keep the conv network fixed. And finally, both sppnet and MR-CNN require features from the RoI network to be cached for training a separate SVM classifier (using hard negative mining). FRCN uses the RoI network itself to train the desired classifiers. In fact, [95] shows that in the unified system using the SVM classifiers at later stages was unnecessary.

4.2.1 Training

Like most deep networks, FRCN is trained using stochastic gradient descent (SGD). The loss per example RoI is the sum of a classification log loss that encourages predicting the correct object (or background) label and a localization loss that encourages predicting an accurate bounding box (see [95] for details).

To share conv network computation between RoIs, SGD mini-batches are created hierarchically. For each mini-batch, N images are first sampled from the dataset, and then B/N RoIs are sampled from each image. Setting $N = 2$ and $B = 128$ works well in practice [95]. The RoI sampling procedure uses several heuristics, which we describe briefly below. One contribution of this Chapter is to

eliminate some of these heuristics and their hyperparameters.

Foreground RoIs. For an example RoI to be labeled as foreground (`fg`), its intersection over union (IoU) overlap with a ground-truth bounding box should be at least 0.5. This is a fairly standard design choice, in part inspired by the evaluation protocol of the PASCAL VOC object detection benchmark. The same criterion is used in the SVM hard mining procedures of R-CNN, sppnet, and MR-CNN. We use the same setting.

Background RoIs. A region is labeled background (`bg`) if its maximum IoU with ground truth is in the interval $[\texttt{bg\_lo}, 0.5)$. A lower threshold of $\texttt{bg\_lo} = 0.1$ is used by both FRCN and sppnet, and is hypothesized in [95] to crudely approximate hard negative mining; the assumption is that regions with some overlap with the ground truth are more likely to be the confusing or hard ones. We show in Section 4.4.4 that although this heuristic helps convergence and detection accuracy, it is suboptimal because it ignores some infrequent, but important, difficult background regions. Our method removes the `bg_lo` threshold.

Balancing `fg-bg` RoIs: To handle the data imbalance described in the beginning of this Chapter, [95] designed heuristics to rebalance the foreground-to-background ratio in each mini-batch to a target of $1:3$ by undersampling the background patches at random, thus ensuring that 25% of a mini-batch is `fg` RoIs. We found that this is an important design decision for the training FRCN. Removing this ratio (*i.e.* randomly sampling RoIs), or increasing it, decreases accuracy by ∼3 points mAP. With our proposed method, we can remove this ratio hyperparameter with no ill effect.

4.3 Our Approach

We propose a simple yet effective online hard example mining algorithm for training Fast R-CNN (or any Fast R-CNN style object detector). We argue that the current way of creating mini-batches for SGD (Section 4.2.1) is inefficient and suboptimal, and we demonstrate that our approach leads to better training (lower training loss) and higher testing performance (mAP).

4.3.1 Online Hard Example Mining

Recall the alternating steps that define a hard example mining algorithm: (a) for some period of time a *fixed* model is used to find new examples to add to the

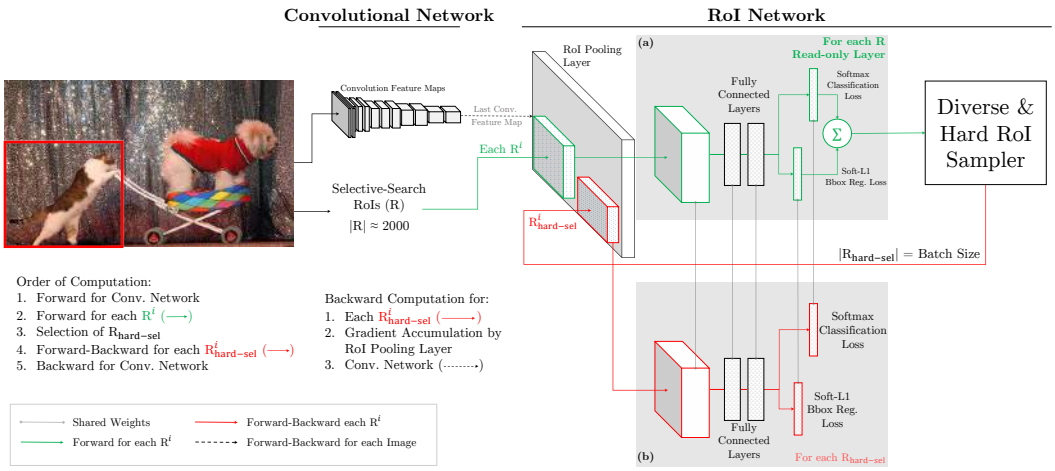


Figure 4.2 – Architecture of the proposed training algorithm. Given an image, and selective search RoIs, the conv network computes a conv feature map. In (a), the *readonly* RoI network runs a forward pass on the feature map and all RoIs (shown in green arrows). Then the Hard RoI module uses these RoI losses to select B examples. In (b), these hard examples are used by the RoI network to compute forward and backward passes (shown in red arrows).

active training set; (b) then, for some period of time the model is trained on the *fixed* active training set. In the context of SVM-based object detectors, such as the SVMs trained in R-CNN or sppnet, step (a) inspects a variable number of images (often 10's or 100's) until the active training set reaches a threshold size, and then in step (b) the SVM is trained to convergence on the active training set. This process repeats until the active training set contains all support vectors. Applying an analogous strategy to FRCN ConvNet training slows learning because no model updates are made while selecting examples from the 10's or 100's of images.

Our main observation is that these alternating steps can be combined with how FRCN is trained using online SGD. The key is that although each SGD iteration samples only a small number of images, each image contains *thousands* of example RoIs from which we can select the hard examples rather than a heuristically sampled subset. This strategy fits the alternation template to SGD by "freezing" the model for only one mini-batch. Thus the model is updated exactly as frequently as with the baseline SGD approach and therefore learning is not delayed.

More specifically, the online hard example mining algorithm (OHEM) proceeds as follows. For an input image at SGD iteration t, we first compute a conv feature map using the conv network. Then the RoI network uses this feature map and the **all** the input RoIs (R), instead of a sampled mini-batch [95], to do a forward pass. Recall that this step only involves RoI pooling, a few fc layers, and loss computation for each

RoI. The loss represents how well the current network performs on each RoI. Hard examples are selected by sorting the input RoIs by loss and taking the B/N examples for which the current network performs worst. Most of the forward computation is shared between RoIs via the conv feature map, so the extra computation needed to forward all RoIs is relatively small. Moreover, because only a small number of RoIs are selected for updating the model, the backward pass is no more expensive than before.

However, there is a small caveat: co-located RoIs with high overlap are likely to have correlated losses. Moreover, these overlapping RoIs can project onto the same region in the conv feature map, because of resolution disparity, thus leading to loss double counting. To deal with these redundant and correlated regions, we use standard non-maximum suppression (NMS) to perform deduplication (the implementation from [95]). Given a list of RoIs and their losses, NMS works by iteratively selecting the RoI with the highest loss, and then removing all lower loss RoIs that have high overlap with the selected region. We use a relaxed IoU threshold of 0.7 to suppress only highly overlapping RoIs.

We note that the procedure described above does not need a `fg-bg` ratio for data balancing. If any class were neglected, its loss would increase until it has a high probability of being sampled. There can be images where the `fg` RoIs are easy (*e.g.* canonical view of a car), so the network is free to use only `bg` regions in a mini-batch; and vice-versa when `bg` is trivial (*e.g.* sky, grass *etc.*), the mini-batch can be entirely `fg` regions.

4.3.2 Implementation details

There are many ways to implement OHEM in the FRCN detector, each with different trade-offs. An obvious way is to modify the loss layers to do the hard example selection. The loss layer can compute loss for all RoIs, sort them based on this loss to select *hard* RoIs, and finally set the loss of all *non-hard* RoIs to 0. Though straightforward, this implementation is inefficient as the RoI network still allocates memory and performs backward pass for **all** RoIs, even though most RoIs have 0 loss and hence no gradient updates (a limitation of current deep learning toolboxes).

To overcome this, we propose the architecture presented in Figure 4.2. Our implementation maintains two copies of the RoI network, one of which is *readonly*. This implies that the readonly RoI network (Figure 4.2(a)) allocates memory only for forward pass of all RoIs as opposed to the standard RoI network, which allocates memory for both forward and backward passes. For an SGD iteration, given the

Table 4.1 – Impact of hyperparameters on FRCN training

	Experiment	Model	N	LR	B	bg_lo	07 mAP
1	Fast R-CNN [95]	VGGM	2	0.001	128	0.1	59.6
2		VGG16					67.2
3	Removing hard mining heuristic (Section 4.4.2)	VGGM	2	0.001	128	**0**	57.2
4		VGG16					67.5
5	Fewer images per batch (Section 4.4.3)	VGG16	**1**	0.001	128	0.1	66.3
6						0	66.3
7	Bigger batch, High LR (Section 4.4.4)	VGGM	1	**0.004**	**2048**	0	57.7
8			2				60.4
9		VGG16	1	**0.003**	**2048**	0	67.5
10			2				68.7
11	Our Approach	VGG16	**1**	0.001	128	0	69.7
12		VGGM	2	0.001	128	0	62.0
13		VGG16					69.9

conv feature map, the readonly RoI network performs a forward pass and computes loss for **all** input RoIs (R) (Figure 4.2, green arrows). Then the hard RoI sampling module uses the procedure described in Section 4.3.1 to select hard examples ($R_{\text{hard-sel}}$), which are input to the regular RoI network (Figure 4.2(b), red arrows)). This network computes forward and backward passes only for $R_{\text{hard-sel}}$, accumulates the gradients and passes them to the conv network. In practice, we use all RoIs from all N images as R, therefore the effective batch size for the readonly RoI network is |R| and for the regular RoI network is the standard B from Section 4.2.1.

We implement both options described above using the Caffe [135] framework (see [95]). Our implementation uses gradient accumulation with N forward-backward passes of single image mini-batches. Following FRCN [95], we use $N = 2$ (which results in $|R| \approx 4000$) and $B = 128$. Under these settings, the proposed architecture (Figure 4.2) has similar memory footprint as the first option, but is $> 2\times$ faster. Unless specified otherwise, the architecture and settings described above will be used throughout this Chapter.

4.4 Design and Ablation Analysis

This Section compares FRCN training with online hard example mining (OHEM) to the baseline heuristic sampling approach. We also compare FRCN with OHEM to a less efficient approach that uses all available example RoIs in each mini-batch, not just the B hardest examples.

4.4.1 Experimental Setup

We conduct experiments with two standard ConvNet architectures: VGG_CNN_M_1024 (VGGM, for short) from [36], which is a wider version of alexnet [155], and VGG16 from [258]. All experiments in this Section are performed on the PASCAL VOC07 dataset. Training is done on the trainval set and testing on the test set. Unless specified otherwise, we will use the default settings from FRCN [95]. We train all methods with SGD for 80k mini-batch iterations, with an initial learning rate of 0.001 and we decay the learning rate by 0.1 every 30k iterations. The baseline numbers reported in Table 4.1 (row 1-2) were reproduced using our training schedule and are slightly higher than the ones reported in [95].

4.4.2 OHEM vs. Heuristic Sampling

Standard FRCN, reported in Table 4.1 (rows $1-2$), uses `bg_lo` $= 0.1$ as a heuristic for hard mining (Section 4.2.1). To test the importance of this heuristic, we ran FRCN with `bg_lo` $= 0$. Table 4.1 (rows $3-4$) shows that for VGGM, mAP drops by 2.4 points, whereas for VGG16 it remains roughly the same. Now compare this to training FRCN with OHEM (rows $11-13$). OHEM improves mAP by 2.4 points compared to FRCN with the `bg_lo` $= 0.1$ heuristic for VGGM, and 4.8 points without the heuristic. This result demonstrates the sub-optimality of these heuristics and the effectiveness of our hard mining approach.

4.4.3 Robust Gradient Estimates

One concern over using only $N = 2$ images per batch is that it may cause unstable gradients and slow convergence because RoIs from an image may be highly correlated [273]. FRCN [95] reports that this was not a practical issue for their training. But this detail might raise concerns over our training procedure because we use examples with high loss from the same image and as a result they may be more highly correlated. To address this concern, we experiment with $N = 1$ in order to increase correlation in an effort to break our method. As seen in Table 4.1 (rows $5-6, 11$), performance of the original FRCN drops by $\sim$1 point with $N = 1$, but when using our training procedure, mAP remains approximately the same. This shows that OHEM is robust in case one needs fewer images per batch in order to reduce GPU memory usage.

4.4.4 Why just hard examples, when you can use all?

Online hard example mining is based on the hypothesis that it is important to consider all RoIs in an image and then select hard examples for training. But what if

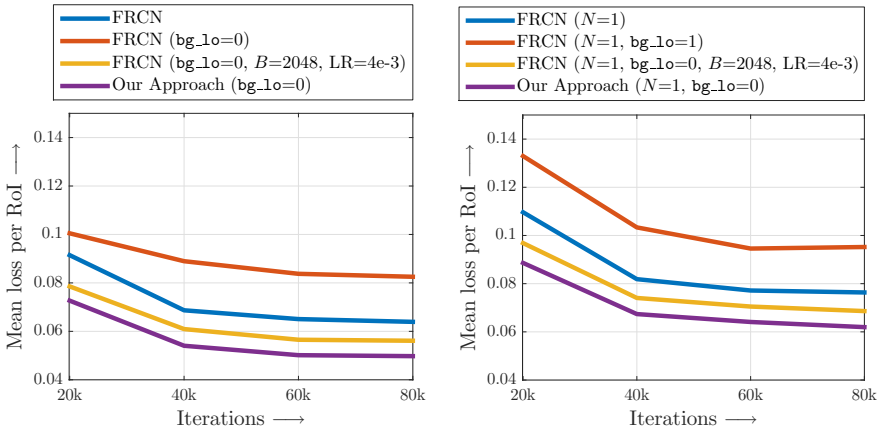


Figure 4.3 – Training loss is computed for various training procedures using VGG16 networks discussed in Section 4.4. We report mean loss per RoI. These results indicate that using hard mining for training leads to lower training loss than any of the other heuristics.

we train with all the RoIs, not just the hard ones? The easy examples will have low loss, and won't contribute much to the gradient; training will automatically focus on the hard examples. To compare this option, we ran standard FRCN training with a large mini-batch size of $B = 2048$, using `bg_lo` $= 0$, $N \in \{1, 2\}$ and with other hyperparameters fixed. Because this experiment uses a large mini-batch, it's important to tune the learning rate to adjust for this change. We found optimal results by increasing it to 0.003 for VGG16 and 0.004 for VGGM. The outcomes are reported in Table 4.1 (rows $7-10$). Using these settings, mAP of both VGG16 and VGGM increased by $\sim$1 point compared to $B = 128$, but the improvement from our approach is still > 1 points over using all RoIs. Moreover, because we compute gradients with a smaller mini-batch size training is faster.

4.4.5 Better Optimization

Finally, we analyze the training loss for the various FRCN training methods discussed above. It's important to measure training loss in a way that does not depend on the sampling procedure and thus results in a valid comparison between methods. To achieve this goal, we take model snapshots from each method every 20k steps of optimization and run them over the entire VOC07 trainval set to compute the average loss over *all* RoIs. This measures the training set loss in a way that does not depend on the example sampling scheme.

Figure 4.3 shows the average loss per RoI for VGG16 with the various hyperparameter settings discussed above and presented in Table 4.1. We see that `bg_lo` $= 0$ results in the highest training loss, while using the heuristic `bg_lo` $= 0.1$ results in a

Table 4.2 – Computational statistics of training FRCN [95] and FRCN with OHEM (using an Nvidia Titan X GPU)

	VGGM		VGG16		
	FRCN	Ours	FRCN	FRCN*	Ours*
time (sec/iter)	0.13	0.22	0.60	0.57	1.00
max. memory (G)	2.6	3.6	11.2	6.4	8.7

*: uses gradient accumulation over two forward/backward passes

much lower training loss. Increasing the mini-batch size to $B = 2048$ and increasing the learning rate lowers the training loss below the `bg_lo` $= 0.1$ heuristic. Our proposed online hard example mining method achieves the lowest training loss of all methods, validating our claims that OHEM leads to better training for FRCN.

4.4.6 Computational Cost

OHEM adds reasonable computational and memory overhead, as reported in Table 4.2. OHEM costs 0.09s per training iteration for VGGM network (0.43s for VGG16) and requires 1G more memory (2.3G for VGG16). Given that FRCN [95] is a fast detector to train, the increase in training time is acceptable for most scenarios.

4.5 Results

In this Section, we evaluate our method on VOC 2012 [72] as well as the more challenging MS COCO [175] dataset. We demonstrate consistent and significant improvement in FRCN performance when using the proposed OHEM approach. Per-class results are also presented on VOC 2007 for comparison with prior work.

Experimental Setup. We use VGG16 for all experiments. When training on VOC07 trainval, we use the SGD parameters as in Section 4.4 and when using extra data (07+12 and 07++12, see Table 4.3 and 4.4), we use 200k mini-batch iterations, with an initial learning rate of 0.001 and decay step size of 40k. When training on MS COCO [175], we use 240k mini-batch iterations, with an initial learning rate of 0.001 and decay step size of 160k, owing to a larger epoch size.

4.5.1 VOC 2007 and 2012 Results

Table 4.3 shows that on VOC07, OHEM improves the mAP of FRCN from 67.2% to 69.9% (and 70.0% to 74.6% with extra data). On VOC12, OHEM leads to an improvement of 4.1 points in mAP (from 65.7% to 69.8%). With extra data, we achieve an mAP of 71.9% as compared to 68.4% mAP of FRCN, an improvement

Table 4.3 – VOC 2007 test detection average precision (%). All methods use VGG16. Training set key: **07**: VOC07 trainval, **07+12**: union of **07** and VOC12 trainval. All methods use bounding-box regression. Legend: **M**: using multi-scale for training and testing, **B**: multi-stage bbox regression. FRCN⋆ refers to FRCN [95] with our training schedule

method	M	B	train set	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
FRCN [95]			07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
FRCN⋆			07	67.2	74.6	76.8	67.6	52.9	37.8	78.7	78.8	81.6	42.2	73.6	67.0	79.4	79.6	74.1	68.3	33.4	65.9	68.7	75.4	68.1
Ours			07	**69.9**	71.2	78.3	69.2	57.9	46.5	81.8	79.1	83.2	47.9	76.2	68.9	83.2	80.8	75.8	72.7	39.9	67.5	66.2	75.6	75.9
FRCN⋆	✓	✓	07	72.4	77.8	81.3	71.4	60.4	48.3	85.0	84.6	86.2	49.4	80.7	68.1	84.1	86.7	80.2	75.3	38.7	71.9	71.5	77.9	67.8
MR-CNN [93]	✓	✓	07	74.9	78.7	81.8	76.7	66.6	61.8	81.7	85.3	82.7	57.0	81.9	73.2	84.6	86.0	80.5	74.9	44.9	71.7	69.7	78.7	79.9
Ours	✓	✓	07	**75.1**	77.7	81.9	76.0	64.9	55.8	86.3	86.0	86.8	53.2	82.9	70.3	85.0	86.3	78.7	78.0	46.8	76.1	72.7	80.9	75.5
FRCN [95]			07+12	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
Ours			07+12	**74.6**	77.7	81.2	74.1	64.2	50.2	86.2	83.8	88.1	55.2	80.9	73.8	85.1	82.6	77.8	74.9	43.7	76.1	74.2	82.3	79.6
MR-CNN [93]	✓	✓	07+12	78.2	80.3	84.1	78.5	**70.8**	**68.5**	88.0	85.9	87.8	**60.3**	**85.2**	73.7	**87.2**	86.5	**85.0**	76.4	48.5	76.3	75.5	**85.0**	**81.0**
Ours	✓	✓	07+12	**78.9**	**80.6**	**85.7**	**79.8**	69.9	60.8	**88.3**	**87.9**	**89.6**	59.7	85.1	**76.5**	87.1	**87.3**	82.4	**78.8**	**53.7**	**80.5**	**78.7**	84.5	80.7

of 3.5 points. Interestingly the improvements are not uniform across categories. Bottle, chair, and tvmonitor show larger improvements that are consistent across the different PASCAL splits. Why these classes benefit the most is an interesting and open question.

4.5.2 MS COCO Results

To test the benefit of using OHEM on a larger and more challenging dataset, we conduct experiments on MS COCO [175] and report numbers from test-dev 2015 evaluation server (Table 4.5). On the standard COCO evaluation metric, FRCN [95] scores 19.7% AP, and OHEM improves it to 22.6% AP.[2] Using the VOC overlap metric of IoU ≥ 0.5, OHEM gives a 6.6 points boost in AP^{50}. It is also interesting to note that OHEM helps improve the AP of medium sized objects by 4.9 points on the strict COCO AP evaluation metric, which indicates that the proposed hard example mining approach is helpful when dealing with smaller sized objects. Note that FRCN with and without OHEM were trained on MS COCO train set.

4.6 Adding Bells and Whistles

We've demonstrated consistent gains in detection accuracy by applying OHEM to FRCN training. In this Section, we show that these improvements are orthogonal to recent bells and whistles that enhance object detection accuracy. OHEM with the following two additions yields state-of-the-art results on VOC and competitive results on MS COCO.

[2] COCO AP averages over classes, recall, and IoU levels. See http://mscoco.org/dataset/#detections-eval for details.

Table 4.4 – VOC 2012 test detection average precision (%). All methods use VGG16. Training set key: **12**: VOC12 trainval, **07++12**: union of VOC07 trainval, VOC07 test, and VOC12 trainval, +COCO: a model trained on COCO trainval and fine-tuned on **07++12**. Legend: **M**: using multi-scale for training and testing, **B**: iterative bbox regression

method	M	B	train set	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
FRCN [95]			12	65.7	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7
Ours[1]			12	**69.8**	81.5	78.9	69.6	52.3	46.5	77.4	72.1	88.2	48.8	73.8	58.3	86.9	79.7	81.4	75.0	43.0	69.5	64.8	78.5	68.9
MR-CNN [93]	✓	✓	12	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Ours[2]	✓	✓	12	**72.9**	85.8	82.3	74.1	55.8	55.1	79.5	77.7	90.4	52.1	75.5	58.4	88.6	82.4	83.1	78.3	47.0	77.2	65.1	79.3	70.4
FRCN [95]			07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Ours[3]			07++12	**71.9**	83.0	81.3	72.5	55.6	49.0	78.9	74.7	89.5	52.3	75.0	61.0	87.9	80.9	82.4	76.3	47.1	72.5	67.3	80.6	71.2
MR-CNN [93]	✓	✓	07++12	73.9	85.5	82.9	76.6	57.8	62.7	79.4	77.2	86.6	55.0	79.1	62.2	87.0	83.4	84.7	78.9	45.3	73.4	65.8	80.3	74.0
Ours[4]	✓	✓	07++12	**76.3**	86.3	85.0	77.0	60.9	59.3	81.9	81.1	91.9	55.8	80.6	63.0	**90.8**	85.1	85.3	80.7	54.9	78.3	70.8	82.8	74.9
Ours[5]	✓	✓	+COCO	**80.1**	**90.1**	**87.4**	**79.9**	**65.8**	**66.3**	**86.1**	**85.0**	**92.9**	**62.4**	**83.4**	**69.5**	90.6	**88.9**	**88.9**	**83.6**	**59.0**	**82.0**	**74.7**	**88.2**	**77.3**

[1]link1, [2]link2, [3]link3, [4]link4, [5]link5

Table 4.5 – MS COCO 2015 test−dev detection average precision (%). All methods use VGG16. Legend: **M**: using multi-scale for training and testing

AP@IoU	area	FRCN†	**Ours**	**Ours** [+M]	**Ours*** [+M]
[0.50 : 0.95]	all	19.7	22.6	24.4	25.5
0.50	all	35.9	42.5	44.4	45.9
0.75	all	19.9	22.2	24.8	26.1
[0.50 : 0.95]	small	3.5	5.0	7.1	7.4
[0.50 : 0.95]	med.	18.8	23.7	26.4	27.7
[0.50 : 0.95]	large	34.6	37.9	38.5	40.3

†from the leaderboard, *trained on trainval set

Multi-scale (M). We adopt the multi-scale strategy from sppnet [120] (and used by both FRCN [95] and MR-CNN [93]). Scale is defined as the size of the shortest side (s) of an image. During training, one scale is chosen at random, whereas at test time inference is run on all scales. For VGG16 networks, we use $s \in \{480, 576, 688, 864, 900\}$ for training, and $s \in \{480, 576, 688, 864, 1000\}$ during testing, with the max dimension capped at 1000. The scales and caps were chosen because of GPU memory constraints.

Iterative bounding-box regression (B). We adopt the iterative localization and bounding-box (bbox) voting scheme from [93]. The network evaluates each proposal RoI to get scores and relocalized boxes R_1. High-scoring R_1 boxes are the rescored and relocalized, yielding boxes R_2. Union of R_1 and R_2 is used as the final set R_F for post-processing, where $\mathrm{R}_\mathrm{F}^\mathrm{NMS}$ is obtained using NMS on R_F with an IoU threshold of 0.3 and weighted voting is performed on each box r_i in $\mathrm{R}_\mathrm{F}^\mathrm{NMS}$ using boxes in R_F with an IoU of $\geq$0.5 with r_i (see [93] for details).

Table 4.6 – Impact of multi-scale and iterative bbox reg.

Multi-scale (**M**)		Iterative bbox reg. (**B**)	VOC07 mAP	
Train	Test		FRCN	**Ours**
			67.2	69.9
	✓		68.4	71.1
		✓	70.8	72.7
	✓	✓	71.9	74.1
✓			67.7	70.7
✓	✓		68.6	71.9
✓		✓	71.2	72.9
✓	✓	✓	72.4	**75.1**

4.6.1 VOC 2007 and 2012 Results

We report the results on VOC benchmarks in Table 4.3 and 4.4. On VOC07, FRCN with the above mentioned additions achieves 72.4% mAP and OHEM improves it to **75.1%**, which is currently the highest reported score under this setting (07 data). When using extra data (07+12), OHEM achieves **78.9%** mAP, surpassing the current state-of-the-art MR-CNN (78.2% mAP). We note that MR-CNN uses selective search and edge boxes during training, whereas we only use selective search boxes. Our multi-scale implementation is also different, using fewer scales than MR-CNN. On VOC12 (Table 4.4), we consistently perform better than MR-CNN. When using extra data, we achieve state-of-the-art mAP of **76.3%** (*vs.* 73.9% mAP of MR-CNN).

Ablation Analysis. We now study in detail the impact of these two additions and whether OHEM is complementary to them, and report the analysis in Table 4.6. Baseline FRCN mAP improves from 67.2% to 68.6% when using multi-scale during both training and testing (we refer to this as **M**). However, note that there is only a marginal benefit of using it at training time. Iterative bbox regression (**B**) further improves the FRCN mAP to 72.4%. But more importantly, using OHEM improves it to **75.1%** mAP, which is state-of-the-art for methods trained on VOC07 data (see Table 4.3). In fact, using OHEM consistently results in higher mAP for all variants of these two additions (see Table 4.6).

4.6.2 MS COCO Results

MS COCO [175] test-dev 2015 evaluation server results are reported in Table 4.5. Using multi-scale improves the performance of our method to 24.4% AP on the standard COCO metric and to 44.4% AP^{50} on the VOC metric. This again shows

the complementary nature of using multi-scale and OHEM. Finally, we train our method using the entire MS COCO trainval set, which further improves performance to **25.5%** AP (and 45.9% AP^{50}). In the 2015 MS COCO Detection Challenge, a variant of this approach finished 4^{th} place overall.

Conclusion We presented an online hard example mining (OHEM) algorithm, a simple and effective method to train region-based ConvNet detectors. OHEM eliminates several heuristics and hyperparameters in common use by automatically selecting hard examples, thus simplifying training. We conducted extensive experimental analysis to demonstrate the effectiveness of the proposed algorithm, which leads to better training convergence and consistent improvements in detection accuracy on standard benchmarks. We also reported state-of-the-art results on PASCAL VOC 2007 and 2012 when using OHEM with other orthogonal additions. Though we used Fast R-CNN throughout this Chapter, OHEM can be used for training any region-based ConvNet detector.

Our experimental analysis was based on the overall detection accuracy, however it will be an interesting future direction to study the impact of various training methodologies on individual category performance.

Chapter 5

Geometric-structure from Multi-modal Data

You can't criticize geometry.
It's never wrong.

Paul Rand

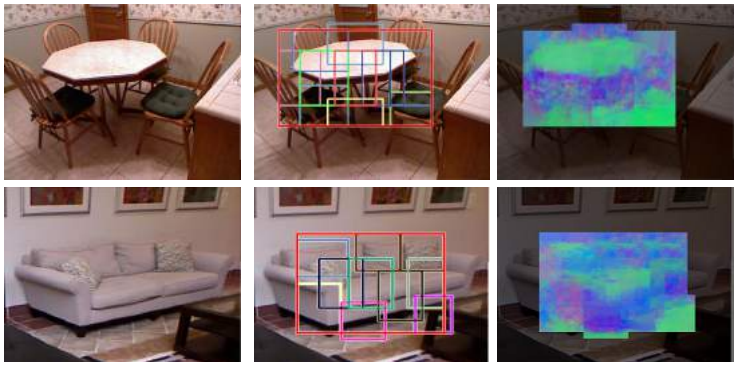


Figure 5.1 – Examples of object detection and surface normal prediction using the proposed G-DPM model. Our G-DPM not only improves the state of the art performance in object detection but it also predicts the surface normals with the detection. Legend for normals: blue: X; green: Y; red: Z.

The standard output of an object recognition systems discussed so far is either a box around the localized object (detection) or per-pixel labels (segmentation). Though critical building blocks for perception, these outputs offer a rather shallow understanding of the recognized object. For example, they have no geometric understanding that augmented reality and robotics applications might need. In this Chapter, we present a system to infer 3D properties of objects from 2D images.

We propose a geometry-driven deformable part-based model (G-DPM) that

can be learned from a set of labeled RGBD images. In a G-DPM, object parts are defined based on their physical properties (i.e., their geometry) rather than just their appearance. Our key hypothesis is that while the arrangement of parts might vary across the instances of object categories, the constituent parts will still have consistent underlying 3D geometry. For example, every sofa has a L-shaped part that is the intersection of a vertical surface and a horizontal surface for sitting. Therefore, the underlying 3D geometry can provide weak supervision to define and initialize the parts. While the learning objective in case of G-DPM is still non-convex (similar to [79]), we show how the depth data can be used as weak supervision to impose geometric constraints and guide latent updates at each step. Empirically this leads to faster convergence, and a better model in terms of detection performance. But more importantly, because our parts have a 3D geometrical representation they can be used to jointly detect objects and infer 3D properties from a single 2D image. Figure 5.1 shows two examples of objects detected by our G-DPM model and the predicted surface normal geometry by the G-DPM. Notice how our approach predicts nicely aligned flat horizontal surface of the table within the bounding box and how the approach predicts the horizontal and vertical surfaces of the couch.

Contributions. Our key contributions include: (1) We propose to marry deformable part-based model with the geometric representation of objects by defining parts based on consistent underlying 3D geometry. (2) We demonstrate how the geometric representation can help us leverage depth data during training and constrain the latent model learning problem. The underlying 3D geometry during training helps us guide the latent steps in the right direction. (3) Most importantly, a joint geometric and appearance based representation not only allows us to achieve state-of-the results on object detection but also allows us to tackle the grand challenge of understanding 3D objects from 2D images.

5.1 Related Work

The idea of using geometric and physical representation for objects and their categories has a rich history in computer vision [27, 182, 187]. While these approaches resulted in some impressive demos such as ACRONYM [28], these systems failed to generalize. That led us to the modern era in computer vision where instead of representing objects in 3D, the focus changed to representing objects using low-level image features such as HOG [54] and using machine learning to learn an appearance model of the object. The most successful approaches in this line of work are the deformable part-based models [79] that extend the rigid template from [54] to a latent part-based model that is trained discriminatively. While there has been a lot

of progress made over the last decade, the performance of these appearance based approaches seems to have been stagnated.

Therefore, recent research has now focused on developing richer representations for objects and effective ways of learning these representations. Most of the recent work on improving deformable part models can be broadly divided in two main categories:

(a) Better 2D Representations and Learning: The first and the most common way is to design better representations using 2D image features. In this area, researchers have looked into using strongly-supervised models for parts [10, 26, 70, 313], using key point annotations to search for parts [25] or discovering mid-level parts in a completely unsupervised manner [260]. Other directions include using sharing to increase data-size across categories [173] or finding visual subcategories based on appearances before learning a part-based model [41, 60].

(b) Using 3D Geometry: The second direction that has been explored is to bring back the flavor of the past and develop rich models by representing 3D geometry explicitly. One of the most common ways to encode viewpoint information is to train a mixture of templates for different viewpoints [102]. An alternative approach is to explicitly consider the 3D nature of the problem and model objects as a collection of local parts that are connected across views [99, 237, 263, 309]. Another way to account for 3D representation is to explicitly model the 3D object in terms of planes [44, 82, 237, 308] or parts [209], and use a rigid template [121], spring model [82] or a CRF [44].

Our approach lies at the intersection of two these directions. Unlike other approaches which incorporate geometry in DPM via CAD models [209] or manually-labeled 3D cuboids [82, 121], our approach uses noisy depth data for training (similar to [85]). This allows us to access more and diverse data (hundreds of images compared to 40 or so CAD models). The scale at which we build 3D priors and do geometric reasoning during latent learning allows us to obtain improvements of as much as 11% in some categories (previous approaches performed at-par or below DPM). We would also like to point out that even though our approach uses depth information during training, it is used as a weak supervisory signal (and not as an extra input feature) to guide the training in the right direction. The discriminative model is only learned in the appearance space. Therefore, we do not require depth at test time and can use G-DPM to detect objects in RGB images. Most other work in object detection/recognition using RGBD [21, 159, 160] uses depth as an extra input feature to learn an object model and therefore, also requires depth information at test time.

5.2 Overview

As input to the system, at training, we use RGB images of object instances along with their underlying geometry in terms of depth data. We convert the depth data into surface normals using the standard procedure from [256]. Our goal is to learn a deformable part-based model where the parts are defined based on their appearance and underlying geometry. We argue that using a geometric representation in conjunction with appearance based deformable parts model not only allows us to have a better initialization but also provides additional constraints during the latent update steps. Specifically, our learning procedure ensures not only that the latent updates are consistent in the appearance space but also that the geometry predicted by underlying parts is consistent with the ground truth geometry. Hence, the depth data is not used as an extra feature, but instead provides weak supervision during the latent update steps.

In this Chapter, we present a proof-of-concept system for building G-DPM. We limit our focus on man-made indoor rigid objects, such as bed, sofa *etc.*, for three reasons: (1) These classes are primarily defined based on their physical properties, and therefore learning a geometric model for these categories makes intuitive sense; (2) These classes have high intra-class variation and are challenging for any deformable parts model. We would like to demonstrate that a joint geometric and appearance based representation gives us a powerful tool to model intra-class variations; (3) Finally, due to the availability of Kinect, data collection for these categories has become simpler and efficient. In our case, we use the NYU v2 dataset [256], which has 1449 RGBD images.

5.3 Technical Approach

Given a large set of training object instances in the form of RGBD data, our goal is to discover a set of candidate parts based on consistent underlying geometry, and use these parts to learn a geometry-driven deformable part-based model (G-DPM). To obtain such a set of candidate parts, we first discover a dictionary of geometric elements based on their depth information (Section 5.3.1) in a category-free manner (pooling the data from all categories). A category-free dictionary allows us to share the elements across multiple object categories.

We use this dictionary to choose a set of parts for every object category based on frequency of occurrence and consistency in the relative location with respect to the object bounding-boxes. Finally, we use these parts to initialize and learn our G-DPM using latent updates and hard mining. We exploit the geometric nature of our parts and use them to enforce additional geometrical constraints at the latent

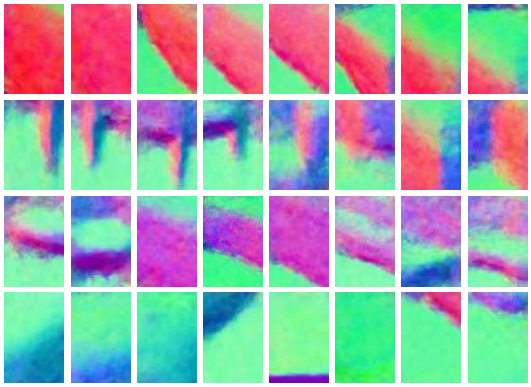


Figure 5.2 – A few elements from the dictionary after the initialization step. They are ordered to highlight the over-completeness of our initial dictionary.

update steps (Section 5.3.3).

5.3.1 Geometry-driven Dictionary of 3D Elements

Given a set of labeled training images and their corresponding surface normal data, our goal is to discover a dictionary of elements capturing 3D information that can act as parts in DPM. Our elements should be: 1) representative: frequent among the object categories in question; 2) spatially consistent with respect to the object. (*e.g.*, a horizontal surface always occurs on the top of a table and bed, while it occurs at center of a chair and a sofa). To obtain a dictionary of candidate elements which satisfy these properties, we use a two step process: first we initialize our dictionary by an over-complete set of elements, each satisfying the representativeness property; and then we refine the dictionary elements based on their relative spatial location with respect to the object.

Initializing the dictionary. We sample hundreds of thousands of patches, in 3 different aspect-ratios (AR), from the object bounding boxes in the training images ($100-500$ patches per object bounding box). We represent these patches in terms of their raw surface normal maps. To extract a representative set of elements for each AR, we perform clustering using simple k-means (with $k \sim 1000$), in raw surface normal space. This clustering process leads to an over-complete set of geometric elements. We remove any cluster with less than N members, for not satisfying the frequency property. We represent every remaining cluster by an element which is the pixel-wise median of the nearest N patches to the cluster center. A few examples of this set of elements is shown in Figure 5.2. In practice, we use $N = 25$. As one can notice from the figure, the dictionary is over-complete. To reject the clusters with

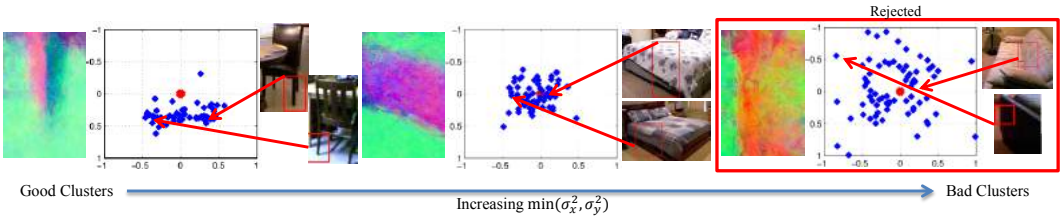


Figure 5.3 – Refinement: After creating an initial dictionary we do the refinement procedure where we find the set of elements that occur at a consistently occur at same spatial location with respect to the object center.

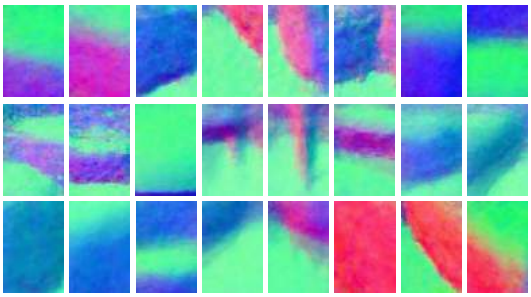


Figure 5.4 – A few examples of resulting elements in dictionary after the refinement procedure.

bad (non-homogenous) members and to remove redundancy we follow this clustering step with a refinement procedure.

Refinement. Given the clusters from the initialization step, we first check each cluster for spatial consistency, i.e., how consistent the cluster is with respect to the center of the object. For this, we record the location of each member in the cluster relative to the object center as: $(dx^i, dy^i) = \left(\frac{(p_x^i - p_x^o)}{w^o}, \frac{(p_y^i - p_y^o)}{h^o}\right)$, where p^o, w^o and h^o are the object center, width and height respectively, and p^i is the center of element i. Examples of this voting scheme are given in Figure 5.3, where each blue dot represents a vote from the cluster's member, and red dot represents object center. To capture consistency in relative locations, we sort the clusters based on $\min(\sigma_x^2(dx), \sigma_y^2(dy))$ (minimum variance of their relative x, y locations). Clusters like the legs of furniture (consistently below the object and closer to the center) and sides of a bed (consistently near the center of object) rank much higher than noisy cluster shown at the right. After pruning bad clusters by thresholding, we perform a step of agglomerative clustering to merge good clusters which are close in feature space (raw surface normals) as well as have consistent distribution of (dx, dy). This gives us a dictionary $\mathcal{D}$ of 3D elements. A few examples of resulting elements are shown in Figure 5.4.

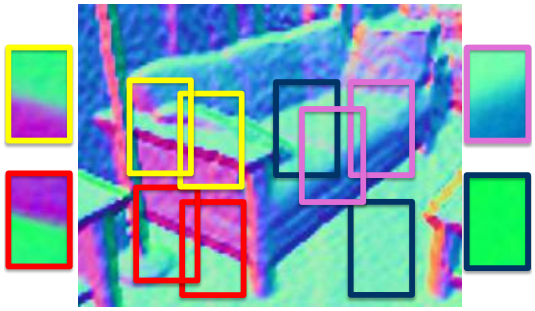


Figure 5.5 – An example of detection/localization of discovered dictionary elements in surface normal space.

5.3.2 From 3D Parts to Object Hypothesis

Given a dictionary of geometric elements $\mathcal{D}$, we would like to discover which geometric elements can act as parts for which object categories. Since our categories share the geometric elements, every element in the dictionary can act as a part for any number of object categories. We represent a part p^j for an object category with three aspects: (a) the geometric element $e_i \in \mathcal{D}$; (b) the relative location $l^j : (dx^j, dy^j)$ of the part with respect to object center (in normalized coordinates); (c) the spring model (or variance in (dx, dy)) for the part, which defines how spatially consistent a part is with respect to the object. Note that an object part is different from the geometric element and a geometric element can act as different parts based on the location (*e.g.*, two armrests for the chair; an armrest is a geometric element but two different parts). The goal is to find set of parts (or an object hypothesis) $\mathbf{p} = [p^1, \ldots, p^N]$, where $p^j : (e_i, l^j)$, that occur consistently in the labeled images.

Similar to DPM [79], we represent each object category as a mixture of components and each component is loosely treated as a category of its own. Therefore, our goal is to find a set of parts for each component of all object categories. Given a set of training images for a component, we first localize each element e in the surface normal map. For example, Figure 5.5 shows the elements detected in the case of a sofa. We then pool the element localizations from all images and find the most frequent elements at different locations in an object. These frequent elements act as candidate parts for representing an object. Figure 5.6(b) shows the candidate parts for one component of three categories: bed, sofa and table.

We now use a greedy approach to select the final parts with the constraints that we have 6-12 parts per object component and that these parts cover at least 60% of the object area. At each step, we select the top-most part hypothesis based on the frequency of occurrence and consistency in the relative location with respect to

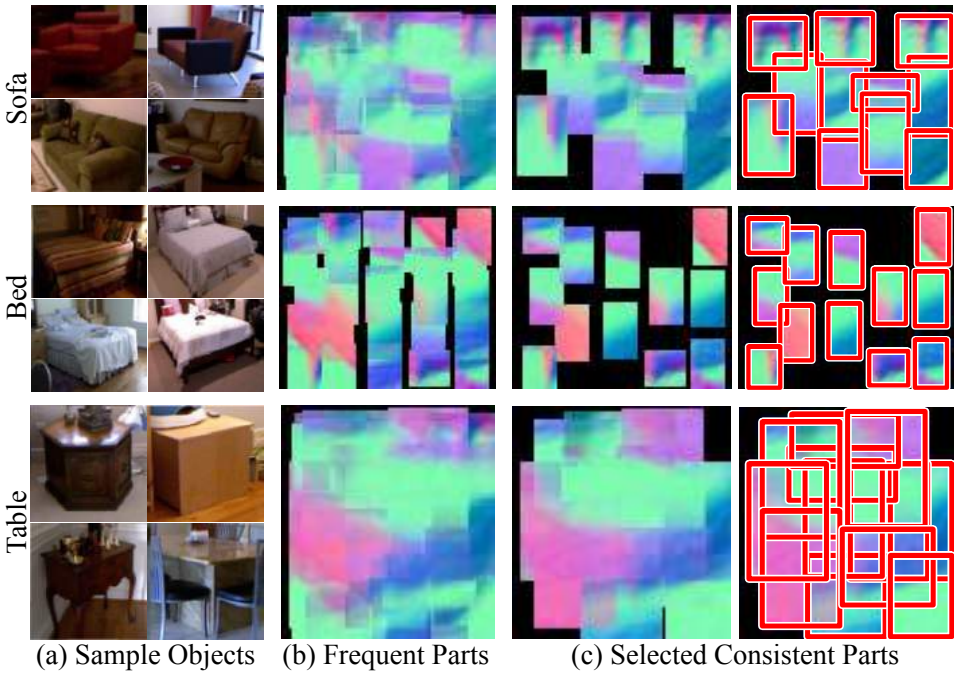


Figure 5.6 – From 3D Parts to object hypothesis: (a) few examples images in the cluster; (b) all the geometrically consistent candidate parts selected (before greedy selection); (c) final part hypothesis for initializing G-DPM (after greedy selection)

the object. Therefore, if a geometric element occurs quite frequently at a particular location, then it is selected as a part for the object. Once we have selected a part, the next part is selected based on frequency and consistency of occurrence, and its overlap with the already selected parts (a part that overlaps a lot with already selected parts is rejected).

5.3.3 Learning G-DPM

Once we have obtained a set of parts for a given object category, we can now use it to initialize the learning of our proposed G-DPM model. Following the general framework of deformable part models [10, 70, 79, 313], we model an object by a mixture of M components, each of which is a non-rigid star-shaped constellation of parts. The key difference between learning the G-DPM and the original DPM lies in the scoring function. Unlike the original model which only captures appearance and location of parts, we explicitly include a geometric consistency term in the scoring function used at the latent update step. This allows us to enforce geometric consistency across the latent update steps and guide the latent updates in the right direction. We will now first discuss a few preliminaries about DPM and then discuss

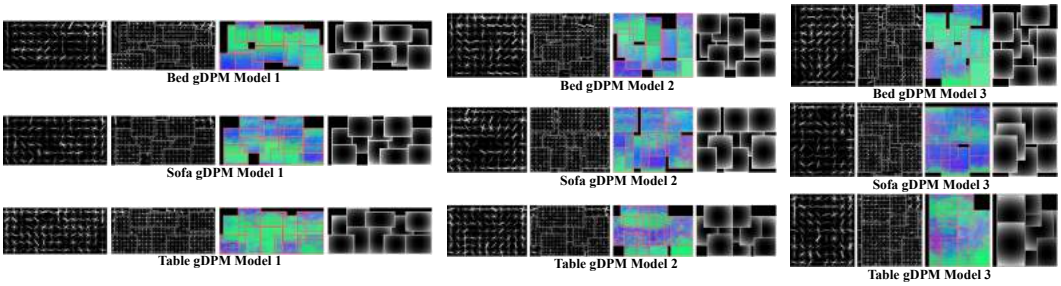


Figure 5.7 – Learned G-DPM models for classes bed, sofa and table. The first visualization in each template represents the learned appearance root filter, the second visualization contains learned part filters super-imposed on the root filter, the third visualization is the surface normal map corresponding to each part and the fourth visualization is of the learned deformation penalty.

how we add the geometric consistency term to the scoring function.

DPM Preliminaries. For each mixture component, indexed by $c \in \{1, \ldots, M\}$, the object hypothesis is specified by $z = (l_0, l_1, \ldots, l_{n_c})$, where $l_i = (u_i, v_i, s_i)$ denotes the (u, v)-position of i-th filter (every part acts a filter) at level s_i in the feature pyramid (root is indexed at 0, and l_0 corresponds to its bounding-box) and n_c is number of parts in component c. Following [79], we enforce that each part is at twice the resolution of the root.

The score of a mixture component c, with model parameter β_c, at any given z (root and part locations) in an image I is given by

$$S(I, z, \beta_c) = \sum_{i=0}^{n_c} F_i \cdot \phi(I, l_i) - \sum_{i=1}^{n_c} d_i \cdot \psi(l_i - l_0) + b \tag{5.1}$$

where the first term scores appearance using image features $\phi(I, l_i)$ (HOG features in this case) and model's appearance parameters $(F_0, \ldots, F_{n_c})$. The second term enforces the deformation penalty using $\psi(l_i - l_0) = \{dx, dy, dx^2, dy^2\}$ where $(dx, dy) = (l_i^x, l_i^y) - (2(l_0^x, l_0^y) - v_i)$ and v_i is the anchor position of the part. Thus, each component's model parameter is $\beta_c = \{F_0, \ldots, F_{n_c}, d_1, \ldots, d_{n_c}, b\}$.

The final score of a DPM model for an object category on an image I at any z is given by

$$S(I, z) = \max_{c \in \{1 \ldots M\}} S(I, z, \beta_c), \tag{5.2}$$

which is the maximum over scores of all the components. Thus, the final object model parameter is $\beta = (\beta_1, \ldots, \beta_M)$ which encapsulates all M mixture components.

Enforcing Geometric Constraints & Learning

Given the training data $\{(x_i, y_i)\}_{1,\ldots,N}$, we aim to learn a discriminative G-DPM. In our case, $x = \{I, I^G, l\}$, where I denotes an RGB image, I^G denotes the surface normal map and l is location of the bounding box, and $y \in \{-1, 1\}$. Similar to [10, 70, 79, 313], we minimize the objective function:

$$L_D(\beta) = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N} \max(0, 1 - y_i f_\beta(x_i)), \tag{5.3}$$

$$f_\beta(x) = \max_z S(I, z) = \max_{z,c} S(I, z, \beta_c). \tag{5.4}$$

The latent variables, z (root and part locations) and c (mixture memberships), make (5.3) non-convex. [79] solves this optimization problem using a coordinate-descent based approach, which iterates between a latent update step and a parameter learning step. In the latent update step, they estimate the latent variables, z and c, by relabeling each positive example. In the parameter learning step, they fix the latent variables and estimate the model parameter β using stochastic gradient descent (SGD).

The latent updates in [79] are made based on image appearance only. However, in our case, we also have a geometric representation of our parts and the underlying depth data for training images. We exploit this and constrain the latent update step such that the part geometry should match the underlying depth data. Intuitively, depth data provides part-level geometric supervision to the latent update step. Thus, enforcing this constraint only affects the latent update step in the above optimization. This is achieved by augmenting the scoring function $S(I, z, \beta_c)$ with a geometric consistency term:

$$f_\beta(x) = \max_{c\in\{1\ldots M\}, z} \left[S(I, z, \beta_c) + \lambda \sum_{i=1}^{n_c} S_G(e^i, \omega(I^G, l_i)) \right] \tag{5.5}$$

where e^i is the geometric element (raw surface normal) corresponding to i-th part, $\omega(I^G, l_i)$ is the raw surface normal map extracted at location l_i, $S_G(\cdot)$ is the geometric similarity function between two raw surface normal maps and λ is the trade-off parameter, controlling how much we want the optimization to focus on geometric consistency. We train our G-DPM models using a modified version of the Latent SVM solver from [79]. In our coordinate-descent approach, the latent update step on positives uses f_β from (5.5) to estimate the latent variables; then we apply SGD to solve for β by using standard f_β (5.4) and hard-negative mining. At test time, we only use the standard scoring function (5.2) (which is also equivalent to setting $\lambda = 0$ in (5.5)).

Table 5.1 – AP performance on the task of object detection

	Bed	Chair	M.+TV	Sofa	Table
DPM (No Parts)	20.94	10.69	6.38	5.51	2.73
DPM	22.39	**14.44**	8.10	7.16	3.53
DPM (Our Parts, No Latent)	26.59	5.71	2.35	6.82	3.41
DPM (Our Parts)	29.15	11.43	4.17	8.30	1.76
G-DPM	**33.39**	13.72	**9.28**	**11.04**	**4.05**

5.4 Experiments

We now present experimental results to demonstrate the effectiveness of adding geometric representation and constraints to a deformable part-based model. We will show how adding 3D parts and geometric constraints not only help improve the performance of our object detector but also help us to develop 3D understanding of the object (in terms of surface normals). We perform our experimental evaluation on the NYU Depth v2 dataset [256]. We learn a G-DPM model for five object categories: bed, chair, monitor+TV (M.+TV), sofa and table. We use 3 components for each object category and some of the learned models are shown in Figure 5.7. This dataset has 1,449 images; we use the train-test splits from [256] (795 training and 654 test images). We convert the object instance segmentation masks (provided by [256]) to bounding boxes for training and testing object detectors. For surface normal prediction for the object, we superimpose the surface normals corresponding to each part and take the pixel-wise median. We also use colorization from [256] to in-paint missing regions in the object for visualization.

Qualitative. Figure 5.8 shows the performance of G-DPM detector on a few examples. Our G-DPM model not only localizes the object better but is also able to predict the surface normals for the detected objects. For example, in the first row, G-DPM not only predicts the flat sittable surface of the couch but it also predicts the vertical backrest and the horizontal surface on the top of it. Similarly, in the second row, our approach is able to predict the horizontal surface of the small table. Figure 5.9 shows one of the false positives of our approach. In this case, a chair is predicted as a sofa by G-DPM but notice the predicted surface normals by G-DPM. Even in the case of wrong category prediction, G-DPM does a reasonable job on the task of predicting surface normals including the horizontal support surface of the chair.

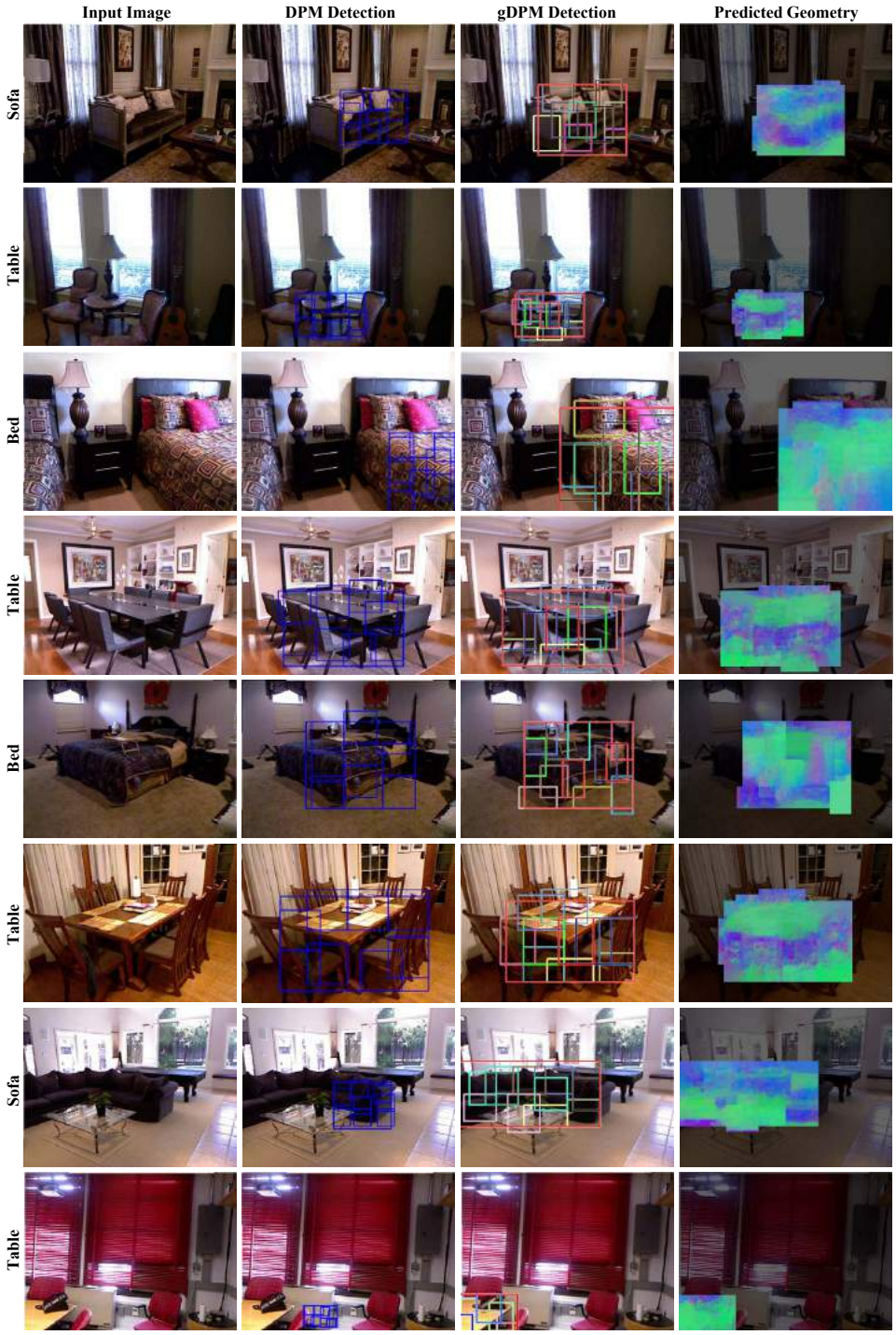


Figure 5.8 – Qualitative Results: Our G-DPM not only localizes the object but also predicts the surface normals of the objects.

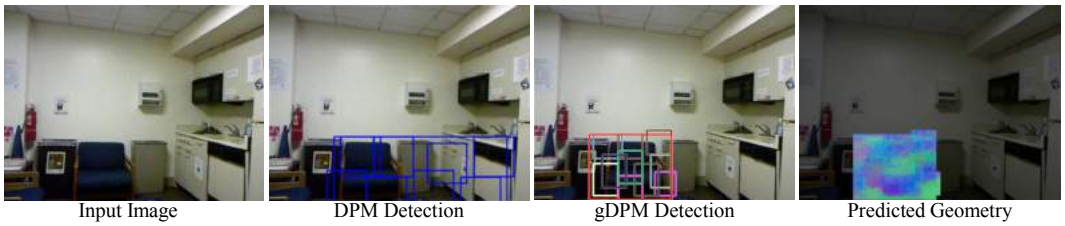


Figure 5.9 – False Positives: Our sofa detector detecting chair. Notice that the geometry still looks plausible.

Quantitative. We now evaluate G-DPM quantitatively on the task of 2D object detection. As a baseline, we compare our approach against the standard DPM model with and without parts. We also evaluate the performance of DPM by treating our initial part hypothesis as strong supervision (ground truth parts) and not doing any latent updates. Finally, we also evaluate the performance of our parts with the standard latent updates which do not consider the geometric constraint based on depth data. Table 5.1 shows the average precision (AP). Our approach improves over the standard DPM by approximately 3.2% mean AP over 5 categories; and for categories like bed and sofa, the improvement is as much as 11% and 4% respectively. We also evaluate our surface normal prediction accuracy in a small quantitative experiment. Against Geometric Context [126], our surface normal prediction is 2° better, in terms of median per-pixel error.

Conclusions. We proposed a novel part-based representation, geometry-driven deformable part-based model (G-DPM), where the parts are defined based on their 3D properties. G-DPM effectively leverages depth data to combine the power of DPMs with the richness of geometric representation. We demonstrate how depth data can be used to define parts and provide weak supervision during the latent update steps. This leads to a better model in terms of detection performance. But more importantly, a joint geometric and appearance based representation allows us to jointly tackle the grand challenge of object detection and understanding 3D objects from 2D images.

Chapter 6

Cross-stitch Networks for Multi-task Learning

> The miracle is this: the more we share, the more we have.
>
> Leonard Nimoy

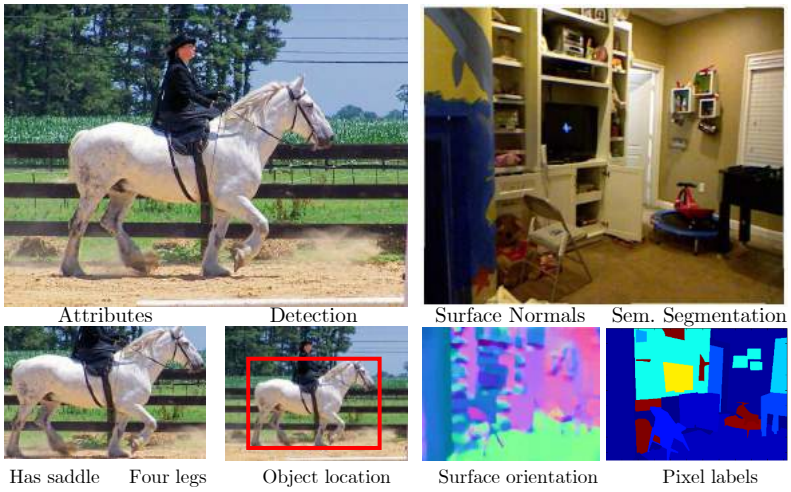


Figure 6.1 – Given an input image, one can leverage multiple related properties to improve performance by using a multi-task learning framework. In this Chapter, we propose *cross-stitch* units, a principled way to use such a multi-task framework for ConvNets.

Visual data often has multiple supervisory labels for recognition tasks (*e.g.*, labels for scenes, objects, attributes, depth, *etc.*). Jointly training models for related tasks, or multi-task learning, has been widely successful. One of the reasons for this success is attributed to the inbuilt sharing mechanism, which allows ConvNets

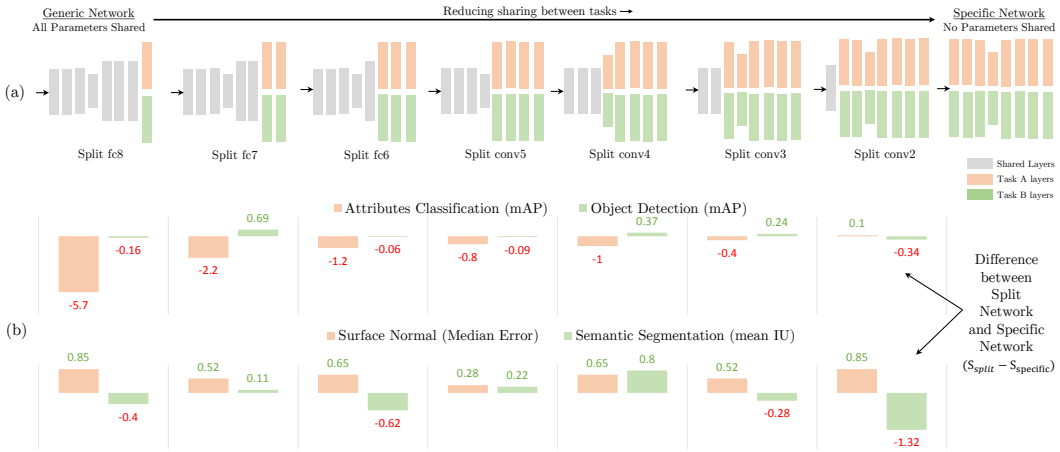


Figure 6.2 – We train a variety of multi-task (two-task) architectures by splitting at different layers in a ConvNet [155] for two pairs of tasks. For each of these networks, we plot their performance on each task relative to the task-specific network. We notice that the best performing multi-task architecture depends on the individual tasks and does not transfer across different pairs of tasks.

to learn representations shared across different categories. This insight naturally extends to sharing between tasks (see Figure 6.1) and leads to further performance improvements, *e.g.*, the gains in segmentation [115] and detection [93, 95]. A key takeaway from these works is that multiple tasks, and thus multiple types of supervision, helps achieve better performance with the same input. But unfortunately, the network architectures used by them for multi-task learning notably differ. There are no insights or principles for how one should choose ConvNet architectures for multi-task learning.

6.0.1 Multi-task sharing: an empirical study

How should one pick the right architecture for multi-task learning? Does it depend on the final tasks? Should we have a completely shared representation between tasks? Or should we have a combination of shared and task-specific representations? Is there a principled way of answering these questions?

To investigate these questions, we first perform extensive experimental analysis to understand the performance trade-offs amongst different combinations of shared and task-specific representations. Consider a simple experiment where we train a ConvNet on two related tasks (*e.g.*, semantic segmentation and surface normal estimation). Depending on the amount of sharing one wants to enforce, there is a spectrum of possible network architectures. Figure 6.2(a) shows different ways of creating such network architectures based on AlexNet [155]. On one end of the spectrum is a fully shared representation where all layers, from the first convolution

(`conv2`) to the last fully-connected (`fc7`), are shared and only the last layers (two `fc8`s) are task specific. An example of such sharing is [95] where separate `fc8` layers are used for classification and bounding box regression. On the other end of the sharing spectrum, we can train two networks separately for each task and there is no cross-talk between them. In practice, different amount of sharing tends to work best for different tasks.

So given a pair of tasks, how should one pick a network architecture? To empirically study this question, we pick two varied pairs of tasks:

- We first pair semantic segmentation (SemSeg) and surface normal prediction (SN). We believe the two tasks are closely related to each other since segmentation boundaries also correspond to surface normal boundaries. For this pair of tasks, we use NYU-v2 [256] dataset.
- For our second pair of tasks we use detection (Det) and Attribute prediction (Attr). Again we believe that two tasks are related: for example, a box labeled as "car" would also be a positive example of "has wheel" attribute. For this experiment, we use the attribute PASCAL dataset [72, 76].

We exhaustively enumerate all the possible *Split* architectures as shown in Figure 6.2(a) for these two pairs of tasks and show their respective performance in Figure 6.2(b). The best performance for both the SemSeg and SN tasks is using the "Split `conv4`" architecture (splitting at `conv4`), while for the Det task it is using the Split `conv2`, and for Attr with Split `fc6`. These results indicate two things – 1) Networks learned in a multi-task fashion have an edge over networks trained with one task; and 2) The best Split architecture for multi-task learning depends on the tasks at hand.

While the gain from multi-task learning is encouraging, getting the most out of it is still cumbersome in practice. This is largely due to the task dependent nature of picking architectures and the lack of a principled way of exploring them. Additionally, enumerating all possible architectures for each set of tasks is impractical. This Chapter proposes *cross-stitch units*, using which a single network can capture all these Split-architectures (and more). It automatically learns an optimal combination of shared and task-specific representations. We demonstrate that such a *cross-stitched* network can achieve better performance than the networks found by brute-force enumeration and search.

6.1 Related Work

Generic Multi-task learning [34, 264] has a rich history in machine learning. The term *multi-task learning* (MTL) itself has been broadly used [5, 74, 132, 225, 311, 312] as an umbrella term to include representation learning and selection [8, 73, 147, 199], transfer learning [205, 222, 315] *etc.* and their widespread applications in other fields, such as genomics [200], natural language processing [49, 50, 179] and computer vision [6, 63, 142, 147, 218, 283, 306, 321]. In fact, many times multi-task learning is implicitly used without reference; a good example being fine-tuning or transfer learning [222], now a mainstay in computer vision, can be viewed as sequential multi-task learning [34]. Given the broad scope, in this Section we focus only on multi-task learning in the context of ConvNets used in computer vision.

Multi-task learning is generally used with ConvNets in computer vision to model related tasks jointly, *e.g.* pose estimation and action recognition [97], surface normals and edge labels [299], face landmark detection and face detection [320, 322], auxiliary tasks in detection [95], related classes for image classification [276] *etc.* Usually these methods share some features (layers in ConvNets) amongst tasks and have some task-specific features. This sharing or split-architecture (as explained in Section 6.0.1) is decided after experimenting with splits at multiple layers and picking the best one. Of course, depending on the task at hand, a different Split architecture tends to work best, and thus given new tasks, new split architectures need to be explored. In this Chapter, we propose *cross-stitch units* as a principled approach to explore and embody such Split architectures, without having to train all of them.

In order to demonstrate the robustness and effectiveness of cross-stitch units in multi-task learning, we choose varied tasks on multiple datasets. In particular, we select four well established and diverse tasks on different types of image datasets: 1) We pair semantic segmentation [122, 247, 248] and surface normal estimation [67, 85, 299], both of which require predictions over all pixels, on the NYU-v2 indoor dataset [256]. These two tasks capture both semantic and geometric information about the scene. 2) We choose the task of object detection [79, 95, 96, 238] and attribute prediction [2, 77, 163] on web-images from the PASCAL dataset [72, 76]. These tasks make predictions about localized regions of an image.

6.2 Cross-stitch Networks

In this Chapter, we present a novel approach to multi-task learning for ConvNets by proposing cross-stitch units. Cross-stitch units try to find the best shared representations for multi-task learning. They model these shared representations using

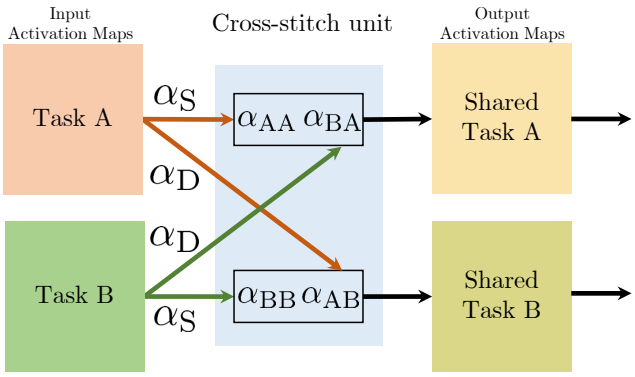


Figure 6.3 – We model shared representations by learning a linear combination of input activation maps. At each layer of the network, we learn such a linear combination of the activation maps from both the tasks. The next layers' filters operate on this shared representation.

linear combinations, and learn the optimal linear combinations for a given set of tasks. We integrate these cross-stitch units into a ConvNet and provide an end-to-end learning framework. We use detailed ablative studies to better understand these units and their training procedure. Further, we demonstrate the effectiveness of these units for two different pairs of tasks. To limit the scope of this Chapter, we only consider tasks which take the same single input, *e.g.*, an image as opposed to say an image and a depth-map [109].

6.2.1 Split Architectures

Given a single input image with multiple labels, one can design "Split architectures" as shown in Figure 6.2. These architectures have both a shared representation and a task specific representation. 'Splitting' a network at a lower layer allows for more task-specific and fewer shared layers. One extreme of Split architectures is splitting at the lowest convolution layer which results in two separate networks altogether, and thus only task-specific representations. The other extreme is using "sibling" prediction layers (as in [95]), which allows for a more shared representation. Thus, Split architectures allow for a varying amount of shared and task-specific representations.

6.2.2 Unifying Split Architectures

Given that Split architectures hold promise for multi-task learning, an obvious question is – At which layer of the network should one split? This decision is highly dependent on the input data and tasks at hand. Rather than enumerating the possibilities of Split architectures for every new input task, we propose a simple

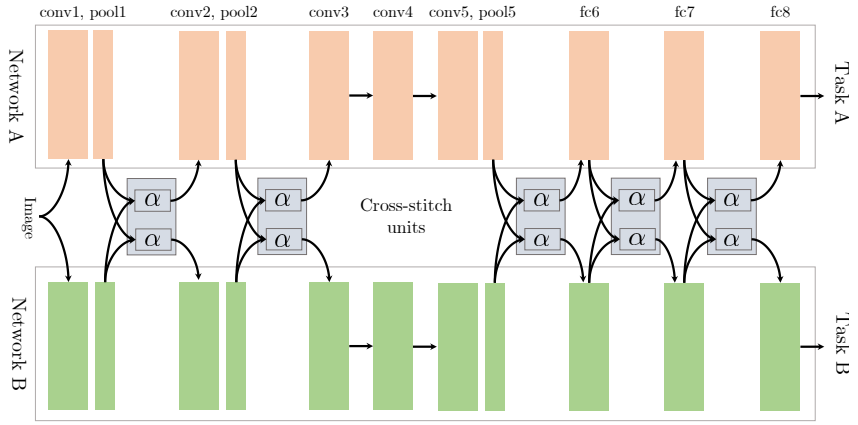


Figure 6.4 – Using cross-stitch units to stitch two AlexNet [155] networks. In this case, we apply cross-stitch units only after pooling layers and fully connected layers. Cross-stitch units can model shared representations as a linear combination of input activation maps. This network tries to learn representations that can help with both tasks A and B. We call the sub-network that gets direct supervision from task A as network A (top) and the other as network B (bottom).

architecture that can learn how much shared and task specific representation to use.

6.2.3 Cross-stitch units

Consider a case of multi task learning with two tasks A and B on the same input image. For the sake of explanation, consider two networks that have been trained separately for these tasks. We propose a new unit, *cross-stitch unit*, that combines these two networks into a multi-task network in a way such that the tasks supervise how much sharing is needed, as illustrated in Figure 6.3. At each layer of the network, we model sharing of representations by learning a linear combination of the activation maps [8, 147] using a *cross-stitch unit*. Given two activation maps $x_{\mathrm{A}}, x_{\mathrm{B}}$ from layer l for both the tasks, we learn linear combinations $\tilde{x}_{\mathrm{A}}, \tilde{x}_{\mathrm{B}}$ (Eq 6.1) of both the input activations and feed these combinations as input to the next layers' filters. This linear combination is parameterized using α. Specifically, at location (i, j) in the activation map,

$$\begin{bmatrix} \tilde{x}_{\mathrm{A}}^{ij} \\ \tilde{x}_{\mathrm{B}}^{ij} \end{bmatrix} = \begin{bmatrix} \alpha_{\mathrm{AA}} & \alpha_{\mathrm{AB}} \\ \alpha_{BA} & \alpha_{\mathrm{BB}} \end{bmatrix} \begin{bmatrix} x_{\mathrm{A}}^{ij} \\ x_{\mathrm{B}}^{ij} \end{bmatrix} \tag{6.1}$$

We refer to this the *cross-stitch* operation, and the unit that models it for each layer l as the *cross-stitch unit*. The network can decide to make certain layers task specific by setting α_{AB} or α_{BA} to zero, or choose a more shared representation by assigning a higher value to them.

Backpropagating through cross-stitch units. Since cross-stitch units are modeled as linear combination, their partial derivatives for loss L with tasks A, B are computed as

$$\begin{bmatrix} \frac{\partial L}{\partial x_{\mathrm{A}}^{ij}} \\ \frac{\partial L}{\partial x_{\mathrm{B}}^{ij}} \end{bmatrix} = \begin{bmatrix} \alpha_{\mathrm{AA}} & \alpha_{\mathrm{BA}} \\ \alpha_{\mathrm{AB}} & \alpha_{\mathrm{BB}} \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial \tilde{x}_{\mathrm{A}}^{ij}} \\ \frac{\partial L}{\partial \tilde{x}_{\mathrm{B}}^{ij}} \end{bmatrix} \tag{6.2}$$

$$\frac{\partial L}{\partial \alpha_{\mathrm{AB}}} = \frac{\partial L}{\partial \tilde{x}_{\mathrm{B}}^{ij}} x_{\mathrm{A}}^{ij}, \quad \frac{\partial L}{\partial \alpha_{\mathrm{AA}}} = \frac{\partial L}{\partial \tilde{x}_{\mathrm{A}}^{ij}} x_{\mathrm{A}}^{ij} \tag{6.3}$$

We denote $\alpha_{\mathrm{AB}}, \alpha_{\mathrm{BA}}$ by α_{D} and call them the *different*-task values because they weigh the activations of another task. Likewise, $\alpha_{\mathrm{AA}}, \alpha_{\mathrm{BB}}$ are denoted by α_{S}, the *same*-task values, since they weigh the activations of the same task. By varying α_{D} and α_{S} values, the unit can freely move between shared and task-specific representations, and choose a middle ground if needed.

6.3 Design decisions for cross-stitching

We use the cross-stitch unit for multi-task learning in ConvNets. For the sake of simplicity, we assume multi-task learning with two tasks. Figure 6.4 shows this architecture for two tasks A and B. The sub-network in Figure 6.4(top) gets direct supervision from task A and indirect supervision (through cross-stitch units) from task B. We call the sub-network that gets direct supervision from task A as network A, and correspondingly the other as B. Cross-stitch units help regularize both tasks by learning and enforcing shared representations by combining activation (feature) maps. As we show in our experiments, in the case where one task has less labels than the other, such regularization helps the "data-starved" tasks.

Next, we enumerate the design decisions when using cross-stitch units with networks, and in later Sections perform ablative studies on each of them.

Cross-stitch units initialization and learning rates: The α values of a cross-stitch unit model linear combinations of feature maps. Their initialization in the range $[0, 1]$ is important for stable learning, as it ensures that values in the output activation map (after cross-stitch unit) are of the same order of magnitude as the input values before linear combination. We study the impact of different initializations and learning rates for cross-stitch units in Section 6.4.

Network initialization: Cross-stitch units combine together two networks as shown in Figure 6.4. However, an obvious question is – how should one initialize the networks A and B? We can initialize networks A and B by networks that were trained on these tasks separately, or have the same initialization and train them jointly.

6.4 Ablative analysis

We now describe the experimental setup in detail, which is common throughout the ablation studies.

Datasets and Tasks: For ablative analysis we consider the tasks of semantic segmentation (SemSeg) and Surface Normal Prediction (SN) on the NYU-v2 [256] dataset. We use the standard train/test splits from [85]. For semantic segmentation, we follow the setup from [108] and evaluate on the 40 classes using the standard metrics from their work

Setup for Surface Normal Prediction: Following [299], we cast the problem of surface normal prediction as classification into one of 20 categories. For evaluation, we convert the model predictions to 3D surface normals and apply the Manhattan-World post-processing following the method in [299]. We evaluate all our methods using the metrics from [85]. These metrics measure the error in the ground truth normals and the predicted normals in terms of their angular distance (measured in degrees). Specifically, they measure the mean and median error in angular distance, in which case lower error is better (denoted by 'Mean' and 'Median' error). They also report percentage of pixels which have their angular distance under a threshold (denoted by 'Within t°' at a threshold of $11.25^\circ, 22.5^\circ, 30^\circ$), in which case a higher number indicates better performance.

Networks: For semantic segmentation (SemSeg) and surface normal (SN) prediction, we use the Fully-Convolutional Network (FCN 32-s) architecture from [180] based on CaffeNet [135] (essentially AlexNet [155]). For both the tasks of SemSeg and SN, we use RGB images at full resolution, and use mirroring and color data augmentation. We then finetune the network (referred to as *one-task network*) from ImageNet [58] for each task using hyperparameters reported in [180]. We fine-tune the network for semantic segmentation for 25k iterations using SGD (mini-batch size 20) and for surface normal prediction for 15k iterations (mini-batch size 20) as they gave the best performance, and further training (up to 40k iterations) showed no improvement. These *one-task networks* serve as our baselines and initializations for cross-stitching, when applicable.

Table 6.1 – Initializing cross-stitch units with different α values, each corresponding to a convex combination. Higher values for α_S indicate that we bias the cross-stitch unit to prefer task specific representations. The cross-stitched network is robust across different initializations of the units

	Surface Normal					Segmentation		
	Angle Distance (Lower Better)		Within t° (Higher Better)			(Higher Better)		
(α_S, α_D)	Mean	Med.	11.25	22.5	30	pixacc	mIU	fwIU
$(0.1, 0.9)$	34.6	18.8	38.5	53.7	59.4	47.9	18.2	33.3
$(0.5, 0.5)$	34.4	18.8	38.5	53.7	59.5	47.2	18.6	33.8
$(0.7, 0.3)$	**34.0**	**18.3**	38.9	54.3	60.1	48.0	18.6	33.6
$(0.9, 0.1)$	**34.0**	**18.3**	**39.0**	**54.4**	**60.2**	**48.2**	**18.9**	**34.0**

Cross-stitching: We combine two AlexNet architectures using the cross-stitch units as shown in Figure 6.4. We experimented with applying cross-stitch units after every convolution activation map and after every pooling activation map, and found the latter performed better. Thus, the cross-stitch units for AlexNet are applied on the activation maps for `pool1`, `pool2`, `pool5`, `fc6` and `fc7`. We maintain one cross-stitch unit per 'channel' of the activation map, *e.g.*, for `pool1` we have 96 cross-stitch units.

6.4.1 Initializing parameters of cross-stitch units

Cross-stitch units capture the intuition that shared representations can be modeled by linear combinations [147]. To ensure that values after the cross-stitch operation are of the same order of magnitude as the input values, an obvious initialization of the unit is that the α values form a convex linear combination, *i.e.*, the different-task α_D and the same-task α_S to sum to one. Note that this convexity is not enforced on the α values in either Equation 6.1 or 6.2, but serves as a reasonable initialization. For this experiment, we initialize the networks A and B with *one-task* networks that were fine-tuned on the respective tasks. Table 6.1 shows the results of evaluating cross-stitch networks for different initializations of α values.

6.4.2 Learning rates for cross-stitch units

We initialize the α values of the cross-stitch units in the range $[0.1, 0.9]$, which is about one to two orders of magnitude larger than the typical range of layer parameters in AlexNet [155]. While training, we found that the gradient updates at various layers had magnitudes which were reasonable for updating the layer parameters, but too small for the cross-stitch units. Thus, we use higher learning rates for the cross-stitch units than the base network. In practice, this leads to faster convergence and better performance. To study the impact of different learning rates, we

Table 6.2 – Scaling the learning rate of cross-stitch units wrt. the base network. Since the cross-stitch units are initialized in a different range from the layer parameters, we scale their learning rate for better training

	Surface Normal					Segmentation		
	Angle Distance (Lower Better)		Within t° (Higher Better)			(Higher Better)		
Scale	Mean	Med.	11.25	22.5	30	pixacc	mIU	fwIU
1	34.6	18.9	38.4	53.7	59.4	47.7	18.6	33.5
10	34.5	18.8	38.5	53.8	59.5	47.8	18.7	33.5
10^2	**34.0**	18.3	39.0	54.4	60.2	**48.0**	18.9	33.8
10^3	34.1	**18.2**	**39.2**	**54.4**	**60.2**	47.2	**19.3**	**34.0**

again use a cross-stitched network initialized with two *one-task networks*. We scale the learning rates (wrt. the network's learning rate) of cross-stitch units in powers of 10 (by setting the `lr_mult` layer parameter in Caffe [135]). Table 6.2 shows the results of using different learning rates for the cross-stitch units after training for 10k iterations. Setting a higher scale for the learning rate improves performance, with the best range for the scale being $10^2 - 10^3$. We observed that setting the scale to an even higher value made the loss diverge.

6.4.3 Initialization of networks A and B

When cross-stitching two networks, how should one initialize the networks A and B? Should one start with task specific *one-task networks* (fine-tuned for one task only) and add cross-stitch units? Or should one start with networks that have not been fine-tuned for the tasks? We explore the effect of both choices by initializing using two *one-task networks* and two networks trained on ImageNet [58, 232]. We train the *one-task* initialized cross-stitched network for 10k iterations and the ImageNet initialized cross-stitched network for 30k iterations (to account for the 20k fine-tuning iterations of the *one-task* networks), and report the results in Table 6.3. Task-specific initialization performs better than ImageNet initialization for both the tasks, which suggests that cross-stitching should be used after training task-specific networks.

6.4.4 Visualization of learned combinations

We visualize the weights α_S and α_D of the cross-stitch units for different initializations in Figure 6.4. For this experiment, we initialize sub-networks A and B using *one-task* networks and trained the cross-stitched network till convergence. Each plot shows (in sorted order) the α values for all the cross-stitch units in a layer (one per

Table 6.3 – We initialize the networks A, B (from Figure 6.4) from ImageNet, as well as task-specific networks. We observe that task-based initialization performs better than task-agnostic ImageNet initialization

	Surface Normal					Segmentation		
	Angle Distance (Lower Better)		Within t° (Higher Better)			(Higher Better)		
Init.	Mean	Med.	11.25	22.5	30	pixacc	mIU	fwIU
ImageNet	34.6	18.8	38.6	53.7	59.4	**48.0**	17.7	33.4
One-task	**34.1**	**18.2**	**39.0**	**54.4**	**60.2**	47.2	**19.3**	**34.0**

channel). We show plots for three layers: `pool1`, `pool5` and `fc7`. The initialization of cross-stitch units biases the network to start its training preferring a certain type of shared representation, *e.g.*, $(\alpha_S, \alpha_D) = (0.9, 0.1)$ biases the network to learn more task-specific features, while $(0.5, 0.5)$ biases it to share representations. Figure 6.4 (second row) shows that both the tasks, across all initializations, prefer a more task-specific representation for `pool5`, as shown by higher values of α_S. This is inline with the observation from Section 6.0.1 that Split `conv4` performs best for these two tasks. We also notice that the surface normal task prefers shared representations as can be seen by Figure 6.4(b), where α_S and α_D values are in similar range.

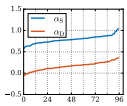
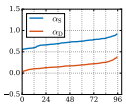
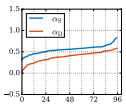
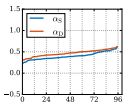
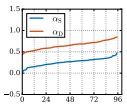
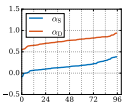
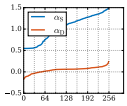
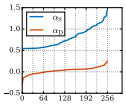
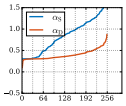
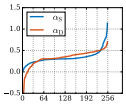
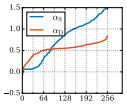
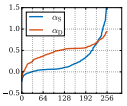
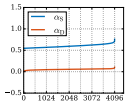
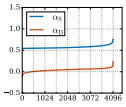
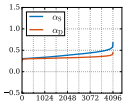
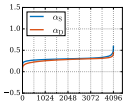
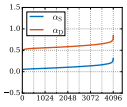
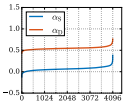
6.5 Experiments

We now present experiments with cross-stitch networks for two pairs of tasks: semantic segmentation and surface normal prediction on NYU-v2 [256], and object detection and attribute prediction on PASCAL VOC 2008 [72, 76]. We use the experimental setup from Section 6.4 for semantic segmentation and surface normal prediction, and describe the setup for detection and attribute prediction below.

Dataset, Metrics and Network: We consider the PASCAL VOC 20 classes for object detection, and the 64 attribute categories data from [76]. We use the PASCAL VOC 2008 [72, 76] dataset for our experiments and report results using the standard Average Precision (AP) metric. We start with the recent Fast-RCNN [95] method for object detection using the AlexNet [155] architecture.

Training: For object detection, Fast-RCNN is trained using 21-way 1-vs-all classification with 20 foreground and 1 background class. However, there is a severe data imbalance in the foreground and background data points (boxes). To circumvent this, Fast-RCNN carefully constructs mini-batches with $1:3$ foreground-to-background ratio, *i.e.*, at most 25% of foreground samples in a mini-batch. Attribute prediction, on the other hand, is a multi-label classification problem with 64

Table 6.4 – We show the sorted α values (increasing left to right) for three layers. A higher value of α_S indicates a strong preference towards task specific features, and a higher α_D implies preference for shared representations. More detailed analysis in Section 6.4.4. Note that both α_S and α_D are sorted independently, so the channel-index across them do not correspond

Layer	(a) $\alpha_S = 0.9, \alpha_D = 0.1$		(b) $\alpha_S = 0.5, \alpha_D = 0.5$		(c) $\alpha_S = 0.1, \alpha_D = 0.9$	
	Segmentation	Surface Normal	Segmentation	Surface Normal	Segmentation	Surface Normal
pool1						
pool5						
fc7						

attributes, which only train using foreground bounding boxes. To implement both tasks in the Fast R-CNN framework, we use the same mini-batch sampling strategy; and in every mini-batch only the foreground samples contribute to the attribute loss (and background samples are ignored).

Scaling losses: Both SemSeg and SN used same classification loss for training, and hence we were set their loss weights to be equal (= 1). However, since object detection is formulated as 1-vs-all classification and attribute classification as multi-label classification, we balance the losses by scaling the attribute loss by 1/64.

Cross-stitching: We combine two AlexNet architectures using the cross-stitch units after every pooling layer as shown in Figure 6.4. In the case of object detection and attribute prediction, we use one cross-stitch unit per layer activation map. We found that maintaining a unit per channel, like in the case of semantic segmentation, led to unstable learning for these tasks.

6.5.1 Baselines

We compare against four strong baselines for the two pairs of tasks and report the results in Table 6.5 and 6.6.

Single-task Baselines: These serve as baselines without benefits of multi-task learning. First we evaluate a single network trained on only one task (denoted by 'One-task') as described in Section 6.4. Since our approach cross-stitches two networks and therefore uses 2× parameters, we also consider an ensemble of two one-task networks (denoted by 'Ensemble'). However, note that the ensemble has

Table 6.5 – Surface normal prediction and semantic segmentation results on the NYU-v2 [256] dataset. Our method outperforms the baselines for both the tasks

	Surface Normal					Segmentation		
	Angle Distance (Lower Better)		Within t° (Higher Better)			(Higher Better)		
Method	Mean	Med.	11.25	22.5	30	pixacc	mIU	fwIU
One-task	34.8	19.0	38.3	53.5	59.2	-	-	-
	-	-	-	-	-	46.6	18.4	33.1
Ensemble	34.4	18.5	38.7	54.2	59.7	-	-	-
	-	-	-	-	-	**48.2**	18.9	33.8
Split `conv4`	34.7	19.1	38.2	53.4	59.2	47.8	19.2	33.8
MTL-shared	34.7	18.9	37.7	53.5	58.8	45.9	16.6	30.1
Cross-stitch [ours]	**34.1**	**18.2**	**39.0**	**54.4**	**60.2**	47.2	**19.3**	**34.0**

$2\times$ network parameters for only one task, while the cross-stitch network has roughly $2\times$ parameters for two tasks. So for a pair of tasks, the ensemble baseline uses $\sim 2\times$ the cross-stitch parameters.

Multi-task Baselines: The cross-stitch units enable the network to pick an optimal combination of shared and task-specific representation. We demonstrate that these units remove the need for finding such a combination by exhaustive brute-force search (from Section 6.0.1). So as a baseline, we train all possible "Split architectures" for each pair of tasks and report numbers for the best Split for each pair of tasks.

There has been extensive work in Multi-task learning outside of the computer vision and deep learning community. However, most of such work, with publicly available code, formulates multi-task learning in an optimization framework that requires all data points in memory [38, 74, 104, 164, 266, 324, 325]. Such requirement is not practical for the vision tasks we consider.

So as our final baseline, we compare to a variant of [2, 326] by adapting their method to our setting and report this as 'MTL-shared'. The original method treats each category as a separate 'task', *a separate network* is required for each category and *all these networks are trained jointly*. Directly applied to our setting, this would require training 100s of ConvNets jointly, which is impractical. Thus, instead of treating each category as an independent task, we adapt their method to our two-task setting. We train these two networks jointly, using end-to-end learning, as opposed to their dual optimization to reduce hyperparameter search.

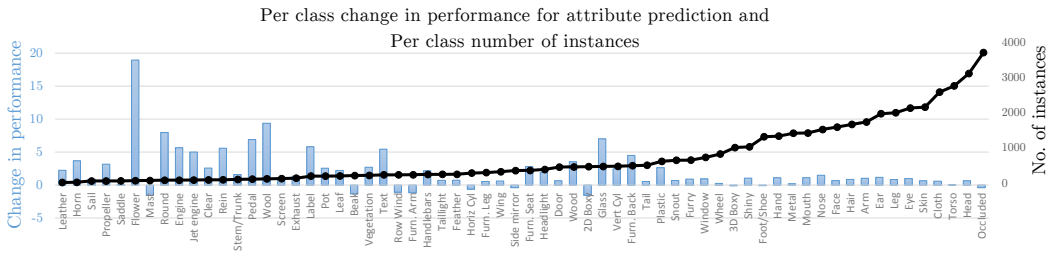


Figure 6.5 – Change in performance for attribute categories over the baseline is indicated by blue bars. We sort the categories in increasing order (from left to right) by the number of instance labels in the train set, and indicate the number of instance labels by the solid black line. The performance gain for attributes with lesser data (towards the left) is considerably higher compared to the baseline. We also notice that the gain for categories with lots of data is smaller.

6.5.2 Semantic Segmentation and Surface Normal Prediction

Table 6.5 shows the results for semantic segmentation and surface normal prediction on the NYUv2 dataset [256]. We compare against two one-task networks, an ensemble of two networks, and the best Split architecture (found using brute force enumeration). The sub-networks A, B (Figure 6.4) in our cross-stitched network are initialized from the one-task networks. We use cross-stitch units after every pooling layer and fully connected layer (one per channel). Our proposed cross-stitched network improves results over the baseline one-task networks and the ensemble. Note that even though the ensemble has 2× parameters compared to cross-stitched network, the latter performs better. Finally, our performance is better than the best Split architecture network found using brute force search. This shows that the cross-stitch units can effectively search for optimal amount of sharing in multi-task networks.

6.5.3 Data-starved categories for segmentation

Multiple tasks are particularly helpful in regularizing the learning of shared representations[34, 74, 276]. This regularization manifests itself empirically in the improvement of "data-starved" (few examples) categories and tasks.

For semantic segmentation, there is a high mismatch in the number of labels per category (see the black line in Figure 6.6). Some classes like *wall*, *floor* have many more instances than other classes like *bag*, *whiteboard etc.* Figure 6.6 also shows the per-class gain in performance using our method over the baseline one-task network. We see that cross-stitch units considerably improve the performance of "data-starved" categories (*e.g.*, *bag*, *whiteboard*).

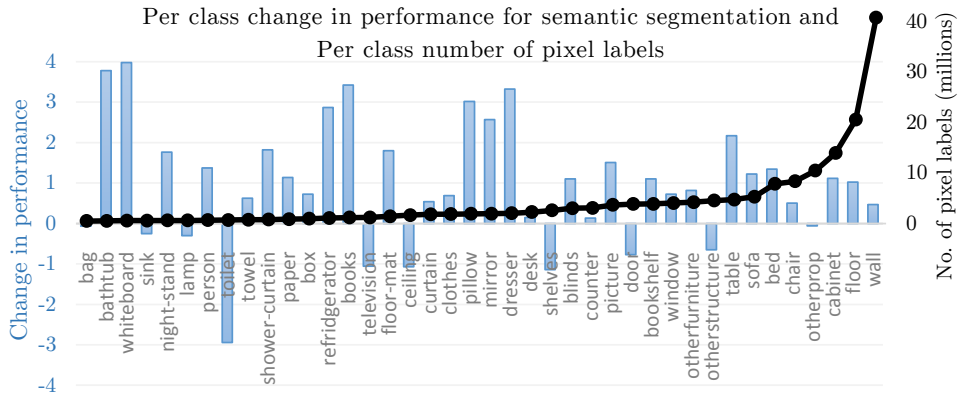


Figure 6.6 – Change in performance (meanIU metric) for semantic segmentation categories over the baseline is indicated by blue bars. We sort the categories (in increasing order from left to right) by the number of pixel labels in the train set, and indicate the number of pixel labels by a solid black line. The performance gain for categories with lesser data (towards the left) is more when compared to the baseline one-task network.

6.5.4 Object detection and attribute prediction

We train a cross-stitch network for the tasks of object detection and attribute prediction. We compare against baseline one-task networks and the best split architectures per task (found after enumeration and search, Section 6.0.1). Table 6.6 shows the results for object detection and attribute prediction on PASCAL VOC 2008 [72, 76]. Our method shows improvements over the baseline for attribute prediction. It is worth noting that because we use a background class for detection, and not attributes (described in 'Scaling losses' in Section 6.5), detection has many more data points than attribute classification (only 25% of a mini-batch has attribute labels). Thus, we see an improvement for the data-starved task of attribute prediction. It is also interesting to note that the detection task prefers a shared representation (best performance by Split `fc7`), whereas the attribute task prefers a task-specific network (best performance by Split `conv2`).

6.5.5 Data-starved categories for attribute prediction

Following a similar analysis to Section 6.5.3, we plot the relative performance of our cross-stitch approach over the baseline one-task attribute prediction network in Figure 6.5. The performance gain for attributes with smaller number of training examples is considerably large compared to the baseline (**4.6**% and **4.3**% mAP for the top 10 and 20 attributes with the least data respectively). This shows that our proposed cross-stitch method provides significant gains for data-starved tasks by learning shared representations.

Table 6.6 – Object detection and attribute prediction results on the attribute PASCAL [76] 2008 dataset

Method	Detection (mAP)	Attributes (mAP)
One-task	44.9 -	- 60.9
Ensemble	**46.1** -	- 61.1
Split `conv2` Split `fc7`	44.6 44.8	61.0 59.7
MTL-shared	42.7	54.1
Cross-stitch [ours]	45.2	**63.0**

Conclusion. We present *cross-stitch* units which are a generalized way of learning shared representations for multi-task learning in ConvNets. Cross-stitch units model shared representations as linear combinations, and can be learned end-to-end in a ConvNet. These units generalize across different types of tasks and eliminate the need to search through several multi-task network architectures on a per task basis. We show detailed ablative experiments to see effects of hyperparameters, initialization *etc.* when using these units. We also show considerable gains over the baseline methods for data-starved categories. Studying other properties of cross-stitch units, such as where in the network should they be used and how should their weights be constrained, is an interesting future direction.

Part II

Recognition Beyond Extensive Supervision

Chapter 7

Learning Visual Similarity

Approach and Applications

> The things that stand out
> are often the oddities.
>
> Pierre Salinger

Figure 7.1 – In this Chapter, we are interested in defining visual similarity between images across different domains, such as photos taken in different seasons, paintings, sketches, *etc*. What makes this challenging is that the visual content is only similar on the higher scene level, but quite dissimilar on the pixel level. Here we present an approach that works well across different visual domains.

Powered by the availability of Internet-scale image and video collections coupled with greater processing speeds, the last decade has witnessed the rise of data-driven approaches in computer vision, computer graphics, and computational photography. Unlike traditional methods, which employ parametric models to capture visual phenomena, the data-driven approaches use visual data directly, without an explicit intermediate representation. These approaches have shown promising results on a wide range of challenging computer graphics problems, including super-resolution and de-noising [29, 86, 111], texture and video synthesis [66, 239], image analogies [123], automatic colorization [284], scene and video completion [117, 302, 303], photo restoration [55], assembling photo-realistic virtual spaces [40, 145], and even

making CG imagery more realistic [136], to give but a few examples.

The central element common to all the above approaches is searching a large dataset to find visually similar matches to a given query – be it an image patch, a full image, or a spatio-temporal block. However, defining a good visual similarity metric to use for matching can often be surprisingly difficult. Granted, in many situations where the data is reasonably homogeneous (*e.g.*, different patches within the same texture image [66], or different frames within the same video [239]), a simple pixel-wise sum-of-squared-differences (L2) matching works quite well. But what about the cases when the visual content is only similar on the higher scene level, but quite dissimilar on the pixel level? For instance, methods that use scene matching *e.g.*, [55, 117] often need to match images across different illuminations, different seasons, different cameras, *etc.* Likewise, retexturing an image in the style of a painting [66, 123] requires making visual correspondence between two very different domains – photos and paintings. Cross-domain matching is even more critical for applications such as Sketch2Photo [40] and CG2Real [136], which aim to bring domains as different as sketches and CG renderings into correspondence with natural photographs. In all of these cases, pixel-wise matching fares quite poorly, because small perceptual differences can result in arbitrarily large pixel-wise differences. What is needed is a visual metric that can capture the important visual structures that make two images appear similar, yet show robustness to small, unimportant visual details. This is precisely what makes this problem so difficult – the visual similarity algorithm somehow needs to know which visual structures are important for a human observer and which are not.

Currently, the way researchers address this problem is by using various image feature representations (SIFT [183], GIST [202], HoG [54], wavelets, *etc.*) that aim to capture the locally salient (i.e., high gradient and high contrast) parts of the image, while downplaying the rest. Such representations have certainly been very helpful in improving image matching accuracy for a number of applications (*e.g.*, [55, 117, 136, 145]). However, what these features encode are purely local transformations – mapping pixel patches from one feature space into another, independent of the global image content. The problem is that *the same* local feature might be unimportant in one context but crucially important in another. Consider, for example, the painting in Figure 7.2. In local appearance, the brush-strokes on the alleyway on the ground are virtually the same as the brush-strokes on the sky. Yet, the former are clearly much more informative as to the content of the image than the latter and should be given a higher importance when matching (Figure 7.2). To do this algorithmically requires not only considering the local features within the context of a given query image, but also having a good way of estimating the impor-

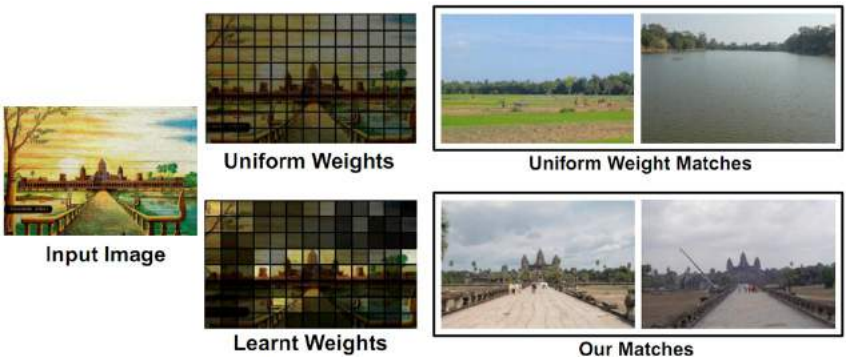


Figure 7.2 – In determining visual similarity, the central question is which visual structures are important for a human observer and which are not. In the painting above, the brush-strokes in the sky are as thick as those on the ground, yet are perceived as less important. In this Chapter, we propose a simple, data-driven learning method for determining which parts of a given image are more informative for visual matching.

tance of each feature with respect to the particular scene's overall visual impression.

What we present in this Chapter is a very simple, yet surprisingly effective approach to visual matching which is particularly well-suited for matching images across different domains. We do not propose any new image descriptors or feature representations. Instead, given an image represented by some features (we will be using the spatially-rigid HoG [54] descriptor for most of this Chapter), the aim is to focus the matching on the features that are the most visually important *for this particular image*. The central idea is the notion of "data-driven uniqueness". We hypothesize, following [22], that the important parts of the image are those that are more unique or rare within the visual world (represented here by a large dataset). For example, in Figure 7.2, the towers of the temple are very unique, whereas the wispy clouds in the sky are quite common. However, since the same local features could represent very different visual content depending of context, unlike [22], our notion of uniqueness is *scene-dependent* i.e., each query image decides what is the best way to weight its constituent parts. Figure 7.2 demonstrates the difference between image matching using a standard uniform feature weighting vs. our uniqueness-based weighting.

We operationalize this data-driven uniqueness by using ideas from machine learning – training a discriminative classifier to discover which parts of an image are most discriminative *in relationship to the rest of the dataset*. This simple approach results in visual matching that is surprisingly versatile and robust. By focusing on the globally salient parts of the image, the approach can be successfully used for generic cross-domain matching without making any domain-specific changes, as

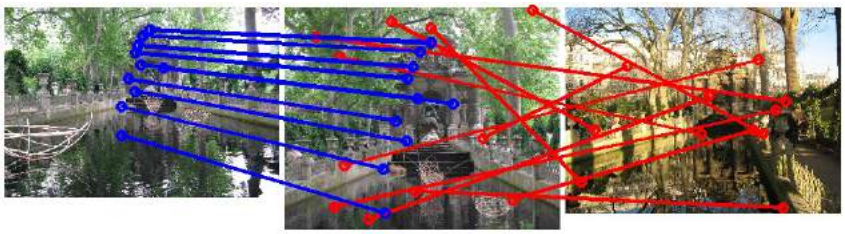


Figure 7.3 – Example of image matching using the SIFT descriptor. While SIFT works very well at matching fine image structure (left), it fails miserably when there is too much local change, such as a change of season (right).

shown on Figure 7.1. The rest of the Chapter is organized as follows: we first give a brief overview of the related work (Section 7.1), then describe our approach in detail (Section 7.2), present an evaluation on several public datasets (Section 7.3), and finally show some of the applications that our algorithm makes possible (Section 7.4).

7.1 Related Work

In general, visual matching approaches can be divided into three broad classes, with different techniques tailored for each:

Exact matching: For finding more images of the exact same physical object (*e.g.*, a Pepsi can) or scene (*e.g.*, another photo of Eiffel Tower under similar illumination), researchers typically use the general bag-of-words paradigm introduced by the Video Google work [261], where a large histogram of quantized local image patches (usually encoded with the SIFT descriptor [183]) is used for image retrieval. This paradigm generally works extremely well (especially for heavily-textured objects), and has led to many successful applications such as Google Goggles. However, these methods usually fail when tasked with finding *similar*, but not identical objects (*e.g.*, try using Google Goggles *app* to find a cup, or a chair). This is because SIFT, being a local descriptor, captures the minute details of a particular object well, but not its overall global properties (as seen in Figure 7.3).

Approximate matching: The task of finding images that are merely "visually similar" to a query image is significantly more difficult and none of the current approaches can claim to be particularly successful. Most focus on employing various image representations that aim to capture the important, salient parts of the image. Some of the popular ones include the GIST [202] descriptor, the Histogram of Gradients (HoG) descriptor [54], various other wavelet- and gradient-based decompositions, or agglomerations, such as the spatial pyramid [165] of visual words.

Also related is the vast field of Content-Based Image Retrieval (CBIR) (see [56] for overview). However, in CBIR the goals are somewhat different: the aim is to retrieve *semantically-relevant* images, even if they do not appear to be visually similar (*e.g.*, a steam-engine would be considered semantically very similar to a bullet train even though visually there is little in common). As a result, most modern CBIR methods combine visual information with textual annotations and user input.

Cross-domain matching: A number of methods exists for matching between particular domains, such as sketches to photographs (*e.g.*, [40, 68]), drawings/paintings to photographs (*e.g.*, [235]), or photos under different illuminants (*e.g.*, [46]), *etc.* However these typically present very domain-specific solutions that do not easily generalize across multiple domains. Of the general solutions, the most ambitious is work by Shechtman and Irani [246], which proposes to describe an image in terms of local self-similarity descriptors that are invariant across visual domains. This work is complementary to ours since it focuses on the design of a cross-domain local descriptor, while we consider relative weighting between the descriptors for a given image, so it might be interesting to combine both.

Within the text retrieval community, the *tf-idf* normalization [14] used in the bag-of-words approaches shares the same goals as our work – trying to re-weight the different features (words in text, or "visual words" in images [261]) based on their relative frequency. The main difference is that in *tf-idf*, each word is re-weighted independently of all the others, whereas our method takes the interactions between all of the features into account.

Most closely related to ours are approaches that try to learn the statistical structure of natural images by using large unlabeled image sets, as a way to define a better visual similarity. In the context of image retrieval, Hoiem et al. [125] estimate the unconditional probability density of images off-line and use it in a Bayesian framework to find close matches; Tieu and Viola [277] use boosting at query-time to discriminatively learn query-specific features. However, these systems require multiple positive query images and/or user guidance, whereas most visual matching tasks that we are interested in need to work automatically and with only a single input image. Fortunately, recent work in visual recognition has shown that it's possible to train a discriminative classifier using a *single positive instance* and a large body of negatives [185, 305], provided that the negatives do not contain any images similar to the positive instance. In this Chapter, we adapt this idea to image retrieval, where one cannot guarantee that the "negative set" will not contain images similar to the query (on the contrary, it most probably will!). What we show is that, surprisingly, this assumption can be relaxed without adversely impacting

the performance.

7.2 Our Approach

The problem considered in this Chapter is the following: how to compute visual similarity between images which would be more consistent with human expectations. One way to attack this is by designing a new, more powerful image representation. However, we believe that existing representations are already sufficiently powerful, but that the main difficulty is in developing the right similarity distance function, which can "pick" which parts of the representation are most important for matching. In our view, there are two requirements for a good visual similarity function: 1) It has to focus on the content of the image (the "what"), rather that the style (the "how") *e.g.*, the images on Figure 7.1 should exhibit high visual similarity despite large pixel-wise differences. 2) It should be *scene-dependent*, that is, each image should have its own unique similarity function that depends on its global content. This is important since the same local feature can represent vastly different visual content, depending on what else is depicted in the image.

7.2.1 Data-driven Uniqueness

The visual similarity function that we propose is based on the idea of "data-driven uniqueness". We hypothesize that what humans find important or salient about an image is somehow related to how unusual or unique it is. If we could re-weight the different elements of an image based on how unique they are, the resulting similarity function would, we argue, answer the requirements of the previous Section. However, estimating "uniqueness" of a visual signal is not at all an easy task, since it requires a very detailed model of our entire visual world, since only then we can know if something is truly unique. Therefore, instead we propose to compute uniqueness in a data-driven way — against a very large dataset of randomly selected images.

The basic idea behind our approach is that the features of an image that exhibit high "uniqueness" will also be the features that would best discriminate this image (the positive sample) against the rest of the data (the negative samples). That is, we are able to map the highly complex question of visual similarity into a fairly standard problem in discriminative learning. Given some suitable way of representing an image as a vector of features, the result of the discriminative learning is a set of weights on these features that provide for the best discrimination. We can then use these same weights to compute visual similarity. Given the learned, query-dependent weight vector $\mathbf{w}_q$, the visual similarity between a query image I_q and any

other image/sub-image I_i can be defined simply as:

$$S(I_q, I_i) = \mathbf{w}_q{}^T \mathbf{x}_i \tag{7.1}$$

where $\mathbf{x}_i$ is I_i's extracted feature vector.

To learn the feature weight vector which best discriminates an image from a large "background" dataset, we employ the linear Support Vector Machine (SVM) framework. We set up the learning problem following [185] which has demonstrated that a linear SVM can generalize even with a single positive example, provided that a very large amount of negative data is available to "constrain the solution". However, whereas in [185] the negatives are guaranteed not to be members of the positive class (that is why they are called negatives), here this is not the case. The "negatives" are just a dataset of images randomly sampled from a large Flickr collection, and there is no guarantee that some of them might not be very similar to the "positive" query image. Interestingly, in practice, this does not seem to hurt the SVM, suggesting that this is yet another new application where the SVM formalism can be successfully applied.

The procedure described above should work with any sufficiently powerful image feature representation. For the majority of our experiments in this Chapter, we have picked the Histogram of Oriented Gradients (HOG) template descriptor [54], due to its good performance for a variety of tasks, its speed, robustness, adaptability to sliding window search, and popularity in the community at the time of this work. We also show how our learning framework can be used with Dense-SIFT (D-SIFT) template descriptor in Section 7.2.4.

To visualize how the SVM captures the notion of data-driven uniqueness, we performed a series of experiments with simple, synthetic data. In the first experiment, we use simple synthetic figures (a combination of circles and rectangles) as visual structures on the query image side. Our negative world consists of just rectangles of multiple sizes and aspect ratios. If everything works right, using the SVM-learned weights should downplay the features (gradients in HoG representation) generated from the rectangle and increase the weights of features generated by the circle, since they are more unique. We use the HoG visualization introduced by [54] which displays the learned weight vector as a gradient distribution image. As Figure 7.4(a) shows, our approach indeed suppresses the gradients generated by the rectangle.

One of the key requirements of our approach is that it should be able to extract visually important regions even when the images are from different visual domains.

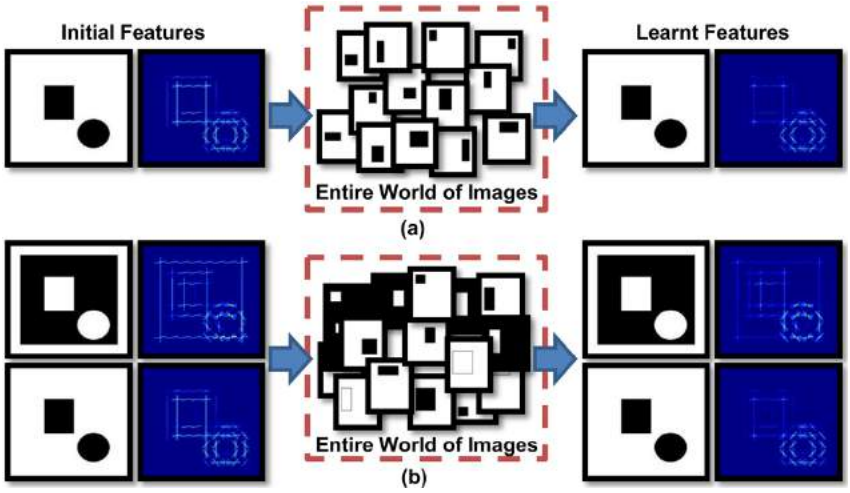


Figure 7.4 – Synthetic example of learning data-driven "uniqueness". In each case, our learned similarity measure boosts the gradients belonging to the circle because they are more unique with respect to a synthetic world of rectangle images.

We consider this case in our next experiment, shown on Figure 7.4(b). Here the set of negatives includes two domains – black-on-white rectangles and white-on-black rectangles. By having the negative set include both domains, our approach should downplay any domain-dependent idiosyncrasies both from the point of view of the query and target domains. Indeed, as Figure 7.4(b) shows, our approach is again able to extract the unique structures corresponding to circles while downplaying the gradients generated due to rectangles, in a domain-independent way.

We can also observe this effect on real images. The Venice bridge painting shown in Figure 7.5 initially has high gradients for building boundaries, the bridge and the boats. However, since similar building boundaries are quite common, they occur a lot in the randomly sampled negative images and hence, their weights are reduced.

7.2.2 Algorithm Description

We set up the learning problem using a single positive and a very large negative set of samples similar to [185]. Each query image (I_q) is represented with a rigid grid-like HoG feature template ($\mathbf{x}_q$). We perform binning with sizing heuristics which attempt to limit the dimensionality of ($\mathbf{x}_q$) to roughly $4-5K$, which amounts to $\sim$ 150 cells for HoG template. To add robustness to small errors due to image misalignment, we create a set of extra positive data-points, $\mathcal{P}$, by applying small transformations (shift, scale and aspect ratio) to the query image I_q, and generating

$\mathbf{x}_i$ for each sample. Therefore, the SVM classifier is learned using I_q and $\mathcal{P}$ as positive samples, and a set containing millions of sub-images $\mathcal{N}$ (extracted from $10,000$ randomly selected Flickr images), as negatives. Learning the weight vector $\mathbf{w}_q$ amounts to minimizing the following convex objective function:

$$L(\mathbf{w}_q) = \sum_{\mathbf{x}_i \in \mathcal{P} \cup I_q} h(\mathbf{w}_q^T \mathbf{x}_i) + \sum_{\mathbf{x}_j \in \mathcal{N}} h(-\mathbf{w}_q^T \mathbf{x}_j) + \lambda ||\mathbf{w}_q||^2 \tag{7.2}$$

We use `LIBSVM` [35] for learning $\mathbf{w}_q$ with a common regularization parameter $\lambda = 100$ and the standard hinge loss function $h(x) = \max(0, 1 - x)$. The hinge-loss allows us to use the hard-negative mining approach [54] to cope with millions of negative windows because the solution only depends on a small set of negative support vectors. In hard-negative mining, one first trains an initial classifier using a small set of training examples, and then uses the trained classifier to search the full training set exhaustively for false positives ('hard examples'). Once sufficient number of hard negatives are found in the training set, one retrains the classifier $\mathbf{w}_q$ using this set of hard examples. We alternate between learning $\mathbf{w}_q$ given a current set of hard-negative examples, and mining additional negative examples using the current $\mathbf{w}_q$ as in [54]. For all experiments in this Chapter, we use 10 iterations of hard-mining procedure; with each iteration requiring more time than the previous one because it becomes harder to find hard-negatives as the classifier improves. Empirically, we found that more than 10 iterations did not provide enough improvement to justify the run-time cost.

The standard sliding window setup [54] is used to evaluate all the sub-windows of each image. For this, the trained classifier is convolved with the HoG feature pyramid at multiple scales for each image in the database. The number of pyramid levels controls the size of possible detected windows in the image. We use simple non-maxima suppression to remove highly-overlapping redundant matches. While the use of sub-window search is expensive, we argue that it is crucial to good image matching for the following reasons. First, it allows us to see millions of negative examples during training from a relatively small number of images ($10,000$). But more importantly, as argued by [125], sub-window search is likely to dramatically increase the number of good matches over the traditional full-image retrieval techniques.

7.2.3 Relationship to Saliency

We found that our notion of data-driven uniqueness works surprisingly well as a proxy for predicting image saliency ("where people look") – a topic of considerable interest to computer graphics. We ran our algorithm on the human gaze dataset from

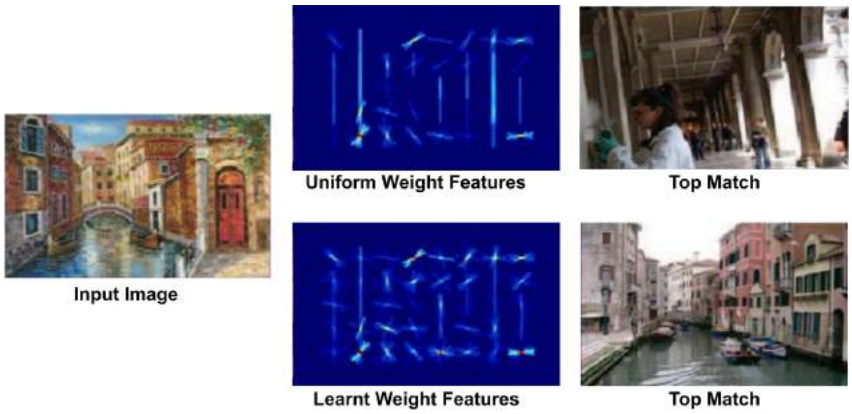


Figure 7.5 – Learning data-driven uniqueness: Our approach down-weighs the gradients on the buildings since they are not as rare as the circular gradients from the bridge.

Judd et al. [141], using a naive mapping from learned HoG weights to predicted pixel saliency by spatially summing these weights followed by normalization. Figure 7.6 compares our saliency prediction against standard saliency methods (summarized in [141]). While our score of 74% (mean area under ROC curve) is below [141] who are the top performers at 78% (without center prior), we beat most classic saliency methods such as Itti *et al.* [131] who only obtained 62%. After incorporating a simple gaussian center prior, our score raises to 81.9%, which is very close to 83.8% of [141].

7.2.4 Other Features

Our framework should be able to work with any rigid grid-like image representation where the template captures feature distribution in some form of histogram of high-enough dimensionality. We performed preliminary experiments using the dense SIFT (D-SIFT) template descriptor (similar to [165]) within our framework for the task of Sketch-to-Image Matching (Section 7.3.2). The query sketch (I_q) was represented with a feature template ($\mathbf{x}_q$) of D-SIFT and sizing heuristics (Section 7.2.2) produced ~ 35 cells for the template (128 dimensions per cell). Figure 7.10 demonstrates the results of these preliminary experiments, where our learning framework improves the performance of D-SIFT baseline (without learning) indicating that our algorithm can be adapted to a different feature representation.

7.3 Experimental Validation on Cross-domain Matching

To demonstrate the effectiveness of our approach, we performed a number of image matching experiments on different image datasets, comparing against the following

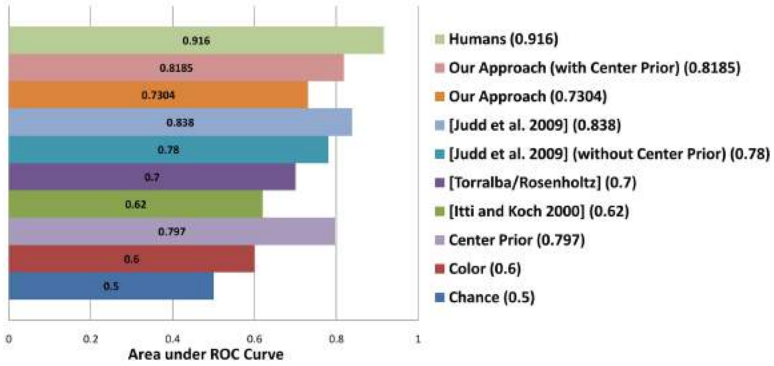


Figure 7.6 – The concept of data-driven uniqueness can also be used as a proxy to predict saliency for an image. Our approach performs better than individual features (such as Itti et al. and Torralba/Rosenholtz, see [141]) and comparable to Judd et al. [141].

popular baseline methods:

Tiny Images: Following [284], we re-size all images to 32x32, stack them into 3072-D vectors, and compare them using Euclidean distance.

GIST: We represent images with the GIST [202] descriptor, and compare them with the Euclidean distance.

BoW: We compute a Bag-of-Words representation for each image using vector-quantized SIFT descriptors [183] and compare the visual word histograms (with *tf-idf* normalization) as in [261].

Spatial Pyramid: For each image, we compute spatial pyramid [165] representation with 3 pyramid levels using Dense-SIFT descriptors of 16x16 pixel patches computed over a grid with spacing of 8 pixels. We used vocabulary of 200 visual words. The descriptors are compared using histogram intersection pyramid matching kernels as described in [165].

Normalized-HoG (N-HoG): We represent each image using the same HoG descriptor as our approach, but instead of learning a query-specific weight vector, we match images directly in a nearest-neighbor fashion. We experimented with different similarity metrics and found a simple normalized HoG (N-HoG) to give the best performance. The N-HoG weight vector is defined as a zero-centered version of the query's HoG features $\mathbf{w}_q = \mathbf{x}_q - mean(\mathbf{x}_q)$. Matching is performed using Equation 7.1, by replacing the learned weight vector with N-HoG weight vector.

In addition, we also compare our algorithm to Google's recently released Search-by-Image feature. It should be noted that the retrieval dataset used by Google is orders of magnitude larger than the tens of thousands of images typically used in

Table 7.1 – Instance retrieval in Holidays dataset + Flickr1M. We report the mean true positive rate from the top-k image matches as a function of increasing dataset size (averaged across a set of 50 Holidays query images)

Top-5				
Dataset Size:	1,490	11,490	101,490	1,001,490
GIST	1.06	1.06	1.06	1.06
Tiny Images	1.06	1.06	1.06	1.06
Spatial Pyramid	34.17	30.63	24.71	19.67
Our Approach	**65.88**	**63.93**	**58.90**	**58.36**

Top-100				
Dataset Size	1,490	11,490	101,490	1,001,490
GIST	19.21	14.17	14.17	14.17
Tiny Images	7.13	5.18	5.18	5.18
Spatial Pyramid	48.88	41.50	34.48	27.92
Our Approach	**68.74**	**68.74**	**66.19**	**61.50**

our datasets, so this comparison is not quite fair. But while Google's algorithm (at the time of this work) shows a reasonable performance in retrieving landmark images with similar illumination, season and viewpoint, it does not seem to adapt well to photos taken under different lighting conditions or photos from different visual domains such as sketches and paintings (see Figure 7.9).

7.3.1 Image-to-Image Matching

While image retrieval is not the goal of this Chapter, the CBIR community has produced a lot of good datasets that we can use for evaluation. Here we consider the instance retrieval setting using the *INRIA Holidays* dataset introduced by Jégou et al. [133] and one million random distractor Flickr images from [117] to evaluate performance. The goal is to measure the quality of the top matching images when the exact instances are present in the retrieval dataset. For evaluation, we follow [133] and measure the quality of rankings as the true positive rate from the list of top $k = 100$ matches as a function of increasing dataset size. Since the average number of true positives is very small for the Holidays dataset, we also perform the evaluation with smaller k. We compare our approach against GIST, Tiny Images and Spatial Pyramid baselines described in Section 7.3 on 50 random Holidays query images and evaluate the top 5 and 100 matches for the same dataset sizes used in [133].

Table 7.1 demonstrates the robustness of our algorithm to adding distractor images – the true positives rate only drops from 69% to 62% when we add 1M distractors (which is of similar order as in [133]), outperforming the state-of-art spatial pyramid matching [165]. It is important to note that even after drastically reducing the ranks under consideration from the top 100 to just the top 5, our rate of true positives drops by only 3% (which attests to the quality of our rankings). For a dataset of one million images and a short-list of 100, [133] return 62% true positives which is only slightly better than our results; however, their algorithm is designed for instance recognition, whereas our approach is applicable to a broad range of cross-domain visual tasks.

Figure 7.7 – Qualitative comparison of our approach against baselines for Sketch-to-Image and Painting-to-Image matching.

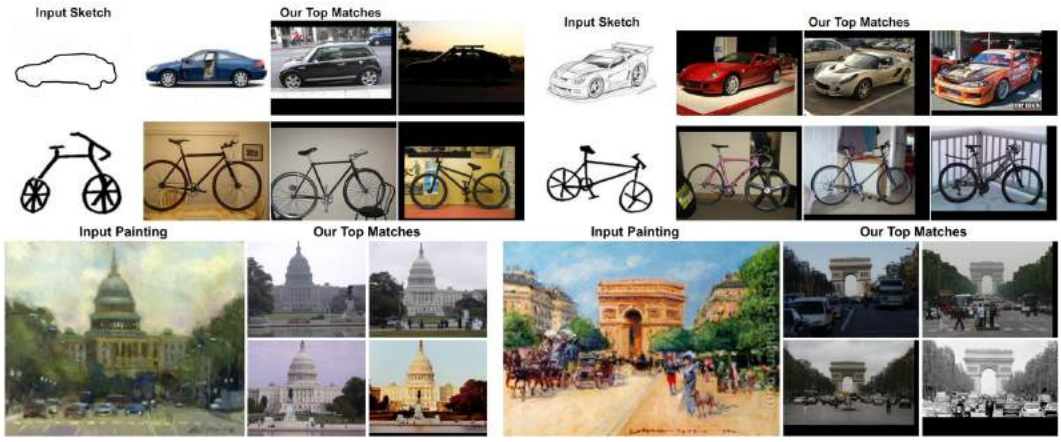


Figure 7.8 – A few more qualitative examples of top-matches for sketch and painting queries.

7.3.2 Sketch-to-Image Matching

Matching sketches to images is a difficult cross-domain visual similarity task. While most current approaches use specialized methods tailored to sketches, here we apply exactly the same procedure as before, without any changes. We collected a dataset of 50 sketches (25 cars and 25 bicycles) to be used as queries (our dataset includes both amateur sketches from the internet as well as freehand sketches collected from non-expert users). The sketches were used to query into the PASCAL VOC dataset [72], which is handy for evaluation since all the car and bicycle instances have been labeled. Figure 7.8(top) show some example queries and the corresponding top retrieval results for our approach and the baselines. It can be seen that our approach not only outperforms all of the baselines, but returns images showing the target object in a very similar pose and viewpoint as the query sketch.

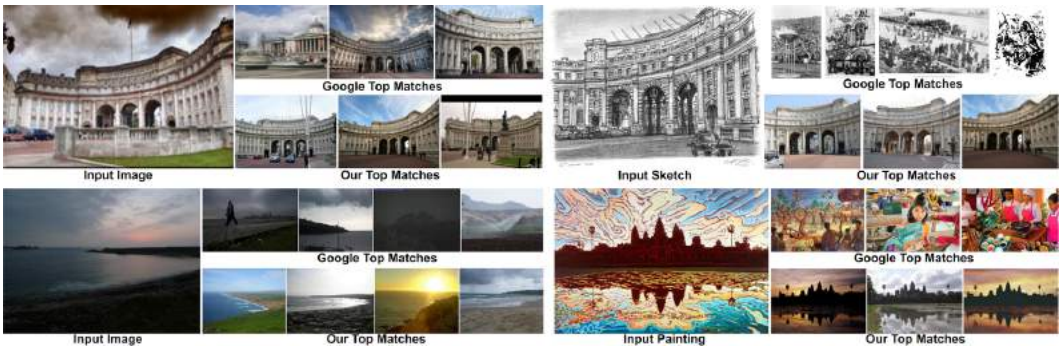


Figure 7.9 – Qualitative comparison of our approach with Google's 'Search–by–Image' feature. While our approach is robust to illumination changes and performs well across different visual domains, Google image search fails completely when the exact matches are not in the database.

For quantitative evaluation, we compared how many car and bicycle images were retrieved in the top-K images for car and bicycle sketches respectively. We used the bounded mean Average Precision (mAP) metric used by [133] [1]. We evaluated the performance of our approach (using HoG and D-SIFT) as a function of dataset size and compare it with the multiple baselines, showing the robustness of our approach to the presence of distractors. For each query, we start with all images of the target class from the dataset, increase the dataset size by adding 1000, 5000 images and finally the entire PASCAL VOC 2007 dataset. Figure 7.10(a) and (b) show mAP as a function of dataset size for cars and bicycles, respectively. For the top 150 matches, we achieve a mAP of 67% for cars and 54% for bicycles (for Learnt-HoG). We also ran our algorithm on the Sketch-Based Image Retrieval (SBIR) Benchmark Dataset [68]. For the top 20 similar images ranked by users, we retrieve 51% of images as top 20 matches, compared 63% using a sketch-specific method of [68]

7.3.3 Painting-to-Image Matching

As another cross-domain image matching evaluation, we measured the performance of our system on matching paintings to images. Retrieving images similar to paintings is an extremely difficult problem because of the presence of strong local gradients due to brush strokes (even in the regions such as sky). For this experiment, we collected a dataset of 50 paintings of outdoor scenes in a diverse set of painting styles geographical locations. The retrieval set was sub-sampled from the 6.4M GPS-tagged Flickr images of [118]. For each query, we created a set of $5,000$ images randomly sampled within a 50 mile radius of each painting's location (to

[1] Maximum recall is bounded by the number of images being retrieved. For example, if we consider only top-150 matches the maximum true positives would be 150 images

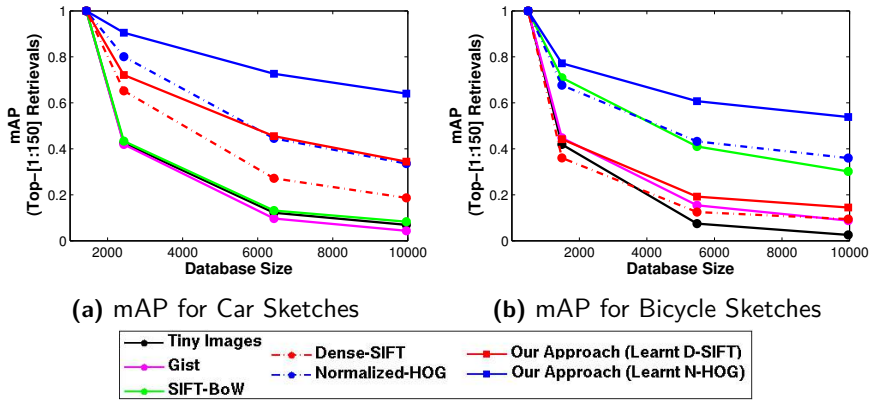


Figure 7.10 – Sketch-to-Image evaluation. We match car/bicycle sketches to sub-images in the PASCAL VOC 2007 dataset and measure performance as the number of distractors increases.

make sure to catch the most meaningful distractors), and 5,000 random images. Qualitative examples can be seen in Figure 7.7.

7.4 Applications of Data-driven Similarity

Our data-driven visual similarity measure can be used to improve many existing matching-based application, as well as facilitate new ones. First, we briefly discuss a few direct applications of improved visual similarity, such as Scene Completion, Internet Re-photogrpahy and Painting2GPS in Section A1. Next, we will discuss how a robust visual similarity opens the door to organising unordered, large visual data. In particular, one can construct a *visual memex* graph ([184]), whose nodes are images/sub-images, and edges are various types of associations, such as visual similarity, context, *etc.* We will show how this graph facilitates exploring a large collection of unordered visual data (Section A2); and even enables discovering object instances and their segmentation masks from weakly-supervised and noisy web data (Section A3).

A1 Improved Image Matching and its Applications

Better Scene Matching for Scene Completion

Data-driven Scene Completion has been introduced by [117]. However, their scene matching approach (using the GIST descriptor) is not always able to find the best matches automatically. Their solution is to present the user with the top 20 matches and let him find the best one to be used for completion. Here we propose to use our approach to automate scene completion, removing the user from the loop. To evaluate the approach, we used the 78 query images from the scene completion

Figure 7.11 – Qualitative examples of scene completion using our approach and [117].

Figure 7.12 – Internet Re-photography. Given an old photograph, we harness the power of large Internet datasets to find visually similar images. For each query we show the top 4 matches, and manually select one of the top matches and create a manual image alignment.

test set [117] along with the top 160 results retrieved by them. We use our algorithm to re-rank these 160 images and evaluate both the quality of scene matches and scene completions against [117].

Figure 7.11 shows a qualitative result for the top match using our approach as compared to the top match from the GIST+ features used by [117]. To compute quantitative results, we performed two small user studies. In the first study, for each query image participants were presented with the best scene match using our approach, [117] and tiny-images [284]. Participants were asked to select the closest scene match out of the three options. In the second study, participants were presented with automatically completed scenes using the top matches for all three algorithms, and were asked to select the most convincing/compelling completion. The order of presentation of queries as well as the order of the three options were randomized. Overall, for the first task of scene matching, the participants preferred our approach in 51.4% cases as opposed 27.6% for [117] and 21% for Tiny-Images. For the task of automatic scene completion, our approach was found to be more convincing in 47.3% cases as compared to 27.5% for [117] and 25.2% for Tiny-Images. The standard-deviation of user responses for most of the queries were surprisingly low.

Internet Re-photography

We were inspired by the recent work on computational re-photography [13], which allows photographers to take modern photos that match a given historical photograph. However, the approach is quite time-consuming, requiring the photog-

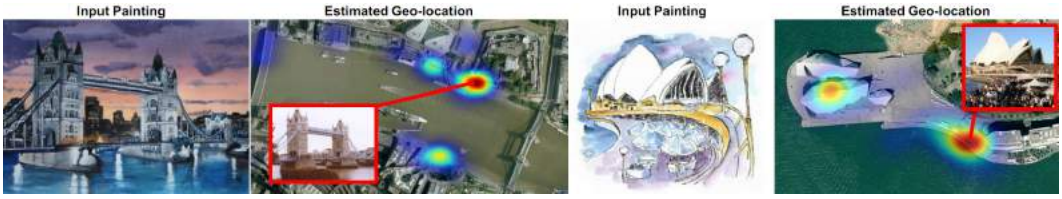


Figure 7.13 – Painting2GPS Qualitative Examples. In these two painting examples (Tower Bridge in London and the Sydney Opera House), we display estimated GPS location of the painting as a density map overlaid onto Google-map, and the top matching image.

Figure 7.14 – Typical failure cases. (Left): relatively small dataset size, compared to Google. (Right): too much clutter in the query image.

rapher to go "on location" to rephotograph a particular scene. What if, instead of rephotographing ourselves, we could simply find the right modern photograph online? This seemed like a perfect case for our cross-domain visual matching, since old and new photographs look quite different and would not be matched well by existing approaches.

We again use the 6.4M geo-tagged Flickr images of [117], and given an old photograph as a query, we use our method to find its top matches from a pre-filtered set of $5,000$ images closest to the old photograph's location (usually at least the city or region is known). Once we have an ordered set of image matches, the user can choose one of the top five matches to generate the best old/new collage. Re-photography examples can be seen in Figure 7.12.

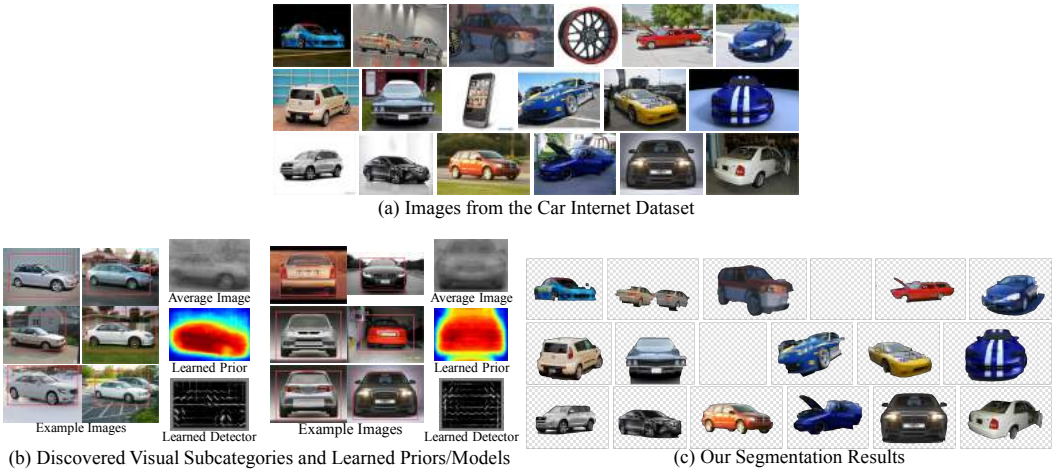
Painting2GPS

Wouldn't it be useful if one could automatically determine from which location a particular painting was painted? Matching paintings to real photos from a large GPS-tagged collection allows us to estimate the GPS coordinates of the input painting, similar to the approach of [118]. We call this application `painting2GPS`. We use painting-to-image matching as described in Section 7.3.3, and then find the GPS distribution using the algorithm in [118]. Qualitative painting2GPS examples overlayed onto Google-map can be seen in Figure 7.13.

Figure 7.15 – Visual Scene Exploration. (Top): Given an input image, we show the top matches, aligned by the retrieved sub-window. The last image shows the average of top 20 matches. (Bottom): A visualization of the memex-graph tour through the photos of the Medici Fountain.

A2 Exploring Large, Un-ordered Visual Data

Having a robust visual similarity opens the door to interesting ways of exploring and reasoning about large visual data. In particular, one can construct a *visual memex* graph (using the terminology from [184]), whose nodes are images/sub-images, and edges are various types of associations, such as visual similarity, context, *etc.* By visually browsing this memex graph, one can explore the dataset in a way that makes explicit the ways in which the data is interconnected. Such graph browsing visualizations have been proposed for several types of visual data, such as photos of a 3D scene [262], large collections of outdoor scenes [145], and faces [150]. Here we show how our visual similarity can be used to align photos of a scene and construct a movie. Given a set of 200 images automatically downloaded from Flickr using keyword search (*e.g.*, "Medici Fountain Paris"), we compute an all-to-all matrix of visual similarities that represents our visual memex graph. Note that because we are using scanning window matching on the detection side, a zoomed-in scene detail can still match to a wide-angle shot as seen on Figure 7.15 (top). Other side-information can also be added to the graph, such as the relative zoom factor, or similarity in season and illumination (computed from photo time-stamps). One can now interactively browse through the graph, or create a visual memex movie showing a particular path from the data, as shown on Figure 7.15 (bottom), and in supplementary videos available at the project page.



(a) Images from the Car Internet Dataset

(b) Discovered Visual Subcategories and Learned Priors/Models

(c) Our Segmentation Results

Figure 7.16 – We propose an approach to discover objects and perform segmentation in noisy Internet images (a). Our approach builds upon the advances in discriminative object detection to discover visual subcategories and build top-down priors for segmentation (b). These top-down priors, discriminative detectors and bottom-up cues are finally combined to obtain segmentations (c).

A3 Object Discovery and Segmentation

In this Section, we focus on automatically discovering object instance and their segmentation masks given a large collection of noisy Internet images of some object class (say "car"). We will only present the details directly relevant to the application of data-driven visual similarity (other details of this work can be found in the original manuscript [42]). The central idea behind our method is to learn top-down priors and use these priors to perform joint segmentation. Approaches such as Class-cut [3], Collect-cut [168] and [156] develop top-down priors based on semantic classes: *i.e.,* they build appearance models for semantic classes such as cars, airplanes *etc.* and use them in a graph-based optimization formulation. However, high intra-class variations within a semantic class leads to weak priors and these priors fail to significantly improve performance.

Interestingly, similar problems have been faced by object detection community as well. Due to high-intra class and pose variations, most approaches find it extremely difficult to learn a single (object vs. background) classifier that can generalize. Therefore, to relieve the classifier of this task, many approaches have considered reorganizing the data into clusters (called visual subcategories) and train a different classifier for each cluster. While some approaches have clustered the data using extra ground-truth annotations (*e.g.*, viewpoint, which may not be available for many large datasets) [102], visual similarity [41, 60] and heuristics (*e.g.*, aspect-ratio [79], that fail to generalize to a large number of categories). Motivated by these recent ap-

proaches, we propose to automatically discover discriminative visual sub-categories and learn top-down segmentation priors based on these sub-categories.

In particular, we will use the notion of **data-driven visual similarity** to build a graph between images or image patches to which will enable us to discover visual subcategories which have well-aligned instances of objects. These visual subcategories are then exploited to build strong top-down priors which are combined with image-based likelihoods to perform segmentation on multiple images simultaneously. Figure 7.16 shows how our approach can extract aligned visual subcategories and develop strong priors for segmentation. Our experimental results indicate that generating priors via visual subcategories indeed leads to state-of-the-art performance in joint segmentation of an object class on standard datasets [231].

This work is closely related to co-segmentation, where the task is to simultaneously segment visually similar objects from multiple images at the same time [15, 139, 152, 229, 231, 290]. Most of these approaches assume that all images have very similar objects with distinct backgrounds, and they try to learn a common appearance model to segment these images. However, the biggest drawback with these approaches is that they are either susceptible to noisy data or assume that an object of interest is present in every image of the dataset. The most relevant related work to our approach is the work of [231], which proposes to use a pixel correspondence-based method for object discovery. They model the sparsity and saliency properties of the common object in images, and construct a large-scale graphical model to jointly infer an object mask for each image. Instead of using pairwise similarities, our approach builds upon recent success of discriminative models and exploits visual subcategories. Our discriminative machinery allows us to localize the object in the scene and the strong segmentation priors help us achieve state-of-the-art performance on the benchmark dataset. Finally, we believe our approach is more scalable than other co-segmentation approaches (including [231]) since we never attempt to solve a global joint segmentation problem, but instead only perform joint segmentation on subsets of the data.

Approach

Our goal is to extract objects and their segments from large, noisy image collections in an unsupervised[2] manner. We assume that the collection is obtained as a query result from search engines, photo albums, *etc.* and therefore, a majority of these images contain the object of interest. However, we still want to reject the

[2]The image collection itself is weakly-supervised. The term 'unsupervised' is used to refer to no human-input for segmentation

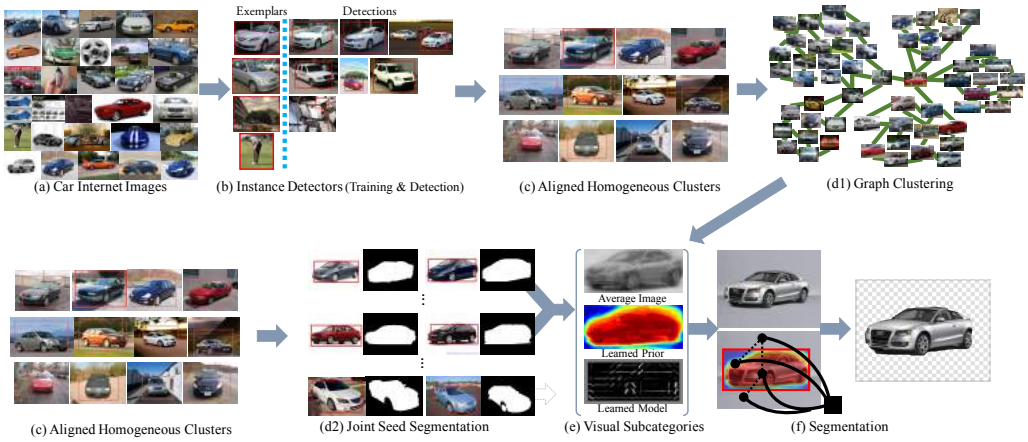


Figure 7.17 – Overview of our approach.

images which are noisy and do not have the object instance. While one can use approaches like graph-cut with center prior to discover the object segments, such an approach fails due to the weak center prior in case of Internet images. What we need is some top-down information, which can be obtained by jointly segmenting the whole collection. Most approaches build class-based appearance models from the entire image collection to guide the segmentation of individual instances. However, in this work, we argue that due to high intra-class and pose variations such priors are still weak and do not improve the results significantly. Instead, we build priors based on visual subcategories where each subcategory corresponds to a 'visually homogeneous' cluster in the data (low intra-class variations) [60]. For example, for an airplane, some of the visual subcategories could be commercial plane in front view, passenger plane in side view, fighter plane in front view *etc.* But how does one seed segmentations for these visual subcategories before learning segmentation priors?

In this work, instead of directly discovering disjoint visual subcategories, we first cluster the visual data into overlapping and redundant clusters (an instance can belong to one or more clusters). The notion of **data-driven visual similarity** is used to build these overlapping clusters. In particular, we use faster and improved version [114] of ideas presented previously in this Section to train instance based detectors and then using these detectors to find similar instances in the training data. Because we use sliding-window detectors, our clusters have nicely aligned visual instances. Exploiting the fact that images in these clusters are well aligned, we run a joint co-segmentation algorithm on each cluster by introducing an extra constraint that pixels in the same location should have similar foreground-background labels. Introducing this extra constraint in conjunction with high-quality clusters leads to

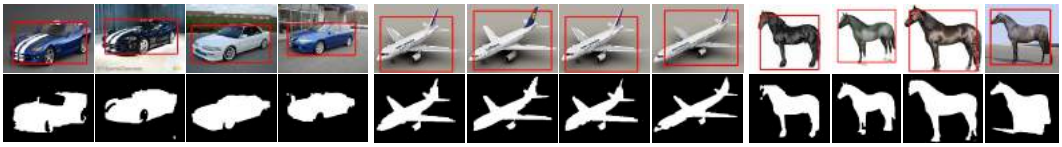


Figure 7.18 – (Top) Examples of strongly aligned and visually coherent clusters that we discovered. (Bottom) We also show the result of our modified co-segmentation approach on these clusters.

clean segmentation labels for the images.

Our clusters are tight (low recall, high precision) with very few instances, and therefore some of the clusters are noisy, which capture the repetition in the noisy images. For example, 5 motorbikes in the car collection group together to form a cluster. To clean-up the noisy clusters, we merge these overlapping and redundant clusters to form visual sub-categories. The subcategories belonging to the underlying categories find enough repetition in the data that they can be merged together. On the other hand, the noisy clusters fail to cluster together and are dropped. Once we have these large subcategories, we pool in the segmentation results from the previous step to create top-down segmentation priors. We also train a discriminative Latent-SVM detector [79] for each of the cluster. These trained detectors are then used to detect instances of object across all the images. We also generate a segmentation mask for each detection by simply transferring the average segmentation for each visual subcategory. Finally, these transferred masks are used as the top-down prior and a graph-cut algorithm is applied to extract the final segment for each image. The outline of our approach is shown in Figure 7.17.

Discovering Aligned and Homogeneous Clusters

To build segmentation priors, we first need to initialize and segment a few images in the collection. We propose to discover strongly coherent and visually aligned clusters (high precision, low recall). Once we have visually homogeneous and aligned clusters, we propose to run a co-segmentation approach with strong co-location constraints and obtain seed segments in the dataset.

To discover visually coherent and aligned clusters, we first build a graph using each image as a node, and connecting it with other image patches using a visual similarity metric (similar to the ones presented previous applications). Specifically, we train an eLDA exemplar detector [114] (which captures the notion of data-driven visual similarity by training a detector using natural world statistics) based on Color-HOG (CHOG) features [244]. Once we have an eLDA detector for each cropped image, we use this detector to detect similar objects on all the images in the collection and select the top k detections with highest scores. Since CHOG feature focuses on

shapes/contours, the resulting clusters are well aligned, which serves as the basis for the following joint segmentation and subcategory discovery step. Note that since we develop a cluster for each image and some images are noisy (do not contain any objects), some of the clusters tend to be noisy as well. Figure 7.18(top) shows some examples of the well aligned clusters extracted using the above approach.

Generating Seed Segmentations

The discovered visually coherent and overlapping clusters in the previous step are aligned due to sliding window search, and they are aligned up to the level of a CHOG grid cell. We can use this strong alignment to constrain the co-segmentation problem and jointly segment the foreground in all images, in the same cluster, using a graph-cut based approach. Notice that objects can occur in different environments and have backgrounds with various conditions. The benefits of segmenting all the images at once is that some instances can be more easily segmented out (*e.g.*, product images with clean, uniformly colored background), and those segmentations can help in segmenting the hard images (*e.g.*, images taken with a low-resolution camera, real-world images with multiple objects, overlaps and occlusions). The mathematical formulation of seed segmentation generation can be found in [42]. Figure 7.18(bottom) shows some examples of segmentations obtained for the visually coherent clusters.

From Clusters to Visual Subcategories

In the last step, we used a standard co-segmentation approach to segment the object of interest in strongly aligned clusters. While one can pool-in results from all such clusters to compute final segmentations, this naïve approach will not work because internet data is noisy, especially for images returned by search engines which are still mainly dependent on text-based information retrieval. Therefore, some clusters still correspond to noise (*e.g.*, a bike cluster is created from car data). But more importantly, our initial clustering operates in the high-precision, low-recall regime to generate very coherent clusters. In this regime, the clustering is strongly discriminative and focuses on using only part of the data. Therefore, as a next step we create larger clusters which will increase the recall of bounding boxes. Specifically, we merge these aligned clusters and create visual subcategories which are still visually homogeneous but avoid over fitting and allow better recall. This clustering step also helps to get rid of noise in the data as the smaller and less consistent (noisy) clusters find it difficult to group and create visual subcategories. One way to merge clusters would be based on similarity of cluster members. However, in our case, we represent each cluster in terms of the detector and create the signature of the

detector based on the detector score on randomly sampled patches (following [191]). More details on using detection signatures for clustering can be found in [42, 191]. Finally, we learn a LSVM detector for each merged cluster.

Generating Segmentations from Subcategories

In the final step, we bring together the discriminative visual subcategory detectors, the top-down segmentation priors learned for each subcategory and the local image evidence to create final segmentation per image. Given the discovered visual subcategories we learn a LSVM detector without the parts [79] for each subcategory. We use these trained detectors to detect objects throughout the dataset. Finally, we transfer the pooled segmentation mask for each subcategory to initialize the grab-cut algorithm. The result of the grab-cut algorithm is the final segmentation of each instance. The experiments demonstrate that this simple combination is quite powerful and leads to state-of-the-art results on the challenging Internet Dataset [231].

Experimental Results

We now present experimental results to demonstrate the effectiveness of our approach on both standard datasets and Internet scale data. Traditional co-segmentation datasets like [15] are too small and clean; however our algorithm is specifically suited for large datasets (1000 images or more per-class). Therefore, we use the new challenging Internet dataset [231] for evaluation. This dataset consists of images automatically downloaded from the Internet with query expansion. It has thousands of noisy images for three categories: airplane, horse, and car, with large variations on pose, scale, view angle, *etc.* Human labeled segmentation masks are also provided for quantitative evaluation.

Figure 7.20 shows some qualitative results. Notice how our approach can extract nice segments even from cluttered scenarios such as cars. Also, our approach can separately detect multiple instances of the categories in the same image. The last row in each category shows some failure cases which can be attributed to weird poses and rotations that are not frequent in the dataset.

Quantitative Evaluation

We now quantitatively evaluate the performance of our approach and compare against the algorithm of [231]. Note that most co-segmentation algorithms cannot scale to extremely large datasets and hence we focus on comparing against [231].

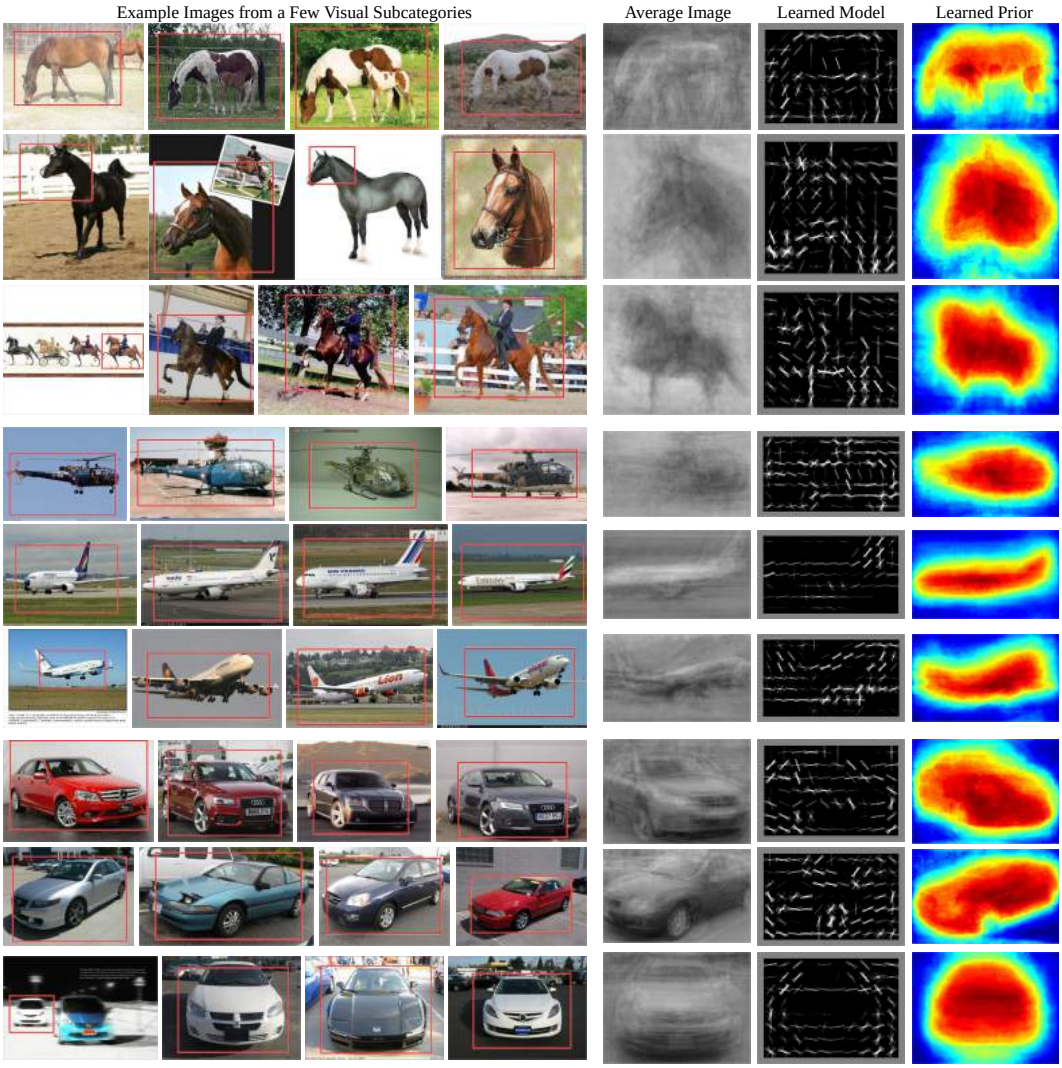


Figure 7.19 – Examples of visual subcategories obtained after merging clusters. We show few instances, the average images, learned Latent SVM model and the segmentation prior for each subcategory.

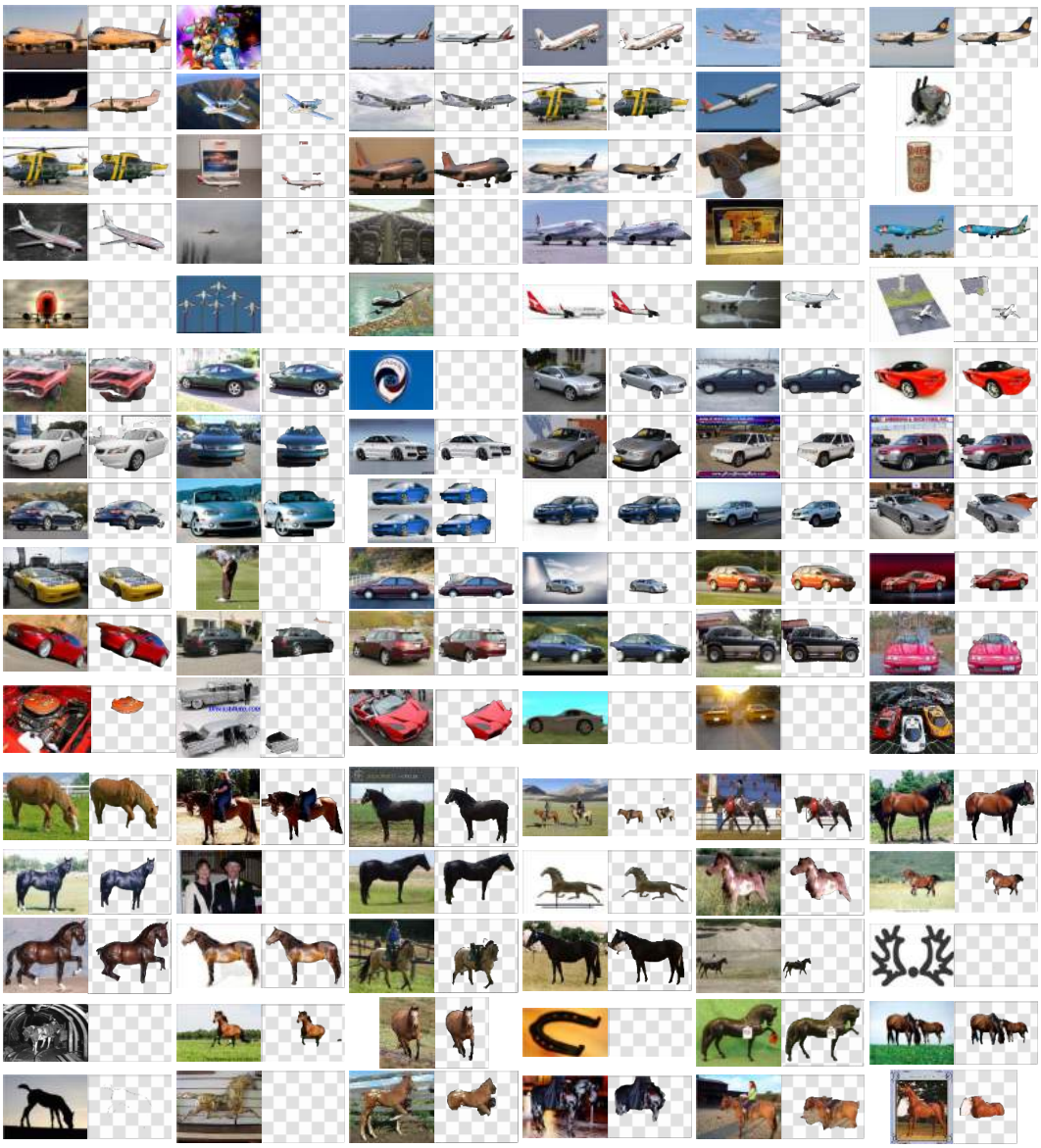


Figure 7.20 – Qualitative results on discovering objects and their segments from noisy Internet images. We show results on three categories: car, horse, and airplane. The last row in each result shows some failure cases.

Table 7.2 – Evaluation on the entire Internet Dataset [231]

	Car		Horse		Airplane	
	P	J	P	J	P	J
[231]	83.38	63.36	83.69	53.89	86.14	55.62
Ours	**87.09**	64.67	**89.00**	**57.58**	**90.24**	**59.97**

Table 7.3 – Evaluation on the 100 images per class subset of Internet Dataset [231]

	Car		Horse		Airplane	
	P	J	P	J	P	J
[139]	58.70	37.15	63.84	30.16	49.25	15.36
[140]	59.20	35.15	64.22	29.53	47.48	11.72
[152]	68.85	0.04	75.12	6.43	80.20	7.90
[231]	85.38	64.42	82.81	**51.65**	88.04	**55.81**
Ours	**87.65**	**64.86**	**86.16**	33.39	**90.25**	40.33

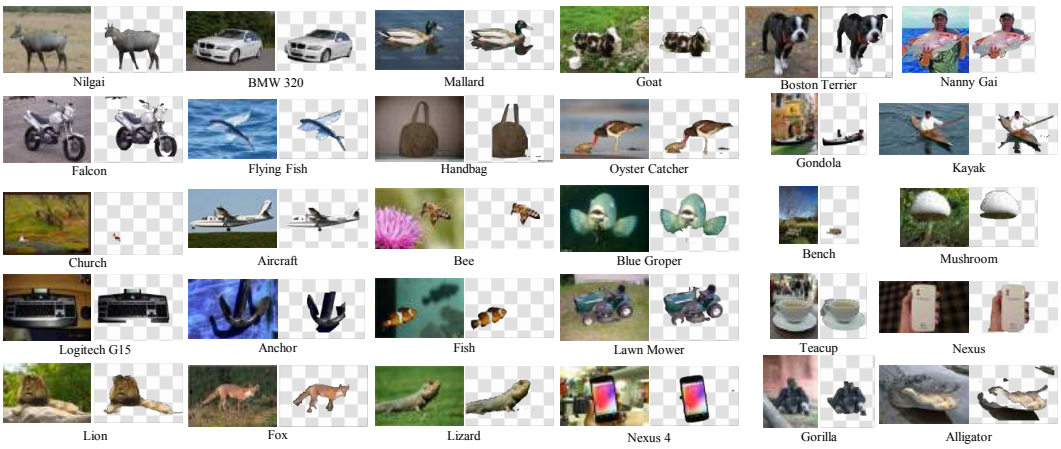


Figure 7.21 – Qualitative results on discovering objects and their segments in NEIL [41]. The last column shows some failure cases.

For our evaluation metric, we use Precision (P) (the average number of pixels correctly labeled) and Jaccard similarity (J) (average intersection-over-union for the foreground objects). Table 7.2 shows the result on the entire dataset. Our algorithm substantially outperforms the state-of-the-art algorithm [231] on segmenting Internet images.

Our algorithm hinges upon the large dataset size and therefore, as our final experiment, we want to observe the behavior of our experiment as the amount of data decreases. We would like a graceful degradation in this case. For this we use a subset of 100 images used in [231]. This experiment also allows us to compare against the other co-segmentation approaches. Table 7.3 summarizes the performance comparison. Our algorithm shows competitive results in terms of precision. This indicates that our algorithm not only works best with a large amount of data, but also degrades gracefully. We also outperform most existing approaches for co-segmentation both in terms of Precision and Jaccard measurement. Finally, we would like to point out that while our approach improves the performance with

increasing size of data, [231] shows almost no improvement with dataset size. This suggests that the quantitative performance of our approach is more scalable with respect to the dataset size.

Chapter 8

Structure-constrained Semi-Supervised Learning

A Case Study

> Creativity is allowing yourself to make mistakes.
> Art is knowing which ones to keep.
>
> Scott Adams

How do we exploit the sea of visual data available online? Most supervised computer vision approaches (including the ones discussed in Part I) are still impeded by their dependence on manual labeling, which, for rapidly growing datasets, requires an incredible amount of manpower. The popularity of Amazon Mechanical Turk and other online collaborative annotation efforts [234, 293] has eased the process of gathering more labeled data, but it is still unclear whether such an approach can scale up with the available data. This is exacerbated by the heavy-tailed distribution of objects in the natural world [284]: a large number of objects occur so sparsely that it would require significant amount of labeling to build reliable models. In addition, human-labeling has a practical limitation in that it suffers from semantic and functional bias. For example, humans might label an image of "Christ"/"Cross" as "Church" due to high-level semantic connections between the two concepts.

An alternative way to exploit a large amount of unlabeled data is semi-supervised learning (SSL). A classic example is the "bootstrapping" method: start with a small number of labeled examples, train initial models using those examples, then use the initial models to label the unlabeled data. The model is retrained using the confident self-labeled examples in addition to original examples. However, most semi-supervised approaches, including bootstrapping, have often exhibit low and

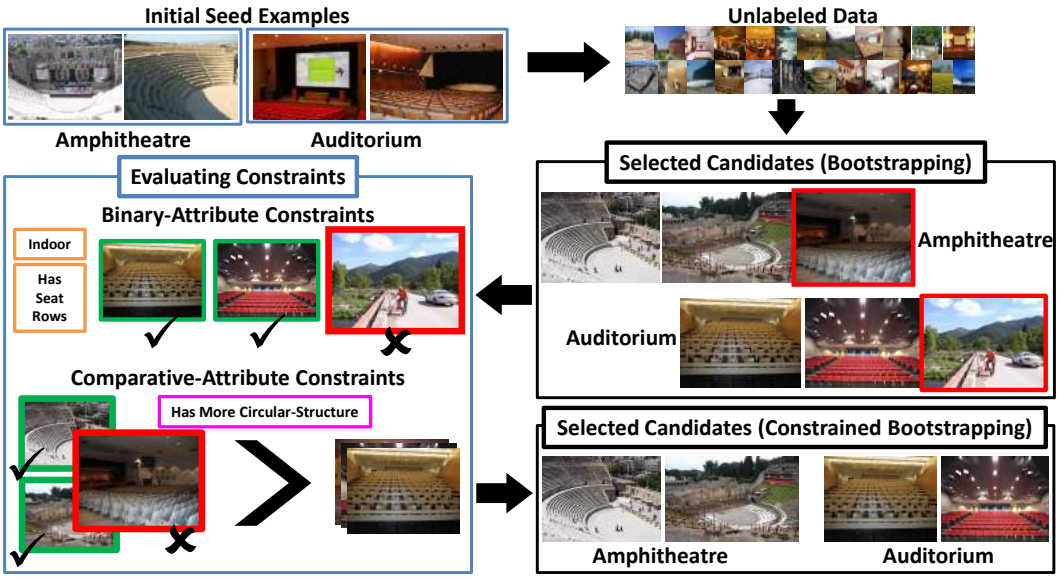


Figure 8.1 – Standard Bootstrapping *vs.* Constrained Bootstrapping: We propose to learn multiple classifiers (auditorium and amphitheater) jointly by exploiting similarities (*e.g.*, both are indoor and have seating) and dissimilarities (e.g, amphitheater has more circular structures than auditorium) between the two. We show that joint learning constrains the SSL, thus avoiding semantic drift.

unacceptable accuracy because the limited number of initially labeled examples are insufficient to constrain the learning process. This often leads to the well known problem of "semantic drift" [51], where newly added examples tend to stray away from the original meaning of the concept. The problem of semantic drift is more evident in the field of visual categorization because intra-class variation is often greater than inter-class variation. For example, "electric trains" resemble "buses" more than "steam engines" and many "auditoriums" appear very similar to "amphitheaters".

This Chapter shows that we can avoid semantic drift and significantly improve performance of bootstrapping approach by imposing additional constraints. We build upon the recent work in information extraction [30], and propose a novel semi-supervised image classification framework where instead of each classifier selecting its own set of images, we jointly select images for each classifier by enforcing different types of constraints. We show that coupling scene categories via attributes and comparative attributes[1] provides us with the constraints necessary for a robust bootstrapping framework. For example, consider the case shown in Figure 8.1. In the case of the naïve bootstrapping approach, the initial "amphitheater" and

[1]Comparative attributes are special forms of attributes that are used to compare and express relationships between two nouns. For example, "banquet halls" are **bigger** than "bedrooms"; "amphitheaters" are **more circular** than "auditoriums".

"auditorium" classifiers select self-labeled images independently which leads to incorrect instances being selected (outlined in red). However, if we couple the scene categories and jointly label all the images in the dataset, then we can use the auditorium images to clean the amphitheater images, since the latter should have more circular structures compared to the former. We demonstrate that the joint labeling indeed makes the data selection robust and improves the performance significantly. While we only explore the application of these new constraints to bootstrapping approaches, we believe they are generic and can be applied to other semi-supervised approaches as well.

Contributions. We present a framework for coupled bootstrap learning and explore its application to the field of image classification. The input to our system is an ontology which defines the set of target categories to be learned, the relationships between those categories and a handful of initial labeled examples. We show that given these initial labeled examples and millions of unlabeled images, our approach can obtain much higher accuracy by coupling the labeling of all the images using multiple classifiers. The key contributions of this Chapter are: (a) a semi-supervised image classification framework which jointly learns multiple classifiers, (b) demonstrating that sharing information across categories via attributes [76, 163] is crucial to constrain the semi-supervised learning problem, (c) extending the notion of sharing across categories and showing that information sharing can also be achieved by capturing dissimilarities between categories. Here we build upon the recent work on relative attributes [207] and comparative adjectives [107] to capture differences across categories. As opposed to image-level labeling, specifying attribute and comparative attribute relationships are significantly cheaper as they scale with number of categories (not with number of images). Note that the attributes need not be semantic at all [221]. However, the general benefit of using semantic attributes is that they are human-communicable [215] and we can obtain them automatically using other sources of data such as text [31].

8.1 Related Work

During the past decade, computer vision has seen some major successes due to the increasing amount of data on the web. Therefore, leveraging more and more data from the web has received considerable interest in the community. While using big data is a promising direction, it is still unclear how we should exploit such a large amount of data. There is a spectrum of approaches based on the amount of human labeling required to use this data. On one end of the spectrum are supervised approaches that use as much hand-labeled data as possible. These

approaches have focused on using the power of crowds to generate hand-labeled training data [234, 293]. Recent works have also focused on active learning [138, 148, 215, 217, 255, 291], to minimize human effort by selecting label requests that are most informative. On the other end of the spectrum are completely unsupervised approaches, which use no human supervision and rely on clustering techniques to discover image categories [146, 233].

In this work, we explore the intermediate range of the spectrum; the domain of semi-supervised approaches. Semi-supervised learning (SSL) techniques use a small amount of labeled data in conjunction with a large amount of unlabeled data to learn reliable and robust visual models. There is a large literature on semi-supervised techniques. For brevity, we only discuss closely related works and refer the reader to recent survey on the subject [327]. The most commonly used semi-supervised approach is the "bootstrapping" method, also known as self-training. However, bootstrapping typically suffers from semantic drift [51] – that is, after many iterations, errors in labeling tend to accumulate. To avoid semantic drift, researchers have focused on several approaches such as using multi-class classifiers [224] or using co-training methods to exploit conditionally independent feature spaces [20]. Another alternative is to use graph-based methods, such as the graph Laplacian, for SSL. These methods capture the manifold structure of the data and encourage similar points to share labels [65]. In computer vision, efficient graph based methods have been used for labeling of images as well [81]. The biggest limitation with graph based approaches is the need for similarity measures that create graphs with no inter-class connections. In the visual world, it is very difficult to learn such a good visual similarity metric. Often, intra-class variations are larger than inter-class variations, which make pair-wise similarity based methods of little utility. To overcome this difficulty, researchers have focused on text based features for better estimation of visual similarity [105].

In this work, we argue that there exists a richer set of constraints in the visual world that can help us constrain the SSL-based approaches. We present an approach to combine a variety of such constraints in a standard bootstrapping framework. Our work is inspired by works from the textual domain [30] that try to couple learning of category and relation classifiers. However, in our case, we build upon recent advances in the field of visual attributes [76, 163] and comparative attributes [107, 207] and propose a set of domain-specific visual constraints to model the coupling between scene categories.

8.2 Constrained Bootstrapping Framework

Our goal is to use the initial set of labeled seed examples ($\mathcal{L}$) and large unlabeled dataset ($\mathcal{U}$) to learn robust image classifiers. Our method iteratively trains classifiers in a self-supervised manner. It starts by training classifiers using a small amount of labeled data and then uses these classifiers to label unlabeled data. The most confident new labels are "promoted" and added to the pool of data used to train the models, and the process repeats. The key difference from the standard bootstrapping approach is the set of constraints that restrict which data points are promoted to the pool of labeled data.

In this work, we focus on learning scene classifiers for image classification. We represent these classifiers as functions ($f : X \rightarrow Y$) which, given input image features x, predict some label y. Instead of learning these classifiers separately, we propose an approach which learns these classifiers jointly. Our central contribution is the formulation of constraints in the domain of image classification. Specifically, we exploit the recently proposed attribute-based approaches [76, 163] to provide another view of the same data and enforce multi-view agreement constraints. We also build upon the recent framework of comparative adjectives [107] to formulate pair-wise labeling constraints. Finally, we use *introspection* to perform an additional step of self-refinement to weed out false positives included in the training set. We describe all the constraints below.

8.2.1 Output Constraint: Mutual Exclusion (ME)

Classification of a single datapoint by multiple scene classifiers is not an independent process. We can use this knowledge to enforce certain constraints on the functions learned for the classifiers. Mathematically, if we know some constraint on output values of two classifiers $f_1 : X \rightarrow Y_1$ and $f_2 : X \rightarrow Y_2$ for an input x, then we can require the learned functions to satisfy these. One such output constraint is the **mutual exclusion constraint** (ME). In mutual exclusion, positive classification by one classifier immediately implies negative classification for the other classifiers. For example, an image classified as "restaurant" can be used as an negative example for "barn", "bridge" *etc.*

Current semi-supervised approaches enforce mutual exclusion by learning a multi-class classifier where a positive example of one class is automatically treated as a negative example for all other classes. However, the multi-class classifier formulation is too rigid for a learning algorithm. Consider, for example, "banquet hall" and "restaurant", which are very similar and likely to be confused by the classifier. For such classes, the initial classifier learned from a few seed examples is not reli-

able enough; hence, adding the mutual exclusion constraint causes the classifier to overfit.

We propose an adaptive mutual exclusion constraint. The basic idea is that during initial iterations, we do not want to enforce mutual exclusion between similar classes (y_1 and y_2), since this is likely to confuse the classifier. Therefore, we relax the ME constraint for similar classes – a candidate added to the pool of one is not used as a negative example for the other. However, after a few iterations, we adapt our mutual exclusion constraints and enforce these constraints across similar classes as well.

8.2.2 Sharing Commonalities: Binary-Attribute Constraint (BA)

For the second constraint, we exploit the commonalities shared by scene categories. For example, both "amphitheaters" and "auditoriums" have large seating capacity; "bedrooms" and "conference rooms" are indoors and man-made. We propose to model these shared properties via attributes [76, 163]. Modeling visual attributes helps us enforce a constraint that the promoted instances must also share these properties.

Formally, we model this constraint similar to multi-view settings [20]. For a function $f : X \rightarrow Y$, we partition X into views (X_a, X_b) and learn two classifiers f_a and f_b which can both predict Y. In our case, $f_a : X_a \rightarrow Y$ is the original classifier which uses low-level features to predict the scene classes. We model the sharing between multiple classes via f_b. f_b is a compositional function ($f_b : X_b \rightarrow A \rightarrow Y$) which uses low-level features to predict attributes A and then uses them to predict scene classes. It should be noted that even though we use multi-view settings to model sharing, it is quite a powerful constraint. In case of sharing, the function f_b updates at each iteration by learning a new attribute classifier, $X_b \rightarrow A$, which collects large amounts of data from multiple scene classes (*e.g.*, the indoor attribute classifier picks up training instances from restaurant, bedroom, conference-room *etc.*). Also, note that the mapping from attribute space to scene class, $A \rightarrow Y$, remains fixed in our case.

8.2.3 Pairwise Constraint: Comparative Attributes (CA)

The above two constraints are unary in nature: these constraints still assume that the labeling procedures for two instances X_1 and X_2 should be completely independent of each other. Graph-based approaches [81, 327] have focussed on constraining labels of instances X_1, X_2 based on similarity – that is, if two images are similar they should have same labels. However, learning semantic similarity using

image features is an extremely difficult problem specifically because of high intra-class variations. In this Chapter, we model stronger and richer pairwise constraints on labeling of unlabeled images using comparative attributes. For example, if an image X_1 is labeled as "auditorium", then another image X_2 can be labeled as "amphitheater" if and only if it has more circular structures than X_1.

Formally, for a given pair of scene classes, $f_1 : X_1 \rightarrow Y_1$ and $f_2 : X_2 \rightarrow Y_2$, we model the pairwise constraints using a function $f^c : (X_1, X_2) \rightarrow Y_c$ and enforce the constraint that f^c should produce a consistent triplet (y_1, y_2, y_c) for a given pair of images (x_1, x_2) . Some examples of consistent triplets in our case would include (field, barn, more open space) and (church, cemetery, has larger structures) which mean 'field has more open space than barn' and 'church has larger structures than cemetery'.

8.2.4 Introspection or Self-Cleaning

In iterative semi-supervised approaches, a classifier should ideally become better and better with each iteration. Empirically, these classifiers tend to make more mistakes in the earlier iterations as they are trained on very small amount of data. Based on these two observations, we introduce an additional step of *introspection* where after every five iterations, starting at fifteen, we use the full framework to score already included training data (instead of the unlabeled data) and drop positives that receive very low scores. This results in further performance improvement of the learned classifiers.

8.3 Mathematical Formulation: Putting it together

We now describe how we incorporate the constraints described above in a bootstrapping semi-supervised approach. Figure 8.2 shows the outline of our approach. We have a set of binary scene classifiers $f_1 ... f_N$, attribute classifiers $f_1^a ... f_K^a$ and comparative attribute classifiers $f_1^c ... f_M^c$. Initially, these classifiers are learned using seed examples but are updated at each iteration using new labeled data. At each iteration, we would like to label the large unlabeled corpus ($\mathcal{U}$) and obtain the confidence of each labeling. Instead of labeling all images separately, we label them jointly using our constraints. We represent all images in $\mathcal{U}$ as nodes in a fully connected graph. The most likely assignment of each image (node) in the graph can be posed as minimizing the following energy function $E(y)$ over class labels assignments $y = \{y_1, .., y_{|\mathcal{U}|}\}$:

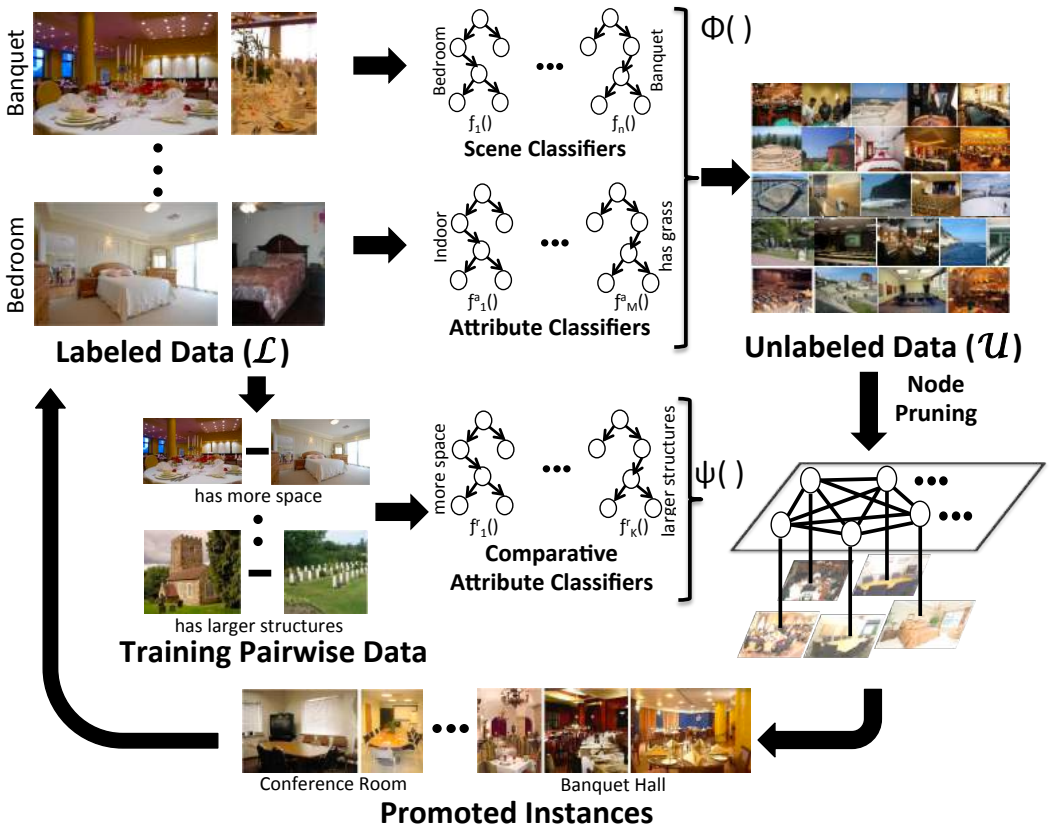


Figure 8.2 – Overview of the proposed Constrained Bootstrapping Framework

$$E(y) = -\left[\sum_{x_i \in \mathcal{U}} \Phi(x_i, y_i) + \lambda \sum_{(x_i, x_j) \in \mathcal{U}^2} \Psi(x_i, x_j, y_i, y_j) \right] \tag{8.1}$$

where $\Phi(x_i, y_i)$ is the unary node potential for image i with features x_i and its candidate label y_i and $\Psi(x_i, x_j, y_i, y_j)$ is the edge potential for labels y_i and y_j of pair of images i and j. It should be noted that y_i denotes assigned label to image i which can take $\{c_1....c_n\}$ possible label assignments.

The unary term $\Phi(x_i, y_i)$ is the confidence in assigning label y_i to image i by the combination of scene and attribute classifier scores. Specifically, if $f_j(x_i)$ is the raw score of j^{th} scene classifier on x_i and $f^{a_k}(x_i)$ is the raw score of k^{th} attribute classifier on x_i, then the unary potential is given by:

$$\Phi(x_i, y_i = c_j) = \sigma(f_j(x_i)) + \lambda_1 \sum_{a_k \in \mathcal{A}} (-1)^{1_{x_i, y_i}(a_k)} \rho(f^{a_k}(x_i)) \tag{8.2}$$

The first term takes in the raw scene classifier scores and converts these scores into potentials using the sigmoid function ($\sigma(t) = \exp(\gamma t)/(1 + \exp(\gamma t))$) [278]. The second term uses a weighted voting scheme for agreement between scene and attribute classifiers (λ_1 is the normalization factor). Here, $\mathcal{A}$ is the set of binary attributes, $\mathbf{1}_{x_i,y_i}(a_k)$ is an indicator function which is 1 if the attribute a_k is detected in the image i but a_k is not a property of scene class y_i (and vice versa). $\rho()$ denotes the confidence in prediction for attribute classifier[2]. Intuitively, this second term votes positive if both the attribute classifier and scene classifier agree in terms of class-attribute relationships. Otherwise, it votes negative where the vote is weighted in terms of the confidence of prediction.

The binary term $\Psi(x_i, x_j, y_i, y_j)$ between images i and j with labels y_i and y_j represents comparative-attribute relations between labeled classes.

$$\Psi(x_i, x_j, y_i, y_j) = \sum_{c_k \in \mathcal{C}} 1_{c_k}(y_i, y_j) \log(\sigma(f^{c_k}(x_i, x_j))) \tag{8.3}$$

where $\mathcal{C}$ denotes the set of comparative attributes, $\mathbf{1}_{c_k}(y_i, y_j)$ denotes if a given comparative attribute c_k exists between pairs of classes y_i and y_j and $f^{c_k}(x_i, x_j)$ is the score of comparative-attribute classifier for the pair of images x_i, x_j. Intuitively, this term boosts the labels y_i and y_j if a comparative attribute c_k scores high on pairwise features. For example, if instances i, j are labeled "conference-room" and "bedroom", their scores get boosted if the comparative attribute "has more space" scores high on pairwise features (since "conference rooms" have more space than "bedrooms").

Promoting Instances. Typically in the semi-supervised problem, $|\mathcal{U}|$ varies from tens of thousand images to millions. Estimating the most likely label for each image in $\mathcal{U}$ necessitates minimizing Eq.(1) which is computationally intractable in general. Since our goal is to find a few very confident images to add to the labeled set $\mathcal{L}$ we do not need to minimize Eq.(1) over the entire $\mathcal{U}$. Instead, we follow the standard practice of pruning the image nodes which have low probability of being classified as one of the n scene classes. Specifically, we evaluate the unary term (Φ), that represents the confidence of assigning label to an image, for the entire $\mathcal{U}$ and use it to prune out and keep only the top-N candidate images for each class. In our experiments, we set N to 3 times the number of instances to be promoted.

While pruning image nodes reduces the search space, exact inference still remains intractable. However, approximate inference techniques like loopy belief prop-

[2]The confidence in prediction is defined as: $\rho(f^{a_k}(x_i)) = max(\sigma(f^{a_k}(x_i)), 1 - \sigma(f^{a_k}(x_i)))$

Table 8.1 – Vocabulary of 15 Scene categories used in this Chapter and their shorthand notation

Scene Categories			
Amphitheater (Am)	Auditorium (Au)	Banquet Hall (BH)	Barn (Bn)
Bedroom (Be)	Bowling Alley (BA)	Bridge (Br)	Casino Indoor (CI)
Cemetery (Ce)	Church Outdoor (CO)	Coast (C)	Conference Room (CR)
Desert Sand (DS)	Field Cultivated (FC)	Restaurant (R)	

agation or Gibbs sampling can be used to find the most likely assignments. In this work, we compute the marginals at each node by running one iteration of loopy belief propagation on the reduced graph. This approximate inference gives us the confidence of candidate class labeling for each image incorporating scene, attribute and comparative-attribute constraints. Now we select C most confidently labeled images for each class ($\mathcal{U}^l$), add them to ($\mathcal{L} \cup \mathcal{U}^l$) $\rightarrow \mathcal{L}$ (and remove from ($\mathcal{U} \backslash \mathcal{U}^l$) $\rightarrow \mathcal{U}$) and re-train our classifiers.

8.3.1 Scene, Attribute and Comparative Attribute Classifiers

We now describe the classifiers used for scenes, attributes and comparative attributes.

Scene & Attribute classifier: Our scene category classifiers as well as attribute classifier are trained using boosted decision trees [126]. We use 20 boosted trees with 8 internal nodes for scene classifier and 40 boosted trees with 8 internal nodes for training our attributes. These classifiers were trained on the 2049 dimensional feature vector from [117]. Our image feature includes 960D GIST [202] features, 75D RGB features (image is resized to 5×5) [284], 30D histogram of line lengths, 200D histogram of orientation of lines and 784D 3D-histogram Lab color space ($14 \times 14 \times 4$).

Comparative attribute classifier: Given a pair of images (x_i, x_j), the goal of comparative attributes classifier is to predict whether the pair satisfies comparative relationships such as "more circular" and "has more indoor space". To model and incorporate comparative attributes, we follow the approach proposed in [107] and train classifiers over differential features $(x_i - x_j)$. We train the comparative attribute classifiers using ground truth pair of images that follow such relationships and for the negative data we use random pair of images and inverse relationships. We used the boosted decision tree classifier with 20 trees and 4 internal nodes.

Table 8.2 – Binary Class-Attribute constraints (BA) used in this Chapter (• stands for the attribute being present for a class.)

	Am	Au	BH	Bn	Be	BA	Br	CI	Ce	CO	C	CR	DS	FC	R
Horizon Visible											•		•	•	
Indoor		•	•		•	•		•				•			•
Has Water							•				•				
Has Building				•			•			•					
Has Seat Rows	•	•													
Has People		•	•			•		•							
Has Grass				•					•					•	
Has Clutter			•												•
Has Chairs & Tables			•					•				•			•
Is Man Made	•	•	•	•	•	•	•	•		•		•			•
Eating Place			•												•
Fun Place						•		•							
Made of Stone	•						•		•	•					
Formal Meeting Place		•										•			
Livable					•										
Part of House					•										
Relaxing Place					•										
Animal-Related Places				•											
Crowd-Related Places	•	•	•												•

8.4 Experiments

We now present experimental results to demonstrate the effectiveness of constraints in bootstrapping based approach. We first present a detailed experimental analysis of our approach using the fully labeled SUN dataset [309]. Using a completely labeled dataset allows us to evaluate the quality of unlabeled images being added to our classifiers and how it affects the performance of our system. Finally, we evaluate our complete system on a large scale dataset which uses approximately 1 million unlabeled images to improve the performance of scene classifiers. For all the experiments we use a fixed vocabulary of scene classes, attributes and comparative attributes as described below.

Vocabulary: We evaluate the performance of our coupled bootstrapping approach in learning 15 scene categories randomly chosen from from SUN dataset (see Table 8.1. We used 19 attributes and 10 comparative attributes, and the relationship between scene category and attributes were defined using a human annotator (see Table 8.2 and 8.3 for the list).

Baselines: The goal of this Chapter is to show the importance of additional constraints in semi-supervised domain. We believe these constraints should improve the performance irrespective of the choice of a particular approach. Therefore, as a baseline, we compare the performance of our constrained bootstrapping approach with

Table 8.3 – Comparative-Attribute constraints (CA) used in our experiments. '$\succ$' is an operator that induces partial ordering using the corresponding comparative attributes. For example, Is more Open — 'Br $\succ$ Ce' means that Bridge is more open then Cemetery (some relations are coupled in groups in this table for compact display)

Is More Open	AM$\succ${Bn, CO}, Br$\succ$Ce, C$\succ${Am, Bn, Br, CO, Ce}, DS$\succ${Am, Bn, Br, Ce, CO}, FC$\succ${Bn, CO, Ce}
Has More Open Space (Outdoor)	FC$\succ${Am, Bn, Ce}, C$\succ${Am, Bn, Br, Ce, FC}, DS$\succ${Am, Bn, Br, Ce, FC}
Has More Open Space (Indoor)	Au$\succ${BH, Be, CI, CR, R}, BH$\succ$Be, BA$\succ$Be, CI$\succ$Be, R$\succ$Be
Has More Seating Space	Au$\succ${BH, Be, CR, R}, CR$\succ$Be, BH$\succ${Be, R}
Has Larger Structures	{Am, Bn, Br, CO}$\succ$Ce, Ce$\succ$FC
Has Taller Structures	Bn$\succ$Ce, CO$\succ$Ce
Has Horizontally Longer Structures	Br$\succ${Bn, Ce, CO}, Am$\succ${Ce, Bn, CO}
Has more water	Br$\succ${Am, Bn, Ce, CO, DS, FC}, C$\succ${Bn, Br, Ce, DS, FC}
Has more sand	DS$\succ${Am, Bn, CO, Ce, C, FC}
Has more greenery	Bn$\succ$CO, FC$\succ${Am, Bn, Br, Ce, CO, C, DS}

two versions of standard bootstrapping approach: one uses independent multiple binary classifiers and the other uses multi-class scene classifiers.

For our first set of experiments, we also compare our constrained bootstrapping framework to the state-of-the-art SSL technique for image classification based on *eigen functions* [81]. Following experimental settings from [81], we map the GIST descriptor for each image down to a 32D space using PCA, use $k = 64$ eigen functions with $\lambda = 50$ and $\epsilon = 0.2$ for computing the Laplacian (see [81] for details).

Evaluation Metric: We evaluate the performance of our approach using two metrics. Firstly, we evaluate the performance of our trained classifier in terms of Average-Precision (AP) at each iteration on a held-out test dataset. Secondly, for the small-scale experiments (Sections 5.1 and 5.2), we also evaluate purity of the promoted instances in terms of the fraction of correctly labeled images.

8.4.1 Using pre-trained attribute classifiers

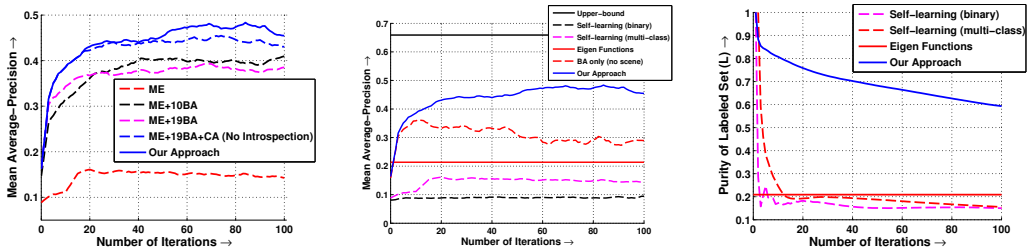
We first evaluate the performance of our approach on SUN dataset. We train the 15 scene classifiers using 2 images each (labeled set). We want to observe the effect of these constraints when the attribute and comparative attribute classifiers are not retrained at each iteration. Therefore, in this case, we used fixed pre-trained attribute classifiers and relative attribute classifiers. These classifiers were trained on 25 examples each (from a held-out dataset). Our unlabeled dataset consists of 18,000 images from SUN dataset. Out of these 18K images, 8.5K images are from these 15 categories and the remainder are randomly sampled from the rest of the dataset. At each iteration, we add 5 images per category from the unlabeled dataset to the classifier.

Figure 8.3 shows examples of how each constraint helps to select better in-

Figure 8.3 – Qualitative Results: We demonstrate how Binary Attribute (middle row) constraints and Comparative Attribute (bottom row) constraints help us promote better instances as compared to naïve Bootstrapping (top row).

stances that should be added to the classifier. The bootstrapping approach clearly faces semantic drift, as it adds "bridge", "coastal" and "park" images to the "amphitheater" classifier. It is the presence of binary attributes such as 'has water' and 'has greenery' that help us to reject these bad candidates. While binary attributes do help to prune lot of bad instances, they sometimes promote bad instances like the "cemetery" image (3^{rd} in 2^{nd} row). However, comparative attributes help us clean such instances. For example, the "cemetery" image is rejected since it has less circular structure. Similarly, the "church" image is rejected since it does not have the long horizontal structures compared to other bridge images. Interestingly, our approach does not overfit to the seed examples and can indeed cover a greater diversity, thus increasing recall. For example, in Figure 8.5, the seed examples for banquet hall include close-view of tables but as iterations proceed we incorporate distant views of banquet hall (eg., 3rd image in iteration 1 and 40).

Next, we quantitatively evaluate the importance of each constraint in terms of performance on held-out test data. Figure 8.4(a) shows the performance of our approach with different combinations of constraints. Our system shows significant



(a) Evaluating Constraints (b) Comparisons with Baselines (c) Purity based Comparison

Figure 8.4 – Quantitative Evaluations: (a) We first evaluate importance of each constraint in our system using control experiments. (b) and (c) Show the comparison of our approach against standard baselines in terms of performance on held-out test data and purity of added instances respectively.

improvement in performance by adding attribute-based constraints. Infact, using a randomly chosen set of ten attributes (ME+10BA) seems to provide enough constraints to avoid semantic drift. Adding another nine attributes to the system does provide some improvement during the initial iterations (ME+19BA). However, at later stages, the effect of these attributes saturate. Finally, we evaluate the importance of comparative attributes by comparing the performance of our system with and without CA constraint. Adding CA constraint does provides significant boost (6-7%) in performance. Figure 8.4(b) shows the comparison of our full system with other baseline approaches. Our approach shows significant improvement over all the baselines which include self-learning approaches based on independent binary classifiers, self-learning based on multi-class classifier and Eigen-functions [81]. We also show the upper-bound on the performance of our approach, which is achieved if all the unlabeled data is manually labeled and used to train the scene classifiers. Since the binary attribute classifiers are pre-trained on some other data, it would be interesting to analyze the performance of these classifiers alone (and without scene classifiers). Our approach performs significantly better than just using attributes alone. This indicates that coupling does provide constraints and help in better labeling of unlabeled data. We also compare the performance of our full system in terms of purity of added instances (See Figure 8.4(c)).

8.4.2 Co-training attributes and comparative attributes

In the previous experiment we showed how each constraint is useful in improving the performance of the semi-supervised approach. To isolate the reasons behind the performance boost, we used fixed pre-trained attribute and comparative attribute classifiers. In this experiment, we train our own attribute and comparative attribute classifiers. These classifiers are trained using the **same 30 images (15 categories** $\times 2$ **images)** which were used to train the initial scene classifiers. Now, we use

Figure 8.5 – Qualitative results showing selected candidates for our approach at iteration 1, 10, 40 and 90. Notice that during the final iterations, there are errors such as bedroom images being added to conference rooms *etc*.

Figure 8.6 – Selected candidates for baseline and our approach at iterations 1 and 60.

the co-training setup where we retrain these attribute and comparative attribute classifiers at each iteration using the new images from unlabeled dataset.

Figure 8.6 shows qualitative results of image instances which are added to the classifiers at iterations 1 and 60. The qualitative results show how the baseline approach suffers from semantic drift and adds "auditorium" to "bedrooms" and "field" to "coast". On the other hand, our approach is more robust and even at 60^{th} iteration adds good instances to the classifier. Figure 8.7 shows the comparison of our approach against baselines. Notice that our approach outperforms all the baselines significantly even though in this case we used the same seed examples to train the attribute classifier. This shows that attribute classifiers can pool information from

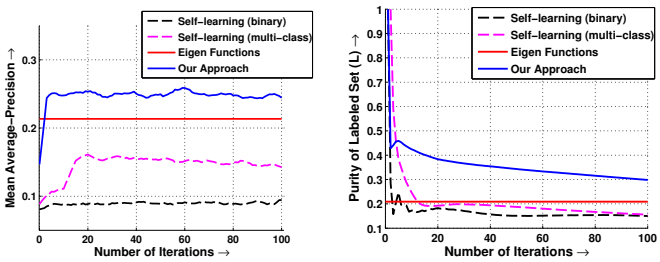


Figure 8.7 – Quantitative Evaluations: We evaluate our approach against standard baselines in terms of (a) mean AP over all scene classes, (b) purity of added instances.

Table 8.4 – Quantitative Results on Large Scale Semi-Supervised Learning (AP Scores)

	Amphitheater	Auditorium	Banquet Hall	Barn	Bedroom	Bowling Alley	Bridge	Casino indoor	Cemetery	Church outdoor	Coast	Conference Room	Desert Sand	Field Cultivated	Restaurant	Mean
Iteration-0	0.517	0.317	0.260	0.309	**0.481**	0.602	0.144	0.647	0.449	0.482	0.539	**0.384**	0.716	**0.700**	0.270	0.455
Self (Binary)	0.557	0.269	**0.324**	0.255	0.458	0.590	0.156	0.644	0.453	0.499	0.466	0.317	0.690	0.572	0.241	0.433
Self (Multi-Class)	0.488	0.254	0.290	0.261	0.443	0.601	0.162	0.655	**0.509**	0.475	0.548	0.322	0.733	0.657	0.303	0.447
Our Approach	**0.587**	**0.333**	0.282	**0.347**	0.474	**0.644**	**0.209**	**0.678**	0.465	**0.506**	**0.582**	0.347	**0.762**	0.654	**0.305**	**0.478**

multiple classes to help avoid semantic drift.

8.4.3 Large scale semi-supervised learning

In the two experiments discussed above, we demonstrated the importance and effectiveness of adding different constraints to the bootstrapping framework. As a final experiment, we now demonstrate the utility of such constraints for large scale learning. We start with 25 seed examples from SUN dataset for each of the 15 categories. Our unlabeled dataset consists of one million images selected from the imagenet dataset [58]. At each iteration, we add 10 images per category from unlabeled dataset to the classifier. Table 8.4 shows the performance of learned scene classifiers after 100 iterations. The constrained bootstrapping approach not only improves the performance by 2.3% but also outperforms all the baselines significantly.

Conclusion. We have presented an approach to constrain semi-supervised learning and reduce semantic drift using binary attributes and comparative attributes. We demonstrate the effectiveness of our approach through extensive experiments including results on a very large dataset of one million images. We believe that our approach can easily scale with categories and in fact its performance would be significantly because increasing the number of classes will add more constraints and enforce extensive sharing which is exactly what our approach exploits. We demonstrate this behaviour in the following Chapters.

Chapter 9

Discovering and Employing Constraints in the Wild

> The true delight is in the finding out rather than in the knowing.
>
> Isaac Asimov

In the previous Chapter, we demonstrated how constraints can be used in a semi-supervised learning framework to reduce semantic drift. We confined the case study to a small list scene categories and manually provided annotated instances and associated constraints. In this Chapter, we propose to apply the ideas from this case study to real world scenarios.

In Section 9.1, we propose a system that can learn these constraints from weakly-supervised, noisy web-data. We extend the list of concepts to scenes, objects and attributes, and type of constraints to scene-object, object-object, scene-attribute and object-attribute relationships. Collectively, discovered instances of concepts and relationships are referred to as Visual Knowledge-base. We show that these automatically discovered relationships are good for constrained SSL, and the newly labeled instances lead to better recognition models. We will only provided relevant details in Section 9.1, other details can be found in the technical report [41].

In Section 9.2, we demonstrate how constraints can be discovered and harnessed in large-scale videos, where only a handful of frames are sparsely labeled with concepts. We study constraints like decorrelated errors, reliable tracking and diverse selection, and show that they are effective in arresting semantic drift in videos. The proposed technique handles detection of multiple objects without assuming exhaustive labeling of object instances on any input frame; and starting with a handful of labeled examples, it can label hundreds of thousands of new examples which also

improve object detectors. We will only provided relevant details in Section 9.2, other details can be found in the technical report [192].

9.1 Constraints from Weakly-supervised Web-data

Recent successes in computer vision can be primarily attributed to the ever increasing size of visual knowledge in terms of labeled instances of scenes, objects, actions, attributes, and the contextual relationships between them. But as we move forward, a key question arises: how will we gather this structured visual knowledge on a vast scale? Recent efforts such as ImageNet [58] and Visipedia [210] have tried to harness human intelligence for this task. However, we believe that these approaches lack both the richness and the scalability required for gathering massive amounts of visual knowledge.

In this Section, we consider an alternative approach of automatically extracting visual knowledge from Internet scale data. The feasibility of extracting knowledge automatically from images and videos will itself depend on the state-of-the-art in computer vision. While we have witnessed significant progress on the task of detection and recognition, we still have a long way to go for automatically extracting the semantic content of a given image. So, is it really possible to use existing approaches for gathering visual knowledge directly from web data?

NEIL – Never Ending Image Learner. We propose NEIL, a constrained semi-supervised learning (SSL) system that exploits the big scale of visual data to automatically extract common sense relationships and then uses these relationships to label visual instances of existing categories. Specifically, NEIL can use web data to extract: (a) Labeled examples of object categories with bounding boxes; (b) Labeled examples of scenes; (c) Labeled examples of attributes; (d) Visual subclasses for object categories; and (e) Common sense relationships about scenes, objects and attributes like "Corolla is a kind of/looks similar to Car", "Wheel is a part of Car", *etc.* (See Figure 9.1). NEIL jointly discovers both labeling of instances as well as relationships amongst them at a gigantic scale, which provides constraints for semi-supervised learning.

9.1.1 Related Work

Recent work has only focused on extracting knowledge in the form of large datasets for recognition and classification [58, 171, 210]. One of the most commonly used approaches to build datasets is using manual annotations by motivated teams of people [210] or the power of crowds [58, 293]. To minimize human effort, recent

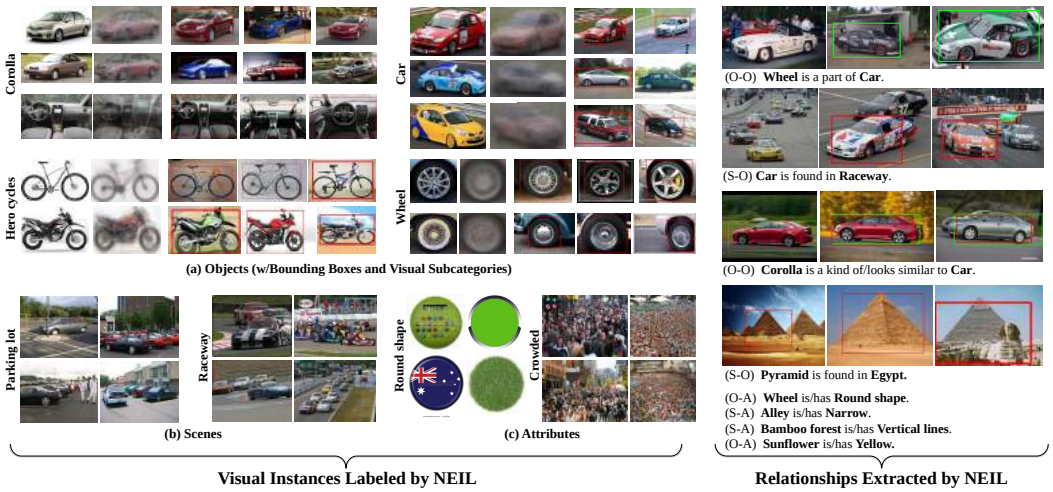


Figure 9.1 – NEIL is a computer program that runs 24 hours a day and 7 days a week to gather visual knowledge from the Internet. Specifically, it simultaneously labels the data and extracts common sense relationships between categories.

works have also focused on active learning [255, 291] which selects label requests that are most informative. However, both of these directions have a major limitation: annotations are expensive, prone to errors, biased and do not scale.

An alternative approach is to use visual recognition for extracting these datasets automatically from the Internet [171, 240, 252]. A common way of automatically creating a dataset is to use image search results and rerank them via visual classifiers [80] or some form of joint-clustering in text and visual space [18, 240]. Another approach is to use a semi-supervised framework [327]. Here, a small amount of labeled data is used in conjunction with a large amount of unlabeled data to learn reliable and robust visual models. These seed images can be manually labeled [252] or the top retrievals of a text-based search [171]. The biggest problem with most of these automatic approaches is that the small number of labeled examples or image search results do not provide enough constraints for learning robust visual classifiers. Hence, these approaches suffer from semantic drift [51]. One way to avoid semantic drift is to exploit additional constraints based on the structure of our visual data. Researchers have exploited a variety of constraints such as those based on visual similarity [65, 81], semantic similarity [105] or multiple feature spaces [20]. However, most of these constraints are weak in nature: for example, visual similarity only models the constraint that visually-similar images should receive the same labels. On the other hand, our visual world is highly structured: object categories share parts and attributes, objects and scenes have strong contextual relationships, *etc.* Therefore, we need a way to capture the rich structure of our visual world and

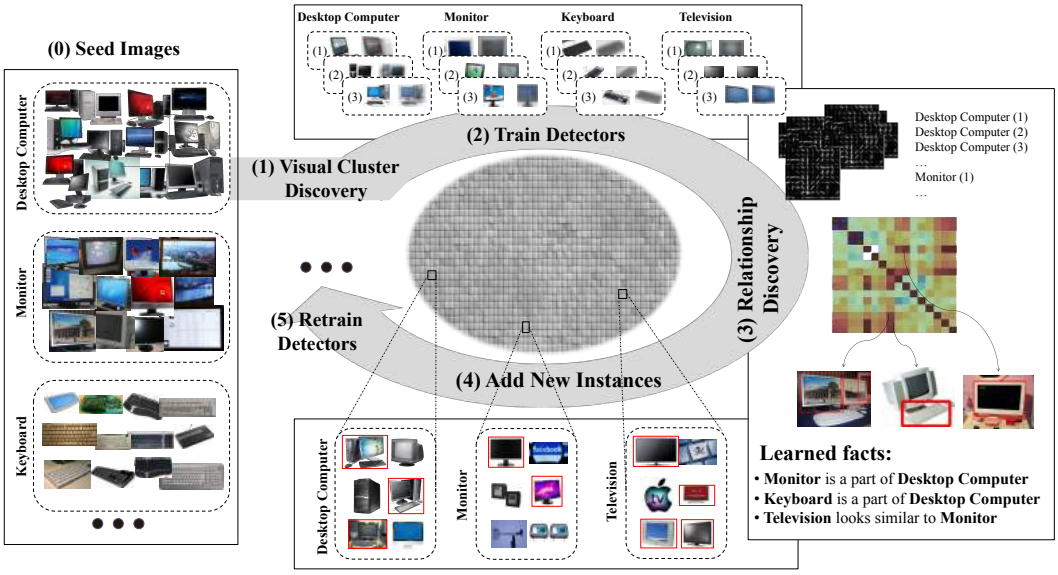


Figure 9.2 – Outline of Iterative Approach

exploit this structure during semi-supervised learning.

In recent years, there have been huge advances in modeling the rich structure of our visual world via contextual relationships [76, 184, 208, 219, 267]. Some of these relationships include: Scene-Object [267], Object-Object [184, 219], Object-Attribute [76, 163, 207], Scene-Attribute [208]. All these relationships can provide a rich set of constraints which can help us improve SSL [30]. But the big question is: how do we obtain these relationships? One way to obtain such relationships is via text analysis [31, 107]. However, as [293] points out that the visual knowledge we need to obtain is so obvious that no one would take the time to write it down and put it on web.

In this work, we argue that, at a large-scale, one can jointly discover relationships and constrain the SSL problem for extracting visual knowledge and learning visual classifiers and detectors. Motivated by a never ending learning algorithm for text [31], we propose a never ending visual learning algorithm that cycles between extracting global relationships, labeling data and learning classifiers/detectors for building visual knowledge from the Internet. Our work is also related to attribute discovery [221, 245] since these approaches jointly discover the attributes and relationships between objects and attributes simultaneously. However, in our case, we only focus on semantic attributes and therefore our goal is to discover semantic relationships and semantically label visual instances.

9.1.2 Technical Approach

Our goal is to extract visual knowledge from the pool of visual data on the web. We define visual knowledge as any information that can be useful for improving vision tasks such as image understanding and object/scene recognition. One form of visual knowledge would be labeled examples of different categories or labeled segments/boundaries. Labeled examples helps us learn classifiers or detectors and improve image understanding. Another example of visual knowledge would be relationships. For example, spatial contextual relationships can be used to improve object recognition. In this Section, we represent visual knowledge in terms of labeled examples of semantic categories and the relationships between those categories. Our knowledge base consists of labeled examples of: (1) Objects (*e.g.*, Car, Corolla); (2) Scenes (*e.g.*, Alley, Church); (3) Attributes (*e.g.*, Blue, Modern). Note that for objects we learn detectors and for scenes we build classifiers; however for the rest of this Section we will use the term detector and classifier interchangeably. Our knowledge base also contains relationships of four types: (1) Object-Object (*e.g.*, Wheel is a part of Car);(2) Object-Attribute (*e.g.*, Sheep is/has White); (3) Scene-Object (*e.g.*, Car is found in Raceway); (4) Scene-Attribute (*e.g.*, Alley is/has Narrow).

The outline of our approach is shown in Figure 9.2. We use text-based image search to download thousands of images for each object, scene and attribute category. Our method then uses an iterative approach to clean the labels and train detectors/classifiers in a semi-supervised manner. For a given concept (*e.g.*, car), we first discover the latent visual sub-categories and bounding boxes for these sub-categories using an exemplar-based clustering approach similar to the one described previously in Section A3. We then train multiple detectors for a concept (one for each sub-category) using the clustering and localization results. These detectors and classifiers are then used for detections on millions of images to learn relationships based on co-occurrence statistics (Section 9.1.2). Here, we exploit the fact the we are interested in macro-vision and therefore build co-occurrence statistics using only confident detections/classifications. Once we have relationships, we use them in conjunction with our classifiers and detectors to label the large set of noisy images (Section 9.1.2. The most confidently labeled images are added to the pool of labeled data and used to retrain the models, and the process repeats itself. At every iteration, we learn better classifiers and detectors, which in turn help us learn more relationships and further constrain the semi-supervised learning problem.

Seeding Classifiers via Web-data. The first step in our semi-supervised algorithm is to build classifiers for visual categories. One way to build initial classifiers is via a few manually labeled seed images. Here, we take an alternative approach

(a) Google Image Search for "tricycle" (b) Sub-category Discovery

Figure 9.3 – An example of how clustering handles polysemy, intra-class variation and outlier removal. (a) Initial image search results. (b) Clusters discovered by our method.

and use text-based image retrieval systems to provide seed images for training initial detectors. For scene and attribute classifiers we directly use these retrieved images as positive data. However, such an approach fails for training object and attribute detectors because of four reasons (Figure 9.3(a)) – (1) Outliers: Due to the imperfectness of text-based image retrieval, the downloaded images usually have irrelevant images/outliers; (2) Polysemy: In many cases, semantic categories might be overloaded and a single semantic category might have multiple senses (*e.g.*, apple can mean both the company and the fruit); (3) Visual Diversity: Retrieved images might have high intra-class variation due to different viewpoint, illumination *etc.*; (4) Localization: In many cases the retrieved image might be a scene without a bounding-box and hence one needs to localize the concept before training a detector. Therefore, to discover object sub-categories, we propose to use the technique presented in Section A3. Figure 9.3 shows an example of discovered clusters.

Discovering Relationships

Once we have initialized object detectors, attribute detectors, attribute classifiers and scene classifiers, we can use them to extract relationships automatically from the data. The key idea is that we do not need to understand each and every image downloaded from the web but instead understand enough images to learn the statistical pattern of detections and classifications at a large scale. These patterns can be used to select the top-N relationships at every iteration. Specifically, we extract four different kinds of relationships:

Object-Object Relationships. The first kind of relationship we extract are object-object relationships which include: (1) Partonomy relationships such as "Eye is a part of Baby"; (2) Taxonomy relationships such as "BMW 320 is a kind of Car"; and (3) Similarity relationships such as "Swan looks similar to Goose". To extract these relationships, we first build a co-detection matrix O_0 whose elements represent the probability of simultaneous detection of object categories i and j. Intuitively,

the co-detection matrix has high values when object detector i detects objects inside the bounding box of object j with high detection scores. To account for detectors that fire everywhere and images which have lots of detections, we normalize the matrix O_0. The normalized co-detection matrix can be written as: $N_1^{-\frac{1}{2}} O_0 N_2^{-\frac{1}{2}}$, where N_1 and N_2 are out-degree and in-degree matrix and (i, j) element of O_0 represents the average score of top-detections of detector i on images of object category j. Once we have selected a relationship between pair of categories, we learn its characteristics in terms of mean and variance of relative locations, relative aspect ratio, relative scores and relative size of the detections, and use these characteristics to label the type of relationship.

Object-Attribute Relationships. The second type of relationship we extract is object-attribute relationships such as "Pizza has Round Shape", "Sunflower is Yellow" *etc.* To extract these relationships we use the same methodology where the attributes are detected in the labeled examples of object categories. These detections and their scores are then used to build a normalized co-detection matrix which is used to find the top object-attribute relationships.

Scene-Object Relationships. The third type of relationship extracted by our algorithm includes scene-object relationships such as "Bus is found in Bus depot" and "Monitor is found in Control room". For extracting scene-object relationships, we use the object detectors on randomly sampled images of different scene classes. The detections are then used to create the normalized co-presence matrix (similar to object-object relationships) where the (i, j) element represents the likelihood of detection of instance of object category i and the scene category class j.

Scene-Attribute Relationships. The fourth and final type of relationship extracted by our algorithm includes scene-attribute relationships such as "Ocean is Blue", "Alleys are Narrow", *etc.* Here, we follow a simple methodology for extracting scene-attribute relationships where we compute co-classification matrix such that the element (i, j) of the matrix represents average classification scores of attribute i on images of scene j. The top entries in this co-classification matrix are used to extract scene-attribute relationships.

Labeling instances for Constrained SSL

Once we have the initial set of classifiers/detectors and the set of relationships, we can use them to find new instances of different objects and scene categories. These new instances are then added to the set of labeled data and we retrain new classifiers/detectors using the updated set of labeled data. These new classifiers are then used to extract more relationships which in turn are used to label more data

Figure 9.4 – Qualitative Examples of Bounding Box Labeling done by NEIL

and so on. One way to find new instances is directly using the detector itself. For instance, using the car detector to find more cars. However, this approach leads to semantic drift. Following our case study, to avoid semantic drift, we use the rich set of relationships we extracted and ensure that the new labeled instances of car satisfy the extracted relationships (*e.g.*, has wheels, found in raceways *etc.*)

Mathematically, let $\mathcal{R}_{\mathcal{O}}$, $\mathcal{R}_{\mathcal{A}}$ and $\mathcal{R}_{\mathcal{S}}$ represent the set of object-object, object-attribute and scene-object relationships at iteration t. If $\phi_i(\cdot)$ represents the potential from object detector i, $\omega_k(\cdot)$ represents the scene potential, and $\psi_{i,j}(\cdot)$ represent the compatibility function between two object categories i, j, then we can find the new instances of object category i using the contextual scoring function

$$\phi_i(x) + \sum_{i,j \in \mathcal{R}_O \bigcup \mathcal{R}_A} \phi_j(x_l)\psi_{i,j}(x, x_l) + \sum_{i,k \in \mathcal{R}_S} \omega_k(x)$$

where x is the window being evaluated and x_l is the top-detected window of related object/attribute category. The above equation has three terms: the first term is appearance term for the object category itself and is measured by the score of the SVM detector on the window x. The second term measures the compatibility between object category i and the object/attribute category j if the relationship (i, j) is part of the catalogue. For example, if "Wheel is a part of Car" exists in the catalogue then this term will be the product of the score of wheel detector and the compatibility function between the wheel window (x_l) and the car window (x). The final term measures the scene-object compatibility. Therefore, if the knowledge base contains the relationship "Car is found in Raceway", this term boosts the "Car" detection scores in the "Raceway" scenes.

At each iteration, we also add new instances of different scene categories. We find new instances of scene category k using the contextual scoring function

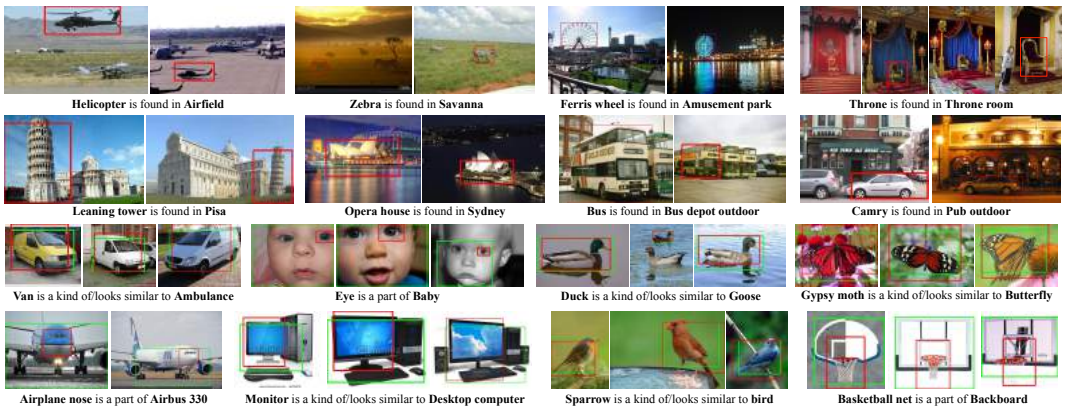


Figure 9.5 – Qualitative Examples of Scene-Object (rows 1-2) and Object-Object (rows 3-4) Relationships Extracted by NEIL.

$$\omega_k(x) + \sum_{m,k \in \mathcal{R}_{A'}} \omega_m(x) + \sum_{i,k \in \mathcal{R}_S} \phi_i(x_l)$$

where $\mathcal{R}_{A'}$ represents the catalogue of scene-attribute relationships. The above equation has three terms: the first term is the appearance term for the scene category itself and is estimated using the scene classifier. The second term is the appearance term for the attribute category and is estimated using the attribute classifier. This term ensures that if a scene-attribute relationship exists then the attribute classifier score should be high. The third and the final term is the appearance term of an object category and is estimated using the corresponding object detector. This term ensures that if a scene-object relationship exists then the object detector should detect objects in the scene.

Implementation Details. To train scene & attribute classifiers, we first extract a 3912 dimensional feature vector from each image. The feature vector includes 512D GIST [202] features, concatenated with bag of words representations for SIFT [183], HOG [54], Lab color space, and Texton [189]. The dictionary sizes are 1000, 1000, 400, 1000, respectively. Features of randomly sampled windows from other categories are used as negative examples for SVM training and hard mining. For the objects and attributes, we use CHOG [244] features with a bin size of 8. We train the detectors using latent SVM model (without parts) [79].

9.1.3 Experiments and Results

We demonstrate the quality of visual knowledge by qualitative results, verification via human subjects and quantitative results on tasks such as object detection

Table 9.1 – mAP performance for scene classification on 12 categories

	mAP
Seed Classifier (15 Google Images)	0.52
Bootstrapping (without relationships)	0.54
NEIL Scene Classifiers	0.57
NEIL (Classifiers + Relationships)	**0.62**

and scene recognition.

NEIL Statistics. While NEIL's core algorithm uses a fixed vocabulary, we use noun phrases from NELL [31] to grow NEIL's vocabulary. So far, NEIL has downloaded more than 2.2 million images, and learned an ontology of 2702 concepts with 8685 visual concepts (including sub-categories). It has labeled more than 1 million bounding boxes, 517k segmentations and 4695 relationships. For bootstrapping this system, we used a few seed images from ImageNet [58], SUN [309] or the top-images from Google image search. The current visual knowledge base can be browsed at www.neil-kb.com.

Qualitative Results. We first show some qualitative results in terms of extracted visual knowledge by NEIL. Figure 9.4 shows the extracted visual sub-categories along with a few labeled instances belonging to each sub-category. It can be seen from the figure that NEIL effectively handles the intra-class variation and polysemy via the clustering process. The purity and diversity of the clusters for different concepts indicate that contextual relationships help make our system robust to semantic drift and ensure diversity.

Figure 9.5 shows some of the qualitative examples of scene-object and object-object relationships extracted by NEIL. It is effective in using a few confident detections to extract interesting relationships. Figure 9.6 shows some of the interesting scene-attribute and object-attribute relationships extracted by NEIL.

Evaluating Quality via Human Subjects Next, we want to evaluate the quality of extracted visual knowledge by NEIL. It should be noted that an extensive and comprehensive evaluation for the whole NEIL system is an extremely difficult task. It is impractical to evaluate each and every labeled instance and each and every relationship for correctness. Therefore, we randomly sample the 500 visual instances and 500 relationships, and verify them using human experts. By iteration 6, 79% of the relationships extracted by NEIL are correct, and 98% of the visual

Table 9.2 – mAP performance for object detection on 15 categories

	mAP
Latent SVM (50 Google Images)	0.34
Latent SVM (450 Google Images)	0.28
Latent SVM (450, Aspect Ratio Clustering)	0.30
Latent SVM (450, HOG-based Clustering)	0.33
Seed Detector (NEIL Clustering)	0.44
Bootstrapping (without relationships)	0.45
NEIL Detector	0.49
NEIL Detector + Relationships	**0.51**

data labeled by NEIL has been labeled correctly. We also evaluate the per iteration correctness of relationships: At iteration 1, more than 96% relationships are correct and by iteration 3, the system stabilizes and 80% of extracted relationships are correct. We also evaluate the quality of bounding-boxes generated by NEIL. For this we sample 100 images randomly and label the ground-truth bounding boxes. On the standard intersection-over-union metric, NEIL generates bounding boxes with 0.78 overlap on average with ground-truth. To give context to the difficulty of the task, the standard Objectness algorithm [4] produces bounding boxes with 0.59 overlap on average.

Using Knowledge for Vision Tasks. Finally, we want to demonstrate the usefulness of the visual knowledge learned by NEIL on standard vision tasks such as object detection and scene classification. Here, we will also compare several aspects of our approach: (a) We first compare the quality of our automatically labeled dataset. As baselines, we train classifiers/detectors directly on the seed images downloaded from Google Image Search. (b) We compare NEIL against a standard bootstrapping approach which does not extract/use relationships. (c) Finally, we will demonstrate the usefulness of relationships by detecting and classifying new test data with and without the learned relationships.

Scene Classification. First we evaluate our visual knowledge for the task of scene classification. We build a dataset of 600 images (12 scene categories) using Flickr images. We compare the performance of our scene classifiers against the scene classifiers trained from top 15 images of Google Image Search (our seed classifier). We also compare the performance with standard bootstrapping approach without using any relationship extraction. Table 9.1 shows the results. We use mean average precision (mAP) as the evaluation metric. As the results show, automatic relationship

Monitor is found in **Control room**
Washing machine is found in **Utility room**
Siberian tiger is found in **Zoo**
Baseball is found in **Butters box**
Bullet train is found in **Train station platform**
Cougar looks similar to **Cat**
Urn looks similar to **Goblet**
Samsung galaxy is a kind of **Cellphone**
Computer room is/has **Modern**
Hallway is/has **Narrow**
Building facade is/has **Check texture**
Trading floor is/has **Crowded**

Umbrella looks similar to **Ferris wheel**
Bonfire is found in **Volcano**

Figure 9.6 – Examples of extracted common sense relationships.

extraction can be used for constrained SSL since the learned classifiers give much better performance.

Object Detection: We also evaluate our extracted visual knowledge for the task of object detection. We build a dataset of 1000 images (15 object categories) using Flickr data for testing. We compare the performance against object detectors trained directly using (top-50 and top-450) images from Google Image Search. We also compare the performance of detectors trained after aspect-ratio, HOG clustering and our proposed clustering procedure. Table 9.2 shows the detection results. Using 450 images from Google image search decreases the performance due to noisy retrievals. While other clustering methods help, the gain by our clustering procedure is much larger. Finally, our detectors trained perform better than standard bootstrapping.

9.2 Constraints from Sparsely-supervised Videos

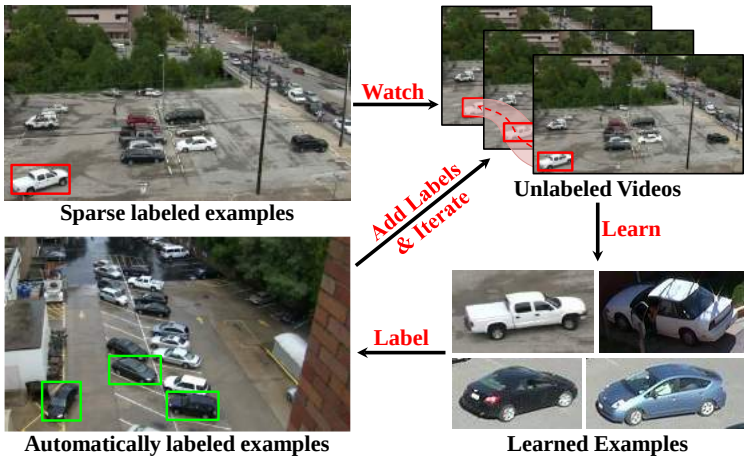


Figure 9.7 – We present a novel formulation of semi-supervised learning for automatically learning object detectors from videos. Our method works with long video to automatically learn bounding box level annotations for multiple object instances. It does not assume exhaustive labeling of every object instance in the input videos, and from a handful of labeled instances can automatically label hundreds of thousands of instances.

The availability of large labeled image datasets [54, 58] has been one of the key factors for advances in recognition. These datasets, which have been largely curated from internet images, have not only helped boost performance [79, 96], but have also fostered the development of new techniques [96, 155]. However, compared to images, videos seem like a more natural source of training data because of the additional temporal continuity they offer for both learning and labeling. So ideally we should have large labeled internet video datasets.

Consider the scale of internet videos – YouTube has 100 hours of video (10 million frames) uploaded every minute. It seems unlikely that human per-image labeling will scale to this amount of data. Given this scale of data and the associated annotation problems [201, 316], which are more pronounced in videos, it is no surprise that richly annotated large video recognition datasets are hard to find. In fact, the available video datasets [149, 151, 188, 201] lack the kind of annotations offered by benchmark image datasets [54, 58, 175].

One way to tackle the labeling problem is using semi-supervised learning (SSL), as proposed previously in this Chapter. In this Section, we present an approach to constrain the semi-supervised learning process [252] in videos. Our technique constrains the SSL process by using *multiple weak cues* - appearance, motion, temporal *etc.*, in video data and automatically learns *diverse new examples*.

Intuitively, algorithms dealing with videos should use appearance and temporal cues using detection and tracking, respectively. One would expect a simple combination of detection and tracking to constitute a semi-supervised framework that would prevent drift since both of these processes would ideally cancel each others' errors. However, as we show in our experiments (Sec. 9.2.4), a naïve combination of these two techniques performs poorly. In the long run, the errors in both detection and tracking are amplified in a coupled system. We can also consider pure detection approaches or pure tracking approaches for this problem. However, pure detection ignores temporal information while pure tracking tends to stray away over a long duration.

We present a scalable framework that discovers objects in video using SSL (see Figure 9.7). It tackles the challenging problem of localizing new object instances in long videos starting from only a few labeled examples. In addition, we present our algorithm in a realistic setting of "sparse labels" [201], i.e., in the few initial "labeled" frames, not all objects are annotated. This setting relaxes the assumption that in a given frame, all object instances have been exhaustively annotated. It implies that we do not know if any unannotated region in the frame belongs to the object category or the background, and thus cannot use any region from our input as negative data. While much of the past work has ignored this type of sparse labeling (and *lack of explicit negatives*), we show ways to overcome this handicap. Figure 9.8 presents an overview of our algorithm. Our proposed algorithm is different from the rich body of work on tracking-by-detection. Firstly, we do not attempt to solve the problem of *data association*. Our framework does not try to identify whether it has seen a particular instance before. Secondly, since it works in the regime of *sparse annotations*, it does not assume that negative data can be sampled from around the current box. Thirdly, we limit expensive computation to a subset of the input frames to scale to a million frames.

Contributions. We present an SSL framework that *localizes multiple unknown objects* in videos. Starting from few *sparsely labeled* objects, it iteratively labels new and useful training examples in the videos. Our key contributions are: 1) We tackle the SSL problem for discovering multiple objects in sparsely labeled videos; 2) We present an approach to constrain SSL by combining multiple weak cues in videos and exploiting decorrelated errors by modeling in multiple feature spaces. We demonstrate its effectiveness as compared to traditional tracking-by-detection approaches; 3) Given the redundancy in videos, the method needs to automatically determine the relevance of training examples to the target detection task. We present a way to include *relevance and diversity of the training examples* in each iteration of SSL, leading to a scalable *incremental learning* algorithm.

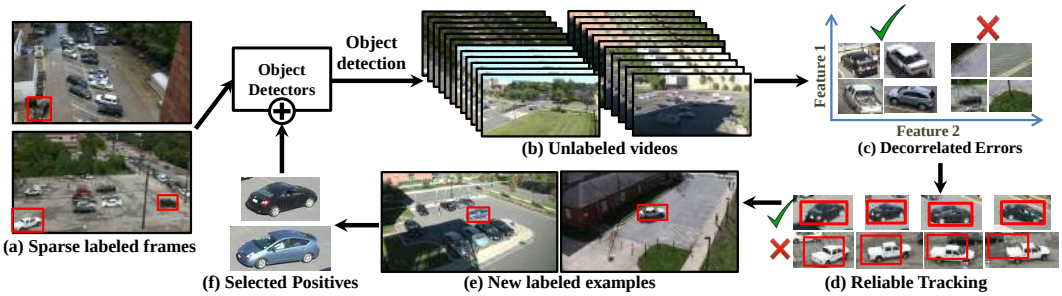


Figure 9.8 – Our approach selects samples by iteratively discovering new boxes by a careful fusion of detection, robust tracking, relocalization and multi-view modeling of positive data. It shows how an interplay between these techniques can be used to learn from large scale unlabeled video corpora.

9.2.1 Related Work

The availability of web scale image and video data has made semi-supervised learning more popular in recent years. In the case of images, many methods [52, 81] rely on image similarity measures, and try to assign similar labels to close-by unlabeled images. However, in the case of real-world images, obtaining good image similarity is hard and hence the simple approaches become less applicable. One major body of work (including previous Sections of this Chapter) [41, 45, 61, 157, 252] tries to overcome this difficulty by leveraging the use of a set of pre-defined attributes for image classes [45, 252] and additionally web-supervision and text [41, 61]. While these methods have good performance for images, they are not well-suited for videos mainly because they treat each image independently and do not use video constraints. One major reason for the success of attribute based methods for SSL was the relatively cheap supervision required for attributes (per-image level tag). In the same spirit, weakly-supervised video approaches use tags available with internet videos.

Weakly-supervised video algorithms have gained popularity largely due to the abundance of video level tags on the internet. The input is a set of videos with video level tags (generally a video belongs to an object category), and the algorithm discovers the object (if present) in the video. These methods, while effective, assume a maximum of one dominant object per video [177, 216, 274, 296]. Some of them additionally assume dominant motion [216] or appearance saliency [169, 177, 216, 274, 296] for the object of interest. The methods of video co-segmentation [37, 87, 106, 137] can be considered a subset of weakly supervised methods. They make a strong assumption that multiple videos contain the exact same object in majority of the frames. This assumption of at most one salient object in a video is rarely satisfied by internet or real world videos. When this assumption does not hold,

methods cannot strongly rely on motion based foreground/background clustering or on appearance saliency. Our proposed work deals with *multiple objects* and can even discover *static* object instances without strongly relying on motion/appearance saliency. However, we do require richer bounding box annotations by way of a few *sparsely labeled* instances. A concurrent work [172] utilizes weakly labeled video for improving detection.

A relevant thread of work which also uses bounding box annotations is that of tracking-by-detection. It has a long and rich history in computer vision and the reader is referred to [206] for a survey. The tracking-by-detection algorithms start with bounding box annotation(s) of the object(s) to track the object(s) over a long period of time. The underlying assumption is that negative data can be sampled from around the object [9, 101, 112, 143, 236, 269, 275] to distinguish between the object and background. This is not valid in the case of *sparsely labeled* videos because the unmarked regions may contain more instances of the same object, rather than background.

Other tracking-by-detection methods [17, 92, 214] do not sample negative data from the input video. They do so at the cost of using a detector trained on additional training data, *e.g.*, [214] uses a DPM [79] trained on images from PASCAL [54]. In contrast, we do not use such additional training data for our method. This allows us to work on object categories which are not in standard datasets.

Multi-object tracking-by-detection approaches also focus on solving the problem of *data association* - given object locations across multiple frames, determine which locations belong to the same object. Data association is critical for long term tracking, and is also very challenging [17]. In contrast, our goal is not to track over long periods, but to get short and reliable tracking. Moreover, we do not require these short tracklets to be associated with each other, and thus have minimal need for data association.

To summarize, our SSL framework operates in a less restrictive domain compared to existing work in weakly-labeled object discovery and tracking-by-detection. The key differences are: 1) We localize multiple objects in a single frame as opposed to zero or one objects. 2) We do not assume strong motion or appearance saliency of objects, thus discovering static object instances as well. 3) We operate in the regime of *sparsely labeled* videos. Thus, in any given frame, all the unmarked region may contain instances of the object. This does not allow using negative data from the input frame. 4) We *do not need explicit negative data* or any pre-trained object models. 5) Finally, the aim of our approach is very different from tracking approaches. We do not want to track objects over a long period of time, but want

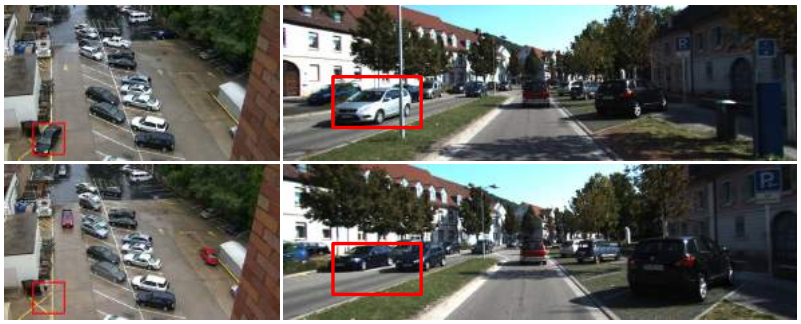


Figure 9.9 – Sparsely labeled positives (as shown in the top row) are used to train Exemplar detectors [185]. Since we do not assume exhaustive labeling of each instance in the image, we cannot sample negative data from around the input boxes. When these detectors (trained without domain negatives) are used, they may learn background features (like the co-occuring yellow stripes or road divider) and give high confidence false positives (bottom row). We address this issue by exploiting decorrelated errors (Section 9.2.3).

short reliable tracklets.

9.2.2 Approach Overview

There are two ways to detect objects in videos – either using detection in individual frames or tracking across frames. In a semi-supervised framework, detection plays the role of constraining the system by providing an appearance prior for the object, while tracking generalizes by providing newer views of the object. So one could imagine a detection and tracking combination, in which one tracks from confident detections and then updates the detector using the tracked samples. However, as we show in our experiments (Section 9.2.4), such a naïve combination does not impose enough constraints for SSL. In contrast, our approach builds on top of this basic combination of detection and tracking to correct their mistakes.

Our algorithm starts with a few sparsely annotated video frames ($\mathcal{L}$) and iteratively discovers new instances in the large unlabeled set of videos ($\mathcal{U}$). Simply put, we first train detectors on annotated objects, followed by detection on input videos. We determine good detections (removing confident false positives) which serve as starting points for short-term tracking. The short-term tracking aims to label new and unseen examples reliably. Amongst these newly labeled examples, we identify good and diverse examples which are used to update the detector without re-training from scratch. We iteratively repeat this fusion of tracking and detection to label new examples. We now describe our algorithm and the constraints we use (illustrated in Figure 9.8).

Sparse Annotations (lack of explicit negatives). We start with a few sparsely annotated frames in a random subset of $\mathcal{U}$. Sparse labeling implies that unlike other

approaches [143], we do not assume exhaustively annotated input, and thus cannot sample negatives from the vicinity of labeled positives. We use random images from the internet as negative data for training object detectors on these sparse labels [251]. We use these detectors to detect objects on a *subset of the video*, *e.g.*, every 30 frames. Training on a few positives without domain negatives can result in high confidence false positives as shown in Figure 9.9. Removing such false positives is important because if we track them, we will add more bad training examples, thus degrading the detector's performance over iterations.

Temporally consistent detections. We first remove detections that are temporally inconsistent using a smoothness prior on the motion of detections.

Decorrelated errors. To remove high confidence false positives (see Figure 9.9), we rely on the principle of *decorrelated errors* (similar to *multi-view* SSL [227, 259]). The intuition is that the detector makes mistakes that are related to its feature representation [295], and a different feature representation would lead to different errors. Thus, if the errors in different feature spaces are decorrelated, one can correct them and remove false positives. This gives us a filtered set of detections.

Reliable tracking. We track these filtered detections to label new examples. Our final goal is not to track the object over a long period. Instead, our goal is to track reliably and label new and hopefully diverse examples for the object detector. To get such reliable tracks we design a conservative *short-term tracking* algorithm that identifies *tracking failures*. Traditional tracking-by-detection approaches [92, 214] rely heavily on the detection prior to identify tracking failures. In contrast, the goal of our tracking is to improve the (weak) detector itself. Thus, heavily relying on input from the detector defeats the purpose of using tracking in our case.

Selection of diverse positives for updating the detector. The reliable tracklets give us a large set of automatically labeled boxes which we use to update our detector. Previous work [216] temporally subsamples boxes from videos, treating each box with equal importance. However, since these boxes come from videos, a large number of them are redundant and do not have equal importance for training our detector. Additionally, the relevance of an example added at the current iteration i depends on whether similar examples were added in earlier iterations. One would ideally want to train (make an *incremental update*) only on new and diverse examples, rather than re-train from scratch on thousands of largely redundant boxes. We address this issue by selection and show a way of training only on diverse, new boxes. After training detectors on diverse examples, we repeat the SSL process to iteratively label more examples.

Stopping criterion of SSL. It is desirable to have SSL algorithms which auto-

matically determine when they should stop. We stop our SSL once our selection algorithm indicates that it does not have any good candidates to select.

9.2.3 Approach Details

We start with a small sparsely labeled set $\mathcal{L}_0$ of bounding boxes and unlabeled input videos $\mathcal{U}$. At each iteration i, using models trained on $\mathcal{L}_{i-1}$, we want to label new boxes in the input videos $\mathcal{U}$, add them to our labeled set $\mathcal{L} = \mathcal{L} \cup \mathcal{L}_i$, and iteratively repeat this procedure. The two key components of this constrained SSL algorithm are detecting outliers using decorrelated errors, and performing reliable tracking. Finally, to we bias our selection to add diverse samples.

Detections with decorrelated errors

We train object detectors on our initial set of examples using random images from Flickr as negatives [251] (Chapter 7). We detect on a uniformly sampled subset of the video frames and remove the temporally inconsistent detections. Since the negative data for training the detectors comes from a different domain than the positives, we still get consistent false positive detections because the detector learns the wrong concept (see Figures 9.9 and 9.11). To remove such false positives, we perform outlier removal in a feature space different from that of the detector. The intuition is that the errors made by learning methods are correlated with their underlying feature representation, and thus using decorrelated feature spaces might help correct them. For outlier removal, we use unconstrained Least Squares Importance Fitting (uLSIF) [144]. These final filtered detections serve as starting points for reliable short term tracking.

Reliable Tracking

We formulate a scalable tracking procedure that effectively capitalizes on priors available from detection, color/texture consistency, objectness [4, 289] and optical flow. More importantly, our tracking procedure is very good at identifying its own failures. This property is vital in our semi-supervised framework since any tracking failure will add wrong examples to the labeled set leading to quick semantic drift (see Figure 9.11). The short-term tracking produces a set of labeled examples $\mathcal{L}_i$.

Our single object tracking computes sparse optical flow using Pyramidal Lucas Kanade [24] on Harris feature points. Since we start with a small set of labeled examples, and do not perform expensive per-frame detection, our detection prior is weak. To prevent tracking drift in such cases we incorporate color/texture consistency by using object proposal bounding boxes [289] obtained from a region around

the tracked box. We address two failure modes of tracking:

Drift due to spurious motion: This occurs while computing optical flow on feature points which are not on the object, *e.g.*, points on a moving background or occlusion. To correct this, we first divide each tracked box into four quadrants and compute the mean flow in each quadrant. We weigh points in each quadrant by their agreement with the flow in the other quadrants. The final dominant motion direction for the box is the weighted mean of the flow for each point. This simple and efficient scheme helps correct the motion of feature points not on the object.

Drift due to appearance change: This is incorporated by object detection boxes and object proposal bounding boxes in the trellis graph formulation described below.

We formulate the tracking problem as finding the min-cost path in a graph $\mathcal{G}$. At each frame we incorporate priors in terms of bounding boxes, *i.e.*, detection bounding boxes, tracked boxes and object proposal bounding boxes. These boxes are the nodes in our graph and we connect nodes in consecutive frames with edges forming a trellis graph. The edge weights are a linear combination of the difference in dominant motions of the boxes (described above), spatial proximity and area change. Tracking through this trellis graph $\mathcal{G}$ is the equivalent of finding the single min-cost path, and is efficiently computed using Dynamic-Programming [17, 214]. As post-processing, we cut the path as soon as the total cost exceeds a set threshold.

Selection algorithm

After we label thousands of boxes $\mathcal{L}_i$ for the current iteration, we use them for improving our object detectors. Since video data is highly redundant, we label few diverse examples and many redundant ones. Training a category detector on these thousands of (redundant) boxes, *from scratch*, in every iteration is suboptimal. We prefer an *incremental training* approach that makes incremental updates to the detector, *i.e.*, trains only on *newly added and diverse* examples rather than everything. This is especially important to prevent drift because even if we label thousands of wrong but redundant boxes, our method picks only a few of them. We find the exemplar detectors [114, 185] suitable for incremental learning as they are trained per bounding box.

For each labeled bounding box in $\mathcal{L}_i$, we compute a detection signature [134, 191, 301] using our exemplar detectors. Boxes where our current set of detectors do not give a high response correspond to examples which are not explained well by the existing set of detectors. Training on these boxes increases the coverage of our detectors. Thus, we compute similarity in this detection signature space to greedily select a set of boxes that are neither similar to our current detectors, nor

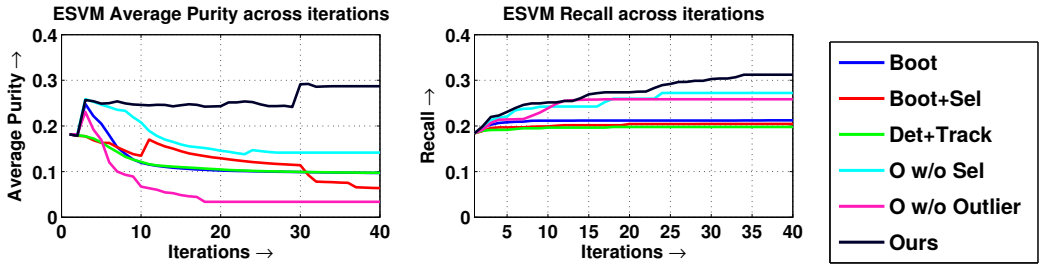


Figure 9.10 – We measure the detection performance of the ESVMs from each ablation method on our test set by computing (left) Average Purity and (right) Recall.

amongst themselves. At each iteration, we limit the number of boxes selected to 10. When this selection approach is unable to find new boxes, we conclude that we have reached the saturation point of our SSL. This serves as our *stopping criterion.*

9.2.4 Experiments and Results

Our algorithm has a fair number of components interacting with each other across iterations. It is difficult to characterize the importance of each component by using the whole system at once. For such component-wise characterization, we divide our experiments in two sets. Our first set of ablative experiments is on a small subset of videos from VIRAT [201]. In the second set of experiments, we demonstrate the scalability of our approach to a million frames from [201]. We also show the generalization of our method to a different dataset (KITTI [91]). In both these cases we evaluate the automatically labeled data in terms of quality, coverage (recall), diversity and relevance to training an object detector. We now describe the experimental setup which is common across all our experiments.

Datasets: Due to limited availability of large video datasets with bounding box annotations, we picked car as our object of interest, and two video datasets with a large number of cars and related objects like trucks, vans *etc.* We chose the VIRAT 2.0 Ground [201] dataset for its large number of frames, and *sparsely* annotated bounding boxes over all the frames. This dataset consists of long hours of surveillance videos (static camera) of roads and parking lots. It has 329 videos (∼1 million frames, and ∼6.5 million annotated bounding boxes (partial ground truth) of cars. We also evaluate on videos from the KITTI [91] dataset which were collected by a camera mounted on a moving car. We use the set 37 videos (∼12,500 frames) which have partial ground truth boxes (∼41,000 boxes, small cars are not annotated).

Dataset characteristics: We chose these datasets with very different character-

Table 9.3 – Comparison of our method with baselines as explained in Section 9.2.4. We train an LSVM [79] on all the automatically labeled data and compute its detection performance on a held-out, fully annotated test set (AP for IOU 0.5)

	Automatic Labeling (LSVM)						Ground Truth		
Iteration	Boot.	Boot.+Sel.	Det+Track	O w/o Sel	O w/o Outlier	**Ours**	Pascal LSVM	Pascal DPM	VIRAT LSVM
10	1.32	9.09	9.09	11.21	7.32	**15.39**	20.89	29.56	41.38
30	1.94	3.03	6.59	10.83	1.41	**17.68**			

istics (motion, size of object *etc.*) to test the generalization of our method. The VIRAT and KITTI datasets both consist of outdoor scene videos with a static and moving camera respectively. The VIRAT dataset captures surveillance videos of multiple cars in parking lots. The cars in this dataset are small compared to the frame size, tightly packed together and viewed from a distance (thus no drastic perspective effects). The KITTI dataset on the other hand, consists of videos taken by a vehicle mounted camera. It has high motion, large objects, and perspective effects. Figure 9.9 shows examples from both the datasets demonstrating their differences.

Detectors: We use the Exemplar-SVM (ESVM) [185] detectors with 5000 random images from Flickr as negatives [251]. Since per frame detection is expensive, we detect once every 30^{th} frame for VIRAT, and every 10^{th} frame for KITTI. We threshold detections at SVM score of -0.75.

Multiple feature spaces for false positive removal: We use the uLSIF [144] algorithm on Pyramidal HOG (PHOG) [23] and color histogram (LAB color with $32 \times 16 \times 16$ bins) features computed on a resized box of 200×300 px. We set the kernel bandwidth for uLSIF by computing the 75^{th} percentile distance between random pairs of points.

Object proposal windows: We obtain 2000 windows per image using selective search [289].

Ablative Analysis of each constraint

To tease apart the contributions of each component described in Section 9.2.2, we design a set of algorithms using only a subset of the components.

Bootstrapping (Boot): In this vanilla form of SSL, we train object detectors on the initial labeled set and perform detection. The confident detections are used as training examples for the next iteration detectors.

Bootstrapping with Selection (Boot+Sel): This algorithm builds upon the bootstrapping approach described above. However, diverse examples are selected

Figure 9.11 – (a) We look at a subset of the bounding boxes used to train ESVMs across iteration. Each row corresponds to an ablation method. The top row shows the randomly chosen initial positive bounding boxes (same for each method). The other methods diverge quickly across iterations, thus showing that constraints are very important for maintaining purity.

(Section 9.2.3) from confident detections to train new detectors.

Detection, Tracking and Clustering (Det+Track): In this algorithm, we use a basic combination of detection and tracking. We start tracking from the confident ESVM detections to label new examples. We then use WHO (or ELDA) [114] clustering [191] on these boxes to select training examples for the new detectors. For clustering, we use WHO features on labeled boxes after resizing, followed by k-means. We choose the best k in the range $(5, 10)$.

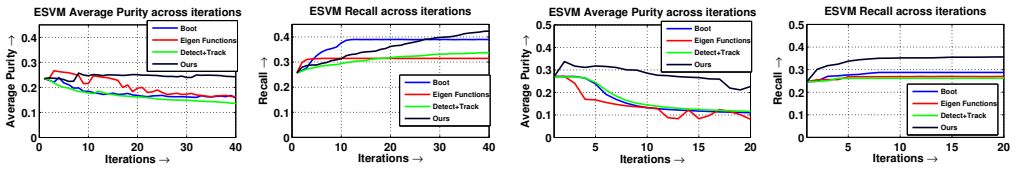
Ours without outlier (O w/o Outlier): This setup uses our entire algorithm except outlier removal (Section 9.2.3). It can roughly be thought of as Detection+Tracking+Selection.

Ours without selection (O w/o Sel): This algorithm uses our algorithm except selection (Section 9.2.3). It uses WHO clustering for selection like Det+Track.

Ours: We use our full algorithm as detailed in Sections 9.2.2.

Ablation dataset: For these set of experiments we use an input set of 25 videos (∼170,000 frames) and a separate test set of 17 videos (∼105,000 frames) which we fully annotated. All methods start with the same sparse labels (only 21 boxes spread across different videos). We run them iteratively till 30 iterations.

Qualitative Results. Figure 9.11 shows a random set of boxes labeled by each ablation method (and used to train ESVMs for that method), along with the initial set of 21 positive examples. We notice that as iterations proceed, the labeling quality (especially "tightness" of the boxes) for all methods degrades. More importantly, the other methods like Boot, Det+Track *etc.* show semantic drift (Figure 9.11 columns 2-5 at iteration 20). We also notice the importance of selection, as Ours w/o Selection loses good localization ability fairly quickly (Figure 9.11 column 6 at iterations 10-30). We believe methods like [53] can be used to further improve the labeling quality.



(a) Purity on KITTI **(b)** Recall on KITTI **(c)** Purity on VIRAT **(d)** Recall on VIRAT

Figure 9.12 – We measure the detection performance of the labeled boxes for our large scale experiments. We test the ESVMs trained at each iteration on the held out test set and compute Average Purity and Recall. Our method outperforms the baselines by a significant margin. It maintains purity while substantially increasing recall.

ESVM Detection performance. For the input videos, we cannot measure labeling purity because of partial ground truth. Instead, we measure the relevance of labeled boxes to detection. We consider detection performance on the test set as a proxy for good labeling. We test the ESVMs selected and trained by each method across iterations on the held out test set. A good labeling would result in an increased detection performance. Figure 9.10 shows Average Purity vs. Recall across iterations for the various methods on the test set. We use Average Purity, which is same as Average Precision [54] but does not penalize double-detections, since we are more interested in whether the ESVMs are good detectors individually, rather than as an ensemble.We consider an instance correctly labeled (or pure) if its Intersection-Over-Union (IOU) with any ground-truth instance is greater than 0.3. Our method shows a higher purity and recall, pointing towards meaningful labeling and selection of the input data. It also shows that every component of our method is crucial for getting good performance. We stop our method at iteration 40 because of our stopping criterion (Section 9.2.3). We got a 2 point drop in purity from iteration 40 to 45, proving the validity of our stopping criterion. This is important since our algorithm would rather saturate than label noisy examples.

Training on all the automatically labeled data. In this Section, we evaluate the effectiveness of all our labeled data. For each algorithm, we **train an LSVM** [79] (only root filters, mixtures and positive latent updates of DPM [79]) on the data it labeled, and test it on the held-out test set. Since it is computationally expensive to train an LSVM on the thousands of boxes, we subsample the labeled boxes (5000 boxes in total for each method using k-means on WHO [114] features). We sample more boxes from the earlier iterations of each method, because their labeling purity decreases across iterations (Figure 9.10). We use the same domain independent negatives [251] for all these LSVMs (left side in Table 9.3). Table 9.3 shows the detection AP performance of all **LSVMs** (measured at IOU of 0.5 [54]) for the

Figure 9.13 – We look at the selected positives for each baseline method across iterations for both KITTI and VIRAT datasets. We notice that the purity of the labeled set drops significantly as iterations proceed. This indicates that constraints specific to video are needed to learn meaningfully over iterations.

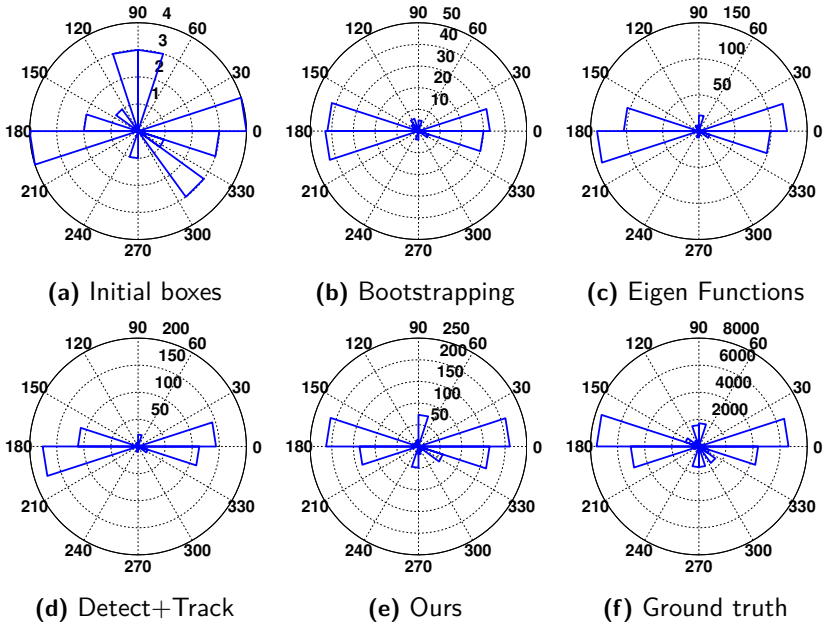
data labeled at iteration 10 and 30. We see that LSVM trained on our labeled data outperforms all other LSVMs. Our performance is close to that of an LSVM trained on the PASCAL VOC 2007 dataset [54]. This validates the high quality of our automatically labeled data.

We also note that the performance of an LSVM trained on the ground truth boxes (VIRAT-LSVM) (5000 boxes from $\sim$1 million ground truth boxes using the same k-means as above) achieves twice the performance. The fact that all LSVMs (except the ones from PASCAL) are trained with the same domain-independent negatives, indicates that the lack of domain-negatives is not the major cause of this limitation. This suggests that automatic labeling has limitations compared to human annotations. On further scrutiny of the LSVM trained on our automatically labeled data, we found that the recall saturates after iteration 30. However, the precision was within a few points of VIRAT-LSVM. Since we work with only the confident detections/tracklets in the high precision/low recall regime, this behavior is not unexpected. This is also important since our algorithm would rather saturate than label noisy positives.

Large scale experiments

In this Section, we evaluate the scalability of our algorithm to millions of frames and object instances. We also test its generalization on two datasets with widely different characteristics - VIRAT (static camera, small cars tightly packed) and KITTI (moving camera, high motion, large cars with perspective effects). We use the *Boot* and *Ours* methods described in Section 9.2.4. As we described in Section 9.2.1, most of the existing approaches make additional assumptions that are not applicable in our setting, or do not have publicly available code. To make a fair comparison against existing methods, we adapt them to our setting.

Baseline - Dense detection and association (Detect + Track): This algorithm is inspired by Geiger et al. [92] which has state-of-the-art results on KITTI. The original algorithm uses per-frame detections, followed by a Kalman filter and data association. We make two changes - 1) To have a comparable detector across



(a) Initial boxes **(b)** Bootstrapping **(c)** Eigen Functions

(d) Detect+Track **(e)** Ours **(f)** Ground truth

Figure 9.14 – Pose variation in automatic labeling of the KITTI dataset. For each algorithm, we plot the 3D pose distribution of all the boxes it labels after 30 iterations. The first and last plots show pose distribution for the initial labeled boxes and all boxes in ground truth respectively. The distribution of boxes labeled by our method is close to the ground truth distribution.

methods, we do not use a pre-trained DPM on PASCAL VOC 2007. We substitute it with the ESVMs we have at the current iteration. 2) We do not use a Kalman filter and use data association over a short term (maximum 300 frames). We select positives for the ESVMs by k-means on WHO features [191].

Baseline - Eigen Functions: We modify the Eigen functions [81] method which was originally designed for image classification. This method uses distances over manifolds to label unlabeled data. We use basic detection and short term tracking to get a set of bounding boxes and use eigen functions to classify them as positive or background. The boxes from previous iterations on which we trained ESVMs are used as positives and random images from Flickr [251] as negative data. We use color histogram ($32 \times 16 \times 16$ LAB space) and PHOG [23] as input to eigen functions.

Datasets: Our input set consists of 312 videos (∼820,000 frames) from VIRAT. We take a held out test set of 17 videos (∼105,000 frames) which we fully annotated. As input, all algorithms start with the same sparse labels consisting of 43 randomly chosen bounding boxes across different videos. For the KITTI dataset we use 30 videos (∼10,000 frames) as our input and 7 videos (∼2000 frames) for testing. All

methods start with the same sparse labels (25 boxes from different videos).

Qualitative Results: We first present qualitative results in Figure 9.13. We notice the same trends as we did in the ablation analysis, namely, bounding boxes tend to get less tight across iterations. For the baseline methods, we notice quick divergence as an increasing number of background patches are classified as car.

ESVM Detection Performance: Following the approach outlined in Section 9.2.4, we compute the detection performance of the ESVMs on the held out test set. This helps us measure the relevance of our labeling to the detection task. Figure 9.12 shows the results of these experiments. We notice that our method outperforms the baselines on both the metrics (Average Purity and Recall). This reinforces the fact that our constraints help arrest semantic drift.

Diversity of labeling: The KITTI dataset provides the 3D pose associated with each labeled car. We use this 3D pose as a proxy for estimating the diversity of our labeled set. In this experiment, we compute the pose of the examples labeled by all methods. Figure 9.14 demonstrates that our labeling procedure covers a diverse range of poses as opposed to baseline methods. The pose distribution of our labeling is closer to the ground truth distribution, while that of the baselines prefers the more "popular" poses, i.e., front/back of cars. Combined with the results of Figure 9.12, this points towards a diverse, and high quality labeling of the data.

Conclusion and Discussion

This dissertation follows a two-pronged strategy for discovering and leveraging the structure in visual data. We demonstrate that it is beneficial to design computational algorithms for visual recognition from the perspective of what underlying structure they can capture, and that explicitly utilizing this structure is essential in leveraging large-scale visual data without large-scale human-annotations.

In Part I of this dissertation, we focus on supervised visual recognition, where we demonstrate that models and learning algorithms that are better at capturing and leveraging visual structure are better at recognition. Taking inspiration from the human visual pathway, Chapters 2 and 3 propose computation models that incorporate context, feedback, and top-down information for recognition. These works are initial attempts at incorporating top-down structure in bottom-up, feedforward ConvNet-based models, and provide a direction of future inquiry. For training these ConvNet-based object recognition systems, in Chapter 4, we propose a novel optimization method that leverages recognition-specific problem structure. This method results in better training convergence and consistent improvements in recognition rates. We believe that developing supervised learning algorithms, which lead to better generalizations or higher accuracy, will remain important in the future. In [300], we are also exploring adversarial training strategies for such recognition systems that generalize better to uncommon occlusions and deformations of objects.

Moving beyond the standard setup for object recognition, Chapter 5 looks at how the underlying geometry of an object can help impose structure while training recognition models and how this structure can be used for deeper 3D understanding of objects just from 2D images. We believe that depth, as a source of information, has been under-appreciated in the current generation of recognition systems, and future research needs to explore ways of better utilizing depth. An example is proposed in Chapter 6, which focuses on the generic multi-task learning setup. It provides a glimpse of how we can design better architectures that can utilize the

shared structure between multiple tasks (*e.g.*, between semantic understanding and geometric understanding).

In Part II, we venture beyond the standard supervised recognition regime. We show that from large amounts of visual data it is possible to discover the underlying structure automatically, and exploiting this structure is an essential ingredient in developing large-scale recognition algorithms without requiring large-scale human annotation efforts. In Chapter 7, we begin with the simplest setting of only having a single exemplar for each concept and propose a similarity metric induced by the low-level structure in large amounts of unlabeled images. We show that this metric can be used for better image matching/retrieval across domains, exploring large image collections, and discovering object categories from weakly-supervised web images. It will be interesting to explore more ways of exploiting this low-level structure in the future (*e.g.*, for improving initializations and training of ConvNet-based models).

Finally, we tackle the problem of effectively utilizing large amounts of unlabeled data along with a small amount of labeled data, *i.e.*, the semi-supervised learning (SSL) paradigm. We propose a constrained-SSL framework in Chapter 8, which uses the semantic structure in visual data as constraints to reliably utilize millions of unlabeled images. In Chapter 9, we use this framework to develop large-scale systems for visual recognition that can automatically discover and exploit useful structure from millions of images (Section 9.1) and videos (Section 9.2). We believe that videos, as a source for understanding the visual structure, are still underutilized; and future research should pursue discovery of richer structure (such as actions, causal relationships) from internet-scale videos, as well as videos from real-world robotics applications (*e.g.*, self-driving cars, drones).

In general, Part II is only scratching the surface of the potential of utilizing large-scale, unlabeled visual data. One of the major driving forces behind rapid success on the supervised recognition paradigm (*cf.* Part I) was the availability of standardized apparatus – datasets, benchmarks, and tools for analysis. Moving forward, other learning paradigms for recognition (unsupervised, semi-supervised, weakly-supervised, noisily-supervised, *etc.*) will also require similar benchmarks and analysis tools. This will include testbeds for comparing improvements in recognition systems with varying amounts of unlabeled data and tools to analyze key aspects of various learning paradigms (*e.g.*, for SSL we studied extent and categorization of semantic drift, coverage, and diversity in Chapters 8 and 9). We believe this will be critical for rapid and reproducible advancements in large-scale visual recognition.

> Today the world changes so quickly that in growing up we take leave not just of youth but of the world we were young in.
>
> Sir Peter Medawar

Broader Context. It is also important to consider ideas in this dissertation in the broader context of the field. The field of Visual Recognition has changed dramatically and rapidly over the duration of this thesis, a fact that was both exciting and frustrating. Consider the computational systems used throughout thesis, where different Chapters use different recognition models each of which was state-of-the-art when the work was being conducted (DPM [79] in Chapters 5 and 9, Fast R-CNN [95] in Chapters 4 and 6, Faster R-CNN [223] in Chapters 2 and 3, *etc.*). As the field moves forward, these methods will, inevitably, be replaced by newer and better variants. We believe that the ideas presented in this thesis (such as incorporating structure via top-down contextual information and feedback, enforcing structure using other modalities, learning and utilizing knowledge) will provide guidance in developing future generations of recognition systems.

In conclusion, this thesis highlights the importance of consciously leveraging structure of our visual world while developing computational models for visual recognition. It is our hope that this thesis will inspire others, and provide them with the insights and tools required to develop better recognition methods.

> Now this is not the end.
> It is not even the beginning of the end.
> But it is, perhaps, the end of the beginning.
>
> Sir Winston Churchill

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] A. H. Abdulnabi, G. Wang, J. Lu, and K. Jia. Multi-task CNN model for attribute prediction. *IEEE Multimedia*, 17, 2015.

[3] B. Alexe, T. Deselaers, and V. Ferrari. Classcut for unsupervised class segmentation. In *European Conference on Computer Vision*, 2010.

[4] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[5] Y. Amit, M. Fink, N. Srebro, and S. Ullman. Uncovering shared structures in multiclass classification. In *International Conference on Machine Learning*, 2007.

[6] V. S. Anastasia Pentina and C. H. Lampert. Curriculum learning of multiple tasks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[7] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[8] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Journal of Machine Learning Research*, 73, 2008.

[9] S. Avidan. Ensemble tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[10] H. Azizpour and I. Laptev. Object detection using strongly-supervised deformable part models. In *European Conference on Computer Vision*, 2012.

[11] T. Bachmann. A hidden ambiguity of the term "feedback" in its use as an explanatory mechanism for psychophysical visual phenomena. *Feedforward and Feedback Processes in Vision*, 2015.

[12] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[13] S. Bae, A. Agarwala, and F. Durand. Computational rephotography. *ACM Trans. Graph.*, 2010. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1805964.1805968.

[14] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing, 1999.

[15] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. iCoseg: Interactive co-segmentation with intelligent scribble guidance. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[16] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[17] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[18] T. Berg and D. Forsyth. Animals on the web. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[19] I. Biederman. *On the semantics of a glance at a scene*. Lawrence Erlbaum, 1981.

[20] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, 1998.

[21] L. Bo, K. Lai, X. Ren, and D. Fox. Object recognition with hierarchical kernel descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[22] O. Boiman and M. Irani. Detecting irregularities in images and in video. In *International Journal of Computer Vision*, 2007.

[23] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[24] J.-Y. Bouget. Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm, 2000.

[25] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3D human pose annotations. In *IEEE International Conference on Computer Vision*, 2009.

[26] S. Branson, S. Belongie, and P. Perona. Strong supervision from weak annotation: Interactive training of deformable part models. In *IEEE International Conference on Computer Vision*, 2011.

[27] R. Brooks. Symbolic reasoning among 3D models and 2D images. *Artificial Intelligence*, 1981.

[28] R. Brooks, R. Creiner, and T. Binford. The acronym model-based vision system. *IJCAI*, 1978.

[29] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[30] A. Carlson, J. Betteridge, E. R. H. Jr., and T. M. Mitchell. Coupling semi-supervised learning of categories and relations. In *NAACL HLT Workskop on SSL for NLP*, 2009.

[31] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

[32] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[33] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[34] R. Caruana. Multitask learning. *Machine learning*, 28, 1997.

[35] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2011.

[36] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

[37] D.-J. Chen, H.-T. Chen, and L.-W. Chang. Video object cosegmentation. In *ACM MM*, 2012.

[38] J. Chen, J. Zhou, and J. Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *SIGKDD*, 2011.

[39] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

[40] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu. Sketch2photo: internet image montage. *ACM Trans. Graph.*, 28, 2009.

[41] X. Chen, A. Shrivastava, and A. Gupta. NEIL: Extracting visual knowledge from web data. In *IEEE International Conference on Computer Vision*, 2013.

[42] X. Chen, A. Shrivastava, and A. Gupta. Enriching visual knowledge bases via object discovery and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[43] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. H. S. Torr. BING: Binarized normed gradients for objectness estimation at 300fps. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[44] H. Chiu, L. Kaelbling, and T. Lozano-Perez. Virtual training for multi-view object class recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[45] J. Choi, M. Rastegari, A. Farhadi, and L. Davis. Adding unlabeled samples to categories by learned attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[46] H. Chong, S. Gortler, and T. Zickler. A perception-based color space for illumination-invariant image processing. In *Proceedings of SIGGRAPH*, 2008.

[47] M. M. Chun and Y. Jiang. Top-down attentional guidance based on implicit learning of visual covariation. *Psychological Science*, 1999.

[48] R. G. Cinbis, J. Verbeek, and C. Schmid. Segmentation driven object detection with Fisher vectors. In *IEEE International Conference on Computer Vision*, 2013.

[49] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *International Conference on Machine Learning*, 2008.

[50] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12, 2011.

[51] J. R. Curran, T. Murphy, and B. Scholz. Minimising semantic drift with mutual exclusion bootstrapping. In *Conference of the Pacific Association for Computational Linguistics*, 2007.

[52] D. Dai and L. Van Gool. Ensemble projection for semi-supervised image classification. In *IEEE International Conference on Computer Vision*, 2013.

[53] Q. Dai and D. Hoiem. Learning to localize detected objects. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[54] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[55] K. Dale, M. K. Johnson, K. Sunkavalli, W. Matusik, and H. Pfister. Image restoration using online photo collections. In *IEEE International Conference on Computer Vision*, 2009.

[56] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 2008.

[57] J. L. Davenport and M. C. Potter. Scene consistency in object and background perception. *Psychological Science*, 2004.

[58] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[59] S. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert. An empirical study of context in object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[60] S. Divvala, A. Efros, and M. Hebert. How important are 'deformable parts' in the deformable parts model? In *European Conference on Computer Vision, Parts and Attributes Workshop*, 2012.

[61] S. Divvala, A. Farhadi, and C. Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[62] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *British Machine Vision Conference*, 2009.

[63] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[64] J. Dong, Q. Chen, S. Yan, and A. Yuille. Towards unified object detection and semantic segmentation. In *2014*, 2014.

[65] S. Ebert, D. Larlus, and B. Schiele. Extracting structures in image collections for object recognition. In *European Conference on Computer Vision*, 2010.

[66] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH*, Computer Graphics Proceedings, Annual Conference Series, 2001.

[67] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *IEEE International Conference on Computer Vision*, 2015.

[68] M. Eitz, K. Hildebrand, T. Boubekeur, and M. Alexa. Sketch-based image retrieval: benchmark and bag-of-features descriptors. *IEEE TVCG*, 2010.

[69] I. Endres and D. Hoiem. Category independent object proposals. In *European Conference on Computer Vision*, 2010.

[70] I. Endres, V. Srikumar, M.-W. Chang, and D. Hoiem. Learning shared body-plans. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[71] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[72] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 2010.

[73] A. Evgeniou and M. Pontil. Multi-task feature learning. *Conference on Neural Information Processing Systems*, 19:41, 2007.

[74] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *SIGKDD*, 2004.

[75] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[76] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[77] A. Farhadi, I. Endres, and D. Hoiem. Attribute-centric recognition for cross-category generalization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[78] D. J. Felleman and D. C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex*, 1991.

[79] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[80] R. Fergus, P. Perona, and A. Zisserman. A visual category filter for Google images. In *European Conference on Computer Vision*, 2004.

[81] R. Fergus, Y. Weiss, and A. Torralba. Semi-supervised learning in gigantic image collections. In *Conference on Neural Information Processing Systems*, 2009.

[82] S. Fidler, S. Dickinson, and R. Urtasun. 3D object detection and viewpoint estimation with a deformable 3D cuboid model. In *Conference on Neural Information Processing Systems*, 2012.

[83] S. Fidler, R. Mottaghi, A. Yuille, and R. Urtasun. Bottom-up segmentation for top-down detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[84] P. Fischer, A. Dosovitskiy, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015.

[85] D. F. Fouhey, A. Gupta, and M. Hebert. Data-driven 3D primitives for single image understanding. In *IEEE International Conference on Computer Vision*, 2013.

[86] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. *IEEE Computer Graphics Applications*, 2002.

[87] H. Fu, D. Xu, B. Zhang, and S. Lin. Object-based multiple foreground video co-segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[88] C. Galleguillos and S. Belongie. Context based object categorization: A critical survey. *CVIU*, 2010.

[89] C. Gatta, A. Romero, and J. van de Veijer. Unrolling loopy top-down semantic feedback in convolutional deep networks. In *CVPR Workshops*, 2014.

[90] A. Gazzaley and A. C. Nobre. Top-down modulation: bridging selective attention and working memory. *Trends in cognitive sciences*, 2012.

[91] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[92] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *IJRR*, 2013.

[93] S. Gidaris and N. Komodakis. Object detection via a multi-region & semantic segmentation-aware cnn model. In *IEEE International Conference on Computer Vision*, 2015.

[94] C. D. Gilbert and M. Sigman. Brain states: top-down influences in sensory processing. *Neuron*, 2007.

[95] R. Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision*, 2015.

[96] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[97] G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik. R-CNNs for pose estimation and action detection. *arXiv preprint arXiv:1406.5212*, 2014.

[98] G. Gkioxari, R. Girshick, and J. Malik. Contextual action recognition with RCNN. In *IEEE International Conference on Computer Vision*, 2015.

[99] D. Glasner, M. Galun, S. Alpert, R. Basri, and G. Shakhnarovich. Viewpoint-aware object detection and pose estimation. In *IEEE International Conference on Computer Vision*, 2011.

[100] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feed-forward neural networks. In *AISTATS*, 2010.

[101] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *European Conference on Computer Vision*, 2008.

[102] C. Gu and X. Ren. Discriminative mixture-of-templates for viewpoint classification. In *European Conference on Computer Vision*, 2010.

[103] C. Gu, J. J. Lim, P. Arbeláez, and J. Malik. Recognition using regions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[104] Q. Gu and J. Zhou. Learning the shared subspace for multi-task clustering and transductive transfer classification. In *ICDM*, 2009.

[105] M. Guillaumin, J. Verbeek, and C. Schmid. Multimodal semi-supervised learning for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[106] J. Guo, Z. Li, L.-F. Cheong, and S. Z. Zhou1. Video co-segmentation for meaningful action extraction. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[107] A. Gupta and L. S. Davis. Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In *European Conference on Computer Vision*, 2008.

[108] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[109] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, 2014.

[110] S. Gupta, B. Hariharan, and J. Malik. Exploring person context and local scene context for object detection. *arXiv preprint arXiv:1511.08177*, 2015.

[111] Y. HaCohen, R. Fattal, and D. Lischinski. Image upsampling via texture hallucination. In *ICCP*, 2010.

[112] S. Hare, A. Saffari, and P. Torr. Struck: Structured output tracking with kernels. In *IEEE International Conference on Computer Vision*, 2011.

[113] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *IEEE International Conference on Computer Vision*, 2011.

[114] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. In *European Conference on Computer Vision*, 2012.

[115] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, 2014.

[116] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[117] J. Hays and A. A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics (SIGGRAPH)*, 2007.

[118] J. Hays and A. A. Efros. im2gps: estimating geographic information from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[119] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[120] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[121] V. Hedau, D. Hoiem, and D. Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. In *European Conference on Computer Vision*, 2010.

[122] G. Heitz and D. Koller. Learning spatial context: Using stuff to find things. In *European Conference on Computer Vision*, 2008.

[123] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin. Image analogies. In *SIGGRAPH*, 2001.

[124] H. S. Hock, G. P. Gordon, and R. Whitehurst. Contextual relations: the influence of familiarity, physical plausibility, and belongingness. *Perception & Psychophysics*, 1974.

[125] D. Hoiem, R. Sukthankar, H. Schneiderman, and L. Huston. Object-based image retrieval using the statistical structure of images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[126] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *International Journal of Computer Vision*, 2007.

[127] A. Hollingworth. Does consistent scene context facilitate object perception? *Journal of Experimental Psychology: General*, 1998.

[128] J. B. Hopfinger, M. H. Buonocore, and G. R. Mangun. The neural mechanisms of top-down attentional control. *Nature neuroscience*, 2000.

[129] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[130] J. Hupe, A. James, B. Payne, S. Lomber, P. Girard, and J. Bullier. Cortical feedback improves discrimination between figure and background by v1, v2 and v3 neurons. *Nature*, 1998.

[131] L. Itti and C. Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 2000.

[132] A. Jalali, S. Sanghavi, C. Ruan, and P. K. Ravikumar. A dirty model for multi-task learning. In *Conference on Neural Information Processing Systems*, 2010.

[133] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European Conference on Computer Vision*, 2008.

[134] L. Jia, H. Su, E. P. Xing, and L. Fei-fei. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *Conference on Neural Information Processing Systems*, 2010.

[135] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[136] M. K. Johnson, K. Dale, S. Avidan, H. Pfister, W. T. Freeman, and W. Matusik. CG2real: Improving the realism of computer generated images using a large collection of photographs. *IEEE TVCG*, 2010.

[137] A. L. Jose C. Rubio, Joan Serrat. Video cosegmentation. In *ACCV*, 2012.

[138] A. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class active learning for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[139] A. Joulin, F. Bach, and J. Ponce. Discriminative clustering for image cosegmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[140] A. Joulin, F. Bach, and J. Ponce. Multi-class cosegmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[141] T. Judd, K. Ehinger, F. Durand, and A. Torralba. Learning to predict where humans look. In *IEEE International Conference on Computer Vision*, 2009.

[142] J. Jung, H. Yim, B. Yoo, C. Choi, D. Park, and J. Kim. Rotating your face using multi-task deep neural network. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[143] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[144] T. Kanamori, S. Hido, and M. Sugiyama. A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, 2009.

[145] B. Kaneva, J. Sivic, A. Torralba, S. Avidan, and W. T. Freeman. Infinite images: Creating and exploring a large photorealistic virtual space. *Proceedings of the IEEE*, 2010.

[146] H. Kang, M. Hebert, and T. Kanade. Discovering object instances from scenes of daily living. In *IEEE International Conference on Computer Vision*, 2011.

[147] Z. Kang, K. Grauman, and F. Sha. Learning with whom to share in multi-task feature learning. In *International Conference on Machine Learning*, 2011.

[148] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active learning with gaussian processes for object categorization. In *IEEE International Conference on Computer Vision*, 2007.

[149] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[150] I. Kemelmacher-Shlizerman, E. Shechtman, R. Garg, and S. M. Seitz. Exploring photobios. In *ACM Transactions on Graphics (SIGGRAPH)*, 2011.

[151] A. R. Z. Khurram Soomro and M. Shah. Ucf101: A dataset of 101 human action classes from videos in the wild. Technical report, CRCV-TR-12-01, 2012.

[152] G. Kim, E. P. Xing, L. Fei-Fei, and T. Kanade. Distributed Cosegmentation via Submodular Optimization on Anisotropic Diffusion. In *IEEE International Conference on Computer Vision*, 2011.

[153] P. Krähenbühl and V. Koltun. Geodesic object proposals. In *European Conference on Computer Vision*, 2014.

[154] D. J. Kravitz, K. S. Saleem, C. I. Baker, L. G. Ungerleider, and M. Mishkin. The ventral visual pathway: an expanded neural framework for the processing of object quality. *Trends in cognitive sciences*, 2013.

[155] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Conference on Neural Information Processing Systems*, 2012.

[156] D. Kuettel, M. Guillaumin, and V. Ferrari. Segmentation propagation in imagenet. In *European Conference on Computer Vision*, 2012.

[157] S. Lad and D. Parikh. Interactively guiding semi-supervised clustering via attribute-based explanations. In *European Conference on Computer Vision*, 2014.

[158] L. Ladickỳ, P. Sturgess, K. Alahari, C. Russell, and P. H. Torr. What, where and how many? combining object detectors and crfs. In *European Conference on Computer Vision*, 2010.

[159] K. Lai, L. Bo, X. Ren, and D. Fox. A scalable tree-based approach for joint object and pose recognition. In *AAAI*, 2011.

[160] K. Lai, L. Bo, X. Ren, and D. Fox. Sparse distance learning for object recognition combining RGB and depth information. In *ICRA*, 2011.

[161] V. A. Lamme and P. R. Roelfsema. The distinct modes of vision offered by feedforward and recurrent processing. *Trends in neurosciences*, 2000.

[162] V. A. Lamme, H. Super, and H. Spekreijse. Feedforward, horizontal, and feedback processing in the visual cortex. *Current opinion in neurobiology*, 1998.

[163] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[164] M. Lapin, B. Schiele, and M. Hein. Scalable multitask representation learning for scene classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[165] S. Lazebnik, C. Schmid, and J. Ponce. Spatial pyramid matching. In *Object Categorization: Computer and Human Vision Perspectives*. Cambridge University Press, 2009.

[166] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989.

[167] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *IEEE*, 1998.

[168] Y. J. Lee and K. Grauman. Collect-cut: Segmentation with top-down cues discovered in multi-object images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[169] Y. J. Lee, J. Ghosh, and K. Grauman. Discovering important people and objects for egocentric video summarization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[170] K. Li, B. Hariharan, and J. Malik. Iterative instance segmentation. *arXiv preprint arXiv:1511.08498*, 2015.

[171] L.-J. Li, G. Wang, and L. Fei-Fei. OPTIMOL: Automatic object picture collection via incremental model learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[172] X. Liang, S. Liu, Y. Wei, L. Liu, L. Lin, and S. Yan. Computational baby learning. *arXiv preprint*, 2014.

[173] J. Lim, R. Salakhutdinov, and A. Torralba. Transfer learning by borrowing examples for multiclass object detection. In *Conference on Neural Information Processing Systems*, 2011.

[174] G. Lin, C. Shen, I. Reid, et al. Efficient piecewise training of deep structured models for semantic segmentation. *arXiv preprint arXiv:1504.01013*, 2015.

[175] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, 2014.

[176] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. *arXiv preprint arXiv:1612.03144*, 2016.

[177] D. Liu, G. Hua, and T. Chen. A hierarchical visual model for video object summarization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[178] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015.

[179] X. Liu, J. Gao, X. He, L. Deng, K. Duh, and Y.-Y. Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. *NAACL*, 2015.

[180] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[181] I. Loshchilov and F. Hutter. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015.

[182] D. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 1987.

[183] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[184] T. Malisiewicz and A. A. Efros. Beyond categories: The visual memex model for reasoning about object relationships. In *Conference on Neural Information Processing Systems*, 2009.

[185] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *IEEE International Conference on Computer Vision*, 2011.

[186] T. Malisiewicz, A. Shrivastava, A. Gupta, and A. A. Efros. Exemplar-svms for visual ob ject detection, label transfer and image retrieval. In *International Conference on Machine Learning (Invited)*, 2012.

[187] D. Marr and H. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proc. Roy. Soc.*, 1978.

[188] M. Marszałek, I. Laptev, and C. Schmid. Actions in context. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[189] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.

[190] Y. Meng, X. Ye, and B. D. Gonsalves. Neural processing of recollection, familiarity and priming at encoding: Evidence from a forced-choice recognition paradigm. *Brain research*, 2014.

[191] I. Misra, A. Shrivastava, and M. Hebert. Data-driven exemplar model selection. In *IEEE Winter Conference on Applications of Computer Vision*, 2014.

[192] I. Misra, A. Shrivastava, and M. Hebert. Watch and learn: Semi-supervised learning of object detectors from videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[193] I. Misra*, A. Shrivastava*, A. Gupta, and M. Hebert. Cross-stitch Networks for Multi-task Learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[194] V. Mnih, N. Heess, A. Graves, et al. Recurrent models of visual attention. In *Conference on Neural Information Processing Systems*, 2014.

[195] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[196] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[197] K. Murphy, A. Torralba, W. Freeman, et al. Using the forest to see the trees: a graphical model relating features, objects and scenes. *Conference on Neural Information Processing Systems*, 2003.

[198] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *IEEE International Conference on Computer Vision*, 2015.

[199] G. Obozinski, B. Taskar, and M. Jordan. Multi-task feature selection. *Statistics Department, UC Berkeley, Tech. Rep*, 2006.

[200] G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20, 2010.

[201] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. K. Aggarwal, H. Lee, L. Davis, E. Swears, X. Wang, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, B. Song, A. Fong, A. Roy-Chowdhury, and M. Desai. A large-scale benchmark dataset for event recognition in surveillance video. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[202] A. Oliva and A. Torralba. Building the gist of a scene: the role of global image features in recognition. *Progress in Brain Research*, 2006.

[203] A. Oliva and A. Torralba. The role of context in object recognition. *Trends in cognitive sciences*, 2007.

[204] S. E. Palmer. The effects of contextual scenes on the identification of objects. *Memory & Cognition*, 1975.

[205] S. J. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[206] Y. Pang and H. Ling. Finding the best from the second bests - inhibiting subjective bias in evaluation of visual tracking algorithms. In *IEEE International Conference on Computer Vision*, 2013.

[207] D. Parikh and K. Grauman. Relative attributes. In *IEEE International Conference on Computer Vision*, November 2011.

[208] G. Patterson and J. Hays. SUN attribute database: Discovering, annotating, and recognizing scene attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[209] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Teaching 3D geometry to deformable part models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[210] P. Perona. Visions of a Visipedia. *Proceedings of IEEE*, 2010.

[211] V. Piëch, W. Li, G. N. Reeke, and C. D. Gilbert. Network model of top-down influences on local gain and contextual interactions in visual cortex. *Proceedings of the National Academy of Sciences*, 2013.

[212] P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to segment object candidates. In *Conference on Neural Information Processing Systems*, 2015.

[213] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollar. Learning to refine object segments. *arXiv preprint arXiv:1603.08695*, 2016.

[214] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[215] A. Prakash and D. Parikh. Attributes for classifier feedback. In *European Conference on Computer Vision*, 2012.

[216] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[217] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, and H.-J. Zhang. Two-dimensional active learning for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[218] A. Quattoni, M. Collins, and T. Darrell. Transfer learning for image classification with sparse prototype representations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[219] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *IEEE International Conference on Computer Vision*, 2007.

[220] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. *arXiv preprint arXiv:1611.00850*, 2016.

[221] M. Rastegari, A. Farhadi, and D. Forsyth. Attribute discovery via predictable discriminative binary codes. In *European Conference on Computer Vision*, 2012.

[222] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *CVPR Workshop*, 2014.

[223] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Conference on Neural Information Processing Systems*, 2015.

[224] E. Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI*, 1999.

[225] B. Romera-Paredes, A. Argyriou, N. Berthouze, and M. Pontil. Exploiting unrelated tasks in multi-task learning. In *Proceedings of Machine Learning Research*, 2012.

[226] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015.

[227] C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. In *IEEE Winter Conference on Applications of Computer Vision*, 2005.

[228] S. Ross, D. Munoz, M. Hebert, and J. A. Bagnell. Learning message-passing inference machines for structured prediction. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[229] C. Rother, T. Minka, A. Blake, and V. Kolmogorov. Cosegmentation of image pairs by histogram matching - incorporating a global constraint into MRFs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[230] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998.

[231] M. Rubinstein, A. Joulin, J. Kopf, and C. Liu. Unsupervised joint object discovery and segmentation in internet images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[232] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115, 2015.

[233] B. Russell, W. Freeman, A. Efros, J. Sivic, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[234] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 2009.

[235] B. C. Russell, J. Sivic, J. Ponce, and H. Dessales. Automatic alignment of paintings and photographs depicting a 3d scene. In *3D Representation and Recognition (3dRR)*, 2011.

[236] A. Saffari, C. Leistner, M. Godec, and H. Bischof. Robust multi-view boosting with priors. In *European Conference on Computer Vision*, 2010.

[237] S. Savarese and L. Fei-Fei. 3D generic object categorization, localization and pose estimation. In *IEEE International Conference on Computer Vision*, 2007.

[238] H. Schneiderman and T. Kanade. Object detection using the statistics of parts. *International Journal of Computer Vision*, 56, 2004.

[239] A. Schodl, R. Szeliski, D. H. Salesin, and I. Essa. Video textures. In *SIGGRAPH*, 2000.

[240] F. Schroff, A. Criminisi, and A. Zisserman. Harvesting image databases from the web. In *IEEE International Conference on Computer Vision*, 2007.

[241] A. G. Schwing and R. Urtasun. Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, 2015.

[242] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[243] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2015.

[244] F. Shahbaz Khan, R. Anwer, J. van de Weijer, A. Bagdanov, M. Vanrell, and A. Lopez. Color attributes for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[245] V. Sharmanska, N. Quadrianto, and C. H. Lampert. Augmented attribute representations. In *European Conference on Computer Vision*, 2012.

[246] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2007.

[247] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22, 2000.

[248] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision*, 2006.

[249] A. Shrivastava and A. Gupta. Building Part-based Object Detectors via 3D Geometry. In *IEEE International Conference on Computer Vision*, 2013.

[250] A. Shrivastava and A. Gupta. Contextual priming and feedback for Faster R-CNN. In *European Conference on Computer Vision*, 2016.

[251] A. Shrivastava, T. Malisiewicz, A. Gupta, and A. A. Efros. Data-driven visual similarity for cross-domain image matching. *ACM Trans. on Graphics*, 2011.

[252] A. Shrivastava, S. Singh, and A. Gupta. Constrained semi-supervised learning using attributes and comparative attributes. In *European Conference on Computer Vision*, 2012.

[253] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[254] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond Skip Connections: Top-Down Modulation for Object Detection. *arXiv:1612.06851*, 2016.

[255] B. Siddiquie and A. Gupta. Beyond active noun tagging: Modeling contextual interactions for multi-class active learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[256] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor Segmentation and Support Inference from RGBD Images. In *European Conference on Computer Vision*, 2012.

[257] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, and F. Moreno-Noguer. Fracking deep convolutional image descriptors. *arXiv preprint arXiv:1412.6537*, 2014.

[258] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[259] V. Sindhwani and P. Niyogi. A co-regularized approach to semi-supervised learning with multiple views. In *International Conference on Machine Learning Workshop*, 2005.

[260] S. Singh, A. Gupta, and A. Efros. Unsupervised discovery of mid-level discriminative patches. In *European Conference on Computer Vision*, 2012.

[261] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *IEEE International Conference on Computer Vision*, 2003.

[262] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski. Finding paths through the world's photos. *ACM Transactions on Graphics (SIGGRAPH)*, 2008.

[263] M. Stark, M. Goesele, and B. Schiele. Back to the future: Learning shape models from 3D CAD data. In *British Machine Vision Conference*, 2010.

[264] C. Stein et al. Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. In *Proceedings of the Third Berkeley symposium on mathematical statistics and probability*, volume 1, 1956.

[265] M. F. Stollenga, J. Masci, F. Gomez, and J. Schmidhuber. Deep networks with internal selective attention through feedback connections. In *Conference on Neural Information Processing Systems*, 2014.

[266] C. Su et al. Multi-task learning with low rank attribute embedding for person re-identification. In *IEEE International Conference on Computer Vision*, 2015.

[267] E. Sudderth, A. Torralba, W. T. Freeman, and A. Wilsky. Learning hierarchical models of scenes, objects, and parts. In *IEEE International Conference on Computer Vision*, 2005.

[268] K.-K. Sung and T. Poggio. Learning and Example Selection for Object and Pattern Detection. In *MIT A.I. Memo No. 1521*, 1994.

[269] J. S. Supancic III and D. Ramanan. Self-paced learning for long-term tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[270] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *Conference on Neural Information Processing Systems*, 2013.

[271] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[272] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.

[273] M. Takáč, A. Bijral, P. Richtárik, and N. Srebro. Mini-batch primal and dual methods for svms. *arXiv preprint arXiv:1303.2314*, 2013.

[274] K. Tang, R. Sukthankar, J. Yagnik, and L. Fei-Fei. Discriminative segment annotation in weakly labeled video. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[275] A. Teichman and S. Thrun. Tracking-based semi-supervised learning. In *RSS*, 2011.

[276] P. Teterwak and L. Torresani. Shared Roots: Regularizing Deep Neural Networks through Multitask Learning. Technical Report TR2014-762, Dartmouth College, Computer Science, 2014.

[277] K. Tieu and P. Viola. Boosting image retrieval. *International Journal of Computer Vision*, 2004.

[278] J. Tighe and S. Lazebnik. Understanding scenes on many levels. In *IEEE International Conference on Computer Vision*, 2011.

[279] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Conference on Neural Information Processing Systems*, 2014.

[280] A. Torralba. Contextual priming for object detection. *International Journal of Computer Vision*, 2003.

[281] A. Torralba and P. Sinha. Statistical context priming for object detection. In *IEEE International Conference on Computer Vision*, 2001.

[282] A. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin. Context-based vision system for place and object recognition. In *IEEE International Conference on Computer Vision*, 2003.

[283] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29, 2007.

[284] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: a large database for non-parametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[285] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[286] Z. Tu and X. Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[287] E. Tulving and D. L. Schacter. Priming and human memory systems. *Science*, 1990.

[288] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 2013.

[289] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders. Segmentation as selective search for object recognition. *International Journal of Computer Vision*, 2011.

[290] S. Vicente, C. Rother, and V. Kolmogorov. Object cosegmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[291] S. Vijayanarasimhan and K. Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[292] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2001.

[293] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *SIGCHI conference on Human factors in computing systems*, 2004.

[294] H. von Helmholtz. *Helmholtz's treatise on physiological optics (1910)*. Optical Society of America, 1925.

[295] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. HOGgles: Visualizing Object Detection Features. *IEEE International Conference on Computer Vision*, 2013.

[296] L. Wang, G. Hua, R. Sukthankar, J. Xue, and N. Zheng. Video object discovery and co-segmentation with extremely weak supervision. In *European Conference on Computer Vision*, 2014.

[297] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *IEEE International Conference on Computer Vision*, 2015.

[298] X. Wang, M. Yang, S. Zhu, and Y. Lin. Regionlets for generic object detection. In *IEEE International Conference on Computer Vision*, 2013.

[299] X. Wang, D. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[300] X. Wang, A. Shrivastava, and A. Gupta. A-Fast-RCNN: Hard Positive Generation via Adversary for Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[301] Y.-X. Wang and M. Hebert. Model recommendation: Generating object detectors from few samples. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[302] Y. Wexler, E. Shechtman, and M. Irani. Space-time completion of video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

[303] O. Whyte, J. Sivic, and A. Zisserman. Get out of my picture! internet-based inpainting. In *British Machine Vision Conference*, 2009.

[304] G. S. Wig, S. T. Grafton, K. E. Demos, and W. M. Kelley. Reductions in neural activity underlie behavioral components of repetition priming. *Nature neuroscience*, 2005.

[305] L. Wolf, T. Hassner, and Y. Taigman. The one-shot similarity kernel. In *IEEE International Conference on Computer Vision*, 2009.

[306] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044, 2010.

[307] D. Wyatte, T. Curran, and R. O'Reilly. The limits of feedforward vision: Recurrent processing promotes robust object recognition when objects are degraded. *Journal of Cognitive Neuroscience*, 2012.

[308] Y. Xiang and S. Savarese. Estimating the aspect layout of object categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[309] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. Sun database: Large scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[310] S. Xie and Z. Tu. Holistically-nested edge detection. *arXiv preprint arXiv:1504.06375*, 2015.

[311] Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8, 2007.

[312] Y. Yang and T. M. Hospedales. A unified perspective on multi-domain and multi-task learning. *arXiv preprint arXiv:1412.7489*, 2014.

[313] Y. Yang and D. Ramanan. Articulated pose estimation with exible mixtures-of-parts. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[314] J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[315] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Conference on Neural Information Processing Systems*, 2014.

[316] J. Yuen, B. C. Russell, C. Liu, and A. Torralba. Labelme video: Building a video database with human annotations. In *IEEE International Conference on Computer Vision*, 2009.

[317] T. P. Zanto, M. T. Rubens, J. Bollinger, and A. Gazzaley. Top-down modulation of visual feature processing: the role of the inferior frontal junction. *Neuroimage*, 2010.

[318] T. P. Zanto, M. T. Rubens, A. Thangavel, and A. Gazzaley. Causal role of the prefrontal cortex in top-down modulation of visual processing and working memory. *Nature neuroscience*, 2011.

[319] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[320] C. Zhang and Z. Zhang. Improving multiview face detection with multi-task deep convolutional neural networks. In *Applications of Computer Vision , 2014 IEEE Winter Conference on*, pages 1036–1041. IEEE, 2014.

[321] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via structured multi-task sparse learning. *International journal of computer vision*, 101 (2):367–383, 2013.

[322] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *European Conference on Computer Vision*, pages 94–108. Springer, 2014.

[323] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *IEEE International Conference on Computer Vision*, 2015.

[324] J. Zhou, J. Chen, and J. Ye. *MALSAR: Multi-tAsk Learning via StructurAl Regularization*. ASU, 2011.

[325] J. Zhou, J. Liu, V. A. Narayan, and J. Ye. Modeling disease progression via fused sparse group lasso. In *SIGKDD*, 2012.

[326] Q. Zhou, G. Wang, K. Jia, and Q. Zhao. Learning to share latent tasks for action recognition. In *IEEE International Conference on Computer Vision*, 2013.

[327] X. Zhu. Semi-supervised learning literature survey. Technical report, CS, UW-Madison, 2005.

[328] Y. Zhu, R. Urtasun, R. Salakhutdinov, and S. Fidler. segdeepm: Exploiting segmentation and context in deep neural networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[329] C. L. Zitnick and P. Dollar. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, 2014.