Visual Knowledge Learning

Xinlei Chen

CMU-LTI-18-001

January, 2018

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
www.lti.cs.cmu.edu

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Language and Information Technologies

Abstract

Understanding images requires rich background knowledge that is not often written down and hard for current computers to acquire. Traditional approach to overcoming this lack of knowledge in computer vision has been to manually summarize them in the form of labels or annotations. While such efforts are impressive, they suffer two critical issues when applied to recognition tasks: *Scalability* and *Usefulness*.

This Ph.D. thesis has made progress toward solving both issues. Instead of manually labeling everything, we develop systems and approaches that can teach computers visual knowledge in a more automatic and scalable way. Specifically, we let them learn by looking at images returned by *web* search engines. We show that even with traditional, imperfect computer vision and natural language technologies, it is nevertheless possible to acquire various types of *explicit* visual knowledge at a large scale, and potentially become better as the system builds up from previous iterations.

Moreover, by adapting end-to-end methods that train deep convolutional networks directly on Internet images, we verify that the intermediate vectorized layers can be convenient and generalizable *implicit* knowledge representations for visual recognition, even with noisy supervision signals. Such representation, while simple, can not only be transformed to discrete relationships as explicit knowledge, but also be exploited to accomplish complex-structured tasks like caption generation.

Finally, we develop reasoning frameworks to use visual knowledge. To this end, we combine both implicit and explicit knowledge into a single pipeline – since the former is effective especially when abundant data is available; and the latter offers supervision, model explainability and alternative ways to help when few examples exist. As one building block, we present a *local*, spatial memory to store instances while preserving the intrinsic layout of the image. To leverage explicit knowledge, we additionally introduce a graph-based module for *global* reasoning. Both are tested to be helpful for enhancing the reasoning ability of current vision systems.

Acknowledgments

First, I owe deep gratitude to my advisor Abhinav Gupta, who motivates me to set high bars, focus on big things and do good research; encourages me in the face of failures and difficulties; helps me in improving writing and speaking skills; and most importantly pushes me back on track with great patience when I have been goofing around and wasting time (it happens). Being able to work closely with him is one of the best choices I have ever made. Without this man, I cannot imagine who and where I will be right now.

Second, I thank my thesis committee members, Martial Hebert, Tom Mitchell, Fei-Fei Li and Andrew Zisserman, for their feedback and flexibility throughout this process. My memory of Martial (also my Ph.D. endeavor) begins with the rigorousness and wit in his computer vision class, which I am always proud to be part of – both as a student and as a teaching assistant. I am genuinely amazed by how Tom can handle complex mathematical problems with simple intuitions and break them down into clean formulations. His life-long enthusiasm for knowledge and research makes himself my role model for never-ending learning! I greatly appreciate Fei-Fei for spending a precious amount of time and effort on my thesis-related research during my internship at CloudML, engaging in discussion of technical details and providing useful ideas even for tiny things like how to make a better figure. I am most influenced by Andrew for his devil-in-detail spirit towards solid research, which will be part of my research style forever.

For some undeserving luck, my Ph.D. journey has crossed incredible minds during my internships. Larry Zitnick, being my first internship mentor, teaches me the invaluable skills toward becoming a independent researcher – how to find insights in experimental results, how to think out-of-the-box and always be ready to defend my own ideas, *etc*.; but most importantly makes me a better person – caring for health and family, working efficiently, learning from what has been overlooked in the past, and always thinking of the greater good of the whole community. I cannot overstate the continued impact of Larry in my career and even in my personal life, and will strive to live-up to his standard. My second internship mentor, Bill Freeman, offers me a remarkable chance to do research in a very different area in computer vision, whose words not only opened up my mind, but also shaped my view of looking at problems. I would also like to thank other members in his VisCam team, in particular Ce Liu – a top Chinese scholar who acts just like a big brother to me, and gives me priceless advice on analyzing results, delivering clear presentations, *etc*. The same also goes to Miki Rubinstein, who is always there for me, helps me in every aspect within and beyond research. I sincerely thank my third internship hosts Li-Jia Li and Shengyang Dai, who endowed me the fabulous opportunity and a favorable research environment to finish the last piece of my thesis at CloudML. Finally, I am extremely honored to have interacted with lots of other CMU faculty members, including but not limited to Alyosha Efros, Kayvon Fatahalian, Eduard Hovy, Kris Kitani, Srinivas Narasimhan, Bhiksha Raj, Deva Ramanan, Roni Rosenfeld, Yaser Sheikh, Leonid Sigal, Michael Tarr and Fernando De la Torre. Your thoughts, intentionally or unintentionally, consciously or unconsciously, enlighten my mind.

While my interest changes over time, the introduction to research during my undergraduate years at Zhejiang University lays the foundation of a curious and critical mind – for which I am in great debt to Deng Cai for his guidance and support. The friendship made at the CAD & CG lab also became part of my treasure in the US, notably with Jialu Liu, Jiajun Lu, Zifei Tong, Chenxia Wu, Dan Xie, Zhou Yu, Jiemi Zhang, Chiyuan Zhang and Chuhang Zou.

I also met amazing new colleagues and friends after coming to CMU. To name some: Elissa Aminoff, Pallavi Baljekar, Aayush Bansal, Leo Boytsov, Yang Cai, Nadine Chang, Xiang Chen, Zhuo Chen, Wen-Sheng Chu, Achal Dave, Jesse Dodge, Carl Doersch, David Fouhey, Dhiraj Gandhi, Matt Gardner, Rohit Girdhar, Yong He, Lu Jiang, Hanbyul Joo, Gunhee Kim, Thomas Kollar, Chen

Kong, Jayant Krishnamurthy, Zhenzhong Lan, Yong-Jae Lee, Jiwei Li, Lei Li, Martin Li, Yin Li, Hector Liu, Wei-Chiu Ma, Aravindh Mahendran, Kenneth Marino, Daniel Maturana, Ishan Misra, Adithya Murali, Keerthiram Murugesan, Prasanna Muthukumar, Ben Newman, Lerrel Pinto, Sentil Purushwalkam, Nick Rhinehart, Alan Ritter, Olga Russakovsky, Abulhair Saparov, Kumar Shaurya Shankar, Yi Shi, Abhinav Shrivastava, Gunnar Sigurdsson, Krishna Singh, Meng Song, Arne Suppe, Yuandong Tian, Minh Vo, Jacob Walker, Di Wang, Naiyan Wang, William Wang, Xiaolong Wang, Yu-Xiong Wang, Fanyi Xiao, Pengtao Xie, Chenyan Xiong, Xuehan Xiong, Diyi Yang, Shoou-I Yu, Zhiding Yu, Hao Zhang, Yimeng Zhang, Ran Zhao, Guoqing Zheng, Feng Zhou, Jiaji Zhou, Tinghui Zhou, and Jun-Yan Zhu. I am especially indebted to: Yang, who helped me settle down in the first year; Di, who practiced driving with me and shared an unforgettable road trip to Niagara; Jacob, who has been my roommates for more than four years and improved my living conditions in all possible ways; Prasanna, who introduced me to all the local to-dos (and not-to-dos) as a Pittsburgh and CMU veteran, most unforgettable the Climbing Wall; David, who helped me most in writing (even this template is his); Abhinav, who lent me super awesome figure-making and vehicle-purchasing suggestions; Xiaolong, with whom I could finally speak some mother tongue in the lab; Gunnar, who taught me Iceland-level skiing and Snapchat-level messaging; Jun-Yan, who played ping-pong with me and made my internship experience more than enjoyable; Lu, who generously offered a shelter during my parents' visit in bay area; Yin, who lent his car to me for surviving the last few days in Pittsburgh, all my co-authors for the deadline stress I may have caused; all my pracitice talk audiance for the time wasted sitting through. My acknowledgments also go to Laura Alford, Mary Jo Bensasi, Sharon Cavlovich, Alison Chiocchi, Alison Day, Carly Devenburgh, Christine Downey, Robert Frederking, Lynnetta Miller, Suzanne Muth, Jess Murphy-True, Mary Stech, and Stacy Young who worked so hard to make the (sometimes seemly impossible) arrangements possible. Half a decade is a long time and I don't own a perfect memory, so I am sure I have forgotten to mention some others, but I thank you all the same.

I thank the legendary atmosphere at Smith Hall 212, which witnessed countless days and sleepless nights of hard work, peer encouragement and funny jokes. I am also grateful that LTI maintained a separate office for me at GHC 5705, where I shared happiness and sorrow from time to time. I can never overstate how fantastic the computing resources are at CMU, and would like to especially thank warp, workhorse and yoda for working even harder than we graduate students are. More on this delicate thank-you list are the double-cooked pork from Chengdu Gourmet, the shrimp pad Thai from Little Asia, the Bulgogi lunch-box from Oishii Bento, and the spare ribs from Rose Tea Cafe (the Squirrel Hill one), all of which usually get picked up late in a silver Mitsubishi Lancer (to which I am deeply indebted for all the tortures it suffered as my first car). There are certainly things to devour and savor in Pittsburgh.

Particularly, a miracle took place in the second year and led me to a special girl – Rujia, at a special time in a special place, who is still with me now. It's definitely not easy to maintain love and trust, sacrifice and devotion through thick and thin, together and apart. Our tale, beginning in PA and traveling through WA to the next stop CA (thanks American Airlines), shall go further on to open newer A++ chapters.

Finally, I would like to dedicate this thesis to my family. Pursuing one's degree and career some 7000 miles away from the hometown is a big challenge despite modern technologies for connection and communication. Sorry mom, dad and granny for missing so many important occasions, and thank you so much for the years of understanding, belief and sometimes lie to make it work.

Contents

List of Figures

List of Tables

Chapter 1

Overview

Over the past few decades, there has been tremendous progress within the field of artificial intelligence (AI): a wide variety of difficult problems, ranging from chess to spam filtering to credit card fraud detection are solved everyday by computers. However, while computers can easily outperform the best humans in chess, there is still a long way to go on the task of images understanding. One major reason for this gap is that the rules of chess can be summarized succinctly and encoded, but understanding pictures or words depends on rich sets of background *knowledge* that is not often written down. Humans acquire this knowledge by observing and interacting with people and the physical world over a long period of time, slowly learning that `runways do not move`, `planes can fly`, `spinner luggages have four wheels`, *etc*. These sorts of interactions are however impossible for a computer, preventing the learning of such world knowledge in an AI system. Most computer systems for perception or natural language understanding have greatly suffered due to a lack of knowledge.

The conventional approach to overcoming this lack of knowledge has been to mimic the way we have taught computers to play chess: We manually summarize them in the form of structured data bases. For example, Cyc [150] attempts to assemble a comprehensive ontology and knowledge base, with the goal of enabling AI with the power of human-like reasoning. Over the past decades, the project has collected hundreds thousands of concepts and millions of facts connecting them. While such efforts are impressive, they suffer two critical issues when applied to practical tasks like visual recognition:

1. **Scalability.** One of the largest image dataset, ImageNet [233], has gathered $\sim$1 million annotations of objects and their bounding boxes within images by crowd-sourcing over a five-year period. While it seems enormous, it is minuscule compared to the largest text knowledge base [63] with $>$1.6 billion facts, and more importantly the billions of images uploaded everyday! Furthermore, if manual annotation of straight-forward visual knowledge like whether `an image contains a window` cannot grow as fast as the size of the web visual data, then how do we expect to scale up the annotation of tricky knowledge like `windows can be opened, but not the ones one airplanes, except WW-I-era planes, among others`?

2. **Usefulness.** While text knowledge base can be naturally used for high-level problems like open-ended question answering [78] in language, it is more tricky to directly incorporate facts like `Southwest Airlines exclusively operate on Boeing-737` to help image understanding. Fundamentally, there is a dispartity bewteen such sparse, clean knowledge

items and our dense, noisy visual world. As far as this difference is concerned, at least a traditional structured knowledge base by itself could not stand alone to solve vision.

This Ph.D. thesis has made progresses toward solving both issues. **First**, instead of manually labeling everything, we developed systems and approaches that can let computers learn visual knowledge in a more automatic and scalable way. **Second**, once visual knowledge is acquired, we investigated on making such knowledge useful. In particular, we look at high-level tasks (*e.g.* object detection) to see if current recognition systems can benefit from the additional reasoning ability provided by background knowledge.

1.1 What Do We Mean by "Visual Knowledge"

Before proceeding, we would first like to more precisely state what we mean by "*visual knowledge*". Quoting Wikipedia's entry for knowledge[1]:

> Knowledge is a familiarity, awareness or understanding of someone or something, such as facts, information, descriptions, or skills, . . . , It can be implicit (as with practical skill or expertise) or explicit (as with the theoretical understanding of a subject); it can be more or less formal or systematic.
>
> —WIKIPEDIA

Derived from the above description, we broadly refer "visual knowledge" as helpful understanding of any form of visual data. Such understanding can be *explicit*, for instance knowing that `an image contains a window`, `a person is beside the window`, and `the window is on a Boeing-787`. While explicit knowledge can broadly cover any expressions in a language or any formula in a science, in this thesis we focus on the specific explicit knowledge that can be expressed by a regular form, *e.g.* subject-verb-object (SVO) triplet, or labeled instances of a category.

On the other hand, understanding can be *implicit*, or not so easily expressible, but vital to the acquisition of skills or the accomplishment of end-tasks. In a way, such knowledge is more aligned with "commonsense", explained as[2]:

> Commonsense is a basic ability to perceive, understand, and judge things that are shared by ("common to") nearly all people and can reasonably be expected of nearly all people without need for debate.
>
> —WIKIPEDIA

Implicit knowledge underlies one's basic ability to perceive and fundamental need to survive. Since it is inherently shared among nearly all people, there is less need to communicate it and, in fact, it becomes the common-ground for connecting other human beings. Therefore, we also refer to this implicit form of knowledge as "visual commonsense" in this thesis.

1.2 Learning Visual Knowledge

With the above clarifications, our first mission is to scale up visual knowledge acquisition by learning. We begin with explicit understanding.

More specifically, rather than listing and labeling everything, we let computers automatically learn by looking at images on the *Internet*, which acts as a proxy for the real world. We show that

[1] https://en.wikipedia.org/wiki/Knowledge

[2] https://en.wikipedia.org/wiki/Common_sense

even with traditional, imperfect vision and natural language technologies, the system is nevertheless capable of acquiring various types of explicit visual knowledge at a large scale [42, 43, 44], and potentially become better as the system builds up from earlier iterations [43]. This part (I) consists of three chapters.

Chapter 2 describes our system Never Ending Image Learner (NEIL), with the goal of automatically extracting explicit visual knowledge from Internet-scale data. NEIL uses a semi-supervised learning algorithm that jointly discovers commonsense relationships (*e.g.*, `Corolla is a kind of car`, or `wheel is a part of car`) and labels instances of the given visual categories. As the initialization step for a given object category, the program retrieves noisy images from the web search engines, and then automatically cleans them up by localizing instances within the set of images. More importantly, We designed co-occurrence based approaches to discover triplet relationships between scenes, objects and attributes, and empirically demonstrated that relationships can regularize the learning process and help us train better visual models than standard bootstrapping.

NEIL was first proposed to focus on bounding box representations for object categories. In **Chapter** 3, we take a step further by segmenting out more detail foreground. We extend the top-down formulation of building detectors to also leveraging the bottom-up cues from images. Specifically, we augment the detectors with foreground segmentation priors learned from aligned visual clusters, or subcategories. The strong priors are then combined with discriminatively trained detectors and image-specific cues to produce clean object segments for each object instance. In this way, NEIL's knowledge base is enriched with segmentations that can potentially benefit tasks that require pixel-level annotations.

Even with the augmentation of segments, NEIL still has problem with polysemous categories [42], a classical example is `Apple`, which can mean both the COMPANY and the FRUIT. On the other hand, NEIL does cluster visually similar image patches into different subcategories. To build the missing link, in **Chapter** 4, we develop algorithms that specifically target at this problem. One interesting fact is that, the visual subcategories in NEIL form a many-to-one mapping to the semantic senses of that category [3]. For example, `Columbia` can be associated to many senses such as UNIVERSITY, RIVER, SPORTSWEAR, or STUDIO. And each sense can be represented by numerous visual subcategories, or visual senses. For UNIVERSITY it can be profile pictures of the professors, buildings and sculptures on campus, *etc*. We show that by jointly discovering senses in both text and image domain with a structure-EM style optimization algorithm, we can enforce such a hierarchical mapping and clean up the NEIL knowledge base. This is because the approach not only receives visual signals from images, but also textual cues from surrounding snippets of the web page.

1.3 Learning Visual Commonsense

Next, we aim to learn implicit visual knowledge, or visual commonsense.

Different from explicit knowledge where we have web as a convenient source to learn from, implicit knowledge seems harder to obtain as it is not written down anywhere. Fortunately, we may *not* need to at all: In the end, our goal is to let machines acquire the skill, *e.g.* detecting an object, or generating a caption. In this sense, knowledge acquisition and representation are only intermediate steps toward this ultimate goal. And by optimizing the goal, we can force the model parameters or weights to learn some necessary implicit knowledge along the way [104].

With this insight and the resurgence of convolutional networks (ConvNets) [136], we started to explore end-to-end approaches that directly attempt to solve certain vision problems in Part II. A nice property of deep ConvNets is their ever-increasing depth, which naturally creates *layered*

[3]Or more strictly speaking noun phrase (NP), as the same string means different things in different contexts.

intermediate representations from raw pixels to end-signals. Therefore, it is handy to intercept the information flow by taking out a designated layer, and regard its vectorized representations as the result of implicit background knowledge enforced on new images. Although such form of knowledge may not be directly explainable or expressible, it is undoubtedly useful as it is directly optimized toward the end-task, at least to the task itself.

And not just itself, ConvNet then started showcasing the astonishing power of pre-training – models pretrained on ImageNet classification challenge significantly advanced the performances on relevant tasks like object detection [87]. From the perspective of visual knowledge learning, this strongly indicates that by learning to classify images, certain generally useful "skills" have been acquired inside the weights of the network. This is exciting, since it means we finally have an algorithm that can condense huge datasets to neat representations, and at the same time keep it *useful* – a term NEIL need to justify as it had so far only worked on small datasets constructed from web images [43, 44] rather than standard testbeds like PASCAL VOC [67].

As a natural step toward marrying NEIL with ConvNets, we started off with training ConvNets directly from the web in **Chapter** 5. We note that ConvNets also need to embrace the web, as it is "data-hungry" – with more training data, it tends to perform better. Despite the impressive results ImageNet can deliver, the dataset is still minuscule compared to the billions of images indexed by search engines. However, web images are likely noisy and come in different types [38]. Accommodating this observation, we designed a stage-wise training and fine-tuning process inspired from curriculum learning [17], and show that it is indeed possible to do converge the network despite the noise, and web-based pre-training can provide very competitive intermediate representations for related tasks. Notably, we find the confusion matrix of a trained network is already a reasonable encoder of pair-wise similarities between categories, and can be easily transformed as a discrete, explicit relationship graph to remedy semantic drifting when fed with noisier, more realistic images.

As another application of the learned implicit knowledge inside the networks, in **Chapter** 6 we explore another task – caption generation [41]. The key motivation is to show that the vectorized representation from networks can indeed encode visual commonsense and perform well on *structured* tasks like generating an entire sentence from beginning to end. Critical to our approach is a recurrent neural network (RNN) that attempts to dynamically build a visual representation of the scene as a caption is being generated or read. The approach, together with many other concurrent attempts, outperformed traditional methods by a big margin and demonstrated the power of end-to-end learning in capturing the temporal dynamics of words in natural languages.

1.4 Reasoning with Visual Knowledge

Finally, we attempt to use visual knowledge, both in the explicit and the implicit form for reasoning.

Why do we need both? On one hand, implicit representations with end-to-end training have significantly advanced our horizon toward solving vision [35, 226, 229]. Few can deny the effectiveness especially when paired with large-scale dataset for training. The dense, vectorized representation makes it harder to interpret, but easier to use and transfer.

On the other hand, explicit knowledge expresses itself in a straight-forward way, potentially giving a clear explanation of how the model works. Moreover, it is not always the case that annotations are abundant. In the case of one-shot [73] or zero-shot [205] learning, the model has few other choice but background knowledge – sometimes in the explicit form – in order to generalize well. Finally, implicit visual knowledge can hardly be as effective without the help of the simplest form of explicit knowledge – *labels*, as unsupervised pipelines of training networks are still catching up as of now.

Therefore, we believe the best bet lies in the combination of the two: Fusing and utilizing both types of knowledge effectively and efficiently. Therefore, in Part III we present a unified framework as our approach toward joint reasoning with visual knowledge bases.

In **Chapter** 7 we made our first building block – spatial memory [40]. Visual knowledge is mostly concerned with relationships between scenes, objects, attributes and parts. Spatial memory offers an intuitive way to store instance-level knowledge in a learnable representation while preserves the intrinsic layout of the two-dimensional image. The spatial structure naturally allows us to conduct *context* reasoning with ConvNets: For example, knowing the location and orientation of `baseball bat` can help find `baseball`. We chose object detection as the task of interest, and devised a pipeline for learning both deduplication (a task currently fulfilled by non-maximal suppression, NMS [87]) and the interplay between objects, semantically and spatially.

However, a spatial memory is not yet enough, as it lacks the essential module to deal with explicit visual knowledge in forms of SVO triplets or language snippets, and more importantly the reasoning only takes place *locally* in the limited square-region of an image, whereas humans can easily connect the phenomenon to his or her *global* background knowledge. To this end, in **Chapter** 8 we introduce a graph-based module to complete our framework. The graph is built with nodes that stand for either region or category, with edges denoting spatial, semantic, or assignment relationships between them. The module performs reasoning by passing messages directly along the graph, and is shown to be useful and resilient for recognition tasks.

1.5 Conclusion & Discussion

For future reference, in **Chapter** 9 we give summarizations to the observations, speculations and potential next steps for works in this dissertation.

Part I

Learning Explicit Visual Knowledge

Chapter 2

Never Ending Image Learner

2.1 Introduction

Many successes in computer vision can be 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 explicit, structured visual knowledge on a vast scale? Efforts such as ImageNet [233] and Visipedia [215] have tried to harness human intelligence for this task. However, we believe that such approaches lack both the richness and the scalability required for gathering massive amounts of visual knowledge. For example, at the time of April 2013, only 7% of the data in ImageNet had bounding boxes and the relationships were still extracted via Wordnet.

In this chapter, we consider an alternative approach of automatically extracting explicit visual knowledge from Internet scale data. The feasibility of extracting knowledge automatically from images and videos will itself depend on the technologies in computer vision. While we have witnessed significant progress on the task of detection and recognition, it is believed that we still have a long way to go for automatically extracting the semantic content of a given image, especially with transitional approaches that rely on manually designed features. So, is it really possible to use such approaches for gathering visual knowledge directly from web data?

2.1.1 NEIL – Never Ending Image Learner

We propose NEIL, a computer program that runs 24 hours per day, 7 days per week to: (a) Semantically understand images on the web; (b) Use this semantic understanding to augment its knowledge base with new labeled instances and relationships; (c) Use this dataset and these relationships to build better classifiers and detectors which in turn help improve semantic understanding. NEIL is a constrained semi-supervised learning (SSL) system that exploits the big scale of visual data to automatically extract relationships and then uses these relationships to label visual instances of existing categories. It is an attempt to develop the world's largest visual structured knowledge base with minimum human effort – one that reflects the factual content of the images on the Internet, and that would be useful to many computer vision and AI efforts. 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 subcategories for object categories; and (e) 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 2.1).

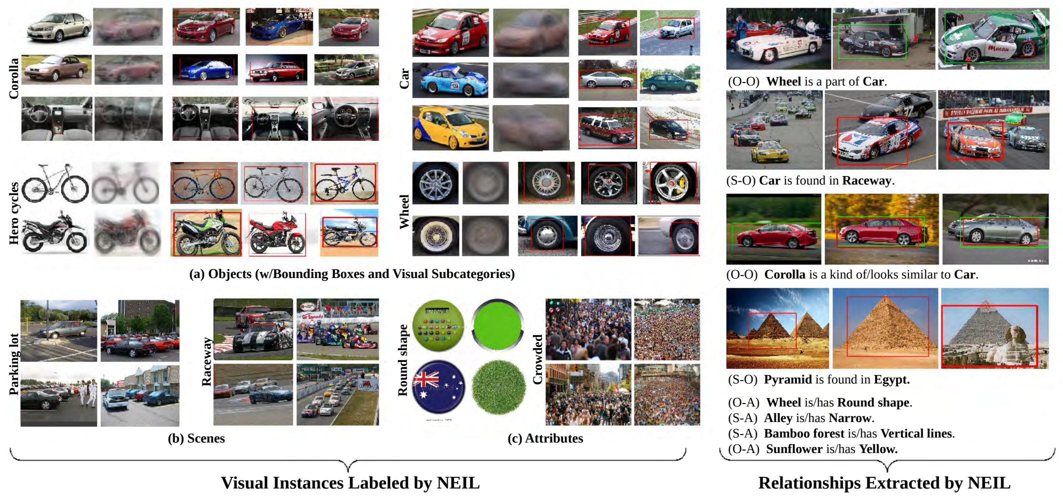

Corolla
Hero cycles
Car
Wheel
(a) Objects (w/Bounding Boxes and Visual Subcategories)
Parking lot
Raceway
Round shape
Crowded
(b) Scenes
(c) Attributes
Visual Instances Labeled by NEIL
(O-O) Wheel is a part of Car.
(S-O) Car is found in Raceway.
(O-O) Corolla is a kind of/looks similar to Car.
(S-O) Pyramid is found in Egypt.
(O-A) Wheel is/has Round shape.
(S-A) Alley is/has Narrow.
(S-A) Bamboo forest is/has Vertical lines.
(O-A) Sunflower is/has Yellow.
Relationships Extracted by NEIL

Figure 2.1: 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 relationships between categories.

We believe our approach is possible for three key reasons:

(a) Macro-vision *vs*. Micro-vision: We use the term “micro-vision” to refer to the traditional paradigm where the input is an image and the output is some information extracted from that image. In contrast, we define “macro-vision” as a paradigm where the input is a large collection of images and the desired output is extracting significant or interesting patterns in visual data (*e.g.*, car is detected frequently in raceways). These patterns help us to extract relationships. Note, the key difference is that macro-vision does not require us to understand every single image in the corpora and extract all possible patterns. Instead, it relies on understanding a few images and statistically combine evidence from these to build our visual knowledge.

(b) Structure of the Visual World: Our approach exploits the structure of the visual world and builds constraints for detection and classification. These global constraints are represented in terms of relationships between categories. Most prior work uses manually defined relationships or learns relationships in a supervised setting. Our key insight is that at a large scale one can simultaneously label the visual instances and extract relationships in a joint semi-supervised learning framework.

(c) Semantically-Driven Knowledge Acquisition: We use an explicit semantic representation for visual knowledge; that is, we group visual data based on semantic categories and develop relationships between them. This allows us to leverage text-based indexing tools such as Google Image Search to initialize our visual knowledge base learning.

Contributions: The chapter’s main contributions are: (a) We propose a never ending learning algorithm for gathering visual knowledge from the Internet via macro-vision. As of October 2013, NEIL has continuously run for 2.5 months on a 200 core cluster; (b) NEIL automatically builds a large visual knowledge base which not only consists of labeled instances of scenes, objects, and attributes but also the relationships between them. While NEIL’s core SSL algorithm works with a fixed vocabulary, we also use noun phrases from NELL’s ontology [32] to grow our vocabulary. As of October 2013, NEIL’s growing knowledge base has an ontology of 1152 object categories, 1034 scene categories, and 87 attributes. NEIL has discovered more than 1700 relationships and labeled more than $400K$ visual instances of these categories. (c) We demonstrate how joint discovery of relationships and labeling of instances at a gigantic scale can provide constraints for improving semi-supervised learning.

2.2 Related Work

Most related work has focused on extracting knowledge in the form of large datasets for recognition and classification [156, 215, 233]. One of the most commonly used approaches to build datasets is using manual annotations by motivated teams of people [215] or the power of crowds [233, 286]. To minimize human effort, recent works have also focused on active learning [249, 283] 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 [156, 240, 246]. A common way of automatically creating a dataset is to use image search results and re-rank them via visual classifiers [75] or some form of joint-clustering in text and visual space [21, 240]. Another approach is to use a semi-supervised framework [314]. 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 [246] or the top retrievals of a text-based search [156]. 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 [48]. 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 [64, 77], semantic similarity [93] or multiple feature spaces [26]. 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 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 [71, 177, 214, 219, 263]. Some of these relationships include: Scene-Object [263], Object-Object [177, 219], Object-Attribute [71, 144, 211], Scene-Attribute [214]. All these relationships can provide a rich set of constraints which can help us improve SSL [31]. For example, scene-attribute relationships such as *amphitheaters are circular* can help improve semi-supervised learning of scene classifiers [246] and Wordnet hierarchical relationships can help in propagating segmentations [137]. But the big question is: how do we obtain these relationships? One way to obtain such relationships is via text analysis [32]. However, as [286] points out that the visual knowledge we need to obtain is so obvious that few would take the time to write it down and put it on the 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 [32], 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 [222, 242] 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.

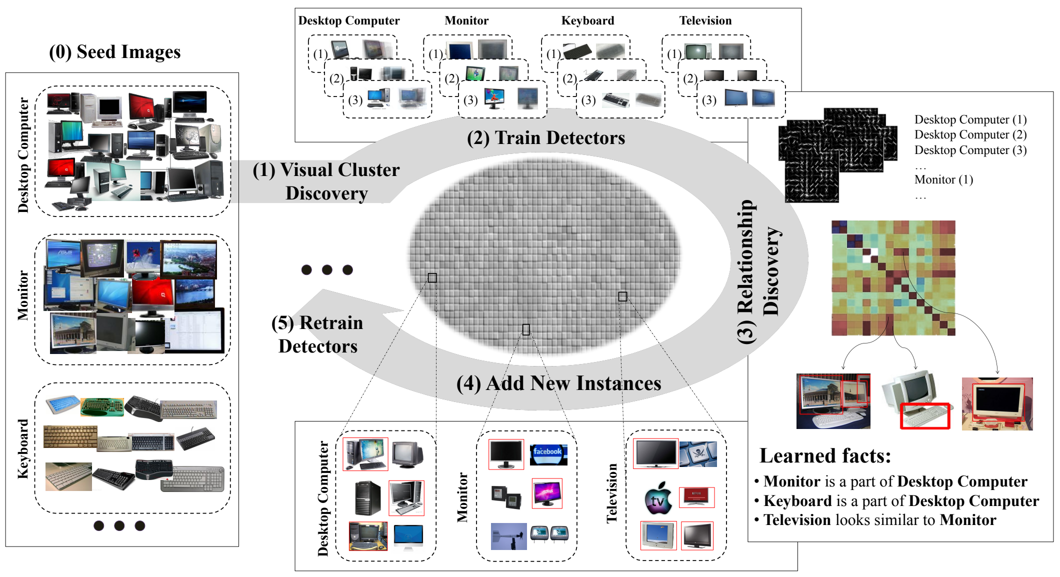
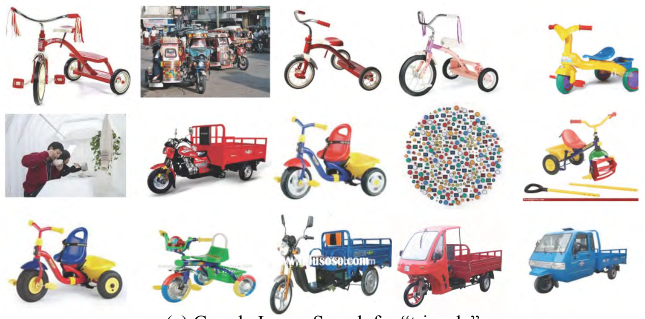


Figure 2.2: Outline of the iterative approach in NEIL.

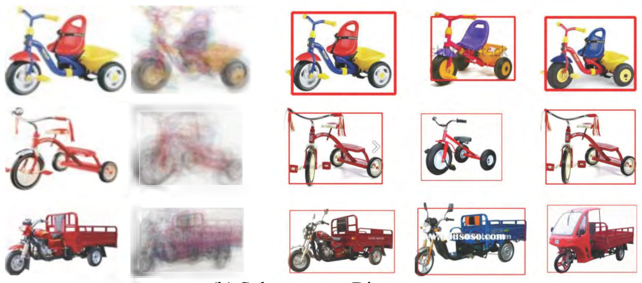
2.3 Approach

Our goal is to extract visual knowledge from the pool of visual data on the web. While visual knowledge is defined as any information that can be useful for improving vision tasks such as image understanding and object/scene recognition, this chapter only focuses on *explicit* knowledge that can be stored in a traditional structured knowledge base. One form of such visual knowledge would be labeled examples of different categories or labeled segments/boundaries. Labeled examples help us learn classifiers or detectors and improve image understanding. Another example of explicit visual knowledge would be relationships. For example, spatial relationships can be used to improve object recognition. NEIL's 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 the chapter 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 white`); (3) Scene-Object (*e.g.*, `car is found in raceway`); (4) Scene-Attribute (*e.g.*, `alley is narrow`).

The outline of our approach is shown in Figure 2.2. We use Google 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 category (*e.g.*, `car`), we first discover the latent visual subcategories and bounding boxes for these subcategories using an exemplar-based clustering approach (Section 2.3.1). We then train multiple detectors for a category (one for each subcategory) 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 2.3.2). 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 2.3.3). The most confidently labeled



(a) Google Image Search for "tricycle"



(b) Sub-category Discovery

Figure 2.3: An example of how clustering handles polysemy, intra-class variation and outlier removal (a). The bottom row shows our discovered clusters.

images are added to the pool of labeled data and used to retrain the models, and the process repeats itself. At every iteration, we aim to learn better classifiers and detectors, which in turn help us learn more relationships and further constrain the semi-supervised learning problem. We now describe each step in detail below.

2.3.1 Seeding Classifiers via Google Image Search

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 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 2.3(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 viewpoints, illumination conditions, *etc*.; (4) Localization: In many cases the retrieved image might be a scene without a bounding box and hence one needs to localize the category before training a detector.

Most of the current approaches handle these problems via clustering. Clustering helps in handling visual diversity [58] and discovering multiple senses of retrieval (polysemy) [173]. It can also help us to reject outliers based on distances from cluster centers. One simple way to cluster would be to use k-means on the set of all possible bounding boxes and then use the representative clusters as visual subcategories. However, clustering using k-means has two issues: (1) High-dimensionality:

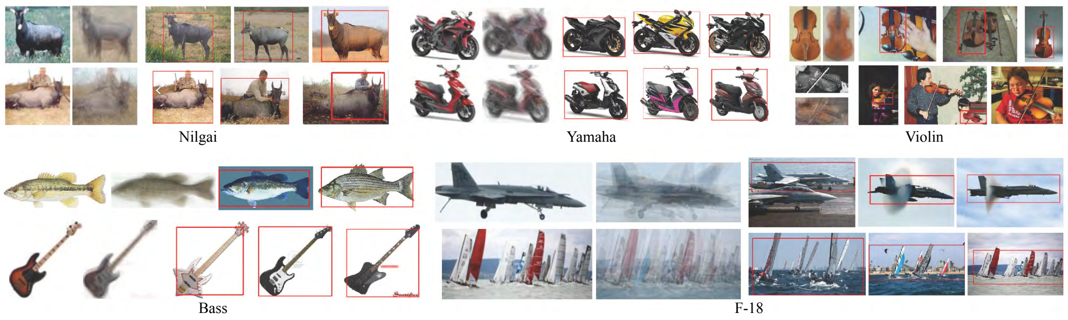


Figure 2.4: Qualitative examples of bounding box Labeling done by NEIL.

We use the Color HOG (CHOG) [126] as the feature representation and standard distance metrics do not work well in such high-dimensions [61]; (2) Scalability: Most clustering approaches tend to partition the complete feature space. In our case, since we do not have bounding boxes provided, every image creates millions of data points and the majority of the data points are outliers. Recent work has suggested that k-means is not scalable and has bad performance in this scenario since it assigns membership to every data point [61].

Instead, we propose to use a two-step approach for clustering. In the first step, we mine the set of downloaded images from Google Image Search to create candidate object windows. Specifically, every image is used to train a detector using exemplar-LDA [97]. These detectors are then used for dense detections on the same set of downloaded images. We select the top K windows which have high scores from multiple detectors. Note that this step helps us prune out outliers as the candidate windows are selected via representativeness (how many detectors fire on them). For example, in Figure 2.3, none of the tricycle detectors fire on the outliers such as circular dots and people eating, and hence these images are rejected at this candidate widow step. Once we have candidate windows, we cluster them in the next step. However, instead of using the high-dimensional CHOG representation for clustering, we use the detection signature of each window (represented as a vector of seed detector ELDA scores on the window) to create a $K \times K$ affinity matrix. The (i, j) entry in the affinity matrix is the dot product of this vector for windows i and j. Intuitively, this step connects candidate windows if the same set of detectors fire on both windows. Once we have the affinity matrix, we cluster the candidate windows using the standard affinity propagation algorithm [80]. Affinity propagation also allows us to extract a representative window (prototype) for each cluster which acts as an iconic image for the object [220] (Figure 2.3). After clustering, we train a detector for each cluster/subcategory using three-quarters of the images in the cluster. The remaining quarter is used as a validation set for calibration.

2.3.2 Extracting 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 Internet but instead understand 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. For example, the nose-face relationship is characterized by low relative window size (nose is less than 20% of face area) and the relative location that nose occurs in center of the face. This is used to define a compatibility function $\psi_{i,j}(\cdot)$ which evaluates if the detections from category i and j are compatible or not. We also classify the relationships into the two semantic categories (part-of, taxonomy/similar) using relative features to have a human-communicable view of visual knowledge base.

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 is 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.

2.3.3 Retraining via Labeling New Instances

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 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, known as bootstrapping, leads to semantic drift. To avoid semantic drift, we use the rich set of relationships we extracted in the previous section and ensure that the new labeled instances of `car` satisfy the extracted relationships (*e.g.*, `has wheels`, `is 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

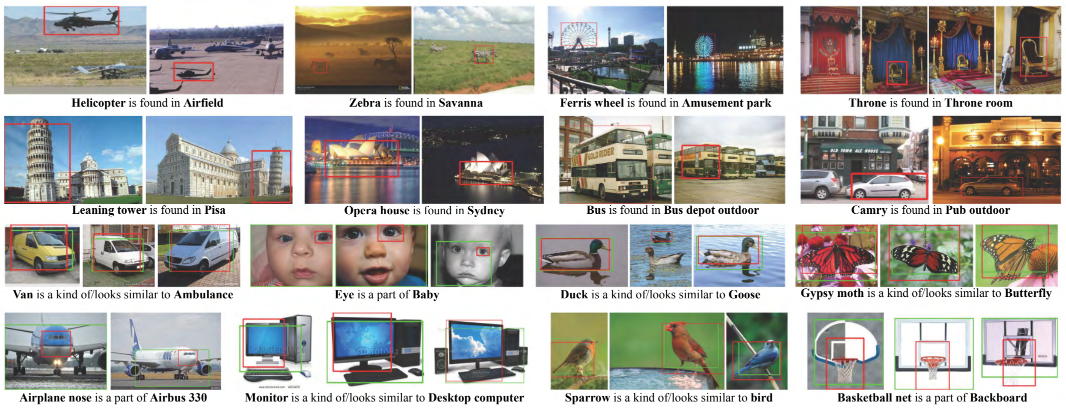

Helicopter is found in **Airfield**
Zebra is found in **Savanna**
Ferris wheel is found in **Amusement park**
Throne is found in **Throne room**
Leaning tower is found in **Pisa**
Opera house is found in **Sydney**
Bus is found in **Bus depot outdoor**
Camry is found in **Pub outdoor**
Van is a kind of/looks similar to **Ambulance**
Eye is a part of **Baby**
Duck is a kind of/looks similar to **Goose**
Gypsy moth is a kind of/looks similar to **Butterfly**
Airplane nose is a part of **Airbus 330**
Monitor is a kind of/looks similar to **Desktop computer**
Sparrow is a kind of/looks similar to **bird**
Basketball net is a part of **Backboard**

Figure 2.5: Qualitative examples of scene-object (rows 1-2) and object-object (rows 3-4) relationships extracted by NEIL.

categories i, j, then we can find the new instances of object category i using the contextual scoring function given below:

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

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 catalog. For example, if `wheel is a part of car` exists in the catalog 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 given below:

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

where $\mathcal{R}_{A'}$ represents the catalog 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.

2.3.4 Implementation Details

To train scene & attribute classifiers, we first extract a 3912 dimensional feature vector from each image. The feature vector includes $512D$ GIST [199] features, concatenated with bag-of-words

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 2.6: Examples of extracted relationships in NEIL.

representations for SIFT [169], HOG [49], Lab color space, and Texton [184]. 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 object and attribute section, we use CHOG [126] features with a bin size of 8. We train the detectors using latent SVM model (without parts) [74].

2.4 Experimental Results

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

2.4.1 NEIL Statistics

While NEIL's core algorithm uses a fixed vocabulary, we use noun phrases from NELL [32] to grow NEIL's vocabulary. As of 10th October 2013, NEIL has an ontology of 1152 object categories, 1034 scene categories and 87 attributes. It has downloaded more than 2 million images for extracting the current structured visual knowledge. For bootstrapping our system, we use a few seed images from ImageNet [233], SUN [298] or (it not in the ontology of those datasets) the top-images from Google Image Search. For the purposes of extensive experimental evaluation in this chapter, we ran NEIL on steroids (200 cores as opposed to 30 cores used generally) for 2.5 months, during which time NEIL has completed 16 iterations and it has labeled more than $400K$ visual instances (including $300,000$ objects with their bounding boxes). It has also extracted 1703 relationships.

2.4.2 Qualitative Results

We first show some qualitative results in terms of extracted visual knowledge by NEIL. Figure 2.4 shows the extracted visual subcategories along with a few labeled instances belonging to each subcategory. 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 categories indicate that contextual relationships help make our system robust to semantic drift and ensure diversity.

Table 2.1: 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**

Figure 2.5 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 2.6 shows some of the interesting scene-attribute and object-attribute relationships extracted by NEIL.

2.4.3 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. At the end of iteration 6, 79% of the relationships extracted by NEIL are correct, and 98% of the visual 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. In the 16 iterations we have observed little sign of semantic drift. 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 (IoU) 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 measure [2] produces bounding boxes with 0.59 overlap on average.

2.4.4 Using Knowledge for Vision Tasks

Finally, we want to demonstrate the usefulness of the visual knowledge learned by NEIL on vision tasks such as object detection and scene classification. Here, we 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 2.1 shows the results. We use mean average precision (mAP) as

Table 2.2: 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**

the evaluation metric. As the results show, automatic relationship extraction helps us to constrain the learning problem, and so the learned classifiers give much better performance. Finally, if we also use the contextual information from NEIL relationships we get a significant boost in 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 2.2 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, detectors trained using NEIL work better than standard bootstrapping.

Chapter 3

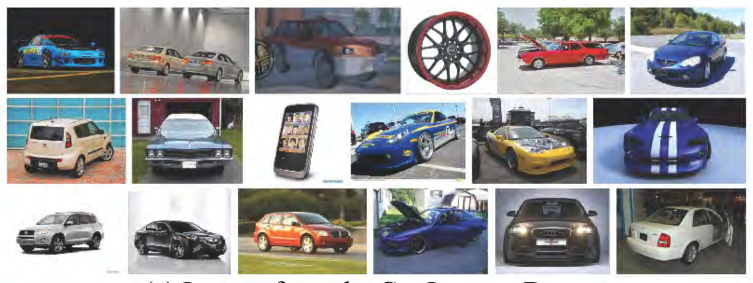
From Bounding Boxes to Segmentations

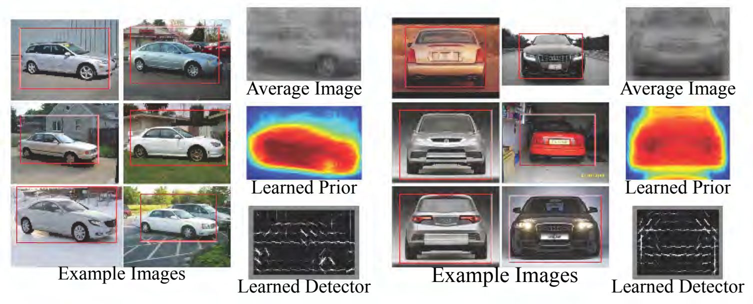
3.1 Introduction

In this chapter, we focus on generating a large segmentation knowledge base which we believe is also the next step in enriching visual knowledge bases such as NEIL [43]. Given a large collection of noisy Internet images of some object category (*e.g.*`car`), our goal is to automatically discover the object instances and their segmentations. There has been some related work on joint segmentation of multiple images, but most of those approaches focus on using generative models for extracting recurring patterns in images. On the other hand, much of the advancement in the field of object detection has come from learning discriminative models using large quantities of visual data [74]. In this chapter, we propose a conceptually simple yet powerful approach that combines the power of generative modeling for segmentation, and the effectiveness of discriminative models for detection in order to segment objects out of noisy web images.

The central idea behind our approach is to learn the top-down priors and use these priors to perform joint segmentation. But how do we develop top-down priors? Approaches such as class-cut [3] and collect-cut [146] develop top-down priors based on semantic categories, *i.e.*, they build appearance models for semantic categories such as `cars`, `airplanes`, *etc.*, and use them in a graph-based optimization formulation.

But are these semantic categories the right way to develop top-down priors? Over the years, we have learned that the high intra-class variations within a semantic category leads to weak priors and these priors fail to significantly improve performance. On the other hand, clustering the data into visual subcategories [43, 58] followed by learning priors on these visual subcategories has shown a lot of promise. In this chapter, we build upon these ideas and use visual subcategories to construct top-down segmentation priors to improve joint segmentation of multiple images. We use the advances in learning exemplar based detectors [97, 178] to discover visual subcategories and "align" the instances in these visual subcategories; these visual subcategories are then exploited to build strong top-down priors which are combined with image evidence based likelihoods to perform segmentation on multiple images. Figure 3.1 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 a then state-of-the-art performance in joint segmentation of an object category on standard datasets [232]. More importantly, we integrated this


(a) Images from the Car Internet Dataset



(b) Discovered Visual Subcategories and Learned Priors/Models

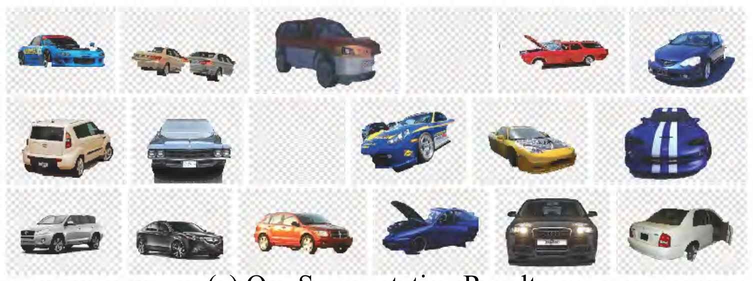

(c) Our Segmentation Results

Figure 3.1: 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).

algorithm in NEIL and it generated approximately $500K$ segmentations using web data as of April 2014.

The code of this pipeline is released[1].

3.2 Related Work

Segmentation is a fundamental problem in computer vision. Early works focused on generating low-level or bottom-up groupings that follow Gestalt laws – the classic pipeline [175] was to use low-level features (such as color, texture, *etc.*) as input to segmentation or clustering methods [132, 243].

[1] https://github.com/endernewton/subdiscover

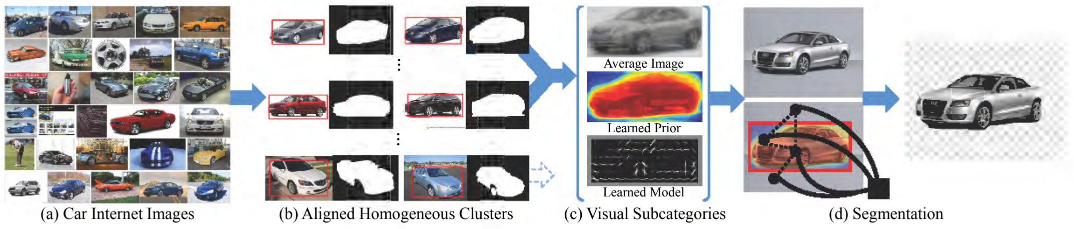


Figure 3.2: Overview of the approach.

However, for real-world images they fail to produce consistent object segmentation. One of the main reasons for the failure of pure bottom-up segmentation is that an object is a complex concept. Generally object segmentation requires combining multiple visually consistent clusters, which turns out to be too difficult for the vanilla bottom-up segmentation algorithms.

One way to incorporate top-down information is to learn priors in terms of semantic object categories in a fully supervised manner [140, 174, 203]. To reduce the burden of this annotation, semi- and weakly-supervised approaches have been developed. For example, class-cut [3] uses image-level object annotation to learn priors. Another popular way to reduce annotation is to use interactive supervision in terms of a few simple scribbles [13, 27, 160, 231]. Finally, approaches have tried to use other kind of priors including bounding boxes [149], context [146, 147], saliency [68] and object probability [2, 141]. However, most of these priors are still learned on semantic object categories which often leads to weak priors due to intra-class and pose variations.

In order to learn priors with little or no annotation, recent approaches have also tried to use object discovery to extract segments from images automatically, followed by learning of priors (see [278] for an overview). A common approach [234, 255] is to treat the unlabeled images as documents and objects as topics, and use generative topic-model approaches such as latent Dirichlet allocation and hierarchical Pitman-Yor to learn the distribution and segmentation of multiple categories of objects simultaneously. However, completely unsupervised object discovery and learning of segmentation prior often tend to be non-robust for the problem itself is under-constrained.

We follow the regime of using web-based supervision to learn segmentation priors [232]. We use query terms to obtain noisy image sets from Internet and then learn models and segmentation priors from these images. However, instead of modeling segmentation priors and constraints based on semantic categories, we model them based on visual subcategories [43], which are visually homogeneous clusters and have much less intra-class variations.

Our work is also related to co-segmentation, where the task is to simultaneously segment visually similar objects from multiple images at the same time [13, 120, 127, 230, 232, 281]. 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 closest work to our approach is the paper by Rubinstein *et al*. [232], 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 since we never attempt to solve a global joint segmentation problem, but instead only perform segmentation on subsets of the data.

3.3 Approach

Our goal is to extract objects and their segments from large, noisy image collections in an unsupervised manner. We assume that the collection is obtained as a query result from search engines, photo albums, *etc*. Therefore, a majority of these images contain the object of interest. However, we still want to reject the images which are noisy and do not have the object instances. While one can use approaches like graph-cut with center prior to discover the object segments, such an approach is inferior due to the weak center prior in the 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) [43, 58]. 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?

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). These overlapping clusters are built using the recent work in training instance based detectors and then using these detectors to find similar instances in the training data [61, 97, 178, 253]. Because we run these detectors in a sliding window manner, 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. The introduction of this extra constraint in conjunction with high-quality clusters leads to clean segmentation labels for the images.

Our clusters are tight – low recall and high precision – with very few instances, and hence some of the clusters are noisy, which capture the repetition in the noisy images. For example, five motorbikes in the car collection can group together to form a cluster. To clean-up the noisy clusters, we merge these overlapping and redundant clusters to form visual subcategories. 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 LSVM detector [74] 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 3.2.

3.3.1 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

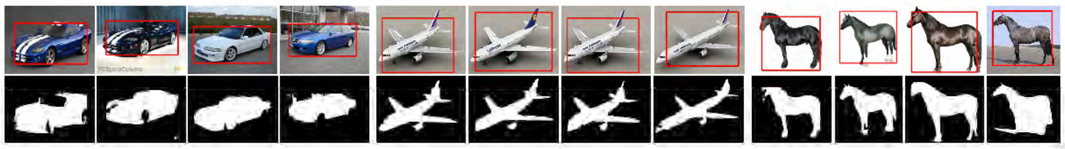


Figure 3.3: (top) Examples of strongly aligned and visually coherent clusters that we discovered. (bottom) We also show the result of our modified co-segmentation approach on these clusters.

approach with strong co-location constraints and obtain seed segments in the dataset. But how do we discover visually coherent and aligned clusters? One naive approach would be to sample a random set of patches and then cluster these patches using standard k-means. However, in the case of random patches it is extremely unlikely to hit multiple occurrences of the same object in a well-aligned manner unless we sample hundreds of thousands of windows per image. On this scale, clustering approaches tend to give incoherent clusters as shown by recent approaches [61]. Motivated by recent work on discriminative clustering via detection [43, 61, 253], we propose an approach to create coherent, aligned but overlapping and redundant clusters in the data.

Our approach is as follows: We first use every image as a cluster seed and we build clusters by detecting similar patches in the rest of the data. Specifically, we train an exemplar detector [97, 178] (eLDA in our case) based on Color-HOG (CHOG) features [126]. 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 serve as the basis for the follow-up 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 3.3(top) shows some examples of the well aligned clusters extracted using the above approach.

3.3.2 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).

Mathematically, we cast the problem as a classical graph cut problem to label every pixel in every image patch as foreground or background. Suppose we have n image patches $I_1, I_2, \ldots, I_n$ that belong to one cluster, each pixel-feature $x_{i,p}$ (for the pixel p) should be labeled as either foreground $c_{i,p} = 1$ or background $c_{i,p} = 0$, where p denotes its location in image i. A labeling C of all the pixels corresponds to a segmentation. We define an energy function over pixels and labels, and the optimal labeling is the one with minimum energy.

The energy function E has four terms, leveraging the instance-level cues and cluster-level cues in a coherent way. The first term $E(i, p; A_i)$ is the unary potential from an appearance model specific to image i, and the second term $E(i, p; A_S)$ is the unary potential from an appearance model shared between all images in the cluster. An instance based appearance model A_i consists of two Gaussian mixture models (GMM), one for the foreground (used when $c_{i,p} = 1$) and one for the background (used when $c_{i,p} = 0$). Each component is a full-covariance Gaussian over the RGB color space. We learn the foreground and background appearance models using the pixels inside and outside the bounding box generated from detections during clustering step.

The third term $E(i, p, q; c_{i,p}, c_{i,q})$ is the pairwise potential where we define:

$$E(i, p, q; c_{i,p}, c_{i,q}) = \delta(c_{i,p} \neq c_{i,q}) e^{-\beta \mathrm{P}_E(x_p, x_q)}, \tag{3.1}$$

as the pairwise compatibility function between labels of pixels (p and q) based on the probability of having an intervening contour (IC) between them [243]. Intuitively, this term penalizes two pixels getting different labels if they do not have an IC between them.

Finally, we want the segmentation masks across the cluster to be aligned and consistent. In our approach, it is modeled as a prior over the pixels: $\mathrm{P}_M(c_p|L_S, p)$ where L_S is the average segmentation mask across the aligned cluster. This denotes the prior probability that each pixel belongs to foreground or background, given the pixel location and the average cluster mask. In terms of energy, the fourth term can be defined as:

$$E(i, p; L_S) = -\log(\mathrm{P}_M(c_p|L_S, p)). \tag{3.2}$$

Since we do not know the segmentation prior (L_S) and appearance models before segmentation, we iterate between the global optimal graph cut step for each image and re-estimating the model parameters and location prior (by taking the mean) until the algorithm converges. Figure 3.3(bottom) shows some examples of segmentations obtained for the visually coherent clusters.

3.3.3 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 naive 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. To compute the segmentations, we exploit the top-down segmentation priors from the previous step.

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, or 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. Therefore, we first create a detector-detection matrix $S \in \mathbb{R}^{N \times M}$, where N is the number of detectors and M is the number of detections. Each entry $S_{i,j}$ in the matrix is filled by the detection score of detector i firing on patch j. Each row i in this matrix can be viewed as a signature of the detector. We then cluster the detectors based on these detection signatures. After normalization, we take the eigenvectors that correspond to the largest eigenvalues of the normalized S and apply k-means clustering to get the cluster index for detectors. Finally, we learn a LSVM detector for each merged cluster.

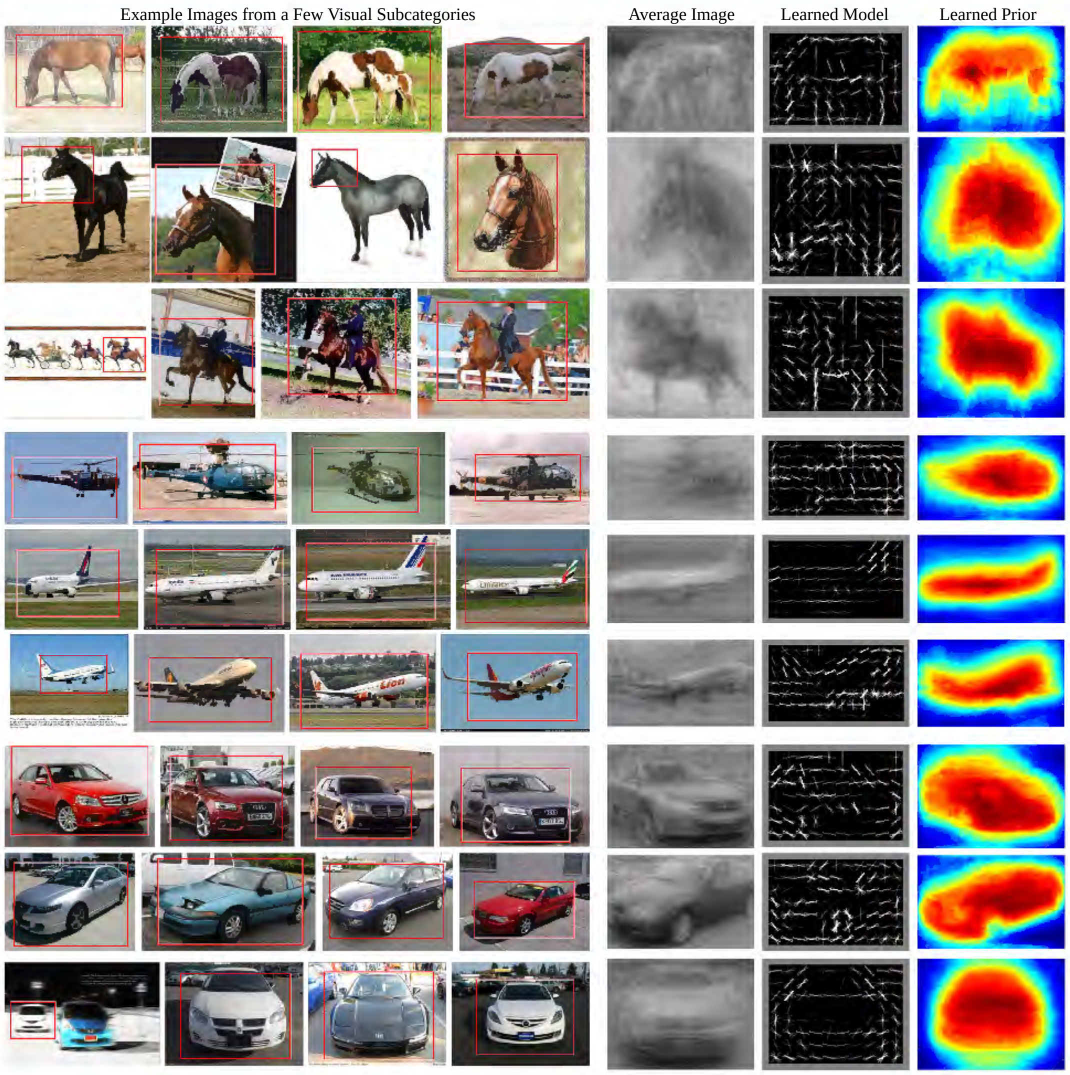


Figure 3.4: Examples of visual subcategories obtained after merging clusters. We show few instances, the average images, learned Latent SVM model and the segmentation prior for each subcategory.

3.3.4 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 [74] 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

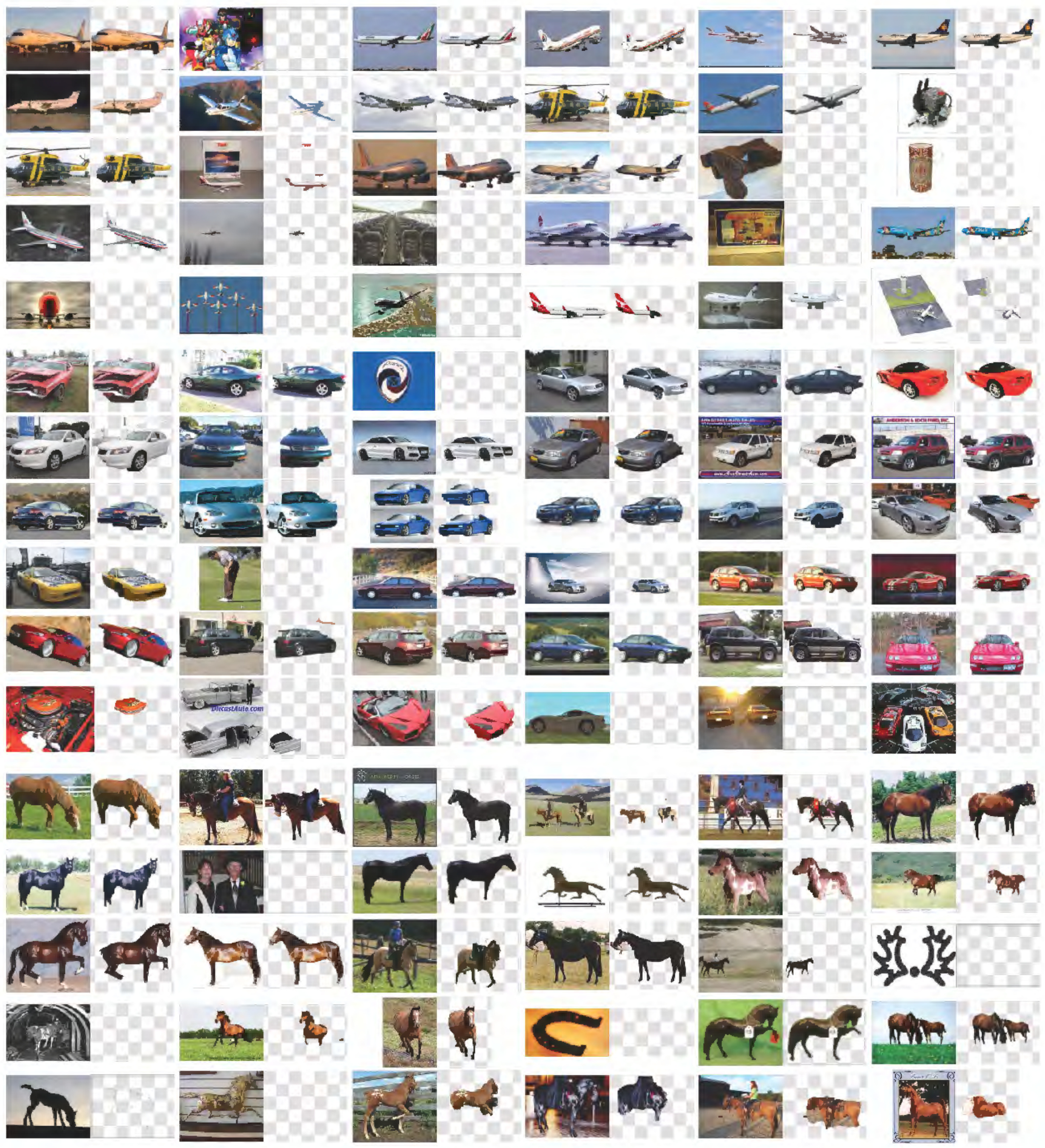


Figure 3.5: Qualitative results on discovering objects and their segments from noisy Internet images. We show results on three categories: car, horse, and airplane. The last row in each result shows some failure cases.

state-of-the-art results on the challenging Internet Dataset [232].

Table 3.1: Performance evaluation on the entire Internet dataset.

	Car		Horse		Airplane	
	P	J	P	J	P	J
[232]	83.38	63.36	83.69	53.89	86.14	55.62
eLDA	85.56	**70.61**	85.86	56.98	85.25	55.31
K-Means	82.11	54.35	87.02	52.99	86.08	51.18
NEIL subcategories	85.49	63.09	82.98	51.49	85.23	50.02
Ours	**87.09**	64.67	**89.00**	**57.58**	**90.24**	**59.97**

Table 3.2: Performance evaluation on the subset of Internet dataset (100 images per class).

	Car		Horse		Airplane	
	P	J	P	J	P	J
[120]	58.70	37.15	63.84	30.16	49.25	15.36
[121]	59.20	35.15	64.22	29.53	47.48	11.72
[127]	68.85	0.04	75.12	6.43	80.20	7.90
[232]	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

3.4 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 [13] 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 [232] 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 3.5 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.

3.4.1 Quantitative Evaluation

We now quantitatively evaluate the performance of our approach and compare against the algorithm of [232]. Note that most co-segmentation algorithms cannot scale to extremely large datasets and hence we focus on comparing against [232]. 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 3.1 shows the result on the entire dataset. Our algorithm substantially outperforms the previous state-of-the-art algorithm [232] on segmenting Internet images.

To understand the importance of each component, we perform detailed ablative analysis. We use the following one-step clustering baselines: (a) No Merging Step (eLDA): Directly using eLDA results followed by pooling the segmentation; (b) No eLDA Step (k-means): Directly using visual subcategories obtained using k-means; (c) No eLDA Step (NEIL subcategories): Using NEIL based clustering [43] to obtain visual subcategories. Our results indicate that the two-step clustering is the

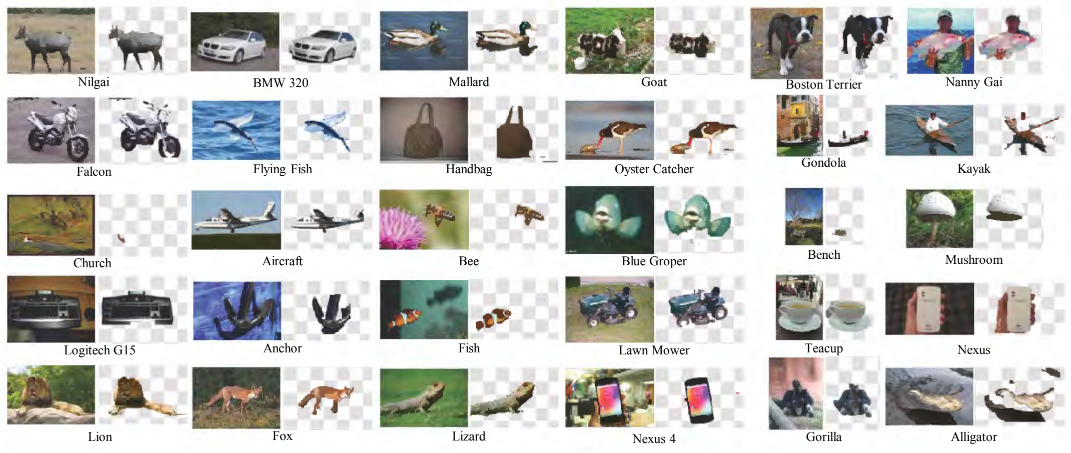


Figure 3.6: Qualitative results on discovering objects and their segments in NEIL [43]. The last column shows some failure cases.

key to obtain high performance in joint segmentation. Finally, we also tried using HOG instead of CHOG and it gave almost similar performance (0.5% fall in P and no fall in J).

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 [232]. This experiment also allows us to compare against the other co-segmentation approaches. Table 3.2 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, Rubinstein *et al*. 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.

We integrated this object discovery and segmentation algorithm in NEIL [43]. As of 14th April 2014, NEIL has automatically generated approximately $500K$ segmentations using web data. Figure 3.6 shows some segmentation results from NEIL.

Chapter 4

Sense Discovery

4.1 Introduction

While NEIL [43] and its text conterpart NELL [32] have shown much promise for building explicit knowledge bases automatically by learning knowledge from free text and images on the web, one issue that limits their performance is the problem of semantic and "visual" polysemy. Polysemy is the capacity for a word or a noun phrase (NP) to have multiple semantic meanings and visual meanings as well. For example, the NP `Apple` can refer to both the COMPANY and the FRUIT. Similarly, in the visual world, `Apple` can refer to images of the FRUIT, the COMPANY LOGO, and even IPHONES and IPADS. Therefore, handling polysemy by extracting multiple senses of a word/NP is an important problem that needs to be addressed.

One obvious way to handle semantic polysemy is to fall back to human developed knowledge bases such as Wordnet [192], Freebase [152] and even Wikipedia [47]. These broad-coverage knowledge bases suffer from the problem of missing information. For example, WordNet has good coverage of common nouns, however it has been criticized for being too fine-grained [258]. In addition it contains very few named entities (people, locations, organizations, *etc.*); Wikipedia and Freebase help to bridge this gap, but a great deal of information is still missing [227]. Furthermore, WordNet or Freebase have little or no information related to visual senses and still require extensive manual labeling to create mappings between semantic and visual senses. In contrast, unstructured data sources such as images and text from the web are much larger and more diverse, which can be readily used to discover both semantic and visual senses.

Instead of relying on manually-compiled resources, we focus on automatically discovering multiple senses of a NP in an unsupervised manner. The common unsupervised paradigm is to represent each NP in terms of text features or image features and then cluster these instances to obtain multiple semantic and visual senses of the NP respectively. Since the semantic and visual senses of a NP are closely related, many approaches have also attempted jointly clustering images and text. Most joint clustering approaches make the simplifying assumption that there exists a one-to-one mapping between semantic and visual senses of a word. This assumption rarely holds in practice, however. For example, while there are two predominant semantic senses of the word `Apple`, there exist multiple visual senses due to appearance variation (GREEN *vs.*RED APPLES), viewpoint changes, *etc.*

We present a generalized co-clustering algorithm that jointly discovers both semantic and visual senses for a given NP . For a given NP (such as `Apple`), we first download web pages which contain both image and text references to the NP. Each web page is treated as a data point and represented in terms of image and text features. We then use our co-clustering algorithm which clusters data

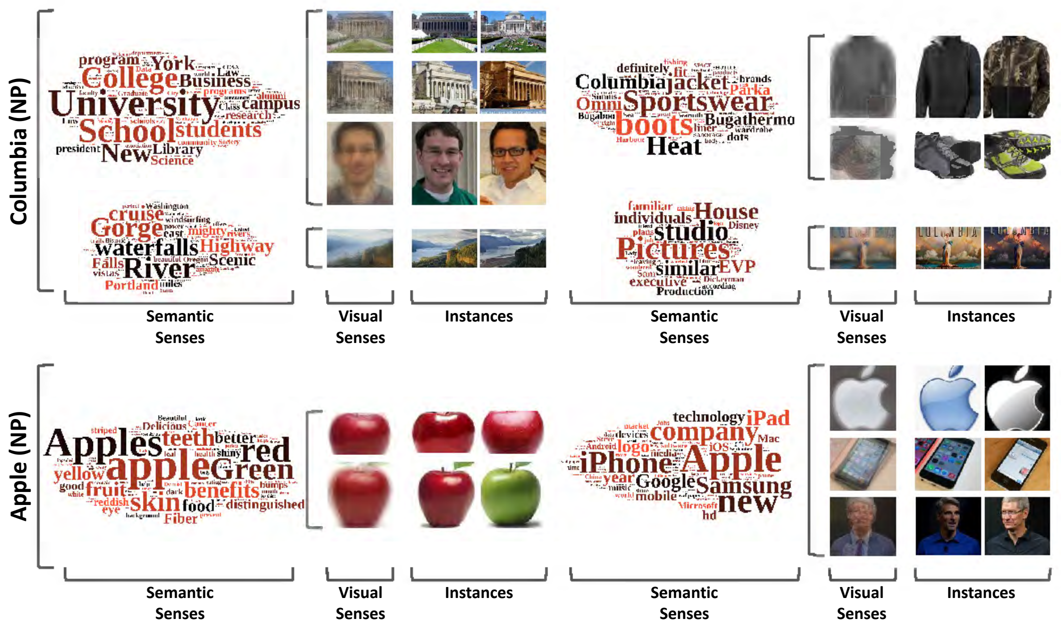


Figure 4.1: We present a co-clustering algorithm that discovers multiple semantic and visual senses of a given NP. In the figure above, we show the multiple senses discovered for the NPs `Columbia` and `Apple`. In the case of `Columbia`, our approach automatically discovers four semantic senses: UNIVERSITY, RIVER, SPORTSWEAR, STUDIO. In case of `Apple`, it discovers two semantic senses: FRUIT and COMPANY. Our approach also discovers multiple visual senses. For example, the SPORTSWEAR sense of Columbia corresponds to two visual senses: JACKET and SHOES. Semantic senses are shown as word clouds with size of each word being proportional to its importance. Visual senses are shown as average images of members belonging to the cluster.

points in image and text feature space separately and learns a one-to-many mapping between the clusters in two feature spaces[1] (See Figure 4.1). We demonstrate the effectiveness of our approach using four different experiments including a large scale experiment of co-clustering on ∼2000 NEIL NPs. We show how the joint space provides constraints that lead to high purity clusters. But more importantly, this joint learning process allows us to infer an alignment between the semantic and visual senses of the NP.

Why use images and text? We believe that the information in images and text is complementary and one needs to harness both to build a system that robustly discovers multiple senses of a word. For example, using images alone, it is almost impossible to differentiate viewpoint changes from conceptual changes. Similarly, using text based systems alone it is hard to differentiate similar semantic meanings. An example of this is the BIKE and CAR meaning of the word `Falcon`. In this case, text-based features tend to cluster the BIKE and the CAR sense together since both are vehicles but co-clustering in the joint space helps us to differentiate between the two.

[1]This can also be formulated as hierarchical clustering where higher level nodes correspond to clusters in text space and lower level nodes correspond to clusters in visual space.

4.1.1 Contributions

Our contributions include: (a) We introduce the problem of joint extraction of semantic and visual senses for given noun phrases and provide a novel formulation of this problem. We demonstrate how joint extraction of senses not only improves clustering but also helps us extract relationships between semantic and visual meanings of words. (b) We propose a generalized co-clustering algorithm where the two domains need not have the same granularity of clustering. We achieve this by enforcing a one-to-many constraint during the clustering process instead of a one-to-one mapping. (c) Finally, we introduce a new challenging dataset called CMU Polysemy-30, containing 30 NPs for the polysemy problem.

4.2 Related Work

A significant amount of previous work has investigated the problem of automatically inducing word senses from statistics derived from text corpora [28, 135, 208]. This approach has been quite successful given the small amount of prior knowledge provided: *e.g.*, Yarowsky [308] proposed an unsupervised approach for word sense disambiguation but suggested the use of dictionary definitions as seeds. But as pointed out before, most knowledge bases still suffer from a great deal of missing information [227].

Extracting visual senses of polysemous words using web images is an extremely difficult problem. There have been early efforts on automatically training visual classifiers using web data to build large datasets automatically [89, 155, 158, 240, 276], find iconic images [20, 220] and improve image retrieval results [75, 173, 217, 289]. Inspired by the success of mixture models for object detection [74], some recent approaches such as [43, 59] have also explored clustering web data and training detection models. For example, NEIL [43] performs clustering in visual appearance space to generate visual subcategories. But since NEIL only uses visual information, it cannot differentiate between semantic and visual polysemy: that is, it cannot label if two clusters correspond to same semantic meaning. On the other LEVAN [59] uses Google N-grams to first discover different senses for each NP. However, each N-gram leads to a different visual cluster, which results in a lot more clusters than NEIL, *e.g.*, hundreds of senses for the NP `Horse`. But similar to NEIL, clustering visually different N-grams (EATING HORSE and JUMPING HORSE) into one semantic cluster would require using further text information.

To handle these problems, past work has also focused on using both images and text on the web for discovering visual and semantic senses. For example, Schroff *et al.* [240] incorporates text features to rerank the images before training visual classifiers. In another work, Berg *et al.* [21] discovers topics using text and then use these topics to cluster the images. However, their approach requires manually selecting the topics for each category. Saenko *et al.* [235, 236] presented a model for learning visual senses using images clustered with text, but rely on WordNet's sense inventory. Another approach [289] uses Wikipedia to find the senses of a word. Lucchi *et al.* [173] used the click-through data and human relevance annotations to learn multiple senses. But the scalability and coverage of such knowledge bases and human annotations is questionable, and therefore in this work we focus on unsupervised approaches. Leoff *et al.* [151] focused on discovering visual senses in completely unsupervised manner by building a joint space of image and text features followed by clustering in this joint space. In addition, Barnard *et al.* [12] looked at the complementary problem of discovering semantic senses using image data. In our work, we propose an approach to jointly discover multiple semantic and visual senses from web data. We demonstrate that a joint solution (with a one-to-many mapping) allows us to improve the clustering performance and extract relationships between the semantic and visual senses of a NP.

Finally, our work is also closely related to approaches in co-clustering [53, 57] and multi-view clustering based approaches [52]. Previous approaches, however, assume a one-to-one mapping between clusters in two spaces. Instead, we relax this assumption and propose a co-clustering based approach where the mapping between clusters in two domains is one-to-many.

4.3 Co-Clustering

Given two domains $\mathcal{D}_1$ and $\mathcal{D}_2$, our goal is to jointly cluster the instances in both domains. Previous approaches have tackled this problem by augmenting feature spaces and performing clustering in the joint space. Other approaches have assumed a one-to-one mapping between the clusters in the two domains [53]. In many scenarios, however, the domains have different granularities and therefore a one-one mapping proves too strong an assumption. For example, if one considers semantic, visual and audio domains, the granularity in each domain is quite different. A cluster in the semantic domain might correspond to multiple clusters in both visual (due to viewpoint, appearance differences *etc.*) and audio domains (due to difference in pronunciations). We break the one-one mapping restriction and propose a generalized co-clustering algorithm.

4.3.1 Formulation

Let us assume that this one-to-many mapping exists from $\mathcal{D}_1$ to $\mathcal{D}_2$. The input to the algorithm is N data points with each point being represented as $X_i =< x_i^1, x_i^2 >$ (x_i^d is the feature representation of the i^{th} data point in domain $\mathcal{D}_d$). The output of our clustering algorithm is a set of clusters in each domain (defined in terms of an assignment of data points to each cluster) and a one-to-many mapping between the clusters in two domains.

We represent the clusters and the one-to-many mapping as a bipartite graph $G = (V^1, V^2, E)$, where V^1 and V^2 are the set of clusters in domain $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively. E represents the set of edges between clusters in V^1 and V^2; therefore, $E_{k,l} = 1$ indicates cluster k in $\mathcal{D}_1$ corresponds to cluster l in $\mathcal{D}_2$. We enforce the one-to-many constraint by ensuring that for each l, $\sum_k E_{k,l} = 1$. Each cluster node (k, l) in domain $(\mathcal{D}_1, \mathcal{D}_2)$ is associated with model parameters (θ_k^1, θ_l^2).

For each data point $X_i =< x_i^1, x_i^2 >$, its cluster membership is represented by a corresponding pair of cluster labels $Y_i =< y_i^1, y_i^2 >$ (y_i^d represents the membership of i^{th} data point in domain $\mathcal{D}_d$). Therefore, given X, our goal is to infer G^*, Θ^*, Y^* such that it maximizes the scoring function $\mathcal{S}(G, \Theta, Y, X)$, which is defined as:

$$\sum_{d \in \{\mathcal{D}_1, \mathcal{D}_2\}} \left(\overbrace{\sum_i \Psi^d(x_i^d, y_i^d, \Theta^d)}^{\text{data likelihood}} + \overbrace{\sum_{i,j} \Phi^d(x_i^d, x_j^d, y_i^d, y_j^d)}^{\text{smoothness}} \right) + \overbrace{\sum_{i,j} \Phi^{12}(x_i^2, x_j^2, y_i^1, y_j^1)}^{\text{cross-domain}}.$$

The first term in the scoring function corresponds to the data likelihood term in the two domains. This term prefers coherent clusters within each domain independently, which are explained using the model parameters Θ^d. The second term in the scoring function is the smoothness term. This term attempts to ensure that if two data points x_i and x_j are similar in domain $\mathcal{D}_d$ they get assigned to the same cluster in that domain. Note that if the structure (*e.g.* number of clusters) is fixed, then the smoothing term is redundant, but here it is essential to (1) regularize the likelihood term and avoid trivial solution (one cluster for each data point - high likelihood and good mapping), and (2) provide both intra- and inter cluster distance metrics to make proper structure movements. The third and final term in the scoring function is the cross-domain term which indicates that if two instances are similar in domain $\mathcal{D}_2$, then these data points should be assigned to same cluster in domain $\mathcal{D}_1$. Note this term is the asymmetric due to asymmetric nature of one-to-many relationship between the two domains. In Section 4.4, we define the specific Ψ^d, Φ^d and Φ^{12} that instantiate this approach in our text-vision application.

4.3.2 Optimization using Iterative Approach

Optimizing the above equation is in general an NP-hard problem. We therefore use an iterative optimization approach inspired by structure-EM for maximizing the above scoring function.

Given a fixed graph structure (fixed number of clusters and mapping), we iterate over estimating Θ and Y using hard-EM style iterations [125] to maximize the scoring function $\mathcal{S}$. That is, given Θ, we perform inference to assign data points to nodes in each domain by estimating the membership variable Y. We enforce the one-to-many constraint using a cautious approach [308], dropping data points which are not congruent with the mapping in structure G. Specifically, we treat the low-scoring data points as noise and discard them. Once we have estimated membership Y, we use this new membership to estimate the new parameters Θ.

After estimating $\mathcal{S}(\cdot)$ for a given G_t, we then take a structure step. Here, we create proposals for changes in structure (splitting a node into two or merging two nodes into one). We greedily select the best proposal G using an approximation function. Using the newly proposed G_{t+1}, we re-estimate the scoring function $\mathcal{S}$ using EM over Θ and Y. If the newly estimated score is higher than the score at previous iteration, we accept the structure step and continue. If the estimated score is lower, we reject the structure step and switch back to $G_{t+t} = G_t$. For initialization, G_0, we use a single node in domain $\mathcal{D}_1$ and K nodes in domain $\mathcal{D}_2$ (We use a high-value of K to ensure that the clusters are consistent). Therefore, the structure steps are split proposals in domain $\mathcal{D}_1$ and merge steps in $\mathcal{D}_2$. The pseudo code is shown in Algorithm 1.

4.4 Discovering Semantic and Visual Senses

We now adapt our generalized co-clustering algorithm to the task of discovering multiple semantic and visual senses. The outline of our approach is shown in Figure 4.2. In this case, $\mathcal{D}_1$ is the text domain and $\mathcal{D}_2$ is the visual domain. Our input data points are obtained by querying Google Image Search for each NP and downloading the top 1000 web pages. We now describe our text and image features, the likelihood, smoothness and the cross-domain terms.

4.4.1 Text Domain

Given a NP and a web page containing the NP, we extract x_i^1 as follows: First, we use the Stanford parser [51] to perform syntactic parsing of the sentences. For each mention of the NP in the web page, we extract features which include dependency paths of length one and two steps away from the NP head. We also include bag-of-word (BOW) features from the web page. In many cases, the associated text might contain topics irrelevant to the NP. To handle this, the BOW representation is constructed based on the sentences which mention the NP. Note that we also use the part-of-speech tags as well to form the BOW representation (therefore, amber_ADJ is treated differently from amber_NN). This leads to a very high-dimensional feature vector. To address this, we use topic model [25] (learned from $1M$ web pages) and project the extracted BOW features to the topics to obtain the final unit-norm feature vector, x_i^1.

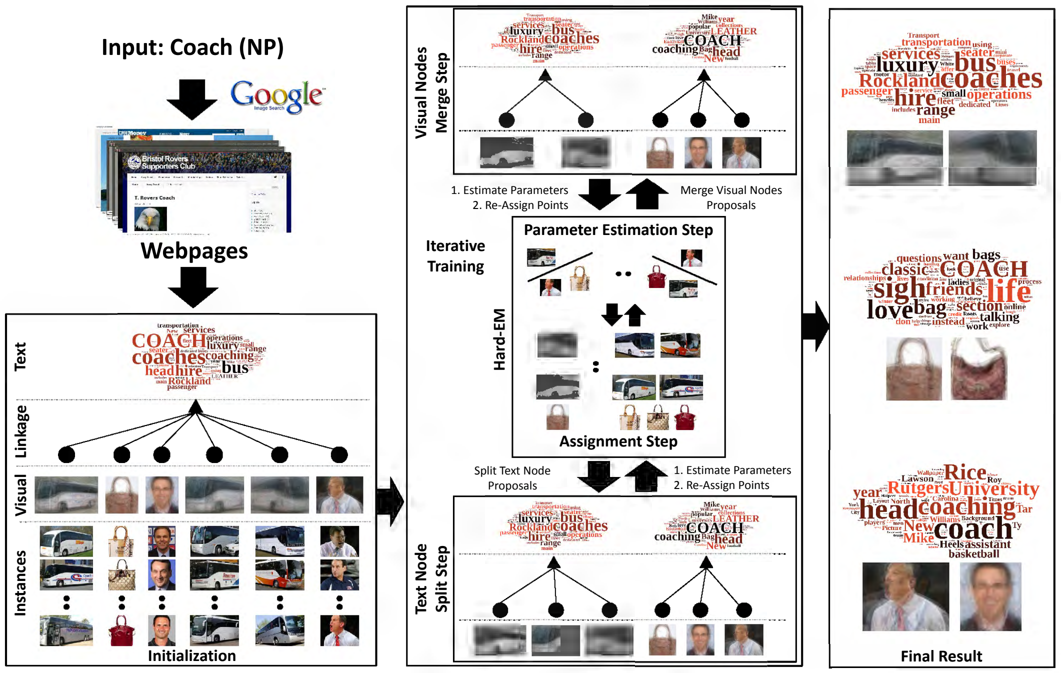


Figure 4.2: We discover semantic and visual senses using the structure-EM approach shown above. Given a NP (*e.g.*, `Coach`), we first download images and web pages. We then initialize the algorithm with a single semantic sense and with visual senses initialized using the clustering algorithm described in [43]. Then, at each iteration, semantic senses are refined and visual senses are generalized using an iterative approach.

Next, we discuss how each cluster is represented in the text domain and what are the associated parameters. We represent each text cluster with the mean feature vector of all the cluster members. Given this representation, we simply model the likelihood term as the histogram intersection $\chi(\cdot,\cdot)$ [11] of the mean and the input feature vector. We define the smoothness term as follows:

$$\Phi^1(x_i^1, x_j^1, y_i^1, y_j^1) = \sum_{i,j} \chi(x_i^1, x_j^1)\mathcal{I}(y_i^1 = y_j^1), \tag{4.1}$$

where $\chi(x_i^1, x_j^1)$ is histogram intersection similarity and $\mathcal{I}(\cdot,\cdot)$ is the indicator function. This term rewards the highly similar data points if they have the same label. The reward is proportional to the dot-product $\Delta(\cdot,\cdot)$ between the two feature points.

4.4.2 Image Domain

To represent the visual data, we first extract image based features. However, in the case of images, modeling the likelihood is quite tricky since the object location inside the image is unknown. To overcome this problem we use the algorithm for the clustering proposed in [44]. Given the set of input images for a NP, this algorithm generates the set of bounding boxes which are the location of objects in those images. It also clusters the visual data into K clusters which we use as initialization for G_0. Once the object location is known we represent the object (x_i^2) using HOG features [49].

Algorithm 1: Iterative Approach to Maximize Scoring Function

```
Input: Data points: $X$ where $X_i =< x_i^1, x_i^2 >$, $x_i^d$ = feature in Domain $d$
Output: Clustering in Two Domains: $Y_i =< y_i^1, y_i^2 >$, $(G,\Theta)$
// Initialization: 1 Cluster in $\mathcal{D}_1$, K Clusters in $\mathcal{D}_2$
$G_0, \Theta_0, Y_0 \leftarrow$ InitializeGraph(X,K)
$\mathcal{S}_0 \leftarrow \mathcal{S}(G_0, \Theta_0, Y_0)$ ;                    // Estimate Initial Score
while Rejects<R do
    // Propose New Structure $G_{t+1}$ based on Split/Merge Proposal
    $G_{t+1} \leftarrow$ GenerateNewProposal($G_t$, $Y$)
    // Use EM to estimate $\Theta_t$, $Y_t$
    while $Y_t$ not converged do
        // Estimate $\Theta_{t'+1}$ based on $G_{t+1}$ and $Y_{t'}$
        $\Theta_{t'+1} \leftarrow$ TrainNewClassifiers($G_{t+1}$,$Y_{t'}$)
        // Estimate $Y_{t'+1}$ based on $G_{t+1}$ and $\Theta_{t'+1}$
        $Y_{t'+1} \leftarrow$ AssignPointstoClusters($G_{t+1}$,$\Theta_{t'+1}$)
    // Estimate New Score $\mathcal{S}_{t+1}$
    $\mathcal{S}_{t+1} \leftarrow \mathcal{S}(G_{t+1}, \Theta_{t+1}, Y_{t+1})$
    if $\mathcal{S}_{t+1} < \mathcal{S}_t$ then
        // Reject if Score Decreases
        $G_{t+1} \leftarrow G_t$, $\mathcal{S}_{t+1} \leftarrow \mathcal{S}_t$, $Y_{t+1} \leftarrow Y_t$, $\Theta_{t+1} \leftarrow \Theta_t$
        Rejects ← Rejects+1
    else
        // Accept the Proposal
return $G_t$, $\Theta_t$
```

Given HOG based representation of object, we represent each cluster in terms of a linear-SVM weight vector(θ_k^2). This linear-SVM is trained by treating cluster members as positive data points and bounding boxes from random images[2] as negative data points.

Therefore, the likelihood score of a point, x_i^2, coming from visual cluster k is defined as the $\theta_k^{2\,T} x_i^2$. For modeling the smoothness term, we compute the similarity ($\Delta(x_i^2, x_j^2)$) between two feature vectors x_i^2 and x_j^2 using the dot product on whitened HOG feature [97, 178]. Given this similarity metric, the smoothness term is similar to the text term where the reward is proportional to the similarity between the two images:

$$\Phi^2(x_i^2, x_j^2, y_i^2, y_j^2) = \sum_{i,j} \Delta(x_i^2, x_j^2)\mathcal{I}(y_i^2 = y_j^2). \tag{4.2}$$

4.4.3 Cross-Domain Term

The final term we need to model is the cross domain term. This term ensures that data points which are similar in the visual domain are assigned to same cluster in the text domain. The term is defined as follows:

$$\Phi^{12}(x_i^2, x_j^2, y_i^1, y_j^1) = \sum_{i,j} \Delta(x_i^2, x_j^2)\mathcal{I}(y_i^1 = y_j^1), \tag{4.3}$$

where $\Delta(\cdot,\cdot)$ is the similarity defined as the dot-product of whitened HOG features of the data point i and j.

4.4.4 Optimization

Given these terms in the scoring function, we now optimize using the structure-EM approach described in Section 4.3.2. In the structure step, we alternate between text split proposals and visual

[2]These random images are scene images obtained from Google Image Search.

Airbus A380	732	AK 47	654	Apple	635	Bass	804	Bean	902	Black Swan	507
Chicken	845	Coach	754	Columbia	779	Corolla	667	Daybed	763	F18	611
Falcon	831	Football	559	Jordan	662	Los Angeles Lakers	651	M 16	664	Mouse	838
Mitsubishi Lancer	663	Motorbike	850	Note	766	Robin	830	Sofa	585	Sparrow	859
Subway	803	Tuna	814	Wagon	824	Whitefish	817	Wolf	779	Yellow Tail	682

Table 4.1: CMU Polysemy-30 Dataset for Discovery of Visual and Semantic Senses

merge proposals. We now describe how we generate the split proposals for nodes in the text domain and how we generate merge proposals in the visual domain.

Split Proposals

Given the set of text nodes V^1, at every alternate iteration we generate proposals by splitting every text node into two nodes. These splits are generated based on the one-to-many mapping between the text node and the visual nodes. Intuitively, we try to create splits by generating new possible semantic senses based on one of the visual senses. Formally, let us suppose, a text node V_l^1 is linked to the visual nodes $V_{l_0}^2, \ldots, V_{l_m}^2$. We generate split proposals by selecting pair of visual nodes and training a text classifiers such that instances belonging to one visual node is treated as positive and the instances belonging to other visual node is treated as negative data. This allows us to create $\binom{m}{2}$ split proposals and we select the best proposal based on the regularized empirical risk of the trained classifier.

Merge Proposals

Given the set of visual nodes that belong to same semantic node, we create proposals for merging two visual nodes based on the likelihood scores. If members of visual cluster l receive high likelihood scores from the classifier associated with cluster l' and the two nodes are assigned to the same semantic node, then we create a proposal to merge the nodes l and l'.

4.4.5 Implementation Details

In order to handle noise in the Google Image Search results, we create an extra cluster/node on the vision side. This allows us to handle outliers in the clustering process. Unlike other clustering approaches which tend to partition the whole feature space, our approach only focuses on extracting semantic and visual senses from the subset of data which is considered high confidence in either domain. The low confidence data points are assigned to the noise cluster and are not considered part of the scoring function.

Some of the data points also have missing data from one domain. For example, we might have images but no text associated with it. Instead of ignoring such data points, we prefer to assign them based on image features alone. This is necessary as in many cases our visual clusters are data starved and using these extra data points help us learn better visual classifiers.

4.5 Experimental Results

We now show the effectiveness of our co-clustering algorithm using extensive experimental analysis. We perform four different experiments and implement several baselines. First, we introduce a new challenging dataset for this task (CMU Polysemy-30) from NEIL. This dataset consists of 30 NPs and $\sim$ 1000 web pages for each NP are downloaded from Google Image Search. We do a clean up

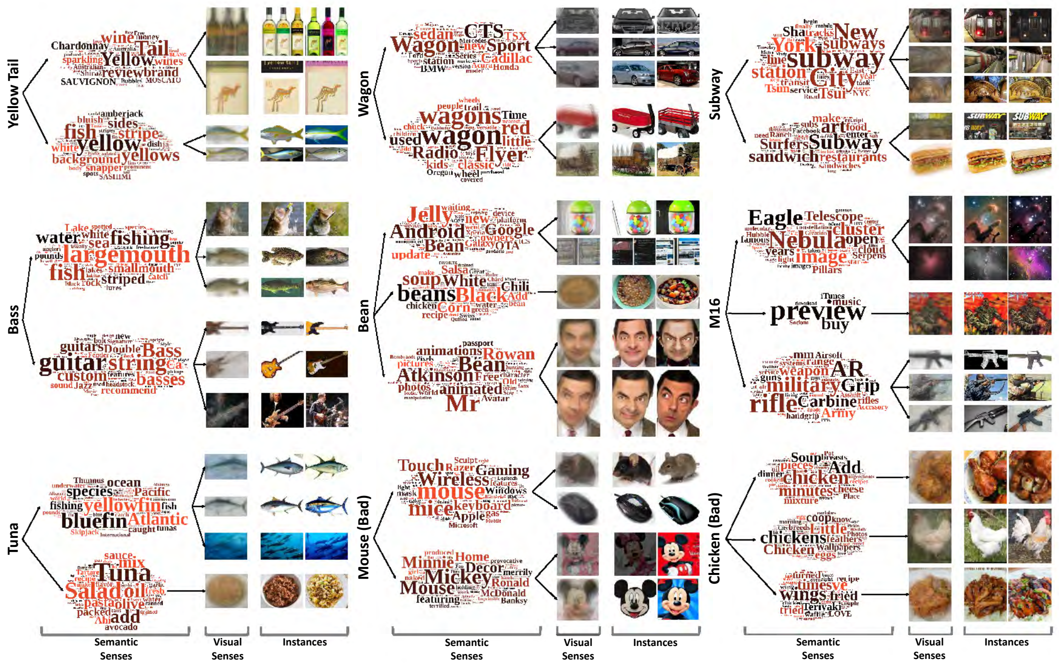


Figure 4.3: Examples of semantic and visual sense discovery of our algorithm. For example, our algorithm automatically discovers two semantic senses for `Bass`: FISH and MUSICAL INSTRUMENT. For the NP `Subway`, we find METRO-TRAIN and SANDWICH. For TUNA, our algorithm discovers FISH and FOOD. For `Bean`, it discovers semantic senses of JELLY BEANS, BEAN FOOD and MR. BEAN. Last two examples in the third row show two failure cases.

step and after accounting for broken links, web pages not reachable, we end up with $\sim$ 750 images per NPs. Table 4.1 shows the list of NPs and the number of data points for each NP.

In order to evaluate the performance of the sense extraction we manually listed all the possible semantic senses for each NPs. We then manually labeled $\sim$ 5600 instances with one of the listed semantic senses. We use accuracy (AC) as one way to measure the clustering performance. Before evaluation, we first obtain a mapping between the ground truth clusters and clusters obtained using Kuhn-Munkres algorithm [216]. We also use the standard normalized mutual information (NMI) [179] metric to evaluate our clustering. Note that higher mutual information implies better clustering performance.

To overcome the human labeling bottleneck, we also perform another experiment which creates pseudo-words to evaluate sense disambiguation [241] (Sec 4.5.3). Next, we perform a retrieval experiment on the MIT ISD dataset [236] which has 5 polysemous concepts (Sec 4.5.4). Finally, we perform a large-scale experiment where we run our algorithm on $\sim$ 2000 NPs (Sec 4.5.5). The list of these concepts is obtained using the NEIL knowledge base [43].

4.5.1 Baselines

We compare the performance of our approach against multiple baselines which use text and image features. For all the baselines, we use two versions: Pre-defined number of clusters (Fixed) and using Eigen-Gap criterion [287] (Eigen) to automatically compute the number of clusters. Our baselines

are:
Spectral Clustering on TF-IDF Features (TF-IDF): Our first baseline uses text based features only and constructs a feature representation of a web page using TF-IDF [179] features over the context words. These features are used in conjunction with cosine distance to create an affinity graph. Finally, we perform spectral clustering over this affinity graph.
Spectral Clustering on BOW (BOW): Our second baseline uses the BOW text features to represent each web page. Similar to our text feature construction, we build BOW over the word and their Part-of-Speech tags. We use histogram intersection as the similarity metric to create the affinity graph.
Image BOW over SIFT and Color Histogram (I-BOW): Our third baseline uses image-based features. Specifically, we use SIFT and Color Histogram features. To create a BOW representation, we perform vector quantization with 1000 words for SIFT and 400 words for Color histogram. Given this visual BOW representation, we create an affinity graph using histogram intersection similarity metric.
Spectral Clustering on Topic Model based Representation: Our fourth baseline uses text-based features used in our algorithm to represent a web page and histogram intersection is used as the similarity metric to create the affinity graph. Note that this baseline is an unsupervised variant of the approach taken by [236], which uses extra information (WordNet) to determine the underlying senses.
Clustering in Joint Space: As our final set of baselines, we implemented the algorithms [151, 289] which perform clustering in the joint space of image and text features. Leoff *et al*. [151] uses BOW for both images and text, while Wan *et al*. [289] uses topic model based representations.

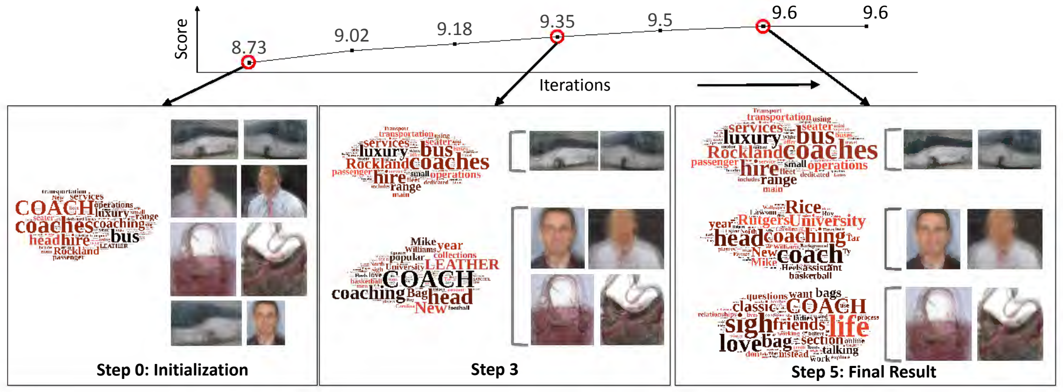
The code for Lucchi & Weston [173] is not available and requires click-through data to train which is proprietary. But we did compare qualitatively against Google Image Search query expansions and use human annotators to quantitatively compare performance against them.

4.5.2 CMU Polysemy-30 Dataset

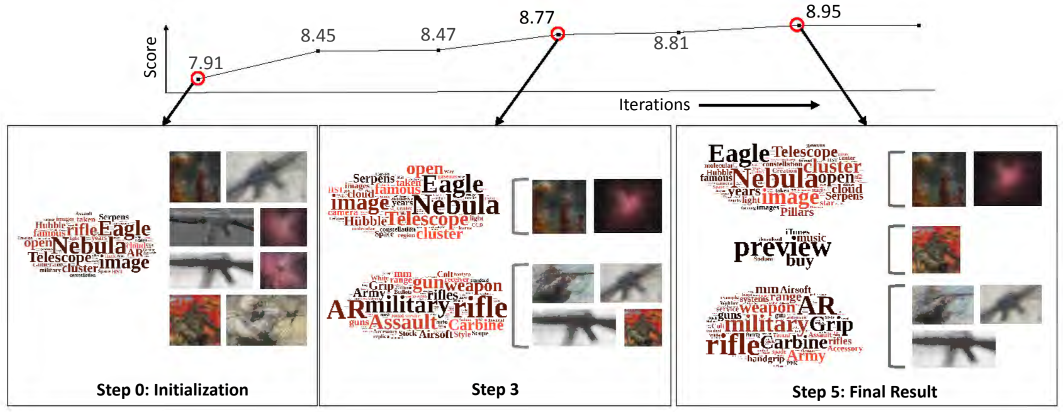
Qualitative Results: We first show qualitative performance of our algorithm. Figure 4.3 shows the qualitative result of our clustering algorithm on some NPs from CMU Polysemy-30: `Yellow Fish`, `Wagon`, `Subway`, `Bass`, `Bean`, `M16` and `Tuna`. Notice how our algorithm discovers the two semantic meanings of `Subway`: the METRO and the SANDWICH BRAND. Specifically, note the text features in the word clouds. For the METRO sense, the main distinguishing features include: New, York, Station, City, tracks, line, transit. For the SANDWICH BRAND sense, the main distinguishing text features are: sandwich, Surfers, restaurants, art, food, sandwiches. Also notice the associated visual senses. For example, for METRO, the visual senses are the STATION and the METRO TRAIN itself. Similarly, for SANDWICH BRAND, visual senses include the SUBWAY LOGO and the SANDWICH itself.

Another interesting example is the `M16` shown on middle right. Our algorithm excellently discovers the right semantic senses: NEBULA, MUSIC ALBUM and RIFLE. Notice the associated text features for each semantic sense. For the nebula sense, the most important words are: Eagle (also known as eagle nebula), Nebula, Telescope, cluster etc. For the music album sense, the important words are preview, buy and iTunes. Figure 4.3 also shows a couple of cases (last two, `Mouse` and `Chicken`) where our algorithm fails to discover all the associated senses.

Figure 4.4 shows how the value of scoring function changes with each iteration for the NP `Coach` and `M16`. For coach in (a), note that the first accepted step is the visual merge cluster which merges the two face clusters together. At the next step it splits the text cluster and now our algorithm has two senses for coach (bus, combination of trainer and handbag). By iteration 5, the algorithm



(a) NP: Coach



(b) NP: M16

Figure 4.4: How the scoring function $\mathcal{S}$ changes with each iteration for NP `Coach`. We also show some qualitative results at landmark iterations.

discovers all three semantic senses: BUS, TRAINER and HANDBAG. On the other hand, for `M16` the text split proposal is accepted first and the merge happens on the vision side.

We also qualitatively compared to Google Image Search query expansions. For example, for `Whitefish`, Google misses the LIGHTHOUSE LOCATION sense and only captures the RESORT sense. (see Figure 4.5).

Quantitative Results: We now discuss the quantitative results and compare the performance of our algorithm against several baselines. Table 4.2 shows the comparative performance for semantic sense discovery. As the quantitative results indicate, our approach outperforms all the baselines by a significant margin. Note that our approach automatically discovers the true number of semantic senses and outperforms the baselines even when the true number of senses are provided to the baselines (fix the number of clusters before spectral clustering) due to noise.

For the evaluation of visual senses, since it is hard to obtain ground truth labels, we computed the purity (how many instances belong to the same semantic sense) for all the visual senses obtained by [44] and our approach. As expected, our algorithm improves purity over iterations, giving 3%

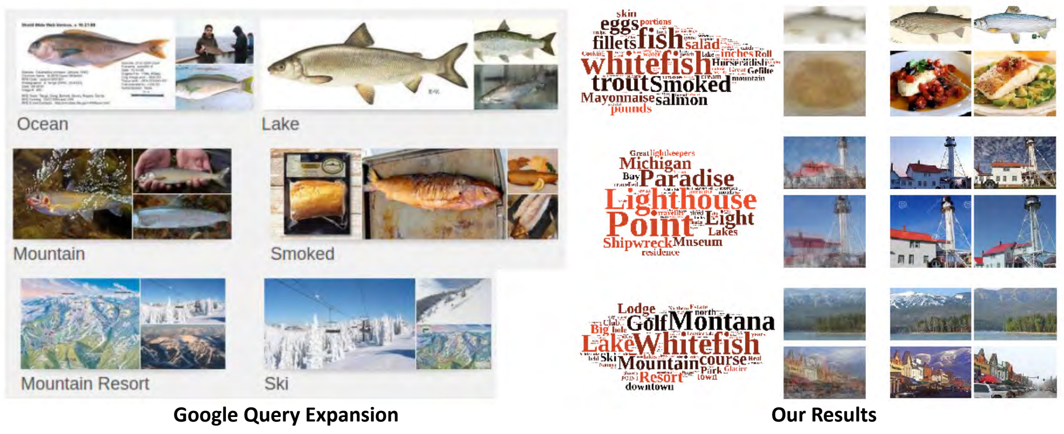

Ocean
Lake
Mountain
Smoked
Mountain Resort
Ski
Google Query Expansion
eggs fish fillets whitefish salad trout Smoked Mayonnaise salmon
Michigan Paradise Lighthouse Point Light Shipwreck Museum
Lodge Golf Montana Lake Whitefish Mountain course Resort
Our Results

Figure 4.5: Comparison with Google query expansion for NP `Whitefish`.

	AC		NMI	
	Fixed	Eigen	Fixed	Eigen
TF-IDF	79.31	76.66	11.04	19.65
BOW	77.94	70.79	15.97	12.64
I-BOW	77.86	70.17	6.84	8.04
[236]	80.52	73.21	18.69	17.75
[151]	78.39	79.89	8.96	27.00
[289]	80.62	73.13	19.03	18.26
Our Approach	**86.70**		**34.11**	

Table 4.2: Quantitative evaluation on the CMU Polysemy-30 dataset.

boost in clustering performance. The boost is significantly higher when two different senses have similar visual appearances (APPLE LOGO looks very similar to APPLE FRUIT).

4.5.3 Pseudo-word Based Evaluation

One bottleneck for the evaluation of sense disambiguation algorithms is the requirement of human labeling. A neat way out of this is the commonly used approach of pseudo-words [241]. More specifically, pseudo polysemous NPs can be created by combining together multiple single-sense NPs. For example, we can combine the web pages downloaded for `Accord` and `Boeing` (non-polysemous words) and treat them as retrieval results for a single pseudo polysemous word `Accord-Boeing`. Now, by construction, we have the labels for semantic sense since the web pages for `Accord` are sense 1 and web pages for `Boeing` are sense 2.

For our experiments, we combine four NPs: `Accord`, `Boeing`, `Tire`, `Cricket ball`. For these four NPs, there are $2^4 - 1 = 15$ possible combinations with these pseudo words having somewhere between 1 to 4 semantic senses per word. Table 4.3 show the comparative performance of our approach against all the baselines. Again, in this case, our approach outperforms all the baselines significantly.

	AC		NMI	
	Fixed	Eigen	Fixed	Eigen
TF-IDF	53.69	49.12	10.32	6.07
BOW	58.98	68.95	15.97	31.64
I-BOW	61.66	57.47	22.03	16.03
[236]	70.04	77.20	46.55	44.52
[151]	54.83	55.52	12.80	18.72
[289]	69.08	72.83	32.80	43.33
Our Approach	**83.89**		**53.81**	

Table 4.3: Quantitative evaluation on the Pseudo-NP dataset.

4.5.4 MIT ISD Dataset

Next, we apply our unsupervised approach for the task of re-ranking image search engine outputs. We use MIT ISD Dataset [236], which was collected automatically from the Yahoo Image Search. It consists of 5 NPs: `Bass`, `Face`, `Mouse`, `Speaker` and `Watch`. For image retrieval, one target sense is picked for each NP: the FISH, the HUMAN FACE, the POINTING DEVICE, the AUDIO DEVICE, and the TIMEPIECE. We evaluated the retrieval performance with Area Under the Curve (AUC). Compared to Yahoo Image Search, our unsupervised approach is able to obtain 20% performance gain on average.

4.5.5 Large-Scale Sense Discovery Experiments on NEIL

Finally, our sense discovery approach is linear in the number of categories and therefore scales reasonably well. As an extension of the CMU Polysemy-30 dataset, we also evaluate our algorithm on 1.8 million images and websites from Google Image search, using a list of $\sim$ 2000 NPs from NEIL [43] as queries. We randomly evaluated the sense discovery for 100 keywords and found our algorithm can recover 82% of senses from Wikipedia.

Part II

Learning Implicit Visual Knowledge

Chapter 5

Using Query for Supervision

5.1 Introduction

The previous chapter concludes our effort in building explicit[1], structured knowledge base from web data in a more automatic and scalable way. Now we shift our attention to the next topic: Learning implicit visual commonsense. Different from explicit knowledge where we have convenient sources (*e.g.* the Internet) to learn from, implicit knowledge refers to a latent reprentation and is almost never written down anywhere by humans.

Yet, the exciting news came that, standard vision datasets, such as ImageNet [233] created by harnessing human intelligence to filter out the noisy images returned by search engines has already helped significantly advance the performances on relevant tasks [74, 87, 136, 312]. Part of the reason is the introduction of representation learning using convolutional networks. For example, training a convolutional network on ImageNet followed by fine-tuning on PASCAL-VOC led to state-of-the-art performance on the object detection challenge [87, 136]. While on the surface, such a fine-tuning approach for detection completely gets rid of explicit knowledge base, we believe because it is pretrained on ImageNet before, the weights in the convolution filters have encoded, or memorized the ImageNet images in an implicit form of visual knowledge. Such an implicit visual commonsense greatly benefits down-stream tasks [87].

But human supervision comes with a cost and its own problems (*e.g.* inconsistency, incompleteness and bias [273]). Therefore, an alternative, and more appealing way is to learn visual commonsense from the web data *directly*, without using any manual labeling. At the same time, while web data based systems like NEIL had shown much promise, it was so far only worked on small datasets constructed from web images [43, 44] rather than standard testbeds like PASCAL VOC [67]. Web images also need to justify themself for being used effectively. However, the big question is, can we actually use millions of images online without using any human supervision?

In fact, researchers have pushed hard to realize this dream of learning implicit visual representations from web data. These efforts have looked at different aspects of webly supervised learning such as:

- **What are the good sources of data?** Researchers have tried various search engines ranging from text/image search engines [21, 75, 282, 291] to Flickr images [202].
- **What types of data can be exploited?** Researchers have tried to explore different types of data, like images-only [43, 156], images-with-text [21, 240] or even images-with-n-grams [59]).

[1]For more developments on learning the temporal aspect of visual categories, please refer to Sigurdsson *et al.* [250]

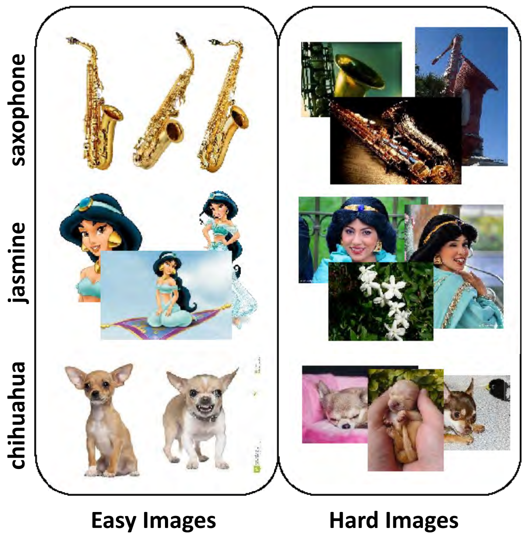


Figure 5.1: We investigate the problem of training a webly supervised network. Two types of visual data are freely available online: image search engine results (left) and photo-sharing websites (right). We train a two-stage network bootstrapping from clean examples retrieved by Google and enhanced by potentially noisy images from Flickr.

- **How do we exploit the data?** Extensive algorithms (*e.g.* probabilistic models [76, 156], exemplar based models [43], deformable part models [59], self organizing map [89] *etc.*) have been developed.
- **What should we learn from web data?** Finally, there has been lot of effort ranging from just cleaning data [69, 202, 297] to training concept representations or models [156, 158, 275], to even discovering contextual relationships [43].

Nevertheless, to the best of our knowledge, while many of these systems have seen orders of magnitudes larger number of images, their performance has never shown to match up against contemporary methods that receive extensive supervision from humans. Why is that?

Of course the biggest issue is the data itself: (1) It contains noise, and (2) Is has bias - image search engines like Google usually operate in the high-precision low-recall regime and tend to bias toward images where a single object is centered with a clean background and a canonical viewpoint [20, 166, 186]. But is it just the data? We argue that it is not just the data, but also the ability of algorithms to learn from large data sources and generalize. For example, traditional approaches which use hand-crafted features (*e.g.* HOG [43]) and classifiers like support vector machines [59] have very few parameters (less capacity to memorize) and are therefore unlikely to effectively use large-scale training data. On the other hand, memory based nearest neighbors classifiers can better capture the distribution given a sufficient amount of data, but are less robust to the noise. Fortunately, ConvNets [136] have resurfaced as a powerful tool for learning from large-scale data: When trained with ImageNet [233] ($\sim$1M images), it is not only able to achieve state-of-the-art performance for the same image classification task, but the learned representation can be readily applied to other relevant tasks [87, 312]. Delving inside the model, it was shown to actually learn concept detectors in its mid-level neurons that are relevant to the categories of interest [310, 312].

Attracted by its amazing capability to harness large-scale data, in this chapter, we investigate

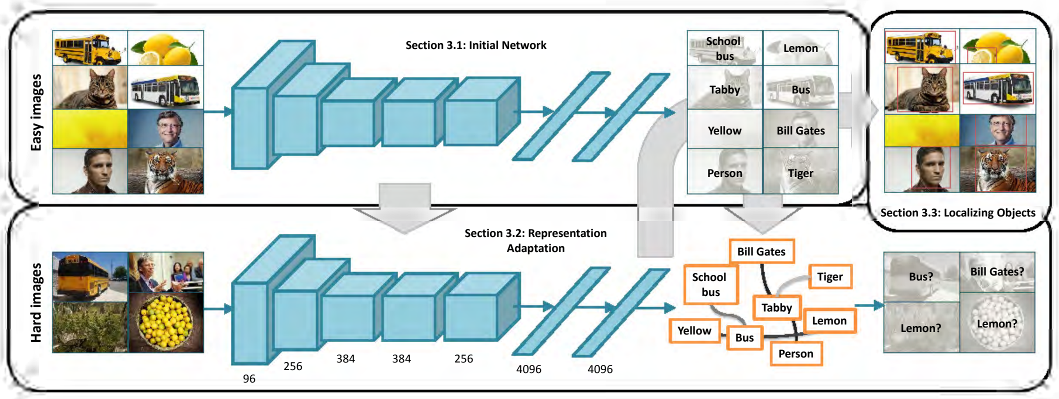


Figure 5.2: Outline of our approach. We first train a ConvNet using easy images from Google Image Search (above). This ConvNet is then used to find relationships and initialize another network (below) which will train on harder scene images on the web. Finally, we use this network to localize objects in the images and train RCNN detectors by using ConvNet features from our network.

webly supervised learning for ConvNets (See Figure 5.1) to learn implicit visual commonsense. Specifically: (1) We present a two-stage webly supervised approach to learning ConvNets. First we show that ConvNets can be readily trained for easy categories with images retrieved by search engines, no bells or whistles. We then adapt this network to hard (Flickr style) web images using the relationships discovered in easy images; (2) We show webly supervised ConvNet also generalizes well to relevant vision tasks, giving state-of-the-art performance compared to ImageNet pretrained ConvNets if there is enough data; (3) We show state-of-the-art performance on VOC data without using a single VOC training image - just using the images from the web. (4) We also show competitive results on scene understanding.

To the best of our knowledge, this is one of the first papers to achieve competitive or even better object detection performance than ImageNet pretrained ConvNets for the same model architecture. We believe this opens up avenues for exploitation of web data to achieve next cycle of performance gain in vision tasks (and at no human labeling costs!).

5.1.1 Why Webly Supervised?

Driven by ConvNets, the field of object detection has seen a dramatic churning in the past two years, which has resulted in a significant improvement in the state-of-the-art performance. But as we move forward, how do we further improve performance of ConvNet-based approaches? We believe there are two directions. The first and already explored area is designing deeper networks [252, 269]. We believe a more juicier direction is to feed more data into these networks (in fact, deeper networks would often need more data to train). But more data needs more human labeling efforts. Therefore, if we can exploit web data for training ConvNets, it would help us move from million to billion image datasets in the future. In this chapter, we take the first step in demonstrating that it is indeed possible to have competitive or even better performance to ImageNet pretrained ConvNets by just exploiting web data at much larger scales.

5.2 Related Work

Mining high-quality visual data and learning good visual representation for recognition from the web naturally form two aspects of a typical chicken-and-egg problem in vision. On one hand, clean and representative seed images can help build better and more powerful models; but on the other hand, models that recognize concepts well are crucial for indexing and retrieving image sets that contain the concept of interest. How to attack this problem has long been attractive to both industry and academia.

From models to data: Image retrieval [256, 257] is a classical problem in this setting. It is not only an active research topic, but also fascinating to commercial image search engines and photo-sharing websites since they would like to better capture data streams on the Internet and thus better serve user's information need. Over the years, various techniques (*e.g*. click-through data) have been integrated to improve search engine results. Note that, using pretrained models (*e.g*. ConvNet [297]) to clean up web data also falls into this category, since extensive human supervision has already been used.

From data to models: A more interesting and challenging direction is the opposite - can models automatically discover the hidden structures in the data and be trained directly from web data? Many people have pushed hard in this line of research. For example, earlier work focused on jointly modeling images and text and used text based search engines for gathering the data [21, 236, 240]. This tends to offer less biased training pairs, but unfortunately such an association is often too weak and hard to capture, since visual knowledge is usually too obvious to be mentioned in the text [43]. As the image search engines mature, more recent work focused on using them to filter out the noise when learning visual models [59, 75, 89, 158, 275, 282, 291]. But using image search engines added more bias to the gathered data [23, 166, 186]. To combat both noise and data bias, recent approaches have taken a more semi-supervised approach. In particular, [43, 156] proposed iterative approaches to jointly learn models and find clean examples, hoping that simple examples learned first can help the model learn harder, more complex examples [17, 139]. However, to the best of our knowledge, human supervision is still a clear winner in performance, regardless of orders of magnitudes more data seen by many of these web learners.

Our work is also closely related to another trend in computer vision: Learning and exploiting visual representation via ConvNets [87, 98, 136, 270]. However, learning these ConvNets from noisy labeled data [224, 264] is still an open challenge. Following the recent success of convolutional networks and curriculum learning [17, 139, 148], we demonstrate that, while directly training ConvNets with high-level or fine-grained queries (*e.g*. random proper nouns, abstract concepts) and noisy labels (*e.g*. Flickr tags) can still be challenging, a more learning approach might provide us the right solution. Specifically, one can bootstrap ConvNet training with easy examples first, followed by a more extensive and comprehensive learning procedure with similarity constraints to learn visual representations. We demonstrate that visual representations learned by our algorithm performs very competitively as compared to ImageNet trained ConvNets.

Finally, our work is also related to learning from weak or noisy labels [46, 56, 207, 260, 290]. There are some works showcasing that ConvNets trained in a weakly supervised setting can also develop detailed information about the object intrinsically [22, 201, 209, 213, 251]. However, different from the assumptions in most weakly-supervised approaches, here our model is deprived of clean human supervision altogether (instead of only removing the location or segmentation). Most recently, novel loss layers have also been introduced in ConvNets to deal with noisy labels [224, 264]. On the other hand, we assume a vanilla ConvNet is robust to noise when trained with simple examples, from which a relationship graph can be learned, and this relationship graph provides powerful constraints when the network is faced with more challenging and noisier data.

5.3 Approach

Our goal is to learn deep representations directly from the massive amount of data online. While it seems that ConvNets are data-guzzlers - small datasets plus millions of parameters can easily lead to over-fitting, we found it is still hard to train a ConvNet naively with random image-text/tag pairs. For example, most Flickr tags correspond to meta information and specific locations, which usually results in extremely high intra-tag variation. One possibility is to use commercial text-based image search engine to increase diversity in the training data. But if thousands of query strings are used some of them might not correspond to a visualizable concept and some of the query strings might be too fine grained (*e.g.* random names of a person or abstract concepts). These non-visualizable concepts and fine-grained categories incur unexpected noise during the training process[2]. One can use specifically designed techniques [43, 59] and loss layers [224, 264] to alleviate some of these problems. But these approaches are based on estimating the empirical noise distribution which is non-trivial. Learning the noise distribution is non-trivial since it is heavily dependent on the representation, and weak features (*e.g.* HOG or when the network is being trained from scratch) often lead to incorrect estimates. On the other hand, for many basic categories commonly used in the vision community, the top results returned by Google image search are pretty clean. In fact, they are so clean that they are biased toward iconic images where a single object is centered with a clean background in a canonical viewpoint [20, 166, 186, 220]. This is good news for learning algorithm to quickly grasp the appearance of a certain concept, but a representation learned from such data is likely biased and less generalizable. So, what we want is an approach that can learn visual representation from Flickr-like images.

Inspired by the philosophy of curriculum learning [17, 139, 148], we take a two-step approach to train ConvNets from the web. In curriculum learning, the model is designed to learn the easy examples first, and gradually adapt itself to harder examples. In a similar manner, we first train our ConvNet model from scratch using easy images downloaded from Google Image Search. Once we have this representation learned we try to feed harder Flickr images for training. Note that training with Flickr images is still difficult because of noise in the labels. Therefore, we apply constraints during fine-tuning with Flickr images. These constraints are based on similarity relationships across different categories. Specifically, we propose to learn a relationship graph and initial visual representation from the easy examples first, and later during fine-tuning, the error can back-propagate through the graph and get properly regularized. To demonstrate the effectiveness of our representation, we do two experiments: (a) First, we use our final trained network using both Google and Flickr images to test on VOC 2007 and 2012 dataset. We use RCNN pipeline for testing our representations; (b) We train object detectors from the cleaned out web data and perform localization. These detectors are tested on standard VOC 2007 dataset. The outline of our approach is shown in Figure 5.2.

5.3.1 Initial Network

As noted above, common categories used in vision nowadays are well-studied and search engines give relatively clean results. Therefore, instead of using random noun phrases, we obtained three lists of categories from ImageNet Challenge [233], SUN database [298] and NEIL knowledge base [43]. ImageNet syn-sets are transformed to its surface forms by just taking the first explanation, with most of them focusing on object categories. To better assist querying and reducing noise, we remove the suffix (usually correspond to attributes, *e.g.* indoor/outdoor) of the SUN categories. Since NEIL

[2]We tried to train a network with search engine results of $\sim$ 7000 entities randomly sampled from web noun phrases but the network does not converge.

is designed to query search engines, its list is comprehensive and favorable, we collected the list for objects and attributes and removed the duplicate queries with ImageNet. The category names are directly used to query Google for images. Apart from removing unreadable images, no pre-processing is performed. This leave us with ~ 600 images for each query. All the images are then fed directly into the ConvNet as training data.

For fair comparison, we use the same architecture (besides the output layer) as the BLVC reference network [117], which is a slight variant of of the original network proposed by [136]. The architecture has five convolutional layers followed by two fully connected layers. After seventh layer, another fully connected layer is used to predict class labels.

5.3.2 Representation Adaptation with Graph

After converging, the initial network has already learned favorable low-level filters to represent the "visual world" outlined by Google Image Search. However, as mentioned before, this visual world is biased toward clean and simple images. For example, it was found that more than 40% of the cars returned by Google are viewed from a 45 degree angle, and horses rarely occur lying on the ground [186]. Moreover, when a concept is a product, lots of the images are wallpapers and advertisements with artificial background and the concept of interest centered (and of course, viewed from the best selling view). On the other hand, photo-sharing websites like Flickr have more realistic images since the users upload their own pics. Though photographic bias still exist, most of the images are closer-looking to the visual world we experience everyday. Datasets constructed from them are shown to generalize better [166, 273]. Therefore, as a next step, we aim to narrow the gap by fine-tuning our representation on Flickr images[3].

For fine-tuning the network with hard Flickr images, we again feed these images as-is for training, with the query words acting as class labels. While we are getting more realistic images, we did notice that the data becomes noisier. Powerful and generalizable as ConvNets are, they are still likely to be diluted by the noisy examples over the fine-tuning process. In an noisy open-domain environment, mistakes are unavoidable. But humans are more intelligent: We not only learn to recognize concepts independently, but also build up interconnections and develop theories to help themselves better understand the world [34]. Inspired by this, we want to train ConvNets with such relationships - with their simplest form being pair-wise look-alike relationships [43, 59].

One way to obtain relationships is through extra sources like WordNet [192] or Word2Vec [190]. However, they are not specifically developed for the visual domain we are interested in. Instead, we take a data-driven approach to discover such relationships in our data: We assume the network will intrinsically develop connections between different categories when clean examples are offered, and all we have to do is to distill the knowledge out.

We take a simple approach by just testing our network on the training set, and take the confusion matrix as the relationships. Mathematically, for any pair of concepts i and j, the relationship R_{ij} is defined as:

$$R_{ij} = P(i|j) = \frac{\sum_{k \in C_i} ConvNet(j|I_k)}{|C_i|}, \tag{5.1}$$

where C_i is the set of indexes for images that belong to concept i, $|\cdot|$ is the cardinality function, and given pixel values I_k, $ConvNet(j|I_k)$ is the network's belief on how likely image k belongs to concept i. We want our graph to be sparse, therefore we just used the top K ($K = 5$ in our experiments) and re-normalized the probability mass.

[3]Flickr images are downloaded using tag search. We use the same query strings as used in Google Image Search.

After constructing the relationship graph, we put this graph (represented as a matrix) on top of the seventh layer of the network, so that now the soft-max loss function becomes:

$$L = \sum_k \sum_i R_{il_k} \log(ConvNet(i|I_k)). \tag{5.2}$$

In this way, the network is trained to predict the context of a category (in terms of relationships to other categories), and the error is back-propagated through the relationship graph to lower layers. Note that, this extra layer is similar to [264], in which R_{ij} is used to characterize the label-flip noise. Different from them, we do not assume all the categories are *mutually exclusive*, but instead *inter related*. For example, `cat` is a hyper-class of `Siamese cat`, and its reasonable if the model believes some examples of `Siamese cat` are more close to the average image of a `cat` than that of the `Siamese cat` and vice versa. Please see experimental section for our empirical validation of this assumption. For fear of semantic drift, in this chapter we keep the initially learned graph structure fixed, but it would be interesting to see how updating the relationship graph performs (like [43]).

5.3.3 Localizing Objects

To show the effectiveness of our representation, after fine-tuning we go back to the problem of organizing the data on the web. That is, clean up the data by removing noise and localizing objects in the images. But shouldn't the ConvNet have learned intrinsically the salient regions in an image for the concepts of interest [22, 209, 251]? Isn't getting clean data as simple as ranking the initial set of images based on the soft-max output? We argue that, while the network has already learned to model the *positive* examples when solving the multi-way classification problem, it has not yet learned the distribution of *negative* data, *e.g.* background clutter. While scenes and attributes are more "stuff-like" and thus finding clean full images might be enough, it is important for objects to be localized well, particularly when they are small in the original image. In fact, since the network is optimized for a classification loss, the representation is learned to be spatially invariant (*e.g.*, the network should output `orange` regardless of where it exists in the image, and how many there are), precisely localizing the object is a very challenging task.

To overcome the difficulty, we developed a subcategory discovery based approach similar to [43] to localize the object given a collection of search engine results. It is based on Google's bias toward images with a single centered object, so we can use such images as seeds to locate similar examples in other images of the collection. Apart from the exemplar based pipeline, there are some significant differences:

- Instead of sliding window based detection framework, we used object proposals from Edge-Box [317], so that for each image, only a few hundred of patches[4] are examined.
- Given the proposals, we compute the seventh layer output ($fc7$) to represent each patch, instead of HOG. The original alignment is lost, but the feature has better generalization power (See qualitative results from Figure 5.4).
- For eLDA [99], we extracted random patches from all the downloaded web data to build the negative correlation matrix.

[4]EdgeBox usually outputs $\sim$ 2000 proposals per image. To further reduce the computation overhead, we only used windows that cover more than 1% of the entire image. We find it only has negligible effect on the final clustering quality, but purged more than 90% of the proposals.

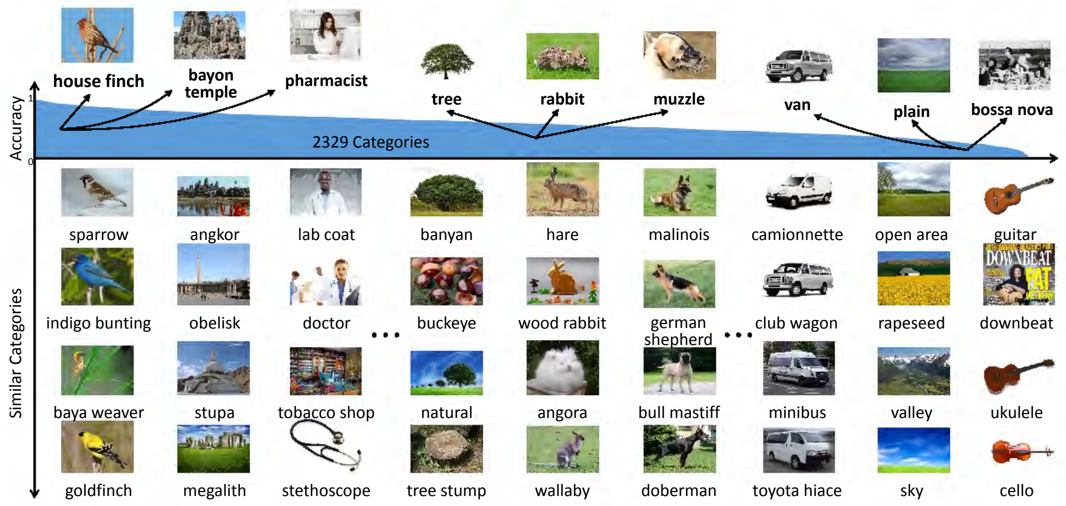


Figure 5.3: Relationships between different categories with the confusion matrix. The horizontal axis is for categories, which are ranked based on ConvNet's accuracy. Here we show random examples from three parts of the distribution: Three top ones, three middle ones, and three bottom ones in terms of accuracy. It can be seen that the relationships are pretty good: For top ones, even though the network can differentiate the categories really well, when it gets confused, it gets confused to similar looking ones. Even for bottom ones when the network gets confused heavily, it is confusing between semantically related categories. Even for very noisy categories like `bossa nova`, the network is able to figure out it is related to musical instruments.

- Affinity propagation [80] is used in [43] for subcategories, whereas we just merged the initial clusters (formed by top detections) from bottom up to get the final subcategories, which works well and takes less time.

Finally after getting the clean examples, we train detectors following the RCNN [87] approach. In the first trial, we simply used the positive examples as-is, and negative patches are randomly sampled from YFCC dataset [271]. Typically, hundreds of positive instances per category are available for training. While this number is comparable to the PASCAL VOC 2007 trainval set (except car, chair and person), one big advantage of Internet is its nearly infinite limit on data. Therefore, we tried two augmentation strategies:

Data augmentation We followed [87] and did data augmentation on the positive training examples. We again used EdgeBox [317] to propose regions of interest on images where the positive example lies in. And whenever a proposal has a higher than 0.5 overlapping (measured by IoU, intersection over union) with any of the positive bounding box, we add it to the pool of our training data.

Category expansion Here we again used the relationship graph to look for synonyms and similar categories in our list of objects for more training examples. After semantic verification, we add the examples into training dataset. We believe adding the examples from these categories should allow better generalization.

Table 5.1: Results on VOC-2007 (PASCAL data used).

VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mAP**
ImageNet-NFT [87]	57.6	57.9	38.5	31.8	23.7	51.2	58.9	51.4	20.0	50.5	40.9	46.0	51.6	55.9	43.3	23.3	48.1	35.3	51.0	57.4	44.7
GoogleO-NFT	57.1	59.9	35.4	30.5	21.9	53.9	59.5	40.7	18.6	43.3	37.5	41.9	49.6	57.7	38.4	22.8	45.2	37.1	48.0	54.5	42.7
GoogleA-NFT	54.9	58.2	35.7	30.7	22.0	54.5	59.9	44.7	19.9	41.0	34.5	40.1	46.8	56.2	40.0	22.2	45.8	36.3	47.5	54.2	42.3
Flickr-NFT	55.3	61.9	39.1	29.5	24.8	55.1	62.7	43.5	22.7	49.3	36.6	42.7	48.9	59.7	41.2	25.4	47.7	41.9	48.8	56.8	44.7
VOC-Scratch [1]	49.9	60.6	24.7	23.7	20.3	52.5	64.8	32.9	20.4	43.5	34.2	29.9	49.0	60.4	47.5	28.0	42.3	28.6	51.2	50.0	40.7
ImageNet-FT [87]	64.2	**69.7**	**50.0**	**41.9**	**32.0**	62.6	71.0	**60.7**	**32.7**	58.5	46.5	**56.1**	60.6	66.8	54.2	31.5	52.8	48.9	57.9	**64.7**	**54.2**
GoogleO-FT	**65.0**	68.1	45.2	37.0	29.6	65.4	73.8	54.0	30.4	57.8	48.7	51.9	**64.1**	64.7	54.0	**32.0**	**54.9**	44.5	57.0	64.0	53.1
GoogleA-NFT	64.2	68.3	42.7	38.7	26.5	65.1	72.4	50.7	28.5	**60.9**	**48.8**	51.2	60.2	65.5	**54.5**	31.1	50.5	**48.5**	56.3	60.3	52.3
Flickr-FT	63.7	68.5	46.2	36.4	30.2	**68.4**	**73.9**	56.9	31.4	59.1	46.7	52.4	61.5	**69.2**	53.6	31.6	53.8	44.5	**58.1**	59.6	53.3

Table 5.2: Results on VOC-2012 (PASCAL data used).

VOC 2012 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mAP**
ImageNet-FT [87]	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	**66.2**	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6	49.6
ImageNet-FT(TV)	**73.3**	67.1	46.3	31.7	30.6	59.4	61.0	**67.9**	27.3	**53.1**	39.1	64.1	**60.5**	70.9	57.2	26.1	**59.0**	40.1	56.2	**54.9**	52.3
GoogleO-FT	72.2	67.3	46.0	**32.3**	**31.6**	**62.6**	62.5	66.5	27.3	52.1	38.9	64.0	59.1	71.6	58.0	27.2	57.6	41.3	56.3	53.7	52.4
Flickr-FT	72.7	**68.2**	**47.3**	32.2	30.6	62.3	**62.6**	65.9	**28.1**	52.2	**39.5**	65.1	60.0	**71.7**	**58.2**	**27.3**	58.0	**41.5**	**57.2**	53.8	**52.7**

5.4 Experimental Results

We now describe our experimental results. Our goal is to demonstrate that the visual representation learned using two-step webly supervised learning is meaningful. For this, we will do four experiments: (1) First, we will show that our learned ConvNet can be used for object detection. Here, we use the approach similar to RCNN [87] where we will fine-tune our learned ConvNet using VOC data. This is followed by learning SVM-detectors using ConvNet features; (2) We will also show that our ConvNet can be used to clean up the web data, *i.e.*, discover subcategories and localize the objects in web images; (3) Third, we will train detectors using the cleaned up web data and evaluate them on VOC data. Note in this case, we will not use any VOC training images. We will only use web images to train both the ConvNet and the subsequent SVMs. (4) Additionally, we will show scene classification results to further showcase the usefulness of the trained representation.

All the networks are trained with the Caffe Toolbox [117]. In total we have 2240 objects, 89 attributes, and 874 scenes. Two networks are trained: (1) The object-attribute network (GoogleO), where the output dimension is 2329; and (2) All included network (GoogleA), where the output dimension is 3203. For the first network, ~ 1.5 million images are downloaded from Google Image Search. Combining scene images, ~ 2.1 million images are used in the second network. The first network is then fine-tuned with ~ 1.2 million Flickr images (Flickr). We set the batch size to be 256 and start with a learning rate of 0.01. The learning rate is reduced by a factor of 10 after every $150K$ iterations, and we stop training at $450K$ iterations. For fine-tuning, we choose a step size of 30K and train the network for a total of $100K$ iterations.

Is Confusion Matrix Informative for Relationships? Before we delve into the results, we want to first show if the following assumption holds: Whether the network has learned to discover the look-alike relationships between concepts in the confusion matrix. To verify the quality of the network, we take the GoogleO net and visualize the top-5 most confusing concepts (including self) to some of the categories. To ensure our selection has a good coverage, we first rank the diagonal of the confusing matrix (accuracy) in the descending order. Then we randomly sample three categories from the top-100, bottom-100, and medium-100 from the list. The visualization can be seen in

Table 5.3: Webly supervised VOC 2007 detection results (No VOC training data used).

VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mAP**
LEVAN [59]	14.0	36.2	12.5	10.3	9.2	35.0	**35.9**	8.4	**10.0**	17.5	6.5	12.9	**30.6**	27.5	6.0	1.5	18.8	10.3	23.5	16.4	17.1
GoogleO	30.2	34.3	16.7	13.3	6.1	43.6	27.4	22.6	6.9	16.4	10.0	21.3	25.0	35.9	7.6	9.3	21.8	17.3	31.0	18.1	20.7
GoogleA	29.5	38.3	15.1	14.0	9.1	44.3	29.3	24.9	6.9	15.8	9.7	22.6	23.5	34.3	9.7	12.7	21.4	15.8	33.4	19.4	21.5
Flickr	32.6	42.8	19.3	13.9	9.2	46.6	29.6	20.6	6.8	17.8	10.2	22.4	26.7	40.8	11.7	**14.0**	19.0	19.0	34.0	21.9	22.9
Flickr-M	**32.7**	**44.3**	17.9	14.0	**9.3**	**47.1**	26.6	19.2	8.2	18.3	10.0	22.7	25.0	42.5	12.0	12.7	**22.2**	20.9	35.6	18.2	23.0
Flickr-C	30.2	41.3	**21.7**	**18.3**	9.2	44.3	32.2	**25.5**	9.8	**21.5**	**10.4**	**26.7**	27.3	**42.8**	**12.6**	13.3	20.4	**20.9**	**36.2**	**22.8**	**24.4**

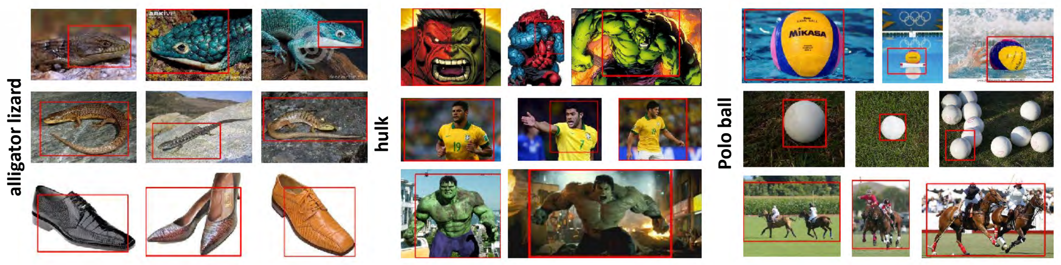


Figure 5.4: We use the learned ConvNet representation to discover bounding boxes for different categories in the training data as well as discover subcategories. Sample results are shown in the figure.

Figure 5.3.

5.4.1 PASCAL VOC Object Detection

Next, we test our webly trained ConvNet model for the task of object detection. We run our experiments on VOC 2007 and VOC 2012 datasets. We follow the RCNN pipeline: Given our trained ConvNet, we first fine-tune the network using trainval images. We then learn a SVM using trainval on fine-tuned $fc7$ features. For VOC 2007, we used a step size of 20K and 100K iterations of fine-tuning. For VOC 2012, since the number of trainval images is doubled, we use 200K iterations of fine-tuning with a step size of 50K. For fair comparison, we didn't tune any parameters in RCNN, so the settings for SVM training are kept identical to those for ImageNet. Since we trained three different networks with different types of training data, we report three different numbers (GoogleO-FT, GoogleA-FT, Flickr-FT). Note that Flickr-FT network corresponds to learning both on Google and Flickr data using two step process and is initialized with GoogleO network.

As baselines we compare against RCNN trained using ConvNet-Scratch features [1] (VOC-Scratch), RCNN trained on ImageNet features without fine-tuning (ImageNet-NFT), RCNN trained on ImageNet features with fine-tuning on VOC trainval (ImageNet-FT) and our webly trained ConvNet without fine-tuning (GoogleO-NFT, GoogleA-NFT and Flickr-NFT). The results on VOC 2007 are indicated in Table 5.1. As the results show, all our networks outperform VOC-Scratch by a huge margin. When it comes to results without fine-tuning on VOC, our Flickr-NFT performs exactly similar to Imagenet-NFT (mAP = 44.7). This clearly indicates that the webly supervised ConvNet learns visual representation comparable to ImageNet pretrained ConvNet. After fine-tuning, our webly supervised ConvNet also performs comparably to Imagenet pretrained ConvNet.

The results on VOC 2012 are reported in Table 5.2. In this case, our two-stage ConvNet with

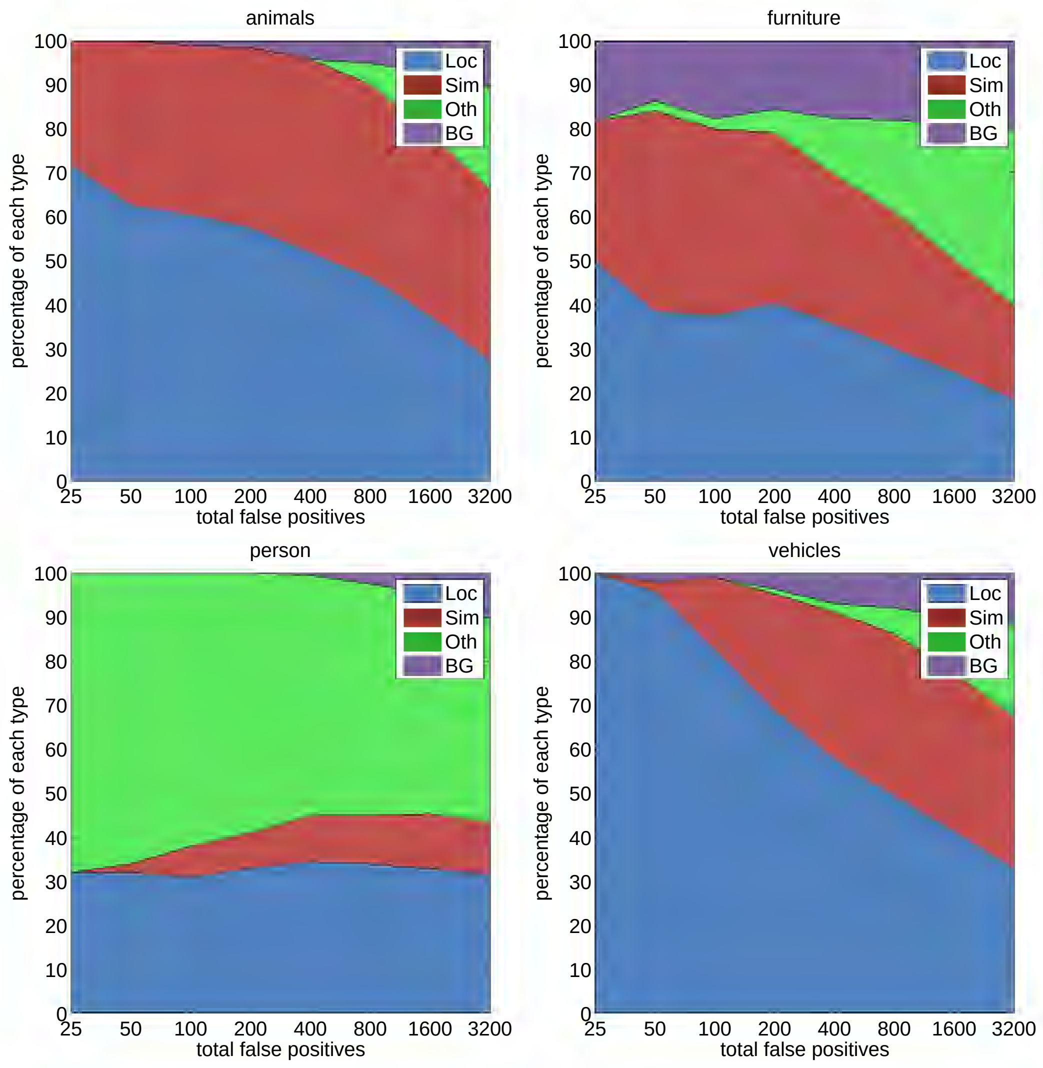


Figure 5.5: Diagnosis analysis using [108] for better understanding of the failure modes of our webly supervised pipeline. Please see top false positives in Figure 5.6 and 5.7.

fine-tuning (Flickr-FT) outperforms the ImageNet pretrained network. Both in case of VOC 2007 and 2012, our webly supervised ConvNet seems to work better for vehicles since we have lots of data for cars and other vehicles (~ 500). On the other hand, ImageNet ConvNet seems to outperform our network on animals such as cat and dog. This is probably because ImageNet has a lot more data for animals. This indicates that the performance of our network might increase further if more query strings for animals are added. Note that the original RCNN paper fine-tuned the ImageNet network using train data alone and therefore reports lower performance. For fair comparison, we fine-tuned both ImageNet network and our webly supervised network on combined trainval images.

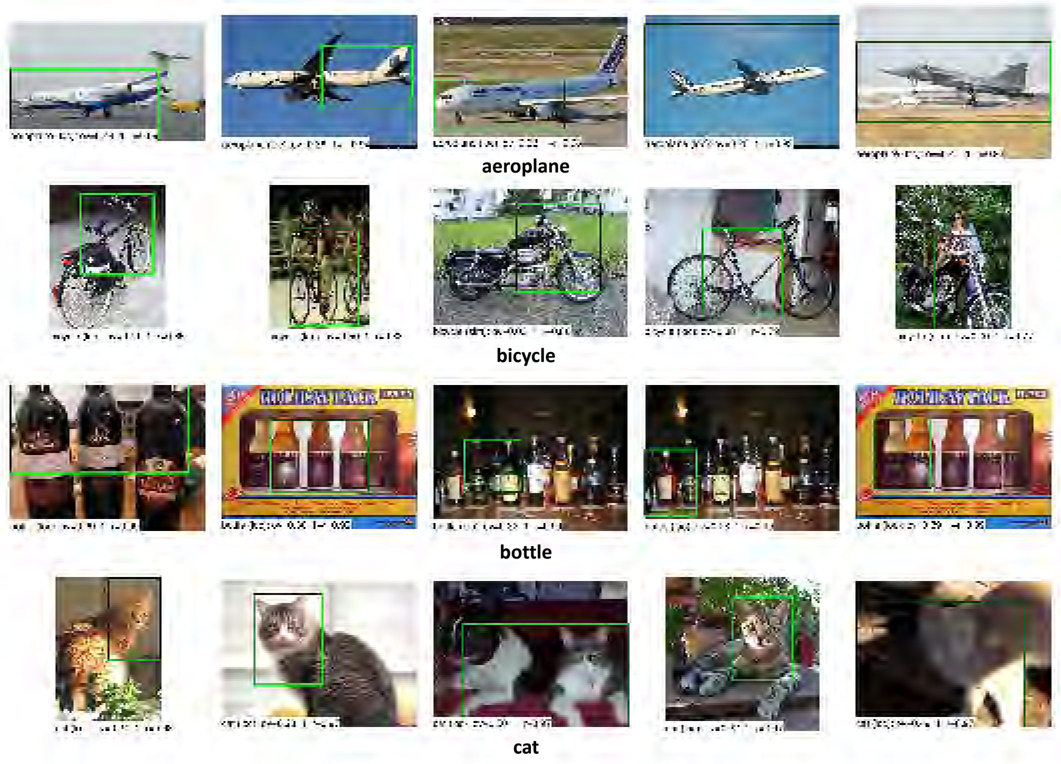


Figure 5.6: Top false positives for selected categories on PASCAL VOC 2007 detection with Flickr-C. From top down: aeroplane, bicycle, bottle, cat.

5.4.2 Object Localization

In this subsection, we are interested to see what is the influence of PASCAL training data for the detection task, since we can localize objects automatically with our proposed approach (see Section 5.3.3). Please refer to Figure 5.4 for the qualitative results on the training localization we can get with $fc7$ features. Compared to [43], the subcategories we get are less homogeneous (*e.g.* people are not well-aligned, objects in different view points are clustered together). But just because of this more powerful representation (and thus better distance metric), we are able to dig out more signal from the training set - since semantically related images can form clusters and won't be purged as noise when an image is evaluated by its nearest neighbors.

Using localized objects, we train RCNN based detectors to detect objects on the PASCAL VOC 2007 test set. We compare our results against [59], who used Google N-grams to expand the categories (*e.g.*`horse` is expanded to `jumping horse`, `racing horse`, *etc.*) and the models were also directly trained from the web. The results are shown in Table 5.3. For our approach, we try five different settings: (a) GoogleO: Features are based on GoogleO ConvNet and the bounding boxes are also extracted only on easy Google Images; (b) GoogleA: Features are based on GoogleA ConvNet and the bounding boxes are extracted on easy images alone; (c) Flickr: Features are based on final two-stage ConvNet and the bounding boxes are extracted on easy images; (d) Flickr-M: Features are based on final two-stage ConvNet and the bounding boxes are extracted on easy and hard images; (e) Flickr-C: Features are based on final two-stage ConvNet and the positive data includes

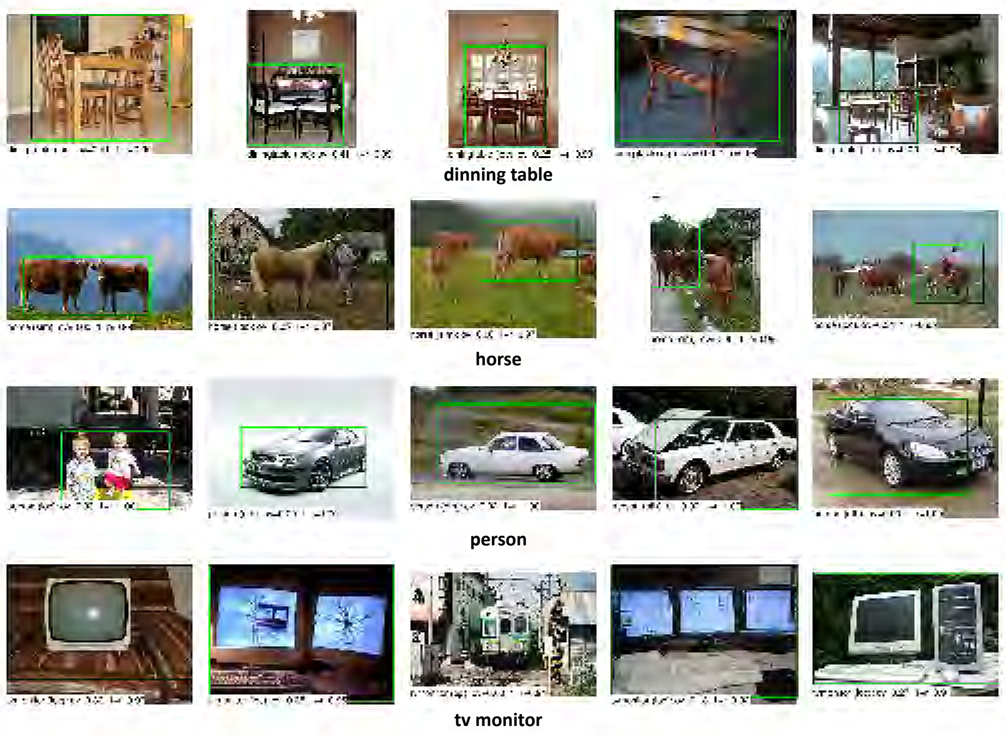


Figure 5.7: Top false positives for selected categories on PASCAL VOC 2007 detection with Flickr-C. From top down: dining table, horse, person, tv monitor.

bounding box of original and related categories. From the results, we can see that the representation based detector training boosts the performance a lot.

This demonstrates that our framework could be a powerful way to learn detectors on the fly without labeling any training images and still yields respectable results. We plan to release this as a service for everyone to train RCNN detectors on the fly.

5.4.3 Failure Modes for Webly Trained Detectors

In this section, we would like to see what are the potential issues of our webly supervised object detection pipeline. We took the results from our best model (Flickr-C) and fed them to the publically available diagnosis tool [108]. Figure 5.5, 5.6 and 5.7 highlight some of the interesting observations we found.

Firstly, localization error accounts for a majority of the false positives. Since Google Image Search do not provide precise location information, the background is inevitably included when the detector is trained (*e.g.* aeroplane, dining table). Multiple instances of an object can also occur in the image, but the algorithm has no clue that they should be treated as separate pieces (*e.g.* bottle). Moreover, since our ConvNet is directly trained on full images, the objective function also biases the representation to be invariant (to spatial locations, *etc*.). All these factors caused serious localization issues.

Second, we did observe some interesting semantic drift between PASCAL categories and Google categories. For example, bicycle can also mean motorcycle on Google. Sense disambiguation for

Table 5.4: Scene Classification Results on MIT Indoor-67 Dataset.

Indoor-67	**Accuracy**
ImageNet [312]	56.8
OverFeat [223]	58.4
GoogleO	58.1
GoogleA	**66.5**
Flickr	59.2

this polysemous word is needed here. Also note that our person detector is severely confused with cars, we suspect it is because `caprice` was added as a related category but it can also mean a car (CHEVY CAPRICE). How to handle such issues is a fascinating future research topic by itself.

5.4.4 Scene Classification

To further demonstrate the usage of ConvNet features directly learned from the web, we also conducted scene classification experiments on the MIT Indoor-67 dataset [218]. For each image, we simply computed the $fc7$ feature vector, which has 4096 dimensions. We did not use any data augmentation or spatial pooling technique, with the only pre-processing step normalizing the feature vector to unit ℓ_2 length [223]. The default SVM parameters (C=1) were fixed throughout the experiments.

Table 5.4 summarizes the results on the default train/test split. We can see our web based ConvNets achieved very competitive performances: All the three networks achieved an accuracy at least on par with ImageNet pretrained models. Fine-tuning on hard images enhanced the features, but adding scene-related categories gave a huge boost to 66.5 (comparable to the ConvNet trained on Places database [312], 68.2). This indicates ConvNet features learned directly from the web are indeed generic.

Moreover, since we can easily get images for semantic labels (*e.g.* actions, N-grams, *etc.*) other than objects or scenes from the web, webly supervised ConvNet bears a great potential to perform well on many relevant tasks - with the cost as low as providing a category list to query for that domain.

Chapter 6

Image to Caption and Back

6.1 Introduction

Training a ConvNet from the web directly is indeed possible, as the previous chapter not only obtained a generalizale representation, but also showed through the confusion matrix that such a representation can at least encode pair-wise similarity relationships, and use them to enhance performance when transferred to other relevant tasks. However, the end-task it targets at is still simple "classification". Even for object detection, the task is casted as a region classification problem [87] where an image patch is singled out from the rest, and a classifier is applied with the addition bit for background class. In this chapter, we investigate an interesting application for learned implicit visual knowledge – caption generation. Arguably, caption is more structured due to the inherent syntax, or statistically correlation between words. If the implicit knowledge can indeed store structure in it, it should also help such tasks as well. Note that, in this chapter we do not directly use representations trained with web images directly, but instead using an existing clean knowledge base – ImageNet, since our point to make is not "whether we can obtain implicit commonsense from the web images directly", but rather "whether such implicit knowledge representation can help more complex tasks". Here we begin with our motivation.

A good image description is often said to "paint a picture in your mind's eye." The creation of a mental image may play a significant role in sentence comprehension in humans [123]. In fact, it is often this mental image that is remembered long after the exact sentence is forgotten [164, 204]. As an illustrative example, Figure 6.1 shows how a mental image may vary and increase in richness as a description is read. Could computer vision algorithms that comprehend and generate image captions take advantage of similar evolving visual representations?

Several papers have explored learning joint feature spaces for images and their descriptions [107, 124, 259]. These approaches project image features and sentence features into a common space, which may be used for image search or for ranking image captions. Various approaches were used to learn the projection, including Kernel canonical correlation analysis (KCCA) [107], recursive neural networks (RNN) [259], or deep neural networks [124]. While these approaches project both semantics and visual features to a common embedding, they are not able to perform the inverse projection. That is, they cannot generate novel sentences or visual depictions from the embedding.

In this chapter, we propose a bi-directional representation capable of generating both novel descriptions from images and visual representations from descriptions. Critical to both of these tasks is a novel representation that dynamically captures the visual aspects of the scene that have already

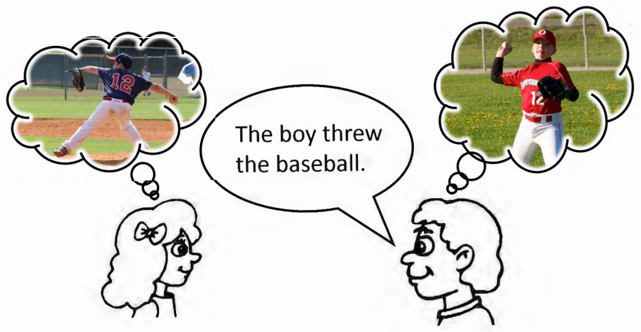


Figure 6.1: Internal visual representations are important for both generating and understanding semantic descriptions of scenes. While visual representations may vary due to a description's ambiguity, a good description conveys the salient aspects of the scene.

been described. That is, as a word is generated or read the visual representation is updated to reflect the new information contained in the word. We accomplish this using RNNs [66, 187, 191]. One long-standing problem of RNNs is their weakness in remembering concepts after a few iterations of recurrence. For instance RNN language models often find difficultly in learning long distance relations [19, 187] without specialized gating units [106]. During sentence generation, our novel dynamically updated visual representation acts as a long-term memory of the concepts that have already been mentioned. This allows the network to automatically pick salient concepts to convey that have yet to be spoken. As we demonstrate, the same representation may be used to create a visual representation of a written description.

We demonstrate our method on numerous datasets. These include the PASCAL sentence dataset [221], Flickr 8K [221], Flickr 30K [221], and the Microsoft COCO dataset [37, 166] containing over 400,000 sentences. When generating novel image descriptions, we demonstrate results as measured by BLEU [210], METEOR [8] and CIDEr [279]. Qualitative results are shown for the generation of novel image captions. We also evaluate the bi-directional ability of our algorithm on both the image and sentence retrieval tasks. Since this does not require the ability to generate novel sentences, numerous previous papers have evaluated on this task. We show results that are better or comparable to previous state-of-the-art results using similar visual features.

6.2 Related Work

The task of building a visual memory lies at the heart of two long-standing AI-hard problems: grounding natural language symbols to the physical world and semantically understanding the content of an image. Whereas learning the mapping between image patches and single text labels remains a popular topic in computer vision [81, 87, 136], there is a growing interest in using entire sentence descriptions together with pixels to learn joint embeddings [90, 107, 124, 259]. Viewing corresponding text and images as correlated, KCCA [107] is a natural option to discover the shared features spaces. However, given the highly non-linear mapping between the two, finding a generic distance metric based on shallow representations can be extremely difficult. Recent papers seek better objective functions that directly optimize the ranking [107], or directly adopts pre-trained representations [259] to simplify the learning, or a combination of the two [90, 124].

With a good distance metric, it is possible to perform tasks like bi-directional image-sentence

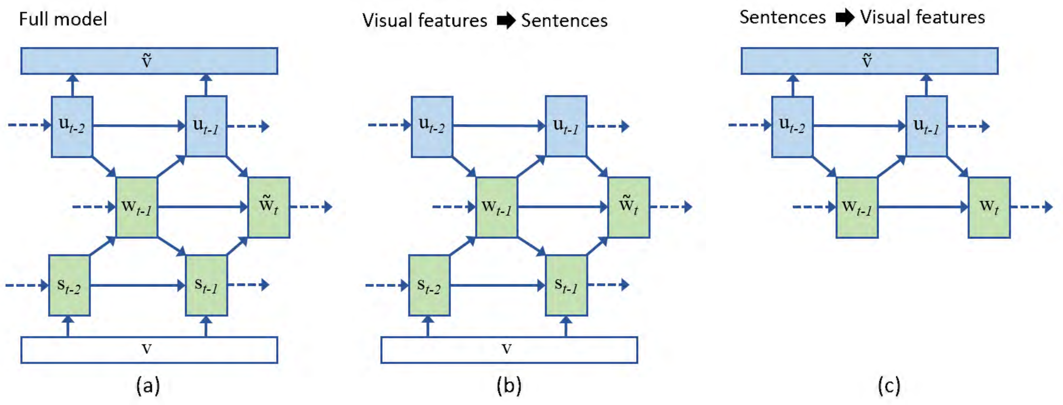


Figure 6.2: Illustration of our model. (a) shows the full model used for training. (b) and (c) show the parts of the model needed for generating sentences from visual features and generating visual features from sentences respectively.

retrieval. However, in many scenarios it is also desired to generate novel image descriptions and to hallucinate a scene given a sentence description. Numerous papers have explored the area of generating novel image descriptions [72, 130, 138, 142, 193, 304, 307]. These papers use various approaches to generate text, such as using pre-trained object detectors with template-based sentence generation [72, 138, 304]. Retrieved sentences may be combined to form novel descriptions [142]. Recently, purely statistical models have been used to generate sentences based on sampling [130] or recurrent neural networks [181]. While [181] also uses a RNN, their model is significantly different from our model. Specifically their RNN does not attempt to reconstruct the visual features, and is more similar to the contextual RNN of [191]. For the synthesizing of images from sentences, the recent paper by Zitnick *et al.* [318] uses abstract clip art images to learn the visual interpretation of sentences. Relation tuples are extracted from the sentences and a conditional random field is used to model the visual scene.

There are numerous papers using recurrent neural networks for language modeling [18, 130, 187, 191]. We build most directly on top of [18, 187, 191] that use RNNs to learn word context. Several models use other sources of contextual information to help inform the language model [130, 191]. Despite its success, RNNs still have difficulty capturing long-range relationships in sequential modeling [19]. One solution is Long Short-Term Memory (LSTM) networks [106, 130, 267], which use “gates” to control gradient back-propagation explicitly and allow for the learning of long-term interactions. However, the main focus of this chapter is to show that the hidden layers learned by “translating” between multiple modalities can already discover rich structures in the data and learn long distance relations in an automatic, data-driven manner.

There are several contemporaneous papers [62, 70, 124, 131, 285] that explore the generation of novel image captions using LSTMs [62, 131, 285], RNNs [124] and traditional maximum entropy language models [70]. Unlike these models, our model dynamically builds a visual representation of the scene as a caption is being generated. As we demonstrate, this can lead to improved results.

6.3 Approach

In this section we describe our approach using recurrent neural networks. Our goals are twofold. First, we want to be able to generate sentences given a set of visual observations or features. Specifically, we want to compute the probability of a word w_t being generated at time t given the set of previously generated words $W_{t-1} = w_1, \dots, w_{t-1}$ and the observed visual features V. Second, we want to enable the capability of computing the likelihood of the visual features V given a set of spoken or read words W_t for generating visual representations of the scene or for performing image search. To accomplish both of these tasks we introduce a set of latent variables U_{t-1} that encodes the visual interpretation of the previously generated or read words W_{t-1}. As we demonstrate later, the latent variables U play the critical role of acting as a long-term visual memory of the words that have been previously generated or read.

Using U, our goal is to compute $P(w_t|V, W_{t-1}, U_{t-1})$ and $P(V|W_{t-1}, U_{t-1})$. Combining these two likelihoods together our global objective is to maximize,

$$\begin{aligned} &P(w_t, V|W_{t-1}, U_{t-1}) \\ &\quad = P(w_t|V, W_{t-1}, U_{t-1})P(V|W_{t-1}, U_{t-1}). \end{aligned} \tag{6.1}$$

That is, we want to maximize the likelihood of the word w_t and the observed visual features V given the previous words and their visual interpretation. Note that in previous papers [181, 191] the objective was only to compute $P(w_t|V, W_{t-1})$ and not $P(V|W_{t-1})$.

6.3.1 Model structure

Our recurrent neural network model structure builds on the prior models proposed by [187, 191]. Mikolov [187] proposed a RNN language model shown by the green boxes in Figure 6.2(a). The word at time t is represented by a vector $\mathbf{w}_t$ using a "one hot" representation. That is, $\mathbf{w}_t$ is the same size as the word vocabulary with each entry having a value of 0 or 1 depending on whether the word was used. The output $\tilde{\mathbf{w}}_t$ contains the likelihood of generating each word. The recurrent hidden state $\mathbf{s}$ provides context based on the previous words. However, $\mathbf{s}$ typically only models short-range interactions due to the problem of vanishing gradients [19, 187]. This simple, yet effective language model was shown to provide a useful continuous word embedding for a variety of applications [188].

Following [187], Mikolov *et al.* [191] added an input layer $\mathbf{v}$ to the RNN shown by the white box in Figure 6.2. This layer may represent a variety of information, such as topic models or parts of speech [191]. In our application, $\mathbf{v}$ represents the set of observed visual features. We assume the visual features $\mathbf{v}$ are constant. These visual features help inform the selection of words. For instance, if a cat was detected, the word "cat" is more likely to be spoken. Note that unlike [191], it is not necessary to directly connect $\mathbf{v}$ to $\tilde{\mathbf{w}}$, since $\mathbf{v}$ is static for our application. In [191] $\mathbf{v}$ represented dynamic information such as parts of speech for which $\tilde{\mathbf{w}}$ needed direct access. We also found that only connecting $\mathbf{v}$ to half of the $\mathbf{s}$ units provided better results, since it allowed different units to specialize on modeling either text or visual features.

The main contribution of this chapter is the addition of the recurrent visual hidden layer $\mathbf{u}$, blue boxes in Figure 6.2(a). The recurrent layer $\mathbf{u}$ attempts to reconstruct the visual features $\mathbf{v}$ from the previous words, *i.e.* $\tilde{\mathbf{v}} \approx \mathbf{v}$. The visual hidden layer is also used by $\tilde{\mathbf{w}}_t$ to help in predicting the next word. That is, the network can compare its visual memory $\mathbf{u}$, which represents what it already said, with what it currently observes $\mathbf{v}$ to predict what to say next. At the beginning of the sentence, $\mathbf{u}$ represents the prior probability of the visual features. As more words are observed, the visual features are updated to reflect the words' visual interpretation. For instance, if the word "sink" is

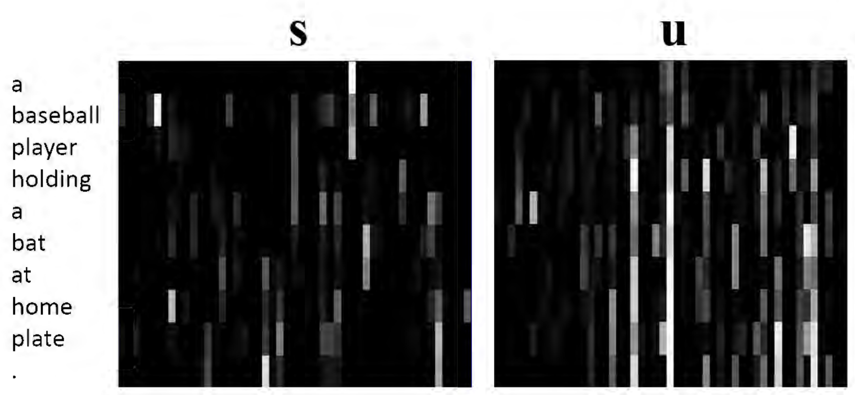


Figure 6.3: Illustration of the hidden units $\mathbf{s}$ and $\mathbf{u}$ activations through time (vertical axis). Notice that the visual hidden units $\mathbf{u}$ exhibit long-term memory through the temporal stability of some units, where the hidden units $\mathbf{s}$ change significantly each time step.

generated, the visual feature corresponding to sink should increase. Other features that correspond to stove or refrigerator might increase as well, since they are highly correlated with sink.

A critical property of the recurrent visual features $\mathbf{u}$ is their ability to remember visual concepts over the long term. The property arises from the model structure. Intuitively, one may assume the visual features shouldn't be estimated until the sentence is finished. That is, $\mathbf{u}$ should not be used to estimate $\mathbf{v}$ until $\mathbf{w}_t$ generates the end of sentence token. However, in our model we force $\mathbf{u}$ to estimate $\mathbf{v}$ at every time step. This helps to learn long-term visual concepts. For instance, if the word "cat" is generated, $\mathbf{u}_t$ will increase the likelihood of the visual feature corresponding to cat. Assuming the "cat" visual feature in $\mathbf{v}$ is active, the network will receive positive reinforcement to propagate $\mathbf{u}$'s memory of "cat" from one time instance to the next. Figure 6.3 shows an illustrative example of the hidden units $\mathbf{s}$ and $\mathbf{u}$. As can be observed, some visual hidden units $\mathbf{u}$ exhibit longer temporal stability.

Note that the same network structure can predict visual features from sentences or generate sentences from visual features. For generating sentences (Fig. 6.2(b)), $\mathbf{v}$ is known and $\tilde{\mathbf{v}}$ may be ignored. For predicting visual features from sentences (Fig. 6.2(c)), $\mathbf{w}$ is known, and $\mathbf{s}$ and $\mathbf{v}$ may be ignored. This property arises from the fact that the word units $\mathbf{w}$ separate the model into two halves for predicting words or visual features respectively. Alternatively, if the hidden units $\mathbf{s}$ were connected directly to $\mathbf{u}$, this property would be lost and the network would act as a normal auto-encoder [284].

6.3.2 Language Model

Our language model typically has between 3,000 and 20,000 words. While each word may be predicted independently, this approach is computationally expensive. Instead, we adopted the idea of word classing [187] and factorized the distribution into a product of two terms:

$$P(w_t|\cdot) = P(c_t|\cdot) \times P(w_t|c_t, \cdot). \tag{6.2}$$

$P(w_t|\cdot)$ is the probability of the word, $P(c_t|\cdot)$ is the probability of the class. The class label of the word is computed in an unsupervised manner, grouping words of similar frequencies together. Generally, this approach greatly accelerates the learning process, with little loss of perplexity. The

	PASCAL		
	PPL	BLEU	METEOR
Midge [193]	-	2.9	8.8
Baby Talk [138]	-	0.5	9.7
Our Approach	25.3	9.8	16.0
Our Approach+FT	24.6	10.4	16.3
Our Approach+VGG	23.8	12.0	17.6
Human	-	20.1	25.0

Table 6.1: Results for novel sentence generation for PASCAL 1K. Results are measured using perplexity (PPL), BLEU (%) [210] and METEOR (METR, %) [8]. Results for Midge [193] and BabyTalk [138] are provided. Human agreement scores are shown in the last row. See the text for more details.

predicted word likelihoods are computed using the standard soft-max function. After each epoch, the perplexity is evaluated on a separate validation set and the learning reduced (cut in half in our experiments) if perplexity does not decrease.

In order to further reduce the perplexity, we combine the RNN model's output with the output from a Maximum Entropy language model [189], simultaneously learned from the training corpus. For all experiments we fix how many words to look back when predicting the next word used by the Maximum Entropy model to three.

For any natural language processing task, pre-processing is crucial to the final performance. For all the sentences, we did the following three steps before feeding them into the RNN model. 1) Use Stanford CoreNLP Tool to tokenize the sentences. 2) Lower case all the letters. 3) Replace words that occur less than 5 times with the word out-of-vocabulary (OOV).

6.3.3 Learning

For learning we use the Backpropagation Through Time (BPTT) algorithm [294]. Specifically, the network is unrolled for several words and standard backpropagation is applied. Note that we reset the model after an End-of-Sentence (EOS) is encountered, so that prediction does not cross sentence boundaries. As shown to be beneficial in [187], we use online learning for the weights from the recurrent units to the output words. The weights for the rest of the network use a once per sentence batch update. The activations for all units are computed using the sigmoid function $\sigma(z) = 1/(1 + \exp(-z))$ with clipping, except the word predictions that use soft-max. We found that Rectified Linear Units (ReLUs) [136] with unbounded activations were numerically unstable and commonly "blew up" when used in recurrent networks.

We used the open source RNN code of [187] and the Caffe framework [117] to implement our model. A big advantage of combining the two is that we can jointly learn the word and image representations: the error from predicting the words can be directly backpropagated to the image-level features. However, deep convolution neural networks require large amounts of data to train on, but the largest sentence-image dataset has only 80K images [166]. Therefore, instead of training from scratch, we choose to fine-tune from the pre-trained 1000-class ImageNet models [233] to avoid potential over-fitting. In all experiments, we use the BVLC reference Net [117] or the Oxford VGG-Net [252].

	Flickr 8K			Flickr 30K			MS COCO Val			MS COCO Test		
	PPL	BLEU	METEOR	PPL	BLEU	METEOR	PPL	BLEU	METEOR	BLEU	METEOR	CIDEr
RNN	17.5	4.5	10.3	23.0	6.3	10.7	16.9	4.7	9.8	-	-	-
RNN+IF	16.5	11.9	16.2	20.8	11.3	14.3	13.3	16.3	17.7	-	-	-
RNN+IF+FT	16.0	12.0	16.3	20.5	11.6	14.6	12.9	17.0	18.0	-	-	-
RNN+VGG	15.2	12.4	16.7	20.0	11.9	15.0	12.6	18.4	19.3	18.0	19.1	51.5
Our Approach	16.1	12.2	16.6	20.0	11.3	14.6	12.6	16.3	17.8	-	-	-
Our Approach+FT	15.8	12.4	16.7	19.5	11.6	14.7	12.0	16.8	18.1	16.5	18.0	44.8
Our Approach+VGG	15.1	13.1	16.9	19.1	12.0	15.2	11.6	18.8	19.6	18.4	19.5	53.1
Human	-	20.6	25.5	-	18.9	22.9	-	19.2	24.1	21.7	25.2	85.4

Table 6.2: Results for novel sentence generation for Flickr 8K, FLickr 30K, MS COCO Validation and MS COCO Test. Results are measured using perplexity (PPL), BLEU (%) [210], METEOR (%) [8] and CIDEr-D (%) [279]. Human agreement scores are shown in the last row. See the text for more details.

6.4 Experimental Results

In this section we evaluate the effectiveness of our bi-directional RNN model on multiple tasks. We begin by describing the datasets used for training and testing, followed by our baselines. Our first set of evaluations measure our model's ability to generate novel descriptions of images. Since our model is bi-directional, we evaluate its performance on both the sentence retrieval and image retrieval tasks.

6.4.1 Datasets

For evaluation we perform experiments on several standard datasets that are used for sentence generation and the sentence-image retrieval task:

PASCAL 1K [221] The dataset contains a subset of images from the PASCAL VOC challenge. For each of the 20 categories, it has a random sample of 50 images with 5 descriptions provided by Amazon's Mechanical Turk (AMT).

Flickr 8K and 30K [221] These datasets consists of 8,000 and 31,783 images collected from Flickr respectively. Most of the images depict humans participating in various activities. Each image is also paired with 5 sentences. These datasets have a standard training, validation, and testing splits.

MS COCO [37, 166] The Microsoft COCO dataset contains 82,783 training images and 40,504 validation images, each with $\sim$5 human generated descriptions. The images are collected from Flickr by searching for common object categories, and typically contain multiple objects with significant contextual information. We used the training set and validation set to train our model in our experiments, and uploaded our generated captions on the testing set (40,775 images) to the COCO server [37] for evaluation. Results using 5 reference captions are reported.

6.4.2 RNN Baselines

To gain insight into the various components of our model, we compared our final model with three RNN baselines. For fair comparison, the random seed initialization was fixed for all experiments. The the hidden layers $\mathbf{s}$ and $\mathbf{u}$ sizes are fixed to 100. We tried increasing the number of hidden units, but results did not improve. For small datasets, more units can lead to overfitting.

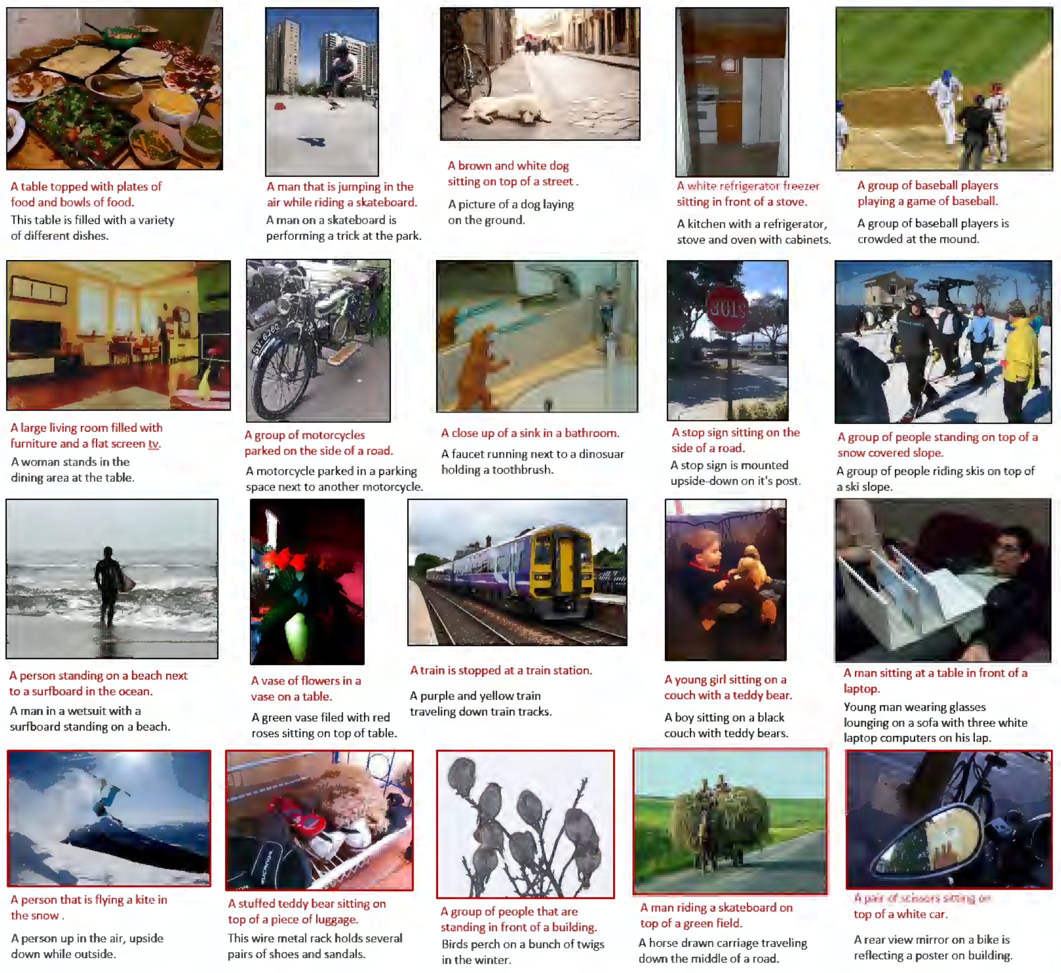

A table topped with plates of food and bowls of food.
This table is filled with a variety of different dishes.
A man that is jumping in the air while riding a skateboard.
A man on a skateboard is performing a trick at the park.
A brown and white dog sitting on top of a street .
A picture of a dog laying on the ground.
A white refrigerator freezer sitting in front of a stove.
A kitchen with a refrigerator, stove and oven with cabinets.
A group of baseball players playing a game of baseball.
A group of baseball players is crowded at the mound.
A large living room filled with furniture and a flat screen tv.
A woman stands in the dining area at the table.
A group of motorcycles parked on the side of a road.
A motorcycle parked in a parking space next to another motorcycle.
A close up of a sink in a bathroom.
A faucet running next to a dinosuar holding a toothbrush.
A stop sign sitting on the side of a road.
A stop sign is mounted upside-down on it's post.
A group of people standing on top of a snow covered slope.
A group of people riding skis on top of a ski slope.
A person standing on a beach next to a surfboard in the ocean.
A man in a wetsuit with a surfboard standing on a beach.
A vase of flowers in a vase on a table.
A green vase filed with red roses sitting on top of table.
A train is stopped at a train station.
A purple and yellow train traveling down train tracks.
A young girl sitting on a couch with a teddy bear.
A boy sitting on a black couch with teddy bears.
A man sitting at a table in front of a laptop.
Young man wearing glasses lounging on a sofa with three white laptop computers on his lap.
A person that is flying a kite in the snow .
A person up in the air, upside down while outside.
A stuffed teddy bear sitting on top of a piece of luggage.
This wire metal rack holds several pairs of shoes and sandals.
A group of people that are standing in front of a building.
Birds perch on a bunch of twigs in the winter.
A man riding a skateboard on top of a green field.
A horse drawn carriage traveling down the middle of a road.
A pair of scissors sitting on top of a white car.
A rear view mirror on a bike is reflecting a poster on building.

Figure 6.4: Qualitative results for sentence generation on the MS COCO dataset. Both a generated sentence (red) using (Our Approach + FT) and a human generated caption (black) are shown. The last row shows several representative failure cases.

RNN based Language Model (RNN) This is the basic RNN language model developed by [187], which has no input visual features.

RNN with Image Features (RNN+IF) This is an RNN model with image features feeding into the hidden layer inspired by [191]. As described in Section 6.3, $\mathbf{v}$ is only connected to $\mathbf{s}$ and not $\tilde{\mathbf{w}}$. For the visual features $\mathbf{v}$ we used the 4096D 7th layer output of the BVLC reference Net [117] after ReLUs. Following [136], we average the five representations computed from cropping the 4 corners and center. This network is pretrained on the ImageNet 1000-way classification task [233]. We experimented with other layers (5th and 6th) but they do not perform as well.

RNN with Image Features Fine-Tuned (RNN+FT) This model has the same architecture as RNN+IF, but the error is back-propagated to the Convolution Neural Network [87]. The CNN is initialized with the weights from the BVLC reference net. The RNN is initialized with the the pre-trained RNN language model. That is, the only randomly initialized weights are the ones from visual features $\mathbf{v}$ to hidden layers $\mathbf{s}$. If the RNN is not pre-trained we found the initial gradients to be too noisy for the CNN. If the weights from $\mathbf{v}$ to hidden layers $\mathbf{s}$ are also pre-trained the search space becomes too limited.

Our implementation takes $\sim$5 seconds to learn a mini-batch of size 128 on a Tesla K40 GPU. It is also crucial to keep track of the validation error and avoid overfitting. We observed this fine-tuning strategy is particularly helpful for MS COCO, but does not give much performance gain on Flickr Datasets before it overfits. The Flickr datasets may not provide enough training data to avoid overfitting.

RNN with Oxford VGG-Net Features (RNN+VGG) In place of the BVLC reference Net features, we have also experimented with Oxford VGG-Net [252] features. Many recent papers [124, 181] have reported better performance with this representation. We again used the last-but-one layer after ReLU to feed into the RNN model.

6.4.3 Sentence generation

Our first set of experiments evaluate our model's ability to generate novel sentence descriptions of images. We experiment on all the image-sentence datasets described previously and compare to the RNN baselines and other previous papers [138, 193]. Since PASCAL 1K has a limited amount of training data, we report results trained on MS COCO and tested on PASCAL 1K. We use the standard train-test splits for the Flickr 8K and 30K datasets. For MS COCO Validation we train and validate on the training set ($\sim$37K/$\sim$3K) to compare variants of our approach. Finally, we report results on the MS COCO Test set using the MS COCO evaluation server [37]. To generate a sentence, we first sample a target sentence length from the multinomial distribution of lengths learned from the training data, then for this fixed length we sample 100 random sentences, and use the one with the lowest loss (negative likelihood, and in case of our model, also reconstruction error) as output.

We choose four automatic metrics for evaluating the quality of the generated sentences, perplexity, BLEU [210], METEOR [8] and CIDEr [279] using the COCO caption evaluation tool [37]. Perplexity measures the likelihood of generating the testing sentence based on the number of bits it would take to encode it. The lower the value the better. BLEU and METEOR were originally designed for automatic machine translation where they rate the quality of a translated sentences given several references sentences. We can treat the sentence generation task as the "translation" of images to sentences. For BLEU, we took the geometric mean of the scores from 1-gram to 4-gram, and used the ground truth length closest to the generated sentence to penalize brevity. For METEOR, we used the latest version. CIDEr [279] is a metric developed specifically for evaluating image captions. We use the variant of CIDEr called CIDEr-D. For BLEU, METEOR and CIDEr higher scores are better. For reference, we also report the consistency between human annotators (using 1 sentence as query and the rest as references for all but MS COCO Test)[1].

Results for PASCAL 1K are shown in Table 6.1. Our approach significantly improves over both Midge [193] and BabyTalk [138] as measured by BLEU and METEOR. Our approach generally provides more naturally descriptive sentences, such as mentioning an image is black and white, or a bus is a "double decker". Midge's descriptions are often shorter with less detail and BabyTalk provides long, but often redundant descriptions. Results on Flickr 8K and Flickr 30K are also provided in Table 6.2.

[1]We used 5 sentences as references for system evaluation, but leave out 4 sentences for human consistency. It is a bit unfair but the difference is usually 1%$\sim$ 2%.

Table 6.3: Flickr 8K Retrieval Experiments. The protocols of [259], [107] and [181] are used respectively in each row. See text for details.

	Sentence Retrieval				Image Retrieval			
	R@1	R@5	R@10	Med r	R@1	R@5	R@10	Med r
Random Ranking	0.1	0.6	1.1	631	0.1	0.5	1.0	500
SDT-RNN [259]	4.5	18.0	28.6	32	6.1	18.5	29.0	29
DeViSE [81]	4.8	16.5	27.3	28	5.9	20.1	29.6	29
DeepFE [124]	12.6	32.9	44.0	14	9.7	29.6	42.5	15
DeepFE+DECAF [124]	5.9	19.2	27.3	34	5.2	17.6	26.5	32
RNN+VGG	8.9	25.7	38.7	20.5	6.5	17.3	28.4	25
Our Approach (T)	9.6	29.1	41.6	17	7.0	23.6	33.6	23
Our Approach (T+I)	9.9	29.2	42.4	16	7.3	24.6	36.0	20
[107]	8.3	21.6	30.3	34	7.6	20.7	30.1	38
RNN+VGG	7.7	23.0	37.2	21	6.8	24.0	33.9	23.5
Our Approach (T)	8.1	24.4	39.1	19	7.4	25.0	37.5	21
Our Approach (T+I)	8.6	25.9	40.1	17	7.6	24.9	37.8	20
M-RNN [181]	14.5	37.2	48.5	11	11.5	31.0	42.4	15
RNN+VGG	14.4	37.9	48.2	10	15.6	38.4	50.6	10
Our Approach (T)	15.2	39.8	49.3	8.5	16.4	40.9	54.8	9
Our Approach (T+I)	15.4	40.6	50.1	8	17.3	42.5	57.4	7

On the MS COCO dataset that contains more images of high complexity we provide BLEU, METEOR and CIDEr scores. Surprisingly our BLEU and METEOR scores (18.5 & 19.4) are just slightly lower than the human score (21.7 & 25.2). Our CIDEr results (52.1) are significantly lower than humans (85.4). The use of image features (RNN + IF) significantly improves performance over using just an RNN language model. Fine-tuning (FT) and our full approach provide additional improvements for all datasets. Results using the VGG-NET [252] (Our approach + VGG) show some improvement. However, we believe with fine-tuning even better results may be achieved. Qualitative results for the MS COCO dataset are shown in Figure 6.4.

It is known that automatic measures are only roughly correlated with human judgment [65, 107, 279], so it is also important to evaluate the generated sentences using human studies. We evaluated 1000 generated sentences on MS COCO Validation by asking human subjects to judge whether it had better, worse or same quality to a human generated ground truth caption. 5 subjects were asked to rate each image, and the majority vote was recorded. In the case of a tie (2-2-1) the two winners each got half of a vote. We find 5.1% of our captions (Our Approach + VGG) are preferred to human captions, and 15.9% of the captions were judged as being of equal quality to human captions. This is an impressive result given we only used image-level visual features for the complex images in MS COCO.

6.4.4 Bi-Directional Retrieval

Our RNN model is bi-directional. That is, it can generate image features from sentences and sentences from image features. To evaluate its ability to do both, we measure its performance on two retrieval tasks. We retrieve images given a sentence description, and we retrieve a description given an image. Since most previous methods are capable of only the retrieval task, this also helps provide experimental comparison.

Table 6.4: Flickr 30K Retrieval Experiments. The protocols of [81] and [181] are used respectively in each row. See text for details.

	Sentence Retrieval				Image Retrieval			
	R@1	R@5	R@10	Med r	R@1	R@5	R@10	Med r
Random Ranking	0.1	0.6	1.1	631	0.1	0.5	1.0	500
DeViSE [81]	4.5	18.1	29.2	26	6.7	21.9	32.7	25
DeepFE+FT [124]	16.4	40.2	54.7	8	10.3	31.4	44.5	13
RNN+VGG	10.2	26.9	36.7	22	7.6	21.3	31.4	27
Our Approach (T)	11.3	30.1	43.2	16	8.2	24.7	37.0	22
Our Approach (T+I)	11.9	32.9	45.1	14	8.4	25.7	36.8	21
M-RNN [181]	18.4	40.2	50.9	10	12.6	31.2	41.5	16
RNN+VGG	14.9	36.7	52.1	11	15.1	41.1	54.1	9
Our Approach (T)	15.8	42.0	57.4	9	17.7	44.9	57.2	7.5
Our Approach (T+I)	16.6	42.5	58.9	8	18.5	45.7	58.1	7

Following other methods, we adopted two protocols for using multiple image descriptions. The first one is to treat each of the ~5 sentences individually. In this scenario, the rank of the retrieved ground truth sentences are used for evaluation. In the second case, we treat all the sentences as a single annotation, and concatenate them together for retrieval.

For each retrieval task we have two methods for ranking. First, we may rank based on the likelihood of the sentence given the image (T). Since shorter sentences naturally have higher probability of being generated, we followed [181] and normalized the probability by dividing it with the total probability summed over the entire retrieval set. Second, we could rank based on the reconstruction error between the image's visual features $\mathbf{v}$ and their reconstructed visual features $\tilde{\mathbf{v}}$ (I). Due to better performance, we use the average reconstruction error over all time steps rather than just the error at the end of the sentence. In Tables 6.3, we report retrieval results on using the text likelihood term only (I) and its combination with the visual feature reconstruction error (T+I). All results use the visual features generated using the VGG-NET [252].

The same evaluation metrics were adopted from previous papers for both the tasks of sentence retrieval and image retrieval. They used R@K (K = 1, 5, 10) as the measurements, which are the recall rates of the (first) ground truth sentences (sentence retrieval task) or images (image retrieval task). Higher R@K corresponds to better retrieval performance. We also report the median/mean rank of the (first) retrieved ground truth sentences or images (Med/Mean r). Lower Med/Mean r implies better performance. For Flickr 8K and 30K several different evaluation methodologies have been proposed. We report three scores for Flickr 8K corresponding to the methodologies proposed by [259], [107] and [181] respectively, and for Flickr 30K [81] and [181].

As shown in Tables 6.3 and 6.4, for Flickr 8K and 30K our approach achieves comparable or better results than all methods except for the recently proposed DeepFE [124]. However, DeepFE uses a different set of features based on smaller image regions. If the similar features are used (DeepFE+DECAF) as our approach, we achieve better results. We believe these contributions are complementary, and by using better features our approach may also show further improvement. In general ranking based on text and visual features (T + I) outperforms just using text (T).

Part III

Reasoning with Visual Knowledge

Chapter 7

Spatial Memory Network

7.1 Introduction

Now we begin the final segment of our jouney on visual knowledge – how to use it. Since visual knowledge is mostly concerned with relationships between scenes, objects, attributes and parts, we believe its best usage is on connecting things together – commonly referred as the "context resoning" problem in computer vision. In this chapter, we present our first building block towards a holistic model of context – Spatial Memory Network (SMN).

Context is long believed to help image understanding. Apart from strong psychological evidence [10, 110, 200, 206] that context is vital for humans to recognize objects, many empirical studies in the computer vision community [30, 35, 60, 82, 83, 133, 183, 194, 245, 274, 277] have also suggested that recognition algorithms can be improved by proper modeling of context.

But what is the right model for context? Consider the problem of object detection. There are two common models of context often used in the community. The first type of model incorporates image or scene level context [14, 109, 153, 177, 195, 247, 272]. The second type models object-object relationships at instance-level [55, 88, 96, 177, 219, 306]. Take the the top-left image of Figure 7.1 as an example, both the *person* and the *tennis racket* can be used to create a contextual prior on where the *ball* should be.

Of these two models, which one is more effective for modeling context? A quick glimpse on the current state-of-the-art approaches, the idea of single region classification [114, 159, 165, 167, 226] with deep ConvNets [101, 252] is still dominating object detection. On the surface, these approaches hardly use any contextual reasoning; but we believe the large receptive fields of the neurons in fact do incorporate image-level context (See Figure 7.1 for evidences). On the other hand, there has been little or no success in modeling object-object relationships or instance-level context in recent years.

Why so? Arguably, modeling the instance-level context is more challenging. Instance-level reasoning for object detection would have to tackle bounding boxes pairs or groups in different classes, locations, scales, aspect ratios, *etc*. Moreover, for modeling image-level context, the grid structure of pixels allows the number of contextual inputs to be reduced efficiently (*e.g.* to a local neighborhood [35, 133, 245] or a smaller scale [252, 293]), whereas such reductions for arbitrary instances appear to be not so trivial. Above all, instance-level spatial reasoning inherently requires modeling *conditional* distributions on previous detections, but our current object detection systems do not have any **memory** to remember what to condition on! Even in the case of multi-class object detection, the joint layout [55] is estimated by detecting all objects in parallel followed by NMS [74]. What we need is an object detection system with memory built inside it!

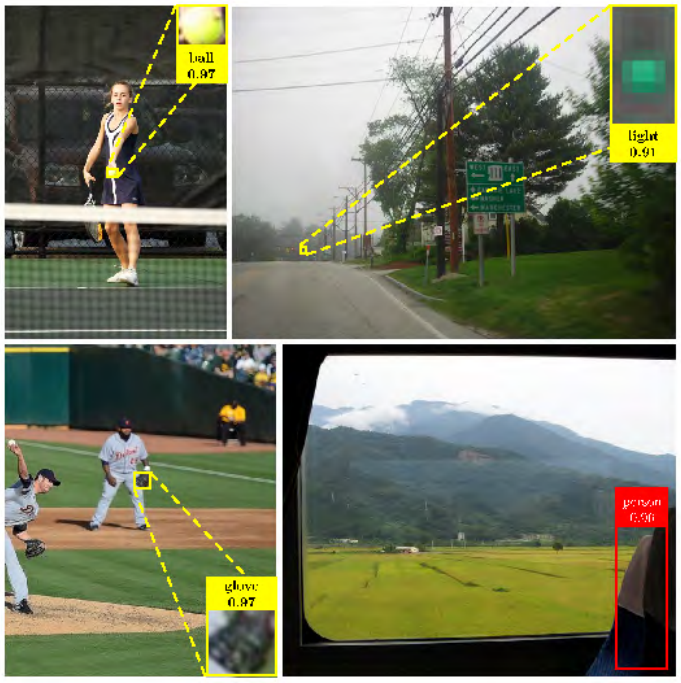


Figure 7.1: Evidence of image-level context reasoning inside ConvNets. All examples are from our baseline faster RCNN detector with VGG16 `conv5_3` features on COCO [166]. Numbers are class confidences. Top and bottom left: three examples where the ConvNet is able to detect tiny and simple-shaped objects much smaller than the receptive field size. Bottom right: a false positive detection for person given the seat on a passenger train. Our Spatial Memory Network takes advantage of this power by encoding multiple object instances into a "pseudo" image representation.

Memory has been successfully used in the recognition community recently for tasks such as captioning [41, 62, 180, 280, 285, 303] or visual question answering [5, 6, 84, 134, 171, 176, 244, 300, 302, 305, 315]. However, these works mostly focus on modeling an image-level memory, without capturing the spatial layout of the understanding so far. On the other hand, modeling object-object relationships requires **spatial** reasoning – not only do we need a memory to store the spatial layout, but also a suitable reasoning module to extract spatial patterns. To this end, this chapter presents a conceptually simple yet powerful solution, SMN, to model the instance-level context efficiently and effectively. Our key insight is that the best spatial reasoning module is a ConvNet itself! In fact, we argue that ConvNets are actually the most generic[1] and effective framework for extracting spatial and contextual information so far! Inspired by this observation, our spatial memory essentially assembles object instances back into a pseudo "image" representation that is easy to be fed into another ConvNet to perform object-object context reasoning.

However, if ConvNets are already so excellent at modeling context, why would we even bother something else? Isn't the image itself the ultimate source of information and therefore the best form of "spatial memory"? Given an image, shouldn't an ultra-deep network already take care of the full reasoning inside its architecture? In spite of these valid concerns, we argue that a spatial memory

[1]Many context models can be built or formulated as ConvNets [299, 311].

still presents as an important next step for object detection and other related tasks, for the following reasons:

- First, we note that current region-based object detection methods are still treating object detection as a *perception* problem, not a *reasoning* problem: the region classifier still produces multiple detection results around an object instance during inference, and relies on manually designed NMS [226] with a pre-defined threshold for de-duplication. This process can be sub-optimal. We show that with a spatial memory that memorizes the already detected objects, it is possible to learn the functionality of NMS automatically.

- Second, replacing NMS is merely a first demonstration for context-based reasoning for object detection. Since the spatial memory is supposed to store both semantic and location information, a legitimate next step would be full context reasoning: *i.e.*, infer the "what" and "where" of other instances based on the current layout of detected objects in the scene. We show evidence for such benefits on COCO [166].

- Third, our spatial memory essentially presents as a general framework to encode instance-level visual knowledge [157], which requires the model to properly handle the spatial (*e.g.* overlaps) and semantic (*e.g.* poses) interactions between groups of objects. Our approach follows the spirit of end-to-end learning, optimizing the representation for an end-task – object detection. Both the representation and the idea can be applied to other tasks that require holistic image understanding [6, 118, 316].

7.2 Related Work

As we already mentioned most related work for context and memory in Section 7.1, in this section we mainly review ideas that use sequential prediction for object detection. A large portion of the literature [91, 143, 172] focuses on sequential approaches for region proposals (*i.e.*, foreground/background classification). The motivation is to relieve the burden for region classifiers by replacing an exhaustive sliding-window search [74] with a smarter and faster search process. In the era of ConvNet-based detectors, such methods usually struggle to keep a delicate balance between efficiency and accuracy, since a convolution based 0/1 classifier (*e.g.* region proposal network [226]) already achieves an impressive performance when maintaining a reasonable speed. Sequential search has also been used for localizing small landmarks [254], but the per-class model assumes the existence of such objects in an image and lacks the ability to use other categories as context.

Another commonly used trick especially beneficial for reducing localization error is iterative bounding box refinement [85, 86, 226, 309], which leverages local image context to predict a better bounding box iteratively. This line of research is complementary to our SMN, since its goal is to locate the original instance *itself* better, whereas our focus is on how to better detect *other* objects given the current detections.

An interesting recent direction focuses on using deep reinforcement learning (DRL) to optimize the sequence selection problem in detection [15, 29, 162, 185]. However, due to the lack of full supervision signal in a problem with high-dimensional action space[2], DRL has so far only been used for bounding box refinements or knowledge-assisted detecton, where the action space is greatly reduced. Nevertheless, SMN can naturally serve as an encoder of the state in a DRL system to directly optimize average precision [103].

[2]Jointly reason about all bounding boxes and all classes.

Note that the idea of using higher-dimensional memory in vision is not entirely new. It has resemblance to spatial attention, which has been explored in many high-level tasks [154, 225, 302, 303]. To bypass NMS, LSTM [106] cells arranged in 2D order [262] and intersection-over-union (IoU) maps [112] have been used for single-class object detection. We also notice a recent trend in using 2D memory as a map for planning and navigation [95, 212]. Our work extends such efforts into generic, multi-class object detection, performing joint reasoning on both space and semantics.

7.3 Faster RCNN

Our spatial memory network is agnostic to the choice of base object detection model. In this chapter we build SMN on top of Faster R-CNN [226] (FRCNN) as a demonstration, which is a state-of-the-art detector that predicts and classifies Regions of Interest (RoIs). Here we first give a brief review of the approach.

7.3.1 Base Network

We use VGG16 [252] as the base network for feature extraction. It has 13 convolutional (`conv`), 5 max-pooling (`pool`), and 2 fully connected (`fc`) layers before feeding into the final classifier, and was pre-trained on the ILSVRC challenge [233]. Given an image $\mathcal{I}$ of height h and width w, feature maps from the last `conv` layer (`conv5_3`) are first extracted by FRCNN. The `conv5_3` feature size (h', w') is roughly γ=1/16 of the original image in each spatial dimension. On top of it, FRCNN proceeds by allocating two sub-networks for region proposal and region classification.

7.3.2 Region Proposal

The region proposal network essentially trains a class-agnostic objectness [4] classifier, proposing regions that are likely to have a foreground object in a sliding window manner [74]. It consists of 3 `conv` layers, one maps from `conv5_3` to a suitable representation for RoI proposals, and two $1{\times}1$ siblings on top of this representation for foreground/background classification and bounding box regression. Note that at each location, anchor boxes [226] of multiple scales (s) and aspect ratios (r) are used to cover a dense sampling of possible windows. Therefore the total number of proposed boxes is $K{\approx}h'{\times}w'{\times}s{\times}r$[3]. During training and testing, $k{\ll}K$ regions are selected by this network as candidates for the second-stage region classification.

7.3.3 Region Classification

Since the base network is originally an image classifier, region classification network inherits most usable parts of VGG16, with two caveats. First, because RoI proposals can be be arbitrary rectangular bounding boxes, RoI pooling [86, 114] is used in place of `pool` on `conv5_3` to match the the square-sized (7×7) input requirement for `fc6`. Second, the $1,000$-way `fc` layer for ILSVRC classification is replaced by two `fc` layers for C-way classification and bounding box regression respectively. Each of the C classes gets a separate bounding box regressor.

[3]Boarder anchors excluded.

Faster
R-CNN

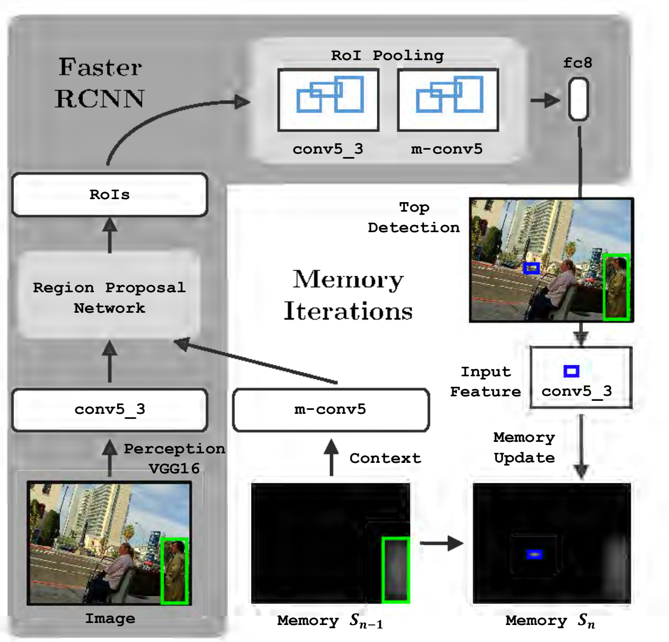


Figure 7.2: Overview of memory iterations for object detection. The original components from FRCNN are shown in the gray area. The old detection (*person*) is marked with a green box, and the new detection (*car*) is marked with blue. Here the network is unrolled one iteration.

7.3.4 De-duplication

We want to point out the often-neglected fact that a standard post-processing step is used in almost all detectors [74, 159, 167, 226] to disambiguate duplications – NMS. For FRCNN, NMS takes place in both stages. First, for region proposals, it prunes out the overlapping RoIs that are likely corresponding to the same object ("one-for-all-class") to train the region classifier. Second, for the final detection results, NMS is applied in an isolated, per-class manner ("one-for-each-class"). In this chapter, we still use NMS for RoI sampling during training [39], and mainly focus on building a model to replace the per-class NMS, with the hope that the model can encode the rich interplay across multiple classes when suppressing redundant detections.

7.4 Spatial Memory Network

To better motivate the use of spatial memory network, we resort to a mathematical formulation of the task at hand. For object detection, the goal is to jointly infer and detect all the object instances $\mathcal{O}=[O_1, O_2, O_3, \cdots, O_N]$ given an image $\mathcal{I}$, where N is the maximum number of object instances for any image[4]. Then the objective function of training a model (*e.g.* FRCNN) $\mathcal{M}$ is to maximize

[4] O_n denotes both the class and location of the object instance. When there is not enough foreground objects, the sequence can be padded with the background class.

the log-likelihood:

$$\begin{aligned}\arg\max_{\mathcal{M}} \mathcal{L} &= \log \mathrm{P}(O_{1:N}|\mathcal{M}, \mathcal{I}) \\ &= \sum_{n=1:N} \log \mathrm{P}\left(O_n|\mathcal{O}_{0:n-1}, \mathcal{M}, \mathcal{I}\right),\end{aligned} \tag{7.1}$$

where $\mathcal{O}_{0:n-1}$ is short for $[O_1, O_2, O_3, \cdots, O_{n-1}]$ and $\mathcal{O}_{0:0}$ is an empty set. Note that this decomposition of the joint layout probability is exact [66], regardless of the order we are choosing.

For a region-based object detector, Eq.(7.1) is approximated by detecting each object instance separately:

$$\arg\max_{\mathcal{M}} \mathcal{L} \approx \sum_{n=1:N} \log \mathrm{P}\left(O_n|\mathcal{M}, \mathcal{I}\right), \tag{7.2}$$

where NMS shoulders the responsibility to model the correlations in the entire sequence of detections. Since NMS is mostly[5] dependent on overlapping patterns, the information it can provide is limited compared to $\mathcal{O}_{0:n-1}$.

How can we do better? Inspired by networks that impose a memory [45, 66, 92, 106, 265] for sequential and reasoning tasks, and the two-dimensional nature of images, we propose to encode $\mathcal{O}_{0:n-1}$ in a spatial memory, where we learn to store all the previous detections. *I.e.*, we introduce memory variable $\mathcal{S}_{n-1}$, which gets updated each time an object instance is detected, and the approximation becomes:

$$\arg\max_{\mathcal{M}, \mathcal{S}} \mathcal{L} \approx \sum_{n=1:N} \log \mathrm{P}\left(O_n|\mathcal{S}_{n-1}, \mathcal{M}, \mathcal{I}\right), \tag{7.3}$$

where the memory $\mathcal{S}$ is jointly optimized with $\mathcal{M}$.

With the above formulation, the inference procedure for object becomes conditional: An empty memory is initialized at first (Section 7.4.1). Once an object instance is detected, selected cells (Section 7.4.2) in the memory gets updated (Section 7.4.4) with features (Section 7.4.3) extracted from the detected region. Then a context model (Section 7.4.5) aggregates spatial and other information from the memory, and outputs (Section 7.4.6) scores that help region proposal and region classification in FRCNN. Then the next potential detection is picked (Section 7.4.7) to update the memory again. This process goes on until a fixed number of iterations have reached (See Figure 7.2 for an overview).

We now describe each module, beginning with a description of the memory itself.

7.4.1 Memory

Different from previous works that either mixes memory with computation [45, 66, 106] or mimics the one-dimensional memory in the Turing machine/von Neumann architecture [288], we would like to build a two-dimensional memory for images. This is intuitive because images are intrinsically 2D mappings of the 3D visual world. But more importantly, we aim to leverage the power of ConvNets for context reasoning, which "forces" us to provide an image-like 2D input.

How big the memory should be spatially? For object detection, FRCNN that operates entirely on `conv5_3` features can already retrieve even tiny objects (*e.g.* the ones in Figure 7.1), suggesting that a resolution $1/16$ of the full image strikes a reasonable balance between speed and accuracy. At each location, the memory cell is a D=256 dimensional vector that stores the visual information discovered so far. Ideally, the initial values within the memory should capture the photographic bias

[5]Since NMS is applied in a per-class manner, there is also semantic information.

of a natural image, *i.e.*, prior about where a certain object tend to occur (*e.g. sun* is more likely to occur in the upper part). But the prior cannot be dependent on the input image size. To this end, we simply initialize the memory with a fixed spatial size ($20 \times 20 \times 256$ cells), and resize it according to the incoming `conv5_3` size using bilinear interpolation. In this way, the memory is fully utilized to learn the prior, regardless of different image sizes.

7.4.2 Indexing

The most difficult problem that previous works [92, 265] face when building an differentiable external memory is the design of memory indexing. The core problem is which memory cell to write to for what inputs. Luckily for our problem, strong correspondence between memory and 2D images solves this problem. Specifically, the target regions to look up in 2D memory are already provided. Furthermore, RoI pooling [86, 114] is precisely the operations needed to *read* off from the spatial memory[6]. The only remaining task is to create a *write* function that updates the memory given a detection. This can be divided into two parts, "what" (Section 7.4.3), and "how" (Section 7.4.4).

7.4.3 Input Features

It may appear trivial, but the decision of what features to insert into the memory requires careful deliberation. First, since `conv5_3` feature preserves spatial information, we need to incorporate it. Specifically, we use RoI pooling (without taking Max) to obtain the feature map at the location, and resize it to 14×14. However, merely having `conv5_3` is not sufficient to capture the higher-level semantic information, especially pertaining which object class is detected. The detection score is particularly useful for disambiguation when two objects occur in the same region, *e.g.*, a *person* riding a *horse*. Therefore, we also include `fc8` SoftMax score as an input, which is appended at each `conv5_3` locations and followed by two 1×1 `conv` layers to fuse the information (see Figure 7.3). We choose the full score over a one-hot class vector, because it is more robust to false detections.

7.4.4 Writing

Given the region location and the input features $\mathbf{x}_n$, we update the corresponding memory cells with a convolutional Gated Recurrent Unit [45] (GRU), which uses 3×3 `conv` filters in place of `fc` layers as weights. The GRU has a reset gate, and an update gate, shared at each location and activated with Sigmoid function $\sigma(\cdot)$. Hyperbolic tangent $tanh(\cdot)$ is used to constrain the memory values between $-1.$ and 1. For alignment, the region from the original memory $\mathcal{S}_{n-1}$ is also cropped with the same RoI pooling operation to 14×14. After GRU, the new memory cells are placed back to $\mathcal{S}_n$ with a reverse RoI operation.

7.4.5 Context Model

Now that the detected objects are encoded in the memory, all we have to do for context reasoning is stacking another ConNet on the top. In the current setup, we use a simple 5-layer all-convolutional network to extract the spatial patterns. Each `conv` filter has a spatial size of 3×3, and channel size of 256. Padding is added to keep the final layer `m-conv5` same size of `conv5_3`. To ease back-propagation, we add residual connections [101] every two layers.

[6]Although RoI pooling only computes partial gradients, back-propagation w.r.t. bounding box coordinates are not entirely necessary [226] and previously found unstable [114].

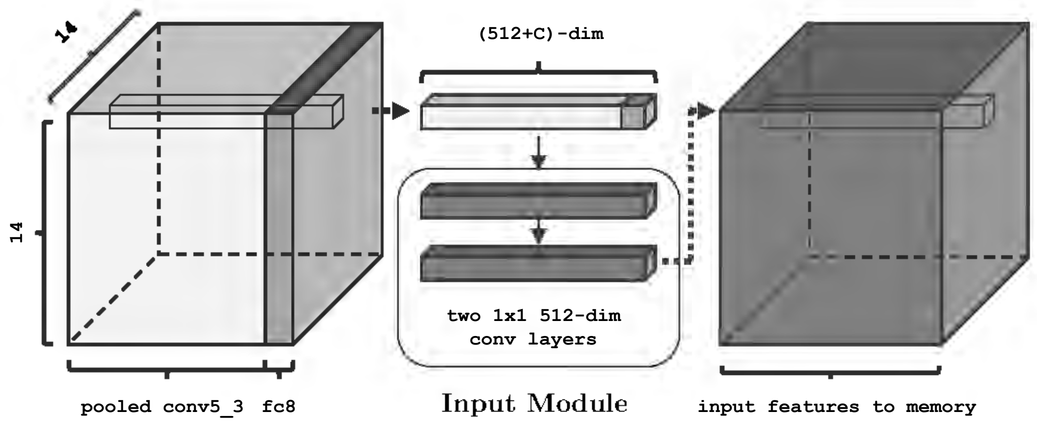


Figure 7.3: Illustration of the input module (Section 7.4.3). It assembles spatial and non-spatial features: detection scores after SoftMax (fc8) are tiled at each location of the RoI pooled 14×14 conv5_3 feature. Two additional conv layers are used to merge the information from two sources. Dotted arrow shows how the feature at one location is transformed.

7.4.6 Output

As for the module that outputs the reasoning results, we treat m-conv5 exactly the same way as conv5_3 in FRCNN: 3 conv layers for region proposal, and 2 fc layers with RoI pooling for region classification. The fc layers have 2048 neurons each.

We design another residual architecture to combine the memory scores with the FRCNN scores (see Figure 7.4): in the first iteration when the memory is empty, we only use FRCNN for detection; from the second iteration on, we add the memory predictions on top of the FRCNN ones, so that the memory essentially provides the additional context to close the gap. This design allows a handy visualization of the prediction difference with/without context. But more importantly, such an architecture is critical to let us converge the full network. Details for this are covered in Section 7.5.1.

7.4.7 Selecting Next Region

Since spatial memory turns object detection into a sequential prediction problem, an important decision to make is which region to take-in next [16]. Intuitively, some objects are more useful serving as context for others (*e.g. person*) [79, 94, 96, 306], and some object instances are easier to detect and less prone to consequent errors. However, in this chapter we simply follow a greedy strategy – the most confident foreground object box is selected to update the memory, leaving more advanced models that directly optimize the sequence [268] as future work.

7.5 Training

Like a standard network with recurrent connections, our SMN is trained by back-propagation through time (BPTT) [295], which unrolls the network multiple times before executing a weight-update. However, apart from the well-known gradient propagation issue, imposing the conditional structure on object detection incurs new challenges for training. Interestingly, the most difficult one we face in our experiment, is the “straightforward” task of de-duplication.

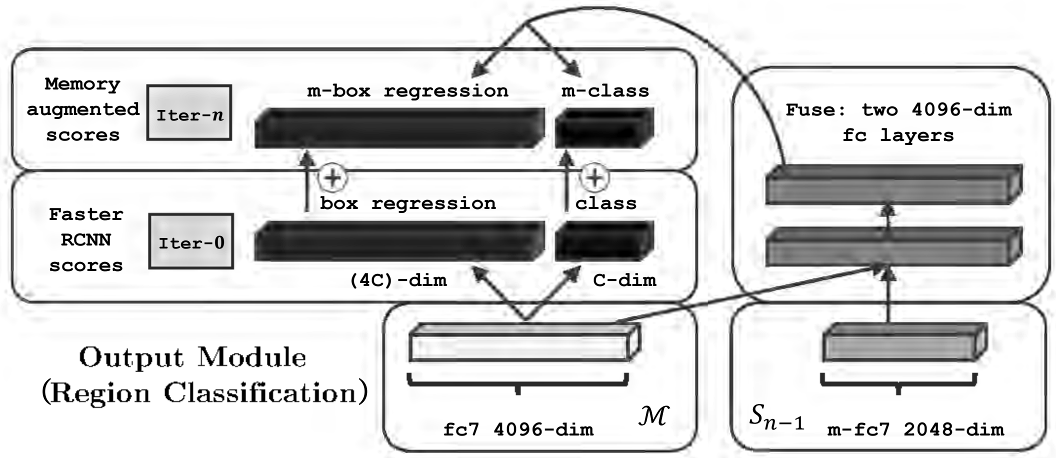


Figure 7.4: Illustration of the output module (Section 7.4.6) for region classification. FRCNN scores are optimized at the first iteration when memory is empty, and then augmented with memory scores in later iterations. Same is done for region proposals. Two additional `fc` layers are used to fuse FRCNN and memory features.

7.5.1 Learning De-duplication

Simply put, the functionality of de-duplication is: how can the network learn that a detected instance should no longer be detected again? More specifically, we need to design the output module (Section 7.4.6) to fuse the memory ($\mathcal{S}$) and FRCNN ($\mathcal{M}$) beliefs and predict intelligently: when the memory is empty, the FRCNN score should be used; but when the memory has the instance stored, the network needs to ignore, or *negate* the cue from FRCNN.

Since multi-layer networks are universal function approximators [111], our first attempt is to fuse the information by directly feeding into a multi-layer network (Figure 7.5 (a)). However, joint-training fails to even converge FRCNN. Suspicious that the longer, weaker supervision might be the cause, we also added skip connections [14] to guide the FRCNN training directly (Figure 7.5 (b)). Yet it still does not help much. Tracking the learning process, we find where the actual problem lies – because the network needs to de-duplicate, it keeps receiving contradicting signals: the normal one that guides perception, and the adversarial one that prevents *more* perception. And because $\mathcal{S}$ also starts off from scratch, the signal it can provide is also weak and unreliable. As a result, part of both error signals are back-propagated to $\mathcal{M}$[7], causing trouble for learning further.

Realizing where the issue is, a direct solution is to just stop the adversarial signal from flowing back and canceling the normal one. Therefore, we stopped the gradient to FRCNN from second iteration on (Figure 7.5 (c)), and the network can successfully converge.

To make it easy for training and showing the confidence changes for consequent detections given the context, we further reduced the architecture to exclude all memory related weights in the first iteration (Figure 7.5 (d)). This way, the change in predictions with/without memory can be read-off directly[8], and training can be done separately for $\mathcal{M}$ and $\mathcal{S}$.

[7]Since there are two sets of scores (from $\mathcal{M}$ and fused `fc`) added together for prediction in Figure 7.5 (b), we find the conflicting signals are also propagated to the *biases* of these predictions: resulting in one going up and the other down while essentially canceling each other.

[8]Otherwise we have to run the inference again with $\mathcal{S}_0$.

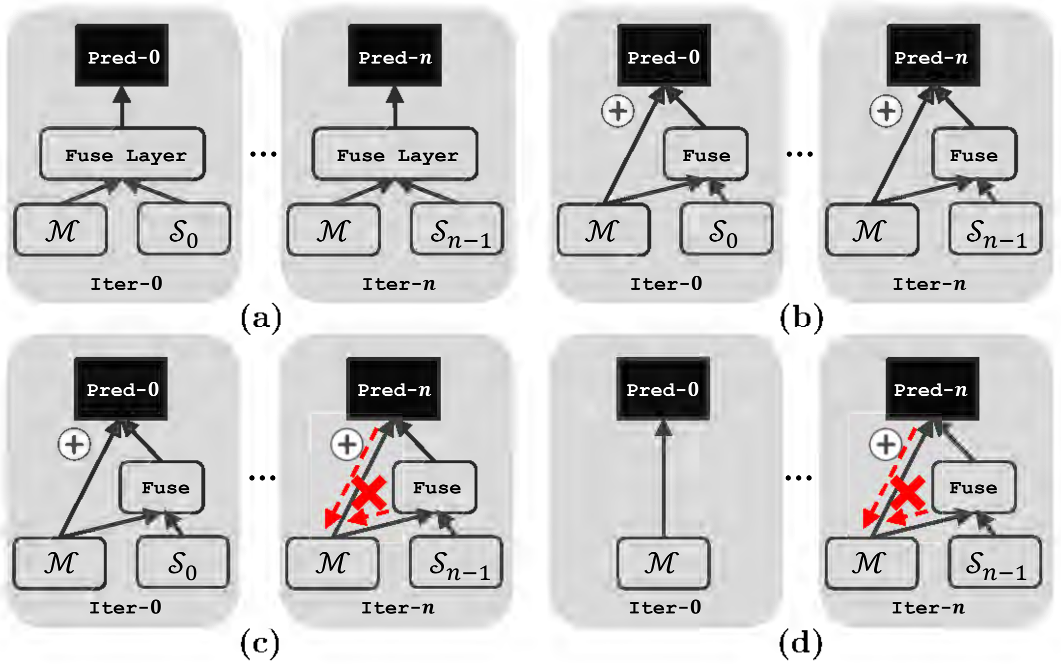


Figure 7.5: Four design choices for learning the functionality of de-duplication. $\mathcal{M}$ is FRCNN features, and $\mathcal{S}_{n-1}$ represents memory features. Each design is shown by two gray panels showing the information flow of Iteration 0 (left) and Iteration n>0 (right). We find it hard to even converge the network when the gradient is back-propagated to FRCNN in all iterations (a) & (b). Stop the gradient in later iterations (c) can successfully converge the network, and our final design (d) separates *perception* from *reasoning* and makes it easy to visualize the effect of context. All design choices are abstract and apply to both region proposal and classification. Please see Section 7.5.1 for more details.

7.5.2 RoI Sampling

To avoid getting overwhelmed by negative boxes, FRCNN enforces a target sampling ratio for foreground/background boxes. The introduction of a spatial memory that learns to de-duplicate, brings in another special type – regions whose label is flipped from previous iterations. To keep these regions from being buried in negative examples too, we changed the sampling distribution to include flipped regions.

It is important to point out that RoI sampling greatly enhances the robustness of our sequential detection system. Because only $k \ll K$ regions are sampled from all regions, the overall most confident RoI is not guaranteed to be picked when updating the memory. This opens up chances for other highly confident boxes to be inserted into the sequence as well [268] and reduces over-fitting.

7.5.3 Multi-Tasking

We also practiced the idea of multi-task learning for SMN. The major motivation is to force the memory to memorize more: the basic SMN is only asked fulfill the mission of predicting the missing objects, which does not necessarily translate to a good memorization of previously detected objects.

E.g., it may remember that one region has an object in general, but does not store more categorical information beyond that. To better converge the memory, we also added a *reconstruction* loss [41, 228], *i.e.*, letting the network in addition predict the object classes it has stored in the memory. Specifically, we add an identical set of branches on top of the `m-conv5` features as FRCNN, for both region proposal and region classification in each iteration. These weights are used to predict only the previously detected objects.

7.5.4 Stage-wise Training

Thanks to the design of our memory augmented prediction, so far we have trained the full model in two separate stages, where FRCNN $\mathcal{M}$, the *perception* model can be optimized independently at first; then the *reasoning* model with spatial memory $\mathcal{S}$ is learned on top of fixed $\mathcal{M}$. This helps us isolate the influence of the base model and focus directly on the study of SMN.

For efficiency, we also follow a curriculum learning [17] strategy: bootstrap a SMN of more iterations (*e.g.* N=10) with a pre-trained SMN of fewer iterations (*e.g.* N=5). As N gets larger, the task becomes harder. Curriculum learning does not require re-learning de-duplication (which we learn with N from 2 to 4), and allows the network to focus more on object-object relationships instead.

7.5.5 Hyper-parameters

Given a pre-trained FRCNN or SMN (in the case of curriculum learning), we train a fixed number of 30k steps. The initial learning rate is set to $1e{-}3$ and reduced to $1e{-}4$ after 20k steps. Since we do not use automatic normalization tricks [115, 168], different variances are manually set when initializing weights from scratch, in order to let different inputs contribute comparably (*e.g.* when concatenating `fc7` and `m-fc7`). Other hyper-parameters are kept the same to the ones used in FRCNN.

7.6 Experimental Results

We highlight the performance of our spatial memory network on COCO [166]. However, for ablative analysis and understanding the behaviour of our system, we use both PASCAL VOC 2007 [67] and COCO [166]. For VOC we use the *trainval* split for training, and *test* for evaluation. For COCO we use *trainval35k* [14] and *minival*. For evaluation, toolkits provided by the respective dataset are used. The main metrics (mAP, AP and AR) are based on detection average precision/recall.

Implementation Details: We use TensorFlow to implement our model, which is built on top of the open-sourced FRCNN implementation[9] serving as a baseline. For COCO, this implementation has an AP of 29.1% compared to the original one 24.2% [226].

Original FRCNN uses NMS for region sampling as well. However, NMS hurts our performance more since we do sequential prediction and one miss along the chain can negatively impact all the follow-up detections. To overcome this disadvantage, we would ideally like to examine all K regions in a sliding window fashion. However, due to the GPU memory limit, the top 5k regions are used instead. We analyze this choice in ablative analysis (Section 7.6.2). Due to the same limitation, our current implementation of SMN can only unroll N=10 times in a single GPU. At each timestep in SMN, we do a soft max-prediction for the top box selected, so that a single box can be assigned to multiple classes. We will also justify and analyze this choice in Section 7.6.2.

[9] `https://github.com/endernewton/tf-faster-rcnn`

Table 7.1: Baseline and initial analysis on COCO 2014 minival when constraining the number of detections N=5/10. AP and AR numbers are from COCO evaluation tool.

N	**Method**	AP	AR-10	AR-S	AR-M	AR-L
-	FRCNN [226]	24.2	33.7	11.7	39.5	54.1
-	Baseline [39]	29.1	38.7	17.7	44.9	56.9
N=5	Baseline	23.8	27.8	7.0	28.7	48.4
	SMN	**24.5**	**28.9**	**7.3**	**29.7**	**50.6**
N=10	Baseline	27.1	33.5	10.8	36.7	53.8
	SMN	**28.1**	**35.0**	**11.5**	**38.1**	**56.4**

Initial Results: Table 7.1 shows the initial results of our approach as described. As it can be seen for N=5 detections per image our SMN give an AP of 24.5% and for N=10 if gives an AP of 28.1%. When the baseline is allowed the same number of detections (N=5, 10), the AP is 23.8% and 27.1%. Therefore, while we do outperform baseline for fixed number of detections per image, due to limited roll-out capability we are still $\sim 1\%$ below the baseline [39].

7.6.1 SMN for Hard Examples

In this section, we want to go beyond N=10 detections and see if the overall detection performance can be improved with SMN. Intuitively, for highly confident detections, ConvNet-based FRCNN is already doing a decent job and not much can be learned from an additional memory. It is the "tails" that need help from the context! This means two things: 1) with a limited resource budget, SMN should be used in later iterations to provide conditional information; and 2) at the beginning of the sequence, a standard FRCNN can work as a proxy. Given these insights, we experimented with the following strategy: For the first N_1 iterations, we use a standard FRCNN to detect easier objects and feed the memory with a sequence ordered by FRCNN confidence (after per-class NMS). Memory gets updated as objects come in, but does not output features to augment prediction. Only for the later N_2 iterations it acts normally as a context provider to detect harder examples. For COCO, we set N_1=50 and bootstrap from a N_2=10 SMN model.

Although SMN is trained with the goal of context reasoning and learns new functionality (*e.g.* de-duplication) that the original FRCNN does not have, it does have introduced more parameters for memory-augmented prediction. Therefore, we also add a MLP baseline, where a 5-layer ConvNet (Section 7.4.5) is directly stacked on top of `conv5_3` for context aggregation, and the same output modules (Section 7.4.6) are used to make predictions.

The results can be found in Table 7.2. As can be seen, on our final system, we are 2.2% better than the baseline FRCNN. This demonstrates our ability to find hard examples. It is worth noting that here *hard* does not necessarily translate to *small*. In fact, our reasoning system also helps big objects, potentially due to its ability to perform de-duplication more intelligently and benefit larger objects that are more likely to overlap.

Qualitative Results: We show a couple of examples of how context using spatial memory can help improve the performance and detections. In the first case, the score of *sheep* gets boosted due to other *sheep*. The score of *horse* decreases due to the detection of *cake* and *table*.

Table 7.2: Final comparison between SMN and baselines. We additionally include MLP baseline where the number of parameters are kept the same as SMN for context aggregation and output. Top 5k regions are used to select proposal instead of NMS.

Method	AP	AP-.5	AP-.75	AP-S	AP-M	AP-L	AR-S	AR-M	AR-L
Baseline [39]	29.4	50.0	30.9	12.2	33.7	43.8	18.5	45.5	58.9
MLP	30.1	50.8	31.7	12.5	34.2	44.5	19.2	47.0	59.8
SMN	**31.6**	**52.2**	**33.2**	**14.4**	**35.7**	**45.8**	**20.5**	**48.8**	**63.2**

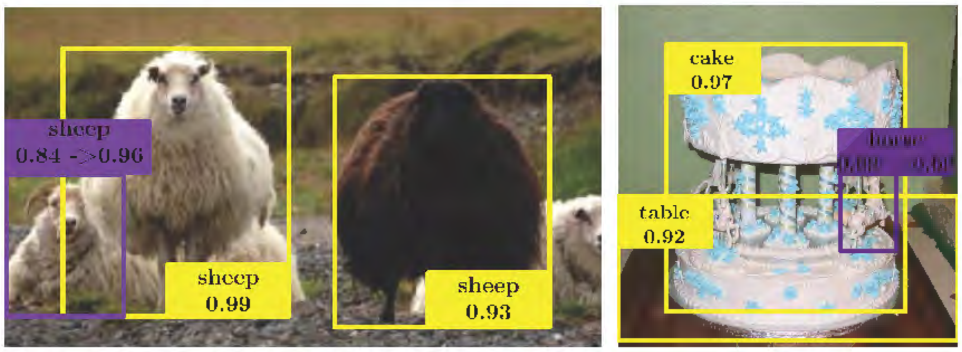


Figure 7.6: Examples of context has helped improve scores by reasoning. Left: the score of *sheep* is increased due to presence of other *sheep* in background. Right: the score of *horse* is decreased due to the detection of *cake* and *table*.

7.6.2 Ablative Analysis

We now perform ablative analysis to explain all our choices for the final implementation. For ablative analysis, we use both VOC and COCO datasets. The numbers are summarized in Table 7.3. For the comparisons shown here, we switch back to the standard NMS-based region sampling and select top k=300 RoIs as in original FRCNN. Also, when we do the roll-out, at each step we choose one detection and perform HardMax (rather than SoftMax): make the hard decision about what class does the selected box belong to – a natural idea for sequential prediction.

For N=5, we compared three models. First, SMN Base, where we simply train the network as is done in FRCNN. Next, regions with flipped labels (Section 7.5.2) are added to replaces some of the negative example – for training region proposal the ratio for positive/flipped/negative is 2:1:1, and for region classification it is 1:1:2. Third, SMN Full, where we keep the previous sampling strategy and in addition include the reconstruction loss (Section 7.5.3). Overall, both strategies help performance but with a seemly different strength: sampling flipped regions helps more on small objects, and multi-task learning helps more on bigger ones.

However, our best performance in Table 7.3 is still behind the baseline and judging from the COCO AR we believe the biggest issue lies in recall. Therefore, we take the best SMN Full model and conduct two other investigations specifically targeting recall. Here we only list the final results, please see Section 7.6.4 for more detailed list.

SoftMax *vs*. HardMax: First, we address a subtle question: if we take top N detections with the memory and compare them directly with top N detections of Faster R-CNN: are these results comparable? It turns out to be not! As mentioned in Section 7.3.4, because NMS is applied in a

Table 7.3: Ablative analysis on VOC 2007 test and COCO 2014 minival. All approaches constrained by detections N=5/10. mAP is used to evaluate VOC, AP and AR numbers are from COCO.

N	**Method**	mAP	AP	AR-10	AR-S	AR-M	AR-L
N=5	Baseline (FR-CNN)	**65.8**	23.6	27.6	**7.0**	**29.1**	**47.4**
	SMN Base	63.6	23.3	27.2	6.7	28.0	46.1
	+ Sample Flipped	64.4	23.5	27.2	6.9	28.4	46.4
	SMN Full	64.6	**23.8**	**27.7**	6.9	28.5	**47.4**
N=10	Baseline	**70.3**	26.9	**33.2**	**10.9**	**36.6**	**52.7**
	SMN Full	67.5	26.6	32.6	10.3	35.6	52.1
	+Tune from N=5	67.8	**27.1**	32.7	10.3	35.9	52.3

Table 7.4: Investigating the recall issue. **S** stands for SoftMax based testing, and **H** for HardMax. $\not{\mathbf{N}}$ is short for Non-aggressive NMS, where top 5k RoIs are directly selected without NMS.

N	**Method**	$\not{\mathbf{N}}$	**Max**	mAP	AP	AR-10	AR-S	AR-M	AR-L
N=5	Baseline	✗	**S**	65.8	23.6	27.6	7.0	29.1	47.4
	SMN Full	✗	**S**	66.4	24.1	28.8	**7.5**	**29.7**	50.0
	Baseline	✗	**H**	65.4	23.5	27.2	6.7	28.6	46.9
	SMN Full	✗	**H**	64.6	23.8	27.7	6.9	28.5	47.4
	Baseline	✓	**S**	66.0	23.8	27.8	7.0	28.7	48.4
	SMN Full	✓	**S**	**66.6**	**24.5**	**28.9**	7.3	**29.7**	**50.6**
N=10	Baseline	✗	**S**	70.3	26.9	33.2	10.9	36.6	52.7
	SMN Full	✗	**S**	69.4	27.7	**35.0**	**11.6**	37.6	55.7
	Baseline	✗	**H**	68.0	26.4	31.9	9.7	35.0	50.7
	SMN Full	✗	**H**	67.8	27.1	32.7	10.3	35.9	52.3
	Baseline	✓	**S**	**70.4**	27.1	33.5	10.8	36.7	53.8
	SMN Full	✓	**S**	70.0	**28.1**	**35.0**	11.5	**38.1**	**56.4**

per-class manner, the actual number of box candidates it can put in the final detection is $k \times C$. To make it more clear, for a confusing region where *e.g.* the belief for *laptop* is 40% and *keyboard* is 35%, NMS can keep both candidates in the top N detections, whereas for SMN it can only keep the maximum one[10]. Therefore, to be fair, we try: a) HardMax for baseline; and b) SoftMax for SMN.
Non-aggressive NMS: Finally, we also evaluate our choice of non-aggressive NMS during RoI sampling. Both baseline and SMN perform better with 5k proposals; however our boost on AP is more significant due to sequential prediction issues.

7.6.3 More Qualitative Results

We show more qualitative results in Figure 7.7 to present how the current version of SMN reasons with context for object detection. On the top left, the confidence of *skis* is increased due to the

[10]It's also a result of our current input feature design, where we only used `fc8` and `conv5_3` features to update the memory without a top-down notion [248] of which class is picked, so there's no more need for SMN to return.

Table 7.5: VOC 2007 test object detection average precision. **S** stands for SoftMax based testing, and **H** for HardMax. **N̸** is short for Non-aggressive NMS, where top 5k RoIs are directly selected without NMS.

N	**Method**	**N̸**	**Max**	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
N=5	Baseline [39]	✗	**S**	65.8	66.8	71.3	66.1	50.0	42.0	74.5	79.6	79.3	42.4	76.3	58.0	77.3	79.4	69.1	70.1	42.5	60.0	64.8	75.0	72.2
	SMN Full	✗	**S**	66.4	66.3	75.3	65.4	53.3	42.2	74.1	79.5	81.9	44.5	72.7	61.9	76.5	77.0	69.7	70.1	41.8	63.9	65.5	75.1	71.6
	Baseline	✗	**H**	65.4	67.3	71.3	60.1	50.0	41.9	74.6	79.6	79.3	42.4	76.4	57.9	77.2	79.4	69.3	70.1	42.5	60.0	64.8	75.1	68.6
	SMN Full	✗	**H**	64.6	60.7	71.4	65.3	50.9	42.2	74.2	79.4	78.9	42.3	67.9	59.0	75.9	72.4	69.5	70.0	41.8	59.6	65.5	75.0	68.1
	Baseline	✓	**S**	66.0	67.4	71.2	66.8	51.5	41.7	75.3	79.8	79.1	43.3	76.7	57.6	76.6	79.6	69.8	70.3	41.2	60.3	64.7	76.0	71.5
	SMN Full	✓	**S**	66.6	65.5	71.4	66.1	54.6	41.7	75.2	79.6	82.4	45.8	75.4	63.1	76.6	77.6	69.2	70.5	41.0	63.7	64.8	76.0	71.1
	Baseline	✓	**H**	65.8	67.4	71.2	66.8	51.2	41.7	75.3	79.8	79.0	43.1	76.7	57.6	76.8	79.6	69.8	70.3	41.3	60.3	64.8	75.8	68.2
	SMN Full	✓	**H**	65.4	60.8	71.3	66.1	51.8	41.9	75.0	79.6	78.4	43.3	75.3	62.8	76.4	78.1	69.2	70.5	40.9	59.2	64.4	75.8	67.9
N=10	Baseline	✗	**S**	70.3	67.5	78.5	67.1	53.4	54.3	78.0	84.7	84.4	48.9	82.1	66.4	77.3	80.8	75.2	77.0	46.1	70.7	64.8	75.0	73.6
	SMN Full	✗	**S**	69.4	66.8	79.0	69.1	52.3	53.9	73.7	82.8	83.6	46.6	78.5	64.2	76.7	80.2	75.0	77.1	44.6	67.2	67.7	75.9	72.7
	Baseline	✗	**H**	68.0	67.5	78.5	67.1	50.4	50.3	74.9	79.9	79.4	47.0	77.0	64.8	77.3	80.9	69.7	77.0	43.4	67.0	65.1	75.0	69.0
	SMN Full	✗	**H**	67.8	67.0	79.0	66.6	49.8	49.8	73.8	79.8	79.6	42.5	75.9	64.1	76.7	80.2	75.1	77.0	42.6	64.7	66.4	76.0	68.5
	Baseline	✓	**S**	70.4	67.5	79.0	67.6	55.2	53.4	78.9	84.5	84.0	49.6	82.0	63.4	80.3	80.6	75.7	77.3	44.8	66.7	65.8	78.5	73.2
	SMN Full	✓	**S**	70.0	68.3	78.1	69.5	55.0	53.6	77.7	85.1	82.5	49.2	78.0	63.8	76.5	80.0	76.0	77.5	44.3	67.6	66.6	78.7	71.8
	Baseline	✓	**H**	68.8	67.3	79.0	67.5	52.2	49.2	75.3	80.1	79.2	47.6	81.8	63.5	76.5	80.6	75.4	77.3	42.1	66.7	64.9	76.0	73.1
	SMN Full	✓	**H**	68.3	66.2	78.1	66.6	51.7	49.9	75.8	85.0	78.9	47.6	76.1	64.1	76.7	79.9	76.2	77.4	42.0	65.1	65.4	76.1	67.9

Table 7.6: VOC 2007 test object detection average precision. We use SoftMax and top 5k RoIs during testing for all the methods compared (except [226]).

Method	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
FRCNN [226]	70.0	68.7	79.2	67.6	54.1	52.3	75.8	79.8	84.3	**50.1**	78.3	65.1	**82.2**	**84.8**	72.9	76.0	44.9	70.9	63.3	76.1	72.6
Baseline [39]	**71.2**	67.6	78.9	67.6	55.2	**56.9**	**78.8**	**85.2**	83.9	49.8	**81.9**	65.5	80.1	84.4	75.7	77.6	**45.3**	70.8	66.9	78.2	**72.9**
MLP	70.9	**71.7**	80.0	**70.9**	**60.0**	56.6	78.2	85.0	**85.5**	47.5	72.7	64.2	76.6	83.5	**75.8**	**77.8**	45.2	**72.3**	**68.1**	76.3	70.4
SMN	71.1	67.1	**81.2**	70.3	55.5	54.0	78.3	85.1	83.7	49.4	80.9	**66.1**	80.1	83.5	75.7	77.7	45.1	69.7	67.1	**78.4**	72.6

detection of *person* and their relative location. On the top right, the confidence of *tennis racket* is increased despite the motion blur owing to the *person* and her pose. Similarly, the *backpack* on the middle left gains confidence due to the *person* carrying it. On the middle right, we show the example of an occluded *sheep* detection. SMN is able to go beyond the overlapping reasoning of NMS, which will prevent the *sheep* in the back from being detected. On the bottom row, we show two failure examples, where the potatoes are mistaken as *pizza* in the container (left), and the suppression of baby (*person*) given the *person* holding it.

7.6.4 Category-wise Ablative Analysis on VOC

We believe it is interesting to check the category-wise numbers and get a more insightful idea for the ablative analysis part.

In Table 7.3, we listed our ablative analysis on different training strategies, but the best mAP we can reach on VOC is still behind the baseline: for N=5 it is 64.6% compared to 65.8%; for N=10 it is 67.8% compared to 70.3%. Judging from the COCO AR metrics, we speculate the issue lies in recall. This motivates us to take the best SMN Full model and conduct an investigation specifically targeting recall.

Turns out, the biggest issue lies in the **SoftMax *vs.* HardMax** strategy. For SMN, we initially deployed a HardMax one: given a bounding box, we proceed with the most confident class (and take the bounding box after regression corresponding to that particular class). This ignores all the rest classes that are potentially competitive. *E.g.*, on COCO *snowboard* is usually confused with *skis*

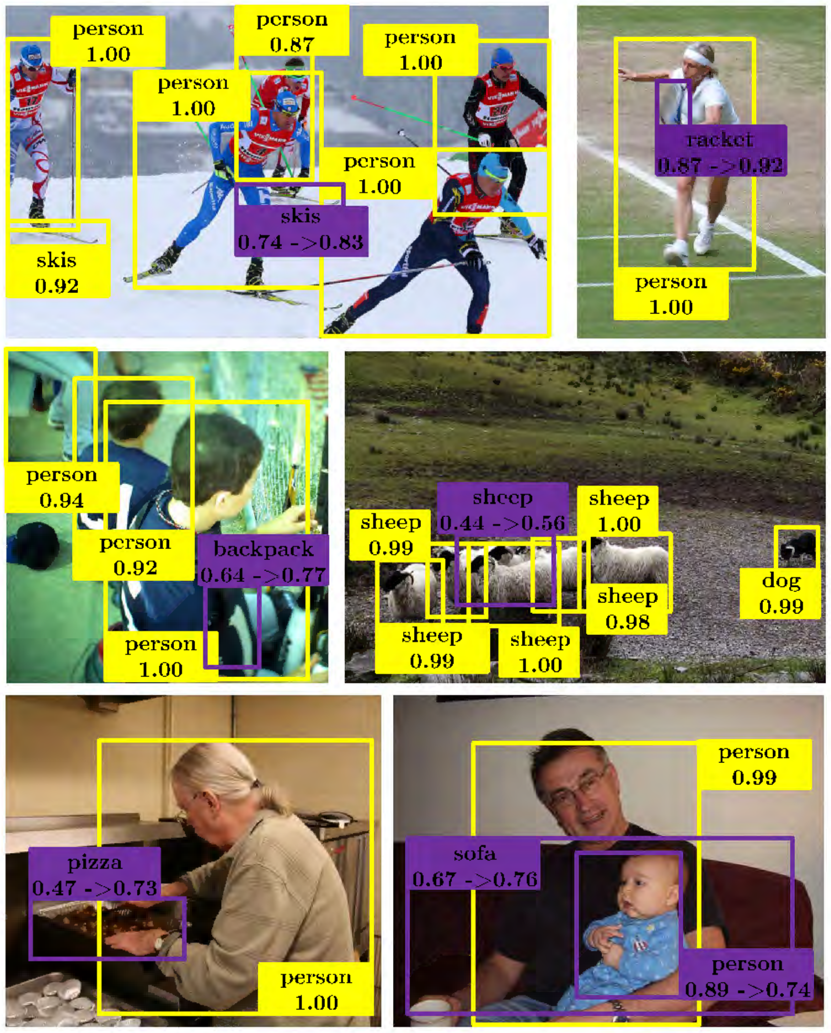


Figure 7.7: Four successful reasoning examples and two failure cases. Please see Section 7.6.3 for a detailed explanation.

when buried in the snow; *hot-dog* is usually confused with *pizza* when held in a person's hands. Note that this does not cause a problem for NMS, because the de-duplication is done in a per-class manner. Therefore, for a confusing bounding box where *e.g.* the belief for *laptop* is 50% and *keyboard* is 35%, NMS can keep both candidates in the top N detections, whereas for SMN *keyboard* is suppressed because *laptop* has a higher confidence. While in theory this is not an issue because SMN can *learn* to revisit the same region, we find it extremely unlikely to happen in practice, mainly due to the heavy burden on SMN to learn de-duplication automatically and may also attribute to our current feature design. To investigate whether it is indeed the case, we added two ablative experiments: a) using HardMax strategy for baseline with NMS, meaning an initial proposal can only be selected *once* – by the most likely class; and b) using SoftMax for SMN where the final N detections can come from all classes of the N bounding boxes returned by sequential prediction.

As shown in Table 7.4, we find it worked in both ways: the recall indeed boosts for SMN when a SoftMax strategy is used; and removing the confusing categories for the same bounding box hurts the recall for NMS. For COCO, the improvement on AR metrics directly reflects this finding, however it is less obvious for VOC. Here we additionally include evidence from category-wise results in Table 7.5 to corroborate the observation. For example, categories like *cow* and *sheep* get consistent improvements in SMN since they are more likely to confuse, where as distinctive categories like *person* almost remain the same. Overall SoftMax outperforms HardMax in most cases.

Normally, k=300 RoIs are selected by NMS (*i.e.* region proposal) before feeding into region classification. However, SMN as a sequential prediction method is more vulnerable to such an aggressive region selection scheme, because one miss along the chain can negatively impact all the follow-up detections. Therefore, in addition to the two strategies, we also include the analysis on the impact of the number of regions sampled. Specifically, we include a non-aggressive NMS scheme, where the top $5k$ proposals are directly selected without NMS.

Note that because the baseline feeds more RoIs (k=300 or k=$5,000$) for final evaluation, it still bears a subtle advantage over SMN when testing with HardMax. For example, if bounding box B_1 suppresses *cow* over *sheep*, there is still chance that a nearby (*e.g.* meansured by IoU) bounding box B_2 where *sheep* is selected over *cow*. On the other hand, SMN gets $N \ll k$ chances for picking the candidates. This difference is reflected when we compare k=300 *vs.* k=$5,000$ for baseline: SoftMax based testing has a larger margin over HardMax when k is smaller. Regardless of this, SMN is still able to achieve on-par (N=10) or better (N=5) results in terms of mAP.

7.6.5 Final Results on VOC

We also excluded the final results on VOC comparing 1) the baseline FRCNN, 2) the MLP where a 5-layer ConvNet is directly stacked on top `conv5_3` for context aggregation, and 3) our SMN. We report the results here. For the final evaluation, we use top 5k RoIs for region sampling and the SoftMax strategy for all methods. Due to the memory limitation, the same idea of first using NMS to find easy examples, storing them in the spatial memory and then predicting with SMN is used (Section 7.6.1). For VOC we set N_1=10 and bootstrap from a N_2=10 SMN model, so in total N=20 bounding boxes are sent for evaluation.

The result can be found in Table 7.6. As can be seen, our method is on-par with baseline and MLP ($\sim$71% mAP). This difference to COCO is reasonable since compared to COCO, there is not much "juice" left for context reasoning, in terms of both *quantity* (number of images to train SMN on top of FRCNN) and *quality* (how difficult the detection of objects are in the scene).

Chapter 8

Iterative Reasoning

8.1 Introduction

Finally, it's time to put everything together, and build a framwork for reasoning.

In recent years, we have made significant advances in standard recognition tasks such as image classification [101], detection [226] or segmentation [9]. Most of these gains are a result of using feed-forward end-to-end learned ConvNet models. Unlike humans where visual reasoning about the space and semantics is crucial [24], our current visual systems lack any context reasoning beyond convolutions with large receptive fields. Therefore, a critical question is how do we incorporate both *spatial* and *semantic* reasoning as we build next-generation vision systems.

Our goal is to build a system that can not only extract and utilize hierarchy of convolutional features, but also improve its estimates via spatial and semantic relationships. But what are spatial and semantic relationships and how can they be used to improve recognition? Take a look at Figure 8.1. An example of spatial reasoning (top-left) would be: if three regions out of four in a line are "window", then the fourth is also likely to be "window". An example of semantic reasoning (bottom-right) would be to recognize "school bus" even if we have seen few or no examples of it – just given examples of "bus" and knowing their connections. Finally, an example of spatial-semantic reasoning could be: recognition of a "car" on road should help in recognizing the "person" inside "driving" the "car".

A key recipe to reasoning with relationships is to *iteratively* build up estimates. Recently, there have been efforts to incorporate such reasoning via top-down modules [229, 292] or using explicit memories [182, 300]. In the case of top-down modules, high-level features which have class-based information can be used in conjunction with low-level features to improve recognition performance. An alternative architecture is to use explicit memory. For example, Chen & Gupta [40] performs sequential object detection, where a *spatial memory* is used to store previously detected objects, leveraging the power of ConvNets for extracting dense context patterns beneficial for follow-up detections.

However, there are two problems with these approaches: a) both approaches use stack of convolutions to perform *local* pixel-level reasoning [60], which can lack a *global* reasoning power that also allows regions farther away to directly communicate information; b) more importantly, both approaches assume enough examples of relationships in the training data – so that the model can learn them from scratch, but as the relationships grow exponentially with increasing number of classes, there is not always enough data. A lot of semantic reasoning requires learning from few or no examples [73]. Therefore, we need ways to exploit additional structured information for visual

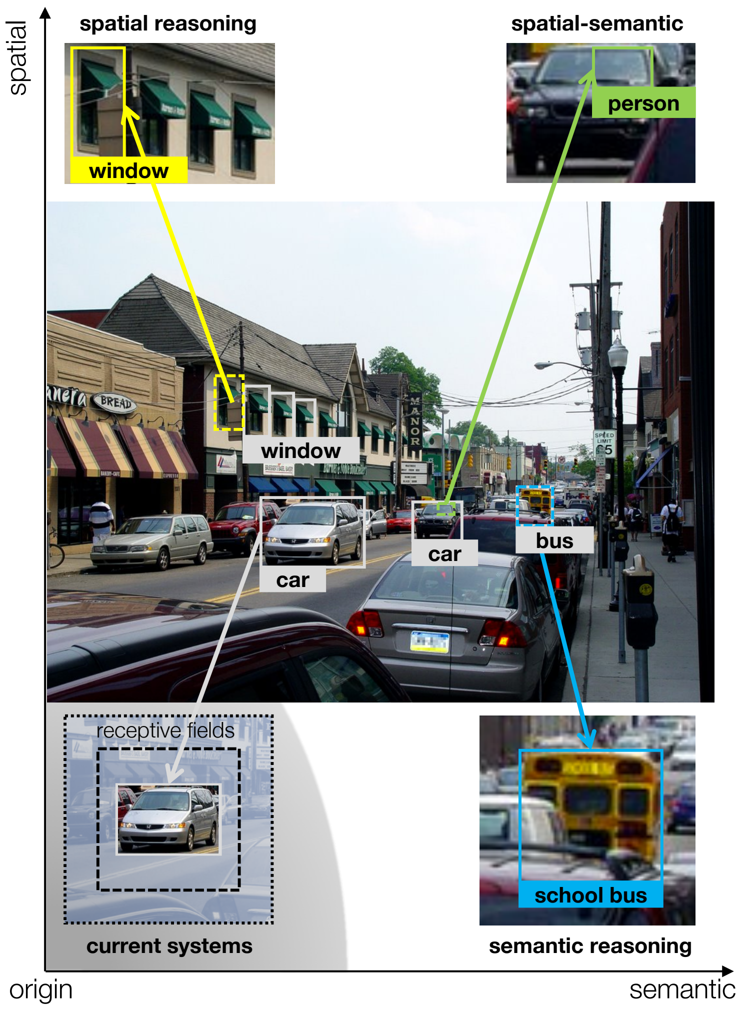


Figure 8.1: Current recognition systems lack the reasoning power beyond convolutions with large receptive fields, whereas humans can explore the rich space of spatial and semantic relationships for reasoning: *e.g*. inferring the fourth "window" even with occlusion, or the "person" who drives the "car". To close this gap, we present a generic framework that also uses relationships to iteratively reason and build up estimates.

reasoning.

In this paper, we put forward a generic framework for both spatial and semantic reasoning. Different from current approaches that are just relying on convolutions, our framework can also learn from structured information in the form of knowledge bases [43, 316] for visual recognition. The core of our algorithm consists of two modules: the local module, based on spatial memory [40], performs pixel-level reasoning using ConvNets. We make major improvements on efficiency by parallel memory updates. Additionally, we introduce a global module for reasoning beyond local regions. In the global module, reasoning is based on a *graph* structure. It has three components: a) a knowledge graph where we represent classes as nodes and build edges to encode different types of semantic relationships; b) a region graph of the current image where regions in the image are nodes and spatial relationships between these regions are edges; c) an assignment graph that assigns regions to classes. Taking advantage of such a structure, we develop a reasoning module specifically designed to pass information on this graph. Both the local module and the global module roll-out iteratively and cross-feed predictions to each other in order to refine estimates. Note that, local and global reasoning are not isolated: a good image understanding is usually a compromise between background knowledge learned *a priori* and image-specific observations. Therefore, our full pipeline joins force of the two modules by an attention [36] mechanism allowing the model to rely on the most relevant features when making the final predictions.

We show strong performance over plain ConvNets using our framework. For example, we can achieve 8.4% absolute improvements on ADE [313] measured by per-class average precision, where by simply making the network deeper can only help $\sim 1\%$. Code will be released.

8.2 Related Work

Visual Knowledge Base. Whereas past five years in computer vision will probably be remembered as the successful resurgence of neural networks, acquiring visual knowledge at a large scale – the simplest form being labeled instances of objects [166, 233], scenes [313], relationships [134] *etc.*– deserves at least half the credit, since ConvNets hinge on large datasets [266]. Apart from providing labels using crowd-sourcing, attempts have also been made to accumulate structured knowledge (*e.g.* relationships [43], n-grams [59]) automatically from the web. However, these works fixate on building knowledge bases rather than using knowledge for reasoning. Our framework, while being more general, is along the line of research that applies visual knowledge base to end tasks, such as affordances [316], image classification [182], or question answering [296].

Context Modeling. Modeling context, or the interplay between scenes, objects and parts is one of the central problems in computer vision. While various previous work (*e.g.* scene-level reasoning [274], attributes [71, 211], structured prediction [55, 133, 277], relationship graph [119, 170, 301]) has approached this problem from different angles, the breakthrough comes from the idea of feature learning with ConvNets [101]. On the surface, such models hardly use any explicit context module for reasoning, but it is generally accepted that ConvNets are extremely effective in aggregating local pixel-to-level context through its ever-growing receptive fields [310]. Even the most recent developments such as top-down module [165, 248, 299], pairwise module [237], iterative feedback [33, 196, 292], attention [305], and memory [40, 300] are motivated to leverage such power and depend on variants of convolutions for reasoning. Our work takes an important next step beyond those approaches in that it also incorporates learning from structured visual knowledge bases directly to reason with spatial and semantic relationships.

Relational Reasoning. The earliest form of reasoning in artificial intelligence dates back to symbolic approaches [197], where relations between abstract symbols are defined by the language of mathematics and logic, and reasoning takes place by deduction, abduction [105], *etc.* However, symbols need to be grounded [100] before such systems are practically useful. Modern approaches, such as path ranking algorithm [145], rely on statistical learning to extract useful patterns to perform relational reasoning on structured knowledge bases. As an active research area, there are recent works also applying neural networks to the graph structured data [50, 102, 128, 161, 182, 198, 239], or attempting to regularize the output of networks with relationships [54] and knowledge bases [113]. However, we believe for visual data, reasoning should be both local and global: discarding the two-dimensional image structure is neither efficient nor effective for tasks that involve regions.

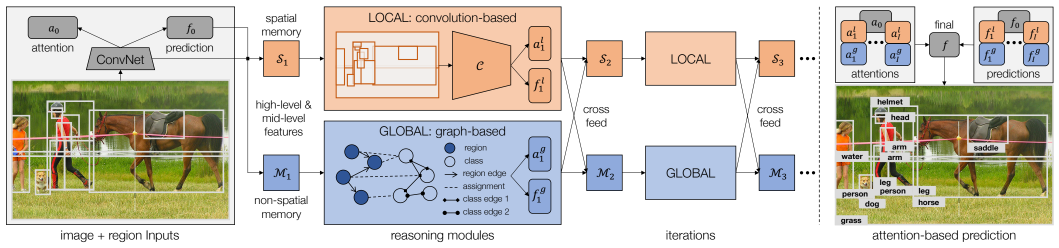


Figure 8.2: Overview of our reasoning framework. Besides a plain ConvNet that gives predictions, the framework has two modules to perform reasoning: a local one (Section 8.3.1) that uses spatial memory $\mathcal{S}_i$, and reasons with another ConvNet $\mathcal{C}$; and a global one (Section 8.3.2) that treats regions and classes as nodes in a graph and reasons by passing information among them. Both modules receive combined high-level and mid-level features, and roll-out iteratively (Section 8.3.3) while cross-feeding beliefs. The final prediction f is produced by combining all the predictions f_i with attentions a_i (Section 8.3.4).

8.3 Reasoning Framework

In this section we build up our reasoning framework. Besides plain predictions p_0 from a ConvNet, it consists of two core modules that reason to predict. The first one, local module, uses a spatial memory to store previous beliefs with parallel updates, and still falls within the regime of convolution based reasoning (Section 8.3.1). Beyond convolutions, we present our key contribution – a global module that reasons directly between regions and classes represented as nodes in a graph (Section 8.3.2). Both modules build up estimation iteratively (Section 8.3.3), with beliefs cross-fed to each other. Finally taking advantage of both local and global, we combine predictions from all iterations with an attention mechanism (Section 8.3.4) and train the model with sample re-weighting (Section 8.3.5) that focuses on hard examples (See Figure 8.2).

8.3.1 Reasoning with Convolutions

Our first building block, the local module, is inspired from [40]. At a high level, the idea is to use a spatial memory $\mathcal{S}$ to store previously detected objects at the very location they have been found. $\mathcal{S}$ is a tensor with three dimensions. The first two, height H and width W, correspond to the reduced size (1/16) of the image. The third one, depth D (=512), makes each cell of the memory c a vector that stores potentially useful information at that location.

$\mathcal{S}$ is updated with both high-level and mid-level features. For high-level, information regarding the estimated class label is stored. However, just knowing the class may not be ideal – more details about the shape, pose *etc*. can also be useful for other objects. For example, it would be nice to know the pose of a "person" playing tennis to recognize the "racket". In this paper, we use the logits f before soft-max activation, in conjunction with feature maps from a bottom convolutional layer h to feed-in the memory.

Given an image region r to update, we first crop the corresponding features from the bottom layer, and resize it to a predefined square (7×7) with bi-linear interpolation as h. Since high-level feature f is a vector covering the entire region, we append it to all the 49 locations. Two 1×1 convolutions are used to fuse the information [40] and form our input features f_r for r. The same region in the memory $\mathcal{S}$ is also cropped and resized to 7×7, denoted as s_r. After this alignment, we

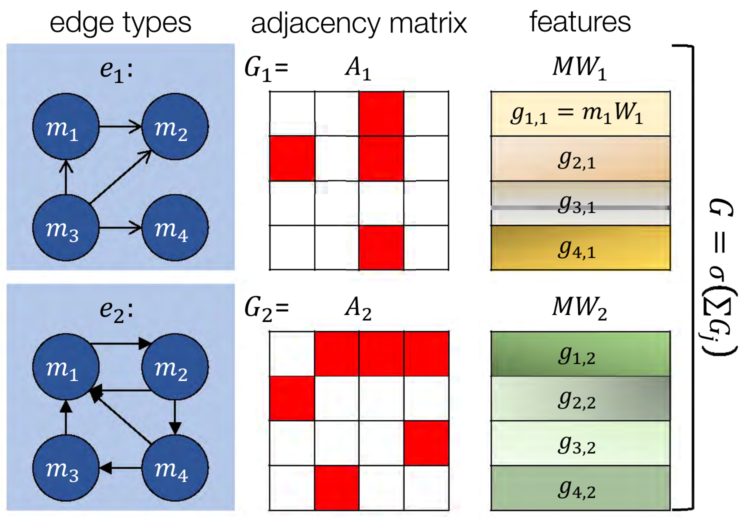


Figure 8.3: Illustration of directly passing information on a graph with multiple edge types. Here four nodes are linked with two edge types. Each node is represented as an input feature vector m_i (aggregated as M). Weight matrix W_j is learned for edge type j to transform inputs. Then adjacency matrix A_j is applied to pass information to linked nodes. Finally, output G is generated by accumulating all edge types and apply activation function.

use a convolutional gated recurrent unit (GRU) [45] to write the memory:

$$s'_r = u \circ s_r + (1 - u) \circ \sigma(W_f f_r + W_s(z \circ s_r) + b), \tag{8.1}$$

where s'_r is the updated memory for r, u is update gate, z is reset gate, W_f, W_s and b are convolutional weights and bias, and $\circ$ is entry-wise product. $\sigma(\cdot)$ is an activation function. After the update, s'_r is placed back to $\mathcal{S}$ with another crop and resize operation[1].

Parallel Updates. Previous work made sequential updates to memory. However, sequential inference is inefficient and GPU-intensive – limiting it to only give ten outputs per image [40]. We provide a major improvement to update the regions in parallel. In the case of overlapping, a cell can be covered multiple times from different regions. Therefore, when placing the regions back to $\mathcal{S}$, we also calculate a weight matrix Γ where each entry $\gamma_{r,c} \in [0, 1]$ keeps track of how much a region r has contributed to a memory cell c: 1 meaning the cell is fully covered by the region, 0 meaning not covered. The final values of the updated cell is the weighted average of all regions.

The actual reasoning module, a ConvNet $\mathcal{C}$ of three 3×3 convolutions and two 4096-D fully-connected layers, takes $\mathcal{S}$ as the input, and builds connections within the local window of its receptive fields to perform prediction. Since the two-dimensional image structure and the location information is preserved in $\mathcal{S}$, such an architecture is particularly useful for relationships with spatial reasoning.

[1]Different from Chapter 7 that introduces an inverse operation to put the region back, we note that crop and resize *itself* with proper extrapolation can simply meet this requirement.

8.3.2 Beyond Convolutions

Our second module goes beyond local regions and convolutions for global reasoning. Here the meaning of *global* is two-fold. First is *spatial*, that is, we want to let the regions farther away to directly communicate information with each other, not confined by the receptive fields of the reasoning module $\mathcal{C}$. Second is *semantic*, meaning we want to take advantage of visual knowledge bases, which can provide relationships between classes that are globally true (*i.e.* commonsense) across images. To achieve both types of reasoning, we build a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ and $\mathcal{E}$ denote node sets and edge sets, respectively. Two types of nodes are defined in $\mathcal{N}$: region nodes $\mathcal{N}_r$ for R regions, and class nodes $\mathcal{N}_c$ for C classes.

As for $\mathcal{E}$, three groups of edges are defined between nodes. First for $\mathcal{N}_r$, a spatial graph is used to encode spatial relationships between regions ($\mathcal{E}_{r\to r}$). Multiple types of edges are designed to characterize the relative locations. We begin with basic relationships such as "left/right", "top/bottom" and we define edge weights by measuring the pixel-level distances between the two. Note that we do not use the raw distance x directly, but instead normalizing it to $[0, 1]$ with a kernel $\kappa(x){=}\exp(-x/\Delta)$ (where $\Delta{=}50$ is the bandwidth), with the intuition that closer regions are more correlated. The edge weights are then used directly in the adjacency matrix of the graph. Additionally, we include edges to encode the coverage patterns (*e.g.* intersection over union, IoU [67]), which can be especially helpful when two regions overlap.

A second group of edges lie between regions and classes, where the assignment for a region to a class takes place. Such edges shoulder the responsibility of propagating beliefs from region to class ($e_{r\to c}$) or backwards from class to region ($e_{c\to r}$). Rather than only linking to the most confident class, we choose full soft-max score p to define the edge weights of connections to all classes. The hope that it can deliver more information and thus is more robust to false assignments.

Semantic relationships from knowledge bases are used to construct the third group of edges between classes ($\mathcal{E}_{c\to c}$). Again, multiple types of edges can be included here. Classical examples are "is-kind-of" (*e.g.* between "cake" and "food"), "is-part-of" (*e.g.* between "wheel" and "car"), "similarity" (*e.g.* between "leopard" and "cheetah"), many of which are universally true and are thus regarded as commonsense knowledge for humans. Such commonsense can be either manually listed [233] or automatically collected [43]. Interestingly, even relationships beyond these (*e.g.* actions, prepositions) can help recognition [182]. Take "person ride bike" as an example, which is apparently more of an image-specific relationship. However, given less confident predictions of "person" and "bike", knowing the relationship "ride" along with the spatial configurations of the two can also help prune other spurious explanations. To study both cases, we experimented with two knowledge graphs in this paper: one created in-house with mostly commonsense edges, and the other also includes more types of relationships accumulated at a large-scale. For the actual graphs used in our experiments, please see Section 8.4.1 for more details.

Now we are ready to describe the graph-based reasoning module $\mathcal{R}$. As the input to our graph, we use $M_r{\in}\mathbb{R}^{R\times D}$ to denote the features from all the region nodes $\mathcal{N}_r$ combined, where D (=512) is the number of feature channels. For each class node n_c, we choose off-the-shelf word vectors [122] as a convenient representation, denoted as $M_c{\in}\mathbb{R}^{C\times D}$. We then extend previous works [198, 239] and pass messages directly on $\mathcal{G}$ (See Figure 8.3). Note that, because our end-goal is to recognize regions better, all the class nodes should only be used as intermediate "hops" for better region representations. With this insight, we design two reasoning paths to learn the output features G_r: a *spatial* path on which only region nodes are involved:

$$G_r^{spatial} = \sum_{e\in\mathcal{E}_{r\to r}} A_e M_r W_e, \tag{8.2}$$

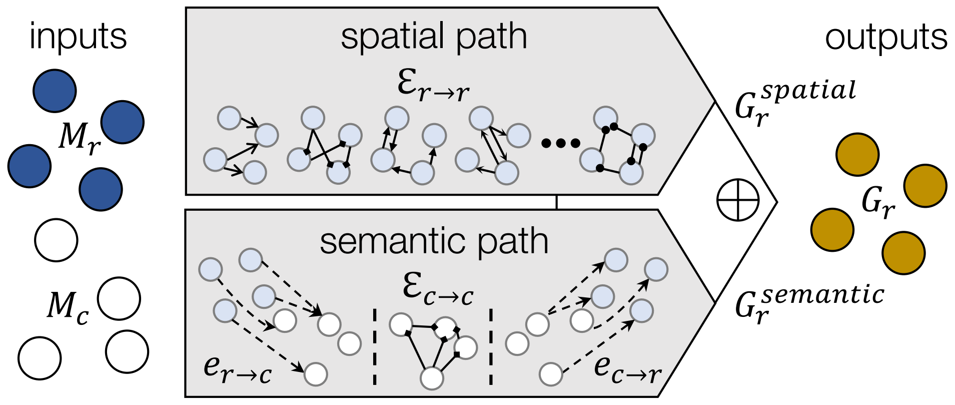


Figure 8.4: Two reasoning paths used in our global reasoning module $\mathcal{R}$. Taking the region and class inputs M_r and M_c, the spatial path directly passes information in the region graph with region-to-region edges $\mathcal{E}_{r\to r}$, whereas the semantic path first assigns regions to classes with $e_{r\to c}$, passes the information on to other classes with class-to-class edges $\mathcal{E}_{c\to c}$, and then propagates back. Final outputs are combined to generate output region features G_r.

where $A_e{\in}\mathbb{R}^{r\times r}$ is the adjacency matrix of edge type e, $W_e{\in}\mathbb{R}^{d\times d}$ is weight (bias is ignored for simplicity). The second reasoning path is a *semantic* one through class nodes:

$$G_c^{semantic} = \sum_{e\in\mathcal{E}_{c\to c}} A_e\sigma(A_{e_{r\to c}}M_rW_{e_{r\to c}} + M_cW_c)W_e, \tag{8.3}$$

where we first map regions to classes through $A_{e_{r\to c}}$ and $W_{e_{r\to c}}$, combine the intermediate features with class features M_c, and again aggregate features from multiple types of edges between classes. Finally, the output for regions G_r are computed by merging these two paths:

$$G_r = \sigma(G_r^{spatial} + \sigma(A_{e_{c\to r}}G_c^{semantic}W_{e_{c\to r}})), \tag{8.4}$$

which first propagates semantic information back to regions, and then applies non-linear activation (See Figure 8.4).

Just like convolution filters, the above-described paths can also be stacked, where the output G_r can go through another set of graph operations – allowing the framework to perform joint spatial-semantic reasoning with deeper features. We use three stacks of operations with residual connections [101] in $\mathcal{R}$, before the output is fed to predict.

8.3.3 Iterative Reasoning

A key ingredient of reasoning is to iteratively build up estimates. But how does information pass from one iteration to another? Our answer is *explicit* memory, which stores all the history from previous iterations. The local module uses spatial memory $\mathcal{S}$, and the global module uses another memory $\mathcal{M}$ but without spatial structures. At iteration i, $\mathcal{S}_i$ is followed by convolutional reasoning module $\mathcal{C}$ to generate new predictions f_i^l for each region. Similarly, global module also gives new predictions f_i^g from $\mathcal{R}$. These new predictions as high-level features can then be used to get the updated memories $\mathcal{S}_{i+1}$ and $\mathcal{M}_{i+1}$. The new memories will lead to another round of updated f_{i+1}s and the iteration goes on.

While one can do local and global reasoning in isolation, both the modules work best in conjunction. Therefore, for our full pipeline we want to join force of both modules when generating the predictions. To this end, we introduce *cross-feed* connections. After reasoning, both the local and global features are then concatenated together to update the memories $\mathcal{S}_{i+1}$ and $\mathcal{M}_{i+1}$ using GRU. In this way, spatial memory can benefit from global knowledge of spatial and semantic relationships, and graph can get a better sense of the local region layouts.

8.3.4 Attention

Inspired from the recent work on attention [36], we make another modification at the model output. Specifically, instead of only generating scores f, the model also has to produce an "attention" value a that denotes the relative confidence of the current prediction compared to the ones from other iterations or modules. Then the fused output is a weighted version of all predictions using attentions. Mathematically, if the model roll-outs I times, and outputs $N{=}2I{+}1$ (including I local, I global and 1 from plain ConvNet) predictions f_n, using attentions a_n, the final output f is calculated as:

$$f = \sum_n w_n f_n, \quad \text{where} \quad w_n = \frac{\exp(-a_n)}{\sum_{n'} \exp(-a_{n'})}. \tag{8.5}$$

Note again that here f_n is the logits before soft-max, which is then activated to produce p_n. The introduction of attention allows the model to intelligently choose feasible predictions from different modules and iterations.

8.3.5 Training

Finally, the overall framework is trained end-to-end, with a total loss function consists of: a) plain ConvNet loss $\mathcal{L}_0$; b) local module loss $\mathcal{L}_i^l$; c) global module loss $\mathcal{L}_i^g$; and d) the final prediction loss with attentions $\mathcal{L}_f$.

Since we want our reasoning modules to focus more on the harder examples, we propose to simply *re-weight* the examples in the loss, based on predictions from previous iterations. Formally, for region r at iteration $i{\geq}1$, the cross-entropy loss for both modules is computed as:

$$\mathcal{L}_i(r) = \frac{\max(1. - p_{i-1}(r), \beta)}{\sum_{r'} \max(1. - p_{i-1}(r'), \beta)} \log(p_i(r)), \tag{8.6}$$

where $p_i(r)$ is the soft-max output of the ground-truth class, and $\beta{\in}[0,1]$ controls the entropy of the weight distribution: when $\beta{=}1$, it is uniform distribution; and when $\beta{=}0$, entropy is minimized. In our experiments, β is set to 0.5. $p_{i-1}(r)$ is used as features without back-propagation. For both local and global, $p_0(r)$ is the output from the plain ConvNet.

8.4 Experiments

In this section we evaluate the effectiveness of our framework. We begin with our experimental setups, which includes the datasets to work with (Section 8.4.1), the task to evaluate on (Section 8.4.2) and details of our implementation (Section 8.4.3). We discuss our results and analyze them in Section 8.4.4 and Section 8.4.5 respectively.

8.4.1 Datasets and Graphs

Datasets are biased [273]. For context reasoning we would naturally like to have scene-focused datasets [313] as opposed to object-focused ones [233]. To showcase the capabilities of our system, we need densely labeled dataset with a large number of classes. Finally, one benefit of using knowledge graph is to transfer across classes, therefore a dataset with *long-tail* distribution is an ideal test-bed. Satisfying all these constraints, ADE [313] and Visual Genome (VG) [134] where regions are densely labeled in open vocabulary are the main picks of our study.

For ADE, we use the publicly released training set $(20,210)$ images for training, and split the validation set $(2,000$ images) into `val-1k` and `test-1k` with $1,000$ images each. The original raw names are used due to a more detailed categorization [313]. We filter out classes with less than five instances, which leaves us with $1,484$ classes. With the help of parts annotations in the dataset, a commonsense knowledge graph is created with five types of edges between classes: a) "is-part-of" (*e.g.* "leg" and "chair"); b) "is-kind-of" (*e.g.* "jacket" and "clothes"); c) "plural-form" (*e.g.* "tree" and "trees"); d) "horizontal-symmetry" (*e.g.* "left-arm" and "right-arm"); e) "similarity" (*e.g.* "handle" and "knob"). Notice that the first four types are directed edges, hence we also include their inverted versions.

For VG, the latest release (v1.4) is used. We split the entire set of $108,077$ images into 100K, $4,077$ and 4K as `train`, `val` and `test` set. Similar pre-processing is done on VG, except that we use synsets [233] instead of raw names due to less consistent labels from multiple annotators. $3,993$ classes are used. For knowledge graph between classes, we take advantage of the relationship annotations in the set, and select the top 10 most frequent relationships to automatically construct edges beyond commonsense relationships constructed for ADE. For each type of relationships, the edge weights are normalized so that each row of the adjacency matrix is summed-up to one. While this approach results in a noisier graph, it also allows us to demonstrate that our approach is scalable and robust to noise.

Finally, we also show experiments on COCO [166]. However, since it is detection oriented – has only 80 classes picked to be mutually-exclusive, and covers less percentage of labeled pixels, we only report results a) without the knowledge graph and b) without a test split (`trainval35k` [40] for training and `minival` for evaluation). This setup is for analysis purposes only.

8.4.2 Task and Evaluation

We evaluate our system on the task of region classification, where the goal is to assign labels to designated regions denoted by rectangular bounding boxes. For both training and testing, we use provided ground-truth locations. We picked this task for three reasons. The **first** one is on evaluation. As the number of classes increases in the vocabulary, *missing* labels are inevitable, which is especially severe for object parts (*e.g.* "rim", "arm") and related classes (*e.g.* "shoes" *vs.* "sneakers") where external knowledge is valuable. If there are missing labels, fair evaluation becomes much more difficult since accuracy becomes impossible to evaluate – cannot tell if a prediction is wrong, or the label itself is missing. Interestingly, such an issue also happens to other research areas (*e.g.* recommendation systems [238] and link prediction [163]). Borrowing ideas from them, a practical solution is to evaluate *only* on what we already know – in our case ground-truth regions. **Second**, although region classification is a simplified version of object detection and semantic segmentation, it maintains a richer set of labels, especially including "stuff" classes like "road", "sky", and object instances. Modeling "stuff-object" and instance-level relationships is a crucial capability which would be missed in a pure detection/segmentation setting. **Finally** as our experiment will show (Section 8.4.5), while object detectors can be used off-the-shelf, the additional manually de-

Table 8.1: Main results on ADE `test-1k` and VG `test`. AP is average precision, AC is classification accuracy. Superscripts show the improvement ∇ over the baseline.

%	**Method**	per-instance		per-class	
		AP$^{\nabla}$	AC$^{\nabla}$	AP$^{\nabla}$	AC$^{\nabla}$
ADE	Baseline	67.0	67.0	40.1	33.2
	w/ ResNet-101	68.2	68.3	40.8	34.4
	w/ 800-input	68.2	68.2	41.0	34.3
	Ensemble	68.7	68.8	42.9	35.3
	Ours$_{\text{-Local}}$	71.6$^{+4.6}$	71.7$^{+4.7}$	47.9$^{+7.8}$	38.7$^{+5.7}$
	Ours$_{\text{-Global}}$	69.8$^{+2.8}$	69.8$^{+2.8}$	44.5$^{+4.4}$	36.8$^{+3.6}$
	Ours$_{\text{-Final}}$	**72.6**$^{+5.6}$	**72.6**$^{+5.6}$	**48.5**$^{+8.4}$	**39.5**$^{+6.3}$
VG	Baseline	49.1	49.6	16.9	12.1
	w/ ResNet-101	50.3	50.8	18.0	**13.0**
	w/ 800-input	49.5	50.0	17.0	12.2
	w/ Ensemble	50.2	50.7	17.7	12.3
	Ours$_{\text{-Local}}$	51.4$^{+2.3}$	51.9$^{+2.3}$	18.8$^{+1.9}$	12.8$^{+0.7}$
	Ours$_{\text{-Global}}$	50.9$^{+1.8}$	51.5$^{+1.9}$	18.3$^{+1.4}$	12.6$^{+0.5}$
	Ours$_{\text{-Final}}$	**51.7**$^{+2.6}$	**52.2**$^{+2.6}$	**19.1**$^{+2.2}$	12.9$^{+0.8}$

fined parameters and components (*e.g*. overlapping threshold for a region to be positive/negative, predefined scale/aspect ratio sets of anchors [226]) in its pipeline pose limitations on how much context can benefit. For example, after non-maximal suppression (NMS), highly overlapping objects (*e.g*. "window" and "shutter") will be suppressed, and ironically this is exactly where context reasoning could have helped. On the other hand, by feeding fixed regions directly for end-to-end learning, we can at least factorize the *recognition* error from the *localization* one [108], and get a clean focus on how context can help discriminating confusing classes.

Since ADE is a segmentation dataset, we convert segmentation masks to bounding boxes. For object classes (*e.g*. "person"), each instance is created a separate box. Part (*e.g*. "head") and part-of-part (*e.g*. "nose") are also included. For VG and COCO, boxes are directly used.

For evaluation, we use classification accuracy (AC) and average precision (AP) [67]. Note that since all the regions are fixed with known labels, there is no need to set a region overlap threshold for AP. Results can be aggregated in two ways: the first way ("per-class") computes metrics separately for each class in the set, and take the mean; since the final scores are all taken from a calibrated softmax output, a second way ("per-instance") that computes metrics simultaneously for all classes. Intuitively, "per-class" assigns more weights to instances from rare classes.

8.4.3 Implementation Details

A simplified version of *tf-faster-rcnn*[2] is used to implement our baseline for region classification, with region proposal branch and bounding box regression components removed. Unless otherwise noted, ResNet-50 [101] pre-trained on ImageNet [233] is used as our backbone image classifier, and images are enlarged to shorter size 600 pixels during both training and testing. Specifically, full-

[2]`https://github.com/endernewton/tf-faster-rcnn`

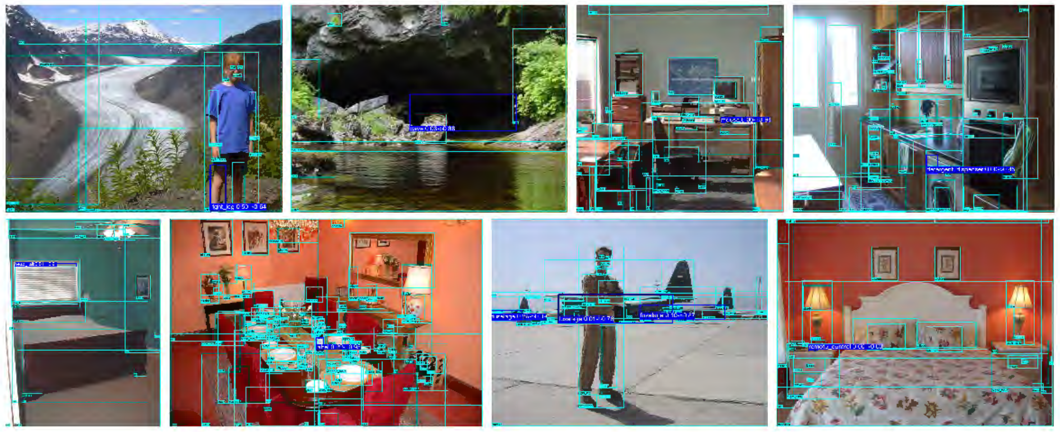


Figure 8.5: Qualitative examples from ADE `test-1k` (best if zoomed-in). For regions highlighted in blue, the predictions from baseline and our model are compared. Other regions are also listed to provide the context. For example, the "right-leg" is less confused with "left-leg" after reasoning (top-left); the "mouse" on the "desk" is predicted despite low resolution (top-third); and "detergent-dispenser" is recognized given the context of "washing-machine" (top-right). At bottom-right we show a failure case where context does not help "remote-control", probably because it has never appeared on the "night-table" before.

image shared convolutional feature maps are computed till the last *conv4* layer. Then the ground-truth boxes are used as regions-of-interest to compute region-specific features (crop and resize to 7×7 without max-pool). All layers of *conv5* and up are then adopted to obtain the final feature for the baseline prediction p_0. Batch normalization parameters are fixed.

For the local module, we use the last *conv4* layer as our mid-level features to feed the spatial memory $\mathcal{S}$. For the global module, mid-level features are the final *conv5* (2048-D) layer after avg-pool. Both features are fused with the logits before soft-max f, and then fed into the memory cells. Word vectors from fastText [122] are used to represent each class, which extracts sub-word information and generalizes well to out-of-vocabulary words. ReLU is selected as the activation function. We roll-out the reasoning modules 3 times and concurrently update all regions at each iteration, as more iterations do not offer more help.

We apply stochastic gradient descent with momentum to optimize all the models, and use the validation set to tune hyper-parameters. Our final setups are: $5e^{-4}$ as the initial learning rate, reduced once $(0.1\times)$ during fine-tuning; $1e^{-4}$ as weight decay; 0.9 as momentum. For ADE, we train 320K iterations and reduce learning rate at 280K. For VG and COCO the numbers are 640K/500K and 560K/320K, respectively[3]. We use a single image per step, and the only data augmentation technique used during training is left-right flipping[4]. No augmentation is used in testing.

8.4.4 Main Results

Quantitative results on ADE `test-1k` and VG `test` are shown in Table 8.1. Besides plain ConvNet p_0, we also add three more baselines. First, we use ResNet-101 as the backbone to see the performance can benefit from deeper networks. Second, we increase the input image size with a

[3]Training longer still reduces the cross-entropy loss on the validation set, but drops both AP and AC.

[4]The labels for class pairs like "left-hand" and "right-hand" are swapped for flipped images.

Table 8.2: Ablative analysis on ADE `test-1k`. In the first row of each block we repeat Local, Global and Final results from Table 8.1. Others see Section 8.4.5 for details.

%	**Analysis**	per-instance		per-class	
		AP	AC	AP	AC
Local	Ours$_{\text{-Local}}$	71.6	71.7	47.9	38.7
	w/o re-weight	71.3	71.3	46.7	37.9
	w/o $\mathcal{C}$	70.9	71.0	46.1	37.5
	w/o $\mathcal{S}$	67.6	67.6	42.1	34.4
Global	Ours$_{\text{-Global}}$	69.8	69.8	44.5	36.8
	w/o re-weight	69.2	69.2	43.8	36.7
	w/o spatial	67.8	67.8	41.5	35.0
	w/o semantic	69.1	69.2	43.9	35.9
	w/o $\mathcal{R}$	67.1	67.2	41.5	34.5
	w/o $\mathcal{M}$ & $\mathcal{R}$	67.1	67.1	41.0	34.0
Final	Ours$_{\text{-Final}}$	72.6	72.6	48.5	39.5
	w/o re-weight	72.1	72.2	47.3	38.6
	w/o cross-feed	72.2	72.2	47.6	39.0
	2 iterations	71.9	72.0	48.1	39.0

shorter side 800 pixels, which is shown helpful especially for small objects in context [165]. Finally, to check whether our performance gain is a result of more parameters, we include model ensemble as the third baseline where the prediction of two separate baseline models are averaged.

As can be seen, our reasoning modules are performing much better than all the baselines on ADE. The local module alone can increase per-class AP by 7.8 absolute points. Although the global module alone is not as effective (4.4% improvement), the performance gain it offers is *complementary* to the local module, and combining both modules we arrive at an AP of 48.5% compared to the baseline AP 40.1%. On the other hand, deeper network and larger input size can only help $\sim$1%, less than model ensembles. Additionally, our models achieve higher per-class metric gains than per-instance ones, indicating that *rare* classes get helped more – a nice property for learning from few examples. Some qualitative results are listed in Figure 8.5.

We see a similar but less significant trend on VG. This can potentially be a result of *noisier* labels – for ADE (and COCO shown later), the per-instance AP and AC values are within 0.1%, intuitively suggesting that *higher* scores usually correspond to correct classifications. However, on VG the difference is at $\sim$0.5%, meaning more of the highly confident predictions are not classified right, which are likely caused by incorrect ground-truths.

8.4.5 Analysis

Our analysis is divided into two major parts. In the first part, we conduct thorough ablative analysis on the framework we have built. We use ADE as an example, see Table 8.2.

As can be seen, re-weighting hard examples with Eq. 8.6 helps around 0.5% regardless of reasoning modules. Spatial memory $\mathcal{S}$ is critical in the local module – if replaced by feeding last *conv4* layer directly the performance drops almost to baseline. Local conetxt aggregator $\mathcal{C}$ is less influential

Table 8.3: Results with missing regions when region proposals are used. COCO `minival` is used since it is more detection oriented. **pre** filters regions before inference, and **post** filters after inference.

Method	**pre**	**post**	per-instance		per-class	
			AP^∇	AC^∇	AP^∇	AC^∇
Baseline			83.2	83.2	83.7	75.9
$\text{Ours}_{\text{-Local}}$			$84.9^{+1.7}$	$84.9^{+1.7}$	$85.8^{+2.1}$	$77.6^{+1.7}$
$\text{Ours}_{\text{-Global}}$			$85.6^{+2.4}$	$85.7^{+2.5}$	$86.9^{+3.2}$	$78.2^{+2.3}$
$\text{Ours}_{\text{-Final}}$			$\mathbf{86.0}^{+2.8}$	$\mathbf{86.0}^{+2.8}$	$\mathbf{87.4}^{+3.7}$	$\mathbf{79.0}^{+3.1}$
Baseline	-	-	87.0	87.0	87.7	80.2
$\text{Ours}_{\text{-Final}}$	✓		$88.6^{+1.6}$	$88.6^{+1.6}$	$89.9^{+2.2}$	$\mathbf{82.6}^{+2.4}$
$\text{Ours}_{\text{-Final}}$		✓	$\mathbf{88.8}^{+1.8}$	$\mathbf{88.8}^{+1.8}$	$\mathbf{90.1}^{+2.4}$	$82.5^{+2.3}$

for ADE since the regions including background are densely labeled. A different story takes place at the global module: removing the reasoning module $\mathcal{R}$ steeply drops performance, whereas further removing memory $\mathcal{M}$ does not hurt much. Finally, for our full pipeline, removing cross-feeding and dropping the number of iterations both result in worse performance.

8.4.6 Missing Regions

So far we have shown results when all the regions are present. Next, we want to analyze if our framework is robust to missing regions: if some percentage of regions are not used for reasoning. This will be a common scenario if we use our framework in the detection setting – the underlying region proposal network [226] may itself miss some regions. We perform this set of experiments on COCO, since its regions are object-focused.

We test three variations. In the first variation, the same region classification pipeline is applied as-is. In the other two, we drop regions. While we could have done it randomly, we simulate the real-world scenario by using region proposals from faster R-CNN [226] (1190K/900K, `minival` detection mAP 32.4%) for testing, where 300 region proposals after NMS are applied to filter the ground-truth regions (max IoU>δ). Evaluation is only done on the remaining regions. Here we choose not to use region proposals directly, since the model has seen ground truth regions only. We test two variations: a) "pre", where the regions are filtered before inference, *i.e.* only the remaining ground-truths are fed for reasoning; "post", where regions are filtered after inference. Note that for the baseline, "pre" and "post" makes no difference performance-wise.

The results are summarized in Table 8.3. Interestingly, despite lacking a knowledge graph, our global module works better than the local module, likely due to its power that allows direct region-to-region communication even for farther-away pairs. Combining the two, we report 3.7% absolute advantage on per-class AP over the baseline even with all classes being objects – no "stuff" classes involved.

In Figure 8.6, we vary δ from 0 to .9: with 0 keeping all regions and 0.9 dropping the most. As the trend shows, while the reasoning module suffers when regions are dropped, it is quiet resilient and the performance degradation is smooth. For example (listed in Table 8.3), with an IoU threshold δ of 0.5 that recalls 78.1% of the ground truth boxes, we still outperform the baseline by 2.4% in the "post" setting, and 2.2% in "pre" where not all regions can be fed for reasoning. The lower gap implies a) region proposals are usually corresponding to easy examples where less context is needed,

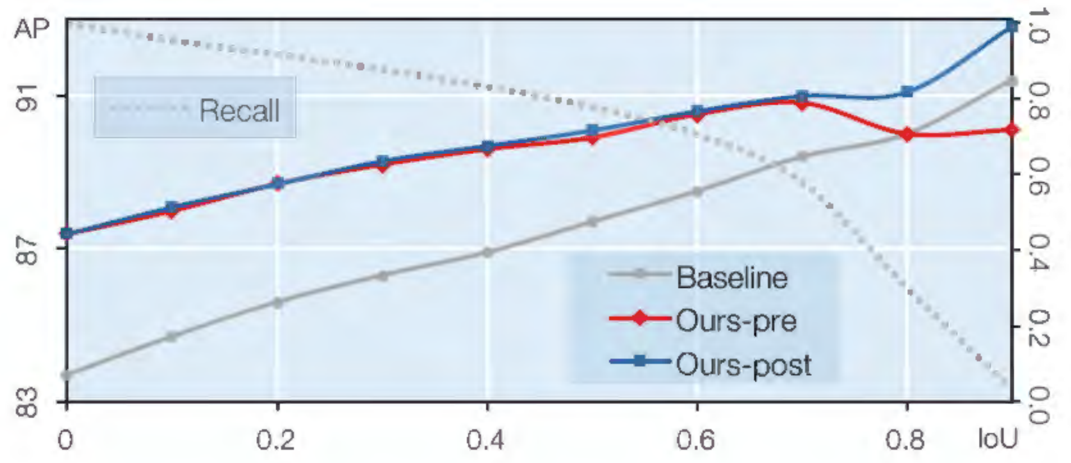


Figure 8.6: Trends of recall and per-class AP when varying IoU threshold δ from 0 to .9 to drop regions. See text for details.

and b) context reasoning frameworks like ours benefit from more known regions. At δ=.8 the recall (30.5%) is so small that it cannot afford much reasoning, and at δ=.9 (recall 3.9%), reasoning even hurts the performance.

Chapter 9

Conclusion & Discussion

This Ph.D. thesis has made progress toward solving two critical issues associated with traditional visual knowledge. First, in terms of *learning* visual knowledge, how can we let machines accumulate instance labels and category relationships automatically at a large scale? Our approach was resorting to the Internet, with the hope that existing image search engines can provide enough support to bootstrap a program that learns both labels and relationships simultaneously. Second, concerning *using* knowledge effectively, how can we empower computers the ability to perform reasoning as human beings do? As a potential answer, we presented a framwework that combines the powers of both explicit, structured knowledge and implicit, contextual knowledge; and demonstrated its usefulness and robustness for visual recognition tasks.

However, as the name of our "never-ending" system suggests, this thesis is not "the end", or not even "the beginning of the end"[1]. Therefore, we believe it would be beneficial to give summarizations to the observations, speculations, lessons and potential future directions for works in this dissertation. For clarity, we index them with questions and group them in sections below.

9.1 Relationships

Modeling Implicit Relationships? From a different point of view, the work of spatial memory [40] is a direct extension (and in some sense a completion) of NEIL, as the latter tries to model explicit, structured relationships between concepts (*e.g.* `Corolla is a kind of car`); whereas the spatial memory desires to model implicit, contextual relationships that can be sometimes hard to express in triplets or even language (*e.g.* finding `traffic lights` given `road`, `car`, `pedestrian` *etc*.). One intriguing direction is to see if spatial memory or its variants can be used to model more complex relationships beyond the few ones in NEIL. For example ones that require both spatial and semantic reasoning to understand, *e.g.*`ride` or `play`.

Relationships Covered in Spatial Memory? Following the previous question, one may ask whether we need to use background knowledge in the form of relationships *at all*, as spatial memory is already a pretty generalized representation by itself. However, despite its great potential to cover various types of relationships, we should point out that it also has drawbacks, at least of the time being. For instance, in Chapter 8 we find experimentally that on COCO [166], a region graph that directly models spatial relationships with edges, though coarse, is able to outperform spatial memory with convolution based reasoning. This is in big contrast to the results on ADE [313], where spatial

[1]But, it perhaps remarks "the end of the beginning" —WINSTON CHURCHILL, 1942.

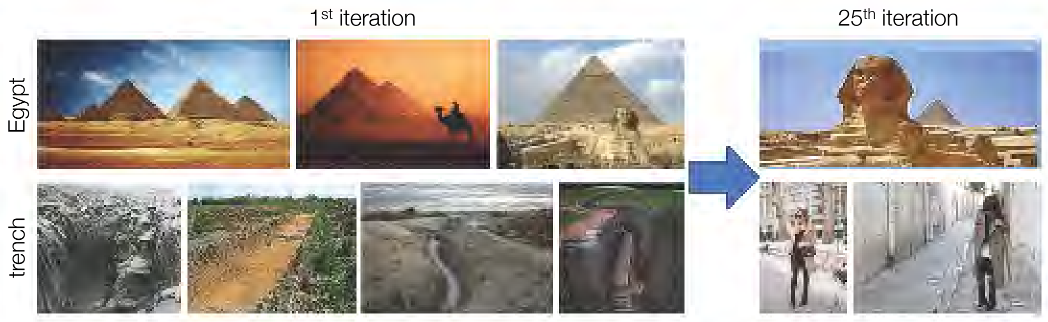


Figure 9.1: We show one positive, and one negative example for the role of relationships in NEIL.

memory exceeds graph message passing in performance with a sizable margin. The most probable reason is that: In order for ConvNet to conduct reasoning on spatial memory, it requires a large underlying receptive field and expensive layer-wise computation for the information to propagate. COCO is a detection dataset, where a certain portion of the pixels are not labeled, while ADE also contains stuff categories that cover the (sometimes vast) empty spaces between objects. As a result, spatial memory can take advantage of the neighboring pixels to help recognition on ADE, but fails to use regions far-apart and gets overtaken by the graph based module.

Benefits of Relationships? We show two types of benefits from relationships in our thesis. First, we show that modeling relationships helps constrain the learning process. NEIL [43] shows that experimentally that relationships can help both scene classification and object detection in the first few iterations. In Figure 9.1, we additionally show two examples of the role of relationships in NEIL. For example, knowing `Sphinx is found in Egypt` helps NEIL generalize healthily examples of `Egypt` to zoomed-in images of `Sphinx`; on the other hand, relating `trench` to `person` (because of the coat style) resulted in `trench` drifting its original meaning of a long ditch on the ground. The same constraining effect is found when we train a ConvNet from the web [38], where having the relationship graph encoding a higher-order statistics is shown to be preventing the network from over-fitting its representation to the noise when fine-tuned on Flickr data. Second, we show that incorporating the background knowledge graph is useful in our reasoning framework, offering a complementary source of information to hinge on.

Other Types of Relationships? Both NEIL and spatial memory network have been focusing on *spatial* and *semantic* relationships only. However, such relationships do not cover all visual knowledge that relationships can model. Here we list a few more possibilities:

- *Physical* relationships. These are relationships that deal with physical properties of categories. For example, typical height, weight of a `person` can be used to measure how far he or she is from the camera. Indeed, there are works [7] that learn such knowledge from the Web. Note that certain physical properties can already be represented by pixels (*e.g.* color, albedo, *etc.*).

- *Functional* relationships. The biggest difference between modern humans and other species is the ability of *changing* the environment [261]. While most animals can only rely on sharp teeth or claws to survive, humans can build tools and use them even if evolutionary not able to achieve certain goals. As a result, many man-made objects are designed to serve the need of humans – a property we call functionality. For instance, `chairs` are shaped with a flat surface (and seat back/arm rest) to support the `sitting` pose of a person. Such a connection can be modeled by relationships and help machines better understand images, in particular with man-made categories.

- *Temporal/dynamic* relationships. So far, all we have talked about is limited to static scenes. However, our visual world comes with a fourth dimension – time. Many semantic concepts also have a temporal aspect or even story-lines [250] associated with them. For example, a visual representation of `wedding` would involve guests entering the venue, followed by exchange of rings and finally celebrations in the wedding reception. So far, such aspects can only be learned with videos or photo albums, which pose many challenges to even process the data (*e.g.* they can have extremely high short-term correlations, and require expensive computation resources). Modeling these aspects, and in particular understanding the causality and correlation between events in sequence better, is probably the next big thing in the field. There are recent prominent works in this direction with graph structures [116].

9.2 Visual Knowledge

Explicit *vs*. Implicit Visual Knowledge? Having put forward the notion of explicit visual knowledge and implicit one, we would like to give a summary of the pros and cons. We characterize their relationship as mutually-beneficial and mutually-dependent. Implicit knowledge targets at usefulness, as it works well not only on the task it is being trained for, but also showcases strong performances on other tasks when transferred. However, implicit knowledge as of not requires large amounts of training examples – a condition not satisfied always. Explicit knowledge, on the other hand, can offer clear model explainability help when few examples are available. It is also important to note that currently implicit knowledge cannot live without the supervision provided by the simplest form of explicit knowledge – labels, as unsupervised or self-supervised approaches are still lagging behind. As a side note, it is also a good future work to focus on how we can converge deep models with structured supervision, which is not only more informative (triplets rather than just a single label), but also more natural in light of how humans learn visual concepts.

Other Visual Knowledge Beyond Relationships? Lots of useful visual knowledge is closely related to the geometrical or physical laws of our visual world. However, encoding, for example, `Newton's second law` $F{=}ma$ or `perspective projection` within a knowledge base is by itself a challenging task, not to mention how we can use it to track a 3D object projected on 2D frames for understanding videos. Therefore, we haven't touched this part in this thesis. Nevertheless, it is definitely an interesting and fruitful direction to check in the future, as right now ConvNets are believed to be mostly doing pattern recognition with their universal function approximation power. If current models could be granted the ability to really understand the mathematical and physical regularities, and efficiently use them whenever needed, it would greatly advance the intelligence of machines, since such knowledge is not "yet another type", but a particular type that can leverage the power of *computation* – one thing that present machines marvel at.

9.3 Reasoning Methodology

More Practical Task? Although we provided convincing reasons in Section 8.4.2, the final evaluation is still done on a very constrained setting – ground truth regions are provided in both training and testing for region classification. For this reason, a natural next step would be to test on more practical tasks. There is some most recent development [129] that intends to unify instance segmentation for objects and semantic segmentation for stuffs. It would be nice to benchmark reasoning frameworks on such tasks.

Top-Down *vs*. Bottom-Up? We show in our segmentation work [44] that by fuse bottom-up approaches with top-down information (from the detector), our approach can achieve impressive results discovering relevant pixels within noisy web images. Later in both our spatial memory network and iterative reasoning framework, we have implicitly consolidated top-down information (detection or prediction results) and bottom-up representations (ConvNet features) together. All these experiments suggest that image understanding improves with both top-down and bottom-up reasoning, regardless of the architecture used. This is especially true when we are faced with the task of finding small objects or recognizing at the pixel-level, where both "what" and "where" are essential.

End-to-End Learning? One lesson the entire community has learned with the recent transition from manually designed features to learned representations is the critical idea of end-to-end learning [136]. While showing promises, NEIL and the other works in Part I also cannot escape the sweep of ConvNets. One possible future direction is to also make NEIL end-to-end. In particular, the original NEIL system only finds relationships based on co-occuring statistics, which likely leads to sets of relationships not ideal for the end-goal. However, designing unbiased and multi-functioning objective functions is not trivial, notably for a general-purpose system like NEIL.

Bibliography

[1] P. Agrawal, R. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*, 2014. 50, 51

[2] B. Alexe, T. Deselares, and V. Ferrari. What is an object? In *TPAMI*, 2010. 16, 20

[3] B. Alexe, T. Deselaers, and V. Ferrari. Classcut for unsupervised class segmentation. In *ECCV*, 2010. 18, 20

[4] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *TPAMI*, 2012. 71

[5] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Learning to compose neural networks for question answering. *arXiv:1601.01705*, 2016. 69

[6] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh. Vqa: Visual question answering. In *ICCV*, 2015. 69, 70

[7] H. Bagherinezhad, H. Hajishirzi, Y. Choi, and A. Farhadi. Are elephants bigger than butterflies? reasoning about sizes of objects. In *AAAI*, 2016. 100

[8] S. Banerjee and A. Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL Workshop*, 2005. 57, 61, 62, 64

[9] A. Bansal, X. Chen, B. Russell, A. Gupta, and D. Ramanan. Pixelnet: Towards a general pixel-level architecture. *arXiv preprint arXiv:1609.06694*, 2016. 85

[10] M. Bar. Visual objects in context. *Nature Reviews Neuroscience*, 2004. 68

[11] A. Barla, F. Odone, and A. Verri. Histogram intersection kernel for image classification. In *ICIP*, 2003. 33

[12] K. Barnard and M. Johnson. Word sense disambiguation with pictures. In *AI*, 2005. 30

[13] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. iCoseg: Interactive co-segmentation with intelligent scribble guidance. In *CVPR*, 2010. 20, 26

[14] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *CVPR*, 2016. 68, 76, 78

[15] M. Bellver, X. Giró-i Nieto, F. Marqués, and J. Torres. Hierarchical object detection with deep reinforcement learning. *arXiv:1611.03718*, 2016. 70

[16] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, 2015. 75

[17] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, 2009. 4, 45, 46, 78

[18] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*. 2006. 58

[19] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *TNN*, 1994. 57, 58, 59

[20] T. L. Berg and A. C. Berg. Finding iconic images. In *CVPR Workshop*, 2009. 30, 43, 46

[21] T. L. Berg and D. A. Forsyth. Animals on the web. In *CVPR*, 2006. 9, 30, 42, 45

[22] A. Bergamo, L. Bazzani, D. Anguelov, and L. Torresani. Self-taught object localization with deep networks. *arXiv:1409.3964*, 2014. 45, 48

[23] A. Bergamo and L. Torresani. Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach. In *NIPS*, 2010. 45

[24] I. Biederman, R. J. Mezzanotte, and J. C. Rabinowitz. Scene perception: Detecting and judging objects undergoing relational violations. *Cognitive psychology*, 1982. 85

[25] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 2003. 32

[26] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, 1998. 9

[27] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *ICCV*, 2001. 20

[28] S. Brody and M. Lapata. Bayesian word sense induction. In *EACL*, 2009. 30

[29] J. C. Caicedo and S. Lazebnik. Active object localization with deep reinforcement learning. In *ICCV*, 2015. 70

[30] P. Carbonetto, N. De Freitas, and K. Barnard. A statistical model for general contextual object recognition. In *ECCV*, 2004. 68

[31] A. Carlson, J. Betteridge, E. R. H. Jr., and T. M. Mitchell. Coupling semi-supervised learning of categories and relations. In *NAACL Workskop*, 2009. 9

[32] 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. 8, 9, 15, 28

[33] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. In *CVPR*, 2016. 87

[34] P. Carruthers and P. K. Smith. *Theories of theories of mind*. 1996. 47

[35] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv:1606.00915*, 2016. 4, 68

[36] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *CVPR*, 2016. 87, 92

[37] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollr, and C. L. Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015. 57, 62, 64

[38] X. Chen and A. Gupta. Webly supervised learning of convolutional networks. In *ICCV*, 2015. 4, 100

[39] X. Chen and A. Gupta. An implementation of faster rcnn with study for region sampling. *arXiv:1702.02138*, 2017. 72, 79, 80, 82

[40] X. Chen and A. Gupta. Spatial memory for context reasoning in object detection. *arXiv preprint arXiv:1704.04224*, 2017. 5, 85, 86, 87, 88, 89, 93, 99

[41] X. Chen and C. Lawrence Zitnick. Mind's eye: A recurrent visual representation for image caption generation. In *CVPR*, 2015. 4, 69, 78

[42] X. Chen, A. Ritter, A. Gupta, and T. Mitchell. Sense discovery via co-clustering on images and text. In *CVPR*, 2015. 3

[43] X. Chen, A. Shrivastava, and A. Gupta. NEIL: Extracting visual knowledge from web data. In *ICCV*, 2013. 3, 4, 18, 20, 21, 22, 26, 27, 28, 30, 33, 36, 40, 42, 43, 45, 46, 47, 48, 49, 53, 86, 87, 90, 100

[44] X. Chen, A. Shrivastava, and A. Gupta. Enriching visual knowledge bases via object discovery and segmentation. In *CVPR*, 2014. 3, 4, 33, 38, 42, 101

[45] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555*, 2014. 73, 74, 89

[46] D. J. Crandall and D. P. Huttenlocher. Weakly supervised learning of part-based spatial models for visual object recognition. In *ECCV*, 2006. 45

[47] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP*, 2007. 28

[48] J. R. Curran, T. Murphy, and B. Scholz. Minimising semantic drift with mutual exclusion bootstrapping. In *Pacific Association for Computational Linguistics*, 2007. 9

[49] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 15, 33

[50] R. Das, A. Neelakantan, D. Belanger, and A. McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. *arXiv preprint arXiv:1607.01426*, 2016. 87

[51] M.-C. De Marneffe, B. MacCartney, C. D. Manning, et al. Generating typed dependency parses from phrase structure parses. In *LREC*, 2006. 32

[52] V. R. de Sa. Spectral clustering with two views. In *ICML Workshop*, 2005. 31

[53] V. R. De Sa. *Unsupervised classification learning from cross-modal environmental structure*. PhD thesis, University of Rochester, 1994. 31

[54] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. Large-scale object classification using label relation graphs. In *ECCV*, 2014. 87

[55] C. Desai, D. Ramanan, and C. C. Fowlkes. Discriminative models for multi-class object layout. *IJCV*, 2011. 68, 87

[56] T. Deselaers, B. Alexe, and V. Ferrari. Weakly supervised localization and learning with generic knowledge. *IJCV*, 2012. 45

[57] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *SIGKDD*, 2001. 31

[58] S. K. Divvala, A. A. Efros, and M. Hebert. How important are 'deformable parts' in the deformable parts model? In *ECCV Workshop*, 2012. 11, 18, 21

[59] S. K. Divvala, A. Farhadi, and C. Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *CVPR*, 2014. 30, 42, 43, 45, 46, 47, 51, 53, 87

[60] S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert. An empirical study of context in object detection. In *CVPR*, 2009. 68, 85

[61] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. Efros. What makes Paris look like Paris? *SIGGRAPH*, 2012. 12, 21, 22

[62] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *CVPR*, 2015. 58, 69

[63] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *SIGKDD*, 2014. 1

[64] S. Ebert, D. Larlus, and B. Schiele. Extracting structures in image collections for object recognition. In *ECCV*, 2010. 9

[65] D. Elliott and F. Keller. Comparing automatic evaluation measures for image description. In *ACL*, 2014. 65

[66] J. L. Elman. Finding structure in time. *Cognitive science*, 1990. 57, 73

[67] M. Everingham, L. VanGool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes(voc) challenge. *IJCV*, 2010. 4, 42, 78, 90, 94

[68] A. Faktor and M. Irani. "clustering by composition" - unsupervised discovery of image categories. In *ECCV*, 2012. 20

[69] J. Fan, Y. Shen, N. Zhou, and Y. Gao. Harvesting large-scale weakly-tagged image databases from the web. In *CVPR*, 2010. 43

[70] H. Fang, S. Gupta, F. Iandola, R. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. Platt, C. L. Zitnick, and G. Zweig. From captions to visual concepts and back. *CVPR*, 2015. 58

[71] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009. 9, 87

[72] A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth. Every picture tells a story: Generating sentences from images. In *ECCV*, 2010. 58

[73] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *TPAMI*, 2006. 4, 85

[74] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 2010. 15, 18, 21, 24, 30, 42, 68, 70, 71, 72

[75] R. Fergus, P. Perona, and A. Zisserman. A visual category filter for Google images. In *ECCV*, 2004. 9, 30, 42, 45

[76] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from internet image searches. *Proceedings of the IEEE*, 2010. 43

[77] R. Fergus, Y. Weiss, and A. Torralba. Semi-supervised learning in gigantic image collections. In *NIPS*, 2009. 9

[78] D. A. Ferrucci. Introduction to “this is watson”. *IBM Journal of Research and Development*, 2012. 1

[79] D. F. Fouhey, V. Delaitre, A. Gupta, A. A. Efros, I. Laptev, and J. Sivic. People watching: Human actions as a cue for single view geometry. *IJCV*, 2014. 75

[80] B. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 2007. 12, 49

[81] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov, et al. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013. 57, 65, 66

[82] C. Galleguillos and S. Belongie. Context based object categorization: A critical survey. *CVIU*, 2010. 68

[83] C. Galleguillos, A. Rabinovich, and S. Belongie. Object categorization using co-occurrence, location and appearance. In *CVPR*, 2008. 68

[84] H. Gao, J. Mao, J. Zhou, Z. Huang, L. Wang, and W. Xu. Are you talking to a machine? dataset and methods for multilingual image question answering. In *NIPS*, 2015. 69

[85] S. Gidaris and N. Komodakis. Attend refine repeat: Active box proposal generation via in-out localization. *arXiv:1606.04446*, 2016. 70

[86] R. Girshick. Fast r-cnn. In *ICCV*, 2015. 70, 71, 74

[87] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 4, 5, 42, 43, 45, 49, 50, 56, 57, 63

[88] G. Gkioxari, R. Girshick, and J. Malik. Contextual action recognition with r* cnn. In *ICCV*, 2015. 68

[89] E. Golge and P. Duygulu. Conceptmap: Mining noisy web data for concept learning. In *ECCV*, 2014. 30, 43, 45

[90] Y. Gong, L. Wang, M. Hodosh, J. Hockenmaier, and S. Lazebnik. Improving image-sentence embeddings using large weakly annotated photo collections. In *ECCV*, 2014. 57

[91] A. Gonzalez-Garcia, A. Vezhnevets, and V. Ferrari. An active search strategy for efficient object class detection. In *CVPR*, 2015. 70

[92] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 2016. 73, 74

[93] M. Guillaumin, J. Verbeek, and C. Schmid. Multimodal semi-supervised learning for image classification. In *CVPR*, 2010. 9

[94] A. Gupta, A. Kembhavi, and L. S. Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *TPAMI*, 2009. 75

[95] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. *arXiv:1702.03920*, 2017. 71

[96] S. Gupta, B. Hariharan, and J. Malik. Exploring person context and local scene context for object detection. *arXiv:1511.08177*, 2015. 68, 75

[97] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. In *ECCV*, 2012. 12, 18, 21, 22, 34

[98] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *ECCV*, 2014. 45

[99] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. In *ECCV*, 2012. 48

[100] S. Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 1990. 87

[101] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 68, 74, 85, 87, 91, 94

[102] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015. 87

[103] P. Henderson and V. Ferrari. End-to-end training of object class detectors for mean average precision. *arXiv:1607.03476*, 2016. 70

[104] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

[105] J. R. Hobbs, M. Stickel, P. Martin, and D. Edwards. Interpretation as abduction. In *ACL*, 1988. 87

[106] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997. 57, 58, 71, 73

[107] M. Hodosh, P. Young, and J. Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *JAIR*, 2013. 56, 57, 65, 66

[108] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. In *ECCV*, 2012. 52, 54, 94

[109] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. *IJCV*, 2008. 68

[110] A. Hollingworth. Does consistent scene context facilitate object perception? *Journal of Experimental Psychology: General*, 1998. 68

[111] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 1989. 76

[112] J. Hosang, R. Benenson, and B. Schiele. Learning non-maximum suppression. *arXiv preprint arXiv:1705.02950*, 2017. 71

[113] Z. Hu, Z. Yang, R. Salakhutdinov, and E. P. Xing. Deep neural networks with massive learned knowledge. In *EMNLP*, 2016. 87

[114] 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. *arXiv:1611.10012*, 2016. 68, 71, 74

[115] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015. 78

[116] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *CVPR*, 2016. 101

[117] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, 2014. 47, 50, 61, 63

[118] J. Johnson, A. Karpathy, and L. Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *CVPR*, 2016. 70

[119] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *CVPR*, 2015. 87

[120] A. Joulin, F. Bach, and J. Ponce. Discriminative clustering for image co-segmentation. In *CVPR*, 2010. 20, 26

[121] A. Joulin, F. Bach, and J. Ponce. Multi-class cosegmentation. In *CVPR*, 2012. 26

[122] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016. 90, 95

[123] M. A. Just, S. D. Newman, T. A. Keller, A. McEleney, and P. A. Carpenter. Imagery in sentence comprehension: an fmri study. *Neuroimage*, 2004. 56

[124] A. Karpathy, A. Joulin, and L. Fei-Fei. Deep fragment embeddings for bidirectional image sentence mapping. *arXiv preprint arXiv:1406.5679*, 2014. 56, 57, 58, 64, 65, 66

[125] M. Kearns, Y. Mansour, and A. Y. Ng. An information-theoretic analysis of hard and soft assignment methods for clustering. In *Learning in graphical models*, 1998. 32

[126] S. Khan, F. Anwer, R. Muhammad, J. van de Weijer, A. Joost, M. Vanrell, and A. Lopez. Color attributes for object detection. In *CVPR*, 2012. 12, 15, 22

[127] G. Kim, E. P. Xing, L. Fei-Fei, and T. Kanade. Distributed Cosegmentation via Submodular Optimization on Anisotropic Diffusion. In *ICCV*, 2011. 20, 26

[128] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 87

[129] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic segmentation. *arXiv preprint arXiv:1801.00868*, 2018. 101

[130] R. Kiros, R. Salakhutdinov, and R. Zemel. Multimodal neural language models. In *ICML*, 2014. 58

[131] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014. 58

[132] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts. *TPAMI*, 2004. 19

[133] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011. 68, 87

[134] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *arXiv:1602.07332*, 2016. 69, 87, 93

[135] J. Krishnamurthy and T. M. Mitchell. Which noun phrases denote which concepts? In *ACL*, 2011. 30

[136] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 3, 42, 43, 45, 47, 57, 61, 63, 102

[137] D. Kuettel, M. Guillaumin, and V. Ferrari. Segmentation propagation in ImageNet. In *ECCV*, 2012. 9

[138] G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg. Baby talk: Understanding and generating simple image descriptions. In *CVPR*, 2011. 58, 61, 64

[139] M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *NIPS*, 2010. 45, 46

[140] M. P. Kumar, P. Torr, and A. Zisserman. Objcut: Efficient segmentation using top-down and bottom-up cues. *TPAMI*, 2010. 20

[141] D. Küttel and V. Ferrari. Figure-ground segmentation by transferring window masks. In *CVPR*, 2012. 20

[142] P. Kuznetsova, V. Ordonez, A. C. Berg, T. L. Berg, and Y. Choi. Collective generation of natural image descriptions. In *ACL*, 2012. 58

[143] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *TPAMI*, 2009. 70

[144] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009. 9

[145] N. Lao, T. Mitchell, and W. W. Cohen. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, 2011. 87

[146] Y. J. Lee and K. Grauman. Collect-cut: Segmentation with top-down cues discovered in multi-object images. In *CVPR*, 2010. 18, 20

[147] Y. J. Lee and K. Grauman. Object-graphs for context-aware category discovery. In *CVPR*, 2010. 20

[148] Y. J. Lee and K. Grauman. Learning the easy things first: Self-paced visual category discovery. In *CVPR*, 2011. 45, 46

[149] V. Lempitsky, P. Kohli, C. Rother, and T. Sharp. Image segmentation with a bounding box prior. In *ICCV*, 2009. 20

[150] D. Lenat and R. V. Guha. Cyc: A midterm report. *AI magazine*, 1990. 1

[151] N. Leoff, C. Alm, and D. Forysth. Discriminating image senses by clustering with multimodal features. In *ACL*, 2006. 30, 37, 39, 40

[152] M. Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *SIGDOC*, 1986. 28

[153] C. Li, D. Parikh, and T. Chen. Extracting adaptive contextual cues from unlabeled regions. In *ICCV*, 2011. 68

[154] J. Li, Y. Wei, X. Liang, J. Dong, T. Xu, J. Feng, and S. Yan. Attentive contexts for object detection. *IEEE Transactions on Multimedia*, 2016. 71

[155] L. Li, G. Wang, and L. Fei-Fei. A visual category filter for google images. In *CVPR*, 2006. 30

[156] L.-J. Li and L. Fei-Fei. Optimol: automatic online picture collection via incremental model learning. *IJCV*, 2010. 9, 42, 43, 45

[157] L.-J. Li, H. Su, L. Fei-Fei, and E. P. Xing. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, 2010. 70

[158] Q. Li, J. Wu, and Z. Tu. Harvesting mid-level visual concepts from large-scale internet images. In *CVPR*, 2013. 30, 43, 45

[159] Y. Li, K. He, J. Sun, et al. R-fcn: Object detection via region-based fully convolutional networks. In *NIPS*, 2016. 68, 72

[160] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *SIGGRAPH*, 2004. 20

[161] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015. 87

[162] X. Liang, L. Lee, and E. P. Xing. Deep variation-structured reinforcement learning for visual relationship and attribute detection. *arXiv:1703.03054*, 2017. 70

[163] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *JASIST*, 2007. 93

[164] L. R. Lieberman and J. T. Culpepper. Words versus objects: Comparison of free verbal recall. *Psychological Reports*, 1965. 56

[165] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. *arXiv:1612.03144*, 2016. 68, 87, 96

[166] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 43, 45, 46, 47, 57, 61, 62, 69, 70, 78, 87, 93, 99

[167] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 68, 72

[168] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv:1506.04579*, 2015. 78

[169] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 15

[170] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei. Visual relationship detection with language priors. In *ECCV*, 2016. 87

[171] J. Lu, J. Yang, D. Batra, and D. Parikh. Hierarchical question-image co-attention for visual question answering. In *NIPS*, 2016. 69

[172] Y. Lu, T. Javidi, and S. Lazebnik. Adaptive object detection using adjacency and zoom prediction. In *CVPR*, 2016. 70

[173] A. Lucchi and J. Weston. Joint image and word sense discrimination for image retrieval. In *ECCV*, 2012. 11, 30, 37

[174] T. Ma and L. J. Latecki. Graph transduction learning with connectivity constraints with application to multiple foreground cosegmentation. In *CVPR*, 2013. 20

[175] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *IJCV*, 2001. 19

[176] M. Malinowski, M. Rohrbach, and M. Fritz. Ask your neurons: A neural-based approach to answering questions about images. In *ICCV*, 2015. 69

[177] T. Malisiewicz and A. Efros. Beyond categories: The visual memex model for reasoning about object relationships. In *NIPS*, 2009. 9, 68

[178] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011. 18, 21, 22, 34

[179] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. 36, 37

[180] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv:1412.6632*, 2014. 69

[181] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille. Explain images with multimodal recurrent neural networks. *arXiv preprint arXiv:1410.1090*, 2014. 58, 59, 64, 65, 66

[182] K. Marino, R. Salakhutdinov, and A. Gupta. The more you know: Using knowledge graphs for image classification. *arXiv preprint arXiv:1612.04844*, 2016. 85, 87, 90

[183] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *CVPR*, 2009. 68

[184] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *TPAMI*, 2004. 15

[185] S. Mathe, A. Pirinen, and C. Sminchisescu. Reinforcement learning for visual object detection. In *CVPR*, 2016. 70

[186] E. Mezuman and Y. Weiss. Learning about canonical views from internet image collections. In *NIPS*, 2012. 43, 45, 46, 47

[187] T. Mikolov. Recurrent neural network based language model. In *INTERSPEECH*, 2010. 57, 58, 59, 60, 61, 63

[188] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *ICLR Workshop*, 2013. 59

[189] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Cernocky. Strategies for training large scale neural network language models. In *ASRU*, 2011. 61

[190] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013. 47

[191] T. Mikolov and G. Zweig. Context dependent recurrent neural network language model. In *SLT*, 2012. 57, 58, 59, 63

[192] G. A. Miller. Wordnet: a lexical database for english. *CACM*, 1995. 28, 47

[193] M. Mitchell, X. Han, J. Dodge, A. Mensch, A. Goyal, A. Berg, K. Yamaguchi, T. Berg, K. Stratos, and H. Daumé III. Midge: Generating image descriptions from computer vision detections. In *EACL*, 2012. 58, 61, 64

[194] 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 *CVPR*, 2014. 68

[195] K. Murphy, A. Torralba, W. Freeman, et al. Using the forest to see the trees: a graphical model relating features, objects and scenes. *NIPS*, 2003. 68

[196] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016. 87

[197] A. Newell. Physical symbol systems. *Cognitive science*, 1980. 87

[198] M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. In *ICML*, 2016. 87, 90

[199] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 2001. 14

[200] A. Oliva and A. Torralba. The role of context in object recognition. *Trends in cognitive sciences*, 2007. 68

[201] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Weakly supervised object recognition with convolutional neural networks. Technical report, 2014. 45

[202] V. Ordonez, G. Kulkarni, and T. L. Berg. Im2text: Describing images using 1 million captioned photographs. In *NIPS*, 2011. 42, 43

[203] B. Packer, S. Gould, and D. Koller. A unified contour-pixel model for figure-ground segmentation. In *ECCV*, 2010. 20

[204] A. Paivio, T. B. Rogers, and P. C. Smythe. Why are pictures easier to recall than words? *Psychonomic Science*, 1968. 56

[205] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell. Zero-shot learning with semantic output codes. In *NIPS*, 2009. 4

[206] t. E. Palmer. The effects of contextual scenes on the identification of objects. *Memory & Cognition*, 1975. 68

[207] M. Pandey and S. Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. In *ICCV*, 2011. 45

[208] P. Pantel and D. Lin. Discovering word senses from text. In *SIGKDD*, 2002. 30

[209] G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille. Weakly-and semi-supervised learning of a dcnn for semantic image segmentation. *arXiv:1502.02734*, 2015. 45, 48

[210] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002. 57, 61, 62, 64

[211] D. Parikh and K. Grauman. Relative attributes. In *ICCV*, 2011. 9, 87

[212] E. Parisotto and R. Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. *arXiv:1702.08360*, 2017. 71

[213] D. Pathak, E. Shelhamer, J. Long, and T. Darrell. Fully convolutional multi-class multiple instance learning. *arXiv:1412.7144*, 2014. 45

[214] G. Patterson and J. Hays. SUN attribute database: Discovering, annotating, and recognizing scene attributes. In *CVPR*, 2012. 9

[215] P. Perona. Visions of a Visipedia. *Proceedings of IEEE*, 2010. 7, 9

[216] D. Plummer and L. Lovász. *Matching Theory*. North-Holland Mathematics Studies. 1986. 36

[217] S. Qiu, X. Wang, and X. Tang. Visual semantic complex network for web images. In *ICCV*, 2013. 30

[218] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *CVPR*, 2009. 55

[219] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *ICCV*, 2007. 9, 68

[220] R. Raguram and S. Lazebnik. Computing iconic summaries of general visual concepts. In *CVPR Workshop*, 2008. 12, 30, 46

[221] C. Rashtchian, P. Young, M. Hodosh, and J. Hockenmaier. Collecting image annotations using Amazon's mechanical turk. In *NAACL Workshop*, 2010. 57, 62

[222] M. Rastegari, A. Farhadi, and D. Forsyth. Attribute discovery via predictable discriminative binary codes. In *ECCV*, 2012. 9

[223] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR Workshop*, 2014. 55

[224] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv:1412.6596*, 2014. 45, 46

[225] M. Ren and R. S. Zemel. End-to-end instance segmentation and counting with recurrent attention. *arXiv:1605.09410*, 2016. 71

[226] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv:1506.01497*, 2015. 4, 68, 70, 71, 72, 74, 78, 79, 82, 85, 94, 97

[227] A. Ritter, L. Zettlemoyer, Mausam, and O. Etzioni. Modeling missing data in distant supervision for information extraction. *TACL*, 2013. 28, 30

[228] A. Rohrbach, M. Rohrbach, R. Hu, T. Darrell, and B. Schiele. Grounding of textual phrases in images by reconstruction. In *ECCV*, 2016. 78

[229] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 4, 85

[230] C. Rother, T. Minka, A. Blake, and V. Kolmogorov. Cosegmentation of image pairs by histogram matching - incorporating a global constraint into MRFs. In *CVPR*, 2006. 20

[231] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": interactive foreground extraction using iterated graph cuts. *SIGGRAPH*, 2004. 20

[232] M. Rubinstein, A. Joulin, J. Kopf, and C. Liu. Unsupervised joint object discovery and segmentation in internet images. In *CVPR*, 2013. 18, 20, 25, 26, 27

[233] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 1, 7, 9, 15, 42, 43, 46, 61, 63, 71, 87, 90, 93, 94

[234] B. C. Russell, W. T. Freeman, A. A. Efros, J. Sivic, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR*, 2006. 20

[235] K. Saenko and T. Darrell. Filtering abstract senses from image search results. In *NIPS*, 2009. 30

[236] K. Saenko and T. Darrell. Unsupervised learning of visual sense models for polysemous words. In *NIPS*, 2009. 30, 36, 37, 39, 40, 45

[237] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. *arXiv preprint arXiv:1706.01427*, 2017. 87

[238] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001. 93

[239] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *TNN*, 2009. 87, 90

[240] F. Schroff, A. Criminisi, and A. Zisserman. Harvesting image databases from the web. *TPAMI*, 2011. 9, 30, 42, 45

[241] H. Schütze. Automatic word sense discrimination. *Computational linguistics*, 1998. 36, 39

[242] V. Sharmanska, N. Quadrianto, and C. H. Lampert. Augmented attribute representations. In *ECCV*, 2012. 9

[243] J. Shi and J. Malik. Normalized cuts and image segmentation. *TPAMI*, 1997. 19, 23

[244] K. J. Shih, S. Singh, and D. Hoiem. Where to look: Focus regions for visual question answering. In *CVPR*, 2016. 69

[245] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*, 2006. 68

[246] A. Shrivastava, S. Singh, and A. Gupta. Constrained semi-supervised learning using attributes and comparative attributes. In *ECCV*, 2012. 9

[247] A. Shrivastava and A. Gupta. Contextual priming and feedback for faster r-cnn. In *ECCV*, 2016. 68

[248] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond skip connections: Top-down modulation for object detection. *arXiv:1612.06851*, 2016. 81, 87

[249] B. Siddiquie and A. Gupta. Beyond active noun tagging: Modeling contextual interactions for multi-class active learning. In *CVPR*, 2010. 9

[250] G. A. Sigurdsson, X. Chen, and A. Gupta. Learning visual storylines with skipping recurrent neural networks. In *ECCV*, 2016. 42, 101

[251] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv:1312.6034*, 2013. 45, 48

[252] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014. 44, 61, 64, 65, 66, 68, 71

[253] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In *ECCV*, 2012. 21, 22

[254] S. Singh, D. Hoiem, and D. Forsyth. Learning a sequential search for landmarks. In *CVPR*, 2015. 70

[255] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering object categories in image collections. In *ICCV*, 2005. 20

[256] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003. 45

[257] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *TPAMI*, 2000. 45

[258] R. Snow, S. Prakash, D. Jurafsky, and A. Y. Ng. Learning to merge word senses. In *EMNLP*, 2007. 28

[259] R. Socher, Q. Le, C. Manning, and A. Ng. Grounded compositional semantics for finding and describing images with sentences. In *NIPS Workshop*, 2013. 56, 57, 65, 66

[260] H. O. Song, R. Girshick, S. Jegelka, J. Mairal, Z. Harchaoui, and T. Darrell. On learning to localize objects with minimal supervision. In *ICML*, 2014. 45

[261] L. S. Stavrianos. *A global history: From prehistory to the 21st century*. 2004. 100

[262] R. Stewart, M. Andriluka, and A. Y. Ng. End-to-end people detection in crowded scenes. In *CVPR*, 2016. 71

[263] E. Sudderth, A. Torralba, W. T. Freeman, and A. Wilsky. Learning hierarchical models of scenes, objects, and parts. In *ICCV*, 2005. 9

[264] S. Sukhbaatar and R. Fergus. Learning from noisy labels with deep neural networks. *arXiv:1406.2080*, 2014. 45, 46, 48

[265] S. Sukhbaatar, J. Weston, R. Fergus, et al. End-to-end memory networks. In *NIPS*, 2015. 73, 74

[266] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *ICCV*, 2017. 87

[267] I. Sutskever, J. Martens, and G. E. Hinton. Generating text with recurrent neural networks. In *ICML*, 2011. 58

[268] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, 1999. 75, 77

[269] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 44

[270] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014. 45

[271] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. Yfcc100m: The new data in multimedia research. *CACM*, 2016. 49

[272] A. Torralba. Contextual priming for object detection. *IJCV*, 2003. 68

[273] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR*, 2011. 42, 47, 93

[274] A. Torralba, K. P. Murphy, W. T. Freeman, M. A. Rubin, et al. Context-based vision system for place and object recognition. In *ICCV*, 2003. 68, 87

[275] L. Torresani, M. Szummer, and A. Fitzgibbon. Efficient object category recognition using classemes. In *ECCV*, 2010. 43, 45

[276] D. Tsai, Y. Jing, Y. Liu, H. A. Rowley, S. Ioffe, and J. M. Rehg. Large-scale image annotation using visual synset. In *ICCV*, 2011. 30

[277] Z. Tu and X. Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *TPAMI*, 2010. 68, 87

[278] T. Tuytelaars, C. H. Lampert, M. B. Blaschko, and W. Buntine. Unsupervised object discovery: A comparison. *IJCV*, 2010. 20

[279] R. Vedantam, C. L. Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. *CVPR*, 2015. 57, 62, 64, 65

[280] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko. Sequence to sequence-video to text. In *ICCV*, 2015. 69

[281] S. Vicente, C. Rother, and V. Kolmogorov. Object cosegmentation. In *CVPR*, 2011. 20

[282] S. Vijayanarasimhan and K. Grauman. Keywords to visual categories: Multiple-instance learning forweakly supervised object categorization. In *CVPR*, 2008. 42, 45

[283] S. Vijayanarasimhan and K. Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. In *CVPR*, 2011. 9

[284] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008. 60

[285] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. *arXiv:1411.4555*, 2014. 58, 69

[286] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *SIGCHI*, 2004. 9

[287] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 2007. 36

[288] J. Von Neumann. First draft of a report on the edvac. *IEEE Annals of the History of Computing*, 1993. 73

[289] K.-W. Wan, A.-H. Tan, J.-H. Lim, L.-T. Chia, and S. Roy. A latent model for visual disambiguation of keyword-based image search. In *BMVC*, 2009. 30, 37, 39, 40

[290] C. Wang, W. Ren, K. Huang, and T. Tan. Weakly supervised object localization with latent category learning. In *ECCV*, 2014. 45

[291] X.-J. Wang, L. Zhang, X. Li, and W.-Y. Ma. Annotating images by mining image search results. *TPAMI*, 2008. 42, 45

[292] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *CVPR*, 2016. 85, 87

[293] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *ICCV*, 2013. 68

[294] R. J. Williams and D. Zipser. Experimental analysis of the real-time recurrent learning algorithm. *Connection Science*, 1989. 61

[295] R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. *Back-propagation: Theory, architectures and applications*, 1995. 75

[296] Q. Wu, P. Wang, C. Shen, A. Dick, and A. van den Hengel. Ask me anything: Free-form visual question answering based on knowledge from external sources. In *CVPR*, 2016. 87

[297] Y. Xia, X. Cao, F. Wen, and J. Sun. Well begun is half done: Generating high-quality seeds for automatic image dataset construction from web. In *ECCV*, 2014. 43, 45

[298] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 15, 46

[299] S. Xie, X. Huang, and Z. Tu. Top-down learning for structured labeling with convolutional pseudoprior. In *ECCV*, 2016. 69, 87

[300] C. Xiong, S. Merity, and R. Socher. Dynamic memory networks for visual and textual question answering. *arXiv*, 2016. 69, 85, 87

[301] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei. Scene graph generation by iterative message passing. *arXiv preprint arXiv:1701.02426*, 2017. 87

[302] H. Xu and K. Saenko. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *ECCV*, 2016. 69, 71

[303] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015. 69, 71

[304] Y. Yang, C. L. Teo, H. Daumé III, and Y. Aloimonos. Corpus-guided sentence generation of natural images. In *EMNLP*, 2011. 58

[305] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola. Stacked attention networks for image question answering. In *CVPR*, 2016. 69, 87

[306] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *CVPR*, 2010. 68, 75

[307] B. Z. Yao, X. Yang, L. Lin, M. W. Lee, and S.-C. Zhu. I2T: Image parsing to text description. *Proceedings of the IEEE*, 2010. 58

[308] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, 1995. 30, 32

[309] D. Yoo, S. Park, J.-Y. Lee, A. S. Paek, and I. So Kweon. Attentionnet: Aggregating weak directions for accurate object detection. In *ICCV*, 2015. 70

[310] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 43, 87

[311] 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 *ICCV*, 2015. 69

[312] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014. 42, 43, 55

[313] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ade20k dataset. *arXiv preprint arXiv:1608.05442*, 2016. 87, 93, 99

[314] X. Zhu. Semi-supervised learning literature survey. Technical report, CS, UW-Madison, 2005. 9

[315] Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei. Visual7w: Grounded question answering in images. In *CVPR*, 2016. 69

[316] Y. Zhu, C. Zhang, C. Ré, and L. Fei-Fei. Building a large-scale multimodal knowledge base system for answering visual queries. *arXiv:1507.05670*, 2015. 70, 86, 87

[317] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 48, 49

[318] C. L. Zitnick and D. Parikh. Bringing semantics into focus using visual abstraction. In *CVPR*, 2013. 58