DOCTORAL THESIS

Visual Learning with Minimal Human Supervision

Ishan Misra
The Robotics Institute
Carnegie Mellon University

Doctoral Committee:
Martial Hebert, *Carnegie Mellon University* (Co-chair)
Abhinav Gupta, *Carnegie Mellon University* (Co-chair)
Deva Ramanan, *Carnegie Mellon University*
Alexei A. Efros, *University of California, Berkeley*
Andrew Zisserman, *University of Oxford*

Tech Report Number: CMU-RI-TR-18-40

Submitted in partial fulfillment of the requirements
of the degree of Doctor of Philosophy in Robotics
August, 2018

Abstract

Machine learning models have led to remarkable progress in visual recognition. A key driving factor for this progress is the abundance of labeled data. Over the years, researchers have spent a lot of effort curating visual data and carefully labeling it. However, moving forward, it seems impossible to annotate the vast amounts of visual data with everything that we care about. This reliance on exhaustive labeling is a key limitation in the rapid deployment of computer vision systems in the real world. Our current systems also scale poorly to the large number of concepts and are passively spoon-fed supervision and data.

In this thesis, we explore methods that enable visual learning without exhaustive supervision. Our core idea is to model the natural regularity and repetition in the visual world in our learning algorithms as their inductive bias. We observe recurring patterns in the visual world - a person always lifts their foot before taking a step, dogs are similar to other furry creatures than to furniture *etc*. This natural regularity in visual data also imposes regularities on the semantic tasks and models that operate on it - a dog classifier must be similar to classifiers of furry animals than to furniture classifiers. We exploit this abundant natural structure or 'supervision' in the visual world in the form of self-supervision for our models, modeling relationships between tasks and labels, and modeling relationships in the space of classifiers. We show the effectiveness of these methods on both static images and videos across varied tasks such as image classification, object detection, action recognition, human pose estimation *etc*. However, all these methods are still passively fed supervision and thus lack agency: the ability to decide what information they need and how to get it. To this end, we propose having an interactive learners that ask for supervision when needed and can also decide what samples they want to learn from.

Acknowledgments

I believe none of this work would be possible without the guidance, encouragement, and mentorship of all my teachers. They have each taught me many important lessons and continue to inspire me.

My PhD advisors Martial Hebert and Abhinav Gupta have my immense gratitude for their mentorship and support over the past six years. They provided me with great ideas, advice, motivation and more importantly taught me to learn from my failures. Martial took me on as a Masters student and gave me the opportunity to pursue graduate research. His support from that day has been unconditional, patient and that created an environment where I could freely pursue research. He has taught me to look at problems holistically and to always keep an eye out for the positives in everything I read or listen to. I thank you for your confidence in me. I am also priviledged to have worked with Abhinav who has set an inspiring high standard for research. He has been a great mentor and a joy to learn from. Abhinav inspires me by his enthusiasm, creativity in asking and answering research questions, dedication to research and the tireless effort he puts in his students to nurture them. His advice and mentorship have been invaluable. Thank you both for being my advisors and teaching me.

I have been lucky to have many great mentors through my graduate life. I am grateful to Larry Zitnick for his guidance and for believing in me. His continued encouragement, creativity and advice have inspired my thought process and research. His mentorship has been vital in helping me get here. Abhinav Shrivastava for being a great friend and mentor throughout these years, right from my first month at CMU. His advice and mentorship on research, experimentation, writing, presentations, Delta miles, ... (there are innumerable such topics) has helped me grow tremendously. Ross Girshick for many insightful discussions, and teaching me how to be a good empiricist. He has helped me become a better researcher and think deeply. Meg Mitchell for an amazing internship, introducing me to NLP and for her continued advice. Laurens van der Maaten for giving me (apart from technical knowledge) succinct life lessons (law of preservation of misery ;)). Rob Fergus for being a patient mentor, listening to all my bad ideas and making them better and for guiding me through research and my job search. Andrew Zisserman for teaching me to always question and examine things. His patient support has been invaluable. Rahul Sukthankar for giving me sage advice and encouragement right from my first year, and for always being extremely approachable and friendly. Cordelia Schmid for her insightful feedback, guidance and encouragement. Drew Bagnell, Fernando de La Torre, Kavyon Fatahalian, Yaser Sheikh, Kris Kitani, David Forsyth for their encouragement and many fruitful discussions. Kannan Srinathan and PJ Narayanan for being phenomenal mentors. Kannan's profound statements still echo in my mind and inspire me.

I would like to thank my thesis committee - Deva Ramanan, Alyosha Efros and Andrew Zisserman for taking the time to give me valuable feedback. My phenomenal collaborators - Elissa Aminoff, Dhruv Batra, Nadine Chang, Jacob Devlin, Debidatta Dwibedi, Rob Fergus, Frank Ferraro, Bryan Ford, Abhinav Gupta, Ross Girshick, Martial Hebert, Angela Jiang, Pushmeet Kohli, Iasonas Kokkinos, Nilesh Kulkarni, Laurens van der Maaten, Nasrin Mostafazedeh, Margaret Mitchell, Devi Parikh, Abhinav Shrivastava, Lucy Vanderwende, Larry Zitnick for giving me the opportunity to work with them. I have learned a lot from each of you. Josef Sivic, Ivan Laptev, Karteek Alahari, Cordelia Schmid for their encouragement. Bryan Russell and Aaron Hertzmann for great

discussions, demos and food recommendations. My labmates in Martial and Abhinav's groups for giving me a great environment in my PhD. A special shout out to - Aayush Bansal, Ricardo Cabral, Xinlei Chen, Nadine Chang, Achal Dave, Debadeepta Dey, Carl Doersch, David Fouhey, Dhiraj Gandhi, Allison Del Giorno, Rohit Girdhar, Kiryong Ha, Ed Hsiao, Hanbyul Joo, Natasha Kholgade, Adithya Murali, Lerrel Pinto, Senthil Purushwalkam, Varun Ramakrishna, Scott Satkin, Kumar Shaurya Shankar, Bhaskar Vaidya, Jack Valmadre, Arun Venkatraman, Francisco Vicente, Jacob Walker, Yuxiong Wang, Xiaolong Wang, Luke Yoder who have all helped me learn. Lynnetta Miller for making sure my CMU journey went ahead without a hitch. Suzanne Muth for helping me glue all the different RI threads. Ed Walter and Bill Love for keeping the clusters and machines operating smoothly. Carly Devenburgh and Alison Day for their help with immigration. Abhinav Gupta and Swati Jarial for being gracious hosts and making life fun by way of decompression parties, celebrations, retreats and providing a superb environment.

Pittsburgh has also given me many valuable friends who have been with me through thick and thin. Yash Puranik for helping me on day zero at Pittsburgh without knowing me, and for his continued friendship ever since. Francisco Vicente for being a great friend and concert buddy. Paco your encouragement has always meant a lot to me. Abhinav Shrivastava and Varuni Saxena for making sure I missed my home a little less, taking care of me, *biryani* (and other great food and drinks), and for always being around to help me with even the smallest of things. Yash Puranik and Jessica Cheng for their love and support, making sure I was fed, *chai*, cookies, late night discussions, and always being there to listen to me. Debadeepta Dey and Shannon Young for making me feel welcome and checking in on me. Rohit Girdhar for his friendship, advice, our daily fun conversations and helping me when I broke my foot. Rachel Burcin for not only helping me in her official capacity, but going above and beyond by being a wonderful friend throughout the years. My knowbc gang - Mayank Gupta, Nikhil Soni, Aditya Deshpande, Kaustav Kundu, Ishaan Singh, Aravindh Mahendran, Archit Jain and Ankit Gandhi for cheering me on and supporting me during several key moments throughout these years.

I would not be in this position without my parents' unconditional love and support. They have been my strongest supporters in every endeavor and have left no stone unturned in supporting me. Their faith in me, despite my early education performance, was the only reason I could undertake graduate studies. Thank you for teaching me the value of hard work and for your countless sacrifices. I thank my sister and first strict teacher, Isha, for her love and fierce support of my decisions. I would not be here without you instilling the joy of reading and teaching me to always forge ahead. Aditya Samant for inspiring me to calmly and confidently handle things. Saloni for loving me, always having my back, tolerating me when I was stressed, patiently listening to me, technical feedback, being a calming presence and for holding me to high standards and helping me achieve them. You have silently but surely made everything better.

Contents

Chapter 1

Introduction

We are so familiar with seeing, that it takes a leap of imagination to realise that there are problems to be solved.

R. L. GREGORY in *Eye and Brain: the psychology of seeing*, 1966

"The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system." This was the famous 'Summer Vision' project [313] at MIT that called for figure-ground analysis and region-description (both problems are a major chunk of visual recognition) to be solved in a *single* summer. This internship post and the quote at the beginning of this chapter by R. L. Gregory tell us that perception is such an *obvious* skill for human that it takes some thinking to appreciate how involved it truly is. It takes even more thinking to then see how making machines 'see' is indeed a hard problem. Unsurprisingly, a single summer internship did not suffice to solve this grand challenge of machine vision, in the 45 years since, machine learning models have made tremendous headway. For tasks like object recognition and image classification, we have seen algorithms reduce error rates by more than 3 times in the last 5 years, and in some cases even surpass human level performance. A major driving force behind this progress is the availability of large scale exhaustive supervision. Tireless data curation efforts led by researchers have let us train high capacity machine learning models with impressive performance. This trend of curating larger datasets and ever larger models has served the recognition community well. However, this begs the question - is the way to solve scene understanding or visual recognition to collect data for *all* the things we care about? Is computer vision mainly about data curation, manual annotation, and 'spoon-feeding' such data to machine learning models?

1.1 Why not just datasets?

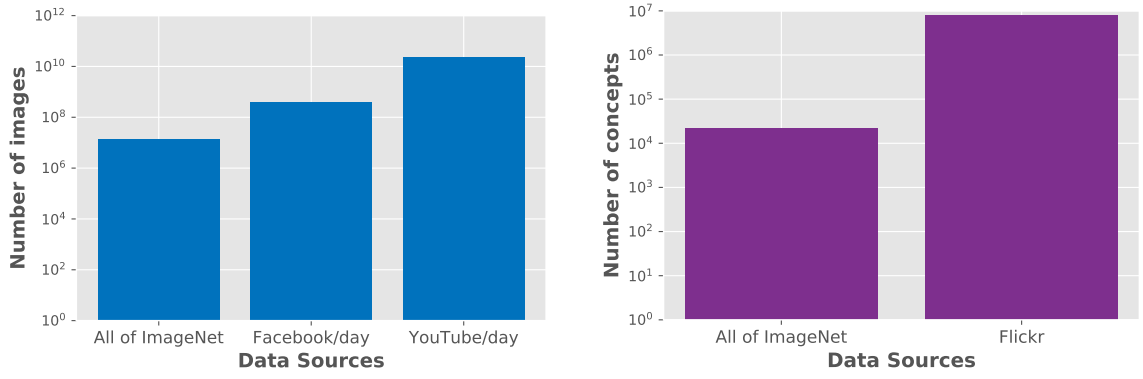
To understand if we can 'solve' computer vision simply by scaling out the data curation and labeling task, we need to first look at the scale of the data we would want to label. A simple way to limit the scope of this problem is to only consider visual data on the web. This gives us a good estimate of the size of data and number of categories.

Scale of data and classes: With the availability of cheap visual sensors in smartphones *etc.*, the amount of visual data on the web has grown tremendously. Figure 1.1 shows us we upload 400 million pictures *every day* and 300 hours of video content *every minute* to just the popular media sharing platforms. To put these numbers in perspective, Facebook adds a full ImageNet sized dataset to itself every hour and YouTube does this every 13 minutes. ImageNet [351] is one of the largest vision datasets with 14 million images and 22k categories, and took about 19 man years to label. Even this large dataset pales in comparison to just internet images. Another rough estimate [408], shows us that there are about 8 million 'tags' or visual concepts on the web, which is again massive compared to ImageNet or any similar vision dataset. Thus, setting up an army of annotators to label this ever growing size of visual data with all the categories we care about is practically impossible.

Rare occurrences: A major difficulty in deploying recognition systems in the real world is the long tail of visual categories or events. As has been empirically demonstrated [419], the number of atypical events is very large and follows the Zipf's law (too many categories are atypical and have less data). This makes it difficult to both curate and label such data. It also poses additional demands of sample efficiency on our learning methods. We cannot always expect to have a large amount of pristinely labeled data to learn a visual concept.

Artificially balanced datasets: Vision datasets have been artificially balanced to handle the long tail of visual data. As an example (Figure 1.2) the number of giraffes in the real world is about 100,000[1] compared to the human population at roughly 7 billion. So, in the real world, the ratio of giraffes to humans is about 10^{-5}. In the benchmark COCO dataset [251], this ratio is about 0.1. This makes it so that giraffes in COCO are *more* frequent than dogs are in the real world (ratio of dogs to humans in the world is 0.005)! This 'balance' makes it convenient for us to train recognition algorithms, but it imposes artificial distributional priors on our models. A machine vision model

[1] International Union for Conservation of Nature, 2015



(a) (b)

Figure 1.1: A comparison of the number of images and concepts in vision datasets and on the web. (a) ImageNet, which took about 19 man years to label, is one of the largest publicly available labeled datasets with about 22k categories. The size of this dataset pales in comparison to the volume of web data. (b) Additionally, a conservative estimate shows that there are roughly 8 million tags (categories) on web data. So even the largest dataset can hardly keep up with the ever growing number of visual concepts or 'things we care about'.

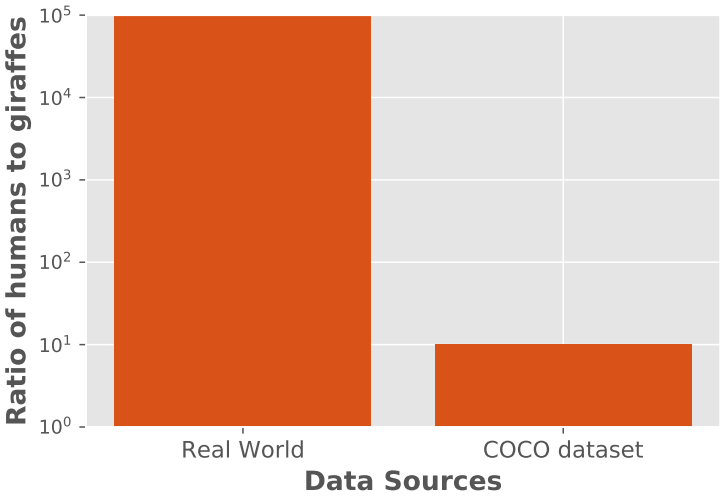


Figure 1.2: The distribution mismatch in the real world and vision datasets. Vision datasets, like COCO [251], are artificially balanced. While this makes training and optimization easy, it creates artificially high occurrences of some classes. As an example, the ratio of humans to giraffes in the real world is 10^5, whereas in COCO it is 10.

is in for a surprise when it finds the real (boring) world with so few giraffes! Such an artificial balancing has also made it hard for us to appreciate the long tail of categories.

1.2 Generalization, interaction and sample efficiency

Spoon-feeding or passive supervision is an effective technique for training models. On the one hand, we have made immense progress on 'closed-world' scenarios where we know the categories and their distributions *apriori*. On the other hand, passive learning just by itself does not create models that can generalize to unseen test distributions or environments in the 'open-world'. This generalization requires two key components missing from current line of research - 1) Interactive learning where models can *ask* for information and decide *when and how* to get it. 2) Sample efficient learning where models can learn from a few samples. Both these components are necessary for real world agents that will encounter unseen concepts and cannot be 'taught' everything beforehand. These agents should learn about atypical events quickly without repeated demonstration, displaying high sample efficiency.

1.3 Thesis Contributions

In this thesis, we study methods that reduce the need for exhaustive human supervision. Such methods have seen renewed interest in the light of data hungry deep learning techniques. Traditional fully supervised algorithms require copious amounts of data and labels to adapt to new tasks. This is because they treat data and labels as mere monolithic vectors with no structure. The high capacity models used have little inductive bias and thus need a lot of data to generalize well.

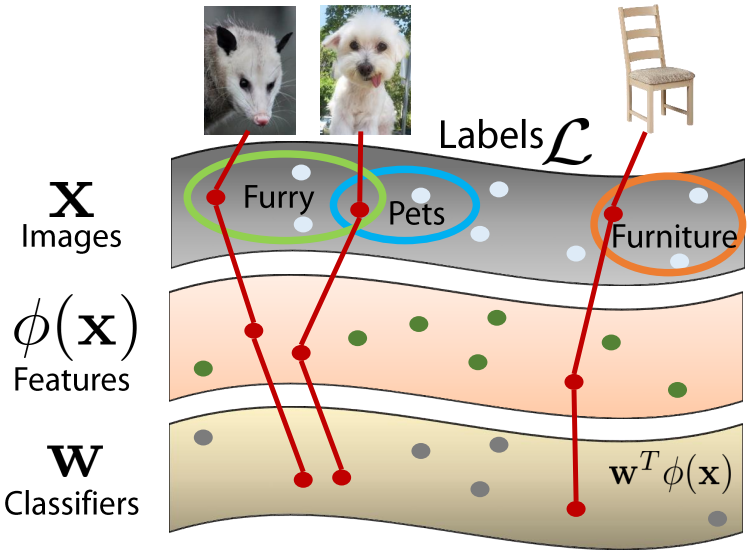


Figure 1.3: Visual data is structured - a possum appears more like a dog than a chair. The tasks or labels defined for visual data determine some semantic groupings which also have an underlying structure, *e.g.*, furry animals are similar to pets than furniture. The classifier space w is designed to capture similarity in the data, feature and label space. It can be viewed as a 'dual' of the feature space $\phi(x)$. This structure implies that a possum classifier is more similar to a dog classifier than a chair classifier. In this thesis we propose to use such structure in visual data, labels and classifiers to reduce the need for human supervision.

Modeling the underlying inductive biases can adapt models to changing tasks without exhaustive supervision or re-training. We show how such techniques can be applied to both static images and videos. As we show in our work, modeling these inductive biases often leads to more interpretable models which is an often desired property. We propose to capture various forms of inductive biases - in the data, task, model and label space to reduce the need for supervision. As we illustrate in Figure 1.3, in this thesis we explore the following axes of inductive biases or structure to reduce the need for supervision:

1. **Natural temporal supervision in data:** Video data offers a natural and powerful source of free supervision in the form of temporal constraints. We use these temporal constraints and sequential nature of video to learn visual representations [288] by predicting the order of frames in a video. We show that the learned representation can be used for tasks such as human pose estimation and action recognition. We also show [285] how such constraints can help us start from a handful of labels to label orders of magnitude more data (100×).
2. **Relationships between tasks:** The structure of the visual world translates into structure in the various semantic tasks it affords. Treating tasks independently ignores their relationships and shared information. We propose a simple and principled way to use multiple sources of supervision using Multi-task Learning in ConvNets [286]. Our proposed mechanism, called 'cross-stitch units', helps explore many different ConvNet architectures for Multi-task Learning without training all of them. It also improves performance for classes with little data.
3. **Relationships between labels:** A natural extension of exploring structure in task space, is to look deeper and find structure within a single task. Labels from a single task impose structure by capturing rich relationships. We propose [289] a way to model the structure in labels and

the annotation process to enable the use of noisy labels. We show significant gains over the state-of-the-art and paves the way for efficient use of noisy labels. This modeling also results in interpretable and more accurate classifiers.

4. **Structure in model space:** Since visual classifiers operate on structured visual data, they naturally capture certain inductive biases in the data. We explore such properties in the model space to compose classifiers for novel unseen classes [290], and to create surprisingly simple yet performant data augmentation techniques for instance detection [89].

While exploring this structure in the data, label and model space reduces the need for supervision, it still does not make our models 'independent'. Our current algorithms are *passive* in nature: we curate and feed data into them to train them. These models have no control over what data is input to them or how many times they need to 'revise' such data. We propose to explore strategies to enable agents to be both interactive so that they can ask for supervision, and also be sample efficient so that they do not need to revisit easy examples over and over again.

Interactive Learning: Our current models lack *agency*: the ability to explore the world and acquire knowledge that fills the gaps. As agents are exposed to the world beyond the datasets they were trained on, they will need to adapt and acquire supervision without being spoon fed. As humans, we have the ability to go to unseen environments, *e.g.*, a foreign country where people speak a different language, and still be able to go about our tasks such as navigation *etc*. and ask for guidance when needed using say gestures. We want *active* and *interactive* agents that can model what they know and what they do not know so that they can ask for supervision when needed. We first explore conversational agents that can ask engaging questions about images and propose a task named Visual Question Generation (VQG) [298]. We show that this task requires visual understanding and contextual understanding. Using the lessons learned in this work, we then explore a visual learning framework which allows an agent to ask the oracle for supervision it needs. This framework, which we call Learning by Asking (LBA) [291], shows the sample efficiency and strong generalization abilities of interactive learners.

1.4 Thesis Organization

This thesis is organized in four sections that show how we can reduce supervision by exploring structure in (I) data, (II) labels and tasks, (III) classifiers and (IV) interaction. These sections are summarized below.

Part I: Structure in Visual Data - In Chapters 2 and Chapters 3 we explore the natural supervision available in videos. As videos naturally define a sequence over images, they help impose constraints on our models and representations. In Chapter 2, we use the temporal constraints in video for semi-supervised learning of object detectors. We start with a handful of positive bounding boxes in videos and train object detectors using them. We use these detectors to label more data in videos, using temporal constraints to remove false positives and iteratively grow the set of bounding boxes. We show how the temporal constraints help label orders of magnitude more diverse data. In Chapter 3, we use the temporal constraints in video to learn a feature representation in a self-supervised fashion. We predict whether a set of images form a valid sequence from a video and use this as our learning signal. We show that a feature representation learned in this way helps learn better action recognition and human pose estimation models.

Part II: Structure in Labels and Tasks - The tasks we use in computer vision depend on visual data and also capture regularity. In Chapter 5, we show how to model such relationships between tasks using Multi-task Learning. We propose a mechanism called 'cross-stitch' unit that helps learn and combine feature representations across different tasks in ConvNet. The relationships between tasks are modeled implicitly by the unit as it learns to combine the representations. We show that such a unit helps improve performance across tasks and in particular helps tasks with little data. We consider pairs of tasks like surface normal prediction and semantic segmentation; and object detection and attribute prediction. In particular we show that reasoning about surface geometry helps understand how to label pixels with semantic classes - flat upward facing surfaces are more likely to tables than bags; and knowing about the object class helps understand what attributes it may have - horses are more likely to have saddles than do sheep. In Chapter 6, we show an efficient way to perform inference across multiple tasks while leveraging the benefit of multiple types of supervision. We propose a system called Mainstream that is performant and scales to many different tasks on video analysis.

In Chapter 4, we show how the structure in visual data transfers to structure in labels. In particular, we show that using such structure we can learn visual classifiers from noisy image level descriptions or labels. The 'noise' in these descriptions or labels is not random but captures the human reporting bias when we talk about visual data. We are not likely to mention the color 'yellow' while talking about bananas, but likely to mention it when talking about apples. We model such peculiarities in this label noise and show that we can not only model it in the learning algorithm but that it also helps learn 'clean' visual classifiers.

Part III: Structure in Classifiers - In Chapters 7 and 8 we show how to use the properties of classifiers to reduce supervision. In Chapter 7, we exploit the property of current instance detection models to show a surprisingly simple technique to synthesize data for them. We notice that for texture rich instances, the models focus more on local appearance rather than global appearance. We show that we can 'cut' instances from real images and 'paste' them in new real scenes while ensuring local realism to generate large amounts of data that can train detectors effectively. Our technique is both simple and outperforms existing synthesis techniques for instance detection. In Chapter 8, we show that the space of classifiers has underlying structure that can be used to compose them. We learn a transformation that takes classifiers of attributes and objects and can compose them together to get a classifier of attribute object pairs. Our transformation models both compositionality and contextuality of visual concepts and can generalize to unseen pairs and even triples of combinations.

Part IV: Interactive Learning - In Chapters 9 and 10, we explore ways to move away from all the passively supervised models explored thus far. In Chapter 9, we propose the task of Visual Question Generation (VQG) in which an agent must ask engaging questions about images. We show that such questions need visual understanding and common sense. Using the lessons learned from question generation, in Chapter 10 we explore *interactive* agents that can ask for supervision rather than working with a fixed dataset alone. We study these agents in the setting of Visual Question Answering (VQA) and propose a framework called Learning by Asking (LBA).

The relevant publication list for each chapter is as follows

1. Chapter 2 - Watch and Learn: Semi-supervised learning of object detectors from video [285] (CVPR 2015)

2. Chapter 3 - Shuffle and Learn: Unsupervised learning using Temporal Order Verification [288] (ECCV 2016)
3. Chapter 4 - Seeing through the Human Reporting Bias: Visual classifiers from Noisy Human-centric Labels [289] (CVPR 2016)
4. Chapter 5 - Cross Stitch Networks for Multi-task learning [286] (CVPR 2016)
5. Chapter 6 - Mainstream: Dynamic Stem-Sharing for Multi-Tenant Video Processing [2] (USENIX ATC 2018)
6. Chapter 7 - Cut Paste and Learn: Surprisingly Easy Texture Synthesis [89] (ICCV 2017)
7. Chapter 8 - From Red Wine to Red Tomato: Composition with Context [290] (CVPR 2017)
8. Chapter 9 - Visual Question Generation [298] (ACL 2016)
9. Chapter 10 - Learning by Asking [291] (CVPR 2018)

Part I

Structure in Visual Data

Chapter 2

Temporal Constraints for Semi-Supervised Learning of Detectors

In actuality, virtually all learning phenomena resulting from direct experiences can occur on a vicarious basis through observation of other people's behaviour and its consequences for them.

ALBERT BANDURA in *Social Learning Theory*, 1991

In this section we use the temporal structure in visual data (videos) to learn with minimal human supervision. As the quote in the beginning of this chapter tells us, it is natural for humans to learn by simply observing or watching other people and objects. We take inspiration from this and focus on how temporal constraints given from video can help us 'watch' (track objects, remove false positive detections) and 'learn' (improve detectors by using tracked objects). We present a scalable semi-supervised learning framework to learn object detectors from a few manual labels. We focus on video data for two reasons - 1) It provides natural temporal supervison. 2) Large scale video datasets are hard to annotate and curate.

The human effort required for labeling vision datasets is huge, e.g., ImageNet [79] required 19 man-years to label bounding boxes in the 1.2 million images harvested from the internet. Consider the scale of internet videos – YouTube has 300 hours of video uploaded every minute. It seems unlikely that human per-image labeling will scale to this amount of data. Given this scale of data and the associated annotation problems [305, 456], which are more pronounced in videos, it is no surprise that richly annotated large video recognition datasets are hard to find. In fact, the available video datasets [212, 273, 305, 384] lack the kind of annotations offered by benchmark image datasets [79, 95, 251]. In this chapter, we wish to automatically increase the amount of labeled data in videos using Semi-Supervised Learning (SSL).

Visual data and especially video has intrinsic spatiotemporal regularity. In this chapter, we capitalize on this spatiotemporal regularity in videos to learn object detectors from a few labels. Our goal is to start with a few annotated examples, and label more examples automatically using Semi-Supervied Learning (SSL). However, a major challenge for any kind of SSL technique is to constrain the learning process to avoid semantic drift, i.e., added noisy samples cause the learner to drift away from the true concept. Recent work [57, 61, 83, 370] has shown ways to constrain this

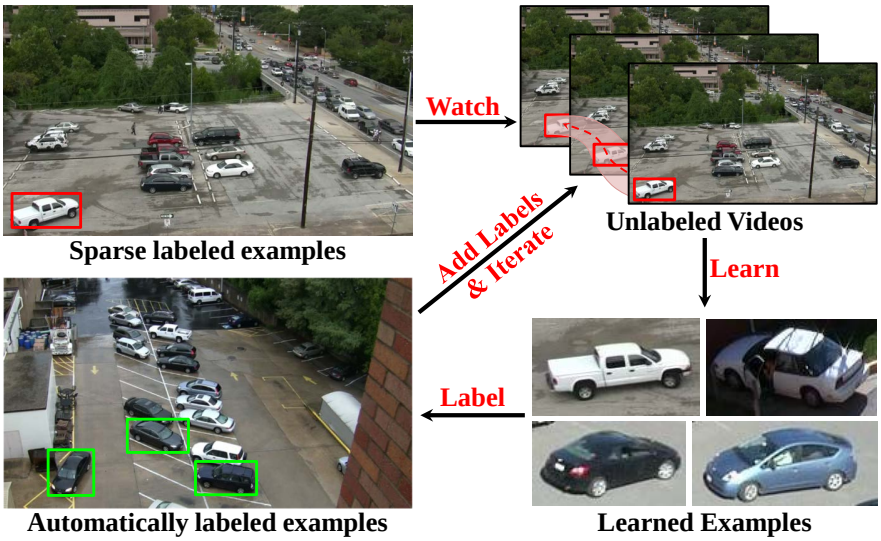


Figure 2.1: We present a novel formulation of semi-supervised learning for automatically learning object detectors from videos. Our method works with long video to automatically learn bounding box level annotations for multiple object instances. It does not assume exhaustive labeling of every object instance in the input videos, and from a handful of labeled instances can automatically label hundreds of thousands of instances.

learning process for images. We present an approach to constrain the semi-supervised learning process [370] in videos. Our technique constrains the SSL process by using *multiple weak cues* - appearance, motion, temporal etc., in video data and automatically learns *diverse new examples*.

Intuitively, algorithms dealing with videos should use appearance and temporal cues using detection and tracking, respectively. One would expect a simple combination of detection and tracking to constitute a semi-supervised framework that would prevent drift since both of these processes would ideally cancel each others' errors. However, as we show in our experiments (Sec. 2.3), a naïve combination of these two techniques performs poorly. In the long run, the errors in both detection and tracking are amplified in a coupled system. We can also consider pure detection approaches or pure tracking approaches for this problem. However, pure detection ignores temporal information while pure tracking tends to stray away over a long duration.

We present a scalable framework that discovers objects in video using SSL (see Figure 2.1). It tackles the challenging problem of localizing new object instances in long videos starting from only a few labeled examples. In addition, we present our algorithm in a realistic setting of "sparse labels" [305], i.e., in the few initial "labeled" frames, not all objects are annotated. This setting, traditonally called Positive Unlabeld (PU) learning [93, 247, 256], relaxes the assumption that in a given frame, all object instances have been exhaustively annotated. It implies that we do not know if any unannotated region in the frame belongs to the object category or the background, and thus cannot use any region from our input as negative data. While much of the past work has ignored this type of sparse labeling (and *lack of explicit negatives*), we show ways to overcome this handicap. Figure 2.2 presents an overview of our algorithm. Our proposed algorithm is different from the rich body of work on tracking-by-detection. Firstly, we do not attempt to solve the problem of *data association*. Our framework does not try to identify whether it has seen a particular instance before. Secondly, since it works in the regime of *sparse annotations*, it does not assume that negative data can be sampled from around the current box. Thirdly, we limit expensive computation to a subset of the input frames to scale to a million frames.

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

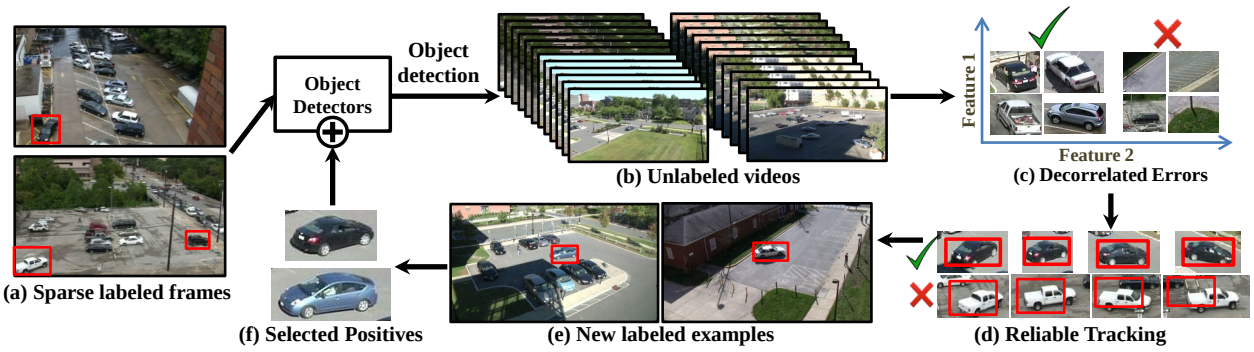


Figure 2.2: Our approach selects samples by iteratively discovering new boxes by a careful fusion of detection, robust tracking, relocalization and multi-view modeling of positive data. It shows how an interplay between these techniques can be used to learn from large scale unlabeled video corpora.

2.1 Background

The availability of web scale image and video data has made semi-supervised learning more popular in recent years. In the case of images, many methods [75, 113] rely on image similarity measures, and try to assign similar labels to close-by unlabeled images. However, in the case of real-world images, obtaining good image similarity is hard and hence the simple approaches become less applicable. One major body of work [57, 61, 83, 228, 370] tries to overcome this difficulty by leveraging the use of a set of pre-defined attributes for image classes [61, 370] and additionally web-supervision and text [57, 83]. While these methods have good performance for images, they are not applicable to videos mainly because they treat each image independently and do not use video constraints. One major reason for the success of attribute based methods for SSL was the relatively cheap supervision required for attributes (per image level tag). In the same spirit, weakly supervised video approaches use tags available with internet videos.

Weakly supervised video algorithms have gained popularity largely due to the abundance of video level tags on the internet. The input is a set of videos with video level tags (generally a video belongs to an object category), and the algorithm discovers the object (if present) in the video. These methods, while effective, assume a maximum of one dominant object per video [257, 326, 401, 431]. Some of them additionally assume dominant motion [326] or appearance saliency [242, 257, 326, 401, 431] for the object of interest. The methods of video co-segmentation [53, 128, 153, 201] can be considered a subset of weakly supervised methods. They make a strong assumption that multiple

videos contain the exact same object in majority of the frames. This assumption of at most one salient object in a video is rarely satisfied by internet or real world videos. When this assumption does not hold, methods cannot strongly rely on motion based foreground/background clustering or on appearance saliency. Our proposed work deals with *multiple objects* and can even discover *static* object instances without strongly relying on motion/appearance saliency. However, we do require richer bounding box annotations by way of a few *sparsely labeled* instances. A concurrent work [249] utilizes weakly labeled video for improving detection.

A relevant thread of work which also uses bounding box annotations is that of tracking-by-detection. It has a long and rich history in computer vision and the reader is referred to [312] for a survey. The tracking-by-detection algorithms start with bounding box annotation(s) of the object(s) to track the object(s) over a long period of time. The underlying assumption is that negative data can be sampled from around the object [26, 148, 162, 206, 355, 396, 404] to distinguish between the object and background. This is not valid in the case of *sparsely labeled* videos because the unmarked regions may contain more instances of the same object, rather than background.

Other tracking-by-detection methods [37, 135, 324] do not sample negative data from the input video. They do so at the cost of using a detector trained on additional training data, e.g., [324] uses a DPM [111] trained on images from PASCAL [95]. In contrast, we do not use such additional training data for our method. This allows us to work on object categories which are not in standard datasets.

Multi-object tracking-by-detection approaches also focus on solving the problem of *data association* - given object locations across multiple frames, determine which locations belong to the same object. Data association is critical for long term tracking, and is also very challenging [37]. In contrast, our goal is not to track over long periods, but to get short and reliable tracking. Moreover, we do not require these short tracklets to be associated with each other, and thus have minimal need for data association.

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

2.2 Approach

2.2.1 Approach Overview

There are two ways to detect objects in videos – either using detection in individual frames or tracking across frames. In a semi-supervised framework, detection plays the role of constraining

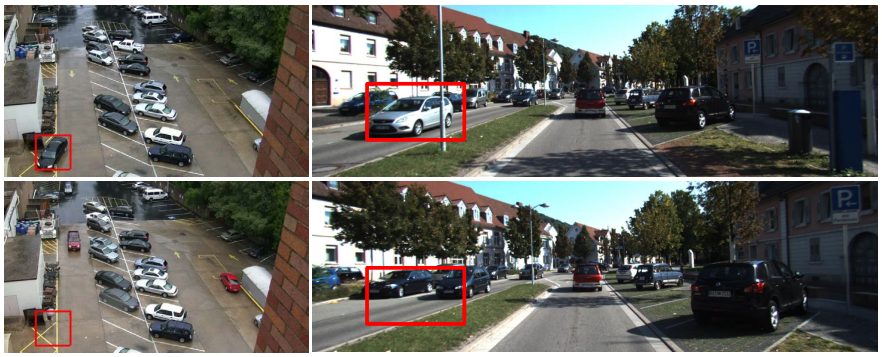


Figure 2.3: Sparsely labeled positives (as shown in the top row) are used to train Exemplar detectors [270]. Since we do not assume exhaustive labeling of each instance in the image, we cannot sample negative data from around the input boxes. When these detectors (trained without domain negatives) are used, they may learn background features (like the co-occuring yellow stripes or road divider) and give high confidence false positives (bottom row). We address this issue by exploiting decorrelated errors (Section 2.2.2)

the system by providing an appearance prior for the object, while tracking generalizes by providing newer views of the object. So one could imagine a detection and tracking combination, in which one tracks from confident detections and then updates the detector using the tracked samples. However, as we show in our experiments (Section 2.3), such a naïve combination does not impose enough constraints for SSL. In contrast, our approach builds on top of this basic combination of detection and tracking to correct their mistakes.

Our algorithm starts with a few sparsely annotated video frames ($\mathcal{L}$) and iteratively discovers new instances in the large unlabeled set of videos ($\mathcal{U}$). Simply put, we first train detectors on annotated objects, followed by detection on input videos. We determine good detections (removing confident false positives) which serve as starting points for short-term tracking. The short-term tracking aims to label new and unseen examples reliably. Amongst these newly labeled examples, we identify good and diverse examples which are used to update the detector without re-training from scratch. We iteratively repeat this fusion of tracking and detection to label new examples. We now describe our algorithm (illustrated in Figure 2.2).

Sparse Annotations (lack of explicit negatives): We start with a few sparsely annotated frames in a random subset of $\mathcal{U}$. Sparse labeling implies that unlike other approaches [206], we do not assume exhaustively annotated input, and thus cannot sample negatives from the vicinity of labeled positives. We use random images from the internet as negative data for training object detectors on these sparse labels [369]. We use these detectors to detect objects on a *subset of the video*, e.g., every 30 frames. Training on a few positives without domain negatives can result in high confidence false positives as shown in Figure 2.3. Removing such false positives is important because if we track them, we will add more bad training examples, thus degrading the detector's performance over iterations.

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

Decorrelated errors: To remove high confidence false positives (see Figure 2.3), we rely on the principle of *decorrelated errors* (similar to *multi-view* SSL [348, 379]). The intuition is that the detector makes mistakes that are related to its feature representation [428], and a different feature represen-

tation would lead to different errors. Thus, if the errors in different feature spaces are decorrelated, one can correct them and remove false positives. This gives us a filtered set of detections.

Reliable tracking: We track these filtered detections to label new examples. Our final goal is not to track the object over a long period. Instead, our goal is to track reliably and label new and hopefully diverse examples for the object detector. To get such reliable tracks we design a conservative *short-term tracking* algorithm that identifies *tracking failures*. Traditional tracking-by-detection approaches [135, 324] rely heavily on the detection prior to identify tracking failures. In contrast, the goal of our tracking is to improve the (weak) detector itself. Thus, heavily relying on input from the detector defeats the purpose of using tracking in our case.

Selection of diverse positives for updating the detector: The reliable tracklets give us a large set of automatically labeled boxes which we use to update our detector. Previous work [326] temporally subsamples boxes from videos, treating each box with equal importance. However, since these boxes come from videos, a large number of them are redundant and do not have equal importance for training our detector. Additionally, the relevance of an example added at the current iteration i depends on whether similar examples were added in earlier iterations. One would ideally want to train (make an *incremental update*) only on new and diverse examples, rather than re-train from scratch on thousands of largely redundant boxes. We address this issue by selection and show a way of training only on diverse, new boxes. After training detectors on diverse examples, we repeat the SSL process to iteratively label more examples.

Stopping criterion of SSL: It is desirable to have SSL algorithms which automatically determine when they should stop. We stop our SSL once our selection algorithm indicates that it does not have any good candidates to select.

Training procedure: We start with a small sparsely labeled set $\mathcal{L}_0$ of bounding boxes and unlabeled input videos $\mathcal{U}$. At each iteration i, using models trained on $\mathcal{L}_{i-1}$, we want to label new boxes in the input videos $\mathcal{U}$, add them to our labeled set $\mathcal{L} = \mathcal{L} \cup \mathcal{L}_i$, and iteratively repeat this procedure.

2.2.2 Detections with decorrelated errors

We train object detectors on our initial set of examples using random images from Flickr as negatives [369]. We detect on a uniformly sampled subset of the video frames and remove the temporally inconsistent detections.

Since the negative data for training the detectors comes from a different domain than the positives, we still get consistent false positive detections because the detector learns the wrong concept (see Figures 2.3 and 2.5). To remove such false positives, we perform outlier removal in a feature space different from that of the detector. The intuition is that the errors made by learning methods are correlated with their underlying feature representation, and thus using decorrelated feature spaces might help correct them.

For outlier removal, we use unconstrained Least Squares Importance Fitting (uLSIF) [207]. uLSIF uses definite inliers (our labeled set $\mathcal{L}_{i-1}$) to identify outliers in unknown data. It scales well with the size of the input and can be computed in closed form. These final filtered detections serve as starting points for reliable short term tracking.

2.2.3 Reliable Tracking

We formulate a scalable tracking procedure that effectively capitalizes on priors available from detection, color/texture consistency, objectness [11, 417] and optical flow. More importantly, our tracking procedure is very good at identifying its own failures. This property is vital in our semi-supervised framework since any tracking failure will add wrong examples to the labeled set leading to quick semantic drift (see Figure 2.5). The short-term tracking produces a set of labeled examples $\mathcal{L}_i$.

Our single object tracking computes sparse optical flow using Pyramidal Lucas Kanade [44] on Harris feature points. Since we start with a small set of labeled examples, and do not perform expensive per-frame detection, our detection prior is weak. To prevent tracking drift in such cases we incorporate color/texture consistency by using object proposal bounding boxes [417] obtained from a region around the tracked box. We address two failure modes of tracking:

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

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

We formulate the tracking problem as finding the min-cost path in a graph $\mathcal{G}$. At each frame we incorporate priors in terms of bounding boxes, i.e., detection bounding boxes, tracked boxes and object proposal bounding boxes. These boxes are the nodes in our graph and we connect nodes in consecutive frames with edges forming a trellis graph. The edge weights are a linear combination of the difference in dominant motions of the boxes (described above), spatial proximity and area change. Tracking through this trellis graph $\mathcal{G}$ is the equivalent of finding the single min-cost path, and is efficiently computed using Dynamic-Programming [37, 324]. As post-processing, we cut the path as soon as the total cost exceeds a set threshold.

2.2.4 Selection algorithm:

After we label thousands of boxes $\mathcal{L}_i$ for the current iteration, we use them for improving our object detectors. Since video data is highly redundant, we label few diverse examples and many redundant ones. Training a category detector on these thousands of (redundant) boxes, *from scratch*, in every iteration is suboptimal. We prefer an *incremental training* approach that makes incremental updates to the detector, i.e., trains only on *newly added and diverse* examples rather than everything. This is especially important to prevent drift because even if we label thousands of wrong but redundant boxes, our method picks only a few of them. We find the exemplar detectors [163, 270] suitable for incremental learning as they are trained per bounding box.

For each labeled bounding box in $\mathcal{L}_i$, we compute a detection signature [196, 283, 436] using

our exemplar detectors. Boxes where our current set of detectors do not give a high response correspond to examples which are not explained well by the existing set of detectors. Training on these boxes increases the coverage of our detectors. Thus, we compute similarity in this detection signature space to greedily select a set of boxes that are neither similar to our current detectors, nor amongst themselves.

More formally, let $\mathcal{L}_i = \{l_1, \ldots, l_k\}$ be the set of labeled boxes at iteration i and $\mathcal{E} = \cup_{j=0}^{i-1} \mathcal{D}_j$ the set of boxes b_n associated with the exemplar detectors from all previous iterations (0 to $i-1$). We compute the $|\mathcal{L}_i| \times |\mathcal{E}|$ detector response matrix R. The entry $R(m,n)$ indicates the response of the detector associated with box b_n on box l_m (also called the detection signature [196]). We row-normalize the matrix R, and denote the detection signature of box l_m by its row $R(m)$. We initialize the set $\mathcal{D}_i \subset \mathcal{L}_i$ with all the boxes l_j which have a low response, i.e., none of the detectors from the previous iterations are confident about detecting these boxes (ESVM score of < -0.8 and IOU < 0.4).

We iteratively grow the set $\mathcal{D}_i$ by adding detectors which minimize the following objective criterion

$$t^* = \underset{l_t \in \mathcal{L}_i}{\operatorname{argmax}} \sum_{b_p \in \mathcal{D}_i, p \neq t} \mathrm{JSD}(R(t)||R(p)) \tag{2.1}$$
$$D_i = D_i \cup \{l_{t^*}\}$$

This objective function favors boxes that are diverse with respect to the existing detectors using the Jensen-Shannon Divergence (JSD) of the detection signatures. At each iteration, we limit the number of boxes selected to 10. When this selection approach is unable to find new boxes, we conclude that we have reached the saturation point of our SSL. This serves as our *stopping criterion.*

2.3 Experiments

Our algorithm has a fair number of components interacting with each other across iterations. It is difficult to characterize the importance of each component by using the whole system at once. For such component-wise characterization, we divide our experiments in two sets. Our first set of ablative experiments is on a small subset of videos from VIRAT [305]. In the second set of experiments, we demonstrate the scalability of our approach to a million frames from [305]. We also show the generalization of our method to a different dataset (KITTI [134]). In both these cases we evaluate the automatically labeled data in terms of quality, coverage (recall), diversity and relevance to training an object detector. We now describe the experimental setup which is common across all our experiments.

Datasets: Due to limited availability of large video datasets with bounding box annotations, we picked car as our object of interest, and two video datasets with a large number of cars and related objects like trucks, vans etc. We chose the VIRAT 2.0 Ground [305] dataset for its large number of frames, and *sparsely* annotated bounding boxes over all the frames. This dataset consists of long hours of surveillance videos (static camera) of roads and parking lots. It has 329 videos ($\sim$1 million frames, and $\sim$6.5 million annotated bounding boxes (partial ground truth) of cars. We also evaluate on videos from the KITTI [134] dataset which were collected by a camera mounted on a moving car.

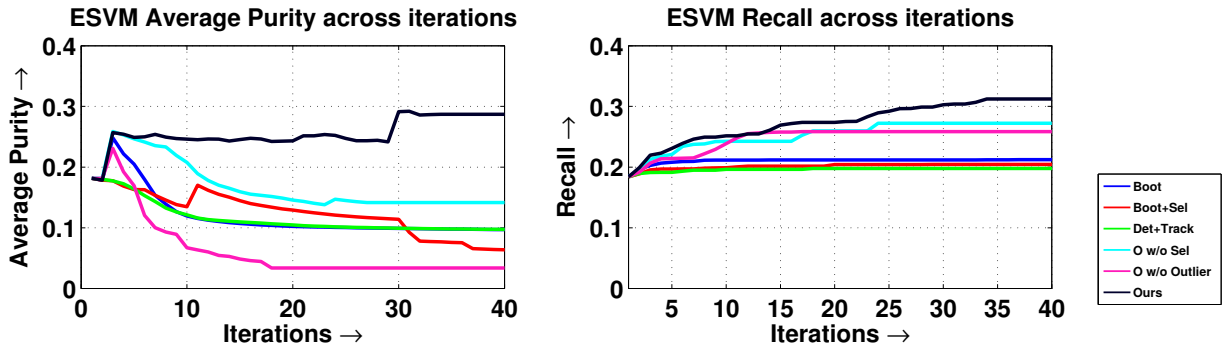


Figure 2.4: We measure the detection performance of the ESVMs from each ablation method on our test set by computing (left) Average Purity and (right) Recall.

Table 2.1: Our method vs. baselines (Section 2.3.1). We train an LSVM [111] on all the automatically labeled data and compute its detection performance on a held-out, fully annotated test set (AP for IOU 0.5).

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

We use the set 37 videos (∼12,500 frames) which have partial ground truth boxes (∼41,000 boxes, small cars are not annotated).

Dataset characteristics: We chose these datasets with very different characteristics (motion, size of object etc.) to test the generalization of our method. The VIRAT and KITTI datasets both consist of outdoor scene videos with a static and moving camera respectively. The VIRAT dataset captures surveillance videos of multiple cars in parking lots. The cars in this dataset are small compared to the frame size, tightly packed together and viewed from a distance (thus no drastic perspective effects). The KITTI dataset on the other hand, consists of videos taken by a vehicle mounted camera. It has high motion, large objects, and perspective effects. Figure 2.3 shows examples from both the datasets demonstrating their visual differences.

Detectors: We use the Exemplar-SVM (ESVM) [270] detectors with 5000 random images from Flickr as negatives [369]. Since per frame detection is expensive, we detect once every 30^{th} frame for VIRAT, and every 10^{th} frame for KITTI. We threshold detections at SVM score of -0.75.

Multiple feature spaces for false positive removal: We use the uLSIF [207] algorithm on Pyramidal HOG (PHOG) [41] and color histogram (LAB color with $32 \times 16 \times 16$ bins) features computed on a resized box of 200×300 px. We set the kernel bandwidth for uLSIF by computing the 75^{th} percentile distance between random pairs of points.

Object proposal windows: We obtain 2000 windows per image using selective search [417].

Figure 2.5: (a) We look at a subset of the bounding boxes used to train ESVMs across iteration. Each row corresponds to an ablation method. The top row shows the randomly chosen initial positive bounding boxes (same for each method). The other methods diverge quickly across iterations, thus showing that constraints are very important for maintaining purity.

2.3.1 Ablative Analysis of each constraint

To tease apart the contributions of each component described in Section 2.2.1, we design a set of algorithms using only a subset of the components.

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

Bootstrapping with Selection (Boot+Sel): This algorithm builds upon the bootstrapping approach described above. However, diverse examples are selected (Section 2.2.4) from confident detections to train new detectors.

Detection, Tracking and Clustering (Det+Track): In this algorithm, we use a basic combination of detection and tracking. We start tracking from the confident ESVM detections to label new examples. We then use WHO (or ELDA) [163] clustering [283] on these boxes to select training examples for the new detectors. For clustering, we use WHO features on labeled boxes after resizing, followed by k-means. We choose the best k in the range $(5, 10)$.

Ours without outlier (O w/o Outlier): This setup uses our entire algorithm except outlier removal (Section 2.2.2). It can roughly be thought of as Detection+Tracking+Selection.

Ours without selection (O w/o Sel): This algorithm uses our algorithm except selection (Section 2.2.4). It uses WHO clustering for selection like Det+Track.

Ours: We use our full algorithm as detailed in Sections 2.2.1.

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

Qualitative Results

Figure 2.5 shows a random set of boxes labeled by each ablation method (and used to train ESVMs for that method), along with the initial set of 21 positive examples. We notice that as iterations proceed, the labeling quality (especially "tightness" of the boxes) for all methods degrades. More importantly, the other methods like Boot, Det+Track etc. show semantic drift (Figure 2.5 columns

2-5 at iteration 20). We also notice the importance of selection, as Ours w/o Selection loses good localization ability fairly quickly (Figure 2.5 column 6 at iterations 10-30). We believe methods like [76] can be used to further improve the labeling quality.

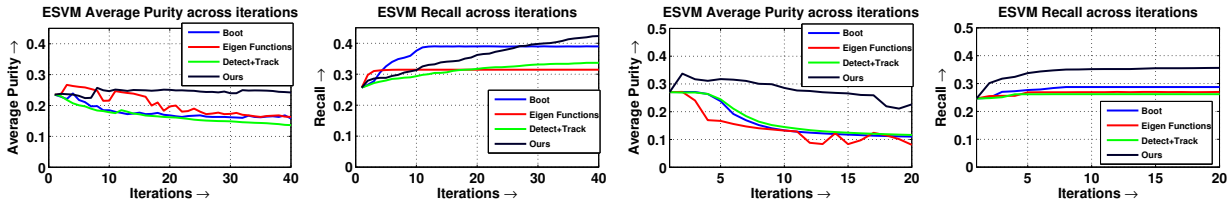
ESVM Detection performance

For the input videos, we cannot measure labeling purity because of partial ground truth (refer to supplementary material for an approximation of purity). Instead, we measure the relevance of labeled boxes to detection. We consider detection performance on the test set as a proxy for good labeling. We test the ESVMs selected and trained by each method across iterations on the held out test set. A good labeling would result in an increased detection performance. Figure 2.4 shows Average Purity vs. Recall across iterations for the various methods on the test set. We use Average Purity, which is same as Average Precision [95] but does not penalize double-detections, since we are more interested in whether the ESVMs are good detectors individually, rather than as an ensemble.We consider an instance correctly labeled (or pure) if its Intersection-Over-Union (IOU) with any ground-truth instance is greater than 0.3. Our method shows a higher purity and recall, pointing towards meaningful labeling and selection of the input data. It also shows that every component of our method is crucial for getting good performance. We stop our method at iteration 40 because of our stopping criterion (Section 2.2.4). We got a 2 point drop in purity from iteration 40 to 45, proving the validity of our stopping criterion. This is important since our algorithm would rather saturate than label noisy examples.

Training on all the automatically labeled data

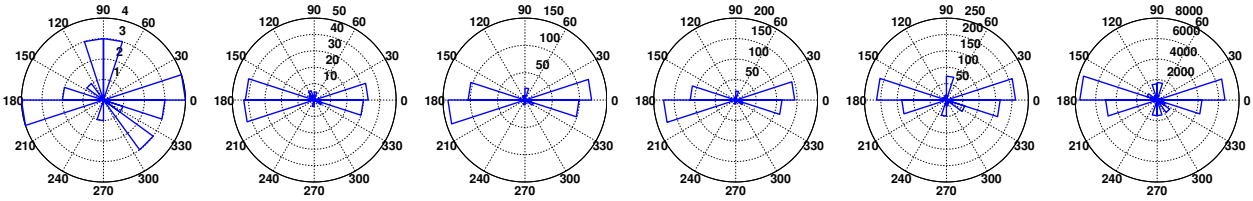
In this section, we evaluate the effectiveness of all our labeled data. For each algorithm, we **train an LSVM** [111] (only root filters, mixtures and positive latent updates of DPM [111]) on the data it labeled, and test it on the held-out test set. Since it is computationally expensive to train an LSVM on the thousands of boxes, we subsample the labeled boxes (5000 boxes in total for each method using k-means on WHO [163] features). We sample more boxes from the earlier iterations of each method, because their labeling purity decreases across iterations (Figure 2.4). We use the same domain independent negatives [369] for all these LSVMs (left side in Table 2.1). Table 2.1 shows the detection AP performance of all **LSVMs** (measured at IOU of 0.5 [95]) for the data labeled at iteration 10 and 30. We see that LSVM trained on our labeled data outperforms all other LSVMs. Our performance is close to that of an LSVM trained on the PASCAL VOC 2007 dataset [95]. This validates the high quality of our automatically labeled data.

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



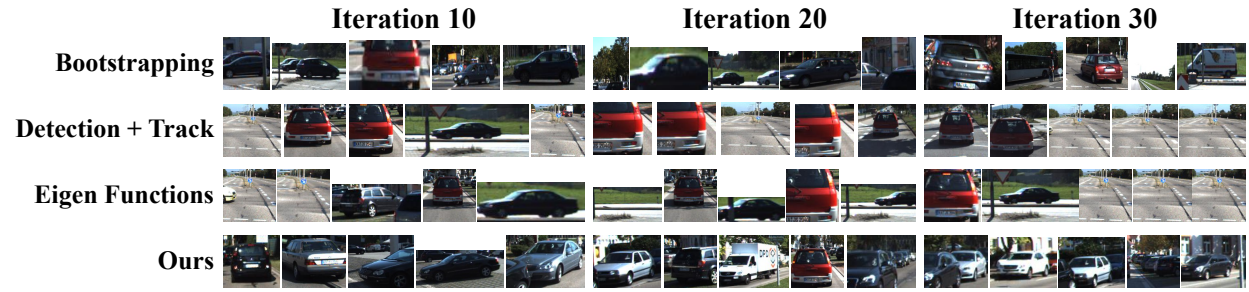
(a) Purity on KITTI (b) Recall on KITTI (c) Purity on VIRAT (d) Recall on VIRAT

Figure 2.6: We measure the detection performance of the labeled boxes for our large scale experiments. We test the ESVMs trained at each iteration on the held out test set and compute Average Purity and Recall. Our method outperforms the baselines by a significant margin. It maintains purity while substantially increasing recall.

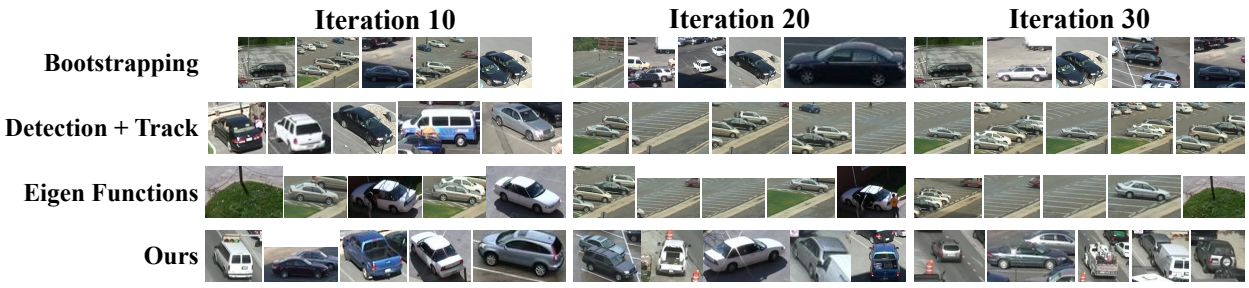


(a) Initial boxes (b) Bootstrapping (c) Eigen Functions (d) Detect+Track (e) Ours (f) Ground truth

Figure 2.7: Pose variation in automatic labeling of the KITTI dataset. For each algorithm, we plot the 3D pose distribution of all the boxes it labels after 30 iterations. The first and last plots show pose distribution for the initial labeled boxes and all boxes in ground truth respectively. The distribution of boxes labeled by our method is close to the ground truth distribution.



(a) KITTI Dataset



(b) VIRAT Dataset

Figure 2.8: We look at the selected positives for each baseline method across iterations for both KITTI and VIRAT datasets. We notice that the purity of the labeled set drops significantly as iterations proceed. This indicates that constraints specific to video are needed to learn meaningfully over iterations.

2.3.2 Large scale experiments

In this section, we evaluate the scalability of our algorithm to millions of frames and object instances. We also test its generalization on two datasets with widely different characteristics - VIRAT (static camera, small cars tightly packed) and KITTI (moving camera, high motion, large cars with perspective effects). We use the *Boot* and *Ours* methods described in Section 2.3.1. As we described in Section 2.1, most of the existing approaches make additional assumptions that are not applicable in our setting, or do not have publicly available code. To make a fair comparison against existing methods, we adapt them to our setting.

Baseline - Dense detection and association (Detect + Track): This algorithm is inspired by Geiger et al. [135] which has state-of-the-art results on KITTI. The original algorithm uses per-frame detections, followed by a Kalman filter and data association. We make two changes - 1) To have a comparable detector across methods, we do not use a pre-trained DPM on PASCAL VOC 2007. We substitute it with the ESVMs we have at the current iteration. 2) We do not use a Kalman filter and use data association over a short term (maximum 300 frames). We select positives for the ESVMs by k-means on WHO features [283].

Baseline - Eigen Functions: We modify the Eigen functions [113] method which was originally designed for image classification. This method uses distances over manifolds to label unlabeled data. We use basic detection and short term tracking to get a set of bounding boxes and use eigen functions to classify them as positive or background. The boxes from previous iterations on which we trained ESVMs are used as positives and random images from Flickr [369] as negative data. We use color histogram ($32 \times 16 \times 16$ LAB space) and PHOG [41] features as input to eigen functions.

Datasets: Our input set consists of 312 videos ($\sim$820,000 frames) from VIRAT. We take a held out test set of 17 videos ($\sim$105,000 frames) which we fully annotated. As input, all algorithms start with the same sparse labels consisting of 43 randomly chosen bounding boxes across different videos. For the KITTI dataset we use 30 videos ($\sim$10,000 frames) as our input and 7 videos ($\sim$2000 frames) for testing. All algorithms start with the same sparse labeled 25 bounding boxes from different videos.

Qualitative Results: We first present qualitative results in Figure 2.8. We notice the same trends as we did in the ablation analysis, namely, bounding boxes tend to get less tight across iterations. For the baseline methods, we notice quick divergence as an increasing number of background patches are classified as car.

ESVM Detection Performance: Following the approach outlined in Section 2.3.1, we compute the detection performance of the ESVMs on the held out test set. This helps us measure the relevance of our labeling to the detection task. Figure 2.6 shows the results of these experiments. We notice that our method outperforms the baselines on both the metrics (Average Purity and Recall). This reinforces the fact that our constraints help arrest semantic drift.

Diversity of labeling: The KITTI dataset provides the 3D pose associated with each labeled car. We use this 3D pose as a proxy for estimating the diversity of our labeled set. In this experiment, we compute the pose of the examples labeled by all methods. Figure 2.7 demonstrates that our labeling procedure covers a diverse range of poses as opposed to baseline methods. The pose distribution of our labeling is closer to the ground truth distribution, while that of the baselines

prefers the more "popular" poses, i.e., front/back of cars. Combined with the results of Figure 2.6, this points towards a diverse, and high quality labeling of the data.

2.4 Discussion

In this chapter, we introduced a semi-supervised learning technique for training object detectors from videos. Our technique addresses the detection of multiple objects without assuming exhaustive labeling of object instances in any input frame. In addition, we introduce constraints like decorrelated errors, reliable tracking and diverse selection which are effective in arresting semantic drift. Our experiments show that such an SSL approach can start with a handful of labeled examples and label hundreds of thousands of new examples which also improve object detectors. In the next chapter, we will see how such temporal constraints can be used to train powerful feature representations again by simply 'observing' videos.

Chapter 3

Temporal Constraints for Representation Learning

How is complex sequential material acquired, processed and represented when there is no intention to learn?

AXEL CLEEREMANS AND JAMES MCCLELLAND in *Learning the Structure of Event Sequences*, 1991

The previous Chapter 2 showed how a careful combination of tracking and detection can create a powerful semi-supervised learning algorithm. This method still needs access to a few human labels in order to start this tracking. In this chapter we want to explore the information available in the temporal structure in videos *without any* explicit supervision. Cleeremans and McClelland (quoted at the top) expressed wonder at the remarkable ability of humans to learn from sequential events without any real 'goal' or 'direction' (supervision). In their experiments on human subjects, they showed that exposure to sequential information increases sensitivity to temporal context, provides priming and the temporal structure is encoded by anticipation of elements in the sequence. Our work can be viewed as a computational framework that draws inspiration from their results.

What can simply watching the world in motion teach us? Videos provide an abundant source of sequential data and temporal supervision. To study the information available in a video signal in isolation, we ask the question: How does an agent learn from the spatiotemporal structure present in video without using supervised semantic labels? Are the representations learned using the unsupervised spatiotemporal information present in videos meaningful? And finally, are these representations complementary to those learned from strongly supervised image data? In this chapter, we explore such questions by using a sequential learning approach.

Sequential data provides an abundant source of information in the form of auditory and visual percepts. Learning from the observation of sequential data is a natural and implicit process for humans [64, 65, 334]. It informs both low level cognitive tasks and high level abilities like decision making and problem solving [395]. For instance, answering the question "Where would the moving ball go?", requires the development of basic cognitive abilities like prediction from sequential data like video [27].

Sequential learning is used in a variety of areas such as speech recognition, robotic path planning, adaptive control algorithms, *etc*. These approaches can be broadly categorized [394] into

two classes: prediction and verification. In sequential prediction, the goal is to predict the signal given an input sequence. A popular application of this in Natural Language Processing (NLP) is 'word2vec' by Mikolov *et al*. [279, 280] that learns distributional representations [118]. Using the continuous bag-of-words (CBOW) task, the model learns to predict a missing word given a sequence of surrounding words. The representation that results from this task has been shown to be semantically meaningful [279]. Unfortunately, extending the same technique to predict video frames is challenging. Unlike words that can be represented using limited-sized vocabularies, the space of possible video frames is extremely large [85], *e.g.*, predicting pixels in a small 256×256 image leads to $256^{2\times3\times256}$ hypotheses! To avoid this complex task of predicting high-dimensional video frames, we use sequential verification.

In sequential verification, one predicts the 'validity' of the sequence, rather than individual items in the sequence. In this chapter, we explore the task of determining whether a given sequence is 'temporally valid', *i.e.*, whether a sequence of video frames are in the correct temporal order, Figure 3.1. We demonstrate that this binary classification problem is capable of learning useful visual representations from videos. Specifically, we explore their use in the well understood tasks of human action recognition and pose estimation. But why are these simple sequential verification tasks useful for learning? Determining the validity of a sequence requires reasoning about object transformations and relative locations through time. This in turn forces the representation to capture object appearances and deformations.

We use a Convolutional Neural Network (CNN) [239] for our underlying feature representation. The CNN is applied to each frame in the sequence and trained "end-to-end" from random initialization. The sequence verification task encourages the CNN features to be both visually and temporally grounded. We demonstrate the effectiveness of our unsupervised method on benchmark action recognition datasets UCF101 [384] and HMDB51 [224], and the FLIC [358] and MPII [20] pose estimation datasets. Using our simple unsupervised learning approach for pre-training, we show a significant boost in accuracy over learning CNNs from scratch with random initialization. In fact, our unsupervised approach even outperforms pre-training with some supervised training datasets. In action recognition, improved performance can be found by combining existing supervised image-based representations with our unsupervised representation. By training on action videos with humans, our approach learns a representation sensitive to human pose. Remarkably, when applied to pose estimation, our representation is competitive with pre-training on significantly larger supervised training datasets [351].

3.1 Background

Our work uses unlabeled video sequences for learning representations. Since this source of supervision is 'free', our work can be viewed as a form of unsupervised learning. Unsupervised representation learning from single images is a popular area of research in computer vision. A significant body of unsupervised learning literature uses hand-crafted features and clustering based approaches to discover objects [99, 352, 382], or mid-level elements [84, 204, 246, 381, 393]. Deep learning methods like auto-encoders [36, 306, 425], Deep Boltzmann Machines [356], variational methods [216, 342], stacked auto-encoders [34, 240], and others [237, 435] learn representations directly from images. These methods learn a representation by estimating latent parameters that help reconstruct the data, and may regularize the learning process by priors such as sparsity [306].

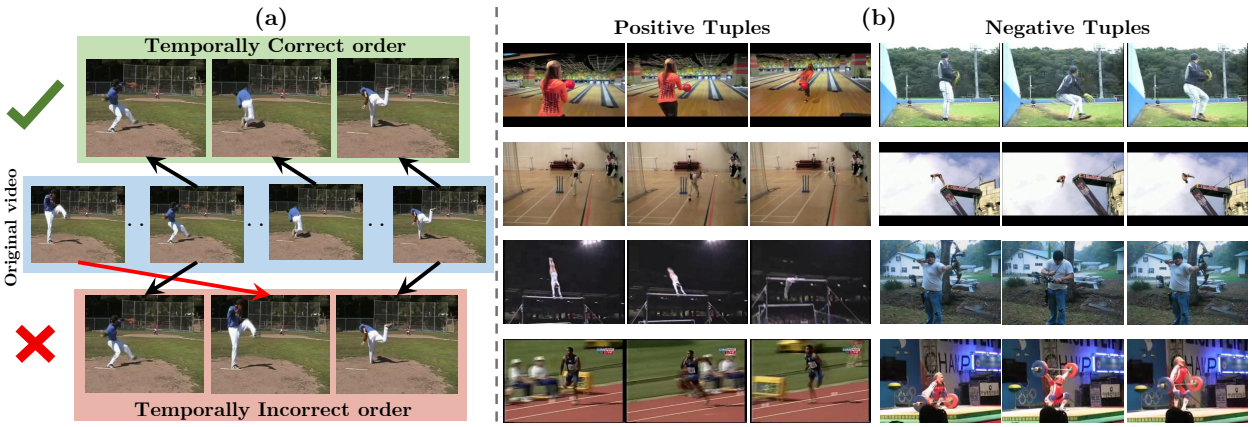


Figure 3.1: **(a)** A video imposes a natural temporal structure for visual data. In many cases, one can easily verify whether frames are in the correct temporal order (shuffled or not). Such a simple sequential verification task captures important spatiotemporal signals in videos. We use this task for unsupervised pre-training of a Convolutional Neural Network (CNN). **(b)** Some examples of the automatically extracted positive and negative tuples used to formulate a classification task for a CNN.

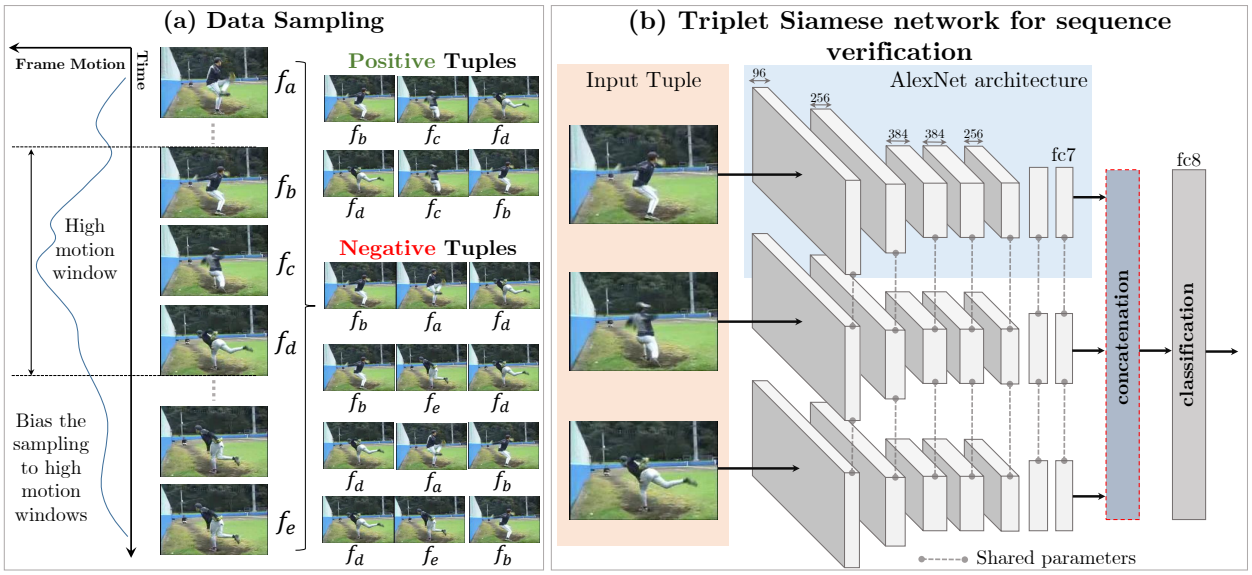


Figure 3.2: **(a)** We sample tuples of frames from high motion windows in a video. We form positive and negative tuples based on whether the three input frames are in the correct temporal order. **(b)** Our triplet Siamese network architecture has three parallel network stacks with shared weights upto the `fc7` layer. Each stack takes a frame as input, and produces a representation at the `fc7` layer. The concatenated `fc7` representations are used to predict whether the input tuple is in the correct temporal order.

Techniques in [85, 363] scale unsupervised learning to large image datasets showing its usefulness for tasks such as pedestrian detection [363] and object detection [85]. In terms of using 'context' for learning, our work is most similar to [85] which uses the spatial context in images. While these approaches are unsupervised, they do not use videos and cannot exploit the temporal structure in them. Our work is most related to work in unsupervised learning from videos [157, 188, 192, 193, 296]. Traditional methods in this domain utilize the spatiotemporal continuity as regularization for the learning process. Since visual appearance changes smoothly in videos, a

common constraint is enforcing temporal smoothness of features [121, 147, 157, 296, 440]. Zhang *et al*. [465], in particular, show how such constraints are useful for action recognition. Moving beyond just temporal smoothness, [193] enforces additional 'steadiness' constraints on the features so that the change of features across frames is meaningful. Our work, in contrast, does not explicitly impose any regularizations on the features. Other reconstruction-based learning approaches include that of Goroshin *et al*. [147] who use a generative model to predict video frames and Srivastava *et al*. [387] who use LSTMs [173]. Unlike our method, these works [147, 296, 387, 403] explicitly predict individual frames, but do not explore large image sizes or datasets. [430, 474] also consider the task of predicting the future from videos, but consider it as their end task and do not use it for unsupervised pre-training.

Several recent chapters [8, 192, 474] use egomotion constraints from video to further constrain the learning. Jayaraman *et al*. [192] show how they can learn equivariant transforms from such constraints. Similar to our work, they use full video frames for learning with little pre-processing. Owens *et al*. [310] use audio signals from videos to learn visual representations. Another line of work [434] uses video data to mine patches which belong to the same object to learn representations useful for distinguishing objects. Typically, these approaches require significant pre-processing to create this task. While our work also uses videos, we explore them in the spirit of sequence verification for action recognition which learns from the raw video with very little pre-processing.

We demonstrate the effectiveness of our unsupervised pre-training using two extensively studied vision tasks - action recognition and pose estimation. These tasks have well established benchmark datasets [20, 224, 358, 384]. As it is beyond the scope of this chapter, we refer the reader to [325] for a survey on action recognition, and [323] for a survey on pose estimation.

3.2 Approach

Our goal is to learn a feature representation using only the raw spatiotemporal signal naturally available in videos. We learn this representation using a sequential verification task and focus on videos with human actions. Specifically, as shown in Figure 3.1, we extract a tuple of frames from a video, and ask whether the frames are in the correct temporal order. In this section, we begin by motivating our use of sequential tasks and how they use the temporal structure of videos. We then describe how positive and negative tuples are sampled from videos, and describe our model.

3.2.1 Task motivation

When using only raw videos as input, sequential verification tasks offer a promising approach to unsupervised learning. In addition to our approach described below, several alternative tasks are explored in Section 3.4.2. The goal of these tasks is to encourage the model to reason about the motion and appearance of the objects, and thus learn the temporal structure of videos. Example tasks may include reasoning about the ordering of frames, or determining the relative temporal proximity of frames. For tasks that ask for the verification of temporal order, how many frames are needed to determine a correct answer? If we want to determine the correct order from just two frames, the question may be ambiguous in cases where cyclical motion is present. For example,

consider a short video sequence of a person picking up a coffee cup. Given two frames the temporal order is ambiguous; the person may be picking the coffee cup up, or placing it down.

To reduce such ambiguity, we propose sampling a three frame tuple, and ask whether the tuple's frames are correctly ordered. While theoretically, three frames are not sufficient to resolve cyclical ambiguity [365], we found that combining this with smart sampling (Section 3.2.2) removes a significant portion of ambiguous cases. We now formalize this problem into a classification task. Consider the set of frames $\{f_1, \ldots, f_n\}$ from an unlabeled video $\mathcal{V}$. We consider the tuple (f_b, f_c, f_d) to be in the correct temporal order (class 1, positive tuple) if the frames obey either ordering $b < c < d$ or $d < c < b$, to account for the directional ambiguity in video clips. Otherwise, if $b < d < c$ or $c < b < d$, we say that the frames are not in the correct temporal order (class 0, negative tuple).

3.2.2 Tuple sampling

A critical challenge when training a network on the three-tuple ordering task is how to sample positive and negative training instances. A naive method may sample the tuples uniformly from a video. However, in temporal windows with very little motion it is hard to distinguish between a positive and a negative tuple, resulting in many ambiguous training examples. Instead, we only sample tuples from temporal windows with high motion. As Figure 3.2 shows, we use coarse frame level optical flow [107] as a proxy to measure the motion between frames. We treat the average flow magnitude per-frame as a weight for that frame, and use it to bias our sampling towards high motion windows. This ensures that the classification of the tuples is not ambiguous. Figure 3.1 (b) shows examples of such tuples.

To create positive and negative tuples, we sample five frames $(f_a, f_b, f_c, f_d, f_e)$ from a temporal window such that $a < b < c < d < e$ (see Figure 3.2 (a)). Positive instances are created using (f_b, f_c, f_d), while negative instances are created using (f_b, f_a, f_d) and (f_b, f_e, f_d). Additional training examples are also created by inverting the order of all training instances, *e.g.*, (f_d, f_c, f_b) is positive. During training it is critical to use the same beginning frame f_b and ending frame f_d while only changing the middle frame for both positive and negative examples. Since only the middle frame changes between training examples, the network is encouraged to focus on this signal to learn the subtle difference between positives and negatives, rather than irrelevant features.

To avoid sampling ambiguous negative frames f_a and f_e, we enforce that the appearance of the positive f_c frame is not too similar (measured by SSD on RGB pixel values) to f_a or f_e. These simple conditions eliminated most ambiguous examples. We provide further analysis of sampling data in Section 3.3.1.

3.2.3 Model Parametrization and Learning

To learn a feature representation from the tuple ordering task, we use a simple triplet Siamese network. This network has three parallel stacks of layers with shared parameters (Figure 3.2). Every network stack follows the standard CaffeNet [195] (a slight modification of AlexNet [221]) architecture from the `conv1` to the `fc7` layer. Each stack takes as input one of the frames from the

tuple and produces a representation at the `fc7` layer. The three `fc7` outputs are concatenated as input to a linear classification layer. The classification layer can reason about all three frames at once and predict whether they are in order or not (two class classification). Since the layers from `conv1` to `fc7` are shared across the network stacks, the Siamese architecture has the same number of parameters as AlexNet barring the final `fc8` layer. We update the parameters of the network by minimizing the regularized cross-entropy loss of the predictions on each tuple. While this network takes three inputs at training time, during testing we can obtain the `conv1` to `fc7` representations of a single input frame by using just one stack, as the parameters across the three stacks are shared.

3.3 Empirical ablation analysis

In this section (and in the Appendix), we present experiments to analyze the various design decisions for training our network. In Sections 3.4 and 3.5, we provide results on both action recognition and pose estimation.

Dataset: We report all our results using split 1 of the benchmark UCF101 [384] dataset. This dataset contains videos for 101 action categories with ~ 9.5k videos for training and ~ 3.5k videos for testing. Each video has an associated action category label. The standard performance metric for action recognition on this dataset is classification accuracy.

Details for unsupervised pre-training: For unsupervised pre-training, we do not use the semantic action labels. We sample about 900k tuples from the UCF101 training videos. We randomly initialize our network, and train for 100k iterations with a fixed learning rate of 10^{-3} and mini-batch size of 128 tuples. Each tuple consists of 3 frames. Using more (4, 5) frames per tuple did not show significant improvement. We use batch normalization [186].

Details for Action Recognition: The spatial network from [376] is a well-established method of action recognition that uses only RGB appearance information. The parameters of the spatial network are initialized with our unsupervised pre-trained network. We use the provided action labels per video and follow the training and testing protocol as suggested in[376, 432]. Briefly, for training we form mini-batches by sampling random frames from videos. At test time, 25 frames are uniformly sampled from each video. Each frame is used to generate 10 inputs after fixed cropping and flipping (5 crops $\times$ 2 flips), and the prediction for the video is an average of the predictions across these 25×10 inputs. We use the CaffeNet architecture for its speed and efficiency. We initialize the network parameters up to the `fc7` layer using the parameters from the unsupervised pre-trained network, and initialize a new `fc8` layer for the action recognition task. We finetune the network following [376] for 20k iterations with a batch size of 256, and learning rate of 10^{-2} decaying by 10 after 14k iterations, using SGD with momentum of 0.9, and dropout of 0.5. While [376] used the wider VGG-M-2048 [52] architecture, we found that their parameters transfer to CaffeNet because of the similarities in their architectures.

Table 3.1: We study the effect of our design choices such as temporal sampling parameters, and varying class ratios for unsupervised pre-training. We measure the tuple prediction accuracy on a held out set from UCF101. We also show action classification results after finetuning the models on the UCF101 action recognition task (split 1).

(a) Varying temporal sampling

$\tau_{\max}$	$\tau_{\min}$	Tuple Pred.	Action Recog.
30	15	60.2	47.2
60	15	**72.1**	**50.9**
60	60	64.3	49.1

(b) Varying class ratios

Class Ratio		Tuple Pred.	Action Recog.
Neg	Pos		
0.5	0.5	52.1	38.1
0.65	0.35	68.5	45.5
0.75	0.25	**72.1**	**50.9**
0.85	0.15	67.7	48.6

3.3.1 Sampling of data

In this section we study the impact of sampling parameters described in Section 3.2.2 on the unsupervised pre-training task. We denote the maximum distance between frames of positive tuples by $\tau_{\max} = |b - d|$. This parameter controls the 'difficulty' of positives: a very high value makes it difficult to see correspondence across the positive tuple, and a very low value gives almost identical frames and thus very easy positives. Similarly, we compute the minimum distance between the frames f_a and f_e used for negative tuples to the other frames by $\tau_{\min} = \min(|a - b|, |d - e|)$. This parameter controls the difficulty of negatives with a low value making them harder, and a high value making them easier.

We compute the training and testing accuracy of these networks on the tuple prediction task on held out videos. This held out set is a union of samples using all the temporal sampling parameters. We show results in Table 3.1 (a). We also use these networks for finetuning on the UCF101 action recognition task. Our results show that the tuple prediction accuracy and the performance on the action recognition task are correlated. A large temporal window for positive sampling improves over a smaller temporal window (Rows 1 and 2), while a large window for negative sampling hurts performance (Rows 2 and 3).

3.3.2 Class ratios in mini-batch

Another important factor when training the model is the class ratios in each mini-batch. As has been observed empirically [139, 340], a good class ratio per mini-batch ensures that the model does not overfit to one particular class, and helps the learning process. For these experiments, we choose a single temporal window for sampling and vary only the ratio of positive and negative tuples per mini-batch. We compare the accuracy of these networks on the tuple prediction task on held out videos in Table 3.1 (b). Additionally, we report the accuracy of these networks after finetuning on the action recognition task. These results show that the class ratio used for unsupervised pre-training can significantly impact learning. It is important to have a larger percentage of negative examples.

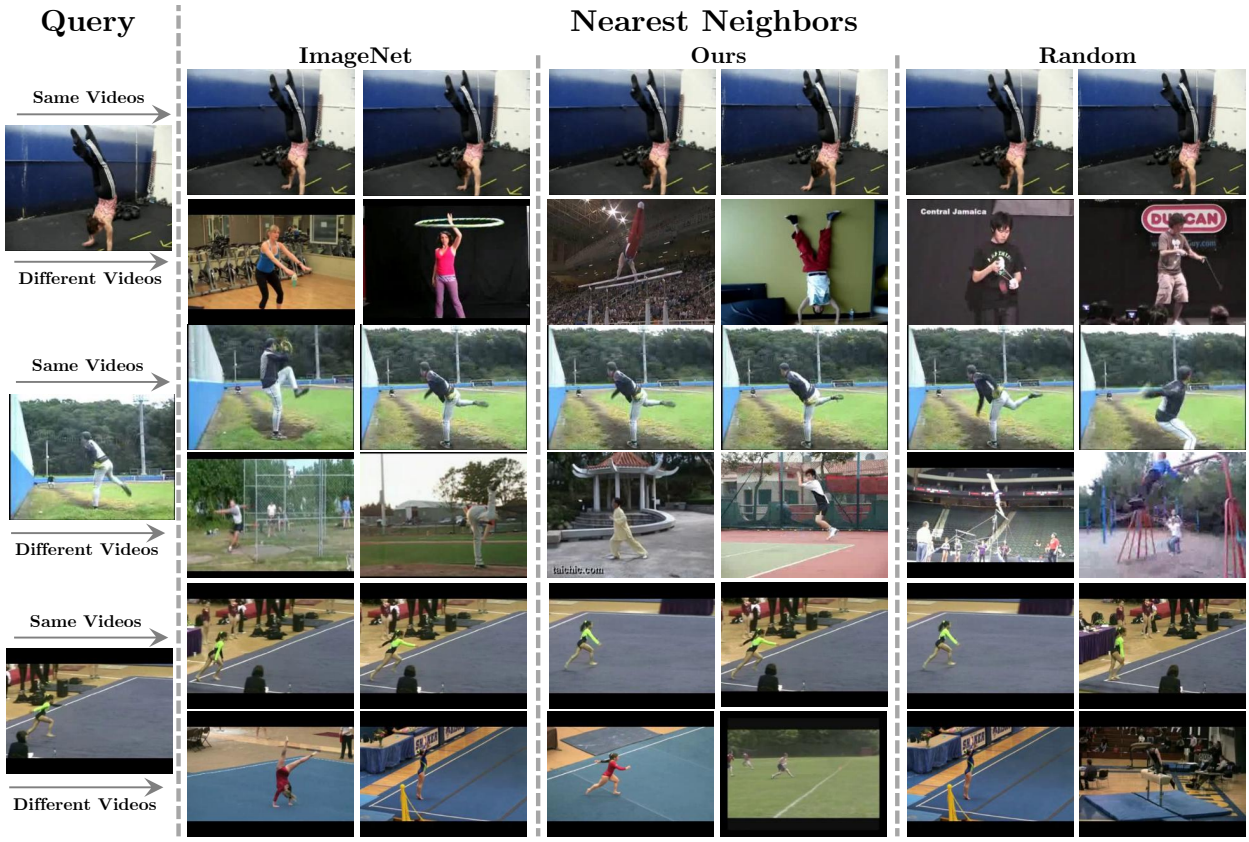


Figure 3.3: We compute nearest neighbors using `fc7` features on the UCF101 dataset. We compare these results across three networks: pre-trained on ImageNet, pre-trained on our unsupervised task and a randomly initialized network. We choose a input query frame from a clip and retrieve results from other clips in the dataset. Since the dataset contains multiple clips from the same video we get near duplicate retrievals (first row). We remove these duplicates, and display results in the second row. While ImageNet focuses on the high level semantics, our network captures the human pose.

3.3.3 What does the temporal ordering task capture?

Nearest Neighbor retrieval

We retrieve nearest neighbors using our unsupervised features on the UCF101 dataset and compare them in Figure 3.3 to retrievals by the pre-trained ImageNet features, and a randomly initialized network. Additional examples are shown in the supplementary materials. We pick an input query frame from a clip and retrieve neighbors from other clips in the UCF101 dataset. Since the UCF101 dataset has clips from the same video, the first set of retrievals (after removing frames from the same input clip) are near duplicates which are not very informative (notice the random network's results). We remove these near-duplicates by computing the sum of squared distances (SSD) between the frames, and display the top results in the second row of each query. These results make two things clear: 1) the ImageNet pre-trained network focuses on scene semantics 2) Our unsupervised pre-trained network focuses on the pose of the person. This would seem to indicate that the information captured by our unsupervised pre-training is complementary to that of ImageNet. Such behavior is not surprising, if we consider our network was trained without

semantic labels, and must reason about spatiotemporal signals for the tuple verification task.

Visualizing `pool5` unit responses

We analyze the feature representation of the unsupervised network trained using the tuple prediction task on UCF101. Following the procedure of [141] we show the top regions for `pool5` units alongwith their receptive field in Figure 3.4. This gives us insight into the network's internal feature representation and shows that many units show preference for human body parts and pose. This is not surprising given that our network is trained on videos of human action recognition, and must reason about human movements for the tuple ordering task.

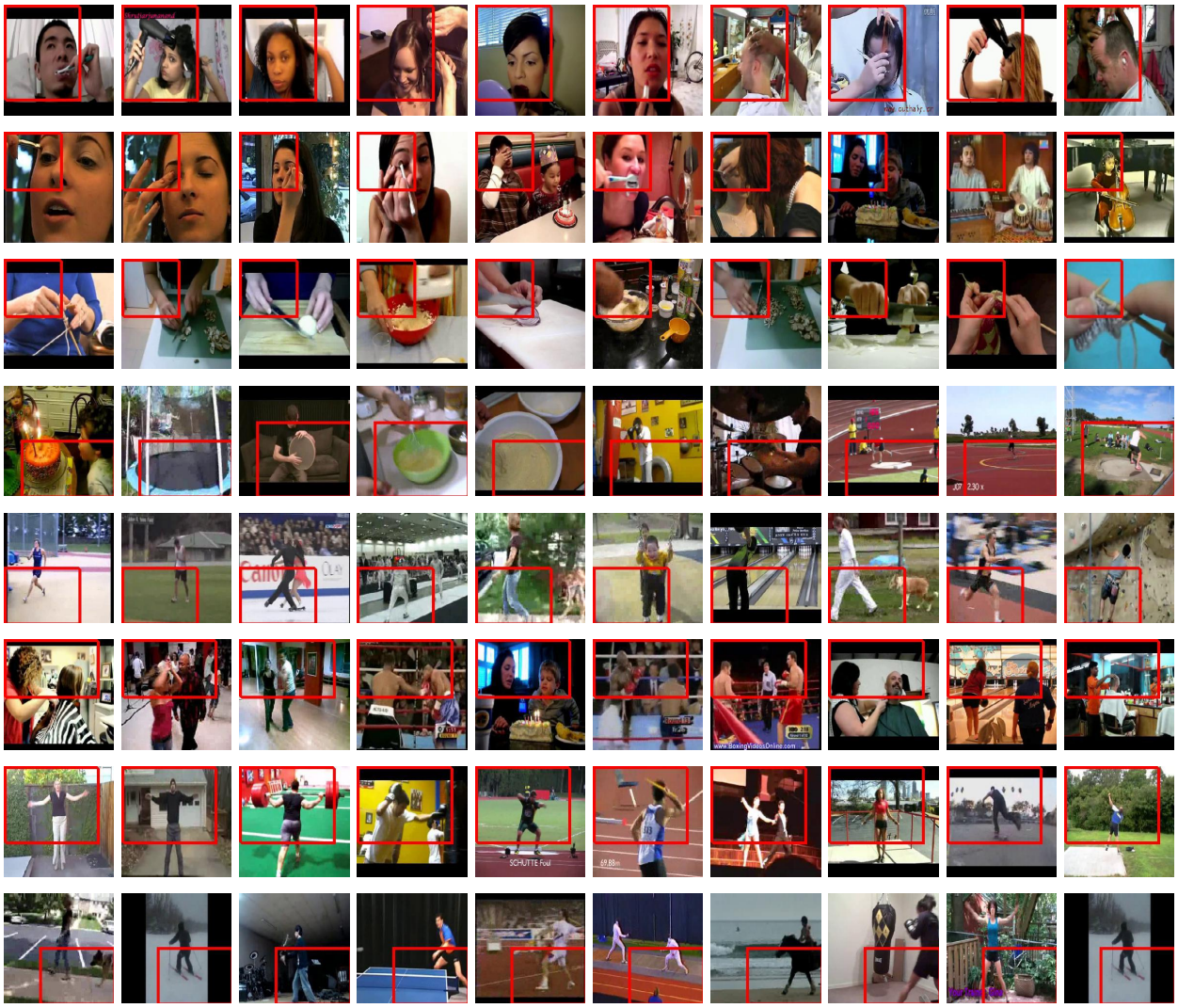


Figure 3.4: In each row we display the top image regions for a unit from the `pool5` layer. We follow the method in [141] and display the receptive fields (marked in red boxes) for these units. As our network is trained on human action recognition videos, many units show preference for human body parts and pose.

3.4 Action Recognition

The previous experiments show that the unsupervised task learns a meaningful representation. In this section we compare our unsupervised method against existing baseline methods and present more quantitative results. We organize our experiments as follows: 1) Comparing our unsupervised method to learning from random initialization. 2) Exploring other unsupervised baselines and comparing our method with them. 3) Combining our unsupervised representation learning method with a supervised image representation. Additional experiments are in the supplementary material. We now describe the common experimental setup.

Datasets and Evaluation: We use the UCF101 [384] dataset which was also used for our ablation analysis in Section 3.3 and measure accuracy on the 101 action classification task. Additionally, we use the HMDB51 [224] dataset for action recognition. This dataset contains 3 splits for train/test, each with about 3.4k videos for train and 1.4k videos for testing. Each video belongs to one of 51 action categories, and performance is evaluated by measuring classification accuracy. We follow the same train/test protocols for both UCF101 and HMDB51 as described in Section 3.3. Note that the UCF101 dataset is about $2.5\times$ larger than the HMDB51 dataset.

Implementation details for pre-training: We use tuples sampled using $\tau_{\max} = 60$ and $\tau_{\min} = 15$ as described in Section 3.3. The class ratio of positive examples per mini-batch is 25%. The other parameters for training/finetuning are kept unchanged from Section 3.3.

Action recognition details: As in Section 3.3, we use the CaffeNet architecture and the parameters from [376] for both training from scratch and finetuning. We described the finetuning parameters in Section 3.3. For training from random initialization (or 'scratch'), we train for 80k iterations with an initial learning rate of 10^{-2}, decaying by a factor of 10 at steps 50k and 70k. The other training parameters (momentum, batch size etc.) are kept the same as in finetuning. We use the improved data augmentation scheme (different aspect-ratio, fixed crops) from [432] for all our methods and baselines. Note that we train or finetune all the layers of the network for all methods, including ours.

3.4.1 Unsupervised pre-training or random initialization?

In these experiments we study the advantage of unsupervised pre-training for action recognition in comparison to learning without any pre-training. We use our tuple prediction task to train a network starting from random initialization on the train split of UCF101. The unsupervised pre-trained network is finetuned on both the UCF101 and HMDB51 datasets for action recognition and compared against learning from scratch (without pre-training). We report the performance in Table 3.2. Our unsupervised pre-training shows a dramatic **improvement of** $+\mathbf{12.4}\%$ over training from scratch in UCF101 and a significant gain of $+4.7\%$ in HMDB51. This impressive gain demonstrates the informativeness of the unsupervised tuple verification task. On HMDB51, we additionally finetune a network which was trained from scratch on UCF101 and report its performance in Table 3.2 indicated by 'UCF supervised'. We see that this network performs worse than our unsupervised pre-trained network. The UCF101 and HMDB51 have only 23 action classes in common [376] and we hypothesize that the poor performance is due to the scratch UCF101 network being unable to generalize to actions from HMDB51. For reference, a model pre-trained

Table 3.2: Mean classification accuracies over the 3 splits of UCF101 and HMDB51 datasets. We compare different initializations and finetune them for action recognition.

Dataset	Initialization	Mean Accuracy
UCF101	Random	38.6
	(Ours) Tuple verification	**50.2**
HMDB51	Random	13.3
	UCF Supervised	15.2
	(Ours) Tuple verification	**18.1**

on the supervised ImageNet dataset [351] and finetuned on UCF101 gives 67.1% accuracy, and ImageNet finetuned on HMDB51 gives an accuracy of 28.5%.

3.4.2 Unsupervised Baselines

In this section, we enumerate a variety of alternative verification tasks that use only video frames and their temporal ordering. For each task, we use a similar frame sampling procedure to the one described in Section 3.3.1. We compare their performance after finetuning them on the task of action recognition. A more informative task should serve as a better task for pre-training.

Two Close: In this task two frames (f_b, f_d) (with high motion) are considered to be temporally close if $|b - d| < \tau$ for a fixed temporal window $\tau = 30$.

Two Order: Two frames (f_b, f_d) are considered to be correct if $b < d$. Otherwise they are considered incorrect. $|b - d| < 30$.

Three Order: This is the original temporal ordering task we proposed in Section 3.2.1. We consider the 3-tuple (f_b, f_c, f_d) to be correct only if the frames obey either ordering $b < c < d$ or $b > c > d$.

We also compare against standard baselines for unsupervised learning from video.

DrLim [157]: As Equation 3.1 shows, this method enforces temporal smoothness over the learned features by minimizing the l_2 distance d between representations (`fc7`) of nearby frames f_b, f_d (positive class or $c = 1$), while requiring frames that are not close (negative class or $c = 0$) to be separated by a margin δ. We use the same samples as in the 'Two Close' baseline, and set $\delta = 1.0$ [296].

$$L(f_b, f_d) = \mathbb{1}(c = 1)d(f_b, f_d) + \mathbb{1}(c = 0)\max(\delta - d(f_b, f_d), 0) \tag{3.1}$$

TempCoh [296]: Similar to the DrLim method, temporal coherence learns representations from video by using the l_1 distance for pairs of frames rather than the l_2 distance of DrLim.

Obj. Patch [434]: We use their publicly available model which was unsupervised pre-trained on videos of objects. As their patch-mining code is not available, we do not do unsupervised pre-training on UCF101 for their model.

Table 3.3: We compare the unsupervised methods defined in Section 3.4.2 by finetuning on the UCF101 and HMDB51 Action recognition (split 1 for both). Method with * was not pre-trained on action data.

Unsup Method →	Two Close	Two Order	DrLim [157]	TempCoh [296]	Three Order (Ours)	Obj. Patch* [434]
Acc. UCF101	42.3	44.1	45.7	45.4	**50.9**	40.7
Acc. HMDB51	15.0	16.4	16.3	15.9	**19.8**	15.6

All these methods (except [434]) are pre-trained on training split 1 of UCF101 without action labels, and then finetuned on test split 1 of UCF101 actions and HMDB51 actions. We compare them in Table 3.3. Scratch performance for test split 1 of UCF101 and HMDB51 is 39.1% and 14.8% respectively. The tuple verification task outperforms other sequential ordering tasks, and the standard baselines by a significant margin. We attribute the low number of [434] to the fact that they focus on object detection on a very different set of videos, and thus do not perform well on action recognition.

Table 3.4: Results of using our unsupervised pre-training to adapt existing image representations trained on ImageNet. We use unsupervised data from training split 1 of UCF101, and show the mean accuracy (3 splits) by finetuning on HMDB51.

Initialization	Mean Accuracy
Random	13.3
(Ours) Tuple verification	18.1
UCF sup.	15.2
ImageNet	28.5
(Ours) ImageNet + Tuple verification	**29.9**
ImageNet + UCF sup.	30.6

3.4.3 Combining unsupervised and supervised pre-training

We have thus far seen that unsupervised pre-training gives a significant performance boost over training from random initialization. We now see if our pre-training can help improve existing image representations. Specifically, we initialize our model using the weights from the ImageNet pre-trained model and use it for the tuple-prediction task on UCF101 by finetuning for 10k iterations. We hypothesize this may add complementary information to the ImageNet representation. To test this, we finetune this model on the HMDB51 [224] action recognition task. We compare this performance to finetuning on HMDB51 without the tuple-prediction task. Table 3.4 shows these results.

Our results show that combining our pre-training with ImageNet helps improve the accuracy of the model (rows 3, 4). Finally, we compare against using multiple sources of supervised data: initialized using the ImageNet weights, finetuned on UCF101 action recognition and then finetuned on HMDB51 (row 5). The accuracy using all sources of supervised data is only slightly better than

the performance of our model (rows 4, 5). This demonstrates the effectiveness of our simple yet powerful unsupervised pre-training.

3.5 Human Pose Estimation

The qualitative results from Sec 3.3.3 suggest that our network captures information about human pose. To evaluate this quantitatively, we conduct experiments on the task of pose estimation using keypoint prediction.

Datasets and Metrics: We use the FLIC (full) [358] and the MPII [20] datasets. For FLIC, we consider 7 keypoints on the torso: head, $2\times$ (shoulders, elbows, wrists). We compute the keypoint for the head as an average of the keypoints for the eyes and nose. We evaluate the Probability of Correct Keypoints (PCK) measure [449] for the keypoints. For MPII, we use all the keypoints on the full body and report the PCKh@0.5 metric as is standard for this dataset.

Model training: We use the CaffeNet architecture to regress to the keypoints. We follow the training procedure in [412][1]. For FLIC, we use a train/test split of 17k and 3k images respectively and finetune models for 100k iterations. For MPII, we use a train/test split of 18k and 2k images. We use a batch size of 32, learning rate of 5×10^{-4} with AdaGrad [88] and minimize the Euclidean loss (l_2 distance between ground truth and predicted keypoints). For training from scratch (Random Init.), we use a learning rate of 5×10^{-4} for 1.3M iterations.

Methods: Following the setup in Sec 3.4.1, we compare against various initializations of the network. We consider two supervised initalizations - from pre-training on ImageNet and UCF101. We consider three unsupervised initializations - our tuple based method, DrLim [157] on UCF101, and the method of [434]. We also combine our unsupervised initialization with ImageNet pre-training.

Our results for pose estimation are summarized in Table 3.5. Our unsupervised pre-training method outperforms the fully supervised UCF network (Sec 3.4.1) by $+7.6\%$ on FLIC and $+2.1\%$ on MPII. Our method is also competitive with ImageNet pre-training on both these datasets. Our unsupervised pre-training is complementary to ImageNet pre-training, and can improve results after being combined with it. This supports the qualitative results from Sec 3.3.3 that show our method can learn human pose information from unsupervised videos.

3.6 Discussion

In this chapter, we studied unsupervised learning from the raw spatiotemporal signal in videos. Our proposed method outperforms other existing unsupervised methods and is competitive with supervised methods. A next step to our work is to explore different types of videos and use other 'free' signals such as optical flow. Another direction is to use a combination of CNNs and RNNs, and to extend our tuple verification task to much longer sequences. We believe combining this with semi-supervised methods like those presented in Chapter 2 is a promising future direction.

[1] Public re-implementation from https://github.com/mitmul/deeppose

Table 3.5: Pose estimation results on the FLIC and MPII datasets.

Init.	PCK for FLIC						PCKh@0.5 for MPII		
	wri	elb	sho	head	Mean	AUC	Upper	Full	AUC
Random Init.	53.0	75.2	86.7	91.7	74.5	36.1	76.1	72.9	34.0
Tuple Verif.	**69.6**	**85.5**	**92.8**	**97.4**	**84.7**	**49.6**	**87.7**	**85.8**	**47.6**
Obj. Patch[434]	58.2	77.8	88.4	94.8	77.1	42.1	84.3	82.8	43.8
DrLim[157]	37.8	68.4	80.4	83.4	65.2	27.9	84.3	81.5	41.5
UCF Sup.	61.0	78.8	89.1	93.8	78.8	42.0	86.9	84.6	45.5
ImageNet	69.6	86.7	93.6	97.9	85.8	51.3	85.1	83.5	47.2
ImageNet + Tuple	**69.7**	**87.1**	**93.8**	**98.1**	**86.2**	**52.5**	**87.6**	**86.0**	**49.5**

Part II

Structure in Labels and Tasks

Chapter 4

Learning from Noisy Human-centric Web Labels

A goal of visual recognition is, therefore, not only to detect and classify objects but also to associate with each a level of priority which we call 'importance'.

MERRIELLE SPAIN AND PIETRO PERONA, in *Some objects are more equal than others: Measuring and predicting importance*, 2008

In the previous chapters, we have seen how utilizing structure in the data space can help us learn better representations and object detectors. All these methods are still restricted to the categories of visual concepts in the pristinely labeled datasets (*e.g.*, [94, 251, 351]). These datasets have 100s to 1000s of 'typical' visual concepts or categories. In contrast, a conservative estimate [408] of the number of visual tags present on Flickr, tells us that there are more than 8 million tags or visual concepts. Thus, our current vision datasets are far from capturing the diversity and variety of visual concepts. Scaling these supervised datasets by more than 800× the number of concepts is an infeasible task. An alternative approach is to relax this requirement of pristinely labeled data. The learning algorithm can be enabled to use readily-available sources of annotated data, such as user-generated image tags or captions from social media services like Flickr or Instagram. Such datasets easily scale to hundreds of millions of photos with millions of distinct tags [408] without expensive curation. In this chapter, we present an approach that can leverage such freely available tag-level supervision.

Images annotated with human-written tags [408] or captions [58] focus on the most important or salient information in an image, as judged implicitly by the annotator. These annotations lack information on minor objects or information that may be deemed unimportant, a phenomenon known as *reporting bias* [146]. For example, Figure 4.1 illustrates two concepts (`bicycle`, `yellow`) that are each present in two images, but only mentioned in one. The bicycle may be considered irrelevant to the overall image in (b); and the bananas in (d) are not described as yellow because humans often omit an object's typical properties when referring to it [268, 439]. Following [38], we refer to this type of labeling as *human-centric annotation*. Human-centric annotations have label noise because the underlying visual concepts are not labeled pristinely. This 'noise' view is common in machine learning and the human-centric (noisy) aspect of these annotations is generally considered a nuisance. On the flip side, we can view this human-centric annotation as a core part of visual recognition itself. Spain and Perona (quoted above) assert that human-centric predictions associate

a 'priority' with visual concept predictions which is a core aspect of recognition. While humans *can* exhaustively label everything, we have the ability to prioritize and describe the most important aspects of an image. Thus, machines should also have the ability to prioritize their predictions, and the label noise can help them learn how to do this.

Training directly on human-centric annotations does not yield a credible visual concept classifier. Instead, it leads to a classifier that attempts to *mimic* the reporting bias of the annotators. To separate reporting bias from visual ground truth, we propose to train a model that explicitly factors human-centric label prediction into a *visual presence* classifier (*i.e.*, "Is this concept visually present in this image?") and a *relevance* classifier (*i.e.*, "Is this concept worth mentioning, given its visual presence?"). We train all these classifiers jointly and end-to-end as multiple "heads" branching from the same shared convolutional neural network (ConvNet) trunk [239, 375].

We demonstrate improved performance on several tasks and datasets. Our experiments on the MS COCO Captions dataset [58] show an improvement in mean average precision (mAP) for the learned visual classifiers when evaluated on both fully labeled data (using annotations from the MS COCO detection benchmark [251]) and on the human generated caption data. We also show that using such visual predictions improves image caption generation quality. Our results on the Yahoo Flickr 100M dataset [408] demonstrate the ability of our model to learn from "in the wild" data (noisy Flickr tags) and **double the performance** of the baseline classification model. Apart from just numerical improvements, our results are interpretable and consistent with research in psychology showing that humans tend not to mention typical attributes [268, 439] unless required for unique identification [307, 361, 405].

4.1 Background

Label noise is ubiquitous in real world data. It can impact the training process of models and decrease their predictive accuracy [29, 203, 301]. Since there are vast amounts of cheaply available noisy data, learning good predictors despite the label noise is of great practical value.

The taxonomy of label noise presented in [126] differentiates between two broad categories of noise: *noise at random* and *statistically dependent noise*. The former does not depend on the data, while the latter does. In practice, one may encounter a combination of both types of noise.

Human-centric annotations [38] exhibit noise that is highly structured and shows statistical dependencies on the data [38, 126, 386, 458]. It is structured in the sense that certain labels are preferentially omitted as opposed to others. Vision researchers have studied human-centric annotations in various settings, such as missing objects in image descriptions [38], scenes [40], and attributes [415] and show that these annotations are noisy [38]. Much of the work on learning from noisy labels focuses on robust algorithms [189, 271], voting methods [33], or statistical queries [214]. Some of these methods [189, 214] require access to clean oracle labels, which may not be readily available.

Explicitly modeling label noise has received increasing attention in recent years [295, 300, 392, 445]. Many of these methods operate under the "noise at random" assumption and treat noise as conditionally independent of the image. [236] models symmetric label noise (independent of the

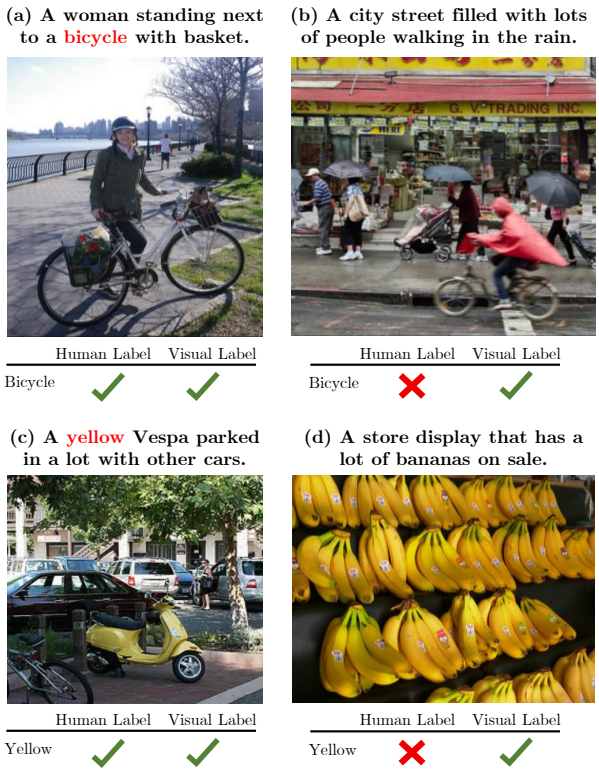


Figure 4.1: Human descriptions capture only some of the visual concepts present in an image. For instance, the bicycle in (a) is described, while the bicycle in (b) is not mentioned. The Vespa in (c) is described as "yellow", while the bananas in (d) are not, as being yellow is typical for bananas.

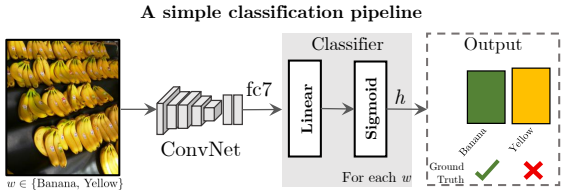


Figure 4.2: A simple classification model for learning from human-centric annotations. The noisy labels (`banana` is not annotated as `yellow`) impede the learning process.

true label), which is a strong assumption for real world data. [300, 392] both model asymmetric label noise that is conditionally independent of the image. Such an assumption ignores the input image (and the objects therein) which directly affects the noisy annotations produced by humans [38].

Recently, Xiao *et al*. [445] introduced an image conditional noise model that attempts to predict what type of noise corrupts each training sample (no noise, noise at random, and structured label swapping noise). Unlike [445], our training algorithm does not require a small amount of cleanly labeled training data to bootstrap parameter estimation. Our model is also specifically designed to handle the noise found in human-centric annotations.

Bootstrapping [336], semi-supervised learning (SSL) [371, 479] and Positive Unlabeled (PU) learning [93, 247, 256, 284] are other ways of learning from noisy labeled data. However, they require access to clean oracle labels. SSL approaches are often computationally impractical [469, 480] or make strong independence assumptions [113] that do not hold in human-centric annotations. Our

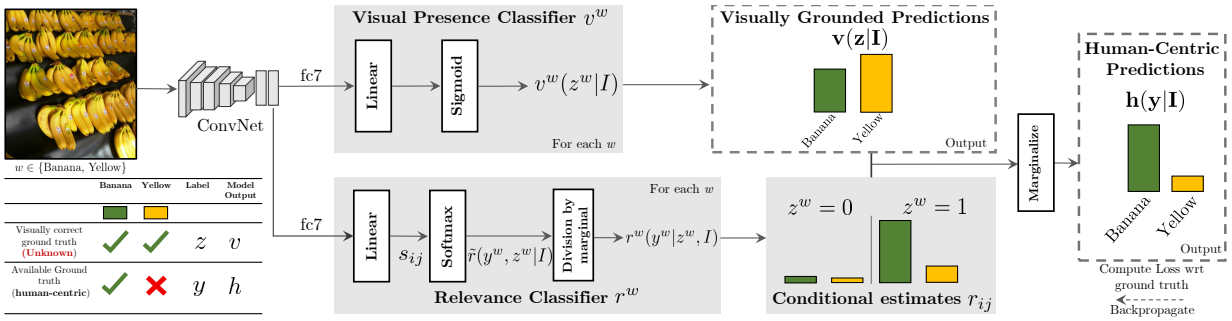


Figure 4.3: Our model uses noisy human-centric annotations y for learning visually grounded classifiers without access to the visually correct ground truth z. It uses two classifiers: a visual presence classifier v and a relevance classifier r. The visual presence classifier v predicts whether the visual concept w is visually present in an image. The relevance classifier r models the noise and predicts whether the concept should be mentioned or not. We combine these predictions to get the human-centric prediction h.

approach, which trains directly on noisy labels, can serve as a starting point for these approaches.

The work described here is also consistent with research in psycholinguistics on object reference and description. Such work demonstrates that humans store typical or "prototypical" representations of objects and their properties [347]; and this background knowledge appears to have an effect on object description [268, 439]. People tend not to mention attributes that are obvious or typical for an object, preferring to name attributes required for conversational relevance [223], unique identification against alternatives [361, 418] and distinguishability [149, 218]. A similar separation between *what is observed* and *what is mentioned* falls out naturally from our proposed model.

4.2 Approach

Our goal is to train visually grounded image classifiers for a set of visual concepts $w \in \mathcal{W}$ (*e.g.*, `banana`, `yellow`, `zebra`) using images and their human-centric annotations. The conventional approach to this problem, shown in Figure 4.2, is to naively apply a supervised learning algorithm: train a classifier h^w for each concept w, to predict its human-centric label $y^w \in \{0, 1\}$ (not mentioned *vs.* mentioned) as a conditional probability distribution $h^w(y^w|\mathcal{I})$, in which $\mathcal{I}$ is an image.

The resulting classifier would attempt to mimic human reporting bias by predicting how a human would label the image $\mathcal{I}$ regardless of whether w is visually present or absent. Thus, the predictions from each classifier h^w will not be visually grounded and do not meet our goal. How can we build classifiers that predict whether a visual concept w is present in an image and "see through" the noise in human-centric annotations?

4.2.1 Factor decoupling

We propose to structure each concept output $h^w(y^w|\mathcal{I})$ in terms of two classifiers v^w and r^w. The first classifier v^w models the conditional probability of the *visual presence* of the concept w in the

image. The second classifier r^w models the conditional probability of the *relevance* of the concept w, conditioned on the image and whether or not w is estimated to be visually present. The human-centric predictor is formed by marginalizing over the concept's visual presence, as described next.

Let $z^w \in \{0, 1\}$ be a latent (or hidden) variable indicating whether the concept w is visually present in an image. Note that the training data only supplies human-centric labels y^w; the true values of z^w are unknown during training. For instance, in Figure 4.1(a), $y^w = 1$ and $z^w = 1$ when the bicycle is present and mentioned, while $y^w = 0$ and $z^w = 1$ in Figure 4.1(b) when the bicycle is present but not mentioned. We refer to z^w as the *visual presence label*.

The conditional probability of the human-centric label given an image $\mathcal{I}$, $h^w(y^w|\mathcal{I})$, can now be computed by marginalizing over the latent visual presence label z^w:

$$h^w(y^w|\mathcal{I}) = \sum_{j \in \{0,1\}} r^w(y^w|z^w = j, \mathcal{I}) v^w(z^w = j|\mathcal{I}). \tag{4.1}$$

An illustration of our model is shown in Figure 4.3. One important property of this formulation is that it allows the model to assign high confidence to unlabeled visual concepts: for an unmentioned concept ($y^w = 0$ and $z^w = 1$), the relevance classifier r^w allows the visual presence classifier v^w to assign a high probability to the true visual label ($z^w = 1$) while still making a prediction that matches the human-centric label ($y^w = 0$). This property enables the model to "see through" human reporting bias.

To simplify notation, we drop the concept index w from y, z, h, v and r when possible. We denote the probability values of r by:

$$r_{ij} = r(y = i|z = j, \mathcal{I}), \;\; \forall (i, j) \in \{0, 1\}^2. \tag{4.2}$$

Another important property of the factorization in Equation (4.1) is that it provides a way to get two different predictions for the concept w in the same image. The model can predict the visual presence of a visual concept; or predict how a human would annotate the image. Depending on the task at hand, one of these predictions may be more appropriate than the other, a point we demonstrate later via experimental results.

4.2.2 Model learning and parameterization

We estimate the model parameters by minimizing the regularized log loss of h^w, summed over all concepts w, on the training data annotated with human-centric labels. Since our model includes latent variables z for each concept and image, one approach could be to use Expectation Maximization (EM) [77]. We choose a direct optimization approach [51] over EM for simplicity, and thus both the model parameters and latent variable posteriors are updated and inferred online. In each SGD minibatch, the model predicts the conditional distributions r and v, marginalizes over the values of z, and uses the log loss of h to drive better estimates of r and v.

The conditional distributions r and v are modeled with a ConvNet [129, 239, 350]. As illustrated

Table 4.1: mAP and PHR values on MS COCO captions ground truth (20k test images). We add our latent model to each baseline to make predictions that are visually grounded (v) or conform to human (h) labels. POS tags are as follows: Nouns (NN), Verbs (VB), Adjectives (JJ), Determiners (DT), Pronouns (PRP), Prepositions (IN).

			Mean Average Precision								Precision at Human Recall							
		Prob	NN 616	VB 176	JJ 119	DT 10	PRP 11	IN 38	Others 30	All 1000	NN ← Count	VB	JJ	DT	PRP	IN	Others	All
VGG16	MILVC [102]	-	41.6	20.7	23.9	33.4	20.4	22.5	16.3	34.0	52.7	32.8	40.5	40.3	32.2	33.0	24.6	45.8
	MILVC + Multiple-`fc8`	-	41.1	20.9	23.7	33.6	21.1	22.8	16.8	33.8	51.2	32.6	40.8	41.1	31.7	33.5	27.3	45.0
	MILVC + Latent (Ours)	v	42.9	21.7	24.9	33.1	19.6	23.0	16.2	35.1	53.6	35.4	43.3	41.3	28.0	36.0	24.4	47.2
	MILVC + Latent (Ours)	h	44.3	22.3	25.8	34.4	21.8	23.6	17.3	**36.3**	55.5	36.3	44.7	42.9	32.1	37.3	26.4	**48.9**
AlexNet	MILVC [102]	-	33.2	16.2	20.1	30.9	16.4	19.9	14.6	27.4	40.0	26.4	36.0	38.2	24.2	27.5	21.9	35.9
	MILVC + Latent (Ours)	v	35.6	17.7	21.9	32.4	16.9	20.7	15.2	29.4	43.9	28.3	37.5	41.2	29.2	29.9	23.3	39.0
	MILVC + Latent (Ours)	h	36.5	18.0	22.4	32.9	17.8	21.4	15.6	**30.1**	45.1	28.7	38.0	41.2	32.2	31.0	24.0	**40.0**
VGG16	Classif.	-	34.9	18.1	20.5	32.8	19.2	21.8	16.3	29.0	42.5	30.4	33.9	40.5	30.4	30.7	23.8	38.2
	Classif. + Multiple-`fc8`	-	34.2	17.7	19.9	32.6	19.0	21.5	15.9	28.4	41.3	27.9	32.3	39.6	29.6	31.2	22.6	36.8
	Classif. + Latent (Ours)	v	37.7	19.6	22.0	32.6	20.2	22.0	16.3	31.2	46.3	32.9	36.8	38.9	32.3	33.1	27.0	41.5
	Classif. + Latent (Ours)	h	38.7	20.1	22.6	33.8	21.2	23.0	17.5	**32.0**	47.8	33.7	37.9	42.5	34.2	34.4	29.0	**42.9**

in Figure 4.3, the ConvNet trunk is shared between the two distributions (per concept) and then branches near the output into two sets of untied parameters. We jointly train one network for all visual concepts $w \in \mathcal{W}$ by treating learning as a multi-label classification problem. Further network architecture details are given in Section 4.3 with experiments.

The conditional probability distribution r models transition probabilities and thus its underlying joint distribution $\tilde{r}_{ij} = \tilde{r}(y = i, z = j|\mathcal{I})$ must be a valid probability distribution. To enforce this constraint, we directly estimate the joint distribution $\tilde{r}$ with a softmax operation on a vector of unnormalized scores.

For each concept w, we first compute four scores s_{ij} using four linear models parameterized by weights m_{ij} and biases b_{ij}, and then normalize them using the softmax function to get a valid joint distribution $\tilde{r}_{ij}$:

$$s_{ij} = m_{ij}^T \phi(\mathcal{I}) + b_{ij}, \tag{4.3}$$

$$\tilde{r}_{ij} = \exp(s_{ij}) / \sum_{i'j'} \exp(s_{i'j'}). \tag{4.4}$$

For $\phi(\mathcal{I})$, we use global image features computed by the shared ConvNet trunk (*e.g.*, `fc7` layer activations from VGG16 [375]). These features capture the global image context, which is helpful in estimating r. Each r_{ij} can then be computed from $\tilde{r}$ by dividing by the marginal $\tilde{r}(z = j|\mathcal{I})$:

$$r_{ij} = \tilde{r}_{ij} / \sum_{i'} \tilde{r}_{i'j}. \tag{4.5}$$

Figure 4.3 illustrates our process of computing r_{ij} from the image. Since our operations for estimating $\tilde{r}$ (and thus r) are differentiable, we can backpropagate their errors to the ConvNet, allowing the full model to be trained end-to-end.

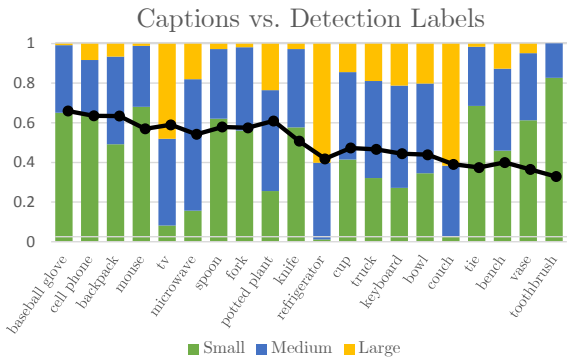


Figure 4.4: We use the MS COCO dataset to display the objects with the highest reporting bias. We use the detection labels for objects and see whether they are mentioned in the caption. The black line shows the probability of an object not being mentioned in the caption. The distribution of sizes for objects that are missed are shown by the color bars. Note that for many categories most unreported objects are small or medium in size (green and blue).

4.3 Experiments

We evaluate our proposed model on two datasets: Microsoft COCO [251] and a random subset of the Yahoo Flickr Creative Commons 100M (YFCC100M) dataset [408]. The YFCC100M dataset includes user-generated image tags, which we take as our source of human-centric annotations. For MS COCO, we take the supplied image captions [58] as human-centric annotations. We use the MS COCO object detection labels for dataset analysis and algorithm evaluation. These labels allow us to verify the accuracy of trained visual presence classifiers, v^w; these labels are never used for training our model.

4.3.1 Experiments on MS COCO 1k visual concepts

Our first set of experiments use the MS COCO 1000 visual concepts from [102]. The visual concepts are the 1000 most common words in the MS COCO captions dataset [58] and include nouns, verbs, adjectives, and other parts of speech (see Table 4.1 for a breakdown).

For training, we generate image labels as 1000-dimensional binary vectors indicating which of the 1000 target visual concepts are present in the any of the 5 reference captions for each training image. The training set includes approximately 80k images. For evaluation, we follow [102] and split the val set into equally sized val and test sets of ∼20k images each; we use the same splits as in [102]. We report results on this 20k image test set.

Human reporting bias in image descriptions

We first analyze the annotation mismatch between the caption labels and the detection labels. We obtain labels for the 73 objects common in both the caption and object detection labels (see Section 4.3.1 for details). We use the notation from Section 4.2.1, and measure the human reporting bias as $r^{\star}_{01}$: the probability of an object not being mentioned in the caption ground truth ($y = 0$) and being present in the detection ground truth ($z = 1$), over all the training images. To account for

object size as a factor in an object not being mentioned, we split these measurements based on the size of the bounding box (sizes as defined in [251]). Figure 4.4 shows this mismatch for the top 20 objects with the highest $r^{\star}_{01}$ for values. A high $r^{\star}_{01}$ value indicates that there is a large mismatch between the caption ground truth and the detection ground truth – objects that are visually present but not mentioned in the captions.

As observed, there is a high degree of human labeling noise in the image descriptions, with the object with highest reporting bias mentioned roughly half as much as it appears.

Evaluating human-centric label prediction

We can evaluate our model in two ways: as a purely visual classifier (v) or as a predictor of human-centric (h) labels. We start with the latter and evaluate our model's predictions against the human-centric MS COCO captions.

As a strong baseline, we use the recently proposed MILVC [102] approach. This method applies a ConvNet (VGG16 [375] or AlexNet [221])[1] in a fully-convolutional way to a large input image to generate a 12×12 grid of human-centric label predictions. It then uses a noisy-OR [427] to compute a single prediction from the 144 intermediate values. During training, the noisy-OR induces a form of multiple instance learning (MIL). Note that the baseline model estimates h labels directly without our decomposition into relevance and visual presence factors. As a second baseline (Classif.), we use a vanilla classification model akin to Figure 4.2, in which an ImageNet [351] pre-trained ConvNet is fine-tuned to directly predict human-centric labels.

We also include an additional baseline variant that adds extra parameters to control for the fact that our proposed model requires adding extra parameters. Specifically, we train a "Multiple-`fc8`" model for each method (MILVC and Classif.) that has the same number of parameters as our model. To train Multiple-`fc8`, we add four extra randomly initialized `fc8` (linear classification) layers, each with their own loss. At test time, we average the predictions of all the `fc8` layers to get the final prediction.

We implement two variants of our model to parallel the MILVC and Classif. baselines. In the first variant (MILVC + Latent), v uses a noisy-OR over a 12×12 grid of visual presence predictions, and for r we average pool the 144 `fc7` activation vectors to obtain a single 4096 dimensional $\phi(\mathcal{I})$ for Equation 4.3. The second variant (Classif. + Latent) generates a 1×1 output for v and `fc7`, and therefore omits the noisy-OR and average pooling. In both cases, h predictions are obtained following Equation 4.1, using both r and v. Like the baselines, our model is trained to minimize the log loss (cross-entropy loss) over h.

To train our model, we set the joint noise distribution $\tilde{r}$ to identity (*i.e.*, $\tilde{r}_{11} = \tilde{r}_{00} = 0.5$) for the first two epochs and then update it for the last two epochs. Table 4.1 shows mean average precision (mAP) [94] and precision at human recall (PHR) [58] on the 20k test set. PHR is a metric proposed in [58], and measures precision based on human agreement. Briefly, this metric uses multiple references per image to compute a "human recall" value, an estimate of the probability that a human will use a particular word for an image. Precision is then computed at this "human

[1] Unless otherwise specified, we use VGG16 for all our experiments.

Table 4.2: Visual classification on 73 classes evaluated using fully labeled data from COCO.

	MILVC [102]	v	h	Using ground truth
mAP	63.7	**66.8**	66.5	76.3

recall" value to get PHR. [58] shows that for the task of predicting visual concepts, PHR is a more stable metric than an AP metric, since it accounts for human agreement.

We report results for the 1000 visual concepts in aggregate, as well as grouped by their part-of-speech (POS) tags, on the MS COCO test split of 20k images (Table 4.1). Our latent variable model improves classification performance over all the baseline networks and architectures by **3 to 4 points** for both metrics (mAP and PHR). Interestingly, the Multiple-`fc8` model, which has the same number of parameters as our latent model, does not show an improvement, even after extensive tuning of learning hyperparameters. This finding makes the contribution of the proposed model evident; the improvement is not simply due to adding extra parameters. It is worth noting that in Table 4.1, h is a better predictor of MS COCO caption labels than v, as h directly models the human-centric labels used for evaluation.

Evaluating visual presence prediction

The decoupling of visual presence (v) predictions and human-centric (h) label predictions allows our model to learn better visual predictors. To demonstrate this, we use the fully-labeled ground truth from the COCO detection annotations to evaluate the visually grounded v label predictions.

Since the 1000 visual concepts include many fine-grained visual categories (*e.g.*, `man`, `woman`, `child`) and synonyms (*e.g.*, `bike`, `bicycle`), we manually specify a mapping from the visual concepts to the 80 MS COCO detection categories, *e.g.*, $\{$`bike`, `bicycle`$\} \rightarrow$ `bicycle`. We find that 73 of the 80 detection categories are present in the 1000 visual concepts. We use this mapping only at evaluation time to compute the probability of a detection category as the maximum of the probabilities of its fine-grained/synonymous categories.

Table 4.2 shows the mean average precision (mAP) of our method, as well as the baseline on these 73 categories. As expected, using the human-centric model (h) for this task of visual prediction hurts performance (slightly). The performance drop is less dramatic when evaluating on these 73 classes because these classes have less label noise for large sized objects as compared to the 1000 visual concepts. We also train a noise-free reference model using the ground-truth visual labels from the detection dataset (*i.e.*, the true values of the latent z labels).

Importance of conditioning on input images

A central point of this chapter (and also in [38]) is that human-centric label noise is statistically dependent on image data. Here we demonstrate that our model is indeed improved by conditioning the noise (relevance) distribution on the input image, in contrast to previous work [300, 392] that estimates noise parameters *without* conditioning on the image.

Table 4.3: We show the importance of conditioning the relevance r on the input image. We measure the classification mAP on MS COCO 1k visual concepts using both the visually grounded (v) and the human-centric (h) predictions. Conditioning on the input image shows improvement over the baseline showing that human reporting bias statistically depends on the input image.

		w/o image		w/ image	
	MILVC [102]	v	h	v	h
mAP	34.0	34.2	34.3	35.1	**36.3**

Figure 4.5: Our model modifies visually correct detections to conform to human labeling. We show this modification for a few images of target visual concepts in the MS COCO Captions dataset. We first show the variation between h (y axis) and v values (x axis) for each concept in a 2D histogram. After thresholding at $v \geq 0.8$, we pick a representative image from each quantile of h (h increases from left to right). As you move from left to right, the model transitions from predicting that a human would not "speak" the word to predicting that a human would speak it. The human-centric h predictions of concepts depend on the image context, *e.g.*, `fence` at a soccer game vs. `fence` between a bear and a human (first row). Our model picks up such signals to not only learn a visually correct `fence` predictor, but also when a `fence` should be mentioned.

To better understand the importance of this conditioning, we consider a model akin to [392]. We estimate the latent distribution r without conditioning on the input image, and compare it to our model that computes r conditioned on the image. Table 4.3 shows that mAP is significantly improved by conditioning on the image. When not conditioned on the image, only minor gains are achieved.

4.3.2 Experiments on Flickr image tagging

Datasets like MS COCO are curated by searching for images with specific objects [251]. In contrast, social media websites like Flickr contain much larger collections of images that are annotated with user-generated content such as tags, keywords, and descriptions. The words found in such data

Figure 4.6: Our model learns clean visual predictors from noisy labels. Here we show corrected false positives: MILVC incorrectly reports a high probability ($h \geq 0.75$) for the concept, while our model correctly reports a low probability ($v \leq 0.3$); and corrected false negatives: MILVC incorrectly reports a low probability ($h \leq 0.3$) for the concept, while our model correctly reports a high probability ($v \geq 0.75$). For example, consider `zebra` vs. `zebras`, and `banana` vs. `bananas` in the last row, where our model correctly "counts" compared to the baseline. Images are from the MS COCO Captions dataset.

exhibit the human-centric annotation properties modeled by our approach.

We test our model on this "real world" data by using a random subset of ∼89k images from the YFCC100M dataset [408]. We ensure that these images have at least 5 and at most 30 human annotated tags that are present in the WordNet [282] lexicon. We split this dataset into 75k training images and 14k test images, and consider the top 1000 tags as the set of visual concepts. We train the baseline MILVC [102] model and our model for 4 epochs following the same hyperparameters used for MS COCO training.

Table 4.4 shows the numerical results of these models evaluated on the test set using the same

Table 4.4: mAP values on a subset of YFCC100M. We add our latent model over the MILVC baseline to make predictions that are visually grounded (v) or that conform to human-centric (h) labels. POS tags: Nouns (NN), Verbs (VB), Adjectives (JJ), Pronouns (PRP), Prepositions (IN).

			Mean Average Precision						
	Count →	Prob	NN 791	VB 10	JJ 148	PRP 13	IN 23	Others 15	All 1000
VGG16	MILVC [102]	-	5.7	9.2	5.2	3.8	8.8	6.1	5.7
	MILVC + Multiple-`fc8`	-	4.6	6.2	3.8	2.7	7.3	3.1	4.5
	MILVC + Latent (Ours)	v	9.8	15.1	8.9	8.3	12.4	12.4	9.8
	MILVC + Latent (Ours)	h	11.2	15.4	9.9	8.2	16.3	12.5	**11.2**

human annotated tags. As explained in Section 4.3.1, we compare against the MILVC baseline, and a model with the same number of parameters as ours (denoted by Multiple-`fc8`). Our model has double the performance of the baseline MILVC model and **increases mAP by 5.5 points**.

4.3.3 Interpretability of the noise model

The relevance classifier r models human labeling noise conditioned on the image. Depending on the image, it can enhance or suppress the visual prediction for each concept. We show such modifications for a few visual concepts in Figure 4.5. After thresholding at $v \geq 0.8$, we pick a representative image from each quantile of h (h increases from left to right). The variation in h values for these high confidence $v \geq 0.8$ images (shown in a 2D histogram in each row) indicates that h and v have been decoupled by our model. The images show that our model captures subtle nuances in the ground truth, *e.g.*, mention a hat worn by a cat, do not mention the color of a pumpkin, definitely mention pink sheep, *etc*. It automatically captures that context is important for certain objects like `fence` and `hat`, while certain attributes are worth mentioning to help distinguish objects like the `orange` pillow. Such connections have been shown in both vision research [38] and psychology [149, 361].

4.3.4 Correcting error modes by decoupling

Modeling latent noise in human-centric annotations allows us to learn clean visual classifiers. In Figure 4.6, we compare our model's visual presence v predictions with the baseline (MILVC) and show a few error modes that it corrects. Our model is able to correct error modes like misspellings (`desert` vs. `dessert` in the first row), localizes objects correctly and out of context (`fridge` in the second row, `net` in the first row, *etc*.) and is better at counting (`zebra`, `banana` last row).

4.3.5 Using word detections for caption generation

We now look at the task of automatic image caption generation and show how our model can help improve the task. We consider a basic Long Short-Term Memory (LSTM) [173] network to generate captions. We use 1000 cells for the LSTM, and learn a 256 dimensional word embedding for the input words. Following [87], our vocabulary consists of words with frequency ≥ 5 in the input captions. The image features (1000 visual concept probabilities) are fed once to the LSTM as its first hidden input. We train this LSTM over all the captions in the MS COCO caption training data for 20 epochs using [195, 302]. We use beam size of 1 for decoding. Table 4.5 shows the evaluation of the automatically generated captions using standard captioning metrics. Using the probabilities from

Table 4.5: LSTM captioning results on MS COCO

	Prob	BLEU-4	ROUGE	CIDEr
MILVC [102]	-	27.7	51.8	89.7
MILVC + Latent (Ours)	h	**29.2**	**52.4**	**92.8**

our model shows an improvement for all evaluation metrics. Thus, modeling the human-reporting bias can help downstream applications that require such human-centric predictions.

4.4 Discussion

In this chapter, we have introduced an algorithm that explicitly models *reporting bias* — the discrepancy between what exists and what people mention — for image labeling. By introducing a latent variable to capture "what is in an image" separate from "what is labeled in an image", we leverage *human-centric* annotations of images to their full potential, inferring visual concepts present in an image separately from the visual concepts worth mentioning. We demonstrate performance improvements over previous work on several tasks, including image classification and image captioning. Further, the proposed model is highly interpretable, capturing which concepts may be included or excluded based on the context and dependencies across visual concepts. Initial inspection of the model's predictions suggests consistency with psycholinguistic research on object description, with typical properties noticed but not mentioned.

The algorithm and techniques discussed here pave the way for new deep learning methods that decouple human performance from algorithmic understanding, modeling both jointly in a network that can be trained end-to-end. Future work may explore different methods to incorporate constraints on the latent variables, or to estimate their posteriors (such as with EM). Finally, to fully exploit the enormous amounts of data which exist "in the wild", algorithms that explicitly handle noisy data are essential. So utilizing structure in the label space helps us learn from noisy labels. In the next chapter, we will see how utilizing structure in the task space can help regularize representations and improve performance.

Chapter 5

Learning from Multiple Tasks (Cross-stitch Units)

Your ability to juggle many tasks will take you far.

RICH CARUANA's Fortune Cookie, 1997

Images offer multiple types of supervisory signals or tasks, *e.g.*, one can predict the category of the object *and* attributes of the object. These multiple types of tasks can offer complementary information which are aid learning. Figure 5.1 shows that knowing something has a saddle helps predict it can be a horse. Thus, understanding the similarity and differences between multiple such tasks can enable learning with limited supervision. As Rich's fortune cookie tells us, this ability to juggle many tasks is also a vital aspect of human learning. In this chapter, we study how using multiple forms of supervision can help regularize feature representations and improve performance on supervision starved categories and tasks.

In the recent recognition literature, using multiple tasks has given performance improvements, *e.g.*, the gains in segmentation [164] and detection [138, 139]. A key takeaway from these works is that multiple tasks, and thus multiple types of supervision, helps achieve better performance with the same input. But unfortunately, the network architectures used by them for multi-task learning differ notably. There are no insights or principles for how one should choose ConvNet architectures for multi-task learning. In this chapter, we propose a principled solution for multi-task learning.

5.0.1 Multi-task sharing: an empirical study

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

To investigate these questions, we first perform extensive experimental analysis to understand the performance trade-offs amongst different combinations of shared and task-specific representations. Consider a simple experiment where we train a ConvNet on two related tasks (*e.g.*, semantic segmentation and surface normal estimation). Depending on the amount of sharing one wants to enforce, there is a spectrum of possible network architectures. Figure 5.2(a) shows different

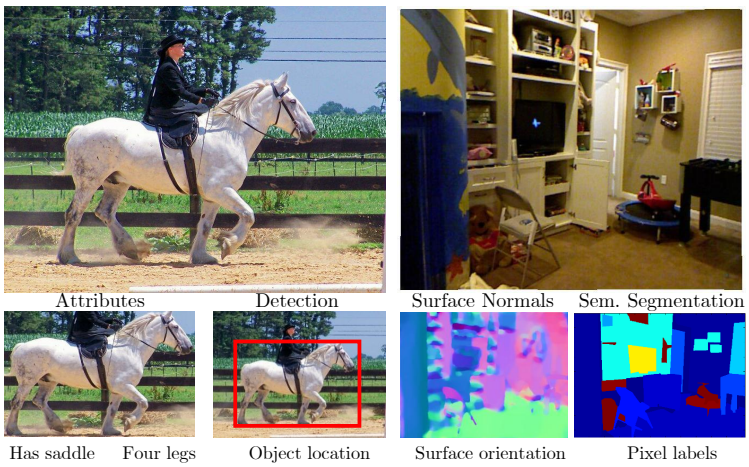


Figure 5.1: Given an input image, one can leverage multiple related properties to improve performance by using a multi-task learning framework. In this chapter, we propose *cross-stitch* units, a principled way to use such a multi-task framework for ConvNets.

ways of creating such network architectures based on AlexNet [221]. On one end of the spectrum is a fully shared representation where all layers, from the first convolution (`conv2`) to the last fully-connected (`fc7`), are shared and only the last layers (two `fc8`s) are task specific. An example of such sharing is [139] where separate `fc8` layers are used for classification and bounding box regression. On the other end of the sharing spectrum, we can train two networks separately for each task and there is no cross-talk between them. In practice, different amount of sharing tends to work best for different tasks.

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

- We first pair semantic segmentation (SemSeg) and surface normal prediction (SN). We believe the two tasks are closely related to each other since segmentation boundaries also correspond to surface normal boundaries. For this pair of tasks, we use NYU-v2 [374] dataset.

- For our second pair of tasks we use detection (Det) and Attribute prediction (Attr). Again we believe that two tasks are related: for example, a box labeled as "car" would also be a positive example of "has wheel" attribute. For this experiment, we use the attribute PASCAL dataset [94, 106].

We exhaustively enumerate all the possible *Split* architectures as shown in Figure 5.2(a) for these two pairs of tasks and show their respective performance in Figure 5.2(b). The best performance for both the SemSeg and SN tasks is using the "Split `conv4`" architecture (splitting at `conv4`), while for the Det task it is using the Split `conv2`, and for Attr with Split `fc6`. These results indicate two things – 1) Networks learned in a multi-task fashion have an edge over networks trained with one task; and 2) The best Split architecture for multi-task learning depends on the tasks at hand.

While the gain from multi-task learning is encouraging, getting the most out of it is still cumbersome in practice. This is largely due to the task dependent nature of picking architectures and

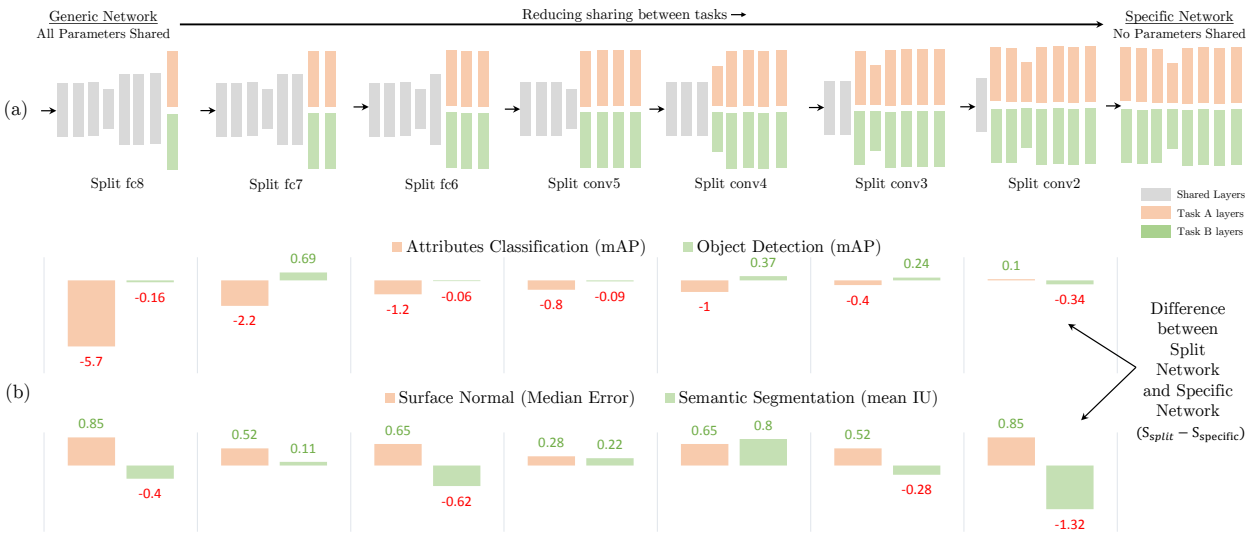


Figure 5.2: We train a variety of multi-task (two-task) architectures by splitting at different layers in a ConvNet [221] for two pairs of tasks. For each of these networks, we plot their performance on each task relative to the task-specific network. We notice that the best performing multi-task architecture depends on the individual tasks and does not transfer across different pairs of tasks.

the lack of a principled way of exploring them. Additionally, enumerating all possible architectures for each set of tasks is impractical. This chapter proposes *cross-stitch units*, using which a single network can capture all these Split-architectures (and more). It automatically learns an optimal combination of shared and task-specific representations. We demonstrate that such a *cross-stitched* network can achieve better performance than the networks found by brute-force enumeration.

5.1 Background

Generic Multi-task learning [46, 388] has a rich history in machine learning. The term *multi-task learning* (MTL) itself has been broadly used [13, 98, 191, 345, 447, 450] as an umbrella term to include representation learning and selection [25, 97, 209, 303], transfer learning [311, 332, 455] *etc*. and their widespread applications in other fields, such as genomics [304], natural language processing [68, 69, 260] and computer vision [15, 86, 205, 209, 328, 411, 443, 464]. In fact, many times multi-task learning is implicitly used without reference; a good example being fine-tuning or transfer learning [332], now a mainstay in computer vision, can be viewed as sequential multi-task learning [46]. Given the broad scope, in this section we focus only on multi-task learning in the context of ConvNets used in computer vision.

Multi-task learning is generally used with ConvNets in computer vision to model related tasks jointly, *e.g*. pose estimation and action recognition [143], surface normals and edge labels [433], face landmark detection and face detection [461, 466], auxiliary tasks in detection [139], related classes for image classification [406] *etc*. Usually these methods share some features (layers in ConvNets) amongst tasks and have some task-specific features. This sharing or split-architecture (as explained in Section 5.0.1) is decided after experimenting with splits at multiple layers and picking the best one. Of course, depending on the task at hand, a different Split architecture tends to work best,

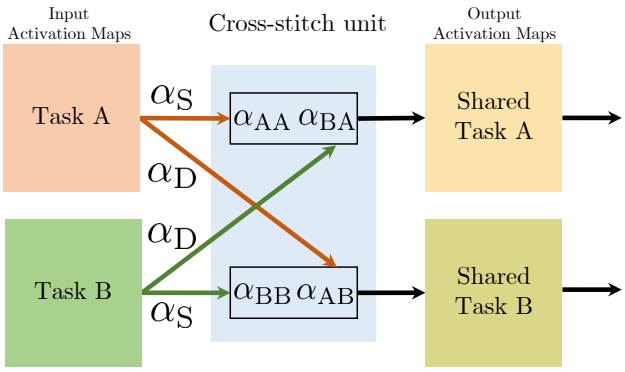


Figure 5.3: We model shared representations by learning a linear combination of input activation maps. At each layer of the network, we learn such a linear combination of the activation maps from both the tasks. The next layers' filters operate on this shared representation.

and thus given new tasks, new split architectures need to be explored. In this chapter, we propose *cross-stitch units* as a principled approach to explore and embody such Split architectures, without having to train all of them.

In order to demonstrate the robustness and effectiveness of cross-stitch units in multi-task learning, we choose varied tasks on multiple datasets. In particular, we select four well established and diverse tasks on different types of image datasets: 1) We pair semantic segmentation [168, 367, 368] and surface normal estimation [90, 122, 433], both of which require predictions over all pixels, on the NYU-v2 indoor dataset [374]. These two tasks capture both semantic and geometric information about the scene. 2) We choose the task of object detection [111, 139, 141, 359] and attribute prediction [3, 104, 233] on web-cross-stitch/images from the PASCAL dataset [94, 106]. These tasks make predictions about localized regions of an image.

5.2 Cross-stitch Networks

In this chapter, we present a novel approach to multi-task learning for ConvNets by proposing cross-stitch units. Cross-stitch units try to find the best shared representations for multi-task learning. They model these shared representations using linear combinations, and learn the optimal linear combinations for a given set of tasks. We integrate these cross-stitch units into a ConvNet and provide an end-to-end learning framework. We use detailed ablative studies to better understand these units and their training procedure. Further, we demonstrate the effectiveness of these units for two different pairs of tasks. To limit the scope of this paper, we only consider tasks which take the same single input, *e.g.*, an image as opposed to say an image and a depth-map [156].

5.2.1 Split Architectures

Given a single input image with multiple labels, one can design "Split architectures" as shown in Figure 5.2. These architectures have both a shared representation and a task specific representation.

'Splitting' a network at a lower layer allows for more task-specific and fewer shared layers. One extreme of Split architectures is splitting at the lowest convolution layer which results in two separate networks altogether, and thus only task-specific representations. The other extreme is using "sibling" prediction layers (as in [139]), which allows for a more shared representation. Thus, Split architectures allow for a varying amount of shared and task-specific representations.

5.2.2 Unifying Split Architectures

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

5.2.3 Cross-stitch units

Consider a case of multi task learning with two tasks A and B on the same input image. For the sake of explanation, consider two networks that have been trained separately for these tasks. We propose a new unit, *cross-stitch unit*, that combines these two networks into a multi-task network in a way such that the tasks supervise how much sharing is needed, as illustrated in Figure 5.3. At each layer of the network, we model sharing of representations by learning a linear combination of the activation maps [25, 209] using a *cross-stitch unit*. Given two activation maps $x_{\mathrm{A}}, x_{\mathrm{B}}$ from layer l for both the tasks, we learn linear combinations $\tilde{x}_{\mathrm{A}}, \tilde{x}_{\mathrm{B}}$ (Eq 5.1) of both the input activations and feed these combinations as input to the next layers' filters. This linear combination is parameterized using α. Specifically, at location (i, j) in the activation map,

$$\begin{bmatrix} \tilde{x}_{\mathrm{A}}^{ij} \\ \tilde{x}_{\mathrm{B}}^{ij} \end{bmatrix} = \begin{bmatrix} \alpha_{\mathrm{AA}} & \alpha_{\mathrm{AB}} \\ \alpha_{BA} & \alpha_{\mathrm{BB}} \end{bmatrix} \begin{bmatrix} x_{\mathrm{A}}^{ij} \\ x_{\mathrm{B}}^{ij} \end{bmatrix} \tag{5.1}$$

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

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

$$\begin{bmatrix} \frac{\partial L}{\partial x_{\mathrm{A}}^{ij}} \\ \frac{\partial L}{\partial x_{\mathrm{B}}^{ij}} \end{bmatrix} = \begin{bmatrix} \alpha_{\mathrm{AA}} & \alpha_{\mathrm{BA}} \\ \alpha_{\mathrm{AB}} & \alpha_{\mathrm{BB}} \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial \tilde{x}_{\mathrm{A}}^{ij}} \\ \frac{\partial L}{\partial \tilde{x}_{\mathrm{B}}^{ij}} \end{bmatrix} \tag{5.2}$$

$$\frac{\partial L}{\partial \alpha_{\mathrm{AB}}} = \frac{\partial L}{\partial \tilde{x}_{\mathrm{B}}^{ij}} x_{\mathrm{A}}^{ij}, \quad \frac{\partial L}{\partial \alpha_{\mathrm{AA}}} = \frac{\partial L}{\partial \tilde{x}_{\mathrm{A}}^{ij}} x_{\mathrm{A}}^{ij} \tag{5.3}$$

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

5.3 Design decisions for cross-stitching

We use the cross-stitch unit for multi-task learning in ConvNets. For the sake of simplicity, we assume multi-task learning with two tasks. Figure 5.4 shows this architecture for two tasks A and B. The sub-network in Figure 5.4(top) gets direct supervision from task A and indirect supervision (through cross-stitch units) from task B. We call the sub-network that gets direct supervision from task A as network A, and correspondingly the other as B. Cross-stitch units help regularize both tasks by learning and enforcing shared representations by combining activation (feature) maps. As we show in our experiments, in the case where one task has less labels than the other, such regularization helps the "data-starved" tasks.

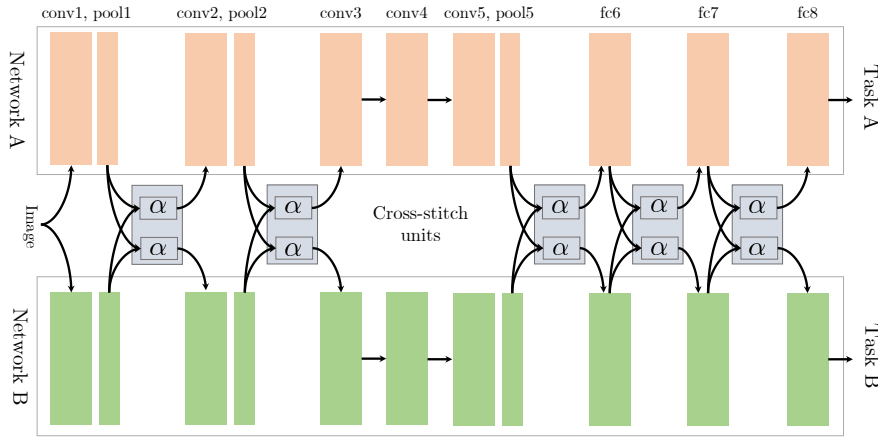


Figure 5.4: Using cross-stitch units to stitch two AlexNet [221] networks. In this case, we apply cross-stitch units only after pooling layers and fully connected layers. Cross-stitch units can model shared representations as a linear combination of input activation maps. This network tries to learn representations that can help with both tasks A and B. We call the sub-network that gets direct supervision from task A as network A (top) and the other as network B (bottom).

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

Cross-stitch units initialization and learning rates: The α values of a cross-stitch unit model linear combinations of feature maps. Their initialization in the range $[0, 1]$ is important for stable learning, as it ensures that values in the output activation map (after cross-stitch unit) are of the same order of magnitude as the input values before linear combination. We study the impact of different initializations and learning rates for cross-stitch units in Section 5.4.

Network initialization: Cross-stitch units combine together two networks as shown in Figure 5.4. However, an obvious question is – how should one initialize the networks A and B? We can initialize networks A and B by networks that were trained on these tasks separately, or have the same initialization and train them jointly.

5.4 Ablative analysis

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

Datasets and Tasks: For ablative analysis we consider the tasks of semantic segmentation (SemSeg) and Surface Normal Prediction (SN) on the NYU-v2 [374] dataset. We use the standard train/test splits from [122]. For semantic segmentation, we follow the setup from [155] and evaluate on the 41 classes using the standard metrics from their work

Setup for Surface Normal Prediction: Following [433], we cast the problem of surface normal prediction as classification into one of 20 categories. For evaluation, we convert the model predictions to 3D surface normals and apply the Manhattan-World post-processing following the method in [433]. We evaluate all our methods using the metrics from [122]. These metrics measure the error in the ground truth normals and the predicted normals in terms of their angular distance (measured in degrees). Specifically, they measure the mean and median error in angular distance, in which case lower error is better (denoted by 'Mean' and 'Median' error). They also report percentage of pixels which have their angular distance under a threshold (denoted by 'Within t°' at a threshold of $11.25^\circ, 22.5^\circ, 30^\circ$), in which case a higher number indicates better performance.

Networks: For semantic segmentation (SemSeg) and surface normal (SN) prediction, we use the Fully-Convolutional Network (FCN 32-s) architecture from [261] based on CaffeNet [195] (essentially AlexNet [221]). For both the tasks of SemSeg and SN, we use RGB cross-stitch/images at full resolution, and use mirroring and color data augmentation. We then finetune the network (referred to as *one-task network*) from ImageNet [79] for each task using hyperparameters reported in [261]. We fine-tune the network for semantic segmentation for 25k iterations using SGD (mini-batch size 20) and for surface normal prediction for 15k iterations (mini-batch size 20) as they gave the best performance, and further training (up to 40k iterations) showed no improvement. These *one-task networks* serve as our baselines and initializations for cross-stitching, when applicable.

Cross-stitching: We combine two AlexNet architectures using the cross-stitch units as shown in Figure 5.4. We experimented with applying cross-stitch units after every convolution activation map and after every pooling activation map, and found the latter performed better. Thus, the cross-stitch units for AlexNet are applied on the activation maps for `pool1`, `pool2`, `pool5`, `fc6` and `fc7`. We maintain one cross-stitch unit per 'channel' of the activation map, *e.g.*, for `pool1` we

Table 5.1: Initializing cross-stitch units with different α values, each corresponding to a convex combination. Higher values for α_{S} indicate that we bias the cross-stitch unit to prefer task specific representations. The cross-stitched network is robust across different initializations of the units.

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

have 96 cross-stitch units.

5.4.1 Initializing parameters of cross-stitch units

Cross-stitch units capture the intuition that shared representations can be modeled by linear combinations [209]. To ensure that values after the cross-stitch operation are of the same order of magnitude as the input values, an obvious initialization of the unit is that the α values form a convex linear combination, *i.e.*, the different-task α_{D} and the same-task α_{S} to sum to one. Note that this convexity is not enforced on the α values in either Equation 5.1 or 5.2, but serves as a reasonable initialization. For this experiment, we initialize the networks A and B with *one-task* networks that were fine-tuned on the respective tasks. Table 5.1 shows the results of evaluating cross-stitch networks for different initializations of α values.

5.4.2 Learning rates for cross-stitch units

We initialize the α values of the cross-stitch units in the range $[0.1, 0.9]$, which is about one to two orders of magnitude larger than the typical range of layer parameters in AlexNet [221]. While training, we found that the gradient updates at various layers had magnitudes which were reasonable for updating the layer parameters, but too small for the cross-stitch units. Thus, we use higher learning rates for the cross-stitch units than the base network. In practice, this leads to faster convergence and better performance. To study the impact of different learning rates, we again use a cross-stitched network initialized with two *one-task networks*. We scale the learning rates (wrt. the network's learning rate) of cross-stitch units in powers of 10 (by setting the `lr_mult` layer parameter in Caffe [195]). Table 5.2 shows the results of using different learning rates for the cross-stitch units after training for 10k iterations. Setting a higher scale for the learning rate improves performance, with the best range for the scale being $10^2 - 10^3$. We observed that setting the scale to an even higher value made the loss diverge.

Table 5.2: Scaling the learning rate of cross-stitch units wrt. the base network. Since the cross-stitch units are initialized in a different range from the layer parameters, we scale their learning rate for better training.

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

Table 5.3: We initialize the networks A, B (from Figure 5.4) from ImageNet, as well as task-specific networks. We observe that task-based initialization performs better than task-agnostic ImageNet initialization.

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

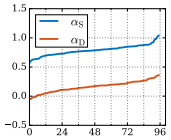
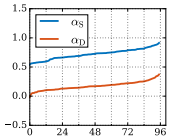
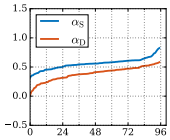
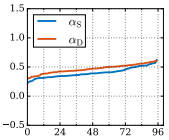
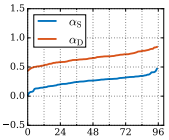
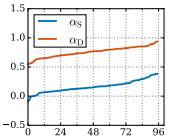
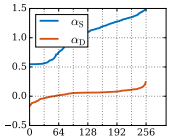
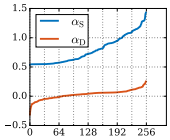
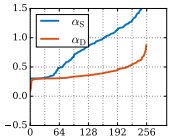
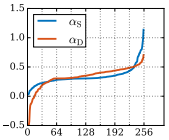
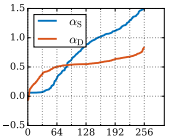
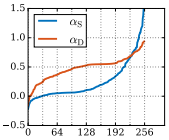
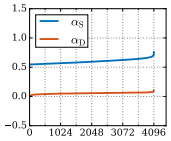
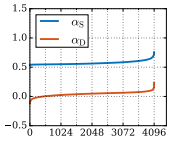
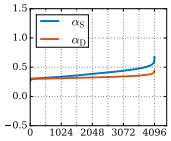
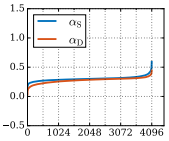
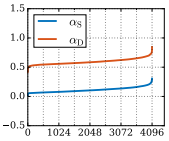
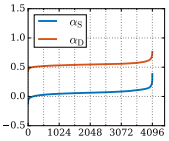
5.4.3 Initialization of networks A and B

When cross-stitching two networks, how should one initialize the networks A and B? Should one start with task specific *one-task networks* (fine-tuned for one task only) and add cross-stitch units? Or should one start with networks that have not been fine-tuned for the tasks? We explore the effect of both choices by initializing using two *one-task networks* and two networks trained on ImageNet [79, 351]. We train the *one-task* initialized cross-stitched network for 10k iterations and the ImageNet initialized cross-stitched network for 30k iterations (to account for the 20k fine-tuning iterations of the *one-task* networks), and report the results in Table 5.3. Task-specific initialization performs better than ImageNet initialization for both the tasks, which suggests that cross-stitching should be used after training task-specific networks.

5.4.4 Visualization of learned combinations

We visualize the weights α_{S} and α_{D} of the cross-stitch units for different initializations in Figure 5.4. For this experiment, we initialize sub-networks A and B using *one-task* networks and trained the cross-stitched network till convergence. Each plot shows (in sorted order) the α values for all the cross-stitch units in a layer (one per channel). We show plots for three layers: `pool1`, `pool5` and `fc7`. The initialization of cross-stitch units biases the network to start its training preferring a certain type of shared representation, *e.g.*, $(\alpha_{\mathrm{S}}, \alpha_{\mathrm{D}}) = (0.9, 0.1)$ biases the network to learn more task-specific features, while $(0.5, 0.5)$ biases it to share representations. Figure 5.4 (second row) shows that both the tasks, across all initializations, prefer a more task-specific representation for `pool5`, as shown by higher values of α_{S}. This is inline with the observation from Section 5.0.1 that Split `conv4` performs best for these two tasks. We also notice that the surface normal task prefers

Table 5.4: We show the sorted α values (increasing left to right) for three layers. A higher value of α_S indicates a strong preference towards task specific features, and a higher α_D implies preference for shared representations. More detailed analysis in Section 5.4.4. Note that both α_S and α_D are sorted independently, so the channel-index across them do not correspond.

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

shared representations as can be seen by Figure 5.4(b), where α_S and α_D values are in similar range.

5.5 Experiments

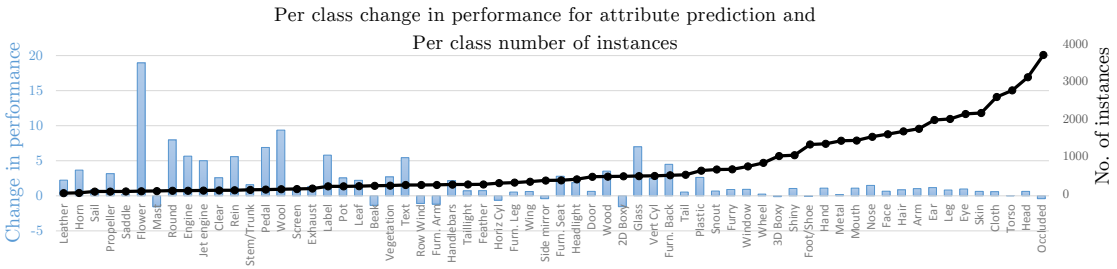


Figure 5.5: Change in performance for attribute categories over the baseline is indicated by blue bars. We sort the categories in increasing order (from left to right) by the number of instance labels in the train set, and indicate the number of instance labels by the solid black line. The performance gain for attributes with lesser data (towards the left) is considerably higher compared to the baseline. We also notice that the gain for categories with lots of data is smaller.

We now present experiments with cross-stitch networks for two pairs of tasks: semantic segmentation and surface normal prediction on NYU-v2 [374], and object detection and attribute prediction on PASCAL VOC 2008 [94, 106]. We use the experimental setup from Section 5.4 for semantic segmentation and surface normal prediction, and describe the setup for detection and attribute prediction below.

Dataset, Metrics and Network: We consider the PASCAL VOC 20 classes for object detection, and the 64 attribute categories data from [106]. We use the PASCAL VOC 2008 [94, 106] dataset for our experiments and report results using the standard Average Precision (AP) metric. We start with the recent Fast-RCNN [139] method for object detection using the AlexNet [221] architecture.

Training: For object detection, Fast-RCNN is trained using 21-way 1-vs-all classification with 20 foreground and 1 background class. However, there is a severe data imbalance in the foreground and background data points (boxes). To circumvent this, Fast-RCNN carefully constructs mini-batches with $1:3$ foreground-to-background ratio, *i.e.*, at most 25% of foreground samples in a mini-batch. Attribute prediction, on the other hand, is a multi-label classification problem with 64 attributes, which only train using foreground bounding boxes. To implement both tasks in the Fast R-CNN framework, we use the same mini-batch sampling strategy; and in every mini-batch only the foreground samples contribute to the attribute loss (and background samples are ignored).

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

Cross-stitching: We combine two AlexNet architectures using the cross-stitch units after every pooling layer as shown in Figure 5.4. In the case of object detection and attribute prediction, we use one cross-stitch unit per layer activation map. We found that maintaining a unit per channel, like in the case of semantic segmentation, led to unstable learning for these tasks.

5.5.1 Baselines

We compare against four strong baselines for the two pairs of tasks and report the results in Table 5.5 and 5.6.

Single-task Baselines: These serve as baselines without benefits of multi-task learning. First we evaluate a single network trained on only one task (denoted by 'One-task') as described in Section 5.4. Since our approach cross-stitches two networks and therefore uses $2\times$ parameters, we also consider an ensemble of two one-task networks (denoted by 'Ensemble'). However, note that the ensemble has $2\times$ network parameters for only one task, while the cross-stitch network has roughly $2\times$ parameters for two tasks. So for a pair of tasks, the ensemble baseline uses $\sim 2\times$ the cross-stitch parameters.

Multi-task Baselines: The cross-stitch units enable the network to pick an optimal combination of shared and task-specific representation. We demonstrate that these units remove the need for finding such a combination by exhaustive brute-force search (from Section 5.0.1). So as a baseline, we train all possible "Split architectures" for each pair of tasks and report numbers for the best Split for each pair of tasks.

There has been extensive work in Multi-task learning outside of the computer vision and deep learning community. However, most of such work, with publicly available code, formulates multi-task learning in an optimization framework that requires all data points in memory [54, 98, 151, 235, 390, 470, 471]. Such requirement is not practical for the vision tasks we consider.

So as our final baseline, we compare to a variant of [3, 472] by adapting their method to our setting and report this as 'MTL-shared'. The original method treats each category as a separate 'task', *a separate network* is required for each category and *all these networks are trained jointly*. Directly applied to our setting, this would require training 100s of ConvNets jointly, which is impractical.

Table 5.5: Surface normal prediction and semantic segmentation results on the NYU-v2 [374] dataset. Our method outperforms the baselines for both the tasks.

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

Thus, instead of treating each category as an independent task, we adapt their method to our two-task setting. We train these two networks jointly, using end-to-end learning, as opposed to their dual optimization to reduce hyperparameter search.

5.5.2 Semantic Segmentation and Surface Normal Prediction

Table 5.5 shows the results for semantic segmentation and surface normal prediction on the NYUv2 dataset [374]. We compare against two one-task networks, an ensemble of two networks, and the best Split architecture (found using brute force enumeration). The sub-networks A, B (Figure 5.4) in our cross-stitched network are initialized from the one-task networks. We use cross-stitch units after every pooling layer and fully connected layer (one per channel). Our proposed cross-stitched network improves results over the baseline one-task networks and the ensemble. Note that even though the ensemble has $2\times$ parameters compared to cross-stitched network, the latter performs better. Finally, our performance is better than the best Split architecture network found using brute force search. This shows that the cross-stitch units can effectively search for optimal amount of sharing in multi-task networks.

5.5.3 Data-starved categories for segmentation

Multiple tasks are particularly helpful in regularizing the learning of shared representations[46, 98, 406]. This regularization manifests itself empirically in the improvement of "data-starved" (few examples) categories and tasks.

For semantic segmentation, there is a high mismatch in the number of labels per category (see the black line in Figure 5.6). Some classes like *wall*, *floor* have many more instances than other classes like *bag*, *whiteboard etc*. Figure 5.6 also shows the per-class gain in performance using our method over the baseline one-task network. We see that cross-stitch units considerably improve the performance of "data-starved" categories (*e.g.*, *bag*, *whiteboard*).

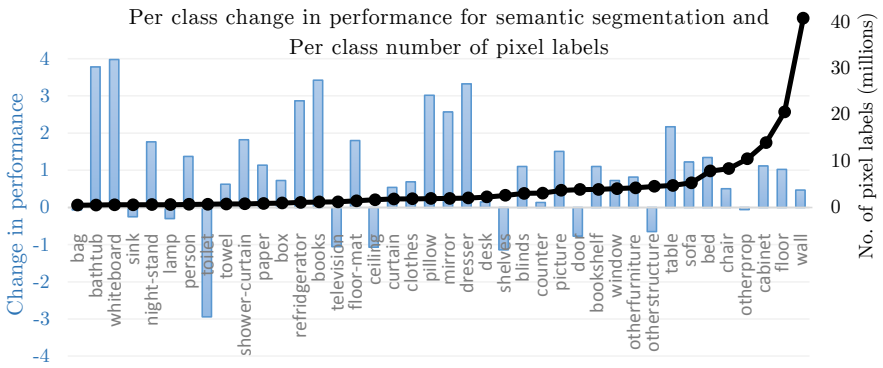


Figure 5.6: Change in performance (meanIU metric) for semantic segmentation categories over the baseline is indicated by blue bars. We sort the categories (in increasing order from left to right) by the number of pixel labels in the train set, and indicate the number of pixel labels by a solid black line. The performance gain for categories with lesser data (towards the left) is more when compared to the baseline one-task network.

Table 5.6: Object detection and attribute prediction results on the attribute PASCAL [106] 2008 dataset

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

5.5.4 Object detection and attribute prediction

We train a cross-stitch network for the tasks of object detection and attribute prediction. We compare against baseline one-task networks and the best split architectures per task (found after enumeration and search, Section 5.0.1). Table 5.6 shows the results for object detection and attribute prediction on PASCAL VOC 2008 [94, 106]. Our method shows improvements over the baseline for attribute prediction. It is worth noting that because we use a background class for detection, and not attributes (described in 'Scaling losses' in Section 5.5), detection has many more data points than attribute classification (only 25% of a mini-batch has attribute labels). Thus, we see an improvement for the data-starved task of attribute prediction. It is also interesting to note that the detection task prefers a shared representation (best performance by Split `fc7`), whereas the attribute task prefers a task-specific network (best performance by Split `conv2`).

5.5.5 Data-starved categories for attribute prediction

Following a similar analysis to Section 5.5.3, we plot the relative performance of our cross-stitch approach over the baseline one-task attribute prediction network in Figure 5.5. The performance gain for attributes with smaller number of training examples is considerably large compared to the baseline (**4.6**% and **4.3**% mAP for the top 10 and 20 attributes with the least data respectively). This shows that our proposed cross-stitch method provides significant gains for data-starved tasks by learning shared representations.

5.6 Discussion

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

Chapter 6

Scaling to Multiple Tasks at Inference

The era of the cloud's total dominance is drawing to a close. The rise of the "internet of things" is one reason why computing is emerging from the centralised cloud and moving to an "edge" of networks and intelligent devices.

The Economist, 2018

In the previous chapter, we showed how using multiple tasks can help improve performance for data-starved tasks. The 'cross-stitch' unit mechanism we introduced was automatically able to find the optimal multi-task architecture for a ConvNet. At training time, this makes the architectural search process both easy and efficient. However, at inference time, the number of parameters of the final multi-task model scale linearly with the number of tasks. This makes inference expensive and is a major bottleneck especially on devices which have limited computation power. In this chapter, we see how to make efficient inference while scaling to multiple tasks. This chapter focuses on video processing on edge devices, *e.g.*, a parking lot camera that records video and processes it to report open parking spots, fender benders, driver behaviors *etc*. These edge devices are a common way to address bandwidth limitations, intermittent connectivity (e.g., in drones), and real-time requirements. Applications executing at the edge, though, face tighter bounds on compute resource availability than in datacenters. These devices have limited compute and power which requires computationally efficient algorithms. Before tuning algorithms for these scenarios, we should ask ourselves how widespread are these devices compared to the traditional compuing hardware (datacenter machines) that we need to care about tuning performance for them? If one is to believe technological predictions (quote from *The Economist* above), 'edge computing' will soon dominate the traditional datacenter based 'cloud computing'. Naturally, optimal video application performance requires tuning for such limited compute resources [74, 159, 208, 462, 483].

Unfortunately, what resources will be available to the application at deployment time is often unknown to the developer. Further, resource availability changes as additional applications arrive and depart. Instead, individual application developers typically develop their models in isolation, assuming either infinite resources or a predetermined resource allotment. When a number of separately tuned models are run concurrently, resource competition forces the video stream to be analyzed at a lower frame rate—leading to unsatisfactory results for the running applications, as frames are dropped and events in those frames are missed. However, due to the popularity of

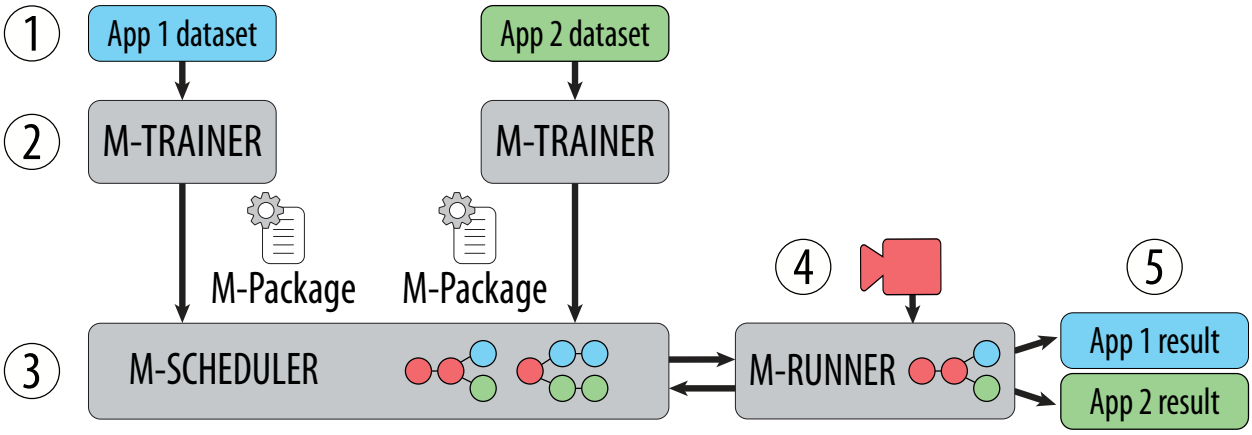


Figure 6.1: Mainstream Architecture. Offline, for each task, M-Trainer takes a labeled dataset and outputs an M-Package. M-Scheduler takes independently generated M-Packages, and chooses the task-specific degree of specialization and frame rate. M-Scheduler deploys the unified multi-task model to M-Runner, performing inference on the edge.

transfer learning (Sec. 6.1) [308, 333, 377, 454], contention can be reduced by eliminating redundant computation between concurrent applications [159].

Mainstream is a new system for video processing that addresses resource contention by dynamically tuning degrees of work sharing among concurrent applications. Specifically, it focuses on sharing portions of DNN inference, which consumes the majority of video processing cycles. Mainstream exploits the potential "shared stem" of computation that results from application developers' use of the standard DNN training approach of transfer learning. In transfer learning, training begins with an existing, *pre-trained* DNN, which is then re-trained for a different task. Typically, only a subset of the pre-trained DNN is specialized; when different applications start with the same pre-trained DNN, Mainstream identifies the common layers and executes them only once per frame.

A critical challenge of exploiting shared stems well is determining how much to share. Application developers usually specialize as much of the pre-trained DNN as is necessary to achieve high model accuracy. More specialization, however, means that less of the pre-trained DNN can be shared. Thus, there is an explicit trade-off between the benefits of greater per-frame accuracy (via more-specialized DNNs) and processing more frames of the input video stream (via more sharing of less-specialized DNNs). The right choice depends on the edge device resources, the number of concurrent applications, and their individual characteristics.

Deployment time model selection. Mainstream moves the final DNN model selection step from application development time to deployment time, when the hardware resources and concurrent application mix are known. By doing so, Mainstream has the necessary information to select the right amount of DNN specialization (and thus sharing) for each application. As applications come and go, Mainstream can dynamically modify its choices. Previous systems like VideoStorm [462] select models by considering each application independently. *The specialization-vs-sharing trade-off, however, can only be optimized when considering applications jointly.* Joint optimization produces a combinatorial set of options, which Mainstream navigates using application metadata and domain-

Figure 6.2: Example computation pipeline for event detection.

specific models; the system uses this information to estimate the effects of DNN specialization and frame sampling rate on application performance. Unlike black-box approaches, Mainstream can jointly optimize for stem-sharing without needing to profile each combination.

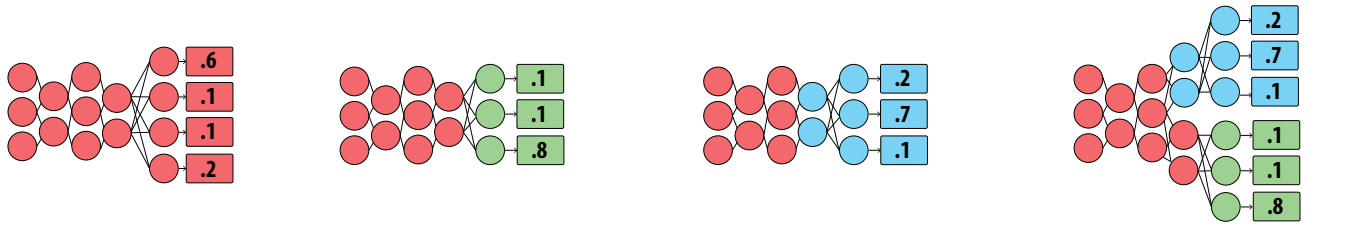
Mainstream consists of three main parts (Fig. 6.1). The *M-Trainer* toolkit helps application developers manage their training process to produce the information needed to allow tuning the degree of specialization at deployment time. Current standard practice is for developers to experiment with different model types, hyperparameters, and degrees of re-training to find the best choice for an assumed resource allocation, discarding the trained DNN models not chosen. M-Trainer instead keeps "less optimal" *candidate* DNN models, together with associated training-time information (e.g., per-frame accuracy, event detection window). The *M-Scheduler* uses this information, together with a profile of per-layer runtime on the target edge device, to determine the best candidate for each application—including the degrees of specialization and, thus, sharing. It efficiently searches the option space to maximize application quality (e.g., average F1 score among the applications). The *M-Runner* runtime system runs the selected DNNs, sharing the identical unspecialized layers.

Experiments with several datasets and event detection tasks on an edge node confirm the importance of making deployment-time decisions and the effectiveness of Mainstream's approach. Results show that dynamic selection of shared stems can improve F1-scores by up to 87X relative to the common approach of using fully-independent per-application DNNs (No-Sharing) and up to 47% compared to a static approach of retraining only the last DNN layer and sharing all others (Max-Sharing). Across a range of concurrent applications, Mainstream adaptively selects a balance between per-frame accuracy and frame sampling rate that consistently provides superior performance over such static approaches.

Contributions. This chapter makes three main contributions. First, it highlights the critical importance of reducing aggregate per-frame CPU work of multiple independently developed video processing applications via stem-sharing; No-Sharing is unable to support even three concurrent applications on our edge node deployment. Second, it identifies the goodness trade-off between per-frame quality and the frame sampling rate dictated by the degree of DNN specialization (and thus the amount of sharing). Third, it describes and demonstrates the efficacy of the Mainstream approach for automatically deploying the right degree of specialization for each submitted application's DNN.

6.1 Background

DNNs are a powerful tool used in computer vision tasks such as human action recognition [377], object detection [142], scene geometry estimation [91], face recognition [400], etc. Fig. 6.2 shows a typical computation pipeline for an image classification application. Although frame ingest and



(a) Base DNN model (b) App. #1's DNN model (c) App. #2's DNN model As executed in Mainstream

Figure 6.3: (a) depicts a base DNN trained from scratch for its task. (b) and (c) show two new task DNNs, fine-tuned against the base DNN. App. #1 freezes more layers during training than App. #2. (d) shows how Mainstream runs the applications concurrently. Layers frozen by both App. #1 and App. #2 can be shared.

image preprocessing are necessary stages of computation, they are low cost and easily shared between concurrent applications. DNNs, on the other hand, are typically unique to each application and computationally expensive: in one image classification application we run, the DNN inference incurs 25X more latency than the preprocessing steps.

DNNs and transfer learning. A machine learning (ML) *model* is a parameterized function that performs a task. *Training* is the process of learning parameter values (called weights) such that the model will approximate the desired function with some measure of accuracy. For example, when training an image classifier, one might examine labeled input images and use gradient descent to find a set of weights that minimizes a loss function over the labels. Using the trained model to find the function's output given a new, unlabeled input is called *inference*.

DNNs are a class of ML models that usually have a large input space, such as the pixels of an image. A DNN can be represented by a graph where nodes are organized into *layers*; each node computes a function of its inputs, which are outputs from the previous layer.

The "deep" in DNNs refers to their many layers. Increasingly, successful applications of DNNs have largely been the result of building models with more layers that take larger vectors of inputs [166, 221, 375, 399]. The success of these models has hinged critically upon the arrival of very large, labeled datasets for training [6, 78, 253].

Training these large models is notoriously hard. One often lacks sufficient labeled data or computational resources to train such a model. *Transfer learning* is an alternative to training a model from scratch. Here, a model that has already been trained on a similar task (a *base DNN* as in Fig. 6.3(a)), is used as an initialization point or feature extractor for the new *target DNN*. During training, a subset of the old parameters are *frozen* and do not change. The remaining free parameters are then retrained on the new task with a new training dataset (Fig. 6.3(b) and Fig. 6.3(c)). This process *fine-tunes* these parameters to achieve a result comparable to end-to-end training on the entire DNN, but does so with much less data and at a much lower computational cost. In practice, few practitioners train networks from scratch, let alone develop novel network architectures. Transfer learning via one of a few popular neural networks is standard practice.

DNNs for image classification and event detection. Although we believe that Mainstream's approach is generally applicable to video stream analysis, in this chapter, we focus on applications

Architecture	Number of Layers	ImageNet Top-1 Accuracy (%)
InceptionV3	314	78.0
MobileNets-224	84	70.7
ResNet-50	177	75.6

Table 6.1: Top-1 accuracy of three neural networks architectures trained on the ImageNet dataset.

that use image-classification DNNs to perform event detection.

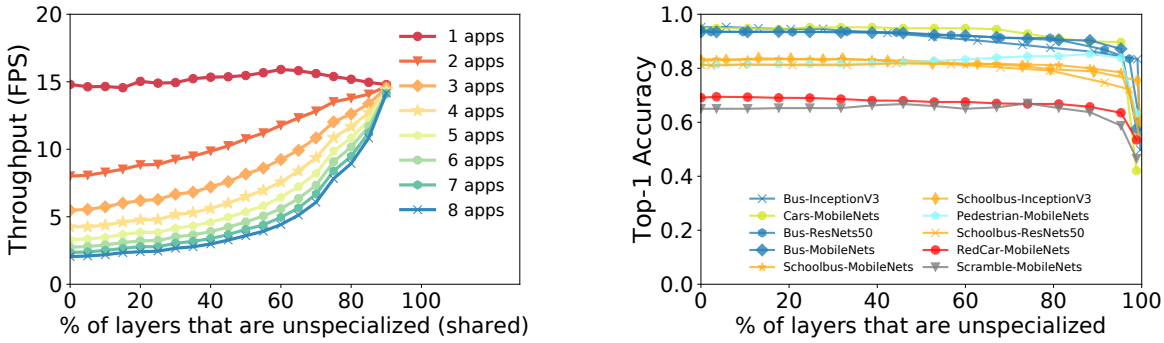
Image classification aims to assign one label from a set of categories or *classes* to each image. For example, a 10-class classifier takes an input image and returns a 10-item vector of probabilities representing the likelihood that the image belongs to each class. *Top-N accuracy* is the probability that the correct label is among the top-N highest probability output labels. So, Top-1 accuracy indicates the fraction of images that the model classifies correctly. We refer to this metric as the *per-frame accuracy* in the context of video classification. Popular neural network architectures for image classification include ResNet [166], InceptionV3 [399], and MobileNets [178]. Table 6.1 describes these three neural networks and their Top-1 accuracy achieved on the ImageNet dataset [78]. Networks trained on ImageNet are popular base-DNNs for image classification tasks.

We define an event as a contiguous group of frames containing some visible phenomenon that we are trying to identify: e.g., a cyclist passing by, or a puff of smoke being emitted. One way of doing event detection is to perform image classification across a sequence of frames. An event is detected if at least one of the contiguous frames is sampled, analyzed, and correctly labeled. Previous works [241] have also used this existence metric to measure recall and precision of range-based queries. (Event detection is not to be confused with object detection, where the goal is to locate an object in a single frame. Indeed, object detection is another way of performing event detection.) We evaluate event classification applications by measuring the *event F1-score*, the harmonic mean between *event recall* and *event precision*. The event recall reports the proportion of ground truth events identified. The event precision reports the proportion of classified events that are true positives. Note that these metrics are relative to the detection of events across multiple frames and are distinct from per-frame metrics (e.g., Top-1 accuracy).

6.2 Mainstream Approach: What and Why

Sharing computation between DNNs. When supporting multiple inference applications on a single infrastructure, the common approach is to execute every application's DNN model independently. We refer to this as a "No Sharing" approach. To avoid redundant work, Mainstream instead computes results for DNN layers common to multiple concurrent applications just once and distributes the outputs of shared stems to the specialized layers of all sharing applications. This is analogous to common subexpression elimination used in other domains, e.g., optimizing compilers or database query planners.

Fig. 6.3 illustrates how compute sharing can be leveraged when two DNNs are fine-tuned



(a) Throughput vs. Sharing

(b) Per-frame accuracy vs. (Potential) sharing

Figure 6.4: Conflicting consequences of DNN compute sharing. (a) shows the frame processing rate for 1–8 concurrent event detection applications as a function of the fraction of the InceptionV3 DNN they share, from No-Sharing on the left to sharing all but the last layer (Max-Sharing) on the right. The experiments are run on the hardware described in Section 6.6. (b) shows Top-1 accuracy as a function of the fraction of unspecialized layers for three popular DNN architectures (ResNet-50 [166], InceptionV3 [399], and MobileNets-224 [178]) using six of the datasets described in Table 6.2. We trained each classifier using all three network architectures but omitted some curves for brevity. The horizontal axis starts from fully specialized DNNs on the left to only the last layer being specialized on the right; recall that potential computation sharing is limited to the unspecialized layers.

from a common pre-trained model and have some unspecialized layers in common. Compute sharing can significantly affect per-frame computation cost and improve throughput for a given CPU resource. Fig. 6.4a quantifies this effect. It shows the throughput achieved by Mainstream running up to eight concurrent InceptionV3-based event detection pipelines, as a function of how many DNN layers they have in common (i.e., their common degree of specialization). With no sharing (the left-most points), adding a second application halves throughput, which continues to degrade geometrically as more applications are added. Moving to the right, throughput improves as more layers are shared. When all but the last layer are shared, additional applications can be run at very low marginal cost.

On the other hand, there are costs to enabling sharing by leaving many layers unspecialized. In particular, the per-frame accuracy of a model may be lower when only a few layers are specialized. Fig. 6.4b shows the effect of specialization on per-frame accuracy for several combinations of DNN architectures and classification tasks. As expected, accuracy decreases slowly as less-specialized networks are employed (and hence more sharing is enabled)— with a large decrease occurring only when the fraction of the network specialized is very small. This characteristic enables Mainstream to share large portions of the network with low accuracy loss.

Adaptive management of the sharing opportunity. Since transfer learning is so commonly used by ML developers, and base models are shared within organizations and on the Internet, there may be many opportunities to exploit inter-DNN redundancy in the unspecialized layers. Most developers either use a popular default of specializing only the last layer (which is great for sharing potential, at the potential cost of model accuracy) or determine the degree of specialization based on the amount of training data available, since retraining too many layers without sufficient training data leads to over-fitting. Notably, each developer decides independently.

The problem with this approach is that the impact of sharing computation on application quality depends on factors only known at deployment time: the set of concurrent applications and the resources of the edge node on which they are run. Hence, Mainstream defers the decision regarding how much specialization to employ from application development time to deployment time.

Impact of sampling rate for event detection. Given the trade-off between per-frame accuracy and frame processing throughput, picking the right degree of specialization is challenging. Consider an application for monitoring environmental pollution from trains, which is being built using a train detector we deployed. When the application detects a train coming into view, it triggers the capture of high fidelity frames of the train's smoke stack (for subsequent pollution analysis).

Increasing specialization to improve per-frame accuracy increases the probability of correctly classifying frames containing trains— but reduces shared computation. This, in turn, leads to less frequent sampling, which removes opportunities to analyze frames containing a particular view of a train. A higher frame rate increases the probability that an event will be observed in more frames, creating more opportunities for detection. So, the question becomes: should one sample more frames using a less accurate model or sample fewer frames using a more accurate model?

Analytical model for event detection. The Mainstream scheduler (Sec. 6.5) navigates this "accuracy vs. sampling rate" space by evaluating various candidate *{specialization, frame rate}* tuples. To do so, however, the scheduler must be able to interpret the benefit at the application (not per-frame) level. Hence, we propose an analytical model (sketched in Equations 6.1-6.4, below) that approximates the *event* F1-score for a given DNN, given estimates of (a) event length, (b) event frequency, (c) the correlation between frames (discussed below), (d) per-frame DNN accuracy, and (e) DNN analysis frame rate. The frame rate (e) comes directly from the scheduler's proposed tuple; similarly, the accuracy (d) associated with a given specialization proposal will be available to the scheduler (see Sec. 6.4). Values for event length (a), frequency (b), and correlation (c) can either be measured using representative video samples, or they can be estimated by the developer.

We are able to predict the application's F1-score by estimating the expected number of frames per event that we will have the opportunity to analyze and computing the probability that analyzing the set of frames will result in a detection. The expected number of frames analyzed per event is dependent on the event length and frame rate. The per-frame Top-1 accuracy represents the probability that we will classify any individual frame correctly. However, this does not factor in the fact that sequential frames of an event may be correlated in some way. We therefore introduce and estimate the *inter-frame correlation*, which measures the marginal benefit of analyzing more frames of a single event.

The inter-frame correlation, $corr$, is based on conditional probabilities. For frames corresponding to an event, if P(X_i) and P($\neg X_i$) are the probabilities of detecting or not detecting the event in frame i, respectively, then $P(\neg X_2 \mid \neg X_1)$ is the probability of not detecting the event in frame X_2, given that we did not detect it in frame X_1. This conditional probability can be measured empirically from training data. Relying on the Kolmogorov definition, we can calculate the probability the event is detected in at least one of the two frames as $1 - P(\neg X_2 \mid \neg X_1) * P(\neg X_1)$. This logic can be extended to approximate the probability of detecting the event in N tries and to estimate recall.

To estimate recall, we calculate the probability that our DNN will correctly classify at least one

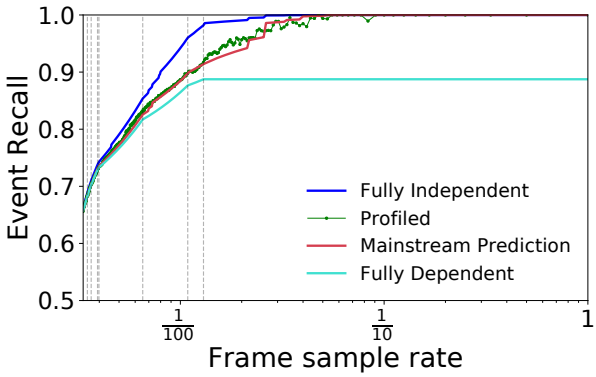


Figure 6.5: Effect of sample-rate on recall, for different inter-frame correlations. The dotted vertical lines represent each train in the dataset, denoting $\frac{1}{trainlength}$, which is the sample rate required to ensure that one frame of that train is analyzed. The "Profiled" line is measured directly from the TRAIN video dataset, and the other three are approximations based on different inter-frame correlations (uncorrelated, fully correlated, and the empirical correlation observed in the TRAIN dataset).

frame of an event using the following steps:

$$N = \begin{cases} \lceil \frac{d}{stride} \rceil & \text{w.p. } \frac{d-(stride\%d)}{stride} \\ \lfloor \frac{d}{stride} \rfloor & else \end{cases} \tag{6.1}$$

$$P_{miss\_1} = 1 - accuracy_{per-frame} \tag{6.2}$$

$$P_{miss\_N} = corr^{N-1} * P_{miss\_1} \tag{6.3}$$

$$recall = 1 - P_{miss\_N} \tag{6.4}$$

We use Eq. 6.1 to calculate N, the expected number of frames of the event that the model will process. Here, d is the event length, and $stride$ is the inverse of the frame rate. Equation 6.3 estimates the probability the DNN misclassifies all N analyzed frames. Recall is the complement: the probability that we correctly classify at least one frame of the event.

To estimate the false positive rate, we repeat this calculation, except that d is the number of frames between events (derived from the event frequency), and P_{miss_1} is the probability of *true* negatives. The true positive rate, the false positive rate, and the event frequency are used to calculate the precision. The F1-score is the harmonic mean between precision and recall.

To evaluate our model, we profile an application and measure the actual recall observed when running the event detector application (e.g., train detection) on the video stream at different sampling rates (Fig. 6.5). This was measured by averaging the recall achieved from 10,000 trials of sampling at each sample rate. The result is plotted by the "Profiled" line. Our analytical model ("Mainstream Prediction") is sufficiently accurate to describe the complex relationship between frame sample rate and application recall. Mainstream uses this analytical model to efficiently optimize for the trade-off between per-frame accuracy and frame rate.

We use Mainstream for event detection but believe its approach can be generally applied to DNN-based tasks where application quality depends not only on its model, but also on its input sampling rate (e.g., object tracking, action recognition.)

6.3 Mainstream Architecture

We have developed Mainstream, a training and runtime system for DNN-based live analytics of camera streams, which (a) enables efficient sharing of computation between detection applications, (b) maximizes event F1-score across all tasks, and (c) allows each task to be independently developed, trained, and deployed. Fig. 6.1 shows the architecture of Mainstream, which consists of three components: M-Trainer, M-Scheduler, and M-Runner.

To deploy a new application to Mainstream, the user provides a corresponding labeled training dataset to their local instance of M-Trainer (Step 1). M-Trainer uses the dataset to train a number of potential models, with varying numbers of specialized layers. This *Model Set* and associated metadata are then assembled into an "M-Package" (Step 2). Note that these are offline steps, performed just once per application prior to deployment, independent of all other tasks. At runtime, M-Scheduler uses the M-Packages of all currently-deployed applications to determine, for each application, the degree of DNN sharing and sampling rate such that, across all applications, the event F1-score is maximized, subject to the resource limits of the edge platform (Step 3). M-Scheduler runs in the datacenter, and is executed once for each scheduling event (e.g., a change in the deployed set of applications, or in available hardware resources). M-Runner then executes the selected model configuration on edge devices (Step 4) and returns app-specific results in real-time (Step 5).

M-Runner is a relatively straightforward execution system for running visual processing pipelines. It accepts a DAG, where each node represents a unit of independent computation, and connections represent data flow. Fig. 6.2 illustrates the logical DAG for an image classification application. Most of the computation is expected in the "DNN" process, which evaluates the merged DNN of all concurrent tasks. This combined DNN, as illustrated in Fig. 6.3(d), represents the set of models selected by M-Scheduler across all tasks. This DNN is structured as a tree, with sets of layers branching from the shared stem. M-Runner executes the shared stem once per frame, reducing the total processing costs of the deployed tasks.

We next describe how M-Trainer independently trains model candidates for potential sharing (Sec. 6.4) and how M-Scheduler dynamically chooses among them (Sec. 6.5).

6.4 Distributed Sharing-Aware Training

M-Trainer produces a set of models for each task so that they can be combined dynamically at runtime to maximize collective performance. Application developers use M-Trainer independently at different times and locations.

One approach to sharing computation between applications would be to jointly train them us-

ing a multi-task network. This, however, requires centralized training of applications. MCDNN [159] proposed fine-tuning models independently and sharing the unspecialized DNN layers. This static approach prevents M-Scheduler from dynamically tailoring stem-sharing to the available resources. In contrast, M-Trainer generates a *set of models* that vary in the number of specialized layers. These models compose an application-specific *Model Set*. The generation of Model Sets allows for the late binding of the degree of specialization to deployment time, when the platform characteristics and co-deployed applications are known.

To construct a Model Set, M-Trainer first analyzes the structure of the base DNN to identify *branchpoints*, the potential boundaries between frozen and specialized layers. Using the app-specific training data provided by the developer, M-Trainer generates a set of fine-tuned DNN models, one per branchpoint, where layers up to the branchpoint are frozen, and the rest are specialized. Only the models at the Pareto-optimal frontier with respect to number of layers specialized and estimated accuracy are actually included in the M-Package. This eliminates from consideration models that reduce accuracy, while requiring more specialization. For example, an overfitted model, caused by specializing too many layers with insufficient training data, will not be included.

Model Sets are bundled together with application metadata into an *M-Package*. This metadata includes the measured per-frame accuracy of each model (we use a portion of the data as the validation dataset.) The expected minimum event duration, event frequency, and inter-frame correlation are optionally measured from the training data and included in the M-Package, or directly provided by the application developer.

The construction of the M-Package is an offline operation, which is run just once per application. For each application, M-Trainer must train multiple models. Although training a model from scratch can be very resource intensive, fine-tuning is much quicker. M-Trainer creates Model Sets with 15 model options in 8 hours on a single GPU (Sect. 6.6). Note that the computation for generating all of the models is easily parallelized in a datacenter setting, and may not be significantly higher than traditional fine-tuning. For example, to find the right number of layers to specialize in order to maximize accuracy, one may need to generate these models anyway. The key difference here is that intermediate runs are not discarded, and the final selection is made at run time by M-Scheduler.

As each application's models are independently trained and analyzed, no coordination or sharing of training data is needed between developers of different tasks. The resulting M-Package, however, contains enough information that M-Scheduler, at run time, can optimize across the independently-developed tasks.

6.5 Dynamic Sharing-Aware Scheduling

At each *scheduling event* (typically, an application submission or termination), M-Scheduler uses the M-Packages created by the various per-application M-Trainers to produce a new overall schedule that optimizes some global objective function, subject to resource constraints (currently, M-Scheduler maximizes average event F1-score across applications). The *schedule* consists of a DNN model selection (indicating the degree of sharing) and target frame-rate for every running

application.[1] The final schedule is a tree-like model with applications splitting from a shared stem at potentially different branch points, with each application able to run at its own frame rate.

M-Scheduler optimization algorithm. With N applications to schedule, S possible specialization settings per application, and R frame-rate settings per application, the number of possible schedules is $(S \cdot R)^N$. Although this space is large (e.g., $N \approx 10$, $R \approx 10$, and $S \approx 10$ in our experiments), M-Scheduler can efficiently determine a good schedule using a greedy heuristic (Algo. 1). We compare the schedules generated by our greedy scheduler to those of an exhaustive scheduler in >4,800 workloads each consisting of up to 10 applications, and find that the greedy schedules are on average within 0.89% of optimal.

Algorithm 1 Scheduler optimization algorithm

```
function GET NEXT MOVE(schedule)
                                  ▷ Finds change to schedule with the highest benefit/cost
end function
function SCHEDULE(budget)
    sched ← GET SCHEDULE(max_sharing, min_fps)
    while True do
        next_move ← GET NEXT MOVE(sched)
        cost ← cost + GET COST(next_move)
        if cost < budget then
            sched ← UPDATE SCHEDULE(next_move)
        else return sched
        end if
    end while
end function
```

Essentially, at each step of our iterative algorithm, the scheduler considers making a *move* which improves the application quality of a single application by tweaking its frame rate and/or model specialization. The algorithm greedily selects the move that yields the best ratio of *benefit* to *cost*, defined below. Naturally, before this iterative refinement, the schedule is initialized to the lowest cost configuration— Max-Sharing with minimum frame rate. At any iteration step, the number of possible moves is bounded by $S \cdot R \cdot N$. The total number of moves per invocation of the scheduler is similarly bounded by $S \cdot R \cdot N$, but in practice is much fewer as the set of potential moves that fit within the computational budget is exhausted.

Measuring the Benefit of a Move. The benefit associated with a move captures the improvement in F1-score for the application associated with that move. This value is calculated using the analytical model presented in Sect. 6.2 and the application metadata in the M-Package.

Measuring the Cost of a Move. The cost value considered represents the computational resources (e.g., CPU-seconds per second) consumed by a given schedule arrangement and depends on the number of shared subgraphs, the number of task-specific subgraphs, and the intended throughput (frame-rate) of each subgraph. The relative cost of a schedule is the sum of the execution time of each model layer, multiplied by the desired throughput. Consider for example a schedule with two applications, both executing at F FPS. Assume they share a compute stem

[1] Here, we assume that some admission control policy (outside the scope of this work) has been applied to ensure that some schedule is feasible for the set of running applications.

A, and then branch to specialized subgraphs B_1 and B_2. If C_A represents the execution cost (in CPU-seconds per frame, say) of A, and C_B the execution cost of B_1 and B_2, then the total cost of the schedule is $F \cdot C_A + 2F \cdot C_B$. Adding a third application based on the same network, using the same branchpoint and frame rate will add another factor of $F \cdot C_B$ to the schedule cost.

To most accurately model the compute costs (e.g., C_A and C_B), a forward pass execution of the base DNN should be executed and measured once on the target hardware. Note that as cost is relatively insensitive to the assigned model weights, each base DNN need only run one time (ever) per target hardware, not once per trained application.

Max-Min Fairness Among Applications. Although stem-sharing improves overall system efficiency, maximizing a global objective may lead to an inequitable allocation of resources for individual applications. Thus, M-Scheduler can also be run in max-min fairness mode, which maximizes the minimum benefit among applications, instead of the average. Max-min fairness is scheduled by searching the space using dynamic programming.

X-Voting To Improve Precision. Mainstream uses *voting* across frames to improve precision, and consequentially F1-score. With X-voting, Mainstream requires X consecutive positives to identify an event. While X-voting decreases false positives, it is not guaranteed to increase precision. For X-voting to improve precision, it must decrease false positives at a higher rate than true positives. The ideal X-voting configuration again depends on the applications and the resources available. A higher X incurs fewer false positives, but requires more cost to sustain high recall (either by increasing FPS or increasing specialization). We evaluate the effect of various X-voting configurations in Sect. 6.7.

6.6 Experimental Setup

To evaluate our system, we implement seven different event detection applications. We refer to the set of applications as 7-HYBRID. These are listed in Table 6.2, along with the datasets we used to train and test them. A pedestrian-detection application (PEDESTRIAN) is trained based on the fully-labeled, publicly-available Urban Tracking video dataset [197]. Our application to classify car models (CAR) uses the Stanford Cars image dataset [219]. Train detection (TRAIN) is based on video of nearby train tracks that we have captured in our camera deployment, and have hand labeled. The remaining classifiers are trained on a video of a nearby intersection, also captured in our camera deployment. We have obtained the necessary permissions and plan to make our Trains and Intersection video dataset available publicly. We reserve a portion of these datasets to create synthetic video workloads for testing.

We use M-Trainer to produce a task-specific M-Package for each application. Model candidates are fine-tuned using the MobileNets-224 model pretrained on ImageNet as a base DNN (implemented in Keras [63] using TensorFlow [**?**]). Each M-Package contains several models with different degrees of fine-tuning as described in Section 6.4. We evaluate Mainstream using the M-Packages and hold-out validation sets from our datasets. Our experiments use the applications in Table 6.2.

Hardware. Training is performed on nodes equipped with Intel® Xeon® E5-2698Bv3 processors

Detection Task	Dataset Description	Number of Images	Avg Event Length	Min Event Length	Event Frequency
PEDESTRIANS	Urban Tracker atrium video	4538	59	2	0.63
BUS	Intersection near CMU video	4762	1039	141	0.27
RED CAR	Intersection near CMU video	9172	228	46	0.08
SCRAMBLE	Intersection near CMU video	1500	412	382	0.16
SCHOOLBUS	Intersection near CMU video	2600	854	92	0.03
TRAINS	Train tracks near CMU video	3066	132	20	0.01
CARS	Images of 23 car models	3042	—	—	—

Table 6.2: Labeled datasets used to train classifiers for event detection applications. Average and minimum event lengths are reported in number of frames. Event length and event frequency only apply to video datasets and not CARS.

(2.0 GHz, 16 cores) and an Nvidia Titan X GPU. All end-to-end experiments use an Intel® NUC with an Intel® Core™ i7-6770HQ processor and 32 GiB DRAM, which is intended to represent an edge processing device. The Train and Intersection videos were captured using an Allied Vision Manta G-1236 GigE Vision camera.

6.7 Evaluation

We evaluate our system in the context of independent DNN-based video processing applications sharing a fixed-resource edge computer. In our evaluation, we show that Mainstream's dynamic approach outperforms static solutions in all of our experimental settings, across various application workloads, computational budgets, and numbers of concurrent applications. Mainstream's X-voting is capable of improving F1-score but, like model specialization, must also be dynamically configured to the resources available. In addition to our benchmarked applications, we show an end-to-end Mainstream deployment of a train detection application used for environmental pollution monitoring.

6.7.1 Mainstream Improves App Quality

Our goal in event detection is to maximize per-*event* F1-score. We compare the F1-score achieved by Mainstream with two baselines: No-Sharing and Max-Sharing. *No-Sharing* is the default behavior for a multi-tenant environment and is the approach taken by systems like TensorFlow Serving [1] and Clipper [74]. No-Sharing maximizes classification accuracy at the cost of a reduced sampling rate and requires no coordination between tenants. *Max-Sharing* is the sharing approach used by MCDNN [159]. It uses partial-DNN sharing by fine-tuning the final layer of concurrent DNNs. In many cases, Max-Sharing provides better F1-score relative to No-Sharing when a non-trivial number of applications share the infrastructure; it sacrifices classification accuracy to maximize the number of frames processed. We show, however, that Max-Sharing is less effective than making deliberate runtime decisions about how much sharing to use.

In order to observe the effects of increasingly constrained resources without a large number

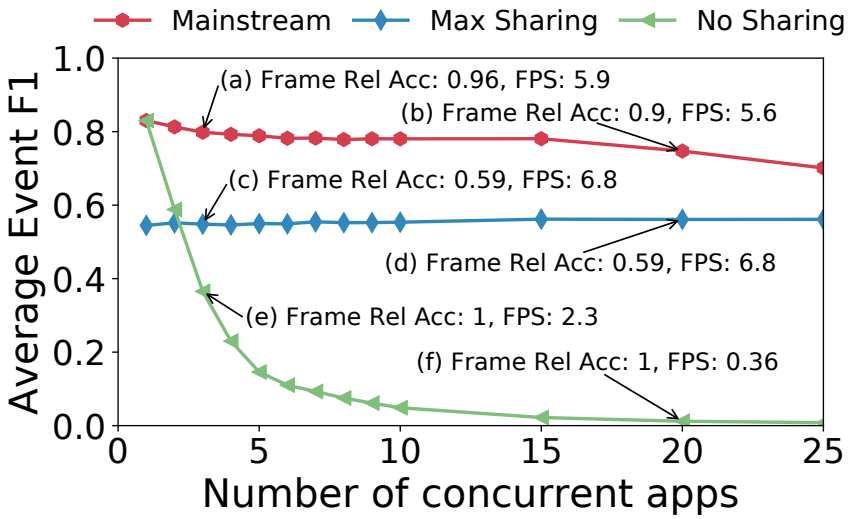


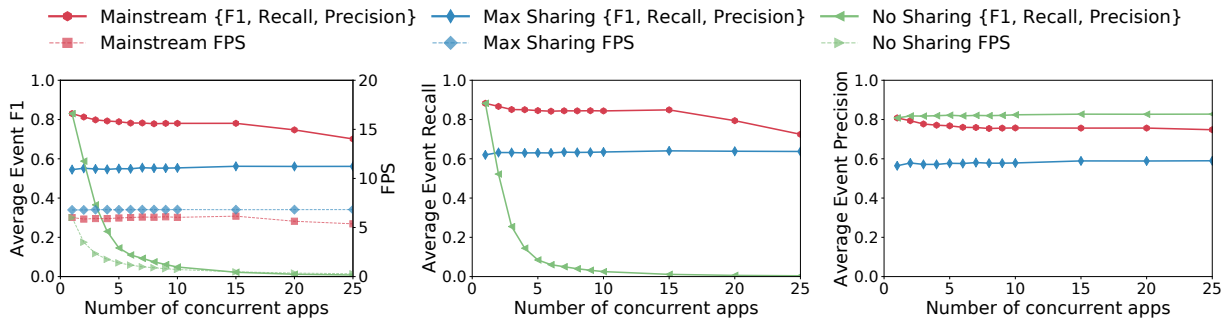
Figure 6.6: Mainstream improves F1-scores vs. No-Sharing between applications or conservatively sharing all layers but the last one. "Frame Rel Acc" is the relative image-level accuracy of the model deployed, compared to the best performing model candidate. "FPS" is the average observed throughput of the deployed applications.

of distinct applications, we generate additional applications by augmenting our application set. Each of the seven classification tasks in Table 6.2 has a corresponding "accuracy-tradeoff curve", which represents the relationship between per-frame accuracy and the shared stem size (Fig. 6.4b). For each application in our experiments, we randomly choose one of the seven classifiers (and its corresponding accuracy-tradeoff curve) and parameterize it with a different event length, event frequency and inter-frame correlation. To capture the effects of diverse application characteristics, the parameters are uniformly sampled from a range of possible values. Each workload consists of up to 30 concurrent applications. In most experiments, we show the behavior averaged across 100 workloads. Our video capture rate for all experiments is 10 FPS.

Mainstream outperforms static approaches. M-Scheduler maximizes per-event F1-score by varying the sampling rate and amount of sharing. Each additional application introduces more resource contention, forcing the system to pick a different balance between accuracy and sampling rate to achieve the best average F1-score.

Fig. 6.6 compares Mainstream with our baseline strategies. Mainstream delivers as much as a 87X higher per-event F1-score than No-Sharing and as much as a 47% higher score vs Max-Sharing. Fig. 6.6 reports F1-scores averaged across 100 workloads. The relationship between the three schedulers holds when examining individual workloads. No-Sharing exhibits low recall because of its low throughput—the system has fewer opportunities to detect the event. Max-Sharing has high throughput but a worse precision because the underlying model accuracy is lower—it evaluates many frames but does so inaccurately. Mainstream outperforms by striking a balance, sometimes choosing a more accurate model, and sometimes choosing to run at a higher throughput.

Mainstream dynamically balances precision and recall. Fig. 6.7 delves into the system effects of Mainstream more deeply. F1-score, recall, and precision are plotted. The average application frame rate is plotted, showing how Mainstream dynamically tailors resource usage to the workload.



(a) F1-score (b) Recall (c) Precision

Figure 6.7: The average event F1-score (Fig. 6.7a), recall (Fig. 6.7b) and precision (Fig. 6.7c) across 100 deployed workloads are shown (solid lines) alongside the average frame-rate across applications (dotted lines). Mainstream dynamically balances recall and precision to maximize aggregate F1-score. With high numbers of concurrent applications, Mainstream sacrifices small amounts of both specialization and frame-rate.

(Not shown is the varying model accuracy.) Optimizing for precision requires careful tuning of the application frame rate. While higher FPS always leads to higher recall, it does not always lead to higher precision. (A high frame rate may only increase false positives without increasing expected true positives.) For instance, No-Sharing's low frame rate and high per-frame accuracy allows it to have the highest precision of the three approaches. When given just a few applications, Mainstream runs specialized models, while throttling the stream rate to avoid unnecessary false positives. As resources become scarce, many applications begin to share more of the network.

Mainstream improves upon Max-Sharing even under tight resource constraints. Fig. 6.8 shows the effect of Mainstream, Max-Sharing, and No-Sharing on a range of computational budgets. We average the event F1-scores across 100 workloads, each with 3 applications. The right-most points represent the scenario of running on computational resources equivalent to an Intel® NUC. With a small workload of three applications, No-Sharing performs better than Max-Sharing, as it is able to run expensive models at a high enough frame rate. As we decrease the available budget, Max-Sharing's conservative sharing approach allows it to be more scalable than No-Sharing. However, even after the computational budget is reduced by 83%, Mainstream still improves application performance, compared to the overly conservative Max-Sharing approach.

6.7.2 Tuned X-Voting improves F1-score

Applications that suffer from low per-frame precision will generate many false positives. An X-voting approach can greatly decrease the incidence of false positives, as X consecutive classifications are needed in order to report a detection. Too large a value of X can hurt recall, causing real events to go unreported. By using X-voting and optimizing the parameter X, Mainstream can improve the overall average event F1-score.

Fig. 6.9 shows the effects of X-voting on F1-scores as X and the number of applications are varied, while total resources are kept fixed. With just a few concurrent applications, running at high frame rates, 7-voting and 5-voting yield the highest F1-scores. With more resource contention,

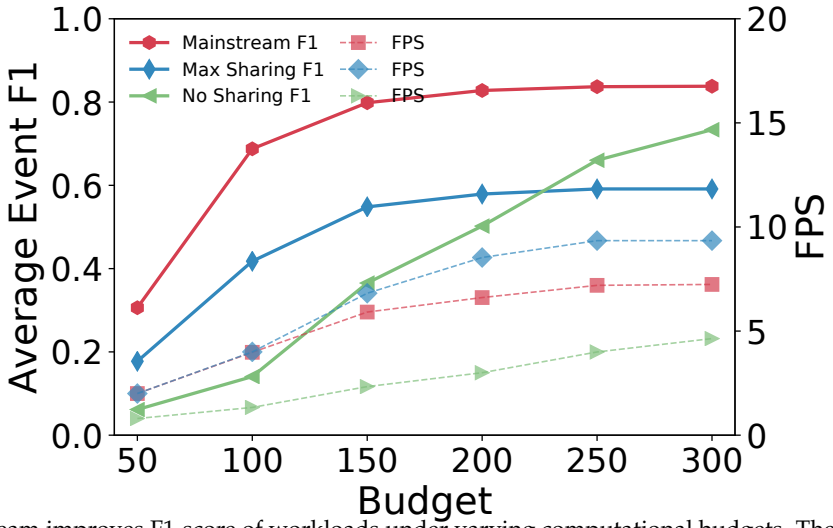


Figure 6.8: Mainstream improves F1-score of workloads under varying computational budgets. The rightmost points on the X axis represent the resources available on an Intel® NUC. Even under heavy resource constraints, there is available capacity for Mainstream to perform optimizations.

and lower throughput, 3-voting becomes the best choice as the cost of dropping true positives outweighs the benefit of reducing false positives for higher values of X. When resources become too constrained, this approach is less viable, e.g., 1-voting becomes the best approach at 25 concurrent apps. Fig. 6.9 also shows the Pareto frontier of F1-scores achievable across all values of X for a given number of concurrent applications.

6.7.3 Mainstream Deployment

We deployed our environmental pollution monitor application and nine other concurrent applications using both Mainstream and a conventional No-Sharing approach for one week on the hardware setup described in Section 6.6. Fig. 6.10 shows the trace of both approaches on a *single* train event sequence, indicating the frames analyzed. A hit represents a correct classification of the train, a miss represents an incorrect classification. We see that Mainstream's deployment samples the stream more frequently, yielding many more hits (and misses) than No-Sharing; the result, though, is that Mainstream detects the train event earlier and more confidently.

We control the false positive rate with 2-voting, requiring Mainstream to have two positive samples before an event is classified. The false positive rate of the TRAIN video drops from 0.028 to 0.00056. No-Sharing and Mainstream achieve a 0 and a 0.00056 false positive rate, respectively. In the analyzed deployment in Fig 6.10, we see that Mainstream still detects the train easily and quickly.

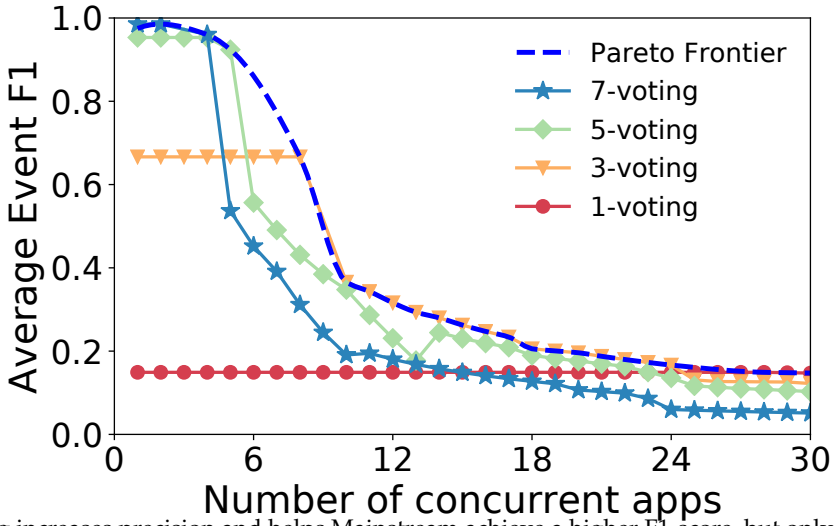


Figure 6.9: X-voting increases precision and helps Mainstream achieve a higher F1-score, but only if frame rate is high enough to avoid hurting recall. Thus, the effects vary by the level of resource contention. The Pareto frontier shows the F1-scores achievable given the dynamic selection of an optimal X-voting scheme for the resource scenario.

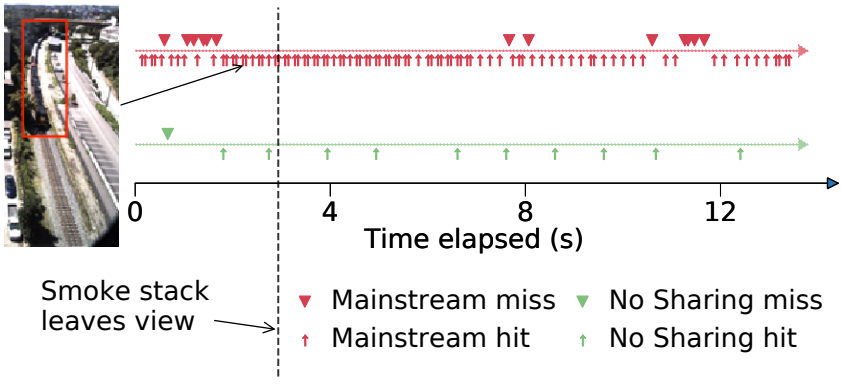


Figure 6.10: Timeline of Mainstream running a train detector app with 9 concurrent applications. Our goal is to detect the train as early as possible, before the smoke stack is out of view (end of window represented by the dotted line). Mainstream detects the train earlier and more confidently than No-Sharing.

6.8 Additional Background

Several recent systems have attempted to tackle the problem of optimizing execution of visual computing pipelines.

VideoStorm [462] is a video analytics system for large-scale clusters and workloads. It analyzes resource use and application-goal-based metrics as a function of tunable parameters of the analytics pipelines, building models for each application independently. It uses these models to allocate resources and select parameters for deployed applications on a target platform, in order to maximize application quality metrics. VideoStorm takes a black-box view of the applications, and assumes that quality and resource consumption of co-deployed applications are independently determined. Therefore, it cannot take into account computation sharing, or optimize the sharing vs. degree of specialization tradeoff. In contrast, Mainstream takes a white-box approach to modeling application quality, and can explicitly tune computation sharing to improve application quality metrics. Compared to VideoStorm, Mainstream sacrifices some generality to solve the joint optimization problem.

MCDNN [159] introduces a static approach to sharing DNN computation, in which each application developer independently determines their amount of model specialization. MCDNN opportunistically shares any identical unspecialized layers between applications. In contrast, Mainstream's training and scheduling components allow late binding and jointly-optimized selection of the degrees of specialization at run time, when resource availability and co-deployed tasks are known.

Inference serving systems. Mainstream is an inference serving system for running neural networks on resource-constrained nodes. Other inference serving systems include Clipper [74], NoScope [208], and TensorFlow Serving [1]. Like Mainstream, these systems optimize for latency and throughput gains. Clipper caches results from multiple models, dynamically chooses from the results, and optimizes the batch size. NoScope replaces expensive neural networks for object detection with cheaper difference detectors and specialized models. TensorFlow Serving increases throughput with batching and hardware acceleration. LASER [7] and Velox [73] are inference serving systems for non-DNN models. LASER deploys linear models while Velox deploys personalized prediction algorithms using Apache Spark.

Unlike Mainstream, these inference serving systems do not share computation between independently trained models. They also target cluster environments. Mainstream targets edge devices with limited resources, where achieving the right degree of DNN computation sharing is particularly important, though such sharing would also be valuable in large data centers.

Reducing DNN inference time. Approaches to reducing DNN inference time for vision applications can be broadly classified into those that reduce model precision [55, 71, 72, 145, 184, 330, 473], use efficient network architectures [178, 185], use anytime prediction methods [180, 182], or employ model compression and sparsification [160, 255, 438]. All of these methods are orthogonal to Mainstream's adaptive DNN computation sharing technique, but share its goal of selecting the right trade-off between per-frame quality and frame throughput.

Multi-task networks. Multi-task learning [9, 48, 217, 263, 267, 287, 329, 466] is a ML approach

in which a single model is trained to perform multiple tasks. Using multiple tasks to train a single model helps achieve better accuracy because of better generalization and complementary information [48, 349]. In the context of DNNs, a multi-task network can have a varying number of shared layers across tasks and task-specific layers [267, 287]. Multi-task learning assumes that all of the tasks are known *a priori*, and that training data for all of the tasks is available for use in a single training process. In contrast, Mainstream allows each task to be developed, trained, and deployed independently, and avoids the need to share or expose proprietary or privacy-sensitive training data between task developers. Note that one can run a multi-task network as a single large application in Mainstream.

6.9 Conclusion

In this chapter, we presented Mainstream which is an efficient way to scale to multiple tasks at test time. Mainstream adaptively orchestrates DNN stem-sharing among concurrent video processing applications sharing the limited resources on an edge device, resulting in much higher aggregate application quality. Experiments with several event detection tasks confirm that Mainstream significantly increases performance to current approaches over a range of concurrency levels.

Part III

Structure in Classifiers

Chapter 7

Surprisingly Easy Synthesis for Instance Detection

Immature poets imitate; mature poets steal.

T. S. ELIOT in *The Sacred Wood: Essays on Poetry and Criticism*, 1920

In this section we look at how structure in the model space can help us learn with limited supervision. Imagine using an object detection system for an environment like your kitchen. Such a system needs to not only recognize different kinds of objects but also distinguish between many different *instances* of the same object category, *e.g.*, *your* cup vs. *my* cup. With the tremendous progress that has been made in visual recognition, as documented on benchmark detection datasets, one may expect to easily take a state-of-the-art system and deploy it for such a setting. However, one of the biggest drawbacks of using such a system is that it is trained only only a few typical categories such as `man`, `cow`, `sheep` *etc*. In a personalized setting we need annotations for *instances* like *your* cup, and typically need lots of such data to train an object detector. We believe that collecting such annotations is a major impediment for rapid deployment of detection systems in robotics or other personalized applications.

In this chapter, we specifically look at instance (object) detection methods and exploit a key property of the detection models to easily synthesize training images. Region-based detectors care more about *local* consistency than the full scene itself. This observation holds even more strongly for texture rich objects and instances. In our work, we devise a simple scheme for image synthesis - we cut object masks from simple images and paste them on background images. This simple approach can help us synthesize large amounts of 'new' training data. In contrast, existing image synthesis approaches try to solve the significantly more difficult task of synthesizing *every pixel* of the full scene. This means they need to not only model *local* consistency but also *global* consistency such as lighting, support relationships *etc*. Solving this full scene synthesis is challenging and these methods have limited applications. As T. S. Eliot stated, when solving or creating masterpieces (full scenes) is hard, it is better to steal (cut-paste) and create something.

Recently, a successful research direction to overcome this annotation barrier, is to use synthetically rendered scenes and objects [213, 299, 391] to train a detection system. This approach requires a lot of effort to make the scenes and objects realistic, ensuring high quality *global and local consistency*. Moreover, models trained on such synthetic data have trouble generalizing to

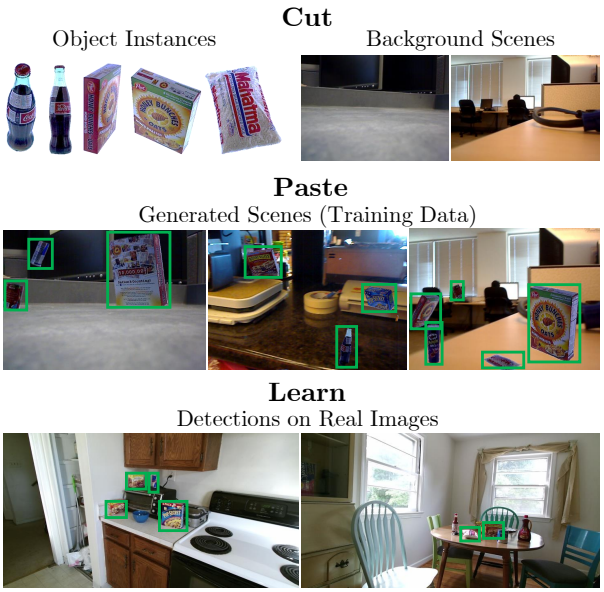


Figure 7.1: We present a simple way to rapidly generate training cut-paste/images for instance detection with minimal human effort. We automatically extract object instance masks and render it on random background cut-paste/images to create realistic training cut-paste/images with bounding box labels. Our results show that such data is competitive with human curated datasets, and contains complementary information.

real data because of the change in image statistics [56, 320]. To address this, an emerging theme of work [154] moves away from graphics based renderings to composing real cut-paste/images. The underlying theme is to 'paste' real object masks in real cut-paste/images, thus reducing the dependence on graphics renderings. Concurrent work [137] estimates scene geometry and layout and then synthetically places object masks in the scene to create realistic training cut-paste/images. However, the scene layout estimation step may not generalize to unseen scenes. In this chapter, we show a simpler approach that does not require such scene geometry estimation to create training cut-paste/images.

Our key insight is that state-of-the art detection methods like Faster-RCNN [341] and even older approaches like DPM [110] *etc*. care more about *local* region-based features for detection than the *global* scene layout. As an example, a cup detector mostly cares about the visual appearance of the cup and its blending with the background, and not so much about where the cup occurs in the scene: the table-top or the ground. We believe that while global consistency is important, only ensuring *patch-level realism* while composing synthetic datasets should go a long way to train these detectors. We use the term *patch-level realism* to refer to the observation that the bounding box containing the pasted object *looks* realistic to the human eye.

However, naively placing object masks in scenes creates subtle pixel artifacts in the cut-paste/images. As these minor imperfections in the pixel space feed forward deeper into the layers of a ConvNet [239], they lead to noticeably different features and the training algorithm focuses on these discrepancies to detect objects, often ignoring to model their complex visual appearance. As our results show (Table 7.1), such models give reduced detection performance.

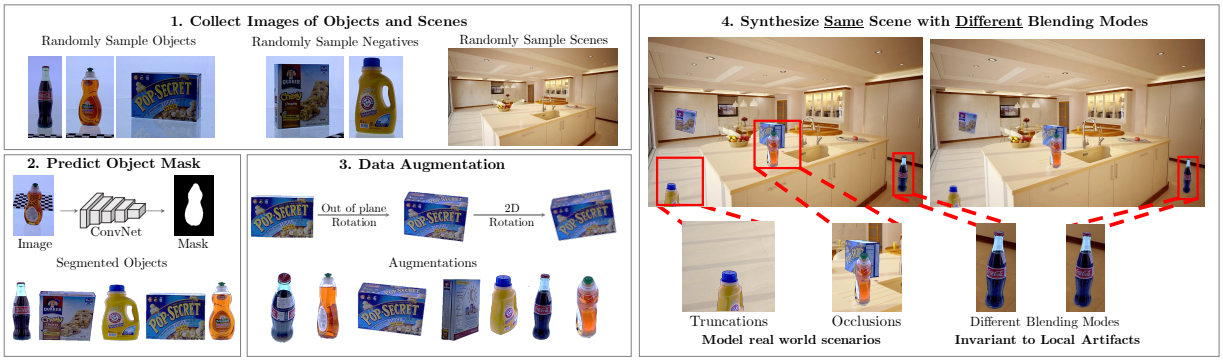


Figure 7.2: We present a simple approach to rapidly synthesize datasets for instance detection. We start with a set of cut-paste/images of the instances and background scenes. We then automatically extract the object mask and segment the object. We paste the objects on the scenes with different blending to ensure that local artifacts are ignored by the detection model. Our results show that this synthesized data is both competitive with real data and contains complementary information.

Since our main goal is to create training data that is useful for training detectors, we resolve these local imperfections and maintain patch level realism. Inspired from methods in data augmentation and denoising auto encoders [425], we generate data that forces the training algorithm to ignore these artifacts and focus only on the object appearance. We show how rendering the same scene with the same object placement and only varying the blending parameter settings (Section 7.4.2) makes the detector robust to these subtle pixel artifacts and improves training. Although these cut-paste/images do not respect global consistency or even obey scene factors such as lighting *etc.*, training on them leads to high performance detectors with little effort. Our method is also complementary to existing work [137, 299, 391] that ensures global consistency and can be combined with them.

Data generated using our approach is surprisingly effective at training detection models. Our results suggest that curated instance recognition datasets suffer from poor coverage of the visual appearances of the objects. With our method, we are able to generate many such cut-paste/images with different viewpoints/scales, and get a good coverage of the visual appearance of the object with minimal effort. Thus, our performance gain is particularly noticeable when the test scenes are different from the training scenes, and thus the objects occur in different viewpoints/scales.

7.1 Background

Instance detection is a well studied problem in computer vision. [468] provides a comprehensive overview of the popular methods in this field. Early approaches, such as [66], heavily depend on extracting local features such as SIFT [265], SURF [31], MSER [278] and matching them to retrieve instances [264, 410]. These approaches do not work well for objects which are not 'feature-rich', where shape-based methods [114, 171, 179] are more successful.

Modern detection methods [140, 141, 341] based on learned ConvNet features [222, 239, 378] generalize across feature rich and feature poor objects [366]. With the availability of powerful commodity GPUs, and fast detection algorithms [259, 335], these methods are suitable for real-time object detection required in robotics. More recently, deep learning based approaches in computer

vision are being adopted for the task of pose estimation of specific objects[292, 442, 460]. Improving instance detection and pose estimation in warehouses will be signifcantly useful for the perception pipeline in systems trying to solve the Amazon Picking Challenge[70].

The use of these powerful methods for object and instance detection requires large amounts of annotated data. This requirement is both impractical and expensive for rapidly deploying detection systems. Sythesizing data is one way to address this issue. [320, 391] use rendered cut-paste/images of objects to do both object detection and pose estimation. They render 3D models of objects from different viewpoints and place them against randomly sampled backgrounds. [299] also highlight the importance of using photo-realsitic models in training CNNs.

There is a wide spectrum of work where rendered datasets are used for computer vision tasks. At one end, we have datasets with cut-paste/images of single objects on random backgrounds [299, 316, 320, 391]. On the other end, there are datasets where the entire scene is rendered [130, 161, 343]. On that spectrum our work lies in between as we do not render the whole world but use real cut-paste/images of both objects and backgrounds to compose new scenes. In this sense, our work closely related to contemporary work from [154] which generates synthetic data for localizing text in scenes.

Sedaghat *et al*. [360] show how an annotated dataset can be created for the task of object pose estimation by taking videos by walking around the object. [169] uses synthetic data from [49] for multi-view instance recognition. [274] use real and synthetic cut-paste/images for 2D-3D alignment.

Similarly, [56, 423] render 3D humans in scenes and use this data for pose estimation. Tasks requiring dense annotation, such as segmentation, tracking *etc*. have also shown to benefit by using such approaches [130, 161, 343, 346]. [383] shows a novel approach for collecting data of objects in a closed domain setting. [14, 136, 229] annotate 3D points belonging to an object in the point cloud reconstruction of a scene and propagate the label to all frames where the object is visible. As synthetic data can be significantly different from real cut-paste/images, [416] shows a domain adaptation approach to overcome this issue. In contrast, our work composes training scenes using real object cut-paste/images as well as real background cut-paste/images.

The existing approaches to sythesizing datasets focus largely on ensuring global consistency and realism [56, 137, 213, 423]. While global consistency is important, we believe that local features matter more for training detection systems. Our approach ensures that when we train our detection model it is invariant to local discrepancies.

7.2 Background

Instance Detection: Instance detection requires accurate localization of a particular object, *e.g*. a particular brand of cereal, a particular cup *etc*. In contrast, generic object detection detects an entire generic category like a cereal box or a cup (see Figure 7.3). In fact, in the instance detection scenario correctly localizing a cereal box of some other brand is counted as a mistake. Instance detection occurs commonly in robotics, AR/VR *etc*., and can also be viewed as fine-grained recognition.

Traditional Dataset Collection: Building detection datasets involves a data curation step and

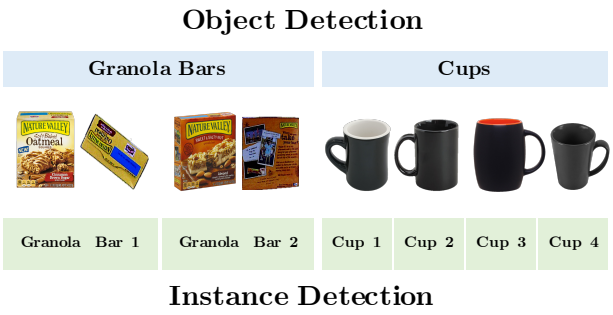


Figure 7.3: Object vs Instance Detection. Instance detection involves fine-grained recognition within the same 'object category'(as shown by the visually similar cups) while also detecting the same instance from different viewpoints(depicted by the different views of the granola bars). In this example, instance recognition must distinguish amongst 6 classes: 2 types of granola bars and 4 types of coffee cups. Object detection would distinguish only amongst 2 classes: coffee cups and granola bars.

an annotation step. Typically, data curation involves collecting internet cut-paste/images for object detection datasets [94, 251]. However, this fails for instance datasets as finding internet cut-paste/images of particular instances is not easy. For instance detection [380] data curation involves placing the instances in varied backgrounds and manually collecting the cut-paste/images. Manually collecting these cut-paste/images requires one to pay attention to ensure diversity in cut-paste/images by placing the object in different backgrounds and collecting different viewpoints. The annotation step is generally crowd sourced. Depending on the type of data, human annotations can be augmented with object tracking or 3D sensor information [14, 136, 229, 383, 429].

Unfortunately, both these steps are not suitable for *rapidly* gathering instance annotations. Firstly, as we show in our experiments, even if we limit ourselves to the same *type* of scene, *e.g.*, kitchens, the curation step can lack diversity and create biases that do not hold in the test setting. Secondly, as the number of cut-paste/images and instances increase, manual annotation requires additional time and expense.

7.3 Approach Overview

We propose a simple approach to rapidly collect data for instance detection. Our results show that our approach is competitive with the manual curation process, while requiring little time and no human annotation.

Ideally, we want to capture all of the visual diversity of an instance. Figures 7.1 and 7.3 show how a single instance appears different when seen from different views, scales, orientation and lighting conditions. Thus, distinguishing between such instances requires the dataset to have good coverage of viewpoints and scales of the object. Also, as the number of classes increases rapidly with newer instances, the long-tail distribution of data affects instance recognition problems. With synthetic data, we can ensure that the data has good coverage of both instances and viewpoints. Figure 7.2 shows the main steps of our method:

1. **Collect object instance cut-paste/images:** Our approach is agnostic to the way the data is

Figure 7.4: A few randomly chosen samples from our synthesized cut-paste/images. We describe the details of our approach in Section 7.4.

Table 7.1: We analyze the effect of various factors in synthesizing data by generating data with different settings and training a detector [341]. We evaluate the trained model on the GMU Dataset [136]. As we describe in Section 7.4, these factors greatly improve the quality of the synthesized data.

	2D Rot.	3D Rot.	Trunc.	Occl.	coca cola	coffee mate	honey bunches	hunt's sauce	mahatma rice	nature v1	nature v2	palmolive orange	pop secret	pringles bbq	red bull	mAP
Blending (Sec 7.4.2)																
No blending	✓	✓	✓	✓	65.7	91.1	**83.2**	59.8	57.7	92.1	84.4	61.4	59.0	38.7	31.9	65.9
Gaussian Blurring	✓	✓	✓	✓	65.3	88.1	80.8	**67.5**	63.7	90.8	79.4	57.9	58.9	**65.7**	40.2	68.9
Poisson [322]	✓	✓	✓	✓	62.9	82.9	63.9	59.4	20.7	84.6	67.9	60.9	73.5	41.0	25.1	58.4
All Blend	✓	✓	✓	✓	76.0	90.3	79.9	65.4	67.3	**93.4**	**86.6**	64.5	73.2	60.4	39.8	72.4
All Blend + same image	✓	✓	✓	✓	**78.4**	**92.7**	81.8	66.2	**69.8**	93.0	82.9	**65.7**	**76.0**	62.9	**41.2**	**73.7**
Data Aug. (Sec 7.4.2)																
No 2D Rotation		✓	✓	✓	63.3	90.4	81.4	63.7	54.2	91.8	82.3	59.2	71.3	68.2	41.4	69.7
No 3D Rotation	✓		✓	✓	73.1	90.6	83.2	63.3	55.8	93.4	82.1	65.9	64.5	45.7	33.6	68.3
No Trunc.	✓	✓		✓	73.4	92.1	77.8	59.9	64.6	92.4	84.6	62.0	74.2	67.4	41.7	71.8
No Occlusion	✓	✓	✓		63.1	84.9	74.4	64.5	50.8	76.9	67.6	55.7	69.0	58.7	28.1	63.1
All	✓	✓	✓	✓	78.4	92.7	81.8	**66.2**	69.8	93.0	82.9	65.7	76.0	62.9	41.2	73.7
All + Distractor	✓	✓	✓	✓	**81.0**	**93.3**	**85.6**	55.6	**73.8**	**94.9**	**87.1**	**68.7**	**79.5**	**77.1**	**42.0**	**76.2**

collected. We assume that we have access to object cut-paste/images which cover diverse viewpoints and have a modest background.

2. **Collect scene cut-paste/images:** These cut-paste/images will serve as background cut-paste/images in our training dataset. If the test scenes are known beforehand (like in the case of a smart-home or a warehouse) one can collect cut-paste/images from those scenes. As we do not compute any scene statistics like geometry or layout, our approach can readily deal with new scenes.
3. **Predict foreground mask for the object:** We predict a foreground mask which separates the instance pixels from the background pixels. This gives us the object mask which can be placed in the scenes.
4. **Paste object instances in scenes:** Paste the extracted objects on a randomly chosen background image. We ensure **invariance to local artifacts** while placing the objects so that the training algorithm does not focus on subpixel discrepancies at the boundaries. We add various modes of blending and synthesize the exact same scene with different blending to make the algorithm robust to these artifacts. We also add **data augmentation** to ensure a diverse viewpoint/scale coverage.

7.4 Approach Details and Analysis

We now present additional details of our approach and provide empirical analysis of our design choices.

7.4.1 Collecting cut-paste/images

We first describe how we collect object/background cut-paste/images, and extract object masks without human effort.

Images of objects from different viewpoints: We choose the objects present in Big Berkeley Instance Recognition Dataset (BigBIRD) [380] to conduct our experiments. Each object has 600 cut-paste/images, captured by five cameras with different viewpoints. Each image also has a corresponding depth image captured by an IR camera.

Background cut-paste/images of indoor scenes: We place the extracted objects from the BigBIRD cut-paste/images on randomly sampled background cut-paste/images from the UW Scenes dataset [229]. There are 1548 cut-paste/images in the backgrounds dataset.

Foreground/Background segmentation: Once we have collected cut-paste/images of the instances, we need to determine the pixels that belong to the instance vs. the background. We automate this by training a model for foreground/background classification. We train a FCN network [262] (based on VGG-16 [378] pre-trained on PASCAL VOC [94] image segmentation) to classify each image pixel into foreground/background. The object masks from the depth sensor are used as ground truth for training this model. We train this model using cut-paste/images of instances which are not present in our final detection evaluation. We use [30] as a post-processing step to clean these results and obtain an object mask. Figure 7.5 shows some of these results. In practice, we found this combination to generalize to cut-paste/images of unseen objects with modest backgrounds and give good quality object masks from input cut-paste/images. It also generalizes to transparent objects, *e.g.*, `coca cola` bottle, where the depth sensor does not work well.

7.4.2 Adding Objects to Images

After automatically extracting the object masks from input cut-paste/images, we paste them on real background cut-paste/images. Naïvely pasting objects on scenes results in artifacts which the training algorithm focuses on, ignoring the object's visual appearance. In this section, we present steps to generate data that forces the training algorithm to ignore these artifacts and focus only on the object appearance. To evaluate these steps empirically, we train a detection model on our synthesized cut-paste/images and evaluate it on a benchmark instance detection dataset (real cut-paste/images).

Detection Model: We use the Faster R-CNN [341] method and initialize the model from a VGG-16 [378] model pre-trained on object detection on the MSCOCO [251] dataset.

Benchmarking Dataset: After training the detection model on our synthetic cut-paste/images, we use the GMU Kitchen dataset [136] for evaluation. There are 9 scenes in this dataset. Three dataset splits with 6 scenes for training and 3 for testing have been provided in [136] to conduct experiments on the GMU Kitchen dataset. We follow these splits for train/test and report the average over them. No cut-paste/images or statistics from this dataset are used for either dataset synthesis or training the detector. We report Mean Average Precision (mAP) at IOU of 0.5 [94] in all our experiments.

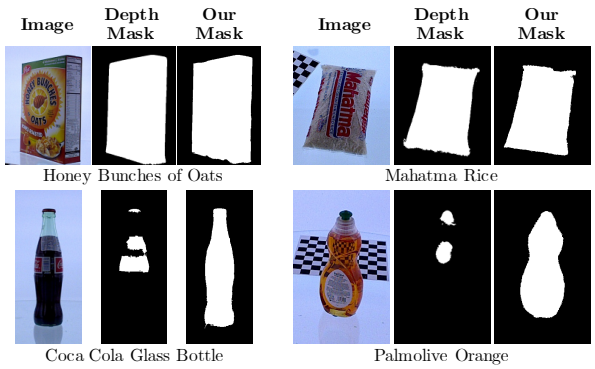


Figure 7.5: Given an image of a new unseen object instance, we use a ConvNet to predict foreground/background pixels. Using these predictions we automatically obtain an object mask. This method generalizes to transparent surfaces where traditional methods relying on depth sensors for segmentation fail (second row).

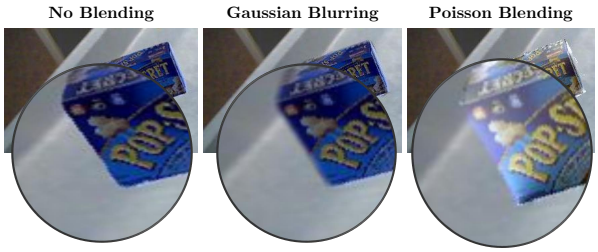


Figure 7.6: Different blending modes used while generating datasets. These modes help the model in ignoring artifacts arising from pasting objects on background scenes. More details in Section 7.4.2

Blending

Directly pasting objects on background cut-paste/images creates boundary artifacts. Figure 7.6 shows some examples of such artifacts. Although these artifacts seem subtle, when such cut-paste/images are used to train detection algorithms, they give poor performance as seen in Table 7.1. As current detection methods [341] strongly depend on local region-based features, boundary artifacts substantially degrade their performance.

The blending step 'smoothens' out the boundary artifacts between the pasted object and the background. Figure 7.6 shows some examples of blending. Each of these modes add different image variations, *e.g.*, Poisson blending [322] smooths edges and adds lighting variations. Although these blending methods do not yield visually 'perfect' results, they improve performance of the trained detectors. Table 7.1 lists these blending methods and shows the improvement in performance after training on blended cut-paste/images.

To make the training algorithm further ignore the effects of blending, we synthesize the exact same scene with the same object placement, and only vary the type of blending used. We denote this by 'All Blend + same image' in Table 7.1. Training on multiple such cut-paste/images where only the blending factor changes makes the training algorithm invariant to these blending factors and **improves performance by 8 AP points** over not using any form of blending.

Figure 7.7: Missed detections on the Active Vision Dataset [14] for a model trained on the hand-annotated GMU Dataset [136]. The model consistently fails to detect certain viewpoints as the training data has poor viewpoint coverage and has biases different from the test set. Each row shows a single instance.

Data Augmentation

While pasting the objects on background, we also add the following modes of data augmentation:

2D Rotation: The objects are rotated at uniformly sampled random angles in between 30 to -30 degrees to account for camera/object rotation changes. Table 7.1 shows a gain of 3 AP points by adding this augmentation.

3D Rotation: As we can control this step, we have many cut-paste/images containing atypical 3D rotations of the instances which is hard to find in real data. Table 7.1 shows a gain of more than 4 AP points because of this augmentation. In Section 7.5.2 and Figure 7.7, we show examples of how a model trained on human collected data consistently fails to detect instances from certain viewpoints because the training data has poor viewpoint coverage and different biases from the test set. This result shows the value of being able to synthesize data with diverse viewpoints.

Occlusion and Truncation: Occlusion and truncation naturally appear in cut-paste/images. They refer to partially visible objects (such as those in Figure 7.2). We place objects at the boundaries of the cut-paste/images to model truncation, ensuring at least 0.25 of the object box is in the image. To add occlusion, we paste the objects with partial overlap with each other (max IOU of 0.75). Like other modes of augmentation, we can easily vary the amount of truncation/occlusion. As Table 7.1 shows, adding truncation/occlusion improves the result by as much as 10 AP points.

Distractor Objects: We add distractor objects in the scenes. This models real-world scenarios with multiple distractor objects. We use additional objects from the BigBIRD dataset as distractors. Presence of synthetic distractors also encourages the learning algorithm to not only latch on to boundary artifacts when detecting objects but also improves performance by 3 AP points.

Figure 7.8: Examples of false positives from the UNC dataset by the detector trained on the hand-annotated bounding boxes from the GMU dataset. Object detectors trained on hand annotated scenes also need new negatives to be able to perform well in newer scenes.

Table 7.2: We compute the performance of training a model on synthetic data and compare it against training on real data. We evaluate on the test split of the GMU Kitchen Dataset [136].

Dataset	coca cola	coffee mate	honey bunches	hunt's sauce	mahatma rice	nature v1	nature v2	palmolive orange	pop secret	pringles bbq	red bull	mAP
Real Images from GMU	81.9	95.3	92.0	87.3	86.5	96.8	88.9	**80.5**	92.3	88.9	58.6	86.3
SP-BL-SS [137]	55.5	67.9	71.2	34.6	30.6	82.9	66.2	33.1	54.3	54.8	17.7	51.7
(Ours) Synthetic Images	81.0	93.3	85.6	55.6	73.8	94.9	87.1	68.7	79.5	77.1	42.0	76.2
SP-BL-SS + Real Images [137]	82.6	92.9	91.4	85.5	81.9	95.5	88.6	78.5	93.6	90.2	54.1	85.0
(Ours) Synthetic + Real Images	**88.5**	**95.5**	**94.1**	**88.1**	**90.3**	**97.2**	**91.8**	80.1	**94.0**	**92.2**	**65.4**	**88.8**

7.5 Experiments

We now compare the effectiveness of our synthesized data against human annotated data on two benchmark datasets. We first describe our common experimental setup.

Synthesized Data: We analyze our design choices in Section 7.4 to pick the best performing ones. We use a total of 33 object instances from the BigBIRD Dataset [380] overlapping with the 11 instances from GMU Kitchen Dataset [136] and the 33 instances from Active Vision Dataset [14]. We use a foreground/background ConvNet (Section 7.4.1) to extract the foreground masks from the cut-paste/images. The foreground/background ConvNet is *not* trained on instances we use to evaluate detection. As in Section 7.4, we use backgrounds from the UW Scenes Dataset [229] We generate a synthetic dataset with approximately 6000 cut-paste/images using all modes of data augmentation from Section 7.4. We sample scale, rotation, position and the background randomly. Each background appears roughly 4 times in the generated dataset with different objects. To model occlusions we allow a maximum IOU of 0.75 between objects. For truncations, we allow at least 25% of the object box to be in the image. For each scene we have three versions produced with different blending modes as described in Section 7.4.2. Figure 7.4 shows samples of generated cut-paste/images. We use this synthetic data for all our experiments. The code used for generating scenes is available at: `https://goo.gl/imXRt7`.

Table 7.3: Evaluation on the entire Active Vision dataset by varying the amount of real data from the GMU Kitchen Scenes *train* dataset

Dataset	coca cola	honey bunches	hunt's sauce	mahatma rice	nature v2	red bull	mAP
Real Images	57.7	34.4	48.0	39.9	24.6	46.6	41.9
Synthetic	63.0	29.3	34.2	20.5	**49.0**	23.0	36.5
Synthetic + Real Images	**69.9**	**44.2**	**51.0**	**41.8**	48.7	**50.9**	**51.1**
10% Real	15.3	19.1	31.6	11.2	6.1	11.7	15.8
10% Real + Syn	66.1	36.5	44.0	26.4	48.9	37.6	43.2
40% Real	55.8	31.6	47.3	27.4	24.8	41.9	38.2
40% Real + Syn	**69.8**	41.0	**55.7**	38.3	**52.8**	47.0	**50.8**
70% Real	55.3	30.6	47.9	36.4	25.0	41.2	39.4
70% Real + Syn	67.5	**42.0**	50.9	**43.0**	48.5	**51.8**	50.6

Model: We use a Faster R-CNN model [341] based on the VGG-16 [378] pre-trained weights on the MSCOCO [251] detection task. We initialize both the RPN trunk and the object classifier trunk of the network in this way. We fine-tune on different datasets (both real and synthetic) and evaluate the model's performance. We fine-tune all models for 25K iterations using SGD+momentum with a learning rate of 0.001, momentum 0.9, and reduce the learning rate by a factor of 10 after 15K iterations. We also use weight decay of 0.0005 and dropout of 0.5 on the fully-connected layers. We set the value of all the loss weights (both RPN and classification) as 1.0 in our experiments. We ensure that the model hyperparameters and random seed do not change across datasets/experiments for consistency.

Evaluation: We report Average Precision (AP) at IOU of 0.5 in all our experiments for the task of instance localization. Following [14], we consider boxes of size at least 50×30 pixels in the cut-paste/images for evaluation.

7.5.1 Training and Evaluation on the GMU Dataset

Similar to Section 7.4, we use the GMU Kitchen Dataset[136] which contains 9 kitchen scenes with $6,728$ cut-paste/images. We evaluate on the 11 objects present in the dataset overlapping with the BigBIRD [380] objects. We additionally report results from [137]. Their method synthesizes cut-paste/images by accounting for global scene structure when placing objects in scenes, *e.g.*, ensure that cups lie on flat surfaces like table tops. In contrast, our method does not use take into account such global structure, but focuses on patch-level realism. We note that their method [137] uses a different background scenes dataset for their synthesis.

Table 7.2 shows the evaluation results. We see that training on the synthetic data is competitive with training on real cut-paste/images (rows 1 vs 3) and also outperforms the synthetic dataset from [137] (rows 2 vs 3). Combining synthetic data with the real data shows a further improvement for all synthetic image datasets (rows 4, 5). These results show that the data generated by our approach is not only competitive with both real data and existing synthetic data, but also provides complementary information. Figure 7.9 shows qualitative examples illustrating this point.

Figure 7.9: We show qualitative detection results and mark true positives in green, false positives in red and arrows to highlight regions. The top two rows are from the GMU Kitchen Scenes [136] and the bottom two rows from the Active Vision Dataset [14]. $(a), (b)$: Model trained on real data misses objects which are heavily occluded (a) or stops detecting objects as viewpoint changes from a to b. $(c), (d)$: Model trained on synthetic data detects occluded and truncated objects. (e): Combining synthetic data removes false positives due to training only on real data. $(g), (h)$: Combining real data removes false positives due to training only on synthetic data. $(f), (g)$: Viewpoint changes cause false negatives. (Best viewed electronically)

7.5.2 Evaluation on the Active Vision Dataset

To test generalization across datasets, we now present experiments where we train on either our synthetic data or the GMU Dataset [136], and evaluate on the Active Vision Dataset [14]. The Active Vision Dataset[14] has 9 scenes and 17,556 cut-paste/images. It has 33 objects in total and 6 objects in overlap with the GMU Kitchen Scenes. We use these 6 objects for our analysis. We do *not* use this dataset for training.

We train a model trained on *all* the cut-paste/images from the GMU Dataset (Section 7.5.1). This model serves as a baseline for our model trained on synthetic data. As Table 7.3 shows, by collecting just 10% cut-paste/images and adding our synthetically generated cut-paste/images, we are able to get more MAP than using the real cut-paste/images in the dataset without the synthetic cut-paste/images. This highlights how useful our approach of dataset generation is in scenarios where there is a dearth of labeled cut-paste/images. Also, the performance gap between these datasets is smaller than in Section 7.5.1.

Failure modes of real data: Upon inspecting the errors [176] made by the GMU model, we see that a common error mode of the detector is its **failure to recognize certain views** in the test-set (see Figure 7.7). These viewpoints were sparsely present in the human annotated training data. In contrast, our synthetic training data has a diverse viewpoint coverage. The model trained on the synthesized cut-paste/images drastically reduces these errors. Combining the synthesized cut-paste/images with the real cut-paste/images from GMU gives a further improvement of **10 AP points** suggesting that synthesized cut-paste/images do provide complementary information.

Varying Real Data: We investigate the effect of varying the number of real cut-paste/images combined with the synthesized data. We randomly sample different amounts of real cut-paste/images from the GMU Dataset and combine them with the synthetic data to train the detector. As a baseline we also train the model on varying fractions of the real data. Table 7.3 shows that by adding synthetic cut-paste/images to just 10% of the real cut-paste/images we get a boost of **10 AP points** over just using real cut-paste/images. This performance is also tantalizingly close to the performance of combining larger fractions of real data. This result reinforces the effectiveness and complementary nature of our approach. In the supplementary material, we present additional such results.

7.6 Discussion

We presented a simple technique to synthesize annotated training cut-paste/images for instance detection. Our key insights were to leverage randomization for blending objects into scenes and to ensure a diverse coverage of instance viewpoints and scales. We showed that patch-based realism is sufficient for training region-proposal based object detectors. Our method performs favorably to existing hand curated datasets and captures complementary information. In a realistic cross-domain setting we show that by combining just 10% of the available real annotations with our synthesized data, our model performs better than using all the real annotations. From a practical standpoint our technique affords the possibility of generating scenes with non-uniform distributions over object viewpoints and scales without additional data collection effort. We believe our work can be combined with existing approaches [137] that focus on global consistency for placing objects and [213] which model realism.

Chapter 8

Composing Visual Classifiers

It is astonishing what language can do. With a few syllables it can express an incalculable number of thoughts.

GOTTLOB FREGE in *Gedankengefüge*, 1923

In Chapter 7, we exploited the region-based structure of detectors for easy data synthesis. In this Chapter, we will try to model this structure in the classifiers and show how such modeling can help us compose to form unseen classifiers. Such a compositional property helps us create classifiers for visual concepts *without any* human supervision.

Imagine a blue elephant. Having never seen a single example of such a creature, humans have no difficulty imagining it, or even recognizing it. Starting from Plato's Theaetetus to the early nineteenth century work of Gottlob Frege [125], compositionality is often regarded as a hallmark of intelligence. The core idea is that a complex concept can be developed by combining multiple simple concepts. In fact, the same idea has been explored in the field of computer vision as well: in the form of attributes [103, 372] or graphical models for SVOs (subject-object-verb triplets) [354]. While the idea of building complex concepts from simpler ones seems intuitive, current state-of-the-art methods for recognition or retrieval follow a more data-driven approach, where complex concepts are learned using hundreds and thousands of labeled examples instead of being composed. Why is that?

Interestingly, even in philosophy, there is clear tension between the idea of compositionality and the principle of contextuality. The principle of contextuality states that one cannot create a model of a simple concept without the context. This has often been stated as one of the main arguments against attributes: a red classifier in red wine is remarkably different from a red classifier in red tomato or even a red car. Figure 8.1 shows more such examples.

This direct tension between compositionality and contextuality leads to the basic exploration of this chapter: do current vision algorithms have such a compositional nature? Can we some respect the principle of contextuality yet create compositional visual classifiers? One approach to capture context is to use the text itself to learn how the modifiers should behave. For example, a modifier like "red" should show similar visual modifications for related concepts like tomatoes and berries. Approaches such as [243] have tried to use text to capture this idea and compose visual classifiers.

Figure 8.1: Visual concepts like objects and attributes are compositional. This compositionality depends on the context and the particular instances being composed. A small elephant is much larger than a small snake! Our surprisingly simple method models both compositionality and contextuality in order to learn visual classifiers. The results of our approach show that it composes while respecting context.

But do we really need taxonomy and knowledge from language to capture contextuality?

In this chapter, we propose an approach that composes classifiers by directly reasoning about them in model space. Our intuition behind this is that the model space is smooth and captures visual similarity, *i.e.*, tomato classifiers are closer to berry classifiers than cars. Thus, modifiers should apply similarly to similar classifiers. One task we consider is composing attribute (adjective) and object (noun) visual classifiers to get classifiers for (attribute, object) pairs. As Figure 8.1 shows, the visual interpretation of attributes depends on the objects they are coupled with, *e.g.*, a small elephant is still much larger than a small snake. Our approach respects such contextuality because it is conditioned on *all* the visual concepts and models them together, rather than in isolation. We show that our compositional transform captures such relations between objects and attributes, and can create visual classifiers for them.

As our experiments show, our approach is able to generalize such compositionality and contextuality to *unseen combinations* of visual concepts (Section 8.3. Our approach naturally extends beyond composing two primitives. We show results on combining subject, object and verb classifiers to unseen combinations of subject-verb-object triples (Section 8.3.3). On all these tasks, our method shows generalization capability beyond existing methods. Section 8.4.5 also shows our method's generalization to unseen primitives. Finally, in Section 8.4, we analyze the various components of our method and its various properties.

8.1 Background

Our work is heavily influenced by the principle of compositionality. This principle has a long standing history in the fields of philosophy, theory of mind, neuroscience, language, mathematics, computer science, *etc*. As such a broad overview is beyond the scope of this thesis, we focus on compositionality in the case of visual recognition. In its most basic form, the principle states that new concepts can be constructed from primitive elements. This principle is relevant for statistical learning as it paves the way for models that train with low sample complexity. Compositional

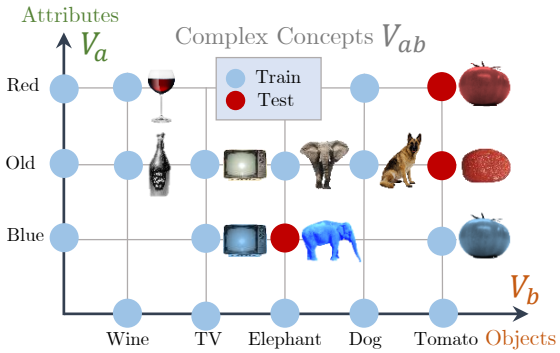


Figure 8.2: We assume that complex visual concepts can be composed using primitive visual concepts. By observing a few such complex concepts and their constituent visual primitives, we aim to learn a compositionality transform that can compose unseen combinations of primitives.

models can learn primitives from large amounts of samples and then compose these primitives to learn new concepts with limited samples [119, 150, 457, 477].

One of the earliest examples of using compositonality for visual recognition is Biederman's Recognition-By-Components theory [39] and Hoffman's part theory [175]. Compositionality is an underlying principle for many modern visual recognition systems [232]. Convolutional Neural Networks [239] have been shown to capture feature representations at multiple semantic and part hierarchies [459]. The parts-based systems such as Deformable Part Models [109], grammars [158, 294, 414, 448, 476, 478], and AND-OR graphs [372, 444, 475] also rely on compositionality of objects to build recognition systems. Compositionality has also been a key building block for systems used for visual question answering [18, 19], image captioning [22], handwritten digits [230, 231], zero-shot detection [220], segmentation and pose estimation [414, 448].

In this chapter, we focus on compositionality to compose unseen combinations of primitive visual concepts. This has been classically studied under the zero-shot Learning paradigm [233]. Zero-shot learning tries to generalize to new visual concepts without seeing any training examples. Generally, these methods [10, 12, 80, 233, 276, 344, 467] rely on an underlying embedding space, such as attributes, in order to recognize unseen categories. It is assumed that the description of the unseen categories is explicitly known in the underlying embedding space. As such explicit knowledge is not always available, another line of work [92, 127, 243, 266, 272, 337] relies on the finding such similarity in the linguistic space. Specifically, they leverage distributional word representations to capture some notion of taxonomy and similarity. However, in this work, we do not make assumptions about the availability of such a common underlying embedding or an external corpus of knowledge. Our aim is to explore compositionality purely in the visual domain.

Another area related to our work is that of transfer learning [47, 62, 80, 311, 318, 402], feature embeddings [211, 353] and low-shot recognition [108, 112, 117, 409]. These methods generalize to new categories by utilizing knowledge gained from familiar categories. Like our method, they rely on the visual similarity of unseen classes in order to generalize existing classifiers or features. However, unlike our method, these approaches need training examples of the 'unseen' classes. We build upon the insight from [437] that meaningful transformations can be learned directly in the model space without external sources of knowledge.

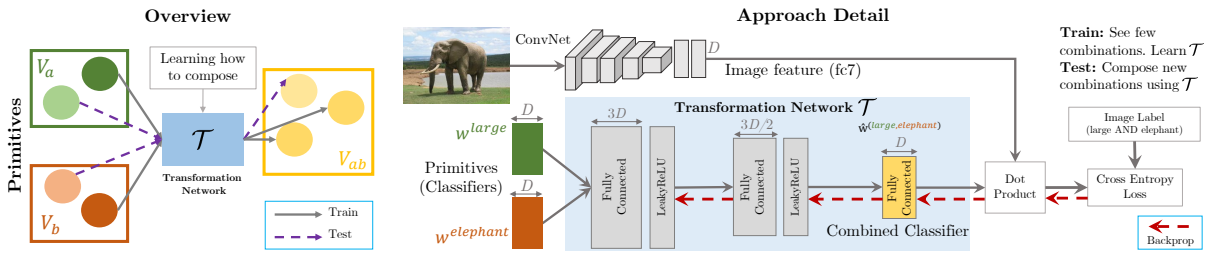


Figure 8.3: Our method composes classifiers from different types of primitive visual concepts. At training time, we assume access to a limited set of combinations, *e.g.*, `(large, elephant)` of these primitives. We model each of these primitives by learning linear classifiers (w) for them. We then learn a transformation network that takes these classifiers as input and composes them to produce a classifier for their combination. At test time, we show that this transformation can generalize to unseen combinations of primitives (Sections 8.3.2 and 8.3.3) and even unseen primitives (Section 8.4.5).

We study compositionality in visual recognition in the context of two well known problems - objects and attributes [103, 187, 194, 315, 319], and subject-verb-object (SVO) phrases [152, 266, 354, 407]. Both these problems capture compositionality of primitive visual concepts. Contextuality is an important aspect of composing primitives in both these problems and leads to varied visual appearance: *e.g.*, small elephant vs. small snake or person sitting on chair vs. person sitting on sofa. As has been noted in [266, 354], annotations for composite or complex visual concepts are far fewer in number than for primitive concepts. Thus, our work has important practical applications as it can compose visual primitives to recognize unseen complex concepts.

8.2 Approach

Our goal is to compose a set of visual concept primitives to get a complex visual concept as output. As a simple example, shown in Figure 8.2, consider the complex concepts spanned by combinations of attributes and objects. Given existing classifiers for an attribute `large` and an object `elephant`, we want to learn to compose them to get a classifier for the (attribute, object) pair of `large elephant`.

We represent our visual primitives by classifiers trained to recognize them. We then learn a transformation of these input visual classifiers to get a classifier that represents the complex visual concept. As shown in Figure 8.3, we parametrize our transformation by a deep network which accepts linear classifiers of the primitives as inputs and produces a linear classifier of the complex visual concept as output. As multi-layered networks can capture complex non-linear functions, we hope that such a network can learn to compose visual primitives while capturing contextuality. We show that such a network can generalize to unseen combinations of visual primitives and compose them.

8.2.1 Intuition

Evidence for visual compositionality exists in neuroscience and has been widely studied [4, 120]. Our intuition behind composing directly in the classifier space is that classifiers themselves represent visual similarity, *e.g.*, a classifier of an elephant is closer to an animal classifier rather than a

plate classifier. Thus, classifiers of 'unseen' combinations of classes can be composed by looking at 'seen' combinations using this visual similarity in classifier space.

8.2.2 Approach Details

We now describe our approach on how to compose complex visual classifiers from two or more simple visual classifiers. Without loss of generality, we will explain the details of our approach for the case of combining two classifiers but our approach can generalize to combine more types of primitives as demonstrated in our experiments.

Let us assume that we want to combine two different types of primitives. We represent these sets of primitives by $(\mathcal{V}_a, \mathcal{V}_b)$. These primitives are composed to form a complex primitive represented as $\mathcal{V}_{ab}$. As an example, consider $\mathcal{V}_a$ as the set of attributes and $\mathcal{V}_b$ as objects and thus $\mathcal{V}_{ab}$ consists of complex concepts formed by attribute, object pairs. We use $a, b, (a, b)$ to denote elements in $\mathcal{V}_a, \mathcal{V}_b, \mathcal{V}_{ab}$ respectively. Continuing our analogy of attributes and objects, $a, b, (a, b)$ can represent `large`, `elephant`, and `(large, elephant)`. We assume our vocabulary consists of M primitives of first type ($\mathcal{V}_a$) and N primitives of second type ($\mathcal{V}_b$). We also assume we have training data for some K complex concepts which combine one of M and N primitives.

We first train a linear classifier (SVM) for every type of primitive. Therefore, the primitive is parametrized by the weight vector for the linear classifier. Using the training data, we obtain weight vectors of $M + N$ primitives. Let us represent the weight vector for primitives $a \in \mathcal{V}_a, b \in \mathcal{V}_b$ as w_a, w_b. We can also train SVMs w_{ab} for complex concepts $(a, b) \in \mathcal{V}_{ab}$ using the training data available. However, since the training data for (a, b) pairs is limited compared to training data for a and b individually, directly training individual w_{ab} classifiers is difficult (see Section 8.3 for experiments). Instead, we want to use w_a and w_b to directly learn about the complex concept (a, b) *without* looking at w_{ab}.

As Figure 8.3 shows, we want to learn a function $\mathcal{T}$ such that it transforms the weights of two primitives (w_a, w_b) and outputs the weight of the complex concept (a, b):

$$\hat{w}_{ab} = \mathcal{T}(w_a, w_b). \tag{8.1}$$

Our training data contains the pairs (w_a, w_b). However, at training time, we do not have all possible combinations of (a, b) but very few combinations ($K << MN$). In order to detect unseen combinations at test time, we want to use compositionality and learn to combine two different primitives a and b to get the combinations (a, b).

We use a multi-linear perceptron to parameterize the function $\mathcal{T}$ and describe the architecture and the loss function.

Architecture: The transformation network $\mathcal{T}$ is a feedforward network with three fully connected layers. We use the LeakyReLU [165] non-linearity in between the layers. Given n SVMs (for n primitive concepts) each of dimensionality D as input, the output sizes of the three layers are $(n + 1)D$, $(n + \frac{1}{2})D$ and D.

Loss Function: We compute the score between the output of the transformation $\mathcal{T}$ and the input image features $\phi(I)$.

$$p = \text{sigmoid}\left(\mathcal{T}(w_a, w_b)^\top \phi(I)\right)$$

This score reflects the compatibility between the model transformation and the image. We want this score to be high only if the image contains the complex concept (a, b) and low otherwise. As an example, we want the score to be high only for `(large, elephant)` and want it to be low for an image containing either `elephant` or `large` (not both), or neither of the two concepts.

We train the parameters of the transformation network $\mathcal{T}$ to minimize the binary cross entropy loss

$$L(I, w_a, w_b) = y \log(p) + (1 - y) \log(1 - p), \tag{8.2}$$

where the image label y is 1 only if the image has the complex concept (a, b) present. During training, we train a single transformation network using positive/negative images from various combinations of the primitives.

8.2.3 Implementation Details

We use linear SVMs, *e.g.*, w_a, w_b trained on `fc7` layer representations from the VGG-M-1024 [375] ConvNet. This ConvNet was pre-trained on the ImageNet dataset [351]. The input classifiers are 1024 dimensional each. The transformation network $\mathcal{T}$ consists of 3 fully connected layers with LeakyReLU [165] non-linearities. We set the slope of LeakyReLU to 0.1. We do *not* update the weights of the ConvNet to ensure a fair comparison to the baselines.

At test time, we first feed in the primitive tuples and cache the classifiers of the complex concepts. Given an image, we then run a single forward pass to get the image features and compute the scores using the cached classifiers.

8.3 Experiments

We now quantify the performance of our method on benchmark datasets. We do so in two settings - 1) compose object and attribute classifiers on the MITStates dataset [187]; and 2) compose three primitives subject, predicate and object classifiers on the Stanford VRD dataset [266].

8.3.1 Common Setup

We first describe the common experimental setup used for these set of experiments.

Metrics: Following [233], we measure the multi-class classification accuracy over the classes in the test split. Existing datasets are not exhaustively labeled [266, 289] in terms of combinations of visual concepts, *i.e.*, a `modern city` can also have `narrow streets` and one or both of these

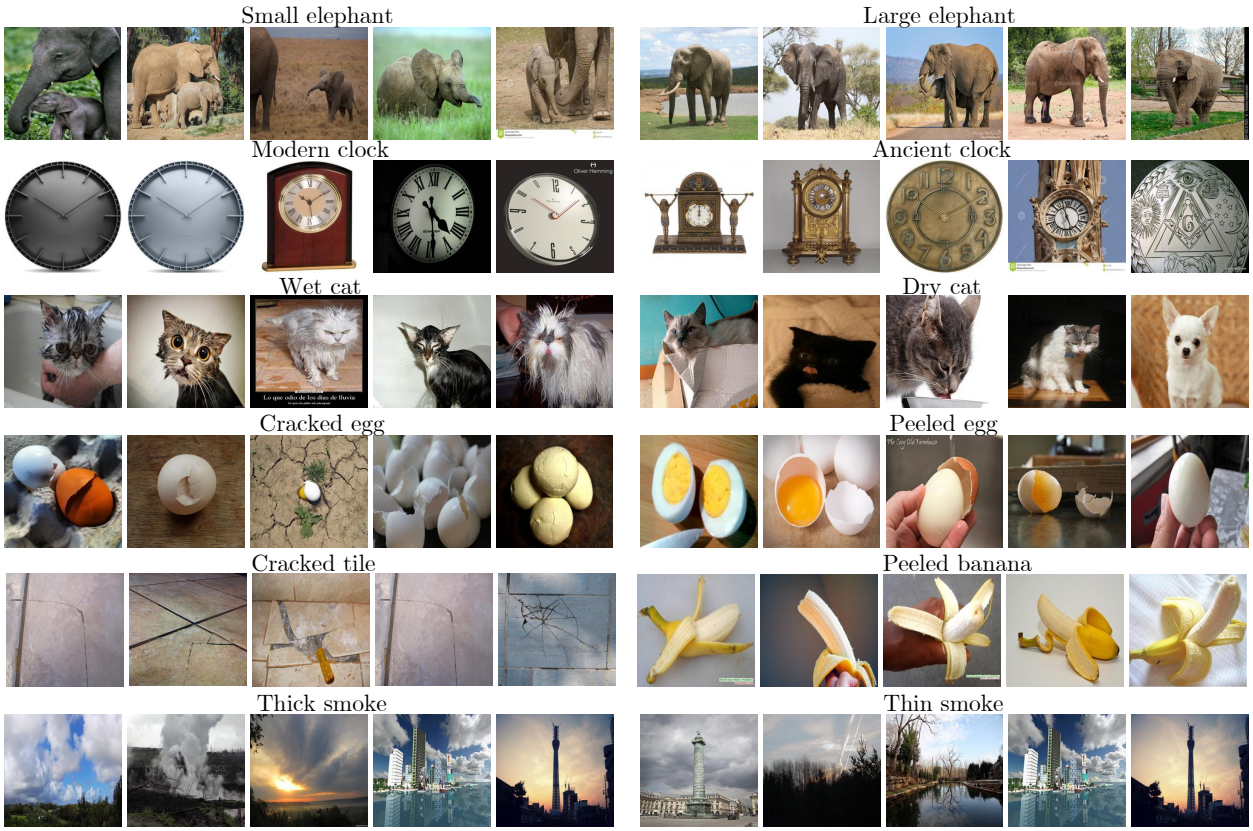


Figure 8.4: We show the top retrievals on the MITStates dataset [187]. These retrievals are computed on unseen combinations of (attribute, object) pairs. We see that our method learns to compose attributes and objects while respecting their context. The last row shows some failure cases of our method.

labels can be missing. To account for this, we follow [351] and use the top-k classification accuracy metric. We also report mean Average Precision [94] (mAP) by computing average precision for each of the classes and taking the mean.

Features and Classifiers: We use the `fc7` representation from the VGG-M-1024 network [375] pre-trained on ImageNet [351]. We learn our base visual classifiers (w_a, w_b) as linear SVMs on these `fc7` features and choose SVM parameters by 4 fold cross-validation using liblinear [100].

Training Details: We describe the architecture for the transform network $\mathcal{T}$ in Section 8.2.3. We train it for $220k$ iterations with a mini-batch size of 256 and momentum 0.9, with a learning rate of 0.01 dropping by a factor of 10 after $200k$ iterations. We form each minibatch with a ratio of 25% positive examples sampled uniformly in the space of the complex visual concepts. The ConvNet weights are not updated for fair comparison. The supplementary material contains additional experiments with end-to-end learning.

Evaluation Setting: As our method does not assume prior knowledge about unseen primitives or complex objects, it is not possible to compare against traditional zero-shot learning methods similar to [233]. Instead, we compare against methods that can directly 'compose' in zero-shot settings without knowing relations to unseen classes at training time.

Baselines without compositionality or contextuality: These baselines do not model compositionality or contextuality explicitly and work directly on the predictions of the base classifiers w_a, w_b. We denote them as:

- **Individual**: This set of baselines does not use compositionality. The probability of the complex concept (a, b) being present in an image is considered to be the probability of only one of the primitives a or b, *i.e.* $p(a, b) = p(a)$ or $p(a, b) = p(b)$. For three primitives a, b, c we can also consider pairs formed by leaving one primitive out, *e.g.*, $p(a, b, c) = p(a)p(b)$ *etc*.
- **Visual Product**: This baseline is inspired from the VisualOnly method from [266]. It does not model contextuality and just 'composes' outputs of classifiers for the primitives, by computing their product, *i.e.*, $p(a, b) = p(a)p(b)$. It can be thought of as late fusion. Unfortunately, since the detectors or training code for [266] were not available at the time of submission, we are unable to directly compare against their implementation/results.

Baselines composing without visual classifiers: These baselines use word embeddings to capture visual similarity, *e.g.*, word embedding of `animal` is closer to `elephant` than `chapter`. They compose using word embeddings of labels, rather than visual classifiers.

- **Label Embeddings (LE)**: This baseline is inspired from the work of [92, 243]. To implement this method, we modify our approach to compute the transform $\mathcal{T}$ on embeddings of the visual primitives, rather than the classifiers. We use the exact same network to compute $\mathcal{T}(e_a, e_b)$, where e_a is an embedding of the primitive a. We use a 300 dimensional word embedding [280] learned using an external corpus (Google News).
- **Label Embeddings Only Regression (LEOR)**: This baseline is inspired from the work of [92, 354, 437]. It is implemented similar to Label Embeddings, except for the loss function. We implement the loss function as a regression to the classifier for the complex visual concept, *e.g.*, a SVM trained on (attribute, object) pairs. Thus transform $\mathcal{T}(e_a, e_b)$ is trained to minimize the euclidean distance to the classifier w_{ab} where w_{ab} is the SVM trained directly on (a, b) pairs.
- **Label Embeddings With Regression (LE+R)**: This baseline combines the loss function from the LE and LEOR baselines and can be viewed as a variation of [92].

8.3.2 Composing objects and attributes

In this section, we learn the transformation on two sets of visual primitives: objects and attributes on the MITStates dataset [187]. We first describe more details about the experimental setup and then present results.

Task: We consider the task of predicting a relevant (attribute, object) pair for a given image in the test set, in an image classification setting. Our test set has (attribute, object) pairs that have never been seen together in the training set. We ensure that all objects and attributes appear *individually* in the training split. We use the unseen (attribute, object) pairs for evaluation using Average Precision and top-k accuracy as described in Section 8.3.1.

Dataset: We use the MITStates Dataset [187] which has pairs of (attribute, object) labels for images (one label per image). It has 245 object classes, 115 attribute classes and about 53k images. We randomly split the dataset into train and test splits such that both splits have *non-overlapping* (attribute, object) pairs. The training split consists of 1292 pairs with 34k images, and the test set

Table 8.1: Evaluating on unseen (attribute, object) pairs on the MITStates Dataset [187]. We evaluate on 700 unseen (attribute, object) pairs on 19k images.

	AP	Top-k Accuracy		
	$k \rightarrow$	1	2	3
Chance	-	0.14	0.28	0.42
Indiv. Att.	2.2	-	-	-
Indiv. Obj.	9.2	-	-	-
Visual Product	8.8	9.8	16.1	20.6
Label Embed (LE)	7.9	11.2	17.6	22.4
LE Only Reg. (LEOR)	4.1	4.5	6.2	11.8
LE+Reg. (LE+R)	6.7	9.3	16.3	20.8
Ours	**10.4**	**13.1**	**21.2**	**27.6**

has 700 pairs with 19k images. Thus, the training and test set have non-overlapping combinations of visual concepts ($\sim 35\%$ unseen concepts) and are suitable for 'zero-shot' learning. We provide more details in the supplementary material.

Baselines: We use the baselines described in Section 8.3.1. We denote the 'Individual' baselines for attributes and objects as 'Indiv Att.' and 'Indiv Obj.' respectively.

Quantitative Results: We summarize the results for our method and the baselines on the MITStates dataset in Table 8.1. We use the *unseen* attribute, object pairs for evaluation. The 'Indiv' baseline methods that do not model both compositionality or contextuality show poor performance. This is to be expected as predicting a `(large, elephant)` using only one of `large` or `elephant` is rather ill-posed. The Indiv Att. baseline performs the worst. We believe the reason is that attribute images are visually very diverse compared to objects (also noted in [319]).

Table 8.1 also shows that the Visual Product baseline gives strong performance in AP, but does not perform well on top-k accuracy. The LE baseline, has the opposite behavior, which suggests that using multiple metrics for evaluation is helpful. We observed that methods with high AP/low accuracy tend to get the object correct while methods with high accuracy/low AP tend to get (attribute, object) pairs correct but generally get objects wrong. Our method shows significant improvement over baselines across all metrics.

We also see that the LEOR and LE+R baselines both have worse performance than the LE baseline. This suggests that using regression to w_{ab} in the loss function is not optimal. On further inspection, we found that the $w_{\text{attribute,object}}$ are poorly trained because of the very few positive examples for (attribute, object) pairs, compared to the larger number examples available for attributes and objects individually. Thus, regressing to these poorly trained classifiers hurts performance. We further explore this in Section 8.4.1.

Qualitative Results: Figure 8.4 shows some qualitative results of our method. For the *unseen* pairs of attributes and objects, we use our transformation $\mathcal{T}$ to predict a classifier and retrieve the top results on the test set. Our model shows both compositionality and contextuality for these concepts. It also shows that our model understands the different 'modes' of appearances for objects. Additional qualitative results are presented in the supplementary material.

Table 8.2: Evaluating subject-predicate-object predictions on unseen tuples. We use the StanfordVRD Dataset [266] with 1029 unseen tuples over 1000 images for evaluation.

	AP	Top-k Accuracy		
	$k \rightarrow$	1	2	3
Chance	-	0.09	0.18	0.27
Indiv. Sub.	2.9	-	-	-
Indiv Pred.	0.4	-	-	-
Indiv. Ob.	3.7	-	-	-
Indiv Sub. Pred.	2.9	-	-	-
Indiv Pred. Ob.	3.6	-	-	-
Indiv Sub. Ob.	4.9	-	-	-
Visual Product	4.9	3.2	5.6	7.6
Label Embed (LE)	4.3	4.1	7.2	10.6
LE Only Reg.(LEOR)	0.9	1.1	1.3	1.3
LE+Reg. (LE+R)	3.9	3.9	7.1	10.4
Ours	**5.7**	**6.3**	**9.2**	**12.7**

We present further results (combining unseen primitives *etc*.) and analysis of our method on this dataset in Section 8.4.

8.3.3 Beyond two primitives: Composing subject, predicate and objects

In this section, we learn our transformation on three sets of visual primitives: subject, predicate and object. We first present additional details on the experimental setup.

Task: We predict a relevant (subject, predicate, object) tuple for a given (ground truth) bounding box from an image. The test set has *unseen tuples* which are used for evaluation. We use the metrics of Average Precision and top-k accuracy as described in Section 8.3.1.

Dataset: We use the recently published StanfordVRD [266] dataset. We use their provided train/test splits of 4k/1k images. The dataset contains SPO subject-predicate-object (generalization of subject-verb-object, SVO) annotations, *e.g.*, `man sitting on a chair` in which the subject-predicate-object tuple is `(man, sitting on, chair)`. In our notation, this dataset consists of three types of visual primitives $(\mathcal{V}_a, \mathcal{V}_b, \mathcal{V}_c)$ as (subject, predicate, object) respectively. The dataset has 7701 such tuples of which 1029 occur only in the test set. The dataset has 100 subjects and objects, and 70 predicates. We use the ground-truth bounding boxes and treat the problem as classification into SPO tuples rather than detection.

Baselines: We use the baselines described in Section 8.3.1. For the 'Individual' baselines we explicitly mention which primitives were used, *e.g.*, 'Indiv Predicate' denotes that only predicate was used, or 'Indiv Pred. Ob.' denotes that product of predicate and object was used.

Quantitative Results: The results for our method and baselines are summarized in Table 8.2. We evaluate all methods on the unseen subject, predicate, object tuples on the test set. Following the trend observed in Section 8.3.2, the 'Indiv' baseline methods show poor performance. Unsurpris-

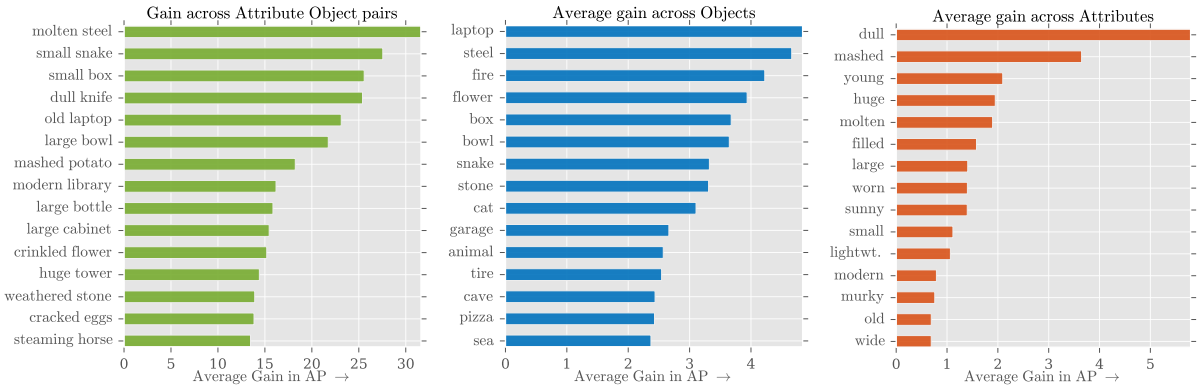


Figure 8.5: We show the top classes and the gain in AP over the Visual Product baseline on the MITStates dataset. We show the gain for (attribute, object) pairs, and for individual objects and attributes (after averaging across pairs).

ingly, predicting a (subject, predicate, object) tuple by considering predictions of only one or two of the primitives does not perform well. We also see that the Indiv. Sub. Ob. baseline shows strong performance, while Indiv. Pred. shows very weak performance. Predicates show higher visual diversity than either subjects or objects and are much more difficult to capture in visual models [266, 354].

Additionally, in Table 8.2, the Indiv Sub. Ob. and Visual Product baselines show similar performance. It again suggests that predicate classifiers do not generalize. Similar to Section 8.3.2, we see that the LEOR and LE+R baselines both have worse performance than the LE baseline. The regression loss used in both these methods regresses to a $w_{\text{subject,predicate,object}}$ classifier. As there are limited examples available for (subject, predicate, object) tuples (compared to examples available individually for the concepts), these classifiers show poor performance (as also noted by [266]). Our method shows improvement over all baseline methods across all metrics, suggesting that the transformation $\mathcal{T}$ has some generalization.

Table 8.3: We analyze the effect of varying the loss function and initialization used to train the transform network $\mathcal{T}$. We test the network on the 700 unseen combinations

Loss	Init	Performance			
		AP	Top-k Accuracy		
			$k \rightarrow$ 1	2	3
Cross Entropy	Gauss.	9.8	10.5	17.4	23.3
Regression	Gauss.	3.1	2.4	3.8	5.1
Cross Ent.+Reg.	Gauss.	7.6	10.2	17.0	22.1
Cross Entropy	Xavier	9.9	10.1	17.2	22.3
Cross Entropy	Identity	**10.4**	**13.2**	**21.2**	**27.6**

Table 8.4: Evaluating on unseen (attribute, object) pairs on the MITStates Dataset [187]. We vary the ratio of unseen pairs to seen pairs and evaluate our method.

Unseen Ratio		AP	Top-k Accuracy		
			$k \rightarrow$ 1	2	3
0.1	Chance	-	1.5	3.0	4.5
0.1	Visual Product	28.7	48.6	58.1	66.2
0.1	Label Embed (LE)	29.2	49.7	59.2	**69.1**
0.1	Ours	**29.8**	**51.4**	**59.6**	68.9
0.3	Chance	-	0.1	0.3	0.4
0.3	Visual Product	8.8	9.8	16.1	20.6
0.3	Label Embed (LE)	7.9	11.1	17.6	22.4
0.3	Ours	**10.4**	**13.2**	**21.2**	**27.6**
0.5	Chance	-	0.1	0.2	0.3
0.5	Visual Product	5.9	6.2	8.8	10.5
0.5	Label Embed (LE)	5.9	7.8	12.6	16.9
0.5	Ours	**8.2**	**10.4**	**17.8**	**23.1**

8.4 Detailed Analysis

We now present detailed analysis of our approach and quantify our architectural design decisions. We also analyze other interesting properties of our learned transform $\mathcal{T}$. For all these experiments, we use the MITStates [187] dataset and follow the experimental setup from Section 8.3.2. We report the results on the 700 unseen pairs from the test set.

8.4.1 Architectural decisions

We analyze the impact of our design decisions on performance of the transformation network $\mathcal{T}$.

Choice of Loss Function: In Section 8.2, we described the Cross Entropy (CE) loss function (Equation 8.2). Here, we explore a few more choices for training our method:

- **Regression:** This loss function is inspired from the work of [92, 437] (also used in Section 8.3). The transform $T(w_{\text{large}}, w_{\text{elephant}})$ is trained to minimize the euclidean distance to the classifier $w_{(\text{large, elephant})}$.
- **Regression+CE:** We combine the Cross Entropy and Regression Loss functions (with a loss weight of 1 each).

Initialization: We found that using standard initialization methods like random gaussian or xavier [144] gave low performance for our transformation network $\mathcal{T}$.

Inspired from [238], we initialize the weights of our network as block diagonal identity matrices. This has the desirable property that immediately from initialization, the network can 'copy' its inputs to produce a reasonable output.

Table 8.3 summarizes the results for both these choices. We notice that the Regression loss performs poorly by itself. As noted in Sections 8.3.2 and 8.3.3, this is because it tries to mimic

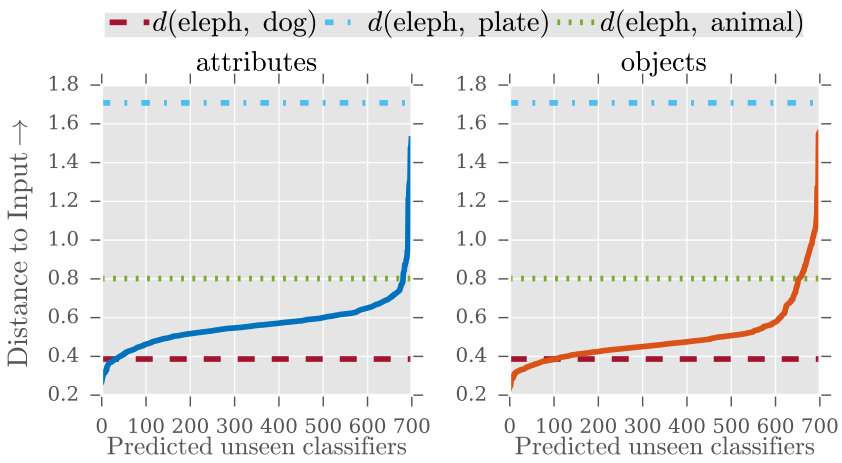


Figure 8.6: We show the distance between the inputs of the transformation network and its output for the unseen pairs. For visualization, we sort this distance individually for both attribute and object inputs. We provide distance between classifiers across 3 pairs of known classes for reference. We see that the transformation modifies all the inputs.

individual classifiers trained for each complex concept, *e.g.*, `(large elephant)`. These classifiers have little data available to train. Among initialization methods, our identity initialization improves performance.

Depth of network: We found that increasing the number of layers in the transformation network $\mathcal{T}$ did not give significant improvements. We opted for the minimal design that gave the best results.

8.4.2 Does the transform 'copy' the inputs?

We compute the distance between the inputs to the transformation network $\mathcal{T}$ and the produced output, *i.e.*, $d(w_a, \mathcal{T}(w_a, w_b))$. We compute this distance over the unseen combinations for the MITStates dataset and show it (after sorting) in Figure 8.6. We see that the transformation changes both inputs and does not just 'copy' them. In the supplementary material, we show that the predicted 'unseen' pairs are different from the 'seen' pairs.

8.4.3 Which classes gain the most?

Figure 8.5 shows the top classes for which our method improves over the Visual Product baseline. We see that the improvement for objects is across both man-made and natural objects. Similarly, the attributes improved by our method have diverse visual interpretations. The pairs for which the baseline is better are generally those where just predicting the object for the (attribute, object) pair gives the best performance, *i.e.*, attributes do not model much information about the object appearance.

Table 8.5: We evaluate our method by composing unseen attributes and objects to form unseen combinations (attributes, objects). We use the MITStates dataset.

	AP	Top-k Accuracy		
	$k \rightarrow$	1	2	3
Chance	-	0.7	1.3	2.0
Visual Product	6.4	7.1	8.6	9.1
Label Embed (LE)	8.4	8.2	12.3	17.4
Ours	**9.6**	**10.1**	**18.3**	**22.9**

8.4.4 Varying the ratio of seen/unseen concepts

We evaluate the effect of decreasing the training data for our transformation network. We vary the ratio of seen/unseen (attribute, object) pairs on MITStates from $[0.1, 0.3, 0.5]$, and train our network for each setting. We compare to the Visual Product and LE baselines from Section 8.3.1. Table 8.4 summarizes the results. We see that our algorithm is sensitive to the amount of training data available. It also shows improvement over baseline methods in all these settings. Comparing the performance for unseen ratios of 0.1 to 0.5, we see that our method's gain over baselines increases as we reduce the training data.

8.4.5 Moving from unseen combinations of primitives to unseen primitives

In this set of experiments, we randomly drop a set of object and attribute primitives from the training set of our transform network $\mathcal{T}$. The network never sees these classifiers at training time. At test time, we evaluate on the attribute, object pairs formed by these 'dropped' primitives. Concretely, we randomly drop 20% of objects and attributes: 49 of the 245 objects and 23 of the 115 attributes. We evaluate on 142 (attribute, object) pairs formed by these dropped primitives. We report these results in Table 8.5. Our method is able to generalize to these unseen input primitives and combine them to form the unseen pairs of concepts.

8.5 Conclusion

We presented a simple approach to compose classifiers to generate classifiers for new complex concepts. Our experiments on composing attributes and objects show that our method respects contextuality. We also show that our method can compose multiple primitives, and can generalize not only to unseen combinations of primitives, but also unseen primitives. It consistently gives better results than the baselines across different metrics and datasets. Thus, in this section we have seen that using structure in the classifier space can help us use simple image synthesis techniques (Chapter 7) and compose classifiers (this chapter). All of these methods help reduce the need for human supervision.

Part IV

Interactive Learning

Chapter 9

Learning to Ask Questions

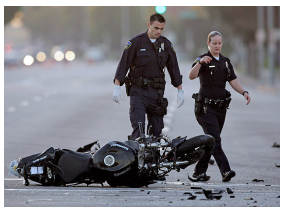
If you want a wise answer, ask a reasonable question.

JOHANN WOLFGANG VON GOETHE in *Wilhelm Meister's Lehrjahre*, 1830

In this section, we argue that next-generation recognition systems need to have *agency* — the ability to decide what information they need and how to get it. Much of the progress is machine learning is due to fully supervised models which are *passively* fed supervision. These models do not have any control over the supervision fed to them. This is in stark contrast to the way we humans learn — by *interacting* with our environment to gain information. The interactive nature of human learning makes it sample efficient (there is less redundancy during training) and also yields a learning curriculum (we ask for more complex knowledge as we learn).

While there are multiple ways to interact with an environment, in this thesis we focus on interactions using natural language, *i.e.*, a question asking-answering setup. We will study these interactions by first proposing a way to *ask* questions in this Chapter. In the next Chapter 10, we will show a self-contained agent that learns purely by interaction.

Learning to ask questions is an important task in AI and is more than a syntactic transformation of a declarative sentence [420]. Deciding what to ask about demonstrates understanding and as such, question generation provides an indication of machine understanding, just as some



Natural Questions:
- Was anyone injured in the crash?
- Is the motorcyclist alive?
- What caused this accident?

Generated Caption:
- A man standing next to a motorcycle.

Figure 9.1: Example image along with its natural questions and automatically generated caption.

educational methods assess students' understanding by their ability to ask relevant questions[1]. Furthermore, training a system to ask a good question (not only answer a question) may imbue the system with what appears to be a cognitive ability unique to humans among other primates [200]. Developing the ability to ask relevant and to-the-point questions can be an essential component of any dynamic learner which seeks information. Such an ability can be an integral component of any conversational agent, either to engage the user in starting a conversation or to elicit task-specific information. In this chapter, we see how an agent can learn to ask questions and start an engaging conversation about images.

Imagine someone has shared the image in Figure 9.1. What is the very first question that comes to mind? Your question is most probably very similar to the questions listed next to the image, expressing concern about the motorcyclist (who is not even present in the image). As you can tell, natural questions are not about what is seen, the policemen or the motorcycle, but rather about what is inferred given these objects, e.g., an accident or injury. As such, questions are often about abstract concepts, i.e., events or states, in contrast to the concrete terms[2] used in image captioning. It is clear that the corresponding automatically generated caption[3] for Figure 9.1 presents only a literal description of objects.

To move beyond the literal description of image content, we introduce the novel task of Visual Question Generation (VQG), where given an image, the system should 'ask a natural and engaging question'. Asking a question that can be answered simply by looking at the image would be of interest to the Computer Vision community, but such questions are neither natural nor engaging for a person to answer and so are not of interest for the task of VQG.

The contributions of this chapter can be summarized as follows: (1) in order to enable the VQG research, we carefully created three datasets with a total of 75,000 questions, which range from object- to event-centric vqg/images, where we show that VQG covers a wide range of abstract terms including events and states (Section 9.2). (2) we collected 25,000 gold captions for our event-centric dataset and show that this dataset presents challenges to the state-of-the-art image captioning models (Section 9.2.3). (3) we perform analysis of various generative and retrieval approaches and conclude that end-to-end deep neural models outperform other approaches on our most-challenging dataset (Section 9.3). (4) we provide a systematic evaluation methodology for this task, where we show that the automatic metric ΔBLEU strongly correlates with human judgments (Section 9.4.3). The results show that while our models learn to generate promising questions, there is still a large gap to match human performance, making the generation of relevant and natural questions an interesting and promising new challenge to the community.

9.1 Background

For the task of image captioning, datasets have primarily focused on objects, e.g. Pascal VOC [96] and Microsoft Common Objects in Context (MS COCO) [252]. MS COCO, for example, includes

[1] http://rightquestion.org/

[2] Concrete terms are the ones that can be experienced with five senses. Abstract terms refer to intangible things, such as feelings, concepts, and qualities

[3] Throughout this chapter we use the state-of-the-art captioning system [101], henceforth MSR captioning system https://www.captionbot.ai/, to generate captions.

complex everyday scenes with 91 basic objects in 328k vqg/images, each paired with 5 captions. Event detection is the focus in video processing and action detection, but these do not include a textual description of the event [21, 50, 446, 452]. The number of actions in each of these datasets is still relatively small, ranging from 40 [453] to 600 [50] and all involve human-oriented activity (e.g. 'cooking', 'gardening', 'riding a bike'). In our work, we are focused on generating questions for static vqg/images of events, such as 'fire', 'explosion' or 'snowing', which have not yet been investigated in any of the above datasets.

Visual Question Answering is a task where the system provides an answer to a question about the image content. The most notable, Visual Question Answering (VQA) [24], is an open-ended (free-form) dataset, in which both the questions and the answers are crowd-sourced, with workers prompted to ask a visually verifiable question which will 'stump a smart robot'. Gao et al. [132] used similar methodology to create a visual question answering dataset in Chinese. COCO-QA (CQA) [339], in contrast, does not use human-authored questions, but generates questions automatically from image captions of the MS COCO dataset by applying a set of transformation rules to generate the wh-question. The expected answers in CQA are by design limited to objects, numbers, colors, or locations. A more in- depth analysis of VQA and CQA datasets will be presented in Section 9.2.1.

In this work, we focus on questions which are interesting for a person to answer, not questions designed to evaluate computer vision. A recently published work on VQA, Visual7W [482], establishes a grounding link on the object regions corresponding to the textual answer. This setup enables a system to answer a question with visual answers (in addition to textual answers). They collect a set of 327,939 7W multiple-choice QA pairs, where they point out that 'where', 'when' and 'why' questions often require high-level commonsense reasoning, going beyond spatial reasoning required for 'which' or 'who' questions. This is more in line with the type of questions that VQG captures, however, the majority of the questions in Visual7w are designed to be answerable by only the image, making them unnatural for asking a human. Thus, learning to generate the questions in VQA task is not a useful sub-task, as the intersection between VQG and any VQA questions is by definition minimal.

Previous work on question generation from textual input has focused on two aspects: the grammaticality [167, 293, 441] and the content focus of question generation, i.e., "what to ask about". For the latter, several methods have been explored: [32] create fill-in-the-blank questions, [275] and [254] use manually constructed question templates, while [227] use crowd-sourcing to collect a set of templates and then rank the potentially relevant templates for the selected content. To our knowledge, neither a retrieval model nor a deep representation of textual input, presented in our work, have yet been used to generate questions.

9.2 Data Collection Methodology

Task Definition: Given an image, the task is to generate a natural question which can potentially engage a human in starting a conversation. Questions that are visually verifiable, i.e., that can be answered by looking at only the image, are outside the scope of this task. For instance, in Figure 9.2, a question about the number of horses (appearing in the VQA dataset) or the color of the field is not of interest. Although in this chapter we focus on asking a question about an image in isolation, adding prior context or history of conversation is the natural next step in this project.

Figure 9.2: Example right and wrong questions for the task of VQG.

We collected the VQG questions by crowd-sourcing the task on Amazon Mechanical Turk (AMT). We provide details on the prompt and the specific instructions for all the crowdsourcing tasks in this chapter in the supplementary material. Our prompt was very successful at capturing non-literal questions, as the good question in Figure 9.2 demonstrates. In the following Sections, we describe our process for selecting the vqg/images to be included in the VQG dataset. We start with vqg/images from MS COCO, which enables meaningful comparison with VQA and CQA questions. Given that it is more natural for people to ask questions about event-centric vqg/images, we explore sourcing eventful vqg/images from Flickr and from querying an image search engine. Each data source is represented by 5,000 vqg/images, with 5 questions per image.

9.2.1 $VQG_{coco-5000}$ and $VQG_{Flickr-5000}$

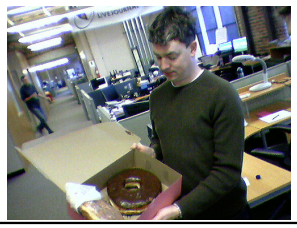
As our first dataset, we collected VQG questions for a sample of vqg/images from the MS COCO dataset[4]. In order to enable comparisons with related datasets, we sampled 5,000 vqg/images of MS COCO which were also annotated by the CQA dataset [339] and by VQA [24]. We name this dataset $VQG_{coco-5000}$. Table 9.1 shows a sample MS COCO image along with annotations in the various datasets. As the CQA questions are generated by rule application from captions, they are not always coherent. The VQA questions are written to evaluate the detailed visual understanding of a robot, so their questions are mainly visually grounded and literal. The table demonstrates how different VQG questions are from VQA and CQA questions.

In Figure 9.3 we provide statistics for the various annotations on that portion of the MS COCO vqg/images which are represented in the $VQG_{coco-5000}$ dataset. In Figure 9.3(a) we compare the percentage of object-mentions in each of the annotations. Object-mentions are words associated with the gold-annotated object boundary boxes[5] as provided with the MS COCO dataset. Naturally, COCO captions (green bars) have the highest percentage of these literal objects. Since object-mentions are often the answer to VQA and CQA questions, those questions naturally contain objects less frequently. Hence, we see that VQG questions include the mention of more of those literal objects. Figure 9.3(b) shows that COCO captions have a larger vocabulary size, which reflects their longer and more descriptive sentences. VQG shows a relatively large vocabulary size as well, indicating greater diversity in question formulation than VQA and CQA. Moreover, Figure 9.3(c) shows that the verb part of speech is represented with high frequency in our dataset.

Figure 9.3(d) depicts the percentage of abstract terms such as 'think' or 'win' in the vocabulary. Following Ferraro et al. [115], we use a list of most common abstract terms in English [421], and

[4] http://mscoco.org/

[5] Note that MS COCO annotates only 91 object categories.



Dataset	Annotations
COCO	- A man holding a box with a large chocolate covered donut.
CQA	- What is the man holding with a large chocolate-covered doughnut in it?
VQA	- Is this a large doughnut?
VQG	- Why is the donut so large? - Is that for a specific celebration? - Have you ever eaten a donut that large before? - Is that a big donut or a cake? - Where did you get that?

Table 9.1: Dataset annotations on the above image.

count all the other words except a set of function words as concrete. This figure supports our expectation that VQG covers more abstract concepts. Furthermore, Figure 9.3(e) shows inter-annotation textual similarity according to the BLEU metric [314]. Interestingly, VQG shows the highest inter-annotator textual similarity, which reflects on the existence of consensus among human for asking a natural question, even for object-centric vqg/images like the ones in MS COCO.

The MS COCO dataset is limited in terms of the concepts it covers, due to its pre-specified set of object categories. Word frequency in $VQG_{coco-5000}$ dataset, as demonstrated in Figure 9.4, bears this out, with the words 'cat' and 'dog' the fourth and fifth most frequent words in the dataset. Not shown in the frequency graph is that words such as 'wedding', 'injured', or 'accident' are at the very bottom of frequency ranking list. This observation motivated the collection of the $VQG_{Flickr-5000}$ dataset, with vqg/images appearing as the middle photo in a story-full photo album [183] on Flickr[6]. The details about this dataset can be found in the supplementary material.

9.2.2 $VQG_{Bing-5000}$

To obtain a more representative visualization of specific event types, we queried a search engine[7] with 1,200 event-centric query terms which were obtained as follows: we aggregated all 'event' and 'process' hyponyms in WordNet [282], 1,000 most frequent TimeBank events [327] and a set of manually curated 30 stereotypical events, from which we selected the top 1,200 queries based on Project Gutenberg word frequencies. For each query, we collected the first four to five vqg/images retrieved, for a total of 5,000 vqg/images, having first used crowdsourcing to filter out vqg/images depicting graphics and cartoons.

[6] http://www.flickr.com
[7] https://datamarket.azure.com/dataset/bing/search

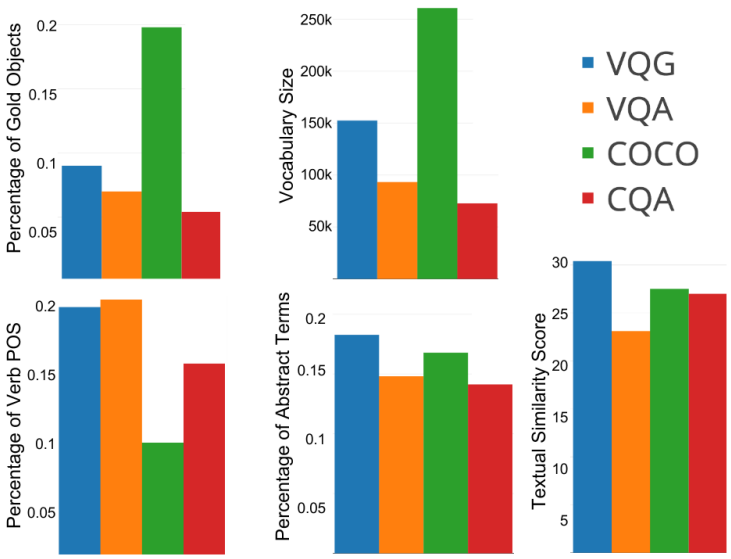


Figure 9.3: Comparison of various annotations on the MS COCO dataset. (a) Percentage of gold objects used in annotations. (b) Vocabulary size (c) Percentage of verb POS (d) Percentage of abstract terms (e) Inter-annotation textual similarity score.

A similar word frequency analysis shows that the $VQG_{Bing-5000}$ dataset indeed contains more words asking about events: *happen*, *work*, *cause* appear in top 40 words, which was our aim in creating the Bing dataset.

Statistics: Our three datasets together cover a wide range of visual concepts and events, totaling 15,000 vqg/images with 75,000 questions.

Figure 9.5 draws the histogram of number of tokens in VQG questions, where the average question length is 6 tokens. Figure 9.6 visualizes the n-gram distribution (with n=6) of questions in the three VQG datasets[8]. Table 9.2 shows the statistics of the crowdsourcing task.

# all vqg/images	**15,000**
# questions per image	**5**
# all workers participated	308
Max # questions written by one worker	6,368
Average work time per worker (sec)	106.5
Median work time per worker (sec)	23.0
Average payment per question (cents)	6.0

Table 9.2: Statistics of crowdsourcing task, aggregating all three datasets.

[8]Please refer to our web page on http://research.microsoft.com/en-us/downloads to get a link to a dynamic visualization and statistics of all n-gram sequences.

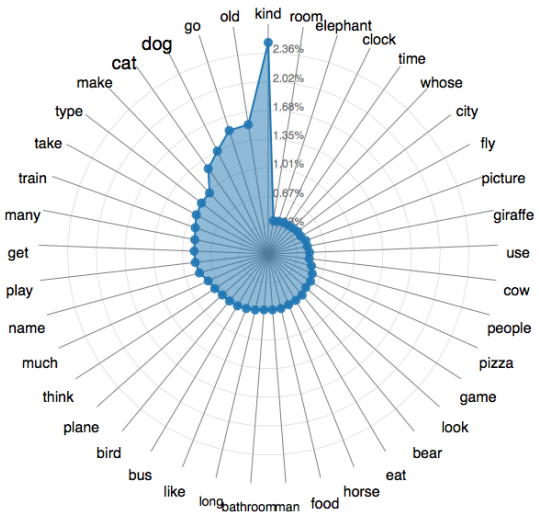


Figure 9.4: Frequency graph of top 40 words in $VQG_{coco-5000}$ dataset.

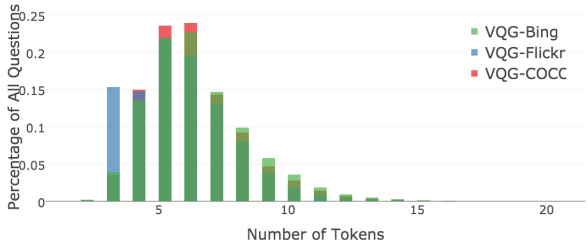


Figure 9.5: Average annotation length of the three VQG datasets.

9.2.3 $Captions_{Bing-5000}$

The word frequencies of questions about the $\boldsymbol{VQG_{Bing-5000}}$ dataset indicate that this dataset is substantially different from the MS COCO dataset. Human evaluation results of a recent work [413] further confirms the significant image captioning quality degradation on out-of-domain data. To further explore this difference, we crowdsourced 5 captions for each image in the $VQG_{Bing-5000}$ dataset using the same prompt as used to source the MS COCO captions. We call this new dataset $\boldsymbol{Captions_{Bing-5000}}$. Table 9.3 shows the results of testing the state-of-the-art MSR captioning system on the $\boldsymbol{Captions_{Bing-5000}}$ dataset as compared to the MS COCO dataset, measured by the standard BLEU [314] and METEOR [81] metrics. The wide gap in the results further confirms that indeed the $\boldsymbol{VQG_{Bing-5000}}$ dataset covers a new class of vqg/images; we hope the availability of this new dataset will encourage including more diverse domains for image captioning.

BLEU		**METEOR**	
Bing	*MS COCO*	*Bing*	*MS COCO*
0.101	0.291	0.151	0.247

Table 9.3: Image captioning results

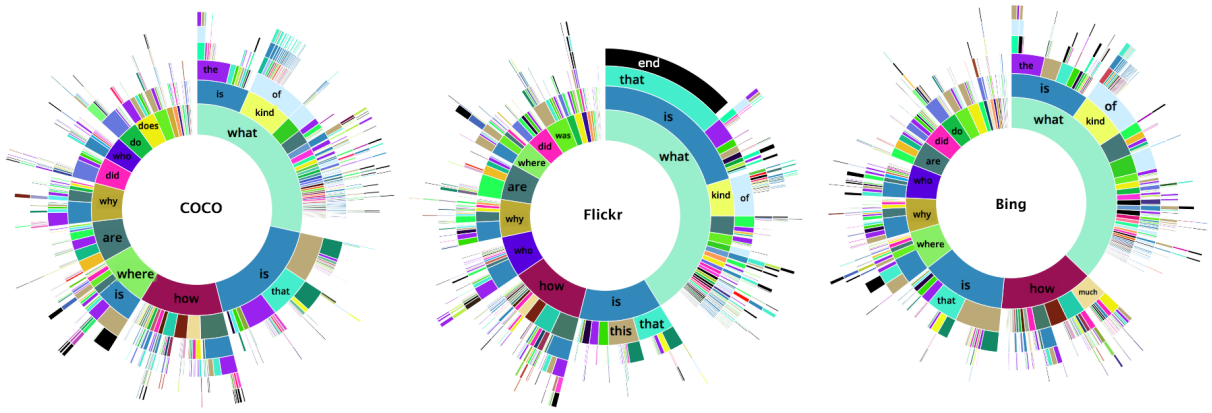


Figure 9.6: VQG N-gram sequences. 'End' token distinguishes natural ending with n-gram cut-off.

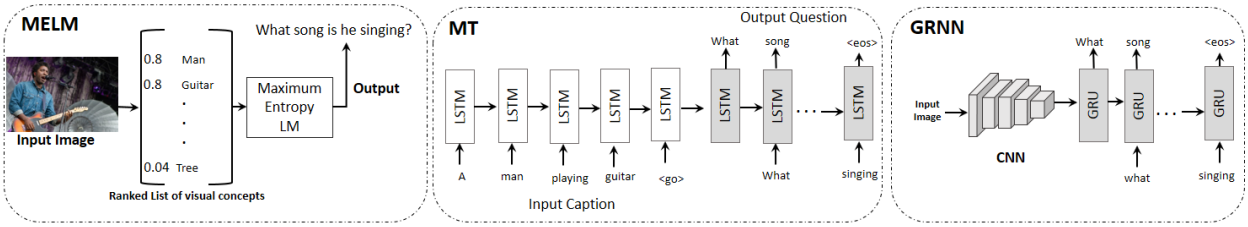


Figure 9.7: Three different generative models for tackling the task of VQG.

9.3 Models

In this Section we present several generative and retrieval models for tackling the task of VQG. For all the forthcoming models we use the VGGNet [375] architecture for computing deep convolutional image features. We primarily use the 4096-dimensional output the last fully connected layer ($fc7$) as the input to the generative models.

9.3.1 Generative Models

Figure 9.7 represents an overview of our three generative models. The MELM model [101] is a pipeline starting from a set of candidate word probabilities which are directly trained on vqg/images, which then goes through a maximum entropy (ME) language model. The MT model is a Sequence2Sequence translation model [60, 397] which directly translates a description of an image into a question, where we used the MS COCO captions and $Captions_{Bing-5000}$ as the source of translation. These two models tended to generate less coherent sentences, details of which can be found in the supplementary material. We obtained the best results by using an end-to-end neural model, GRNN, as follows.

Gated Recurrent Neural Network (GRNN): This generation model is based on the state-of-the-art multimodal Recurrent Neural Network model used for image captioning [82, 426]. First, we transform the $fc7$ vector to a 500-dimensional vector which serves as the initial recurrent state to a 500-dimensional Gated Recurrent Unit (GRU). We produce the output question one word at a time

Q.	 Explosion	 Hurricane	 Rain Cloud
Human	- What caused this explosion? - Was this explosion an accident?	- What caused the damage to this city? - What happened to this place?	- Are those rain clouds? - Did it rain?
GRNN	- How much did the fire cost? - What is being burned here?	- What happened to the city? - What caused the fall?	- What kind of clouds are these? - Was there a bad storm?
KNN	- What caused this fire?	- What state was this earthquake in?	- Did it rain?
Caption	- A train with smoke coming from it.	- A pile of dirt.	- Some clouds in a cloudy day.

Table 9.4: Sample generations by different systems on $VQG_{bing-5000}$, in order: Human$_{consensus}$ and Human$_{random}$, GRNN$_{bing}$ and GRNN$_{all}$, KNN+min$_{bleu-all}$, MSR captions. Q is the query-term.

using the GRU, until we hit the end-of-sentence token. We train the GRU and the transformation matrix jointly, but we do not back-propagate the CNN due to the size of the training data. The neural network is trained using Stochastic Gradient Descent with early stopping, and decoded using a beam search of size 8. The vocabulary consists of all words seen 3 or more times in the training, which amounts to 1942 unique tokens in the full training set. Unknown words are mapped to to an `<unk>` token during training, but we do not allow the decoder to produce this token at test time.

9.3.2 Retrieval Methods

Retrieval models use the caption of a nearest neighbor training image to label the test image [82, 105, 174, 309]. For the task of image captioning, it has been shown that up to 80% of the captions generated at test time by a near state-of-the-art generation approach [426] were exactly identical to the training set captions, which suggests that reusing training annotations can achieve good results. Moreover, basic nearest neighbor approaches to image captioning on the MS COCO dataset are shown to outperform generation models according to automatic metrics [82]. The performance of retrieval models of course depends on the diversity of the dataset.

We implemented several retrieval models customized for the task of VQG. As the first step,

we compute K nearest neighbor vqg/images for each test image using the $fc7$ features to get a candidate pool. We obtained the most competitive results by setting K dynamically, as opposed to the earlier works which fix K throughout the testing. We observed that candidate vqg/images beyond a certain distance made the pool noisy, hence, we establish a parameter called $max\text{-}distance$ which is an upper bound for including a neighbor image in the pool. Moreover, our experiments showed that if there exists a very similar image to the test image, the candidate pool can be ignored and that test image should become the only candidate[9]. For addressing this, we set a $min\text{-}distance$ parameter. All these parameters were tuned on the corresponding validation sets using the Smoothed-BLEU [250] metric against the human reference questions.

Given that each image in the pool has five questions, we define the one-best question to be the question with the highest semantic similarity[10] to the other four questions. This results in a pool of K candidate questions. The following settings were used for our final retrieval models:

- **1-NN**: Set K=1, which retrieves the closest image and emits its one-best.
- **K-NN+min**: Set K=30 with $max\text{-}distance = 0.35$, and $min\text{-}distance = 0.1$. Among the 30 candidate questions (one-best of each image), find the question with the highest similarity to the rest of the pool and emit that: we compute the textual similarity according the two metrics, Smoothed-BLEU and Average-Word2Vec (gensim)[11].

Table 9.4 shows a few example vqg/images along with the generations of our best performing systems. For more examples please refer to the web page of the project.

9.4 Evaluation

While in VQG the set of possible questions is not limited, there is consensus among the natural questions (discussed in Section 9.2.1) which enables meaningful evaluation. Although human evaluation is the ideal form of evaluation, it is important to find an automatic metric that strongly correlates with human judgment in order to benchmark progress on the task.

9.4.1 Human Evaluation

The quality of the evaluation is in part determined by how the evaluation is presented. For instance, it is important for the human judges to see various system hypotheses at the same time in order to give a calibrated rating. We crowdsourced our human evaluation on AMT, asking three crowd workers to each rate the quality of candidate questions on a three-point semantic scale.

[9] At test time, the frequency of finding a train set image with $distance \leq 0.1$ is 7.68%, 8.4% and 3.0% in COCO, Flickr and Bing datasets respectively.

[10] We use BLEU to compute textual similarity. This process eliminates outlier questions per image.

[11] Average-Word2Vec refers to the sentence-level textual similarity metric where we compute the cosine similarity between two sentences by averaging their word-level Word2Vec [281] vector representations. Here we use the GenSim software framework [338].

		$\text{Human}_{consensus}$	Human_{random}	GRNN_{X}	GRNN_{all}	1-NN_{bleu-X}	$\text{1-NN}_{gensim-X}$	K-NN+min_{bleu-X}	$\text{K-NN+min}_{gensim-X}$	$\text{1-NN}_{bleu-all}$	$\text{1-NN}_{gensim-all}$	$\text{K-NN+min}_{bleu-all}$	$\text{K-NN+min}_{gensim-all}$
Human Evaluation													
	Bing	2.49	2.38	1.35	**1.76**	1.72	1.72	1.69	1.57	1.72	1.73	1.75	1.58
	COCO	2.49	2.38	1.66	1.94	1.81	1.82	1.88	1.64	1.82	1.82	**1.96**	1.74
	Flickr	2.34	2.26	1.24	**1.57**	1.44	1.44	1.54	1.28	1.46	1.46	1.52	1.30
Automatic Evaluation													
BLEU	Bing	87.1	83.7	**12.3**	11.1	9.0	9.0	11.2	7.9	9.0	9.0	11.8	7.9
	COCO	86.0	83.5	13.9	14.2	11.0	11.0	19.1	11.5	10.7	10.7	**19.2**	11.2
	Flickr	84.4	83.6	9.9	9.9	7.4	7.4	10.9	5.9	7.6	7.6	**11.7**	5.8
MET.	Bing	62.2	58.8	**16.2**	15.8	14.7	14.7	15.4	14.7	14.7	14.7	15.5	14.7
	COCO	60.8	58.3	18.5	18.5	16.2	16.2	**19.7**	17,4	15.9	15.9	19.5	17.5
	Flickr	59.9	58.6	14.3	**14.9**	12.3	12.3	13.6	12.6	12.6	12.6	14.6	13.0
ΔBLEU	Bing	63.38	57.25	**11.6**	10.8	8.28	8.28	10.24	7.11	8.43	8.43	11.01	7.59
	COCO	60.81	56.79	12.45	12.46	9.85	9.85	16.14	9.96	9.78	9.78	**16.29**	9.96
	Flickr	62.37	57.34	9.36	9.55	6.47	6.47	9.49	5.37	6.73	6.73	**9.8**	5.26

Table 9.5: Results of evaluating various models according to different metrics. X represents training on the corresponding dataset in the row. Human score per model is computed by averaging human score across multiple vqg/images, where human score per image is the median rating across the three raters.

	METEOR	**BLEU**	**ΔBLEU**
r	**0.916** (4.8e-27)	0.915 (4.6e-27)	0.915 (5.8e-27)
ρ	0.628 (1.5e-08)	0.67 (7.0e-10)	**0.702** (5.0e-11)
τ	0.476 (1.6e-08)	0.51 (7.9e-10)	**0.557** (3.5e-11)

Table 9.6: Correlations of automatic metrics against human judgments, with p-values in parentheses.

9.4.2 Automatic Evaluation

The goal of automatic evaluation is to measure the similarity of system-generated question hypotheses and the crowdsourced question references. To capture n-gram overlap and textual similarity between hypotheses and references, we use standard Machine Translation metrics, BLEU [314] and METEOR [81]. We use BLEU with equal weights up to 4-grams and default setting of METEOR version 1.5. Additionally we use ΔBLEU [131] which is specifically tailored towards generation tasks with diverse references, such as conversations. ΔBLEU requires rating per reference, distinguishing between the quality of the references. For this purpose, we crowd-sourced three human ratings (on a scale of 1-3) per reference and used the majority rating.

The pairwise correlational analysis of human and automatic metrics is presented in Table 9.6, where we report on Pearson's r, Spearman's ρ and Kendall's τ correlation coefficients. As this table reveals, ΔBLEU strongly correlates with human judgment and we suggest it as the main evaluation metric for testing a VQG system. It is important to note that BLEU is also very competitive with ΔBLEU, showing strong correlations with human judgment. Hence, we recommend using BLEU for any further benchmarking and optimization purposes. BLEU can also be used as a proxy for ΔBLEU for evaluation purposes whenever rating per reference are not available.

9.4.3 Results

In this section, we present the human and automatic metric evaluation results of the models introduced earlier. We randomly divided each VQG-5000 dataset into train (50%), val (25%) and test (25%) sets. In order to shed some light on differences between our three datasets, we present the evaluation results separately on each dataset in Table 9.5. Each model (Section 9.3.2) is once trained on all train sets, and once trained only on its corresponding train set (represented as X in the results table). For quality control and further insight on the task, we include two human annotations among our models: 'Human$_{consensus}$' (the same as one-best) which indicates the consensus human annotation on the test image and 'Human$_{random}$' which is a randomly chosen annotation among the five human annotations.

It is quite interesting to see that among the human annotations, Human$_{consensus}$ achieves consistently higher scores than Human$_{random}$. This further verifies that there is indeed a common intuition about what is the most natural question to ask about a given image. As the results of human evaluation in Table 9.5 shows, GRNN$_{all}$ performs the best as compared with all the other models in 2/3 of runs. All the models achieve their best score on $VQG_{COCO-5000}$, which was expected given the less diverse set of vqg/images. Using automatic metrics, the GRNN$_X$ model outperforms other models according to all three metrics on the $VQG_{Bing-5000}$ dataset.

Among retrieval models, the most competitive is K-NN+min_bleu_all, which performs the best on $VQG_{COCO-5000}$ and $VQG_{Flickr-5000}$ datasets according to BLEU and ΔBLEU score. This further confirms our effective retrieval methodology for including $min\text{-}distance$ and n-gram overlap similarity measures. Furthermore, the boost from 1-NN to K-NN models is considerable according to both human and automatic metrics. It is important to note that none of the retrieval models beat the GRNN model on the Bing dataset. This additionally shows that our Bing dataset is in fact more demanding, making it a meaningful challenge for the community.

9.5 Discussion

In this chapter, we introduced the novel task of 'Visual Question Generation', where given an image, the system is tasked with asking a natural question. This task focuses on developing the capability to ask relevant and to-the-point questions, a key intelligent behavior that an AI system should demonstrate. We believe that VQG is one step towards building such a system, where an engaging question can naturally start a conversation. Another natural extension of this work is to include question generation within a conversational system [244, 385], where the context and conversation history affect the types of questions being asked. In the next chapter, we will see such an extension in which an agent must ask questions in order to learn new concepts.

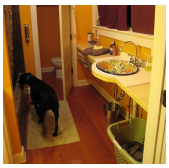
		
Human	- How long did it take to make that ice sculpture?	- Is the dog looking to take a shower?
GRNN	- How long has he been hiking?	- Is this in a hotel room?
KNN	- How deep was the snow?	- Do you enjoy the light in this bathroom?

Table 9.7: Examples of errors in generation. The rows are Human$_{consensus}$, GRNN$_{all}$, and KNN+min$_{bleu-all}$.

Chapter 10

Learning by Asking Questions

[...]Plain question and plain answer make the shortest road out of most perplexities.

MARK TWAIN in *Life on the Mississippi*, 1883

Chapter 9 introduced the task of Visual Question Generation (VQG) and showed how models can be trained to ask meaningful questions. These interactive agents that can ask questions are important for sample efficient learning that generalizes to unseen distributions. However, a major drawback of VQG is the subjective evaluation of the generated questions. In this chapter, we propose an interactive learning framework that has a task-driven objective evaluation of the generated questions.

We explore interactive learners in the context of visual question answering (VQA; [23, 198, 481]). Instead of training on a fixed, large-scale dataset, we propose an alternative *interactive* VQA setup called *learning-by-asking* (LBA): at training time, the learner receives only images and decides *what questions to ask*. Questions asked by the learner are answered by an oracle (human supervision). At test-time, LBA is evaluated exactly like VQA using well understood metrics. Interactive learners which can ask questions and acquire supervision are a key requirement as we move towards more AI-complete tasks as it is extremely hard to curate and collect datasets for these tasks. As Mark Twain said, asking questions and getting answers is a vital skill for solving complex problems in life.

The interactive nature of LBA requires the learner to construct meta-knowledge about what it knows and to select the supervision it needs. If successful, this facilitates more sample efficient learning than using a fixed dataset, because the learner will not ask redundant questions.

We explore the proposed LBA paradigm in the context of the CLEVR dataset [198], which is an artificial universe in which the number of unique objects, attributes, and relations are limited. We opt for this synthetic setting because there is little prior work on asking questions about images: CLEVR allows us to perform a controlled study of the algorithms needed for asking questions. We hope to transfer the insights obtained from our study to a real-world setting.

Building an interactive learner that can ask questions is a challenging task. First, the learner

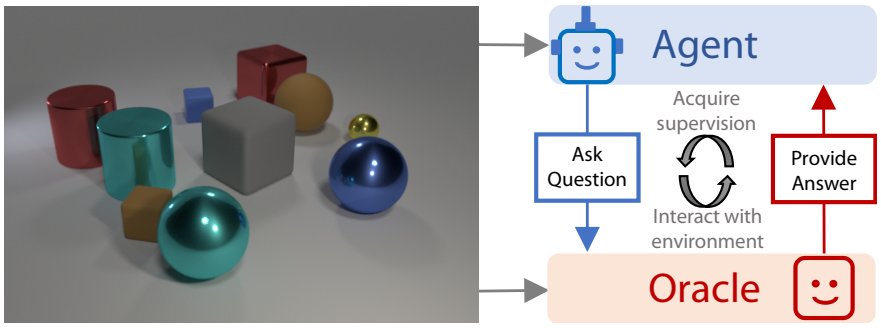


Figure 10.1: **The Learning-by-Asking (LBA) paradigm.** We present an open-world Visual Question Answering (VQA) setting in which an agent interactively learns by asking questions to an oracle. Unlike standard VQA training, which assumes a fixed dataset of questions, in LBA the agent has the potential to learn more quickly by asking "good" questions, much like a bright student in a class. LBA does not alter the test-time setup of VQA.

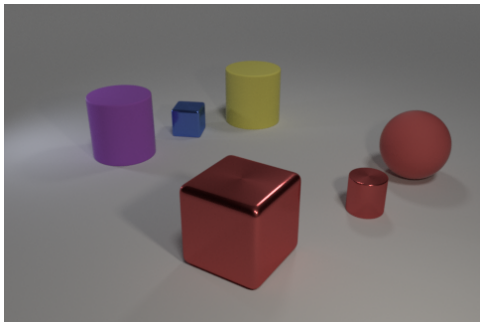
needs to have a "language" model to form questions. Second, it needs to understand the input image to ensure the question is relevant and coherent. Finally (and most importantly), in order to be sample efficient, the learner should be able to evaluate its own knowledge (self-evaluate) and ask questions which will help it to learn new information about the world. The only supervision the learner receives from the interaction is the answer to the questions it poses.

We present and study a model for LBA that combines ideas from visually grounded language generation [297], curriculum learning [35], and VQA. Specifically, we develop an epsilon-greedy [398] learner that asks questions and uses the corresponding answers to train a standard VQA model. The learner focuses on mastering concepts that it can rapidly improve upon, before moving to new types of questions. We demonstrate that our LBA model not only asks meaningful questions, but also *matches the performance* of human-curated data. Our model is also *sample efficient* and by interactively asking questions it reduces the number of training samples needed to obtain the baseline question-answering accuracy by 40%.

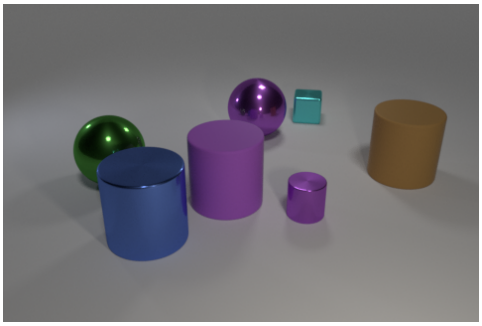
10.1 Background

Visual question answering (VQA) is a surrogate task designed to assess a system's ability to thoroughly understand images. It has gained popularity in recent years due to the release of several benchmark datasets [23, 269, 481]. Motivated by the well-studied difficulty of analyzing results on real-world VQA datasets [190, 331, 463], Johnson *et al*. [198] recently proposed a more controlled, synthetic VQA dataset that we adopt in this work.

Current VQA approaches follow a traditional supervised learning paradigm. A large number of image-question-answer triples are collected and a subset of this data is randomly selected for training. Learning-by-asking (LBA) uses an alternative and more challenging setting: training images are drawn from a distribution, but the learner decides what question it needs to ask to learn the most. The learner receives only answer level supervision from these interactions. It must learn to formulate questions as well as model its own knowledge to remove redundancy in question-asking. LBA also has the potential to generalize to open-world scenarios.



✗ What size is the purple cube?
✗ What size is the red thing in front of the yellow cylinder?



✗ What color is the shiny sphere?
✗ What is the color of the cube to the right of the brown thing?

Figure 10.2: Examples of **invalid** questions for images in the CLEVR universe. Even syntactically correct questions can be invalid for a variety of reasons such as referring to absent objects, incorrect object properties, invalid relationships in the scene or being ambiguous, *etc*.

There is also significant progress on building models for VQA using LSTMs with convolutional networks [172, 239], stacked attention networks [451], module networks [17, 181, 199], relational networks [357], and others [321]. LBA is independent of the backbone VQA model and can be used with any existing architecture.

Visual question generation (VQG) was recently proposed as an alternative to image captioning [258, 297]. Our work is related to VQG in the sense that we require the learner to generate questions about images, however, our objective in doing so is different. Whereas VQG focuses on asking questions that are relevant to the image content, LBA requires the learner to ask questions that are both relevant and informative to the learner when answered. A positive side effect is that LBA circumvents the difficulty of evaluating the quality of generated questions (which also hampers image captioning [16]), because the question-answering accuracy of our final model directly correlates with the quality of the questions asked.

Active learning (AL) involves a collection of unlabeled examples and a learner that selects which samples will be labeled by an oracle [210, 248, 364, 424]. Common selection criteria include entropy [202], boosting the margin for classifiers [5, 67] and expected informativeness [177]. Our setting is different from traditional AL settings in multiple ways. First, unlike AL where an agent selects the image to be labeled, in LBA the agent selects an image and *generates a question*. Second, instead of asking for a single image level label, our setting allows for richer questions about objects, relationships *etc*. for a single image. While [373] did use simple predefined template questions for AL, templates offer limited expressiveness and a rigid query structure. In our approach, questions are generated by a learned language model. Expressive language models, like those used in our work, are likely necessary for generalizing to real-world settings. However, they also introduce a new challenge: there are many ways to generate invalid questions, which the learner must learn to discard (see Figure 10.2).

Exploratory learning centers on settings in which an agent explores the environment to acquire supervision [133, 389]; it has been studied in the context of, among others, computer games and navigation [225, 317], multi-user games [277], inverse kinematics [28], and motion planning for humanoids [123]. Exploratory learning problems are generally framed with reinforcement learning in which the agent receives (delayed) rewards, which are used to learn a policy that

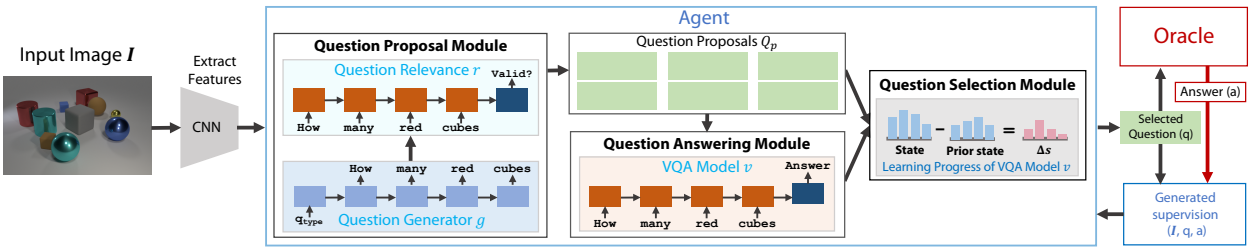


Figure 10.3: **Our approach to the learning-by-asking setting for VQA.** Given an image $\mathbf{I}$, the agent generates a diverse set of questions using a question generator g. It then filters out "irrelevant" questions using a relevance model r to produce a list of question proposals. The agent then answers its own questions using the VQA model v. With these predicted answers and its self-knowledge of past performance, it selects one question from the proposals to be answered by the oracle. The oracle provides answer-level supervision from which the agent learns to ask informative questions in subsequent iterations.

maximizes the expected rewards. A key difference in the LBA setting is that it does *not* have sparse delayed rewards. Contextual multi-armed bandits [45, 234, 245] are another class of reinforcement learning algorithms that more closely resemble the LBA setting. However, unlike bandits, online performance is irrelevant in our setting: our aim is not to minimize regret, but to minimize the error of the final VQA model produced by the learner.

10.2 Learning by Asking

We now formally introduce the learning-by-asking (LBA) setting. We denote an image by $\mathbf{I}$, and assume there exists a set of all possible questions $\mathcal{Q}$ and a set of all possible answers $\mathcal{A}$. At training time, the learner receives as input: (1) a training set of N images, $\mathcal{D}_{\text{train}} = \{\mathbf{I}_1, \ldots, \mathbf{I}_N\}$, sampled from some distribution $p_{\text{train}}(\mathbf{I})$; (2) access to an oracle $o(\mathbf{I}, q)$ that outputs an answer $a \in \mathcal{A}$ given a question $q \in \mathcal{Q}$ about image $\mathbf{I}$; and (3) a small bootstrap set of $(\mathbf{I}, q, a)$ tuples, denoted $\mathcal{B}_{\text{init}}$.

The learner receives a budget of B answers that it can request from the oracle. Using these B oracle consultations, the learner aims to construct a function $v(a|\mathbf{I}, q)$ that predicts a score for answer a to question q about image $\mathbf{I}$. The small bootstrap set is provided for the learner to initialize various model components; as we show in our experiments, training on $\mathcal{B}_{\text{init}}$ alone yields poor results.

The challenge of the LBA setting implies that, at training time, *the learner must decide which question to ask about an image* and the only supervision the oracle provides are the answers. As the number of oracle requests is constrained by a budget B, the learner must ask questions that maximize (in expectation) the learning signal from each image-question pair sent to the oracle.

At test time, we assume a standard VQA setting and evaluate models by their question-answering accuracy. The agent receives as input M pairs of images and questions

$$\mathcal{D}_{\text{test}} = \{(\mathbf{I}_{N+1}, q_{N+1}), \ldots, \mathbf{I}_{N+M}, q_{N+M})\},$$

sampled from a distribution $p_{\text{test}}(\mathbf{I}, q)$. The images in the test set are sampled from the same

distribution as those in the training set: $\sum_{q \in \mathcal{Q}} p_{\text{test}}(\mathbf{I}, q) = p_{\text{train}}(\mathbf{I})$. The agent's goal is to maximize the proportion of test questions that it answers correctly, that is, to maximize:

$$\frac{1}{M} \sum_{m=1}^{M} \mathbb{I}[\underset{a}{\text{argmax}}\, v(a|\mathbf{I}_{N+m}, q_{N+m}) = o(\mathbf{I}_{N+m}, q_{N+m})].$$

We make no assumptions on the marginal distribution over test questions, $p_{\text{test}}(q)$.

10.3 Approach

We propose a LBA agent built from three modules: (1) a **question proposal module** that generates a set of question proposals for an input image; (2) a **question answering module** (or VQA model) that predicts answers from $(\mathbf{I}, q)$ pairs; and (3) a **question selection module** that looks at both the answering module's state and the proposal module's questions to pick a single question to ask the oracle. After receiving the oracle's answer, the agent creates a tuple $(\mathbf{I}, q, a)$ that is used as the online learning signal for all three modules. Each of the modules is described in a separate subsection below; the interactions between them are illustrated in Figure 10.3.

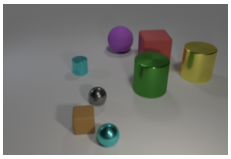
For the CLEVR universe, the **oracle** is a program interpreter that uses the ground-truth scene information to produce answers. As this oracle only understands questions in the form of programs (as opposed to natural language), our question proposal and answering modules both represent questions as programs. However, unlike [181, 199], we do *not* exploit prior knowledge of the CLEVR programming language in any of the modules; instead, it is treated as a simple means that is required to communicate with the oracle. Figure 10.4 shows a few example images, questions and programs from CLEVR.

When the LBA model asks an invalid question, the oracle returns a special answer indicating (1) that the question was invalid and (2) whether or not all the objects that appear in the question are present in the image.

10.3.1 Question Proposal Module

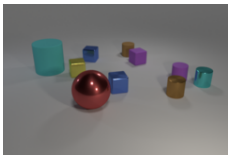
The question proposal module aims to generate a diverse set of questions (programs) that are relevant to a given image. To achieve this, we employ two subcomponents: (1) a **question generation model** g that produces questions $q_g \sim g(q|\mathbf{I})$; and (2) a **question relevance model** $r(\mathbf{I}, q_g)$ that predicts whether a generated question q_g is *relevant* to an image $\mathbf{I}$. Figure 10.2 shows examples of irrelevant questions that need to be filtered by r. The question generation and relevance models are used repeatedly to produce a set of question proposals, $\mathcal{Q}_p \subseteq \mathcal{Q}$.

Our **question generation model**, $g(q|\mathbf{I})$, is an image-captioning model that uses a LSTM conditioned on image features (first hidden input) to generate a question. To increase the diversity of generated questions, we also condition the LSTM on the "question type" while training [116] (we use the predefined question types or families from CLEVR). Specifically, we first sample a question type q_{type} uniformly at random and then sample a question from the LSTM using a beam



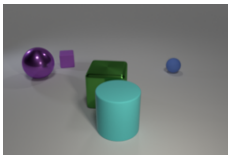
```
[
 {
  "function": "
      scene",
  "inputs": [],
  "value_inputs":
      []
 },
 {
  "function": "
      filter_size
      ",
  "inputs": [
   0
  ],
  "value_inputs":
      [
   "small"
  ]
 },
 {
  "function": "
      filter_materia
      ",
  "inputs": [
   1
  ],
  "value_inputs":
      [
   "rubber"
  ]
 },
 {
  "function": "
      unique",
  "inputs": [
   2
  ],
  "value_inputs":
      []
 },
 {
  "function": "
      query_color
      ",
  "inputs": [
   3
  ],
  "value_inputs":
      []
 }
]
```

Q: What color is the tiny rubber thing?
A: brown



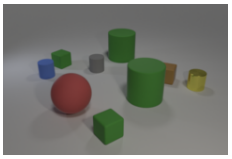
```
[
 {
  "function": "
      scene",
  "inputs": [],
  "value_inputs":
      []
 },
 {
  "function": "
      filter_shape
      ",
  "inputs": [
   0
  ],
  "value_inputs":
      [
   "sphere"
  ]
 },
 {
  "function": "
      unique",
  "inputs": [
   1
  ],
  "value_inputs":
      []
 },
 {
  "function": "
      query_color
      ",
  "inputs": [
   2
  ],
  "value_inputs":
      []
 }
]
```

Q: What color is the ball?
A: red



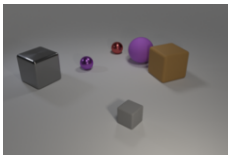
```
[
 {
  "function": "
      scene",
  "inputs": [],
  "value_inputs":
      []
 },
 {
  "function": "
      filter_size
      ",
  "inputs": [
   0
  ],
  "value_inputs":
      [
   "small"
  ]
 },
 {
  "function": "
      filter_shape
      ",
  "inputs": [
   1
  ],
  "value_inputs":
      [
   "sphere"
  ]
 },
 {
  "function": "
      unique",
  "inputs": [
   2
  ],
  "value_inputs":
      []
 },
 {
  "function": "
      query_color
      ",
  "inputs": [
   3
  ],
  "value_inputs":
      []
 }
]
```

Q: The small sphere is what color?
A: blue



```
[
 {
  "function": "
      scene",
  "inputs": [],
  "value_inputs":
      []
 },
 {
  "function": "
      filter_color
      ",
  "inputs": [
   0
  ],
  "value_inputs":
      [
   "yellow"
  ]
 },
 {
  "function": "
      unique",
  "inputs": [
   1
  ],
  "value_inputs":
      []
 },
 {
  "function": "
      query_material
      ",
  "inputs": [
   2
  ],
  "value_inputs":
      []
 }
]
```

Q: What is the yellow object made of?
A: metal



```
[
 {
  "function": "
      scene",
  "inputs": [],
  "value_inputs":
      []
 },
 {
  "function": "
      filter_size
      ",
  "inputs": [
   0
  ],
  "value_inputs":
      [
   "large"
  ]
 },
 {
  "function": "
      filter_color
      ",
  "inputs": [
   1
  ],
  "value_inputs":
      [
   "green"
  ]
 },
 {
  "function": "
      count",
  "inputs": [
   2
  ],
  "value_inputs":
      []
 }
]
```

Q: How many large green things are there?
A: 0

Figure 10.4: Examples of programs, questions and answers from the CLEVR dataset.

size of 1 and a sampling temperature of 1.3. For each image, we filter out all the questions that have been previously answered by the oracle.

Our **question relevance model**, $r(\mathbf{I}, q)$, takes the questions from the generator g as input and filters out irrelevant questions to construct a set of question proposals, $\mathcal{Q}_p$. The special answer provided by the oracle whenever an invalid question is asked (as described above) serves as the online learning signal for the relevance model. Specifically, the model is trained to predict (1) whether or not a image-question pair is valid and (2) whether or not all objects that are mentioned in the question are all present in the image. Questions for which both predictions are positive (*i.e.*, that are deemed by the relevance model to be valid and to contain only objects that appear in the image) are put in the question proposal set, $\mathcal{Q}_p$. We sample from the generator until we have 50 question proposals per image that are predicted to be valid by $r(\mathbf{I}, q)$.

10.3.2 Question Answering Module (VQA Model)

Our question answering module is a standard VQA model, $v(a|\mathbf{I}, q)$, that learns to predict the answer a given an image-question pair $(\mathbf{I}, q)$. The answering module is trained online using the supervision signal from the oracle.

A key requirement for selecting good questions to ask the oracle is the VQA model's capability to self-evaluate its current state. We capture the state of the VQA model at LBA round t by keeping track of the model's question-answering accuracy $s_t(a)$ per answer a on the training data obtained so far. The state captures information on *what the answering module already knows*; it is used by the question selection module.

10.3.3 Question Selection Module

The question selection module defines a policy, $\pi(\mathcal{Q}_p; \mathbf{I}, s_{1,\ldots,t})$, that selects the most informative question to ask the oracle from the set of question proposals $\mathcal{Q}_p$. To select an informative question, the question selection module uses the current state of the answering module (how well it is learning various concepts) and the difficulty of each of the question proposals. These quantities are obtained from the state $s_t(a)$ and the beliefs of the current VQA model, $v(a|\mathbf{I}, q)$ for an image-question pair, respectively.

The state $s_t(a)$ contains information about the current knowledge of the answering module. The difference in the state values at the current round, t, and a past round, $t - \Delta$, measures how fast the answering module is improving for each answer. Inspired by curriculum learning [28, 35, 226], we use this difference to select questions on which the answering module can improve the fastest. Specifically, we compute the the expected accuracy improvement under the answer distribution for each question $q_p \in \mathcal{Q}_p$:

$$h(q_p; \mathbf{I}, s_{1,\ldots,t}) = \sum_{a \in \mathcal{A}} v(a|\mathbf{I}, q_p) \left(\frac{s_t(a) - s_{t-\Delta}(a)}{s_t(a)} \right). \tag{10.1}$$

We use the expected accuracy improvement as an informativeness value that the learner uses to pick a question that helps it improve rapidly (thereby enforcing a curriculum). In particular, our selection policy, $\pi(\mathcal{Q}_p; \mathbf{I}, s_{1,\ldots,t})$, uses the informativeness scores to select the question to ask the oracle using an epsilon-greedy policy [398]. The greedy part of the selection policy is implemented via $\operatorname{argmax}_{q_p \in \mathcal{Q}_p} h(q_p; \mathbf{I}, s_{1,\ldots,t})$, and we set $\epsilon = 0.1$ to encourage exploration. Empirically, we find that our policy automatically discovers an easy-to-hard curriculum (see Figures 10.7 and 10.9). In all experiments, we set $\Delta = 20$; whenever $t < \Delta$, we set $s_{t-\Delta}(a) = 0$.

Interpretation as Learning Progress: The term Δs is a first order indicator of the improvement in the answering modules performance. This term can be interpreted as the learning progress [28] of the answering module. As equation 10.1 shows, the informativeness score h favors question proposals with a high learning progress. If an answer has low learning progress, then the informativeness for a question proposal yielding that answer will be low.

10.3.4 Training Phases

Our model is trained in three phases: (1) an initialization phase in which the generation, relevance, and VQA models (g, r and v) are pre-trained on a small bootstrap set, $\mathcal{B}_{\text{init}}$, of $(\mathbf{I}, q, a)$ tuples; (2) an online learning-by-asking (LBA) phase in which the model learns by interactively asking questions and updates r and v; and (3) an offline phase in which a new VQA model v_{offline} is trained from scratch on the union of the bootstrap set and all of the $(\mathbf{I}, q, a)$ tuples obtained by querying the oracle in the online LBA phase.

Online LBA training phase. At each step in the LBA phase (see Figure 10.3), the proposal module picks an image $\mathbf{I}$ from the training set $\mathcal{D}_{\text{train}}$ uniformly at random.[1] It then generates a set of relevant question proposals, $\mathcal{Q}_p$ for the image. The answering module tries to answer each question proposal. The selection module uses the state of the answering module along with the answer distributions obtained from evaluating the answering module to pick an informative question, q, from the question proposal set. This question is asked to the oracle o, which provides just the answer $a = o(\mathbf{I}, q)$ to generate a training example $(\mathbf{I}, q, a)$. This training example is used to perform a single gradient step on the parameters of the answering module v and the relevance model r. The language generation model g remains fixed because the oracle does not provide a direct learning signal for it. This process is repeated until the training budget of B oracle answer requests is exhausted.

Offline VQA training phase. We evaluate the quality of the asked questions by training a VQA model v_{offline} from scratch on the union of the bootstrap set, $\mathcal{B}_{\text{init}}$, and the $(\mathbf{I}, q, a)$ tuples generated in the LBA phase. We find that offline training of the VQA model leads to slightly improved question-answering accuracy and reduces variance.

10.3.5 Implementation Details

The LSTM in g has 512 hidden units. After a linear projection, the image features are fed as its first hidden state. We input a discrete variable representing the question type as the first token into the

[1] A more sophisticated image selection policy may accelerate learning. We did not explore this in our study.

LSTM before starting generation. Following [199], we use a prefix-tree program representation for the questions.

We implement the relevance model, r, and the VQA model, v, using the stacked attention network architecture [451] using the implementation of [199]. The only modification we make is to concatenate the spatial coordinates to the image features before computing attention as in [357]. We do not share weights between r and v.

To generate the invalid pairs $(\mathbf{I}, q)$ for bootstrapping the relevance model, we permute the pairs from the bootstrap set $\mathcal{B}_{\text{init}}$ and assume that all such permuted pairs are invalid. This provides us with the $e_{\text{valid}} = 0$ label to train r. As the bootstrap set does not have e_{present} labels, we only use these labels to train r during the online LBA phase.

Our models use image features from a ResNet-101 [166] pre-trained on ImageNet [351], in particular, from the `conv4_23` layer of that network. Additional implementation details on learning rate schedules, embedding sizes, *etc*. are presented in the supplementary material.

10.4 Experiments

Datasets. We evaluate our LBA approach in the CLEVR universe [198], which provides a training set (`train`) with 70k images and 700k $(\mathbf{I}, q, a)$ tuples. We use 70k of these tuples as our bootstrap set, $\mathcal{B}_{\text{init}}$. We evaluate the quality of the data collected by LBA by measuring the question-answering accuracy of the final VQA model, v_{offline}, on the CLEVR validation (`val`) [198] set. Because CLEVR `train` and `val` have identical answer and question-type distributions, which gives models trained on CLEVR `train` an inherent advantage. Thus, we also measure question-answering accuracy on the CLEVR-Humans [199] dataset, which has a different distribution; see Figure 10.10.

Models. Unless stated otherwise, we use the stacked attention model as the answering module v and evaluate three different choices for the final offline VQA model v_{offline}:

CNN+LSTM encodes the image using a CNN, the question using an LSTM, and predicts answers using an MLP.

CNN+LSTM+SA extends CNN+LSTM with the stacked attention (SA) model [451] described in Section 10.3.2. This is the same as our default answering module v.

FiLM [321] is a CNN that uses question features from a GRU [59] to modulate the image features in each CNN layer.

Unless stated otherwise, we use CNN+LSTM+SA models in all ablation analysis experiments, even though it has lower VQA performance than FiLM, because it trains much faster (6 hours *vs*. 3 days). For all v_{offline} models, we use the training hyperparameters from their respective chapters.

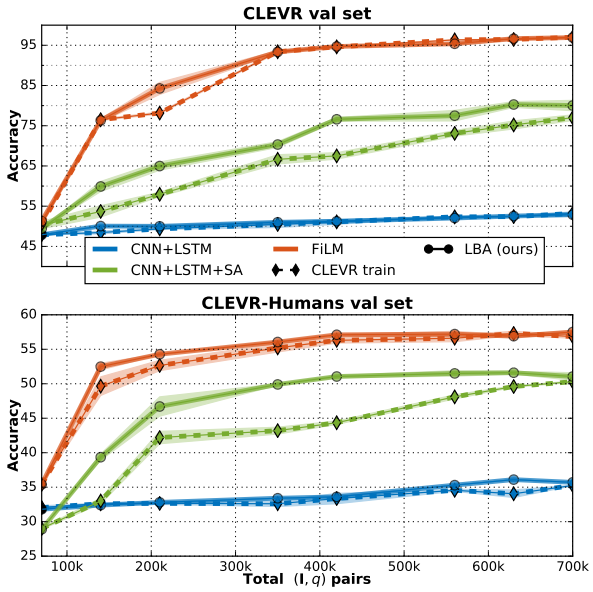


Figure 10.5: **Top:** CLEVR `val` accuracy for VQA models trained on CLEVR `train` (diamonds) *vs*. LBA-generated data (circles). **Bottom:** Accuracy on CLEVR-Humans for the same set of models.

10.4.1 Quality of LBA-Generated Questions

In Figure 10.5, we compare the quality of the LBA-generated questions to CLEVR `train` by measuring the question-answering accuracy of VQA models trained on both datasets. The figure shows (top) CLEVR `val` accuracy and (bottom) CLEVR-Humans accuracy. Questions in CLEVR-Humans are given in English. To apply our VQA models, which understand input in the CLEVR programming language, we automatically translate the CLEVR-Humans questions using the English-to-CLEVR-program model from [199]. The translation does not affect the accuracy of the VQA models (see Section 10.4.8). From these plots, we draw four observations.

(1) Using the bootstrap set alone (leftmost point) yields poor accuracy and LBA provides a significant learning signal.

(2) The quality of the LBA-generated training data is at least as good as that of the CLEVR `train`. This is an impressive result given that CLEVR `train` has the dual advantage of matching the distribution of CLEVR `val` and being human curated for training VQA models. Despite these advantages, LBA matches and sometimes surpasses its performance. More importantly, LBA shows better generalization on CLEVR-Humans which has a different answer distribution (see Figure 10.10).

(3) LBA data is sometimes more sample efficient than CLEVR `train`: for instance, on both CLEVR `val` and CLEVR-Humans. The CNN+LSTM+SA model only requires 60% of $(\mathbf{I}, q, a)$ LBA tuples to achieve the accuracy of the same model trained on all of CLEVR `train`.

(4) Finally, we also observe that our LBA agents have low variance at each sampled point during training. The shaded error bars show one standard deviation computed from 5 independent runs

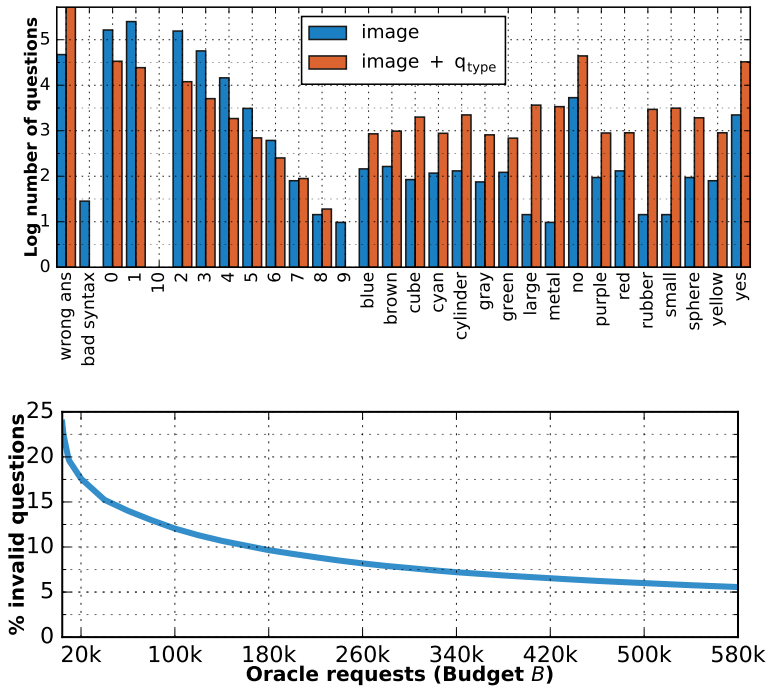


Figure 10.6: **Top:** Histogram of answers to questions generated by g with and without question-type conditioning. **Bottom:** Percentage of invalid questions sent to the oracle.

Generator g	Relevance r	Budget B 0k	70k	210k	350k	560k	630k
$\mathbf{I}$	None	49.4	43.2	45.4	49.8	52.9	54.7
$\mathbf{I} + \mathrm{q}_{\mathrm{type}}$	None	49.4	46.3	49.5	58.7	60.5	63.4
$\mathbf{I} + \mathrm{q}_{\mathrm{type}}, \tau = 0.3$	Ours	49.4	60.6	67.4	70.2	70.8	70.1
$\mathbf{I} + \mathrm{q}_{\mathrm{type}}, \tau = 0.7$	Ours	49.4	60.2	70.5	76.7	77.5	77.6
$\mathbf{I} + \mathrm{q}_{\mathrm{type}}, \tau = 1.3$	Ours	49.4	60.3	71.4	76.9	79.8	78.2
$\mathbf{I} + \mathrm{q}_{\mathrm{type}}$	Perfect	49.4	67.7	75.7	80.0	81.2	81.1

Table 10.1: CLEVR `val` accuracy for six budgets B. We condition the generator on the image ($\mathbf{I}$) or on the image and the question type ($\mathbf{I} + \mathrm{q}_{\mathrm{type}}$), vary the generator sampling temperatures τ, and use three different relevance models. We re-run the LBA pipeline for each of these settings.

using different random seeds. This is an important property for drawing meaningful conclusions from interactive training environments (*c.f.*, [170]).

Qualitative results. Figure 10.7 shows five samples from the LBA-generated data at various iterations t. They provide insight into the curriculum discovered by our LBA agent. Initially, the model asks simple questions about colors (row 1) and shapes (row 2). It also makes basic mistakes (rightmost column of rows 1 and 2). As the answering module v improves, the selection policy π asks more complex questions about spatial relationships and counts (rows 3 and 4).

v_{offline} Model	Budget B 0k	70k	210k	350k	560k	630k
CNN+LSTM	47.1	48.0	49.2	49.1	52.3	52.7
CNN+LSTM+SA	49.4	63.9	68.1	76.1	78.4	82.3
FiLM	51.2	76.2	92.9	94.8	95.2	97.3

Table 10.2: CLEVR `val` accuracy for three v_{offline} models when FiLM is used as the online answering module v.

10.4.2 Analysis: Question Proposal Module

Analyzing the generator g. We evaluate the diversity of the generated questions by looking at the distribution of corresponding answers. In Figure 10.6 (top) we use the final LBA model to generate 10 questions for each image in the training set. We plot the histogram of the answers to these questions for generators with and without "question type" conditioning. The histogram shows that conditioning the generator g on question type leads to better coverage of the answer space. We also note that about 4% of the generated questions have invalid programming language syntax.

We observe in the top two rows of Table 10.1 that the increased question diversity translates into improved question-answering accuracy. Diversity is also controlled by the sampling temperature, τ, used in g. Rows 3-5 show that a lower temperature, which gives less diverse question proposals, negatively impacts final accuracy.

Analyzing the relevance model r. Figure 10.6 (bottom) displays the percentage of invalid questions sent to the oracle at different time steps during online LBA training. The invalid question rate decreases during training from 25% to 5%, even though question complexity appears to be increasing (Figure 10.7). This result indicates that the relevance model r improves significantly during training.

We can also decouple the effect of the relevance model r from the rest of our setup by replacing it with a "perfect" relevance model (the oracle) that flawlessly filters all invalid questions. Table 10.1 (row 6) shows that the accuracy and sample efficiency differences between the "perfect" relevance model and our relevance model are small, which suggests our model performs well.

10.4.3 Analysis: Question Answering Module

Thus far we have tested our policy π with only one type of answering module v, CNN+LSTM+SA. Now, we verify that π works with other choices by implementing v as the FiLM model and rerunning LBA. As in Section 10.4.1, we evaluate the LBA-generated questions by training the three v_{offline} models. The results in Table 10.2 suggest that our selection policy generalizes to a new choice of the online VQA model v used in the answering module.

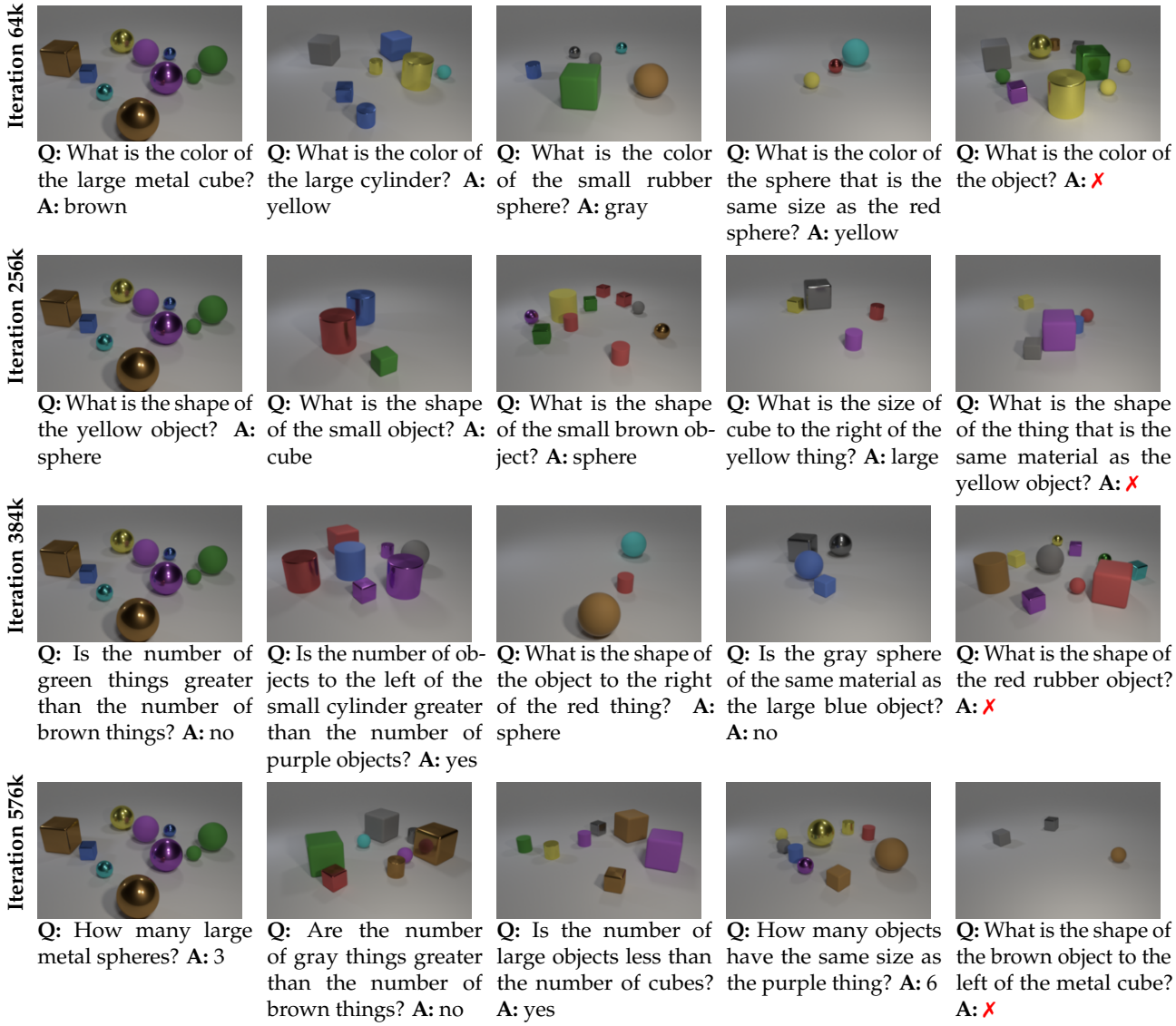


Figure 10.7: Example questions asked by our LBA agent at different iterations (manually translated from programs to English). Our agent asks increasingly sophisticated questions as training progresses — starting with simple color questions and moving on to shape and count questions. We also see that the invalid questions (right column) become increasingly complex.

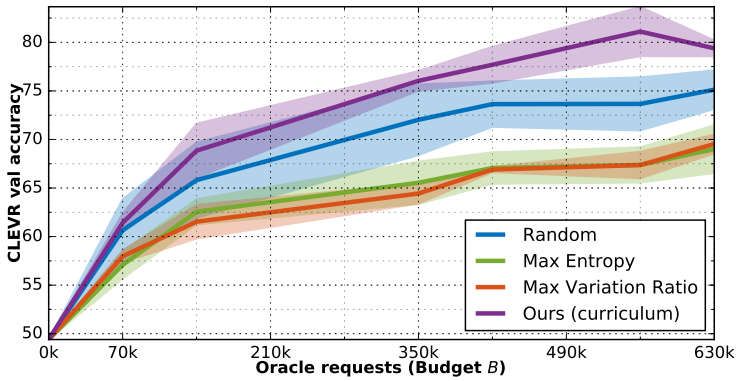


Figure 10.8: Accuracy of CNN+LSTM+SA trained using LBA with four different policies for selecting question proposals (Sec 10.3.3). Our selection policy is more sample efficient.

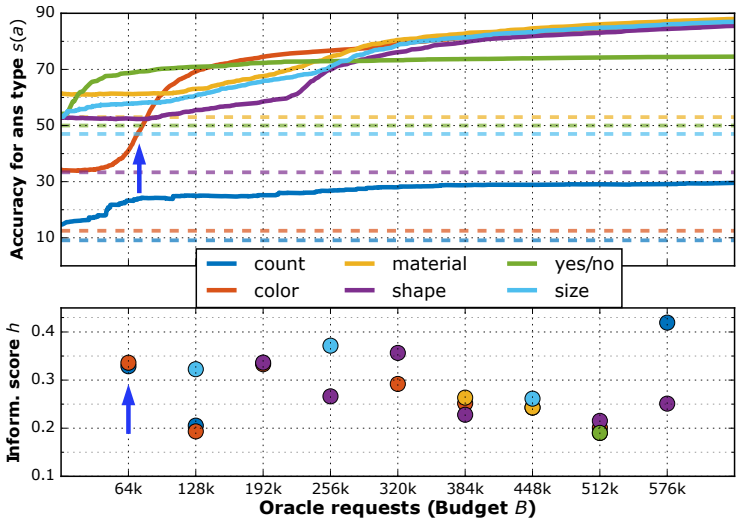


Figure 10.9: **Top:** Accuracy during training (solid lines) and chance level (dashed lines) per answer type. **Bottom:** Normalized informative scores per answer type, averaged over 10k questions. See Section 10.4.4 for details.

10.4.4 Analysis: Question Selection Module

To investigate the role of the selection policy in LBA, we compare four alternatives: (1) random selection from the question proposals; (2) using the prediction entropy of the answering module v for each proposal after four forward passes with dropout (like in [362]); (3) using the variation ratio [124] of the prediction; and (4) our curriculum policy from Section 10.3.3. We run LBA training with five different random seeds and report the mean accuracy and stdev of a CNN+LSTM+SA model for each selection policy in Figure 10.8. In line with results from prior work [362], the entropy-based policies performs worse than random selection. By contrast, our curriculum policy substantially outperforms random selection of questions. Figure 10.9 plots the normalized informativeness score h (Equation 10.1) and the training question-answering accuracy per answer type $s(a)$. These plots also provide insight into the behavior of the curriculum selection policy, π. Specifically, we observe a delayed pattern: a peak in the the informativeness score (blue arrow) for an answer

type is followed by an uptick in the accuracy (blue arrow) on that answer type. We also observe that the policy's informativeness score suggests an easy-to-hard ordering of questions: initially (after 64k requests), the selection policy prefers asking the easier `color` questions, but it gradually moves on to `size` and `shape` questions and, eventually, to the difficult `count` questions. We emphasize that this easy-to-hard curriculum is learned automatically without any extra supervision.

10.4.5 Varying the Size of the Bootstrap Data

We vary the size of the bootstrap set $\mathcal{B}_{\text{init}}$ used for initializing the g, r, v models and analyze its effect on the LBA generated data. In Table 10.3 we show the accuracy of the final v_{offline} model on CLEVR `val`. A smaller bootstrap set results in reduced performance. We also see that with less than 5% (rows 1 and 2) of the CLEVR training dataset as our bootstrap set, LBA asks questions that can match the performance using the entire CLEVR training set. Empirically, we observed that the generator g performs well on smaller bootstrap sets. However, the relevance model r needs enough valid and invalid (permuted) $(\mathbf{I}, q, a)$ tuples in the bootstrap set to filter irrelevant question proposals. As a result, a smaller bootstrap set affects the sample efficiency of LBA.

	Budget B						
$\lvert\mathcal{B}_{\text{init}}\rvert$	0k	70k	140k	210k	350k	560k	630k
20k	48.2	56.4	63.5	66.9	72.6	75.8	76.2
35k	48.8	58.6	64.3	68.7	74.9	76.1	76.3
70k	49.4	61.1	67.6	72.8	78.0	78.2	79.1

Table 10.3: Accuracy on CLEVR validation data at different budgets B as a function of the bootstrap set size, $|\mathcal{B}_{\text{init}}|$.

10.4.6 More Questions or More Images?

We investigate what is more important for learning: the number of images or the number of questions. Specifically, we vary the size of the image set $\mathcal{D}_{\text{train}}$ used for the LBA setting. The results in Table 10.4 show that our model can achieve similar results to using the full image set using only 49k of the 70k images. Using 35k images, our model achieves the same accuracy as a model trained on the full CLEVR training set (which has 70k images).

10.4.7 Hyperparameters for Models

Generator g: We use 512 hidden units in the LSTM. Each program token is first embedded in a 32-dimensional space before being input to the LSTM. The token embedding is learned jointly with the rest of the model. The generator is trained on the bootstrap set $\mathcal{B}_{\text{init}}$ to generate a question q from the image $\mathbf{I}$ by minimizing the cross-entropy loss for generating the question sequence. The question type is a discrete variable that can take one of 90 possible values (the CLEVR datasets defines 90 question families [198]). The image features are input as the first hidden input. We perform spatial max-pooling on the $1024 \times 14 \times 14$ image features from an ImageNet [351] pre-trained ResNet-101 [166] `conv4_23` layer to get 1024 channel features. We then project the features down to 512 dimensions using a linear projection (learned jointly with the model). The model is optimized using SGD with a learning rate of 1.0 and momentum of 0.9 with gradients clipped at

	Budget B						
Number of images	0k	70k	140k	210k	350k	560k	630k
1k	49.4	53.2	55.8	57.4	61.1	62.5	64.9
2k	49.4	53.9	57.7	65.1	66.0	68.4	69.7
5k	49.4	56.0	62.8	65.4	67.9	67.6	68.3
7k	49.4	54.2	60.1	66.4	67.5	70.6	71.3
14k	49.4	56.9	60.2	65.8	68.5	69.5	71.5
21k	49.4	56.8	61.5	68.1	68.7	69.8	73.4
35k	49.4	59.7	61.6	70.6	70.8	74.3	76.6
49k	49.4	59.5	66.9	72.7	76.7	78.1	77.2
70k	49.4	61.1	67.6	72.8	78.0	78.2	79.1
CLEVR train set (70k)	49.4	55.1	57.5	65.6	72.4	74.8	77.2

Table 10.4: Accuracy on CLEVR validation data during training as a function of the number of images used.

maximum L_2-norm of 5.0. We use a mini-batch size of 64 and decay the learning rate by 0.25 every $5,000$ iterations. The model is trained for a total of $20,000$ iterations.

Relevance model r: We use the stacked-attention network implementation from [451] in our experiments. We use Adam [215] to minimize the cross-entropy loss of this model, using a fixed learning rate of $5e-4$. We use L_2 weight decay of $4e-5$ as a regularizer. The model uses image features from an ImageNet [351] pre-trained ResNet-101 `conv4_23` layer (1024 channels). Following [357], we concatenate spatial coordinates (two channels x and y) to these features to finally get 1026 channel image features of spatial resolution 14×14.

VQA model v: The stacked-attention network serves as the default choice for v. We use the same hyperparameters as in the relevance model r, but do not share weights between v and r.

Details on Oracle: We use the oracle as implemented in [198] with the modification that it returns the e_{valid} and e_{present} labels in addition to the valid answers. The oracle uses the ground-truth scene graph and relationships provided in CLEVR to answer questions about an image. It takes the program as input and provides the answer. It determines invalid questions as those that do not successfully execute on the scene graph.

10.4.8 Translation for CLEVR Humans

In Section 5.1 of the main chapter, we evaluate our models on the CLEVR Humans set. As our models v_{offline} are trained using programs, we translate the natural language in the CLEVR Humans set using the language-to-program module provided by [199]. In Table 10.5, we show that such a translation does not affect the accuracy of the models. To do so, we train models on the natural language questions in CLEVR (rather than on the question programs), and evaluate the resulting models on CLEVR-Humans. We compare them with models trained on the CLEVR question programs (like in the main paper), which we evaluate on the translated [199]. The results in the table show that both models perform similarly and, therefore, which shows that the language-to-program translation does not impact question-answering accuracy.

Language	Model	Accuracy
NLP	CNN+LSTM+SA	50.1
Programs	CNN+LSTM+SA	50.2
NLP	FiLM	56.4
Program	FiLM	56.3

Table 10.5: Accuracy of stacked attention networks (CNN+LSTM+SA) and FiLM on the CLEVR Humans validation set. We show that testing the program on either natural language or a program translated version gives similar accuracy.

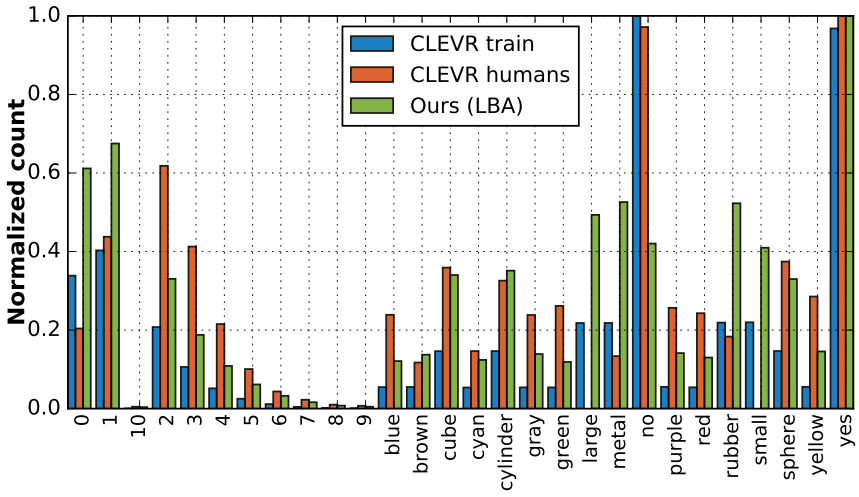


Figure 10.10: Answer distribution of CLEVR `train`, LBA-generated data, and the CLEVR-Humans dataset.

10.5 Discussion

This chapter introduces the learning-by-asking (LBA) paradigm and proposes a model in this setting. LBA moves away from traditional *passively* supervised settings where human annotators provide the training data to an *interactive* setting where the learner seeks out the supervision it needs. While passive supervision has driven progress in visual recognition, it does not appear well suited for general AI tasks such as visual question answering (VQA). Curating large amounts of diverse data which generalizes to a wide variety of questions is a difficult task. Our results suggest that interactive settings such as LBA may help learning with higher sample efficiency. Such high sample efficiency is crucial as we move to complex visual understanding tasks.

An important property of LBA is that it does not tie the distribution of questions and answers seen at training time to the distribution at test time. This more closely resembles the real-world deployment of VQA systems where the distribution of user-posed questions to the system is unknown and difficult to characterize beforehand [43]. The CLEVR-Humans distribution in Figure 10.10 is an example of this. This issue poses clear directions for future work [42]: we need to develop VQA models that are less sensitive to distributional variations at test time; and not evaluate them under a single test distribution (as in current VQA benchmarks).

A second major direction for future work is to develop a "real-world" version of a LBA system in which (1) CLEVR images are replaced by natural images and (2) the oracle is replaced by a

human annotator. Relative to our current approach, several innovations are required to achieve this goal. Most importantly, it requires the design of an effective mode of communication between the learner and the human "oracle". In our approach, the learner uses a simple programming language to query the oracle. A real-world LBA system needs to communicate with humans using diverse natural language. The efficiency of LBA learners may be further improved by letting the oracle return privileged information that does not just answer an image-question pair, but that also explains *why* this is the right or wrong answer [422].

Chapter 11

Conclusion

A conclusion is the place where you get tired of thinking.

MARTIN HENRY FISCHER in *Encore: A Continuing Anthology*, 1945

In this thesis, we explored the challenge of visual learning with minimal human supervision. We believe that the techniques highlighted in this thesis are crucial in order to scale learning beyond typical categories in fully supervised datasets. As we make progress on pattern recognition tasks and move more towards AI-complete tasks, we need models that can learn in a sample efficient manner, self-supervise and learn interactively. An additional and often competing requirement, as vision systems become more ubiquitous, is that they should be more computationally efficient - scaling to multiple tasks on cheap hardware such as edge devices (webcams *etc*.). We studied these challenges in the context of visual data (both images and video) leveraging tools from a variety of disciplines such as computer vision, machine learning, natural language processing, psycholinguistics and computer systems. Our interdisciplinary solutions have been published at conferences in computer vision, natural language processing, and computer systems.

We now summarize the contributions of this thesis. In Part I we used the structure in data to learn better object detectors (Chapter 2) and powerful feature representations in a self-supervised manner (Chapter 3). Part II used structure in the labels and tasks to learn from noisy labeled images on the web (Chapter 4) and how to learn better multi-task architectures (Chapter 5). We addressed the practical applications of these systems in the context of edge computing becoming necessary and dominant. We showed how to make multi-task and multi-tenant systems practical for inference on edge devices (Chapter 6). In Part III we showed that the structure in classifier space can be used to easily synthesize data (Chapter 7) and compose classifiers for zero-shot recognition(Chapter 8). Finally, in Part IV we showed the power of interactive learners that can ask questions (Chapter 9) and learn in a sample efficient manner (Chapter 10).

Bibliography

[1] Tensorflow Serving. https://www.tensorflow.org/serving/. 6.7.1, 6.8

[2] Mainstream: Dynamic stem-sharing for multi-tenant video processing. In *USENIX Annual Technical Conference (USENIX ATC 18)*, Boston, MA, 2018. USENIX Association. URL https://www.usenix.org/conference/atc18/presentation/jiang. 5

[3] Abrar H Abdulnabi, Gang Wang, Jiwen Lu, and Kui Jia. Multi-task CNN model for attribute prediction. *IEEE Multimedia*, 17, 2015. 5.1, 5.5.1

[4] Moshe Abeles, Markus Diesmann, Tamar Flash, Theo Geisel, Michael Herrmann, and Mina Teicher. Compositionality in neural control: an interdisciplinary study of scribbling movements in primates. *Frontiers in computational neuroscience*, 7, 2013. 8.2.1

[5] Yotam Abramson and Yoav Freund. Active learning for visual object recognition. *Technical report, UCSD*, 2004. 10.1

[6] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. YouTube-8M: A large-scale video classification benchmark. *CoRR*, abs/1609.08675, 2016. URL http://arxiv.org/abs/1609.08675. 6.1

[7] Deepak Agarwal, Bo Long, Jonathan Traupman, Doris Xin, and Liang Zhang. Laser: A scalable response prediction platform for online advertising. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14. ACM, 2014. URL http://doi.acm.org/10.1145/2556195.2556252. 6.8

[8] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *ICCV*, 2015. 3.1

[9] Karim Ahmed and Lorenzo Torresani. Branchconnect: Large-scale visual recognition with learned branch connections. *CoRR*, abs/1704.06010, 2017. URL http://arxiv.org/abs/1704.06010. 6.8

[10] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for attribute-based classification. In *CVPR*, 2013. 8.1

[11] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *CVPR*, 2010. 2.2.3

[12] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. Word spotting and recognition with embedded attributes. *TPAMI*, 36(12), 2014. 8.1

[13] Yonatan Amit, Michael Fink, Nathan Srebro, and Shimon Ullman. Uncovering shared structures in multiclass classification. In *ICML*, 2007. 5.1

[14] Phil Ammirato, Patrick Poirson, Eunbyung Park, Jana Kosecka, and Alexander C. Berg. A dataset for developing and benchmarking active vision. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017. 7.1, 7.2, 7.7, 7.5, 7.9, 7.5.2

[15] Viktoriia Sharmanska Anastasia Pentina and Christoph H. Lampert. Curriculum learning of multiple tasks. In *CVPR*, 2015. 5.1

[16] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. SPICE: Semantic propositional image caption evaluation. In *ECCV*, 2016. 10.1

[17] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. In *NAACL*, 2016. 10.1

[18] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. In *NAACL*, 2016. 8.1

[19] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *CVPR*, 2016. 8.1

[20] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *CVPR*, June 2014. 3, 3.1, 3.5

[21] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 9.1

[22] Lisa Anne Hendricks, Subhashini Venugopalan, Marcus Rohrbach, Raymond Mooney, Kate Saenko, and Trevor Darrell. Deep compositional captioning: Describing novel object categories without paired training data. In *CVPR*, 2016. 8.1

[23] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. VQA: Visual question answering. In *CVPR*, 2015. 10, 10.1

[24] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual question answering. In *International Conference on Computer Vision (ICCV)*, 2015. 9.1, 9.2.1

[25] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *JMLR*, 73, 2008. 5.1, 5.2.3

[26] Shai Avidan. Ensemble tracking. In *CVPR*, 2005. 2.1

[27] Rosalind Baker, Matthew Dexter, Tom E Hardwicke, Aimee Goldstone, and Zoe Kourtzi. Learning to predict: exposure to temporal sequences facilitates prediction of future events. *Vision research*, 99:124–133, 2014. 3

[28] Adrien Baranes and Pierre-Yves Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 2013. 10.1, 10.3.3, 10.3.3

[29] Kobus Barnard and David Forsyth. Learning the semantics of words and pictures. In *ICCV*, 2001. 4.1

[30] Jonathan T Barron and Ben Poole. The fast bilateral solver. In *European Conference on Computer Vision*, pages 617–632. Springer International Publishing, 2016. 7.4.1

[31] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008. 7.1

[32] Lee Becker, Sumit Basu, and Lucy Vanderwende. Mind the gap: Learning to choose gaps for question generation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 742–751, Montréal, Canada, June 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N12-1092. 9.1

[33] Eyal Beigman and Beata Beigman Klebanov. Learning with annotation noise. In *ACL-IJCNLP*, 2009. 4.1

[34] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *NIPS*, 2007. 3.1

[35] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*. ACM, 2009. 10, 10.3.3

[36] Yoshua Bengio, Eric Thibodeau-Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop. *arXiv preprint arXiv:1306.1091*, 2013. 3.1

[37] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *TPAMI*, 2011. 2.1, 2.2.3

[38] Alexander C Berg, Tamara L Berg, Hal Daume III, Jesse Dodge, Amit Goyal, Xufeng Han, Alyssa Mensch, Margaret Mitchell, Aneesh Sood, Karl Stratos, et al. Understanding and predicting importance in images. In *CVPR*, 2012. 4, 4.1, 4.3.1, 4.3.3

[39] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94, 1987. 8.1

[40] Ali Borji, Dicky N Sihite, and Laurent Itti. What stands out in a scene? a study of human explicit saliency judgment. *Vision research*, 91, 2013. 4.1

[41] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Representing shape with a spatial pyramid kernel. In *CVPR*, 2007. 2.3, 2.3.2

[42] Leon Bottou. Two Big Challenges in Machine Learning. http://leon.bottou.org/talks/2challenges. Accessed: Nov 15, 2017. 10.5

[43] Leon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X. Charles, D. Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *JMLR*, 2013. 10.5

[44] Jean-Yves Bouget. Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm, 2000. 2.2.3

[45] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 2012. 10.1

[46] Rich Caruana. Multitask learning. *Machine learning*, 28, 1997. 5.1, 5.5.3

[47] Rich Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998. 8.1

[48] Rich Caruna. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998. 6.8

[49] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 7.1

[50] Yu-Wei Chao, Zhan Wang, Yugeng He, Jiaxuan Wang, and Jia Deng. HICO: A benchmark for recognizing human-object interactions in images. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015. 9.1

[51] Olivier Chapelle. Modeling delayed feedback in display advertising. In *KDD*, 2014. 4.2.2

[52] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014. 3.3

[53] Ding-Jie Chen, Hwann-Tzong Chen, and Long-Wen Chang. Video object cosegmentation. In *ACM MM*, 2012. 2.1

[54] Jianhui Chen, Jiayu Zhou, and Jieping Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *SIGKDD*, 2011. 5.5.1

[55] Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15. JMLR.org, 2015. URL http://dl.acm.org/citation.cfm?id=3045118.3045361. 6.8

[56] Wenzheng Chen, Huan Wang, Yangyan Li, Hao Su, Zhenhua Wang, Changhe Tu, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. Synthesizing training images for boosting human 3d pose estimation. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 479–488. IEEE, 2016. 7, 7.1

[57] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. NEIL: Extracting Visual Knowledge from Web Data. In *ICCV*, 2013. 2, 2.1

[58] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015. 4, 4.3, 4.3.1, 4.3.1

[59] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014. 10.4

[60] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 9.3.1

[61] Jonghyun Choi, M. Rastegari, A. Farhadi, and L.S. Davis. Adding unlabeled samples to categories by learned attributes. In *CVPR*, 2013. 2, 2.1

[62] Jonghyun Choi, Mohammad Rastegari, Ali Farhadi, and Larry S Davis. Adding unlabeled samples to categories by learned attributes. In *CVPR*, 2013. 8.1

[63] François Chollet et al. Keras. https://keras.io, 2015. 6.6

[64] Axel Cleeremans. *Mechanisms of implicit learning: Connectionist models of sequence processing*. MIT press, 1993. 3

[65] Axel Cleeremans and James L McClelland. Learning the structure of event sequences. *Journal of Experimental Psychology: General*, 120(3):235, 1991. 3

[66] Alvaro Collet, Manuel Martinez, and Siddhartha S Srinivasa. The moped framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research*, 30(10):1284–1306, 2011. 7.1

[67] Brendan Collins, Jia Deng, Kai Li, and Li Fei-Fei. Towards scalable dataset construction: An active learning approach. *ECCV*, 2008. 10.1

[68] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, 2008. 5.1

[69] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *JMLR*, 12, 2011. 5.1

[70] Nikolaus Correll, Kostas E Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M Romano, and Peter R Wurman. Lessons from the amazon picking challenge. *arXiv preprint arXiv:1601.05484*, 2016. 7.1

[71] Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *Advances in Neural Information Processing Systems*, 2016. 6.8

[72] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. 2015. 6.8

[73] D. Crankshaw, P. Bailis, J. E. Gonzalez, H. Li, Z. Zhang, M. J. Franklin, A. Ghodsi, and M. I. Jordan. The missing piece in complex analytics: Low latency, scalable model management and serving with Velox. In *CIDR*, 2015. 6.8

[74] Daniel Crankshaw, Xin Wang, Guilio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. Clipper: A low-latency online prediction serving system. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, Boston, MA, 2017. USENIX Association. ISBN 978-1-931971-37-9. URL https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/crankshaw. 6, 6.7.1, 6.8

[75] Dengxin Dai and L. Van Gool. Ensemble projection for semi-supervised image classification. In *ICCV*, 2013. 2.1

[76] Qieyun Dai and D. Hoiem. Learning to localize detected objects. In *CVPR*, 2012. 2.3.1

[77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977. 4.2.2

[78] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 6.1, 6.1

[79] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2, 5.4, 5.4.3

[80] Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven, and Hartwig Adam. Large-scale object classification using label relation graphs. In *ECCV*. Springer, 2014. 8.1

[81] Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*, 2014. 9.2.3, 9.4.2

[82] Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. Language models for image captioning: The quirks and what works. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 100–105, Beijing, China, July 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-2017. 9.3.1, 9.3.2

[83] Santosh Divvala, Ali Farhadi, and Carlos Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *CVPR*, 2014. 2, 2.1

[84] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Mid-level visual element discovery as discriminative mode seeking. In *NIPS*, 2013. 3.1

[85] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 3, 3.1

[86] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013. 5.1

[87] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 4.3.5

[88] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12, 2011. 3.5

[89] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection. In *ICCV*, 2017. 4, 6

[90] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 5.1

[91] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014. 6.1

[92] Mohamed Elhoseiny, Babak Saleh, and Ahmed Elgammal. Write a classifier: Zero-shot learning using purely textual descriptions. In *ICCV*, 2013. 8.1, 8.3.1, 8.4.1

[93] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *SIGKDD*, 2008. 2, 4.1

[94] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007), . 4, 4.3.1, 5.0.1, 5.1, 5.5, 5.5.4, 7.2, 7.4.1, 7.4.2, 8.3.1

[95] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007), . 2, 2.1, 2.3.1, 2.3.1

[96] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010. ISSN 0920-5691. doi: 10.1007/s11263-009-0275-4. URL http://dx.doi.org/10.1007/s11263-009-0275-4. 9.1

[97] A Evgeniou and Massimiliano Pontil. Multi-task feature learning. *NIPS*, 19:41, 2007. 5.1

[98] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *SIGKDD*, 2004. 5.1, 5.5.1, 5.5.3

[99] Alon Faktor and Michal Irani. "clustering by composition"–unsupervised discovery of image categories. In *ECCV*. 2012. 3.1

[100] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *JMLR*, 2008. 8.3.1

[101] Hao Fang, Saurabh Gupta, Forrest N. Iandola, Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. From captions to visual concepts and back. *CoRR*, abs/1411.4952, 2014. URL http://arxiv.org/abs/1411.4952. 3, 9.3.1

[102] Hao Fang, Saurabh Gupta, Forrest N. Iandola, Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. From captions to visual concepts and back. In *CVPR*, 2015. 4.1, 4.3.1, 4.3.1, 4.2, 4.3, 4.3.2, 4.4, 4.5

[103] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *CVPR*, 2009. 8, 8.1

[104] Ali Farhadi, Ian Endres, and Derek Hoiem. Attribute-centric recognition for cross-category generalization. In *CVPR*, 2010. 5.1

[105] Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. Every picture tells a story: Generating sentences from images. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 15–29, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15560-X, 978-3-642-15560-4. URL http://dl.acm.org/citation.cfm?id=1888089.1888092. 9.3.2

[106] Alireza Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *CVPR*, 2009. 5.0.1, 5.1, 5.5, 5.6, 5.5.4

[107] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image analysis*. Springer, 2003. 3.2.2

[108] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *TPAMI*, 28, 2006. 8.1

[109] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008. 8.1

[110] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 7

[111] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2010. 2.1, 2.1, 2.3.1, 5.1

[112] Andras Ferencz, Erik G Learned-Miller, and Jitendra Malik. Building a classification cascade for visual identification from one example. In *ICCV*, volume 1, 2005. 8.1

[113] Rob Fergus, Yair Weiss, and Antonio Torralba. Semi-supervised learning in gigantic image collections. In *NIPS*, 2009. 2.1, 2.3.2, 4.1

[114] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Object detection by contour segment networks. In *European conference on computer vision*, pages 14–28. Springer, 2006. 7.1

[115] Francis Ferraro, Nasrin Mostafazadeh, Ting-Hao Huang, Lucy Vanderwende, Jacob Devlin, Michel Galley, and Margaret Mitchell. A survey of current datasets for vision and language research. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 207–213, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL `http://aclweb.org/anthology/D15-1021`. 9.2.1

[116] Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, C Lawrence Zitnick, et al. Visual storytelling. In *NAACL*, 2016. 10.3.1

[117] Michael Fink. Object classification from a single example utilizing class relevance metrics. *NIPS*, 17, 2005. 8.1

[118] John R Firth. A synopsis of linguistic theory 1930-1955. 1957. 3

[119] Jerry A Fodor. *The language of thought*, volume 5. Harvard University Press, 1975. 8.1

[120] Jerry A Fodor and Zenon W Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988. 8.2.1

[121] Peter Földiák. Learning invariance from transformation sequences. *Neural Computation*, 3(2), 1991. 3.1

[122] David F. Fouhey, Abhinav Gupta, and Martial Hebert. Data-driven 3D primitives for single image understanding. In *ICCV*, 2013. 5.1, 5.4

[123] Mikhail Frank, Jürgen Leitner, Marijn Stollenga, Alexander Förster, and Jürgen Schmidhuber. Curiosity driven reinforcement learning for motion planning on humanoids. *Frontiers in neurorobotics*, 2014. 10.1

[124] Linton C Freeman. *Elementary applied statistics: for students in behavioral science*. John Wiley & Sons, 1965. 10.4.4

[125] Gottlob Frege. Sense and reference. *The philosophical review*, 57(3):209–230, 1948. 8

[126] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *NNLS*, 25, 2014. 4.1

[127] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013. 8.1

[128] Huazhu Fu, Dong Xu, Bao Zhang, and Stephen Lin. Object-based multiple foreground video co-segmentation. In *CVPR*, 2014. 2.1

[129] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4), 1980. 4.2.2

[130] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2016. 7.1

[131] Michel Galley, Chris Brockett, Alessandro Sordoni, Yangfeng Ji, Michael Auli, Chris Quirk, Margaret Mitchell, Jianfeng Gao, and Bill Dolan. deltaBLEU: A discriminative metric for generation tasks with intrinsically diverse targets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 445–450, Beijing, China, July 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-2073. 9.4.2

[132] Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. Are you talking to a machine? dataset and methods for multilingual image question answering. *CoRR*, abs/1505.05612, 2015. URL http://arxiv.org/abs/1505.05612. 9.1

[133] Thomas David Garvey. Perceptual strategies for purposive vision. 1976. 10.1

[134] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 2.3, 2.3

[135] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *IJRR*, 2013. 2.1, 2.2.1, 2.3.2

[136] Georgios Georgakis, Md Alimoor Reza, Arsalan Mousavian, Phi-Hung Le, and Jana Košecká. Multiview rgb-d dataset for object instance detection. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 426–434. IEEE, 2016. 7.1, 7.2, 7.1, 7.4.2, 7.7, 7.2, 7.5, 7.5.1, 7.9, 7.5.2

[137] Georgios Georgakis, Arsalan Mousavian, Alexander C Berg, and Jana Kosecka. Synthesizing training data for object detection in indoor scenes. *arXiv preprint arXiv:1702.07836*, 2017. 7, 7, 7.1, 7.2, 7.5.1, 7.6

[138] Spyros Gidaris and Nikos Komodakis. Object detection via a multi-region & semantic segmentation-aware cnn model. *arXiv preprint arXiv:1505.01749*, 2015. 5

[139] Ross Girshick. Fast R-CNN. In *ICCV*, 2015. 3.3.2, 5, 5.0.1, 5.1, 5.2.1, 5.5

[140] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015. 7.1

[141] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 3.3.3, 3.4, 5.1, 7.1

[142] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 6.1

[143] Georgia Gkioxari, Bharath Hariharan, Ross Girshick, and Jitendra Malik. R-cnns for pose estimation and action detection. *arXiv preprint arXiv:1406.5212*, 2014. 5.1

[144] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010. 8.4.1

[145] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D. Bourdev. Compressing deep convolutional networks using vector quantization. *CoRR*, abs/1412.6115, 2014. URL `http://arxiv.org/abs/1412.6115`. 6.8

[146] Jonathan Gordon and Benjamin Van Durme. Reporting bias and knowledge extraction. In *Automated Knowledge Base Construction (AKBC) 2013: The 3rd Workshop on Knowledge Extraction, at CIKM 2013*, AKBC'13, 2013. 4

[147] Ross Goroshin, Joan Bruna, Jonathan Tompson, David Eigen, and Yann LeCun. Unsupervised learning of spatiotemporally coherent metrics. In *ICCV*, 2015. 3.1

[148] Helmut Grabner, Christian Leistner, and Horst Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, 2008. 2.1

[149] Richard Langton Gregory. Eye and brain: The psychology of seeing. 1966. 4.1, 4.3.3

[150] Ulf Grenander. *General pattern theory-A mathematical study of regular structures*. Clarendon Press, 1993. 8.1

[151] Quanquan Gu and Jie Zhou. Learning the shared subspace for multi-task clustering and transductive transfer classification. In *ICDM*, 2009. 5.5.1

[152] Sergio Guadarrama, Niveda Krishnamoorthy, Girish Malkarnenkar, Subhashini Venugopalan, Raymond Mooney, Trevor Darrell, and Kate Saenko. Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *ICCV*, 2013. 8.1

[153] Jiaming Guo, Zhuwen Li, Loong-Fah Cheong, and Steven Zhiying Zhou1. Video co-segmentation for meaningful action extraction. In *CVPR*, 2013. 2.1

[154] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2315–2324, 2016. 7, 7.1

[155] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *CVPR*, 2013. 5.4

[156] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *ECCV*, 2014. 5.2

[157] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*. IEEE, 2006. 3.1, 3.4.2, 3.3, 3.5, 3.5

[158] Feng Han and Song-Chun Zhu. Bottom-up/top-down image parsing with attribute grammar. *TPAMI*, 31, 2009. 8.1

[159] Seungyeop Han, Haichen Shen, Matthai Philipose, Sharad Agarwal, Alec Wolman, and Arvind Krishnamurthy. MCDNN: An Approximation-Based Execution Framework for Deep Stream Processing Under Resource Constraints. In *Conference MobiSys'16 The 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobieSys '16. ACM, 2016. 6, 6, 6.4, 6.7.1, 6.8

[160] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. Eie: Efficient inference engine on compressed deep neural network. *SIGARCH Comput. Archit. News*, 44(3):243–254, June 2016. ISSN 0163-5964. doi: 10.1145/3007787.3001163. URL http://doi.acm.org/10.1145/3007787.3001163. 6.8

[161] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Scenenet: Understanding real world indoor scenes with synthetic data. *arXiv preprint arXiv:1511.07041*, 2015. 7.1

[162] S. Hare, A. Saffari, and P.H.S. Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011. 2.1

[163] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. In *ECCV*, 2012. 2.2.4, 2.3.1, 2.3.1

[164] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *ECCV*. Springer, 2014. 5

[165] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 8.2.2, 8.2.3

[166] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6.1, 6.1, 6.4, 10.3.5, 10.4.7

[167] Michael Heilman and Noah A. Smith. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617, Los Angeles, California, June 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N10-1086. 9.1

[168] Geremy Heitz and Daphne Koller. Learning spatial context: Using stuff to find things. In *ECCV*. 2008. 5.1

[169] David Held, Sebastian Thrun, and Silvio Savarese. Robust single-view instance recognition. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 2152–2159. IEEE, 2016. 7.1

[170] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *arXiv:1709.06560*, 2017. 10.4.1

[171] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):876–888, 2012. 7.1

[172] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997. 10.1

[173] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9, 1997. 3.1, 4.3.5

[174] Micah Hodosh, Peter Young, and Julia Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *J. Artif. Int. Res.*, 47(1):853–899, May 2013. ISSN 1076-9757. URL http://dl.acm.org/citation.cfm?id=2566972.2566993. 9.3.2

[175] Donald D Hoffman and Whitman A Richards. Parts of recognition. *Cognition*, 18, 1984. 8.1

[176] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. In *European conference on computer vision*, pages 340–353. Springer, 2012. 7.5.2

[177] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv:1112.5745*, 2011. 10.1

[178] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL http://arxiv.org/abs/1704.04861. 6.1, 6.4, 6.8

[179] Edward Hsiao, Alvaro Collet, and Martial Hebert. Making specific features less discriminative to improve point-based 3d object recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2653–2660. IEEE, 2010. 7.1

[180] Hanzhang Hu, Debadeepta Dey, J Andrew Bagnell, and Martial Hebert. Anytime neural networks via joint optimization of auxiliary losses. *arXiv preprint arXiv:1708.06832*, 2017. 6.8

[181] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. *ICCV*, 2017. 10.1, 10.3

[182] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense convolutional networks for efficient prediction. *arXiv preprint arXiv:1703.09844*, 2017. 6.8

[183] Ting-Hao Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross B. Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, C. Lawrence Zitnick, Devi Parikh, Lucy Vanderwende, Michel Galley, and Margaret Mitchell. Visual storytelling. In *Proceedings of NAACL 2016*. Association for Computational Linguistics, 2016. URL http://arxiv.org/abs/1604.03968. 9.2.1

[184] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *CoRR*, 2016. URL http://arxiv.org/abs/1609.07061. 6.8

[185] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. 2016. URL http://arxiv.org/abs/1602.07360. 6.8

[186] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 3.3

[187] Phillip Isola, Joseph J. Lim, and Edward H. Adelson. Discovering states and transformations in image collections. In *CVPR*, 2015. 8.1, 8.3, 8.4, 8.3.2, 8.1, 8.4, 8.4

[188] Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H Adelson. Learning visual groups from co-occurrences in space and time. *arXiv preprint arXiv:1511.06811*, 2015. 3.1

[189] Hamid Izadinia, Bryan C Russell, Ali Farhadi, Matthew D Hoffman, and Aaron Hertzmann. Deep classifiers from image tags in the wild. In *Workshop on Community-Organized Multimodal Mining: Opportunities for Novel Solutions*. ACM, 2015. 4.1

[190] Allan Jabri, Armand Joulin, and Laurens van der Maaten. Revisiting visual question answering baselines. In *ECCV*. Springer, 2016. 10.1

[191] Ali Jalali, Sujay Sanghavi, Chao Ruan, and Pradeep K Ravikumar. A dirty model for multi-task learning. In *NIPS*, 2010. 5.1

[192] Dinesh Jayaraman and Kristen Grauman. Learning image representations equivariant to ego-motion. *ICCV*, 2015. 3.1

[193] Dinesh Jayaraman and Kristen Grauman. Slow and steady feature analysis: Higher order temporal coherence in video. *arXiv preprint arXiv:1506.04714*, 2015. 3.1

[194] Dinesh Jayaraman, Fei Sha, and Kristen Grauman. Decorrelating semantic visual attributes by resisting the urge to share. In *CVPR*, 2014. 8.1

[195] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACMM*, 2014. 3.2.3, 4.3.5, 5.4, 5.4.2

[196] Li jia Li, Hao Su, Eric P. Xing, and Li Fei-fei. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, 2010. 2.2.4

[197] Jodoin, J.-P., Bilodeau, G.-A., and N Saunier. Urban tracker: Multiple object tracking in urban mixed traffic. In *IEEE Winter conference on Applications of Computer Vision (WACV14)*, March 2014. 6.6

[198] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CVPR*, 2016. 10, 10.1, 10.4, 10.4.7

[199] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. *ICCV*, 2017. 10.1, 10.3, 10.3.5, 10.4, 10.4.1, 10.4.8

[200] Joseph Jordania. *Who Asked the First Question? The Origins of Human Choral Singing, Intelligence, Language and Speech*. Logos, 2006. 9

[201] Antonio López Jose C. Rubio, Joan Serrat. Video cosegmentation. In *ACCV*, 2012. 2.1

[202] Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class active learning for image classification. In *CVPR*, 2009. 10.1

[203] Armand Joulin, Laurens van der Maaten, Allan Jabri, and Nicolas Vasilache. Learning visual features from large weakly supervised data. *arXiv preprint arXiv:1511.02251*, 2015. 4.1

[204] Mayank Juneja, Andrea Vedaldi, CV Jawahar, and Andrew Zisserman. Blocks that shout: Distinctive parts for scene classification. In *CVPR*, 2013. 3.1

[205] Junho Yim1 Heechul Jung, ByungIn Yoo, Changkyu Choi, Dusik Park, and Junmo Kim. Rotating your face using multi-task deep neural network. In *CVPR*, 2015. 5.1

[206] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *TPAMI*, 2012. 2.1, 2.2.1

[207] T. Kanamori, S. Hido, and M. Sugiyama. A least-squares approach to direct importance estimation. *JMLR*, 2009. 2.2.2, 2.3

[208] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. Noscope: Optimizing neural network queries over video at scale. In *Proceedings of the VLDB Endowment, Vol. 10, No. 11*, 2017. 6, 6.8

[209] Zhuoliang Kang, Kristen Grauman, and Fei Sha. Learning with whom to share in multi-task feature learning. In *ICML*, 2011. 5.1, 5.2.3, 5.4.1

[210] Ashish Kapoor, Kristen Grauman, Raquel Urtasun, and Trevor Darrell. Active learning with gaussian processes for object categorization. In *ICCV*, 2007. 10.1

[211] Andrej Karpathy, Armand Joulin, and Fei Fei F Li. Deep fragment embeddings for bidirectional image sentence mapping. In *NIPS*, 2014. 8.1

[212] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 2

[213] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. Rendering synthetic objects into legacy photographs. In *ACM Transactions on Graphics (TOG)*, volume 30, page 157. ACM, 2011. 7, 7.1, 7.6

[214] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *JACM*, 45, 1998. 4.1

[215] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 10.4.7

[216] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 3.1

[217] Iasonas Kokkinos. Ubernet: Training a 'universal' convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. *CoRR*, abs/1609.02132, 2016. URL `http://arxiv.org/abs/1609.02132`. 6.8

[218] Ruud Koolen, Albert Gatt, Martijn Goudbeek, and Emiel Krahmer. *Journal of Pragmatics*, 43 (13):3231–3250, 2011. 4.1

[219] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 6.6

[220] Vivek Krishnan and Deva Ramanan. Tinkering under the hood: Interactive zero-shot learning with net surgery. *arXiv preprint arXiv:1612.04901*, 2016. 8.1

[221] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 3.2.3, 4.3.1, 5.0.1, 5.2, 5.4, 5.4, 5.4.2, 5.5, 6.1

[222] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 7.1

[223] A. Kronfeld. Conversationally relevant descriptions. *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, 1989. 4.1

[224] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. 3, 3.1, 3.4, 3.4.3

[225] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *NIPS*, 2016. 10.1

[226] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NIPS*, 2010. 10.3.3

[227] Igor Labutov, Sumit Basu, and Lucy Vanderwende. Deep questions without deep understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015. 9.1

[228] S. Lad and D. Parikh. Interactively guiding semi-supervised clustering via attribute-based explanations. In *ECCV*, 2014. 2.1

[229] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011. 7.1, 7.2, 7.4.1, 7.5

[230] Brenden M Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, volume 172, page 2, 2011. 8.1

[231] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Concept learning as motor program induction: A large-scale empirical study. In *Proceedings of the 34th Annual Conference of the Cognitive Science Society*, 2012. 8.1

[232] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *arXiv preprint arXiv:1604.00289*, 2016. 8.1

[233] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009. 5.1, 8.1, 8.3.1

[234] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *NIPS*, 2008. 10.1

[235] Maksim Lapin, Bernt Schiele, and Matthias Hein. Scalable multitask representation learning for scene classification. In *CVPR*, 2014. 5.5.1

[236] Jan Larsen, L Nonboe, Mads Hintz-Madsen, and Lars Kai Hansen. Design of robust neural network classifiers. In *Acoustics, Speech and Signal Processing*, volume 2, 1998. 4.1

[237] Quoc V Le. Building high-level features using large scale unsupervised learning. In *ICASSP*, 2013. 3.1

[238] Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015. 8.4.1

[239] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1, 1989. 3, 4, 4.2.2, 7, 7.1, 8.1, 10.1

[240] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. In *NIPS*, 2006. 3.1

[241] Tae Jun Lee, Justin Gottschlich, Nesime Tatbul, Eric Metcalf, and Stan Zdonik. Precision and recall for range-based anomaly detection. *SysML*, Feb 2018. 6.1

[242] Y. J. Lee, J. Ghosh, and K. Grauman. Discovering important people and objects for egocentric video summarization. In *CVPR*, 2012. 2.1

[243] Jimmy Lei Ba, Kevin Swersky, Sanja Fidler, et al. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *CVPR*, 2015. 8, 8.1, 8.3.1

[244] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A persona-based neural conversation model. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2016. URL `http://arxiv.org/abs/1603.06155`. 9.5

[245] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, 2010. 10.1

[246] Quannan Li, Jiajun Wu, and Zhuowen Tu. Harvesting mid-level visual concepts from large-scale internet images. In *CVPR*, 2013. 3.1

[247] Xiaoli Li and Bing Liu. Learning to classify texts using positive and unlabeled data. In *IJCAI*, volume 3, 2003. 2, 4.1

[248] Xin Li and Yuhong Guo. Adaptive active learning for image classification. In *CVPR*, 2013. 10.1

[249] Xiaodan Liang, Si Liu, Yunchao Wei, Luoqi Liu, Liang Lin, and Shuicheng Yan. Computational baby learning. *arXiv preprint*, 2014. 2.1

[250] Chin-Yew Lin and Franz Josef Och. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. doi: 10.3115/1218955.1219032. URL http://dx.doi.org/10.3115/1218955.1219032. 9.3.2

[251] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 1.1, 1.2, 2, 4, 4.3, 4.3.1, 4.3.2, 7.2, 7.4.2, 7.5

[252] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312. 9.1

[253] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. pages 740–755", 2014. 6.1

[254] David Lindberg, Fred Popowich, John Nesbit, and Phil Winne. Generating natural language questions to support learning on-line. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 105–114, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2114. 9.1

[255] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Penksy. Sparse convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 6.8

[256] Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S Yu. Building text classifiers using positive and unlabeled examples. In *ICDM*, 2003. 2, 4.1

[257] D. Liu, Gang Hua, and Tsuhan Chen. A hierarchical visual model for video object summarization. *PAMI*, 2010. 2.1

[258] Feng Liu, Tao Xiang, Timothy M Hospedales, Wankou Yang, and Changyin Sun. iVQA: Inverse visual question answering. *arXiv:1710.03370*, 2017. 10.1

[259] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer, 2016. 7.1

[260] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. *NAACL*, 2015. 5.1

[261] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *arXiv preprint arXiv:1411.4038*, 2014. 5.4

[262] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 7.4.1

[263] Mingsheng Long and Jianmin Wang. Learning multiple tasks with deep relationship networks. *CoRR*, abs/1506.02117, 2015. URL http://arxiv.org/abs/1506.02117. 6.8

[264] David G Lowe. Local feature view clustering for 3d object recognition. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001. 7.1

[265] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 7.1

[266] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *ECCV*, 2016. 8.1, 8.1, 8.3, 8.3.1, 8.2, 8.3.3

[267] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogério Schmidt Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. *CVPR*, 2016. URL http://arxiv.org/abs/1611.05377. 6.8

[268] Mitchell M., Reiter E., and van Deemter K. Typicality and object reference. 2013. 4, 4.1

[269] Mateusz Malinowski and Mario Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input. In *NIPS*, 2014. 10.1

[270] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011. 2.3, 2.2.4, 2.3

[271] Naresh Manwani and PS Sastry. Noise tolerance under risk minimization. *Cybernetics*, 43, 2013. 4.1

[272] Junhua Mao, Xu Wei, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan L Yuille. Learning like a child: Fast novel visual concept learning from sentence descriptions of images. In *ICCV*, 2015. 8.1

[273] Marcin Marszałek, Ivan Laptev, and Cordelia Schmid. Actions in context. In *CVPR*, 2009. 2

[274] Francisco Massa, Bryan C Russell, and Mathieu Aubry. Deep exemplar 2d-3d detection by adapting from real to rendered views. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6024–6033, 2016. 7.1

[275] Karen Mazidi and Rodney D. Nielsen. Linguistic considerations in automatic question generation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 321–326, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P14-2053. 9.1

[276] Thomas Mensink, Efstratios Gavves, and Cees GM Snoek. Costa: Co-occurrence statistics for zero-shot classification. In *CVPR*, 2014. 8.1

[277] Kathryn E Merrick and Mary Lou Maher. *Motivated reinforcement learning: curious characters for multiuser games*. Springer Science & Business Media, 2009. 10.1

[278] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A comparison of affine region detectors. *International journal of computer vision*, 65(1-2):43–72, 2005. 7.1

[279] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. 3

[280] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013. 3, 8.3.1

[281] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119, 2013. URL http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionali 11

[282] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11): 39–41, 1995. 4.3.2, 9.2.2

[283] Ishan Misra, Abhinav Shrivastava, and Martial Hebert. Data-driven exemplar model selection. In *WACV*, 2014. 2.2.4, 2.3.1, 2.3.2

[284] Ishan Misra, Abhinav Shrivastava, and Martial Hebert. Watch and learn: Semi-supervised learning of object detectors from videos. In *CVPR*, 2015. 4.1

[285] Ishan Misra, Abhinav Shrivastava, and Martial Hebert. Watch and learn: Semi-supervised learning of object detectors from videos. In *CVPR*, 2015. 1, 1

[286] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch Networks for Multi-task Learning. In *CVPR*, 2016. 2, 4

[287] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch Networks for Multi-task Learning. In *CVPR*, 2016. 6.8

[288] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. Shuffle and Learn: Unsupervised Learning using Temporal Order Verification. In *ECCV*, 2016. 1, 2

[289] Ishan Misra, C. Lawrence Zitnick, Margaret Mitchell, and Ross Girshick. Seeing through the Human Reporting Bias: Visual Classifiers from Noisy Human-Centric Labels. In *CVPR*, 2016. 3, 3, 8.3.1

[290] Ishan Misra, Abhinav Gupta, and Martial Hebert. From Red Wine to Red Tomato: Composition with Context. In *CVPR*, 2017. 4, 7

[291] Ishan Misra, Ross Girshick, Rob Fergus, Martial Hebert, Abhinav Gupta, and Laurens van der Maaten. Learning by Asking Questions. In *CVPR*, 2018. 1.3, 9

[292] Chaitanya Mitash, Kostas E Bekris, and Abdeslam Boularias. A self-supervised learning system for object detection using physics simulation and multi-view pose estimation. *arXiv preprint arXiv:1703.03347*, 2017. 7.1

[293] Ruslan Mitkov and Le An Ha. Computer-aided generation of multiple-choice tests. In Jill Burstein and Claudia Leacock, editors, *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing*, pages 17–22, 2003. URL http://www.aclweb.org/anthology/W03-0203.pdf. 9.1

[294] Eric Mjolsness. Connectionist grammars for high-level vision. *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*, pages 423–451, 1994. 8.1

[295] Volodymyr Mnih and Geoffrey E Hinton. Learning to label aerial images from noisy data. In *ICML*, 2012. 4.1

[296] Hossein Mobahi, Ronan Collobert, and Jason Weston. Deep learning from temporal coherence in video. In *ICML*, 2009. 3.1, 3.4.2, 3.4.2, 3.3

[297] Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. Generating natural questions about an image. In *ACL*, 2016. 10, 10.1

[298] Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. Generating natural questions about an image. In *ACL*, 2016. 1.3, 8

[299] Yair Movshovitz-Attias, Takeo Kanade, and Yaser Sheikh. How useful is photo-realistic rendering for visual learning? In *Computer Vision–ECCV 2016 Workshops*, pages 202–217. Springer International Publishing, 2016. 7, 7, 7.1

[300] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *NIPS*. 2013. 4.1, 4.3.1

[301] David F Nettleton, Albert Orriols-Puig, and Albert Fornells. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial intelligence review*, 33, 2010. 4.1

[302] NLPcaffe. NLPcaffe. `http://github.com/Russell91/NLPCaffe`. 4.3.5

[303] Guillaume Obozinski, Ben Taskar, and Michael Jordan. Multi-task feature selection. *Statistics Department, UC Berkeley, Tech. Rep*, 2006. 5.1

[304] Guillaume Obozinski, Ben Taskar, and Michael I Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20, 2010. 5.1

[305] Sangmin Oh, A. Hoogs, A. Perera, N. Cuntoor, Chia-Chih Chen, Jong Taek Lee, S. Mukherjee, J. K. Aggarwal, Hyungtae Lee, L. Davis, E. Swears, Xioyang Wang, Qiang Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, Bi Song, A. Fong, A. Roy-Chowdhury, and M. Desai. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR*, 2011. 2, 2, 2.3, 2.3

[306] Bruno A Olshausen et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583), 1996. 3.1

[307] David R Olson. Language and thought: Aspects of a cognitive theory of semantics. *Psychological review*, 77, 1970. 4

[308] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 6

[309] Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. Im2text: Describing images using 1 million captioned photographs. In *Neural Information Processing Systems (NIPS)*, 2011. 9.3.2

[310] Andrew Owens, Phillip Isola, Josh McDermott, Antonio Torralba, Edward H Adelson, and William T Freeman. Visually indicated sounds. In *CVPR*, 2016. 3.1

[311] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22, 2010. 5.1, 8.1

[312] Yu Pang and Haibin Ling. Finding the best from the second bests - inhibiting subjective bias in evaluation of visual tracking algorithms. In *ICCV*, 2013. 2.1

[313] Seymour A Papert. The summer vision project. 1966. 1

[314] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002. 9.2.1, 9.2.3, 9.4.2

[315] Devi Parikh and Kristen Grauman. Relative attributes. In *ICCV*, 2011. 8.1

[316] Dennis Park and Deva Ramanan. Articulated pose estimation with tiny synthetic videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 58–66, 2015. 7.1

[317] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017. 10.1

[318] Novi Patricia and Barbara Caputo. Learning to learn, from transfer learning to domain adaptation: A unifying perspective. In *CVPR*, 2014. 8.1

[319] Genevieve Patterson and James Hays. Coco attributes: Attributes for people, animals, and objects. In *ECCV*, 2016. 8.1, 8.3.2

[320] Xingchao Peng, Baochen Sun, Karim Ali, and Kate Saenko. Learning deep object detectors from 3d models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1278–1286, 2015. 7, 7.1

[321] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. *arXiv:1709.07871*, 2017. 10.1, 10.4

[322] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 313–318. ACM, 2003. 7.1, 7.4.2

[323] Xavier Perez-Sala, Sergio Escalera, Cecilio Angulo, and Jordi Gonzalez. A survey on model based approaches for 2d and 3d visual human pose recovery. *Sensors*, 14, 2014. 3.1

[324] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*, 2011. 2.1, 2.2.1, 2.2.3

[325] Ronald Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6), 2010. 3.1

[326] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In *CVPR*, 2012. 2.1, 2.2.1

[327] J. Pustejovsky, P. Hanks, R. Sauri, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro, and M. Lazo. The TIMEBANK corpus. In *Proceedings of Corpus Linguistics 2003*, pages 647–656, Lancaster, March 2003. 9.2.2

[328] Ariadna Quattoni, Michael Collins, and Trevor Darrell. Transfer learning for image classification with sparse prototype representations. In *CVPR*, 2008. 5.1

[329] Rajeev Ranjan, Vishal M Patel, and Rama Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *arXiv preprint arXiv:1603.01249*, 2016. 6.8

[330] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016. 6.8

[331] Arijit Ray, Gordon Christie, Mohit Bansal, Dhruv Batra, and Devi Parikh. Question relevance in vqa: identifying non-visual and false-premise questions. *arXiv:1606.06622*, 2016. 10.1

[332] Ali S Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *CVPR Workshop*, 2014. 5.1

[333] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2014, Columbus, OH, USA, June 23-28, 2014*, pages 512–519, 2014. doi: 10.1109/CVPRW.2014.131. URL http://dx.doi.org/10.1109/CVPRW.2014.131. 6

[334] Arthur S Reber. Implicit learning and tacit knowledge. *Journal of experimental psychology: General*, 118(3):219, 1989. 3

[335] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 7.1

[336] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014. 4.1

[337] Scott Reed, Zeynep Akata, Bernt Schiele, and Honglak Lee. Learning deep representations of fine-grained visual descriptions. *arXiv preprint arXiv:1605.05395*, 2016. 8.1

[338] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en. 11

[339] Mengye Ren, Ryan Kiros, and Richard Zemel. Question answering about images using visual semantic embeddings. In *Deep Learning Workshop, ICML 2015*, 2015. 9.1, 9.2.1

[340] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 3.3.2

[341] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 7, 7.1, 7.1, 7.4.2, 7.4.2, 7.5

[342] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014. 3.1

[343] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision*, pages 102–118. Springer, 2016. 7.1

[344] Bernardino Romera-Paredes and PHS Torr. An embarrassingly simple approach to zero-shot learning. In *ICML*, 2015. 8.1

[345] Bernardino Romera-Paredes, Andreas Argyriou, Nadia Berthouze, and Massimiliano Pontil. Exploiting unrelated tasks in multi-task learning. In *ICAIS*, 2012. 5.1

[346] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 7.1

[347] Eleanor Rosch. Principles of categorization. *Concepts: core readings*, pages 189–206, 1999. 4.1

[348] C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. In *WACV*, 2005. 2.2.1

[349] Sebastian Ruder. An overview of multi-task learning in deep neural networks. abs/1706.05098, 2017. URL http://arxiv.org/abs/1706.05098. 6.8

[350] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985. 4.2.2

[351] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115, 2015. 1.1, 3, 3.4.1, 4, 4.3.1, 5.4.3, 8.2.3, 8.3.1, 10.3.5, 10.4.7

[352] Bryan C Russell, William T Freeman, Alexei A Efros, Josef Sivic, and Andrew Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR*, 2006. 3.1

[353] Fereshteh Sadeghi, C Lawrence Zitnick, and Ali Farhadi. Visalogy: Answering visual analogy questions. In *NIPS*, 2015. 8.1

[354] Mohammad Amin Sadeghi and Ali Farhadi. Recognition using visual phrases. In *CVPR*, 2011. 8, 8.1, 8.3.1, 8.3.3

[355] Amir Saffari, Christian Leistner, Martin Godec, and Horst Bischof. Robust multi-view boosting with priors. In *ECCV*, 2010. 2.1

[356] Ruslan Salakhutdinov and Geoffrey E Hinton. Deep boltzmann machines. In *ICAIS*, 2009. 3.1

[357] Adam Santoro, David Raposo, David GT Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *arXiv:1706.01427*, 2017. 10.1, 10.3.5, 10.4.7

[358] Benjamin Sapp and Ben Taskar. Modec: Multimodal decomposable models for human pose estimation. In *CVPR*, 2013. 3, 3.1, 3.5

[359] Henry Schneiderman and Takeo Kanade. Object detection using the statistics of parts. *IJCV*, 56, 2004. 5.1

[360] Nima Sedaghat and Thomas Brox. Unsupervised generation of a viewpoint annotated car dataset from videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1322, 2015. 7.1

[361] Julie C Sedivy. Pragmatic versus form-based accounts of referential contrast: Evidence for effects of informativity expectations. *Journal of psycholinguistic research*, 32(1), 2003. 4, 4.1, 4.3.3

[362] Ozan Sener and Silvio Savarese. A geometric approach to active learning for convolutional neural networks. *arXiv:1708.00489*, 2017. 10.4.4

[363] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *CVPR*, 2013. 3.1

[364] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010. 10.1

[365] Claude E Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37, 1949. 3.2.1

[366] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014. 7.1

[367] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *TPAMI*, 22, 2000. 5.1

[368] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*. 2006. 5.1

[369] A. Shrivastava, T. Malisiewicz, A. Gupta, and A. A. Efros. Data-driven visual similarity for cross-domain image matching. *ACM Trans. on Graphics*, 2011. 2.2.1, 2.2.2, 2.3, 2.3.1, 2.3.2

[370] A. Shrivastava, S. Singh, and A. Gupta. Constrained semi-supervised learning using attributes and comparative attributes. In *ECCV*, 2012. 2, 2.1

[371] Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Constrained semi-supervised learning using attributes and comparative attributes. In *ECCV*. 2012. 4.1

[372] Zhangzhang Si and Song-Chun Zhu. Learning and-or templates for object recognition and detection. *TPAMI*, 35, 2013. 8, 8.1

[373] Behjat Siddiquie and Abhinav Gupta. Beyond active noun tagging: Modeling contextual interactions for multi-class active learning. In *CVPR*, 2010. 10.1

[374] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 5.0.1, 5.1, 5.4, 5.5, 5.5, 5.5.2

[375] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4, 4.2.2, 4.3.1, 6.1, 8.2.3, 8.3.1, 9.3

[376] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 3.3, 3.4, 3.4.1

[377] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014. 6, 6.1

[378] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 7.1, 7.4.1, 7.4.2, 7.5

[379] Vikas Sindhwani and Partha Niyogi. A co-regularized approach to semi-supervised learning with multiple views. In *ICML Workshop*, 2005. 2.2.1

[380] Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. Bigbird: A large-scale 3d database of object instances. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 509–516. IEEE, 2014. 7.2, 7.4.1, 7.5, 7.5.1

[381] Saurabh Singh, Abhinav Gupta, and Alexei Efros. Unsupervised discovery of mid-level discriminative patches. *ECCV*, 2012. 3.1

[382] Josef Sivic, Bryan C Russell, Alexei A Efros, Andrew Zisserman, and William T Freeman. Discovering objects and their location in images. In *ICCV*, 2005. 3.1

[383] Shuran Song, Linguang Zhang, and Jianxiong Xiao. Robot in a room: Toward perfect object recognition in closed environments. 7.1, 7.2

[384] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 2, 3, 3.1, 3.3, 3.4

[385] Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205, Denver, Colorado, May–June 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N15-1020`. 9.5

[386] Merrielle Spain and Pietro Perona. Some objects are more equal than others: Measuring and predicting importance. In *ECCV*. Springer, 2008. 4.1

[387] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. *arXiv preprint arXiv:1502.04681*, 2015. 3.1

[388] Charles Stein et al. Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. In *Proceedings of the Third Berkeley symposium on mathematical statistics and probability*, volume 1. 1956. 5.1

[389] Jan Storck, Sepp Hochreiter, and Jürgen Schmidhuber. Reinforcement driven information acquisition in non-deterministic environments. In *Proceedings of the international conference on artificial neural networks, Paris*, 1995. 10.1

[390] Chi Su et al. Multi-task learning with low rank attribute embedding for person re-identification. In *ICCV*, 2015. 5.5.1

[391] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015. 7, 7, 7.1

[392] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. In *ICLR Workshop*, 2015. 4.1, 4.3.1

[393] Jian Sun and Jean Ponce. Learning discriminative part detectors for image classification and cosegmentation. In *ICCV*, 2013. 3.1

[394] Ron Sun and C Lee Giles. Sequence learning: from recognition and prediction to sequential decision making. *IEEE Intelligent Systems*, 16(4), 2001. 3

[395] Ron Sun, Edward Merrill, and Todd Peterson. From implicit skills to explicit knowledge: A bottom-up model of skill learning. *Cognitive science*, 25(2), 2001. 3

[396] James Steven Supancic III and Deva Ramanan. Self-paced learning for long-term tracking. In *CVPR*, 2013. 2.1

[397] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014. URL http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks. 9.3.1

[398] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998. 10, 10.3.3

[399] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 6.1, 6.1, 6.4

[400] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, pages 1701–1708, Washington, DC, USA, 2014. IEEE Computer Society. ISBN 978-1-4799-5118-5. doi: 10.1109/CVPR.2014.220. URL http://dx.doi.org/10.1109/CVPR.2014.220. 6.1

[401] K. Tang, R. Sukthankar, J. Yagnik, and L. Fei-Fei. Discriminative segment annotation in weakly labeled video. In *CVPR*, 2013. 2.1

[402] Kevin Tang, Vignesh Ramanathan, Li Fei-Fei, and Daphne Koller. Shifting weights: Adapting object detectors from image to video. In *NIPS*, 2012. 8.1

[403] Graham W Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. In *ECCV*. 2010. 3.1

[404] Alex Teichman and Sebastian Thrun. Tracking-based semi-supervised learning. In *RSS*, 2011. 2.1

[405] Jay M Tenenbaum. On locating objects by their distinguishing features in multisensory images. *Computer Graphics and Image Processing*, 2, 1973. 4

[406] Piotr Teterwak and Lorenzo Torresani. Shared Roots: Regularizing Deep Neural Networks through Multitask Learning. Technical Report TR2014-762, Dartmouth College, Computer Science, 2014. 5.1, 5.5.3

[407] Jesse Thomason, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Raymond J Mooney. Integrating language and vision to generate natural language descriptions of videos in the wild. In *COLING*, volume 2, page 9, 2014. 8.1

[408] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. The new data and new challenges in multimedia research. *arXiv preprint arXiv:1503.01817*, 2015. 1.1, 4, 4.3, 4.3.2

[409] Sebastian Thrun and Tom M Mitchell. Learning one more thing. Technical report, DTIC Document, 1994. 8.1

[410] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of cnn activations. *arXiv preprint arXiv:1511.05879*, 2015. 7.1

[411] Antonio Torralba, Kevin P Murphy, and William T Freeman. Sharing visual features for multiclass and multiview object detection. *PAMI*, 29, 2007. 5.1

[412] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, 2014. 3.5

[413] Kenneth Tran, Xiaodong He, Lei Zhang, Jian Sun, Cornelia Carapcea, Chris Thrasher, Chris Buehler, and Chris Sienkiewicz. Rich image captioning in the wild. In *Proceedings of Deep Vision Workshop at CVPR 2016*. IEEE, June 2016. URL http://research.microsoft.com/apps/pubs/default.aspx?id=264408. 9.2.3

[414] Zhuowen Tu, Xiangrong Chen, Alan L Yuille, and Song-Chun Zhu. Image parsing: Unifying segmentation, detection, and recognition. *IJCV*, 2005. 8.1

[415] Naman Turakhia and Devi Parikh. Attribute dominance: What pops out? In *ICCV*, 2013. 4.1

[416] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4068–4076, 2015. 7.1

[417] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders. Segmentation as selective search for object recognition. *IJCV*, 2011. 2.2.3, 2.3

[418] Ielka van der Sluis, Albert Gatt, and Kees van Deemter. Manual for the tuna corpus: Referring expressions in two domains. *Technical Report AUCS/TR0705*, 2006. 4.1

[419] Grant Van Horn and Pietro Perona. The devil is in the tails: Fine-grained classification in the wild. *arXiv preprint arXiv:1709.01450*, 2017. 1.1

[420] Lucy Vanderwende. The importance of being important: Question generation. In *In Workshop on the Question Generation Shared Task and Evaluation Challenge*, Arlington, VA, 2008. 9

[421] Lucy Vanderwende, Arul Menezes, and Chris Quirk. An amr parser for english, french, german, spanish and japanese and a new amr-annotated corpus. Proceedings of NAACL 2015, June 2015. URL http://research.microsoft.com/apps/pubs/default.aspx?id=241902. 9.2.1

[422] Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: similarity control and knowledge transfer. *JMLR*, 2015. 10.5

[423] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. *arXiv preprint arXiv:1701.01370*, 2017. 7.1

[424] Sudheendra Vijayanarasimhan and Kristen Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. *IJCV*, 2014. 10.1

[425] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008. 3.1, 7

[426] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition*, 2015. URL http://arxiv.org/abs/1411.4555. 9.3.1, 9.3.2

[427] Paul Viola, John C. Platt, and Cha Zhang. Multiple instance boosting for object detection. In *NIPS*, 2006. 4.3.1

[428] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. HOGgles: Visualizing Object Detection Features. *ICCV*, 2013. 2.2.1

[429] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, 101(1):184–204, 2013. 7.2

[430] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating the future by watching unlabeled video. *arXiv preprint arXiv:1504.08023*, 2015. 3.1

[431] L. Wang, G. Hua, R. Sukthankar, J. Xue, and N. Zheng. Video object discovery and co-segmentation with extremely weak supervision. In *ECCV*, 2014. 2.1

[432] Limin Wang, Yuanjun Xiong, Zhe Wang, and Yu Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015. 3.3, 3.4

[433] X. Wang, David F. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *CVPR*, 2015. 5.1, 5.4

[434] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015. 3.1, 3.4.2, 3.3, 3.5, 3.5

[435] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *ECCV*, 2016. 3.1

[436] Yu-Xiong Wang and Martial Hebert. Model recommendation: Generating object detectors from few samples. In *CVPR*, 2015. 2.2.4

[437] Yu-Xiong Wang and Martial Hebert. Learning to learn: Model regression networks for easy small sample learning. In *ECCV*, 2016. 8.1, 8.3.1, 8.4.1

[438] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *Advances in Neural Information Processing Systems*, 2016. URL http://arxiv.org/abs/1608.03665. 6.8

[439] Hans Westerbeek, Ruud Koolen, and Alfons Maes. Stored object knowledge and the production of referring expressions: The case of color typicality. *Frontiers in Psychology*, 6, 2015. 4, 4.1

[440] Laurenz Wiskott and Terrence J Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4), 2002. 3.1

[441] John H Wolfe. Automatic question generation from text-an aid to independent study. In *ACM SIGCUE Outlook*, volume 10, pages 104–112. ACM, 1976. 9.1

[442] Jay M Wong, Vincent Kee, Tiffany Le, Syler Wagner, Gian-Luca Mariottini, Abraham Schneider, Lei Hamilton, Rahul Chipalkatty, Mitchell Hebert, David Johnson, et al. Segicp: Integrated deep semantic segmentation and pose estimation. *arXiv preprint arXiv:1703.01661*, 2017. 7.1

[443] John Wright, Yi Ma, Julien Mairal, Guillermo Sapiro, Thomas S Huang, and Shuicheng Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98 (6):1031–1044, 2010. 5.1

[444] Tianfu Wu and Song-Chun Zhu. A numerical study of the bottom-up and top-down inference processes in and-or graphs. *IJCV*, 93, 2011. 8.1

[445] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *CVPR*, 2015. 4.1

[446] Yuanjun Xiong, Kai Zhu, Dahua Lin, and Xiaoou Tang. Recognize complex events from static images by fusing deep channels. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 9.1

[447] Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. Multi-task learning for classification with dirichlet process priors. *JMLR*, 8, 2007. 5.1

[448] Yi Yang and Deva Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *CVPR*, 2011. 8.1

[449] Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures of parts. *TPAMI*, 35, 2013. 3.5

[450] Yongxin Yang and Timothy M Hospedales. A unified perspective on multi-domain and multi-task learning. *arXiv preprint arXiv:1412.7489*, 2014. 5.1

[451] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *CVPR*, 2016. 10.1, 10.3.5, 10.4, 10.4.7

[452] Bangpeng Yao, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas J. Guibas, and Li Fei-Fei. Human action recognition by learning bases of action attributes and parts. In *International Conference on Computer Vision (ICCV)*, Barcelona, Spain, November 2011. 9.1

[453] Bangpeng Yao, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas J. Guibas, and Li Fei-Fei. Action recognition by learning bases of action attributes and parts. In *International Conference on Computer Vision (ICCV)*, Barcelona, Spain, November 2011. 9.1

[454] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3320–3328, Cambridge, MA, USA, 2014. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969033.2969197. 6

[455] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014. 5.1

[456] Jenny Yuen, Bryan C. Russell, Ce Liu, and Antonio Torralba. Labelme video: Building a video database with human annotations. In *ICCV*, 2009. 2

[457] Alan Yuille and Roozbeh Mottaghi. Complexity of representation and inference in compositional models with part sharing. *JMLR*, 2016. 8.1

[458] Kiwon Yun, Yifan Peng, Dimitris Samaras, Gregory J Zelinsky, and Tamara Berg. Studying relationships between human gaze, description, and computer vision. In *CVPR*, 2013. 4.1

[459] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 8.1

[460] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker Jr, Alberto Rodriguez, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. *arXiv preprint arXiv:1609.09475*, 2016. 7.1

[461] Cha Zhang and Zhengyou Zhang. Improving multiview face detection with multi-task deep convolutional neural networks. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 1036–1041. IEEE, 2014. 5.1

[462] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J. Freedman. Live Video Analytics at Scale with Approximation and Delay-Tolerance. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, NSDI '17, 2017. 6, 6, 6.8

[463] Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Yin and yang: Balancing and answering binary visual questions. In *CVPR*, 2016. 10.1

[464] Tianzhu Zhang, Bernard Ghanem, Si Liu, and Narendra Ahuja. Robust visual tracking via structured multi-task sparse learning. *International journal of computer vision*, 101(2):367–383, 2013. 5.1

[465] Zhang Zhang and Dacheng Tao. Slow feature analysis for human action recognition. *TPAMI*, 34(3), 2012. 3.1

[466] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *Computer Vision–ECCV 2014*, pages 94–108. Springer, 2014. 5.1, 6.8

[467] Ziming Zhang and Venkatesh Saligrama. Zero-shot learning via semantic similarity embedding. In *ICCV*, 2015. 8.1

[468] Liang Zheng, Yi Yang, and Qi Tian. Sift meets cnn: a decade survey of instance retrieval. *arXiv preprint arXiv:1608.01807*, 2016. 7.1

[469] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16(16):321–328, 2004. 4.1

[470] J. Zhou, J. Chen, and J. Ye. *MALSAR: Multi-tAsk Learning via StructurAl Regularization*. ASU, 2011. 5.5.1

[471] Jiayu Zhou, Jun Liu, Vaibhav A Narayan, and Jieping Ye. Modeling disease progression via fused sparse group lasso. In *SIGKDD*, 2012. 5.5.1

[472] Qiang Zhou, Gang Wang, Kui Jia, and Qi Zhao. Learning to share latent tasks for action recognition. In *ICCV*, 2013. 5.5.1

[473] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. 6.8

[474] Yipin Zhou and Tamara L. Berg. Temporal perception and prediction in ego-centric video. In *ICCV*, 2015. 3.1

[475] Long Zhu, Yuanhao Chen, Yifei Lu, Chenxi Lin, and Alan Yuille. Max margin and/or graph learning for parsing the human body. In *CVPR*, 2008. 8.1

[476] Long Zhu, Yuanhao Chen, Alan Yuille, and William Freeman. Latent hierarchical structural learning for object detection. In *CVPR*, 2010. 8.1

[477] Long Leo Zhu, Yuanhao Chen, and Alan Yuille. Recursive compositional models for vision: Description and review of recent work. *Journal of Mathematical Imaging and Vision*, 41, 2011. 8.1

[478] Song-Chun Zhu and David Mumford. *A stochastic grammar of images*. Now Publishers Inc, 2007. 8.1

[479] Xiaojin Zhu. Semi-supervised learning. In *Encyclopedia of Machine Learning*. 2010. 4.1

[480] Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919, 2003. 4.1

[481] Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. Visual7w: Grounded question answering in images. In *CVPR*, 2016. 10, 10.1

[482] Yuke Zhu, Oliver Groth, Michael S. Bernstein, and Li Fei-Fei. Visual7w: Grounded question answering in images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, IEEE, 2016. 9.1

[483] Shlomo Zilberstein. Using anytime algorithms in intelligent systems. In *AI Magazine, 17(3):73-83*, 1996. 6