The Dissertation Committee for Dinesh Jayaraman
certifies that this is the approved version of the following dissertation:

Embodied Learning for Visual Recognition

Committee:

Kristen Grauman, Supervisor

Alexei Efros

Joydeep Ghosh

Scott Niekum

Andrea Thomaz

Embodied Learning for Visual Recognition

by

Dinesh Jayaraman

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2017

அம்மாவுக்கும் அப்பாவுக்கும் அன்புடன் என் சமர்ப்பணம்.

Dedicated to my parents, Jayaraman and Padmavathy.

Acknowledgments

I have a great many people to thank. First and foremost, I must thank my parents, Jayaraman and Padmavathy. They have always motivated me to pursue knowledge and learning, and given me every opportunity to do so. My parents worked very hard to get to a position that afforded me the luxury to focus on the long term and pursue a PhD. Together, my parents and my sister, Janani, have been constant pillars of support in my life despite my impudence and many rebellions growing up, and even at times when I have not been appreciative of it. For these and many other things, I am forever grateful to them.

Over the years, my PhD advisor, Prof. Kristen Grauman, has been in equal parts a wonderful teacher, an inspiring mentor, a brilliant collaborator, and a reliable colleague and friend who has always looked out for my best interests. I will forever be thankful to her for the effort she has invested behind the scenes to allow me the intellectual freedom to define and shape my research without ever having to worry about funding, and for always being available to offer advice on scientific and other matters alike. She has truly been an academic parent to me and will remain an important influence on the rest of my life.

Prof. Alyosha Efros has been a second mentor to me. When I was still only starting out in graduate school, he shepherded me through a summer at UC Berkeley during which his scientific verve, generosity, and optimism left a lasting impression

on me. I also thank Prof. Joydeeph Ghosh, Prof. Scott Niekum, and Prof. Andrea Thomaz for helping shape my research as my thesis committee members.

My labmates over the years have taught me much. I thank Sung Ju, Jaechul, Sunil, Adriana, Lu Zheng, Chao-Yeh, Suyog, Aron, Bo, Viktoriia, Yu-Chuan, Antonino, Ziad, Danna, Wei-Lin, and Tushar. I will miss our ping pong games, conference deadline all-nighters, reading group discussions, and intense debates.

I have been fortunate during my time in Austin to have developed some great friendships outside the lab too. Thanks to Akhila, Deepti, Prem, Ravi, and Tarun in particular for helping to take the stress out of grad school and for bearing with me when I constantly claimed to be too busy to have fun.

I owe much of my academic curiosity and general worldview to four intellectually stimulating years at the Indian Institute of Technology Madras (IITM). My friends from there, too many to list, remain among my most valued.

Through various schools in my childhood, IITM, and UT, I have been taught by some exceptionally great teachers. In approximate chronological order: Ms. Ribero, Ms. Lakshmi, Ms. Padmini Raghavan, Ms. Uma Arthi, Mr. Bala, Prof. Aravind (my undergraduate thesis advisor), Prof. Shanthi Pavan, Prof. Milind Brahme, Prof. K. Srilata, Prof. Andrew Thangaraj, Prof. David Morton, Prof. Gustavo de Veciana, and my PhD advisor Prof. Kristen Grauman. I thank them all.

I reserve my last word of gratitude for you, the reader. The contents herein represent nearly three years of my efforts in the hope of learning something of value to someone. I hope that my thesis repays your attention.

Embodied Learning for Visual Recognition

Dinesh Jayaraman, Ph.D.
The University of Texas at Austin, 2017

Supervisor: Kristen Grauman

Over the fifty years or so of its existence, the field of visual recognition has gradually moved away from the first entirely manually designed rule-based methods, to methods that *learn* increasingly more and more components of their recognition pipelines *from data*. Naturally, this has placed increasing demands on the quantity and quality of data available for learning. Thus, the status quo in *computational* visual recognition is to learn from large datasets of manually curated images annotated by human workers. For example, today's popular datasets for visual learning comprise millions of manually curated images annotated by human workers through tens of thousands of hours of crowdsourced labor costing up to hundreds of thousands of dollars.

On the other hand, converging evidence from disciplines like cognitive science, psychology, and biology indicate that visual perception in *biological systems* develops very differently. Vision in nature develops in the context of acting and moving in the world, where the visual experience is inextricably tied to the motor activity behind it. Given the long history of biologically inspired scientific advances, this evidence raises the question: could modern computer vision systems benefit from such "embodied" learning too?

An agent continuously acting, moving and monitoring its environment has many avenues of knowledge available to it, going well beyond what can be learned from observing only orderless "bags of images" with category labels like today's standard datasets. As an example, such an agent may exploit ordered image sequences, i.e., its observed continuous video stream, with freely available relationships among images or between images and other sensor streams. It may act deliberately to affect its environment and then use the observed results of those actions as a form of self-acquired supervision. Such forms of supervision may allow agents to discover knowledge not available through the standard supervised paradigm.

In this PhD thesis, I aim to show that these and other advantages of *embodied* vision systems may allow large improvements over the current status quo across a wide range of standard visual recognition tasks. I proceed in five stages. First, I propose a new answer to the question: how can unlabeled video augment visual learning? Our solution, which is based on the temporal smoothness of the visual world, generalizes and outperforms a large body of existing approaches. Second, I show the advantages of augmenting such continuous visual streams with proprioceptive "motor signal" data streams. With such proprioceptive knowledge, it becomes possible to learn to predict the effects of one's own actions, and I show that this induces general visual abilities. Third, I show how to exploit the setting where at training time, an agent has access not only to the effect of one action, but to the effects of all possible actions.

Starting in the fourth stage of my thesis, I examine the visual recognition problem as it applies to agents that are embodied at *testing time* as well as at

training time. Embodied agents can not only probe their environments to acquire supervision without manual intervention for visual learning, but also acquire new observations to better perform visual tasks during deployment *after* training. Thus in this new setting, I study: how can an agent learn appropriate behaviors to acquire the most useful visual observations at test time? First, I investigate the learning of such intelligent data-directed action in the context of a supervised categorization task. I show how it is possible to train an end-to-end system that optimizes agent behaviors for category recognition, and further, that such a system benefits from being simultaneously trained to predict the effects of its actions. Finally, in the fifth stage of my thesis, I move back to an unsupervised setting, and show that it is possible to learn exploratory or curious data acquisition behaviors that seek generically informative observations without training on a specific supervised task.

Throughout, I evaluate these proposed methods on challenging datasets against a variety of previously proposed state-of-the-art approaches and other pertinent baselines. Our key empirical findings include the following: (1) Unlabeled web video is often competitive with large manually labeled image sets as a data source for learning visual representations. (2) Egomotion can aid visual learning, e.g., video data from a car-mounted camera, with accompanying registered steering data can significantly improve visual learning for generic scene recognition. (3) Embodied systems deployed for scene recognition tasks after training can also enjoy significant additional advantages by exploring the scenes through intelligently planned motions. Our results serve as compelling evidence of the benefits of studying and learning to perform visual perception in embodied settings.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction and Overview

Computational visual recognition is the area of artificial intelligence that deals with the broad problem of allowing a machine to *understand* images and videos in the myriad ways that humans do. Early attempts in this field have focused on fundamental problems like identifying previously seen object instances (instance recognition), or identifying the *category* to which a newly observed object or scene belongs (category recognition). While the first attempts towards solving these problems employed heuristic handwritten-rule-based methods, research in recognition since the mid-nineties has come to be dominated by discriminative machine learning techniques that learn semantic concepts in a data-driven way. To train a typical recognition system, images relevant to the target task are first manually captured or downloaded from the Web, and then further annotated by humans to indicate the desired outputs of the system. In particular, within the duration of my graduate studies, great progress in these tasks has been driven by the advent of neural network-based "deep" machine learning techniques that learn feature spaces jointly with their relationships to semantic concepts.

Thus, the trajectory of visual recognition research over the last fifty or so years of the field's existence can be seen as gradually moving away from the first,

entirely hand-coded rule-based methods, to methods that *learn* increasingly more and more components of their recognition pipelines *from data*. Naturally, this places increasing demands on the quantity and quality of data available to learn from. This has led to a situation where the status quo in *computational* visual recognition is to train systems on large datasets of millions of manually curated images annotated by human workers through tens of thousands of hours of crowdsourced labor costing up to hundreds of thousands of dollars [35, 91, 92, 103, 162, 192].

As computer vision systems get better at tasks like category recognition, the focus within the vision research community is shifting to other aspects of visual intelligence and its role in general artificial intelligence. In the only context in which we have observed intelligence, the biological context, the emergence of intelligence and superior visual abilities in different families of animals has each time been closely tied to the emergence of the ability to move and act in their environments [58, 106, 119, 130]. Cognitive scientists have also empirically verified that self-generated motions are critical to the development of visual perceptual skills in animals [69], and a sizeable research program studies "embodied cognition" — the hypothesis that cognition is strongly influenced by aspects of an agent's body beyond the brain itself [53, 54, 173].

Progress in standard visual recognition tasks in the last few years has been fueled by access to today's largest painstakingly curated and manually labeled datasets [35, 91, 92, 103, 162, 192]. However, intuitively and on the basis of the evidence cited above, certain kinds of knowledge are available to an agent continuously, acting, moving, and monitoring its environment that are not available from observing only

orderless, independent, and identically distributed (i.i.d.) "bags of images" with category labels like today's standard datasets. For instance, such an agent may exploit *ordered image sequences*, i.e., its observed video stream, with freely available relationships among visual observations over time or between visual and other sensor streams. An agent that is able to act or move in its environment also has open to it the possibility of using the observed results of its actions as a form of self-acquired "supervision by experimentation", e.g., it may tap an object to determine its material properties, walk around it to observe a less ambiguous view of it, or learn natural world physics by dropping, pushing or throwing objects and learning to anticipate their behavior. Thus these forms of supervision may allow a system to discover new forms of knowledge and perform tasks that the standard manually supervised paradigm is poorly suited for.

Moreover, alleviating the non-scalable and expensive curation and labeling requirements involved in compiling today's standard datasets is a worthwhile goal in itself, so that all or most visual learning may happen without manual supervision. Replacing manual supervision with alternative forms of supervision in this manner would have many advantages: (1) it would open up the possibility of exploiting much larger datasets for visual learning. This could potentially drive even better-performing computer vision systems for conventional tasks, since the evidence over the last few years has suggested that visual learning benefits from ever-higher capacity models trained on ever-larger datasets, (2) it would enable easy development of visual applications for more narrow, non-standard domains for which large labeled datasets neither currently exist, nor are likely to be curated in the future, such as

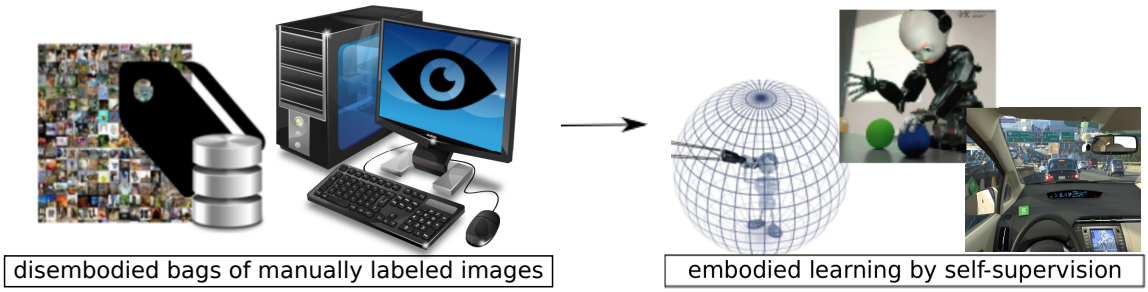


Figure 1.1: Today's standard vision approaches are trained with manually supervised learning from disordered, disembodied "bags of labeled images". In this dissertation, I aim to move these to *embodied* settings where the visual agent is able to monitor, act on and move in a world, and affect its own future observations of the world, both during learning and during eventual deployment.

say, vision for an inter-planetary rover, and (3) compared to standard supervised learning, it more closely resembles what we know about the mechanisms of learning in the biological visual systems we hope to ultimately emulate.

While the computer vision enterprise in its beginnings was very much motivated by the end-goal of enabling autonomously acting agents in the real world, its aforementioned trajectory towards learning-based supervision-hungry methods have engendered a change in this focus: nearly all of computer vision today deals primarily with disembodied off-the-shelf datasets of human-captured images and video, restricting its applications. While the field has made much progress on perception in this limited setting, through this thesis, I argue that our vision and learning techniques are now at a stage of maturity where it is fruitful to once more consider the problems of vision not just on stationary computers, but in the much larger class of real-world embodied agents (see Figure 1.1). This prompts some key questions: How must embodied agents learn to perceive their sensory inputs? How must they

act intelligenty towards a goal? How must they explore, observe, and inspect their surroundings to acquire their own observations?

To help motivate my thesis research, consider the famous "kitten carousel" experiment (1963) in which psychologists Held and Hein examined visual learning in kittens [69]. They designed a carousel-like apparatus in which two kittens could be harnessed. For eight weeks after birth, the kittens were kept in a dark environment, except for one hour a day on the carousel. One kitten, the "active" kitten, could move freely of its own volition while attached. The other kitten, the "passive" kitten, was carried along in a basket and could not control his own movement; rather, he was forced to move in exactly the same way as the active kitten. Figure 1.2 shows a schematic of the apparatus. Thus, both kittens had identical visual experiences. However, while the active kitten simultaneously experienced signals about his own motor actions, the passive kitten was simply along for the ride. It saw what the active kitten saw, but it could not simultaneously learn from self-generated motion signals.

The outcome of the experiment is remarkable. After eight weeks, the active kitten's visual perception was indistinguishable from kittens raised normally, whereas the passive kitten suffered fundamental problems. The implication is clear: proper perceptual development requires leveraging *self-generated movement in concert with continuous monitoring of visual feedback*. Specifically, the active kitten had two advantages over the passive kitten: (1) it had proprioceptive knowledge of the specific motions of its body that were causing the visual responses it was observing, and (2) it had the ability to select those motions in the first place. The results of this

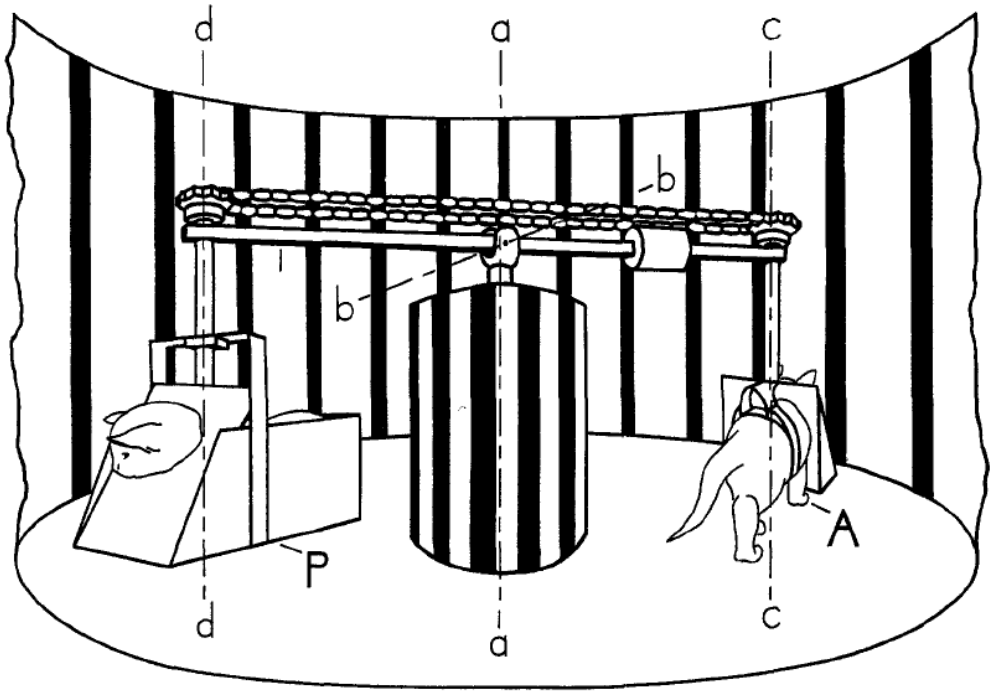


Figure 1.2: Schematic figure from [69] showing the apparatus for the kitten carousel study. The active kitten 'A' was free to move itself in both directions around the three axes of rotation a-a, b-b and c-c, while pulling the passive kitten 'P' through the equivalent movements around a-a, b-b and d-d by means of the mechanical linkages in the carousel setup.

experiment establish that these advantages are critical to the development of visual perception.

I contend that today's visual recognition algorithms are crippled much like the passive kitten. The culprit: learning from disembodied bags of labeled snapshots, which is the dominant paradigm today. Whether learning object categories, scene classes, body poses, or features themselves, the status quo idea is to discover patterns within an orderless collection of snapshots, blind to their physical source.

In my research, I have targeted this deficiency, moving learning methods for visual recognition from the "passive kitten" to the embodied "active kitten" scenario progressively, in several stages:

- Learning representations from *video*, rather than disordered bags of images, to embed temporal dynamics into image features [77]. (Chapter 3)
- Simultaneously exploiting observer egomotions during visual learning from videos [75, 78]. (Chapter 4)
- Learning image representations for object views by learning to hallucinate unobserved viewpoints from an arbitrary view [80]. (Chapter 5)
- Visual learning while simultaneously learning to *move*, i.e., active self-generated motion for vision [76]. (Chapter 6)
- Learning generic exploratory behaviors to intelligently inspect objects and scenes [79]. (Chapter 7)

I will spend the rest of this chapter introducing my work on each of these stages in turn, mainly discussing the motivating questions and high-level ideas but also briefly describing my solutions and highlighting some key results. Chapters 3, 4, 6, 5, and 7 will then cover each stage in detail, recapping the main motivations and then describing the methods and results in detail.

1.1 Learning steady representations encoding visual dynamics from video

Visual feature learning with deep neural networks has yielded dramatic gains for image recognition tasks in recent years [92, 149]. While the main techniques involved in these methods have been known for some time, a key factor in their recent success is the availability of large human-labeled image datasets like ImageNet [35]. Deep convolutional neural networks (CNNs) designed for image recognition typically have millions of parameters, necessitating notoriously large training databases to avoid overfitting.

As discussed above, however, visual learning should not be restricted to sets of category-labeled exemplars. Taking human learning as an obvious example, children build up visual representations through constant observation and action in the world. This hints that machine-learned representations would also be well served to exploit long-term *video* observations, even in the absence of deliberate labels. Indeed, researchers in cognitive science find that *temporal coherence* plays an important role in visual learning. For example, altering the natural temporal contiguity of visual stimuli hinders translation invariance in the inferior temporal cortex [100], and functions learned to preserve temporal coherence share behaviors observed in complex cells of the primary visual cortex [17].

As the first major contribution of my thesis research, I explore how to exploit unlabeled video, as might be obtained freely from the Web, to improve visual feature learning. In particular, I am interested in improving learned image representations

for visual recognition tasks.

Prior work leveraging video for feature learning focuses on the concept of *slow feature analysis* (SFA). First formally proposed in [174], SFA exploits temporal coherence in video as "free" supervision to learn image representations invariant to small transformations. In particular, SFA encourages the following property: in a learned feature space, temporally nearby frames should lie close to each other, i.e., for a learned representation $\mathbf{z}$ and adjacent video frames $\boldsymbol{a}$ and $\boldsymbol{b}$, one would like $\mathbf{z}(\boldsymbol{a}) \approx \mathbf{z}(\boldsymbol{b})$. The rationale behind SFA rests on a simple observation: high-level semantic visual concepts associated with video frames typically change only gradually as a function of the pixels that compose the frames. Thus, representations useful for recognizing high-level concepts are also likely to possess this property of "slowness". Another way to think about this is that scene changes between temporally nearby frames are usually small and represent label-preserving transformations. A slow representation will tolerate minor geometric or lighting changes, which is essential for high-level visual recognition tasks. The value of exploiting temporal coherence for recognition has been repeatedly verified in ongoing research, including via modern deep convolutional neural network implementations [15, 62, 64, 118, 168, 195].

However, existing approaches require only that high-level visual signals change slowly over time. Crucially, they fail to capture *how* the visual content changes over time. In contrast, our idea is to incorporate the *steady visual dynamics* of the world, learned from video. For instance, if trained on videos of walking people, slow feature-based approaches would only require that images of people in nearby poses be mapped *close* to one another. In contrast, we aim to learn a feature space in which

frames from a novel video of a walking person would follow a smooth, predictable trajectory. A learned *steady* representation capturing such dynamics would be influenced not only by object motions, but also other types of visual transformations. For instance, it would capture how colors of objects in the sunlight change over the course of a day, or how the views of a static scene change as a camera moves around it.

To this end, we propose *steady feature analysis* [77]—a generalization of slow feature learning. The key idea is to impose higher order temporal constraints on the learned visual representation. Beyond encouraging temporal coherence, i.e., *small feature differences* between nearby frame pairs, we would like to encourage *consistent feature transitions* across sequential frames. In particular, to preserve second order slowness, we look at triplets of temporally close frames $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, and encourage the learned representation to have $\mathbf{z}(\boldsymbol{b}) - \mathbf{z}(\boldsymbol{a}) \approx \mathbf{z}(\boldsymbol{c}) - \mathbf{z}(\boldsymbol{b})$. We develop a regularizer that uses contrastive loss over tuples of frames to achieve such mappings with CNNs. Whereas slow feature learning insists that the features not change too quickly, the proposed steady learning insists that—in whichever way the features are evolving—they *continue to evolve in that same way* in the immediate future. See Figure 1.3.

We hypothesize that higher-order temporal coherence could provide a valuable prior for recognition by embedding knowledge of the rich dynamics of the visual world into the feature space. We empirically verify this hypothesis using five datasets for a variety of recognition tasks, including object instance recognition, large-scale scene recognition, and action recognition from still images. In each case, by aug-

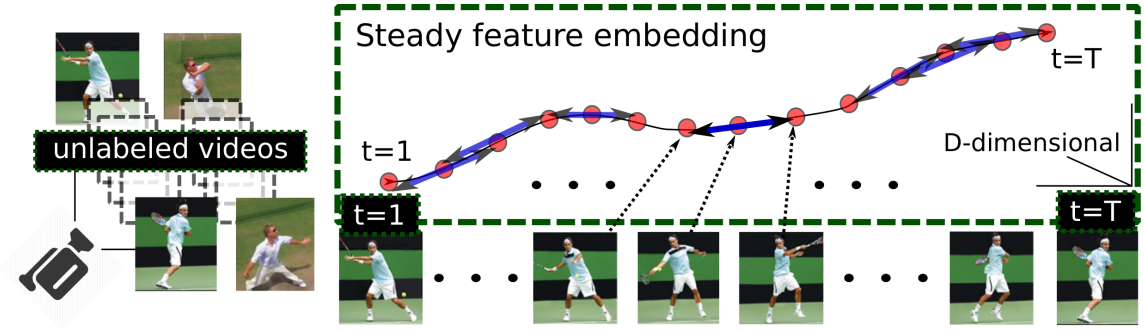


Figure 1.3: In Chapter 3, we exploit unlabeled videos to learn "steady features" that exhibit consistent feature transitions among sequential frames.

menting a small set of labeled exemplars with unlabeled video, the proposed method generalizes better than both a standard discriminative CNN as well as a CNN regularized with existing slow temporal coherence metrics [64, 118]. We further show that in the absence of all labeled data, our approach allows the learning of generically useful visual representations that are competitive and in some cases better than representations trained by standard supervised pretraining approaches. Our results reinforce that unsupervised feature learning from unconstrained video is an exciting direction, with promise to offset the large labeled data requirements of current state-of-the-art computer vision approaches by exploiting virtually unlimited unlabeled video.

Chapter 3 discusses the approach for this work in more detail and presents experiments and results. This work was published at CVPR 2016 [77].

1.2 Learning image representations tied to observer motion

Learning through constantly monitoring a visual stream, as above, is only the first step towards a solution for embodied vision. As we can see from the kitten carousel experiment, or in general from observing biological perceptual systems, vision in nature develops in the context of acting and moving in the world. Without leveraging the accompanying motor signals initiated by the observer, learning from video data does not escape the passive kitten's predicament.

Inspired by this concept, in Chapter 4, we propose to treat visual learning as an embodied process, where the visual experience is inextricably linked to the motor activity behind it.[1] In particular, our goal is to learn representations that exploit the parallel signals of egomotion and pixel appearance. As I will explain below, we hypothesize that downstream processing will benefit from access to such representations.

To this end, we attempt to learn the connection between how an observer moves, and how its visual surroundings change. We do this by exploiting motor signals accompanying unlabeled egocentric video, of the sort that one could obtain from a wearable platform like Google Glass, a self-driving car, or even a mobile phone camera.

To understand what we mean by learning the egomotion-vision connection, consider the "guess the new view" game, depicted in Figure 1.4(a). Given only one view of an object or a scene, the problem of computing what other views would look

[1]Depending on the context, the motor activity could correspond to either the 6-DOF egomotion of the observer moving in the scene or the second-hand motion of an object being actively manipulated, e.g., by a person or robot's end effectors.

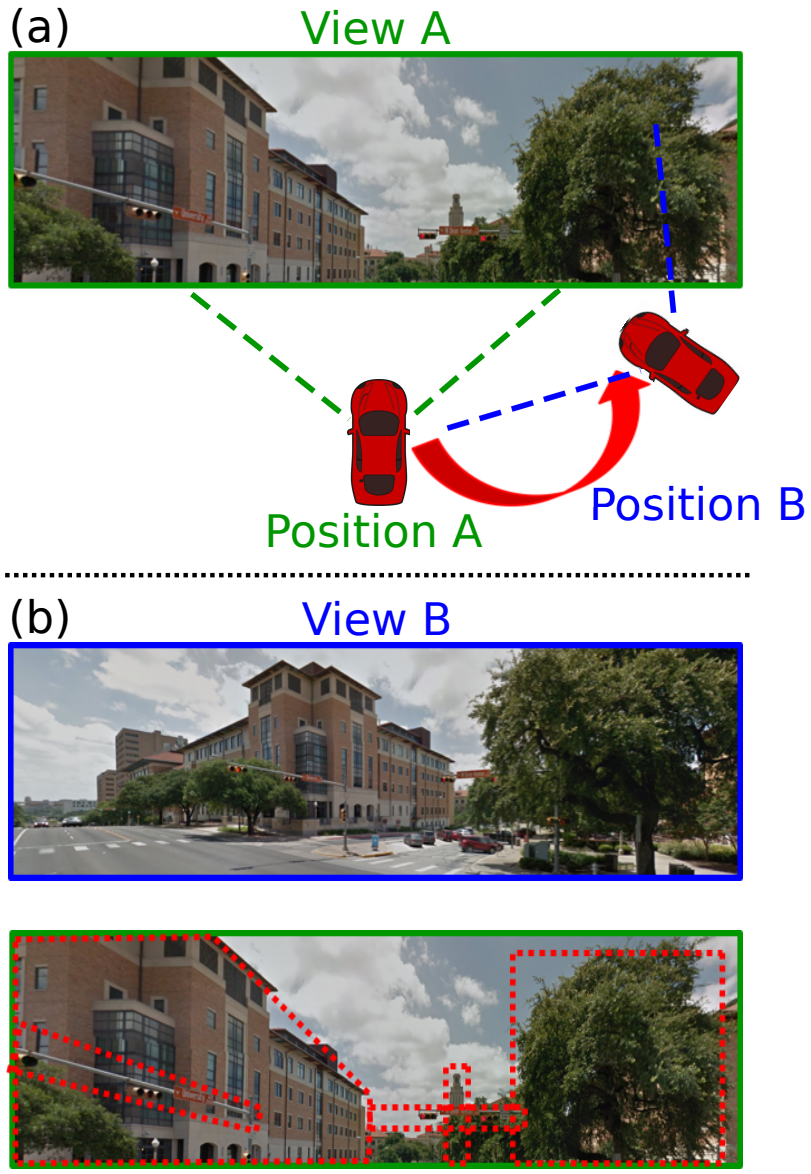


Figure 1.4: Guess the new view: (a) Given the view, View A, out of the windshield of the car when in position A, can you guess what the view (View B) would look like, when the car shifts to position B? (b) View B, the answer to (a), can be reasonably guessed from semantic, geometric, depth and contextual cues from View A, as shown in red outlines below view B. See text for explanation.

like is severely underdetermined. Yet, most often, humans are able to hallucinate such views. For instance, in the example of Figure 1.4(a), there are many hints in the first view that allow us to reasonably guess many aspects of the new view following the car's rotation. For instance, the traffic lights indicate that the observer must be at an intersection; the tree in the first view is probably closer to the camera than the tower, and will occlude the tower after the observer has moved; and it is even possible to extrapolate an entirely unseen face of the building using only geometric and semantic priors on the symmetry of buildings. The true view from the new position is shown in Figure 1.4(b).

We hypothesize that learning to solve this egomotion-conditioned view prediction task may help visual learning. As shown in the example above, view prediction draws on complex visual skills such as semantics (recognizing "building", "tree", "tower", and so on), depth and 3D geometry (the "tree" and the "tower"), and context ("traffic lights" $\Rightarrow$ "intersection"). These are general visual skills that are not limited to the view prediction task, but instead transfer well to many other tasks, including recognition. Moreoever, view prediction offers a way to acquire these skills entirely without manual labels.

As the second major contribution of my thesis research, I exploit this fact by incorporating the view prediction task above into an unsupervised *equivariant* feature learning approach using egocentric video and motor signals [75, 78]. During training, the input image sequences are accompanied by a synchronized stream of ego-motor sensor readings; however, they need not possess any semantic labels. The ego-motor signal could correspond, for example, to the inertial sensor measurements

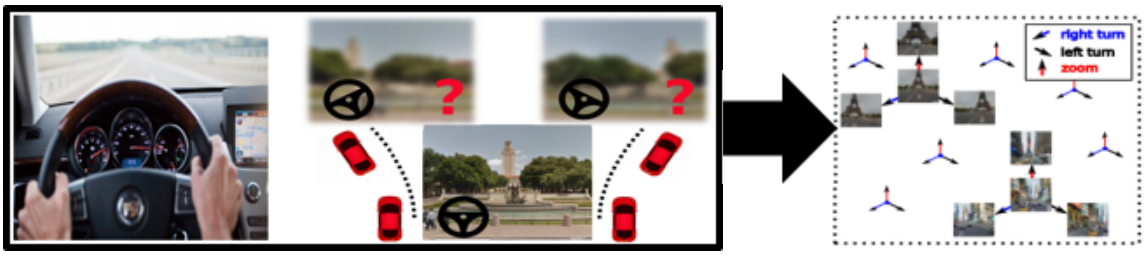


Figure 1.5: In Chapter 4, we propose an embodied approach to learn image embeddings from unlabeled video. Starting from egocentric video together with observer egomotion signals, we train a system on a "view prediction" task (left), to learn *equivariant* visual features that respond predictably to observer egomotion (right). In this target equivariant feature space, pairs of images related by the same egomotion are related by the same *feature transformation* too.

received alongside video on a wearable or car-mounted camera. The objective is to learn a feature mapping from pixels in a video frame to a space that is equivariant to various motion classes. In other words, the learned features should *change in predictable and systematic ways as a function of the transformation applied to the original input.* See Figure 1.5. We develop a convolutional neural network (CNN) approach that optimizes a feature map for the desired egomotion-based equivariance. To exploit the features for recognition, we augment the network with a classification loss when class-labeled images are available. In this way, egomotion serves as side information to regularize the features learned, which we show facilitates category learning when labeled examples are scarce.

In sharp contrast to our idea, previous work on visual features—whether hand-designed or learned—primarily targets feature *invariance* [15, 42, 62, 118, 147, 148, 151, 164, 168, 195]. Invariance is a special case of equivariance, where transformations of the input have no effect. Typically, one seeks invariance to small transformations,

e.g., the orientation binning and pooling operations in SIFT/HOG and modern CNNs both target invariance to local translations and rotations. While a powerful concept, invariant representations require a delicate balance: "too much" invariance leads to a loss of useful information or discriminability. In contrast, more general equivariant representations are intriguing for their capacity to impose structure on the output space without forcing a loss of information. Equivariance is "active" in that it exploits observer motor signals, like Hein and Held's active kitten.

The main contribution of this component of my thesis is a novel feature learning approach that couples ego-motor signals and unlabeled video. To the best of my knowledge, ours is the first attempt to ground feature learning in physical activity. The limited prior work on unsupervised feature learning with video [62, 114, 118, 133, 168] learns only passively from observed scene dynamics, uninformed by explicit motor sensory cues. Furthermore, while equivariance is explored in some recent work, unlike our idea, it typically focuses on 2D image transformations as opposed to 3D egomotion [88, 141] and considers existing features [98, 163]. Finally, whereas existing methods that learn from image transformations focus on view synthesis applications [71, 94, 114], we explore recognition applications of learning jointly equivariant and discriminative feature maps.

We apply our approach to three public datasets. On pure equivariance as well as recognition tasks, our method consistently outperforms the most related techniques in feature learning. In the most challenging test of our method, we show that features learned from video captured on a vehicle can improve image recognition accuracy on a disjoint domain. In particular, we use unlabeled KITTI [51, 52] car

data to regularize feature learning for the 397-class scene recognition task for the SUN dataset [177]. Our results show the promise of departing from the "bag of images" mindset, in favor of an embodied approach to feature learning.

Chapter 4 discusses the approach for this work in more detail and presents experiments and results. This work first appeared at ICCV 2015 [75], and was expanded in [78].

1.3 Unsupervised learning through one-shot image-based shape reconstruction

In Section 1.2 above, I introduced our approach to learn image representations from video captured together with registered egomotion information. Since the setting uses precaptured off-the-shelf egocentric video, it is restricted to using only the useful content that was incidentally captured in that video. For instance, a video feed from a car-mounted camera might only have two views of a building, leaving much of the building unobserved. In contrast, an agent acting and learning in the real world is not restricted to using a small number of views. In the next component of this dissertation, I explore the setting where, rather than one video in each training environment from which sparse, useful views are mined and exploited for learning, the agent instead has access to full "viewgrids" consisting of all possibly useful views. I will now motivate and introduce our approach that takes advantage of such "viewgrids" to learn representations that explicitly encode 3D information from only a single 2D view. In a sense, this approach generalizes the ideas of Sec 1.2 to the setting where the learning agent can explore all egomotions from a given position

during training.

While visual perception relies largely on 2D observations, the visual world and objects in particular are inherently three dimensional entities. Inferring 3D geometry from 2D views is therefore a necessary component of object perception. Cognitive psychologists have found strong evidence supporting this view. In a seminal series of discoveries [146], Shepard and collaborators observed that humans attempting to determine if two views portrayed the same abstract 3D shape spent time that was linearly proportional to the 3D angular rotation between those views. Further, they showed that the time taken was independent of whether the 3D shape underwent simple in-plane rotations in the picture plane, or more complex out-of-plane rotations. These findings are striking. In particular, they suggest that humans explicitly form mental representations of 3D shape from individual 2D views, and suggest that the act of mentally rotating such representations is integral to registering object views, and by extension, to object perception and recognition.

Inspired by this, we ask the question: can we learn image representations encoding geometry from 2D views, for use in computer vision systems? We propose to do this by training a system for the same target task as was set for the humans in Shepard *et al*'s study: mental rotation.

More concretely, we set up the task of single view image-based shape reconstruction. Given one 2D view of an object from an arbitrary viewpoint, the system must output views corresponding to arbitrary rotations from the current viewpoint. This task is illustrated in Figure 1.6.

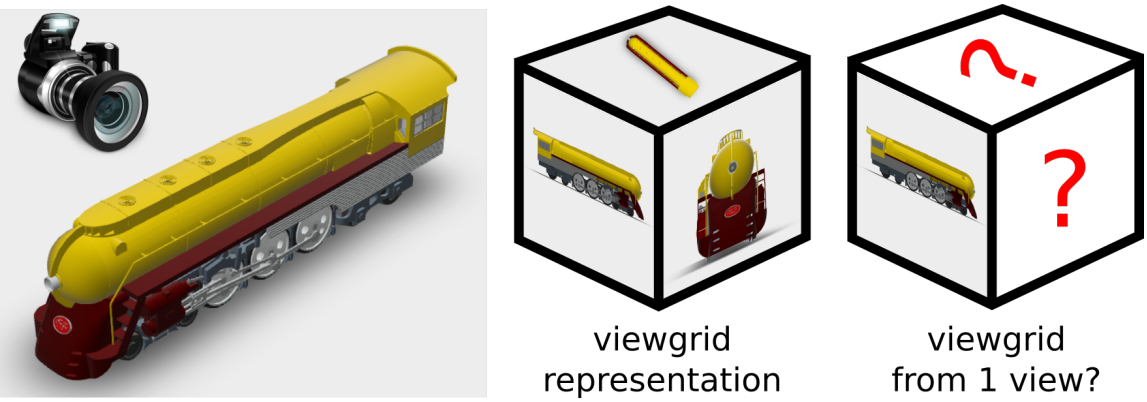


Figure 1.6: One-shot image-based shape reconstruction (Chapter 5): A 3D shape is represented by its "viewgrid" — views from evenly spaced viewpoints. Given one 2D view of an arbitrary shape, we train a deep network to produce the remaining views in the viewgrid and ask: does this training induce transferable representations in the network? (note: this schematic shows 6 views around a solid object arrange in a cube for easy illustration; in practice, our viewgrids are more densely sampled (details in Chapter 5).

With an infinite number of views from all around an object, classic geometric methods allow full 3D shape recovery [67]. These methods can operate with very limited object understanding, as they are completely agnostic to the semantics of the object, and work for arbitrary shapes that might not even represent real-world objects. With real world objects however, 3D understanding is possible from much sparser observations by using cues such as semantics and shading. This suggests the use of learning-based approaches for reconstruction.

Vision researchers have recently begun to investigate single view 3D reconstruction by employing deep learning methods [31, 43, 56, 84, 135, 159, 175, 183, 193]. However, these few prior attempts (1) all target reconstruction as an end task in itself, (2) largely build category-specific models for common categories (e.g., chairs, cars, faces), and (3) rely on deep neural networks pretrained heavily on manually

supervised classification tasks to yield better reconstructions. In contrast, we wish to investigate this task as a means to learn generic visual representations that embed knowledge of 3D shape properties from arbitrary object views. As suggested above, we hypothesize that downstream vision tasks such as recognition would benefit from an image representation that "lifts" 2D views of objects to inferred 3D shapes.

To this end, we train a category-agnostic deep feed-forward neural network from scratch, to produce a complete image-based shape representation given a single view of a generic object, in one forward pass. In order to perform this one-shot reconstruction task, the network must learn a representation that encodes knowledge of the full 3D object shape. Therefore, after training our deep neural network model on this unsupervised task, we extract representations from intermediate layers in the model to use as input features for recognition tasks.

Through experiments on widely used shape datasets and comparison against various baselines, we validate that (i) our system successfully learns the training task of category-agnostic image-based shape reconstruction, generalizing even to categories that were not seen in the training set, and (ii) the representations learned in the process are generically useful, in particular transferring well to the semantic tasks of object recognition and retrieval. Our results establish the promise of explicitly targeting 3D understanding as a means to learn generically useful visual representations.

Chapter 5 discusses our one-shot reconstruction approach in detail and presents our experiments and results. This work is currently under review [80] and will be made public in August 2017.

1.4 End-to-end active visual category recognition

While in the above setting, the system has *knowledge* of the camera motions in precaptured third-party egocentric videos or in full grids of views of 3D objects, the fourth component of this dissertation considers the setting where the system has not only knowledge but also *control* over the camera movements for the acquisition of data during training as well as testing. This setting is meant to represent visual recognition systems mounted on autonomous moving agents.

People consistently direct their senses in order to better understand their surroundings. For example, one might swivel around in an armchair to observe a person behind him, rotate a coffee mug on his desk to read an inscription, or walk to a window to observe the rain outside.

In sharp contrast to such scenarios, recent recognition research has been focused almost exclusively on static image recognition: the system takes a single snapshot as input, and produces a category label estimate as output. As discussed above, the ease of collecting large labeled datasets of images has enabled major advances on this task in recent years, as evident for example in the striking gains made on the ImageNet challenge [137]. Yet, despite this recent progress, recognition performance remains low for more complex, unconstrained images [103].

To illustrate the problem, Figure 1.7 shows some examples of web images and images captured by a human head-mounted camera that was not explicitly controlled to capture well-framed images. Recognition systems mounted on autonomous moving agents acquire unconstrained visual input which may be difficult to recognize

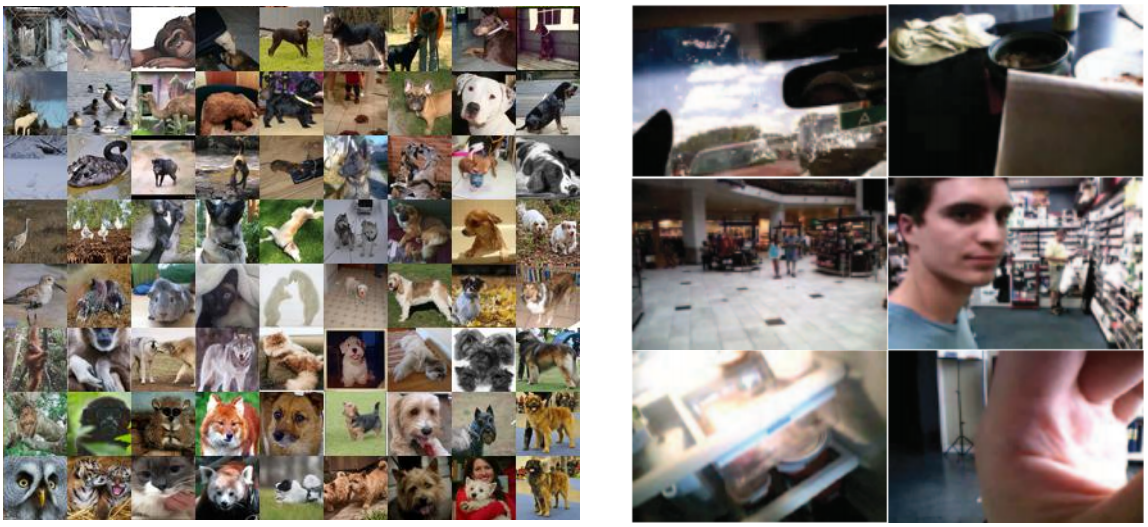


Figure 1.7: Examples of web images from ImageNet [137] (left) and randomly chosen frames from a human head-worn camera [97] (right). Cameras mounted on autonomous agents typically acquire ill-framed images that can be very hard to recognize one frame at a time, compared to web images which are human-captured and usually capture important content prominently in the foreground. However, autonomous moving visual agents can direct their cameras to acquire multiple views. Our active recognition approach employs reinforcement learning to learn policies to intelligently acquire views to facilitate scene and object category recognition.

effectively, *one frame at a time*. However, similar to the human actor in the opening examples above, such systems have the opportunity to improve their performance by moving their camera apparatus or manipulating objects to acquire new information, as shown in Figure 1.8. This control of the system over its sensory input has tremendous potential to improve its recognition performance. While such mobile agent settings (such as mobile robots and autonomous vehicles) are closer to reality today than ever before, the problem of *learning to actively move* to direct the acquisition of data remains underexplored in modern visual recognition research.

The problem we are describing fits into the realm of *active vision*, which has a

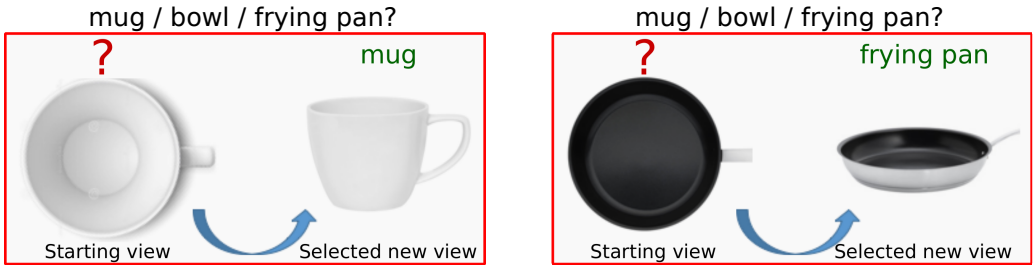


Figure 1.8: A schematic illustrating the active categorization of two objects. A moving vision system may not recognize objects after just one view, but may intelligently choose to acquire new views to disambiguate amongst its competing hypotheses.

rich history in the literature (e.g., [4, 7, 27, 38, 140, 171]). Active vision offers several technical challenges that are unaddressed in today's standard passive scenario. In order to perform active vision, a system must learn to intelligently direct the acquisition of input to be processed by its recognition pipeline. In addition, recognition in an active setting places different demands on a system than in the standard passive scenario. To take one example, "nuisance factors" in still image recognition—such as pose, lighting, and viewpoint changes—become *avoidable* factors in the active vision setting, since in principle, they can often be overcome merely by moving the agent to the right location.

This calls for a major change of approach. Rather than strive for invariance to nuisance factors as is the standard in static image recognition, an intriguing strategy is to learn to *identify when conditions are non-ideal for recognition* and to *actively select the correct agent motion* that will lead to better conditions. In addition, recognition decisions must be made based on intelligently fusing evidence from multiple observations. Figure 1.9 illustrates a generic active recognition pipeline.

I contend that these three functions of an active vision system—control, per-

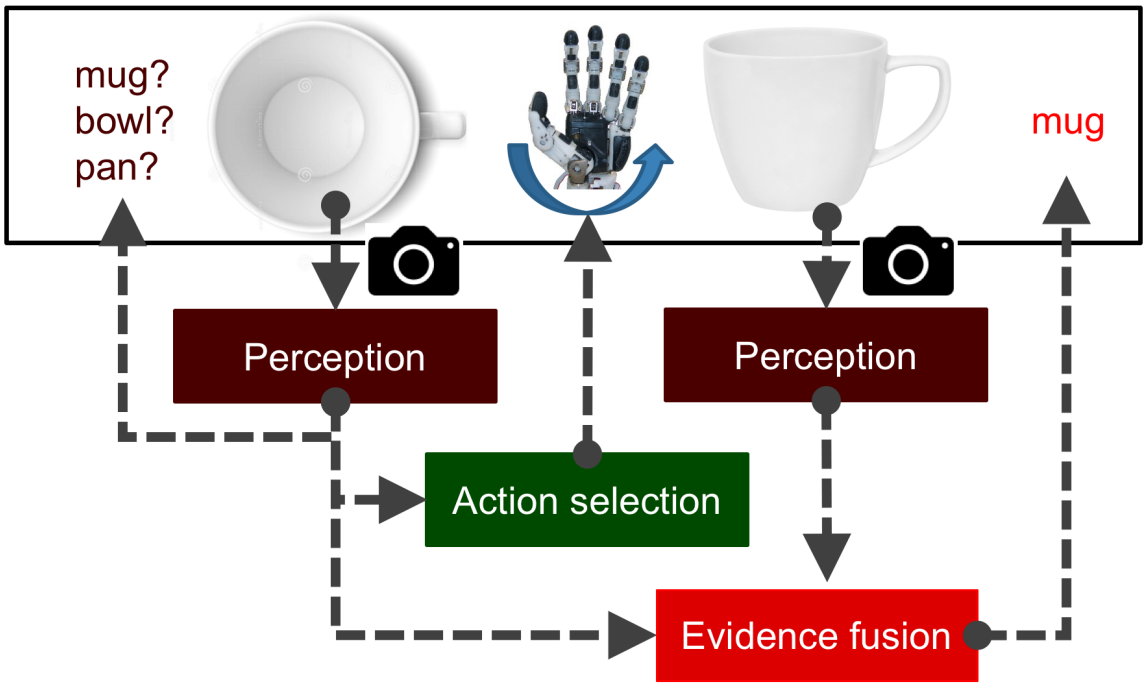


Figure 1.9: A generic active recognition pipeline illustrating the three functions of an active vision system—control, per-view perception, and evidence fusion. We aim to learn all three functions jointly and end-to-end.

view perception, and evidence fusion—are closely intertwined, and must be tailored to work together. In particular, as the first contribution of this paper, we propose to learn all three modules of an active vision system simultaneously and end-to-end. We employ a stochastic neural network to learn intelligent motion policies (control), a standard neural network to process inputs at each timestep (per-view perception), and a modern recurrent neural network (RNN) to integrate evidence over time (evidence fusion). Given an initial view and a set of possible agent motions, our approach uses reinforcement learning to learn how to move in the 3D environment to produce accurate categorization results.

Additionally, I hypothesize that motion planning for active vision requires an agent to internally "look before it leaps". That is, it ought to simultaneously reason about the effect of its motions on future inputs. To demonstrate this, as a second contribution of this active recognition component of my thesis, we jointly train our active vision system to have the ability to predict *how its internal representation of its environment will evolve* conditioned on its choice of motion. As I will explain below, this may be seen as preferring equivariance, i.e., predictable feature responses to pose changes, rather than invariance as is standard in passive recognition pipelines. This builds upon the ideas introduced above in Section 1.2, where I introduce our method for exploiting equivariance to observer motions as a useful inductive bias for feature learning.

Through experiments on two datasets, we validate both our key ideas: (1) RNN-based end-to-end active categorization and (2) learning to forecast the effects of self-motion at the same time one learns how to move to solve the recognition task. We study both a scene categorization scenario, where the system chooses how to move around a previously unseen 3D scene, and an object categorization scenario, where the system chooses how to manipulate a previously unseen object that it holds. Our results establish the advantage of our end-to-end approach over both passive and traditional active methods.

Chapter 6 discusses our approach in more detail and presents our experiments and results. This work appeared at ECCV 2016 [76].

1.5 Learning to look around

In the last section, I introduced our approach for learning to select observations most informative to a supervised and pre-specified object or scene category recognition task. In this final component of my thesis, I ask: can an agent learn curiosity-driven, exploratory "look around" behaviors *in the absence of any supervision*, and which are not specific to any one task, but instead generically useful?

As discussed in Section 1.4, visual perception requires not only making inferences from observations, but also making decisions about *what to observe*. Individual views of an environment afford only a small fraction of all information relevant to a visual agent. For instance, an agent with a view of a television screen in front of it may not know if it is in a living room or a bedroom. An agent observing a mug from the side may have to move to see it from above to know if it is empty. An agent surveying a rescue site may need to explore at the onset to get its bearings.

In principle, complete certainty in perception is only achieved by making every possible observation—that is, looking around in all directions, or systematically examining all sides of an object—yet observing all aspects is often inconvenient if not intractable. In practice, however, not all views are equally informative. The natural visual world contains regularities, suggesting not every view needs to be sampled for near-perfect perception. For instance, humans rarely need to fully observe an object to understand its 3D shape [86, 152, 153], and one can often understand the primary contents of a room without literally scanning it [161]. Given a set of past observations, some new views are more useful than others. This leads us to investigate the question: *how can a learning system make intelligent decisions about*

how to acquire new exploratory visual observations?

Today, much of the computer vision literature deals with inferring visual properties from a fixed observation. For instance, there are methods to infer shape from multiple views [67], depth from monocular views [139], or category labels of objects [92]. The implicit assumption is that the input visual observation is already appropriately captured. We contend that this assumption neglects a key part of the challenge: intelligence is often required to obtain proper inputs in the first place. Arbitrarily framed snapshots of the visual world are ill-suited both for human perception [44, 126] and for machine perception [5, 179]. Circumventing the acquisition problem is only viable for passive perception algorithms running on disembodied stationary machines, which are tasked only with processing human-captured imagery.

In contrast, we are interested in *learning to observe* efficiently—a critical yet understudied problem for autonomous embodied visual agents. An agent ought to be able to enter a new environment or pick up a new object and intelligently (non-exhaustively) look around. This capability would be valuable in both task-driven scenarios (e.g., a drone searches for signs of a particular activity) as well as scenarios where the task itself unfolds simultaneously with the agent's exploratory actions (e.g., a search-and-rescue robot enters a burning building and dynamically decides its mission).

We address the general setting, where exploration is not specialized to one task (as in the active classifier introduced in Section 1.4), but should benefit perception tasks in general. To this end, we formulate the learning objective as *active observation completion*: a system must intelligently acquire a small set of observa-

tions, so that it is then able to hallucinate all other possible observations. The agent continuously updates its internal model of a target scene or 3D shape based on all previously observed views. The goal is not to produce photorealistic predictions, but rather to represent the agent's evolving internal state. Its task is to select actions leading to new views that will efficiently complete its internal model. Posing the active view acquisition problem in terms of observation completion has two key advantages: generality and low cost (label-free) training data. It is also well-motivated by findings that infants' abilities to actively manipulate and inspect objects correlates with learning to complete 3D shapes [153].

We develop a reinforcement learning solution for active visual completion. Our approach uses recurrent neural networks to aggregate information over a sequence of views. The agent is rewarded based on its predictions of unobserved views.

We explore two applications of our idea. See Figure 1.10. In the first, the agent scans an omnidirectional natural scene through its limited field of view camera; here the goal is to select efficient camera motions so that after a few glimpses, it has modeled unobserved portions of the scene well. In the second, the agent manipulates a 3D object to inspect it; here the goal is to select efficient manipulations so that after only a small number of actions, it has a full model of the object's 3D shape. In both cases, the system must learn to leverage visual regularities (such as shape primitives and context) that suggest the likely contents of unseen views, focusing actions on portions that are difficult to hallucinate.

Chapter 7 discusses our one-shot reconstruction approach in detail and presents

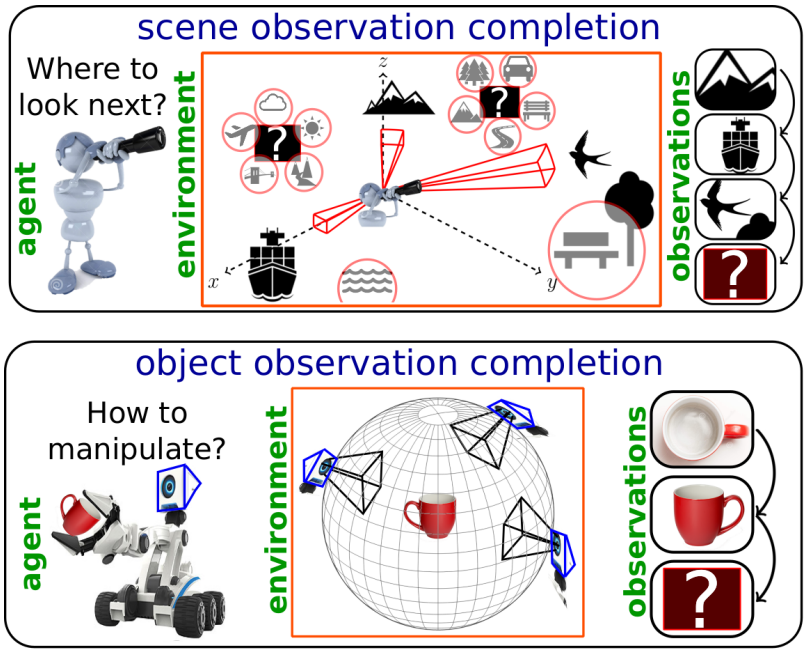


Figure 1.10: Looking around efficiently is a complex task requiring the ability to reason about regularities in the visual world using cues like context and geometry. (Left) An agent that has observed limited portions of its environment can reasonably hallucinate some unobserved portions (e.g., water near the ship), but is much more uncertain about other portions. Where should it look next? (Right) An agent inspecting a mug. Having seen a top view and a side view, how must it rotate the mug now to get maximum new information?

our experiments and results. This work [79] is currently under review and will be made public in June 2017.

Dissertation Roadmap: In the above, I have provided a bird's eye-view of the five components of my dissertation. Next, Chapter 2 reviews related work such as on unsupervised visual feature learning, sensorimotor feature learning, active vision, and learning-based reconstruction. After this, in chapters 3, 4, 5, 6, and 7, I discuss the

proposed methods outlined above in Sections 1.1, 1.2, 1.3, 1.4, and 1.5 respectively. Finally, Chapter 8 summarizes the key findings of my thesis work and highlights some interesting directions for follow-up research.

Chapter 2

Related Work

In this chapter, I review prior work that is related to the work presented in Chapters 3, 4, 5, 6, and 7. The material presented here aims both to set the stage to understand our proposed methods against their respective contexts, and to provide a useful starting point for readers interested in understanding the literature surrounding the topics of my thesis.

However, in later chapters, I will not rely on a careful reading of the material presented here. Where I deem prior work critical to a technical understanding of our proposed approaches, I will review them once again (and in greater detail) in the corresponding chapters, with the aim of keeping chapters self-contained to the extent possible.

2.1 Vision from/for motion and action

Concurrently with our work, and independently of it, a growing body of work [3, 19, 21, 29, 32, 99, 155, 170] has recently begun to study the interaction between high-level visual tasks and agent actions or motions during learning. Among these, some works [19, 29, 99, 170] focus on end-to-end learning of visual representa-

tions targeting action tasks such as driving. In [29], a neural network learns to drive a simulated car in a video game. In [99], end-to-end reinforcement learning trains robotic agents to perform simple tasks. For the active vision approach I develop in Chapter 6, perhaps the most conceptually relevant among these is [170]. Their method learns an image feature space in which optimal control actions can be computed as simple linear functions of current visual observations, with applications to simulated control tasks. In contrast, our approach in Chapter 6 employs an end-to-end scheme for learning embeddings that encode complete *histories* of observations and agent actions, with the aim of exposing this *temporally aggregated information* to an active visual recognition controller.

Sensorimotor feature embeddings [21, 32, 155] combine (possibly non-visual) sensor streams together with proprioception or other knowledge about the actions of the agent on which the sensors are mounted. In [21], dimensionality reduction and manifold learning are guided by proprioceptive knowledge, in similar spirit to our egomotion-equivariant features in Chapter 4, but only for simplistic toy scenarios. Sensorimotor embeddings are trained to reflect the geometry of an agent's environment in [155]. The theoretical beneficial properties of visual representations that vary linearly with observer motion are studied in [32]. These properties discussed in [32] are relevant both to our steady features (Chapter 3) where consecutive frame triplets are encouraged to be collinear, and to our egomotion-equivariant features (Chapter 4) where representations are trained to transform in "simple" ways in response to observer motion.

Finally, concurrently with our equivariant representation learning work in

Chapter 4 and independent of it, [3] also learns visual representations from video with associated egomotion sensor streams, but uses a different approach. Rather than learn to predict a new view in the feature space given the starting view and the egomotion as we do, their method learns to predict the *egomotion*, given the original and final views. Conceptually, while our approach explicitly targets a desired property, egomotion-equivariance, in the learned feature space, the method of [3] treats their egomotion-regression task as a generic proxy task for representation learning. We present empirical comparisons against their methods in Chapter 4.

2.2 Active perception

While in the last section we discussed work that uses actions at training time to enable the learning of perception or control tasks, we now discuss the "active vision" setting where an embodied agent deployed after training has the option of acting or moving to influence its observations.

The idea that a subject's actions may play an important role in perception can be traced back almost 150 years [22] in the cognitive psychology literature [7]. Recent surveys of relevant work can be found in [7, 10, 18]. "Active perception", the idea of exploiting *intelligent control strategies* (e.g., agent motion, object manipulation, and camera parameter changes) for *goal-directed data acquisition* to improve machine vision, was pioneered by [4, 9, 11, 171]. While most research in this area has targeted low-level vision problems [4, 11, 116] such as segmentation, structure from motion, depth estimation, optical flow estimation, or the "semantic search" task of object

localization [6, 49, 70, 150], approaches targeting *active recognition* are most directly related to our work in Chapter 6.

Most prior active recognition approaches attempt to identify during training those canonical or "special" views that minimize ambiguity among candidate labels [27, 36, 38, 140, 171]. At test time, such systems iteratively estimate the current pose, then select moves that take them to such pre-identified informative viewpoints. These approaches are typically applicable only to *instance* recognition problems, since broader categories can be too diverse in appearance and shape to fix "special viewpoints".

In contrast, our approach in Chapter 6 handles complex real-world categories. To the best of our knowledge, very little prior work attempts this challenging task of active *category* recognition (as opposed to instance recognition) [20, 132, 176, 185]. The increased difficulty is due to the fact that with complex real-world categories, it is much harder to anticipate new views conditioned on actions. Since new instances will be seen at test time, it is not sufficient to simply memorize the geometry of individual instances, as many existing active instance recognition methods effectively do.

Recently, in [176], information gain is used in view planning for active categorization. Their system learns to *predict* the next views of *unseen test objects* conditioned on various candidate agent motions starting from the current view by estimating 3D models from 2.5D RGBD images. It then estimates the information gain on its category beliefs from each such motion, and finally greedily selects the estimated most informative "next-best" move. While our idea for learning to predict action-conditional future views of novel instances is similarly motivated, we refrain

from explicit greedy reasoning about the next move. Instead, our approach uses reinforcement learning (RL) in a stochastic recurrent neural network to learn optimal *sequential* movement policies over multiple timesteps. The closest methods to ours in this respect are [125] and [110], both of which employ Q-learning [169] in feedforward neural networks to perform view selection, and target relatively simpler visual tasks compared to this work.

In addition to the above, an important novelty of our active vision approach in Chapter 6 is in learning the entire system end-to-end. Active recognition approaches must broadly perform three separate functions: action selection, per-instant view processing, and belief updates based on the history of observed views. While previous approaches have explored several choices for action selection, they typically train a "passive" per-instant view recognition module offline and fuse predictions across time using some manually defined heuristic [36, 38, 110, 132, 140]. For example, recently, a deep neural network is trained to learn action policies in [110] after pretraining a per-view classifier and using a simple Naive Bayes update heuristic for label belief fusion. In contrast, we train all three modules jointly within a single active recognition objective.

All of the active perception literature (including our work in Chapter 6) confines itself to the problem of selecting actions for specific pre-specified end goals. In Chapter 7, we go beyond this. Rather than target a recognition task, we aim to learn a data acquisition strategy useful to perception in general, hence framing it as active "observation completion". This allows us to learn from unlabeled viewpoint-calibrated observations rather than from manually labeled data.

Finally, with a few very recent exceptions [5, 110], the evaluation of active vision methods has been restricted to simplified, unrealistic (often virtual) settings, whereas we employ complex real-world imagery in our experiments in both Chapter 6 and 7, benchmarking our approach on realistic, yet repeatable and benchmarkable data against several previously proposed techniques.

2.3 Saliency and attention

Visual saliency and attention are also related to active vision [2, 8, 13, 66, 104, 117, 128, 143, 160, 181]. While active vision systems aim to form policies to acquire new data, saliency and attention systems aim to block out "distractors" in *existing* data by identifying portions of input images/video to focus on, often as a faster alternative to sliding window-based methods. Attention systems thus sometimes take a "foveated" approach [25, 117]. In contrast, in our settings in Chapters 6 and 7, the system never holds a snapshot of the entire environment at once. Rather, its input at each timestep is one portion of its complete physical 3D environment, and it must choose motions leading to more informative—possibly non-overlapping—viewpoints. Its decision is not where to focus within a current observation, but rather where to look for a *new* observation. Another difference between the two settings is that the focus of attention may move in arbitrary jumps (saccades) without continuity, whereas active vision agents may only move continuously.

Saliency is the main way in which the question of "looking around" generically has been posed in the computer vision literature. Our active observation completion

task in Chapter 7 may be thought of as providing an alternative definition for saliency or informativeness — those views are salient which are most useful to complete a full model of the environment. As such, in our experiments, we compare our policy for looking around against one driven by traditional saliency.

Sequential attention systems using recurrent neural networks in particular have seen significant interest of late [117], with variants proving successful across several attention-based tasks [8, 143, 181]. We adopt the basic attention architecture of [117] as a starting point for our model, and develop it further to accommodate the active vision setting, instill look-ahead capabilities, and select camera motions surrounding a 3D object that will most facilitate categorization.

2.4 Active visual localization and mapping

Active visual simultaneous localization and mapping (SLAM) aims to limit samples needed to densely reconstruct a 3D environment using geometric methods [34, 87, 89, 111, 154]. While related to our active observation completion setting in Chapter 7, this literature is different in several important ways. Beyond measuring uncertainty in the current scene, our learning approach capitalizes on learned context from previous experiences with *different* scenes/objects. While purely geometric methods are confined to using exactly what they see, and hence typically require dense observations, our approach can infer substantial missing content using semantic and contextual cues. Finally, our scope is broader than SLAM, in that it also applies to image-based panoramic scene completion.

2.5 Optimal sensor placement

The sensor placement literature studies how to place sensors in a distributed network to provide maximum coverage [37, 90, 167]. This is superficially similar to our active completion problem in Chapter 7. However unlike in our setting, the sensors are static, i.e., their positions are preset, and their number is fixed. Further, sensor placement is based on coverage properties of the sensors, whereas our model must *react* to past sequential observations obtained by its sensors. Further, whereas sensors can be arbitrarily spaced, our agent in Chapter 7 is constrained to move its sensor within small neighborhoods at each discrete timestep, to be realistic.

2.6 Unsupervised feature learning

Another recurring theme in this dissertation is the idea of learning "useful" image representations suitable for standard recognition tasks, without supervision. Recall from Chapter 1 that computer vision systems have come to rely largely on painstakingly curated and manually labeled bags of category-labeled snapshots [35, 91, 92, 103, 162, 192], costing up to hundreds of thousands of dollars. Throughout this dissertation, I ask: is this really necessary? Could the majority of visual learning instead be performed through observation and/or experimentation?

This idea of unsupervised visual learning is a very active area of recent research interest. Several papers [3, 24, 40, 48, 63, 64, 95, 118, 123, 166, 168, 189] have trained unsupervised image representations within deep neural networks.

Note that the word "unsupervised" in this context is used slightly differently from its traditional use in machine learning— here unsupervised refers only to a lack of deliberate human supervision and the training process may involve solving "supervised machine learning" problems like classification and regression, so long as the target labels or regression outputs are not manually specified. Throughout this dissertation, I will use "supervised" and "unsupervised" in this sense, unless otherwise specified.

Indeed, the standard method for unsupervised feature learning has become to set up unsupervised "proxy" or "surrogate" tasks. A deep network is optimized to perform this unsupervised task, and upon convergence, its internal representations are treated as the desired unsupervised image features, and tested for their usefulness as representations for other (typically supervised) tasks. For example, efficent generative codes for image synthesis are learned in [24]—here, image pixels are both the input and the target for a regression task. In [166], features are trained to predict pixel-level optical flow maps for video frames—again, optical flow is automatically computed from video, so no manual intervention is involved. Concurrently with our work, a method to learn features that vary linearly in time is proposed in [63], for the specific task of extrapolating future video frames given a pair of past frames. They report qualitative results for toy video frame synthesis. While our formulation in Chapter 3 also encourages collinearity in the feature space, our aim is to learn generally useful features from real videos without supervision, and we report results on natural image scene, object, and action recognition tasks.

In Chapter 4, we show how equivariant representations may be learned in an

unsupervised manner in a different setting, where the camera egomotions involved in capturing unlabeled video are known. A different formulation for learning unsupervised representations from egomotion-labeled video was proposed in [3], which we compare against empirically.

Finally, in Chapter 5, we examine yet another setting for unsupervised representation learning, where full image-based 3D shape knowledge is available for every training instance. Throughout, we compare these approaches against several previously proposed methods for learning unsupervised features.

2.7 Invariant visual representations

Invariance is known to be valuable for visual representations. To build a robust object recognition system, the image representation must incorporate some degree of invariance to changes in pose, illumination, and appearance. Descriptors like SIFT [107], HOG [33], and aspects of CNNs like pooling and convolution, are hand-designed for invariance to small shifts and rotations. Feature learning work aims to *learn* invariances from data [42, 147, 148, 151, 164]. Strategies include augmenting training data by perturbing image instances with label-preserving transformations (e.g., translation, scaling, and intensity scaling) [42, 148, 164], and inserting linear transformation operators into the feature learning algorithm [151]. A good representation will ensure that jittered versions originating from the same content all map close by in the learned feature space.

Related to the unsupervised feature learning theme discussed in Section 2.6, several works aim to induce invariance into learned representations as a useful prior. In particular, many attempts learn invariant image features by observing video, i.e., rather than hand-designing jittered content for data augmentation as described above, they rely on natural variations in factors such as pose, illumination, and occlusions occurring among nearby frames in video [15, 62, 118, 168, 195].

These works based on exploiting temporal coherence fall under the umbrella of "slow feature analysis" techniques. The idea is to impose requirements that the features being learned vary slowly over continuous video, since visual stimuli can only gradually change between adjacent frames. These methods either produce a holistic image embedding [15, 62, 64, 118], or else track local patches to learn a localized representation [48, 168, 195, 196]. Most methods exploit the learned features for object recognition [15, 118, 168, 195], while others employ them for dimensionality reduction [64] or video frame retrieval [62]. In [118], a standard deep CNN architecture is augmented with a temporal coherence regularizer, then trained using video of objects on clean backgrounds rotating on a turntable. The method of [15] builds on this concept, proposing the use of decorrelation to avoid trivial solutions to the slow feature criterion, with applications to handwritten digit classification. The authors of [62] propose injecting an auto-encoder loss and explore training with unlabeled YouTube video. Building on SFA subspace ideas [174], researchers have also examined slow features for action recognition [190], facial expression analysis [186], future prediction [165], and temporal segmentation [105, 121].

Our work presented in this dissertation generalizes invariant feature learning

along two directions. First, related to all the above methods, Chapter 3 describes our method that aims to learn features from unlabeled video. However, whereas all the past work aims to preserve feature *slowness*, our idea is to preserve *higher order* feature coherence, i.e., *steadiness*. Our learning objective is the first to move beyond adjacent frame neighborhoods, requiring not only that sequential features change gradually, but also that they change in a similar manner in adjacent time intervals.

Secondly, invariance may be seen as a special case of a more general property called equivariance, discussed in the next section. While invariant features discard all sensitivity to pose, illumination etc., equivariant features are instead *predictably* sensitive. As we will show, the steady features of Chapter 3 implicitly seek equivariance to common image transformations. Later, in Chapter 4, we exploit video coupled with ego-motor signals to explicitly target egomotion equivariance, since equivariant features offer conceptual and empirical advantages over purely invariant representations for several applications.

2.8 Equivariant representations

As discussed above, while invariant features are unresponsive to transformations, equivariant features respond in easily *predictable* ways to image transformations. Invariance discards information, while equivariance *organizes* it. Equivariant features can also be hand-designed or learned. For example, equivariant or "covariant" operators are designed to detect repeatable interest points [163]. Recent work explores ways to learn descriptors with in-plane translation/rotation equivariance [88, 141]. While the latter does perform feature learning, its equivariance prop-

erties are crafted for specific 2D image transformations. In contrast, our approach in Chapter 4 targets more complex equivariances arising from natural observer motions (3D egomotion) that cannot easily be crafted, and our method learns them from data.

Methods to learn representations with disentangled latent factors [71, 94] aim to sort properties like pose and illumination into distinct portions of the feature space. For example, the transforming auto-encoder learns to explicitly represent instantiation parameters of object parts in equivariant hidden layer units [71]. Such methods target equivariance in the limited sense of inferring pose parameters, which are appended to a conventional feature space designed to be invariant. In this sense, they are closer to methods that learn image transformations, as described above in Section 2.9. In contrast, our formulation in Chapter 4 encourages equivariance over the *complete* feature space; we show the impact as an unsupervised regularizer when training a recognition model with limited training data.

The work of [98] quantifies the invariance/equivariance of various standard representations, including CNN features, in terms of their responses to specified in-plane 2D image transformations (affine warps, flips of the image). We adopt the definition of equivariance used in that work, but our goal in Chapter 4 is entirely different. Whereas [98] quantifies the equivariance of existing descriptors, our approach learns a feature space that is equivariant.

2.9 Synthesis of transformed views and features

As discussed above, the equivariant features we target in Chapter 4 respond *predictably* to transformations. Somewhat related to this, there is recent interest in "visual prediction" problems in various contexts, often using CNNs [39, 47, 133, 165, 166, 176]. For example, one can train CNNs to predict the features of the next video frame conditioned on the last frame [165], or predict dense optical flow from a single image [166]. Recurrent networks can also learn to extrapolate videos based on previously observed frames [133]. These future frame prediction approaches do not attempt to reason about *causes* of view transformations, e.g., camera motions, similar to our steady feature analysis method described in Chapter 3. However, while these methods aim to predict future views in the fixed spaces of pixels or pretrained features, our steady feature analysis technique (Chapter 3) is based on the idea that feature spaces trained to make simple extrapolation of future views possible also acquire useful properties for recognition tasks.

Also related to some of the work in this dissertation is another class of methods that deals with inferring 3D from 2D observations. While 3D vision has long been tackled with geometry and densely sampled views [67], recent work explores ways to inject learning into the solution [31, 43, 56, 65, 94, 113, 114, 135, 175, 183, 193]. Several of these works aim to directly map a view [31, 56, 84, 135, 175] to a 3D representation of the object, such as a voxel occupancy grid. An out-of-category test in [183] shows the potential and challenges for direct 3D outputs. Instead, in Chapters 5 and 7, we target unseen *view synthesis*, where a 3D model is not directly learned, but maintained implicitly. Framing the problem as novel view synthesis has the advantage of readily available ground truth. Finally, in both Chapters 5

and 7, we treat the problem of inferring 3D from 2D views as a proxy or surrogate task— the reconstruction is not the end goal in itself, but training on reconsruction is hypothesized to induce generically useful feature and exploratory policy learning respectively.

Other recent work is more relevant to the work in this dissertation, targeting novel view synthesis [30, 39, 43, 47, 71, 81, 94, 159, 184, 193]. The goal is to infer an image as a function of a specified transformation or viewpoint. When provided with two 2D views, methods can learn to predict intermediate views—as, for example, in transforming autoencoders [71], deep stereo [47], deep morphing [81], and the learned similarity functions of [39]. Alternatively, when the input consists of a single view, methods can learn to render the observed object from new camera poses. A tensor completion approach organizes views by people's body poses and camera viewpoints in order to infer "missing" novel views [30]. The inverse graphics network of [94] generates new images of an object with varying pose and lighting, having learned a disentangled representation with a deep convolutional neural network (CNN). Related to this, recurrent convolutional encoder-decoder networks are explored for rendering rotated objects from a single input image [184]. Training with synthetic models, a generative CNN can produce images with specified object types, viewpoints, colors, etc. [43]. A learning objective based on "appearance flow" allows a CNN to learn how to map pixels in the input to its proper destination in a new target view [193]. A convolutional autoencoder architecture in [159] synthesizes views for various rotations from an observed view before combining them into a 3D model and refining the views further.

The main restriction of such view synthesis approaches is their category-specificity. Existing methods concentrate on object-specific models (e.g., chairs, cars, people) and/or 3D CAD objects, making them inapplicable to unseen categories and environments. In contrast, we consider *generalizing* synthesis across objects. To our knowledge, ours are the first view synthesis results for unseen categories. Secondly, whereas all prior work learns to aggregate and extrapolate from passively captured views in one shot, our work in Chapter 7 is the first to consider active, sequential acquisition of informative views. Thirdly, our "observation completion" framework in Chapter 7 is more general than 3D object view synthesis— unlike any prior work, it considers inferring the appearance of complete 360 degree scenes. Finally, once again, in both Chapters 5 and 7, we learn to infer new views not as an end goal in itself but instead as a means to unsupervised feature and exploratory behavior learning.

This body of prior work on inferring novel views is also broadly related to our equivariance (Chapter 4) and lookahead (Chapter 6) approaches. The distinction is similar to the difference pointed out above between our steady feature analysis idea and future frame prediction approaches. While the these methods target view synthesis under specified transformations in the pixel space as an end goal in itself, our approaches aim to *learn* feature spaces such that view synthesis *in those learned feature spaces* is possible through simple operators, with the end goal being that the features thus learned prove useful for generic recognition tasks. Our look-ahead module in Chapter 6 goes further, learning to predict the evolution of *temporally aggregated* features—computed from a complete history of seen views—as a func-

tion of observer motion choices, and integrates this idea with the closely tied active recognition problem.

2.10 Egocentric vision

Egocentric video is that which is captured from the point-of-view of an observer moving and acting within a scene, such as through a head-mounted camera. We exploit such egocentric video recordings as natural data sources throughout this work to explore our ideas related to embodied vision. Recently, there is renewed interest in egocentric computer vision methods targeting applications such as egocentric video summarization [60, 97, 108] and activity recognition [46, 129, 138]. There is also work focusing on exploiting camera motion accompanying first person visual streams. For example, some recent methods use egomotion cues to separate foreground and background [134, 180] or infer the wearer's gaze [101, 182]. While most work relies solely on apparent image motion, the method of [180] exploits a robot's motor signals to detect moving objects and [120] uses reinforcement learning to form robot movement policies by exploiting correlations between motor commands and observed motion cues. None of these prior works perform feature learning using motor signals and pixels in concert as we propose for egomotion-equivariant feature learning in Chapter 4.

2.11 Image completion

In Chapter 7, we pose the task of "looking around" as observation completion. Completion tasks appear in other contexts within vision and graphics. Inpainting and texture synthesis fill small holes in images by modeling local textures (e.g., [50, 127]). Influential scene completion work [68] showed that larger holes can be filled by pasting appropriate regions from similar-looking scenes. Recent work explores unsupervised "proxy tasks" to learn useful deep visual representations, based on various forms of observation completion like inpainting and colorization [95, 127, 189]. Our observation completion setting differs from these in that 1) it requires agent *action*, 2) a much smaller fraction of the overall environment is observable at a time, 3) our target is a representation of multimodal beliefs, rather than a photorealistic rendering, and 4) completion is used to learn exploratory *behaviors* rather than features

Having briefly overviewed the key ideas of the five components of this thesis in Chapter 1, and reviewed relevant prior literature in this chapter, we now move to discussing the technical details of the approach together with results for each component in the upcoming chapters.

Chapter 3

Learning steady representations encoding visual dynamics from video

[1]In this chapter, I address the problem of replacing expensive large manually curated and labeled image datasets with unsupervised video for visual learning, which I introduced earlier in Section 1.1.

To recap, visual feature learning with deep neural networks has yielded dramatic gains for image recognition tasks in recent years [92, 149], largely due to the availability of large human-labeled image datasets like ImageNet [35, 103] that can cost up to hundreds of thousands of dollars to collect. But does visual learning really have to rely on such "bags of labeled snapshots"? The answer is clearly no, since children acquire much of their visual learning through constant observation and action in the world [17, 100]. This suggests that it is possible to build up representations of the visual world by exploiting long-term *video* observations, with no deliberate labels attached.

Prior work on learning image representations from video have focused on the

[1]The work in this chapter was supervised by Prof. Kristen Grauman and originally published in: "Slow and steady feature analysis: higher order temporal coherence in video". Dinesh Jayaraman and Kristen Grauman. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, June 2016.

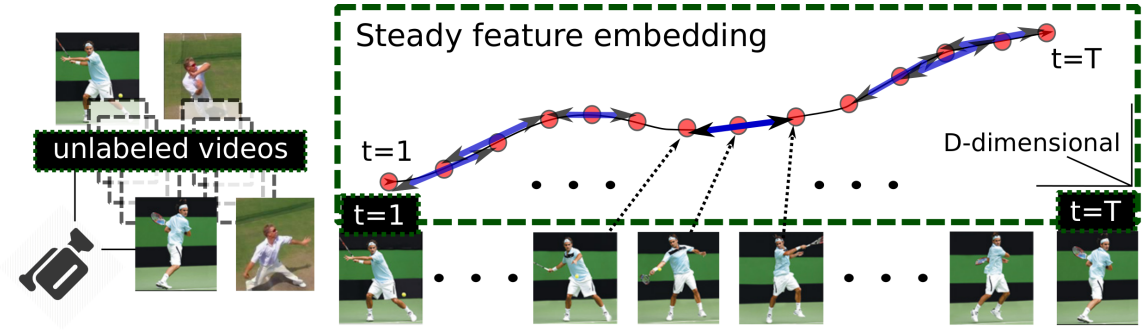


Figure 3.1: From unlabeled videos, we learn "steady features" that exhibit consistent feature transitions among sequential frames.

principle of "slow feature analysis"—good image representations should not vary much between consecutive video frames, since high level concepts associated with those frames typically change only slowly over time. While this idea captures that high-level visual signals change *slowly* over time, this is a very impoverished view of visual dynamics in the world, since it captures only invariances to small changes. For instance, slow feature analysis does not model that an object sliding on a smooth surface maintains constant speed over time, or that the poses of a walking person over time follow each other in a specific sequence.

In the work I present in this chapter, we will show that it is possible to more efficiently exploit video to learn image representations, mining richer signals than just invariances. To take the example of the walking person above, her poses over time will follow a smooth, predictable trajectory in the feature space that we will learn. Figure 3.1 shows a high-level illustration of this idea of "steady feature analysis".

In the rest of this chapter, I will first describe our approach for learning these steady representations from unlabeled video in Section 3.1, before presenting our

experiments and results in Sec 3.2.

3.1 Approach

Given auxiliary raw unlabeled video, we wish to learn an embedding amenable to a supervised classification task. We pose this as a feature learning problem in a convolutional neural network, where the hidden layers of the network are tuned not only with the backpropagation gradients from a classification loss, but also with gradients computed from the unlabeled video that exploit its temporal steadiness.

3.1.1 Problem setup

A *supervised training dataset* $\mathcal{S} = \{(\boldsymbol{x}_i, c_i)\}$ provides target class labels $c_i \in \mathcal{C} = [1, 2, .., C]$ for images $\boldsymbol{x}_i \in \mathcal{X}$ (represented in pixel space). The *unsupervised training dataset* $\mathcal{U} = \{\boldsymbol{x}_t\}$ consists of ordered video frames, where $\boldsymbol{x}_t$ is the video frame at time instant t.[2]

Importantly, we do *not* assume that the video $\mathcal{U}$ necessarily stems from the same categories or even precisely the same domain as images in $\mathcal{S}$. For example, in results we will demonstrate cases where $\mathcal{S}$ and $\mathcal{U}$ consist of natural scene images and autonomous vehicle video, respectively; or Web photos of human actions and YouTube video spanning dozens of distinct activities. The idea is that training with diverse unlabeled video should allow the learner to recover fundamental cues

[2]For notational simplicity, we will describe our method assuming that the unsupervised training data is drawn from a single continuous video, but it is seamless to train instead with a batch of unlabeled video clips.

about how objects move, how scenes evolve over time, how occlusions occur, how illumination varies, etc., independent of their specific semantic content.

The full image-pixels-to-class label classifier we learn will have the compositional form $\hat{c}_{\boldsymbol{\theta},W} = f_W \circ \mathbf{z}_{\boldsymbol{\theta}}(.)$, where $\mathbf{z}_{\boldsymbol{\theta}} : \mathcal{X} \rightarrow \mathcal{R}^D$ is a D-dimensional feature map operating on images in the pixel space, and $f_W : \mathcal{R}^D \rightarrow \mathcal{C}$ takes as input the feature map $\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x})$, and outputs the class estimate. We learn a linear classifier f_W represented by a $C \times D$ weight matrix W with rows $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_C$. At test time, a novel image is classified as $\hat{c}_{\boldsymbol{\theta},W} = \arg\max_i \boldsymbol{w}_i^T \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x})$.

To learn the classifier $\hat{c}_{\boldsymbol{\theta},W}$, we optimize an objective function of the form:

$$(\boldsymbol{\theta}^*, W^*) = \arg\min_{\boldsymbol{\theta},W} L_s(\boldsymbol{\theta}, W, \mathcal{S}) + \lambda L_u(\boldsymbol{\theta}, \mathcal{U}), \tag{3.1}$$

where $L_s(.)$ represents the supervised classification loss, $L_u(.)$ represents an unsupervised regularization loss term, and λ is the regularization hyperparameter. The parameter vector $\boldsymbol{\theta}$ is common to both losses because they are both computed on the learned feature space $\mathbf{z}_{\boldsymbol{\theta}}(.)$. The supervised loss L_s is a standard softmax classification loss over the supervised training dataset.

In the following, we first discuss how the unsupervised regularization loss $L_u(.)$ may be constructed to exploit temporal smoothness in video (Section 3.1.2). Then we generalize this to exploit higher order coherence, i.e., steadiness (Section 3.1.3). Section 3.1.4 then shows how a neural network corresponding to $\hat{c}_{\boldsymbol{\theta},W}$ may be trained to minimize Equation (3.1) above.

3.1.2 Background review: First-order temporal coherence

As discussed previously in Chapters 1 and 2, "slow feature analysis" (SFA) [174] seeks to learn image features that vary slowly over the frames of a video, with the aim of learning useful invariances. This idea of exploiting "slowness" or "temporal coherence" for feature learning has been explored in the context of neural networks [15, 62, 64, 118, 195]. We briefly review that underlying objective before introducing the proposed higher order generalization of temporal coherence.

A temporal neighbor pair dataset $\mathcal{U}_2$ is first constructed from the unlabeled video $\mathcal{U}$, as follows:

$$\mathcal{U}_2 = \{\langle (j,k), p_{jk} \rangle : \boldsymbol{x}_j, \boldsymbol{x}_k \in \mathcal{U} \text{ and } p_{jk} = \mathbb{1}(0 \leq j - k \leq T)\}, \tag{3.2}$$

where T is the temporal neighborhood size, and the subscript 2 signifies that the set consists of *pairs*. $\mathcal{U}_2$ indexes image pairs with neighbor-or-not binary annotations p_{jk}, automatically extracted from the video. We discuss the setting of T in results. In general, one wants the time window spanned by T to include motions that are small enough to be label-preserving, so that correct invariances are learned; in practice this is typically on the order of a second or less.

With this dataset, the SFA property translates as $\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_j) \approx \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_k)$, $\forall p_{jk} = 1$. A simple formulation of this as an unsupervised regularizing loss would be as follows:

$$R'_2(\boldsymbol{\theta}, \mathcal{U}) = \sum_{(j,k) \in \mathcal{N}} d(\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_j), \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_k)), \tag{3.3}$$

where $d(.,.)$ is a distance measure (e.g., ℓ_1 in [118] and ℓ_2 in [64]), and $\mathcal{N} \subset \mathcal{U}_2$ denotes the subset of "positive" neighboring frame pairs, i.e., those for which $p_{jk} = 1$. This

loss by itself admits problematic minimizers such as $\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0, \forall \boldsymbol{x} \in \mathcal{X}$, which corresponds to $R_2' = 0$. Such solutions may be avoided by a *contrastive* [64] version of the loss function that also exploits "negative" (non-neighbor) pairs:

$$\begin{aligned} R_2(\boldsymbol{\theta}, \mathcal{U}) &= \sum_{(j,k) \in \mathcal{U}_2} D_\delta(\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_j), \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_k), p_{jk}) \\ &= \sum_{(j,k) \in \mathcal{U}_2} p_{jk}\ d(\mathbf{z}_{\boldsymbol{\theta}j}, \mathbf{z}_{\boldsymbol{\theta}k}) + \overline{p_{jk}}\ \max(\delta - d(\mathbf{z}_{\boldsymbol{\theta}j}, \mathbf{z}_{\boldsymbol{\theta}k}), 0), \end{aligned} \tag{3.4}$$

where $\mathbf{z}_{\boldsymbol{\theta}i}$ denotes $\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$ and $\overline{p} = 1 - p$. As shown above, the contrastive loss $D_\delta(\boldsymbol{a}, \boldsymbol{b}, p)$ penalizes distance between $\boldsymbol{a}$ and $\boldsymbol{b}$ when the pair are neighbors ($p = 1$), and encourages distance between them when they are not ($p = 0$), up to a margin δ.

3.1.3 Key idea: higher-order temporal coherence

The slow feature formulation of Equation (3.4) encourages feature maps that produce small first-order temporal derivatives in the learned feature space: $d\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_t)/dt \approx 0$. This first-order temporal coherence is restricted to learning to ignore small jitters in the visual signal.

Our idea is to model higher order temporal coherence in the unlabeled video, so that the features can further capture rich structure in *how* the visual content changes over time. In the general case, this means we want a regularizer that encourages higher order derivatives to be small: $d^n\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_t)/dt^n \approx 0, \forall n = 1, 2, ..N$. Accordingly, we need to generalize from pairs of temporally close frames to tuples of frames.

In this work, we focus specifically on learning *steady* features—the second-order case, which can be encoded with triplets of frames, as we will see next. In

a nutshell, whereas slow learning insists that the features not change too quickly, steady learning insists that feature *changes* in the immediate future remain similar to those in the recent past.

First, we create a triplet dataset $\mathcal{U}_3$ from the unlabeled video $\mathcal{U}$ as:

$$\mathcal{U}_3 = \{\langle (l, m, n), p_{lmn} \rangle : \boldsymbol{x}_l, \boldsymbol{x}_m, \boldsymbol{x}_n \in \mathcal{U} \text{ and } p_{lmn} = \mathbb{1}(0 \leq m - l = n - m \leq T)\}. \tag{3.5}$$

$\mathcal{U}_3$ indexes image triplets with binary annotations indicating whether they are in-sequence, evenly spaced frames in the video, within a temporal neighborhood T. In practice, we select "negatives" ($p_{lmn} = 0$) from triplets where $m - l \leq T$ but $n - m \geq 2T$ to provide a buffer and avoid noisy negatives.

We construct our steady feature analysis regularizer using these triplets, as follows:

$$R_3(\boldsymbol{\theta}, \mathcal{U}) = \sum_{(l,m,n) \in \mathcal{U}_3} D_\delta(\mathbf{z}_{\boldsymbol{\theta}l} - \mathbf{z}_{\boldsymbol{\theta}m},\ \mathbf{z}_{\boldsymbol{\theta}m} - \mathbf{z}_{\boldsymbol{\theta}n},\ p_{lmn}), \tag{3.6}$$

where $\mathbf{z}_{\boldsymbol{\theta}l}$ is again shorthand for $\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_l)$ and D_δ refers to the contrastive loss defined above. For positive triplets—meaning those occurring in sequence and within a temporal neighborhood—the above loss penalizes distance between the adjacent pairwise feature *difference* vectors. For negative triplets, it *encourages* this distance, up to a maximum margin distance δ. Effectively, R_3 encourages the feature representations of positive triplets to be collinear, i.e., $\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_l) - \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_m) \approx \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_m) - \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_n)$. See Figure 3.1.

Our final optimization objective combines the first and second order losses

(Equation (3.4) and (3.6)) into the unsupervised regularization term:

$$L_u(\boldsymbol{\theta},\mathcal{U}) = R_2(\boldsymbol{\theta},\mathcal{U}) + \lambda' R_3(\boldsymbol{\theta},\mathcal{U}), \tag{3.7}$$

where λ' controls the relative impact of the two terms. Recall this regularizer accompanies the classification loss in the main objective of Equation (3.1).

Equivariance-inducing property of $R_3(\boldsymbol{\theta},\mathcal{U})$: While first-order coherence encourages invariance, the proposed second-order coherence may be seen as encouraging the more general property of *equivariance*.

The mapping $\mathbf{z}(.)$ is equivariant to an image transformation g if there exists some "simple" function $\mathbf{f}_g : \mathcal{R}^D \rightarrow \mathcal{R}^D$ such that $\mathbf{z}(g\boldsymbol{x}) \approx \mathbf{f}_g(\mathbf{z}(\boldsymbol{x}))$. As we demonstrate later in Chapter 4, equivariance has been found to be useful for visual representations (cf. [71, 75, 78, 98, 141]). To see how feature steadiness is related to equivariance, consider a video with frames $\boldsymbol{x}_t, 1 \leq t \leq T$. Given a small temporal neighborhood Δt, frames $\boldsymbol{x}_{t+\Delta t}$ and $\boldsymbol{x}_t$ must be related by a *small* transformation g (small because of *first* order temporal coherence assumption), i.e., $\boldsymbol{x}_{t+\Delta t} = g\boldsymbol{x}_t$. Assuming *second* order coherence of video, this transformation g itself remains approximately constant in a small temporal neighborhood, so that, in particular, $\boldsymbol{x}_{t+2\Delta t} \approx g\boldsymbol{x}_{t+\Delta t}$.

Now, for equivariant features $\mathbf{z}(.)$, by the definition of equivariance and the observations above, $\mathbf{z}(\boldsymbol{x}_{t+2\Delta t}) \approx \mathbf{f}_g(\mathbf{z}(\boldsymbol{x}_{t+\Delta t})) \approx \mathbf{f}_g \circ \mathbf{f}_g(\mathbf{z}(\boldsymbol{x}_t))$. Further, given that g is a small transformation, $\mathbf{f}_g$ is well-approximated in a small neighborhood by its first order Taylor approximation, so that: (1) $\mathbf{z}(\boldsymbol{x}_{t+\Delta t}) \approx \mathbf{z}(\boldsymbol{x}_t) + \mathbf{c}(t)$, and (2) $\mathbf{z}(\boldsymbol{x}_{t+2\Delta t}) \approx \mathbf{z}(\boldsymbol{x}_t) + 2\mathbf{c}(t)$. In other words, under the realistic assumption that natural

videos evolve smoothly, within small temporal neighborhoods, feature equivariance is equivalent to the second order temporal coherence formulated in Eq (3.6), with l, m, n set to $t, t + \Delta t, t + 2\Delta t$ respectively. This connection between equivariance and the second order temporal coherence induced by R_3 helps motivate why we can expect our feature learning scheme to benefit recognition.

3.1.4 Form of the feature mapping function

We use a convolutional neural network (CNN) architecture to represent the feature mapping function $\mathbf{z}_{\boldsymbol{\theta}}(.)$. The parameter vector $\boldsymbol{\theta}$ represents the CNN's learned layer weight matrices. See Section 3.2.1 for architecture choices.

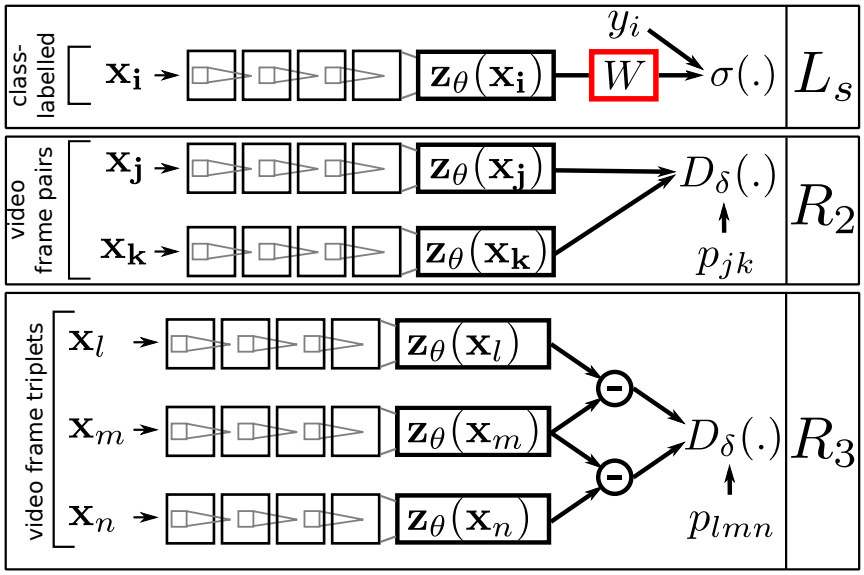


Figure 3.2: "Siamese" network configuration (shared weights for the $\mathbf{z}_{\boldsymbol{\theta}}$ layer stacks) with portions corresponding to the 3 terms L_s, R_2 and R_3 in our objective. R_2 and R_3 compose the unsupervised loss L_u in Equation (3.1). L_s is the supervised loss for recognition in static images.

To optimize Equation (3.1) with the regularizer in Equation (3.7), we employ standard mini-batch stochastic gradient descent (as implemented in [82]) in a "Siamese" setup, with 6 replicas of the stack $\mathbf{z}_{\boldsymbol{\theta}}(.)$, as shown in Figure 3.2, 1 stack for L_s (input: supervised training samples $\boldsymbol{x}_i$), 2 for R_2 (input: temporal neighbor pairs $(\boldsymbol{x}_j, \boldsymbol{x}_k)$) and 3 for R_3 (input: triplets $(\boldsymbol{x}_l, \boldsymbol{x}_m, \boldsymbol{x}_n)$). The shared layers are initialized to the same random values and modified by the same gradients (sum of the gradients of the 3 terms) in each training iteration, so they remain identical throughout. More details are in Section 3.2.1.

3.2 Experiments

We test our approach using five challenging public datasets for three tasks—object, scene, and action recognition—spanning 432 categories. We also analyze its ability to learn higher order temporal coherence with a sequence completion task.

3.2.1 Experimental setup

Our three recognition tasks (specified by the names of the unsupervised and supervised datasets as $\mathcal{U} \rightarrow \mathcal{S}$) are NORB→NORB object recognition, KITTI→SUN scene recognition and HMDB→PASCAL-10 single-image action recognition. Table 3.1 (left) summarizes key dataset statistics. Figure 3.3 shows depicts the unsupervised-to-supervised transfer tasks together with examples from each dataset.

Supervised datasets $\mathcal{S}$ (1) **NORB** [96] has 972 images each of 25 toys against clean backgrounds captured over a grid of camera elevations and azimuths. (2)

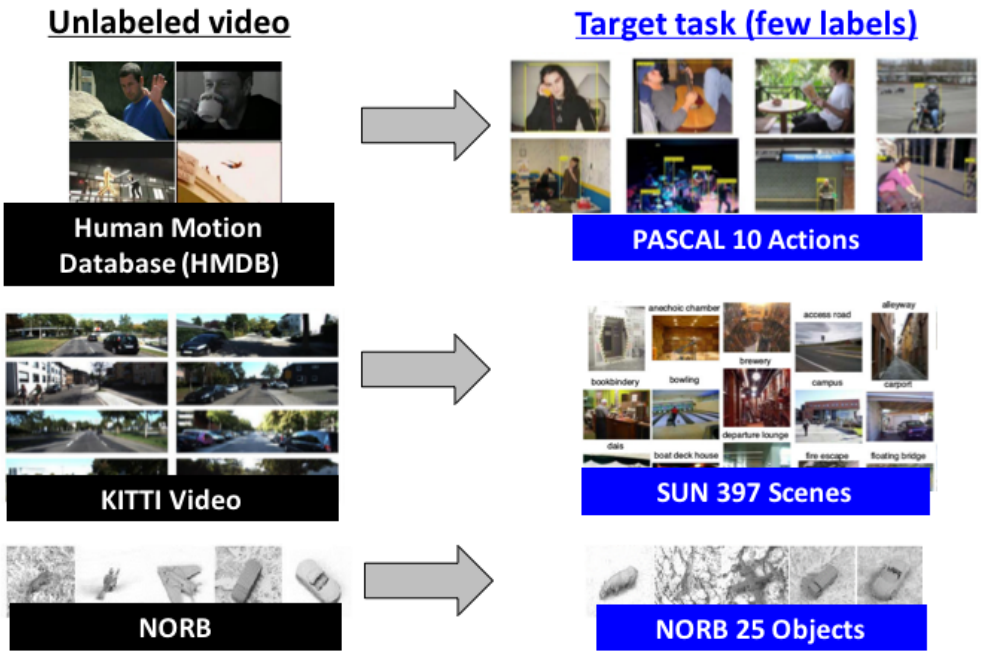


Figure 3.3: Examples from unsupervised video datasets (left) and the supervised image datasets (right). Representations trained on unlabeled videos are evaluated on supervised tasks using the datasets on the right. See Section 3.2.1.

SUN [177] contains Web images of 397 scene categories. (3) **PASCAL-10** [45] is a still-image human action recognition dataset with 10 categories. For all three datasets, we use few labeled training images (see Table 3.1), since unsupervised regularization schemes should have most impact when labeled data is scarce [118]. This is an important scenario, given the "long tail" of categories lacking ample labeled exemplars.

Unsupervised datasets $\mathcal{U}$ (1) **NORB** consists of pose-registered turntable images (not video), but it is straightforward to generate the pairs and triplets for $\mathcal{U}_2$

Task	Img/frame dims	#Classes	Recog. Task	#Train	#Test	Unsup. Input Type	#Pairs (1:3)	#Triplets (1:1)
NORB→NORB	96×96×1	25	object	150	8100	pose-reg. images	50,000	75,000
KITTI→SUN	32×32×1	397	scene	2382	7940	car-mounted video	100,000	100,000
HMDB→PASCAL-10	32×32×3	10	action	50	2000	Web video	100,000	100,000

Table 3.1: Statistics for the unsupervised and supervised datasets ($\mathcal{U} \rightarrow \mathcal{S}$) used in the recognition tasks (positive to negative ratios for pairs and triplets indicated in headers).

and $\mathcal{U}_3$ assuming smooth motions in the annotated pose space. We mine these pairs and triplets from among the 648 images per class that are not used for testing. (2) **KITTI** [52] has videos captured from a car-mounted camera in a variety of locations around the city of Karlsruhe. Scenes are largely static except for traffic, but there is large and systematic camera motion. (3) **HMDB** [93] contains 6849 short Web and movie video clips containing 51 diverse actions. We select 1000 clips at random. While some videos include camera motion (e.g., to follow an athlete running), most have stationary cameras and small human pose-change motions. The time window T is a hyperparameter of both our method as well as existing SFA methods. We fix $T = 2$ and $T = 0.5$ seconds for KITTI and HMDB, respectively, based on cross-validation for best performance by the SFA baselines.

Baselines We compare our slow-and-steady feature analysis approach (SSFA) to four methods, including two key existing methods for learning from unlabeled video. The three unsupervised baselines are: (1) UNREG: An unregularized network trained only on the supervised training samples $\mathcal{S}$. (2) SFA-1: An SFA approach proposed in [118] that uses ℓ_1 for $d(.)$ in Equation 3.4. (3) SFA-2: Another SFA variant [64] that sets the distance function $d(.)$ to the ℓ_2 distance in Eq 3.4. The SFA methods

train with the unlabeled pairs, while SSFA trains with both the pairs and triplets

These comparisons are most crucial to gauge the impact of the proposed approach versus the state of the art for feature learning with unlabeled video. Also note that the SFA baselines are known (see e.g., [3]) to be superior to alternative paradigms for unsupervised learning such as autoencoders [72]. However, we are also interested in knowing to what extent learning from unlabeled video can even start to compete with methods learned from heavily labeled data (which costs substantial human effort). Thus, we also compare against a *supervised* pretraining and finetuning approach denoted SUP-FT (details in Section 3.2.3).

Network architectures For the NORB→NORB task, we use a fully connected network architecture: input → 25 hidden units → ReLU nonlinearity → D=25 features. For the other two tasks, we resize images to 32×32 to allow fast and thorough experimentation with standard CNN architectures known to work well with tiny images [1], producing D=64-dimensional features. The 32×32 CNN architecture [1] representing $\mathbf{z}_{\boldsymbol{\theta}}$, used for the KITTI→SUN and HMDB→PASCAL-10 tasks is shown in Figure 3.4. Recognition tasks on 32×32 images are much harder than with full-sized images, so these are highly challenging tasks.

Optimization details We initialize neural networks according to the scheme proposed in [59], and run Nesterov accelerated stochastic gradient descent using the open source Caffe [82] package, until validation classification loss converges or begins to rise. Optimization hyperparameters are selected greedily through cross-validation

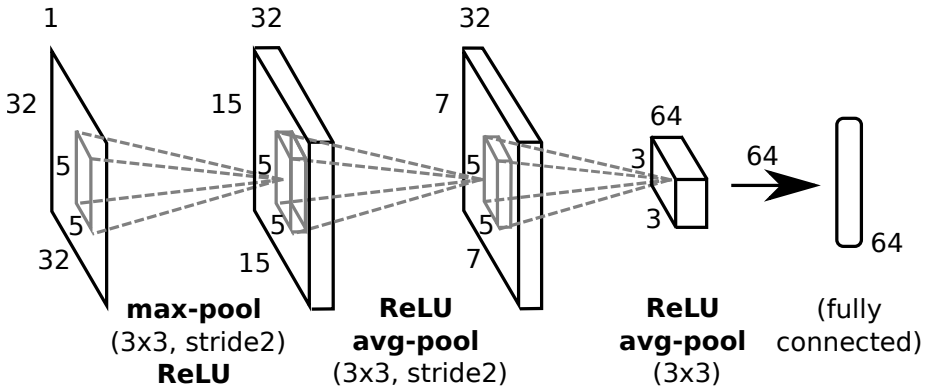


Figure 3.4: 32×32 CNN architecture used for the KITTI→SUN and HMDB→PASCAL-10 tasks

in the following order: base learning rate, λ and λ' (starting from λ=λ'=0). Our validated (λ,λ') values for NORB→NORB, KITTI→SUN, and HMDB→PASCAL respectively are (0.1,0.3),(3,0.1), and (0.3,1).

Specifically, for each task, the optimal base learning rate (from 0.1, 0.01, 0.001, 0.0001) was first identified for UNREG. Next λ was set through a logarithmic grid search (steps of $10^{0.5}$), with λ' set to 0, i.e., this parameter was optimized for SFA-2. The margin parameter δ of the contrastive loss in $R_2(.)$ was set to 1.0 for all methods – this affects the objective function only up to a feature scaling operation, and so may be set to any positive value. For SSFA, a similar search was then performed over λ' (logarithmic grid search with steps of $10^{0.5}$), and then a small search for the contrastive loss margin δ in $R_3(.)$ (over 0, 0.1 and 1). Setting the margin to $\delta = 0$ in a contrastive loss reduces it to the simple distance loss over positive samples.

3.2.2 Quantifying steadiness

First we use a sequence completion task to analyze how well the desired steadiness property is induced in the learned features. We compose a set of sequential triplets from the pool of test images, formed similarly to the positives in Equation (3.5). At test time, given the first two images of each triplet, the task is to predict what the third looks like.

We apply our SSFA to infer the missing triplet item as follows. Recall that our formulation encourages sequential triplets to be collinear in the feature space. As a result, given $\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_1)$ and $\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_2)$, we can extrapolate $\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_3)$ as $\tilde{\mathbf{z}}_{\boldsymbol{\theta}}(\boldsymbol{x}_3) = 2\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_2) - \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_1)$. To backproject to the image space, we identify an image closest to $\tilde{\mathbf{z}}_{\boldsymbol{\theta}}(\boldsymbol{x}_3)$ in feature space. Specifically, we take a large pool $\mathcal{C}$ of candidate images, map them all to their features via $\mathbf{z}_{\boldsymbol{\theta}}$, and rank them in increasing order of distance from $\tilde{\mathbf{z}}_{\boldsymbol{\theta}}(\boldsymbol{x}_3)$. The rank r of the correct candidate $\boldsymbol{x}_3$ is now a measure of sequence completion performance.

For the candidate set $\mathcal{C}$ for NORB, the entire NORB test image set was used. For the video datasets KITTI and HMDB though, it is practically difficult to include all image frames in the candidate set $\mathcal{C}$. To avoid having to compute features and perform very expensive nearest neighbor search over a very large number of frames, we form a randomly sub-sampled $\mathcal{C}$ instead, as follows. Starting from empty $\mathcal{C}$, we added (1) all the unique images among the query pairs (2) their corresponding ground truth completion images and (3) a minimum number N of randomly chosen frames from each video represented within $\mathcal{C}$ until this point. This ensures that the task is non-trivial by adding distractors from the same video as the ground truth

Datasets→	NORB	KITTI	HMDB
SFA-1 [118]	0.95	31.04	2.70
SFA-2 [64]	0.91	8.39	2.27
SSFA (ours)	**0.53**	**7.79**	**1.78**

Table 3.2: Sequence completion normalized correct candidate rank η. Lower is better. (See Section 3.2.2.)

candidate image, which are likely to have similar appearance. We used N=10 for KITTI and N=5 for HMDB to keep the total numbers of images manageable. Finally, we select from $|\mathcal{C}|$ =8100, 5000 and 5000 candidates respectively for NORB, KITTI and HMDB, for each of N =20,000, 1000 and 1,000 query pairs respectively for the three datasets.

Table 3.2 reports the mean percentile rank $\eta = \mathbb{E}[r/|\mathcal{C}|] \times 100$ over all query pairs. Lower η is better. Clearly, our SSFA regularization induces steadiness in the feature space, reducing η nearly by half compared to baseline regularizers on NORB and by large margins on HMDB too. Our regularizer R_3 is closely matched to this task, so these gains are expected. Note however that these gains are reported after training to minimize the *joint* objective, which includes L_s and R_2, apart from R_3, and with regularization weights tuned for *recognition* tasks.

Figure 3.5 shows sequence completion examples from all three video datasets. Particularly impressive results are the third NORB example (where despite a difficult viewpoint, the sequence is completed correctly by the top-ranked candidate), and the third HMDB example, where a highly dynamic baseball pitch sequence is correctly completed by the third ranked image. The top-ranked candidate for this example

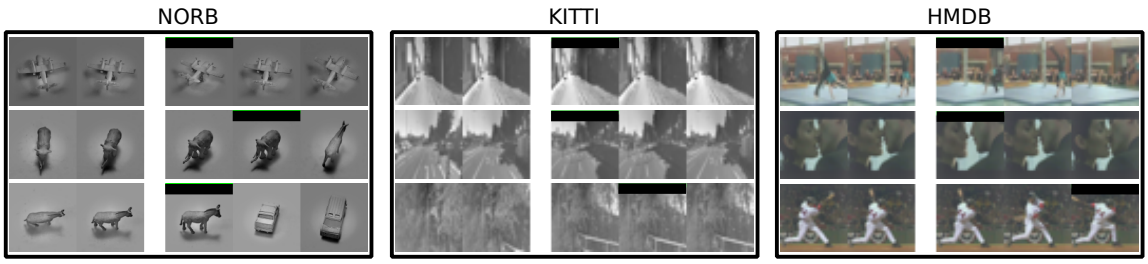


Figure 3.5: Sequence completion examples from all three video datasets. In each instance, a query pair is presented on the left, and the top three completion candidates as ranked by our method are presented on the right. Ground truth frames are marked with black highlights.

illustrates a common failure mode—the second image of the query pair is itself picked to complete the sequence. This may reflect the fact that HMDB sequences in particular exhibit very little motion (camera motions rare, mostly small object motions). Usually, as in the third KITTI example, even the top-ranked candidates other than the ground truth frame are highly plausible completions.

3.2.3 Recognition results

Now we report results on the three unsupervised-to-supervised recognition tasks. Table 3.3 shows the results. Our SSFA method comprehensively outperforms not only the purely supervised UNREG baseline, but also the popular SFA-1 and SFA-2 slow feature learning approaches, beating the best baseline for each task by 9%, 36% and 9% respectively. The results on KITTI→SUN and HMDB→PASCAL-10 are particularly impressive because the unsupervised and supervised dataset domains are mismatched. All KITTI data comes from a single car-mounted road-facing camera driving through the streets of one city, whereas SUN images are downloaded from the Web, captured by different cameras from diverse viewpoints, and cover 397 scene

Task type→	Objects	Scenes		Actions
Datasets→	NORB→NORB	KITTI→SUN		HMDB→PASCAL-10
Methods↓	[25 cls]	[397 cls]	[397 cls, top-10]	[10 cls]
random	4.00	0.25	2.52	10.00
UNREG	24.64±0.85	0.70±0.12	6.10±0.67	15.34±0.28
SFA-1 [118]	37.57±0.85	1.21±0.14	8.24±0.25	19.26±0.45
SFA-2 [64]	39.23±0.94	1.02±0.12	6.78±0.32	19.04±0.24
SSFA (ours)	**42.83±0.33**	**1.65±0.04**	**9.19±0.10**	**20.95±0.13**

Table 3.3: Recognition results (mean ± standard error of accuracy % over 5 repetitions) (Section 3.2.3). Our method outperforms both existing slow feature/temporal coherence methods and the unregularized baseline substantially, across three distinct recognition tasks.

categories mostly unrelated to roads. PASCAL-10 images are bounding-box-cropped and therefore centered on single persons, while HMDB videos, which are mainly clips from movies and Web videos, often feature multiple people, are not as tightly focused on the person performing the action, and are of low quality, sometimes with overlaid text etc.

Aside from the diversity of tasks (object, scene, and action recognition), our unsupervised datasets also exhibit diverse types of motion. NORB is generated from planned, discrete camera manipulations around a central object of interest. The KITTI camera moves through a real largely static landscape in smooth motions on roads at varying speeds. HMDB videos on the other hand are usually captured from stationary cameras with a mix of large and small foreground and background object motions. Even the dynamic camera videos in HMDB are sometimes captured from hand-held devices leading to jerky motions, where our temporal steadiness assumptions might be stressed.

Pairing unsupervised and supervised datasets: Thus far, our pairings of unsupervised and supervised datasets reflect our attempt to learn from video that *a priori* seems related to the ultimate recognition task, e.g., HMDB human action videos are paired with PASCAL-10 Action still images. However, as discussed above, the domains are only roughly aligned. Curious about the impact of the choice of unlabeled video data, we next try swapping out HMDB for KITTI in the PASCAL action recognition task. On this new KITTI→PASCAL task, we still easily outperform our nearest baseline, although our gain drops by $\approx 0.9\%$ (SFA-2:19.06% vs. our SSFA:20.01%). Despite the fact that the human motion dynamics of HMDB ostensibly match the action recognition task better than the egomotion dynamics of KITTI (where barely any people are visible), we maintain our advantage over the purely slow methods. This indicates that there is reasonable flexibility in the choice of unlabeled videos fed to SSFA.

Increasing supervised training sets: Thus far, we have kept labeled sets small to simulate the "long tail" of categories with scarce training samples where priors like ours and the baselines' have most impact. In a preliminary study for larger training pools, we now increase SUN training set sizes from 6 to 20 samples per class for KITTI→SUN, for a total of 7,940 labeled training images. Our method retains a 20% gain over existing slow methods (SSFA: 3.24% vs SFA-2: 2.65%). This suggests our approach is valuable even with larger supervised training sets.

Varying unsupervised training set size: To observe the effect of unsupervised training set size, we now restrict SSFA to use varying-sized subsets of unlabeled video

on the HMDB→PASCAL-10 task. Performance scales roughly log-linearly with the duration of video observed, as shown in Figure 3.6. This suggests that even larger gains may be achieved simply by training SSFA with more freely available unlabeled video.

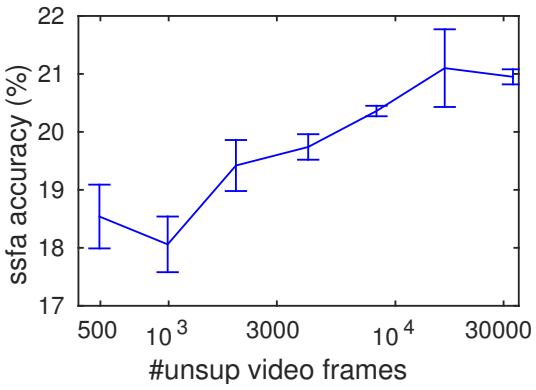


Figure 3.6: SSFA classification accuracy vs. duration of unsupervised video (mean, standard error over 5 runs). Performance scales log-linearly with the duration of observed video, suggesting that even larger gains may be possible by simply exploiting more freely available video.

Purely unsupervised feature learning: We now evaluate the usefulness of features trained to optimize the unsupervised SSFA loss L_u (Equation (3.7)) alone (rather than the full objective of Equation (3.1), which includes a supervised classification loss L_s). Features trained on HMDB are evaluated at various stages of training, on the task of k-nearest neighbor classification on PASCAL-10 (k =5, and 100 training images per action). Figure 3.7 shows the results. Starting at $\approx$ 17.8% classification accuracy for randomly initialized networks, unsupervised SSFA training steadily improves the discriminative ability of features. This shows that SSFA can train useful

image representations even without jointly optimizing a supervised objective.

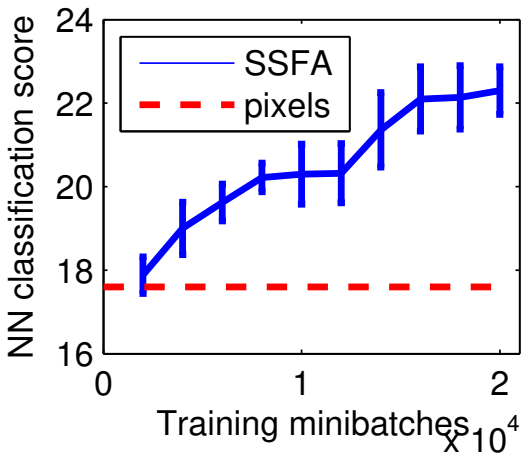


Figure 3.7: SSFA k-NN accuracy improvement with SSFA training (mean, standard error over 5 runs).

Comparison to supervised pretraining and finetuning: Recently, a two-stage supervised pretraining and finetuning strategy (SUP-FT) has emerged as the leading approach to solve visual recognition problems with limited training data where high-capacity models like deep neural networks may not be directly learned [41, 57, 85, 124]. In the first stage ("supervised pretraining"), a neural network "NET1" is first trained on a related problem for which large training datasets *are* available. In a second stage ("finetuning"), the weights from NET1 are used to initialize a second network ("NET2") with similar architecture. NET2 is then trained on the target task, using reduced learning rates to minimally modify the features learned in NET1.

In principle, completely unsupervised feature learning approaches like ours have important advantages over the SUP-FT paradigm. In particular, (1) they can

leverage essentially infinite unlabeled data without requiring expensive human labeling effort thus potentially allowing the learning of higher capacity models and (2) they do not require the existence of large *"related"* supervised datasets from which features may be meaningfully transferred to the target task. While the pursuit of these advantages continues to drive vigorous research, unsupervised feature learning methods still underperform supervised pretraining for image classification tasks, where great effort has gone into curating large labeled databases, e.g., ImageNet [35], CIFAR [91].

As a final experiment, we examine how the proposed unsupervised feature learning idea competes with the popular supervised pretraining model. To this end, we adopt the CIFAR-100 dataset consisting of 100 diverse object categories as a basis for supervised pretraining.[3] The new baseline SUP-FT trains NET1 on CIFAR, then finetunes NET2 for either PASCAL-10 action or SUN scene recognition tasks using the exact same (few) labeled instances given to our method. In parallel, our method "pretrains" only via the SSFA regularizer learned with unlabeled HMDB / KITTI video respectively for the two tasks. Our method uses *zero* labeled CIFAR data.

Fig 3.8 shows the results. On PASCAL-10 action recognition (left), our method significantly outperforms SUP-FT pretrained with all 50,000 images of CIFAR-100! Gathering image labels from the crowd for large multi-way problems can take on average 1 minute per image [137], meaning we are getting better results while also

[3]We choose CIFAR-100 for its compatibility with the 32×32 images used throughout our results in this chapter, which let us leverage standard CNN architectures known to work well with tiny images [1]. CIFAR-100 classification is the most widely benchmarked tiny image classification task, much as ImageNet ILSVRC is most widely benchmarked for larger images.

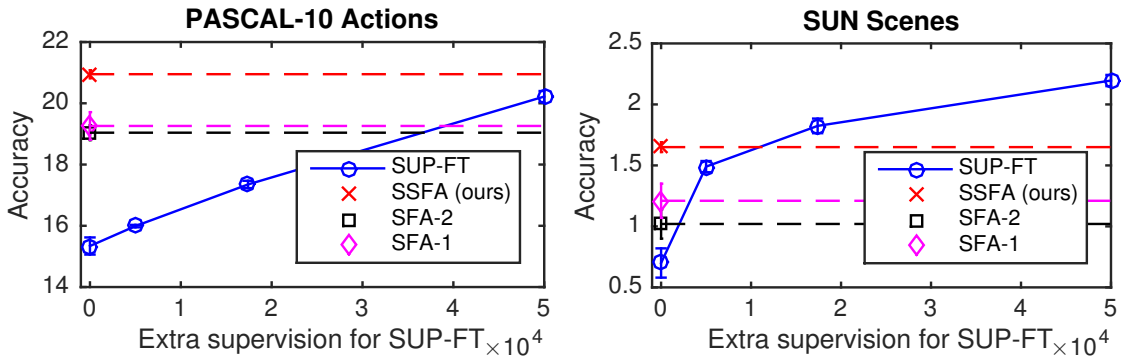


Figure 3.8: Comparison to CIFAR-100 supervised pretraining SUP-FT, at various supervised training set sizes. Flat dashed lines reflect that our method (and SFA) always use zero additional labels.

saving $\sim$ 830 hours of human effort. On SUN scene recognition (right), SSFA outperforms SUP-FT with 5K labels and remains competitive even when the supervised method has a 17,500 label advantage. However, SUP-FT-50K's advantage on the SUN task is more noticeable; its gain is similar to our gain over the best slow-feature method.

The upward trend in accuracy for SUP-FT with more CIFAR-100 labeled data indicates that it successfully transfers generic recognition cues to the new tasks. On the other hand, the fact that it fares worse on PASCAL actions than SUN scenes reinforces that *supervised* transfer depends on having large curated datasets in a *strongly related* domain. In contrast, our approach successfully "transfers" what it learns from purely unlabeled video. In short, our method can achieve better results with substantially less supervision. More generally, we view it as an exciting step towards unlabeled video bridging the gap between unsupervised and supervised pretraining for visual recognition.

3.3 Conclusion

In this chapter, I formulated an unsupervised feature learning approach that exploits higher order temporal coherence in unlabeled video, and demonstrated its powerful impact for several recognition tasks. Despite over 15 years of research surrounding slow feature analysis (SFA), its variants and applications, we are the first to identify that SFA is only the first order approximation of a more general temporal coherence idea. This basic observation leads to our intuitive approach that can be easily plugged into applications where first order temporal coherence has already been found useful [15, 62, 64, 105, 118, 121, 168, 186, 190, 195]. To our knowledge, ours are the first results where unsupervised learning from video actually surpasses the accuracy of today's favored approach, heavily supervised pretraining, on generic recognition tasks.

Some questions surrounding the work in this chapter remain that could yield fruitful directions for follow-up work. Firstly, while we have laid out the general principle of higher-order temporal coherence, we have only empirically evaluated *second*-order coherence. It may be expected that higher orders yield diminishing returns, but this remains to be confirmed in practice. Secondly, standard temporal coherence methods have been strengthened through ranking-based losses recently [142, 168], and it would be of value to investigate how to formulate higher-order coherence losses within these frameworks. Thirdly, due to implementation issues, we have confined our experiments to small images. More efficient implementations are possible, which would allow evaluation on datasets of larger images. Finally, a thorough and systematic study of the effect of temporal neighborhood sizes on temporal coherence-

based learning would be of empirical value to many temporal coherence approaches in general, including ours.

Learning through constantly monitoring a visual stream, as described in this chapter, is only the first step towards a solution for embodied visual learning. In the next chapter, we take one further step, modeling a setting where the embodied agent at training time has additional access to proprioceptive information about its own motions.

Chapter 4

Learning image representations tied to observer motion

[1]In the last chapter, we exploited the temporal sequentiality of frames in video to mine signals for unsupervised representation learning. This may be thought of as learning by continuous passive observation of the visual world. While we have begun to move away from the "bag of labeled snapshots" manually supervised paradigm and towards embodied visual learning, learning from video as in Chapter 3 is only the first step.

In embodied agents, visual observations are inextricably tied to the motor activity behind them. A cat moving through its environment knows *how* it is moving to cause the observed changes to its visual sensory input over time, and as we saw in Held and Hein's kitten carousel study [69] (described before in Chapter 1), this knowledge is critical to its visual development. In other situations, the proprioceptive "motor signal" knowledge most relevant to the visual observation may not be the egomotion of the observer moving in the scene, but the second-hand motion of an

[1]The work in this chapter was supervised by Prof. Kristen Grauman and originally published in: "Learning image representations tied to egomotion". Dinesh Jayaraman and Kristen Grauman. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, December 2015. An expanded article appeared in "Learning egomotion-tied image representations from unlabeled video". Dinesh Jayaraman and Kristen Grauman. In Internation Journal of Computer Vision Special Issue of Best Papers from ICCV 2015, March 2017.

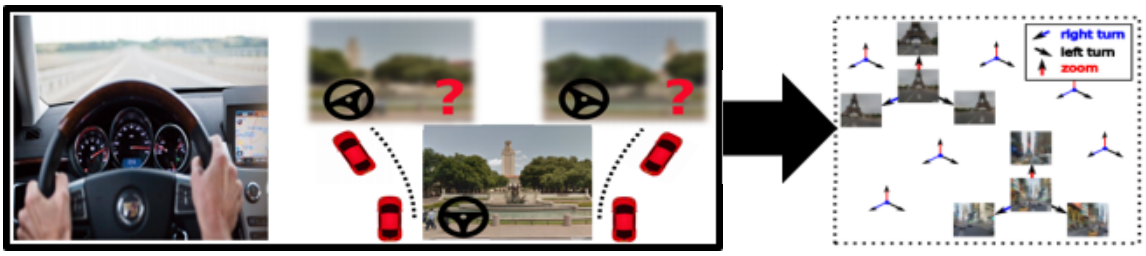


Figure 4.1: Our approach learns an image embedding from unlabeled video. Starting from egocentric video together with observer egomotion signals, we train a system on a "view prediction" task (left), to learn *equivariant* visual features that respond predictably to observer egomotion (right). In this target equivariant feature space, pairs of images related by the same egomotion are related by the same *feature transformation* too.

object being actively manipulated, e.g., by a person or robot's end effectors. In either case, vision must develop in the context of acting and moving in the world. As the celebrated psychologist J.J. Gibson wrote, "We perceive in order to move, but we must also move in order to perceive." [55].

In this chapter, I attempt to exploit the structure in this joint space of visual observations and agent actions or motions towards visual learning, as I introduced earlier in Section 1.2. We do this by exploiting motor signals accompanying unlabeled egocentric video, of the sort that one could acquire through a wearable platform like Google Glass, or a self-driving car. Figure 4.1 shows a schematic illustration of our idea. Section 4.1 describes our technical approach, and Section 4.2 later presents our key empirical findings.

4.1 Approach

Our goal is to learn an image representation that is equivariant with respect to egomotion transformations, given video together with registered camera egomotion

information from the time of capture.

4.1.1 Problem setup

Let $\boldsymbol{x}_i \in \mathcal{X}$ be an image in the original pixel space, and let $\boldsymbol{y}_i \in \mathcal{Y}$ be its associated ego-pose representation. The ego-pose captures the available motor signals, and could take a variety of forms. For example, $\mathcal{Y}$ may encode the complete observer camera pose (its position in 3D space, pitch, yaw, roll), some subset of those parameters, or any reading from a motor sensor paired with the camera.

As input to our learning algorithm, we have a training set $\mathcal{U}$ of N_u unlabeled image pairs and their associated ego-poses, $\mathcal{U} = \{\langle(\boldsymbol{x}_i, \boldsymbol{x}_j), (\boldsymbol{y}_i, \boldsymbol{y}_j)\rangle\}_{(i,j)=1}^{N_u}\}$. The image pairs originate from video sequences, though they need not be adjacent frames in time. The set may contain pairs from multiple videos and cameras. Note that this training data does *not* have any semantic labels (e.g., object categories); they are "labeled" only in terms of the ego-motor sensor readings. Since our method relies on freely available motion sensor readings associated with video streams (e.g., from Google glass, self-driving cars, or even hand-held mobile devices), rather than on expensive manually supplied labels, it is effectively unsupervised.[2]

In the following, we first explain how to translate ego-pose information into

[2]One could attempt to apply our idea using camera poses inferred from the video itself (i.e., with structure from motion). However, there are conceptual and practical advantages to relying instead on external sensor data capturing egomotion. First, the sensor data, when available, is much more efficient to obtain and can be more reliable. Second, the use of an external sensor parallels the desired effect of the agent learning from its proprioception motor signals, as opposed to bootstrapping the visual learning process from a previously defined visual odometry module based on the same visual input stream.

pairwise "motion pattern" annotations (Section 4.1.2). Then, Section 4.1.3 defines the precise nature of the equivariance we seek, and Section 4.1.4 defines our learning objective. We define a variant of our approach using non-discrete egomotion patterns and non-linear equivariance maps in Section 4.1.5. Then, in Section 4.1.6, we show how a feedforward neural network architecture may be trained to produce the desired equivariant feature space. Finally, Section 4.1.7 shows how our equivariant feature learning scheme may be used to enhance recognition with limited training data.

4.1.2 Mining discrete egomotion patterns

First we want to organize training sample pairs into a discrete set of egomotion patterns $\mathcal{G}$. For instance, one egomotion pattern might correspond to "tilt downwards by approximately 20°". As we will see in Section 4.1.4, translating raw egomotion signals into a few discrete motion patterns helps to simplify the design of our system. While one could collect new data explicitly controlling for the patterns (e.g., with a turntable and camera rig), we prefer a data-driven approach that can leverage video and ego-pose data collected "in the wild".

To this end, we discover clusters among pose difference vectors $\boldsymbol{y}_i - \boldsymbol{y}_j$ for pairs (i, j) of temporally close frames from video (typically less than 1 second apart; see Section 4.2.1 for details). For simplicity we apply k-means to find G clusters, though other methods are possible. Let $p_{ij} \in \mathcal{P} = \{1, \ldots, G\}$ denote the motion pattern ID, i.e., the cluster to which $(\boldsymbol{y}_i, \boldsymbol{y}_j)$ belongs. We can now replace the ego-pose vectors in $\mathcal{U}$ with motion pattern IDs: $\langle (\boldsymbol{x}_i, \boldsymbol{x}_j), p_{ij} \rangle$. [3]

[3]For movement with d degrees of freedom, setting $G \approx d$ should suffice (cf. Section 4.1.3).

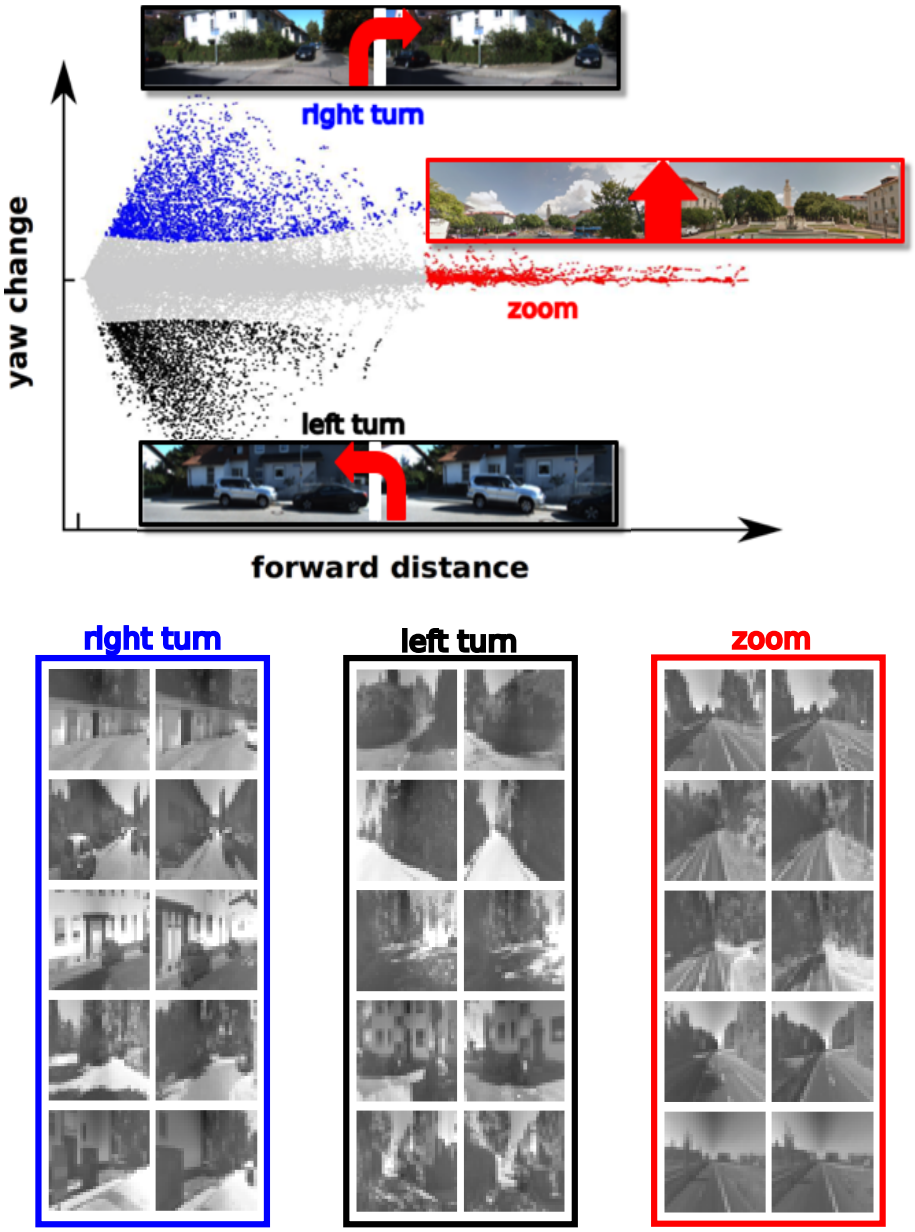


Figure 4.2: Motion pattern discovery in KITTI car-mounted videos. (Top) Largest motion clusters in the "forward distance"-"yaw" space correspond to forward motion or "zoom", "right turn" and "left turn" respectively. (Bottom) Some example pairs corresponding to discovered motion patterns. Within each box corresponding to one motion pattern, each row corresponds to a pair.

Figure 4.2 illustrates motion pattern discovery on frame pairs from the KITTI dataset [51, 52] videos, which are captured from a moving car. Here $\mathcal{Y}$ consists of the position and yaw angle of the camera. So, we are clustering a 2D space consisting of forward distance and change in yaw. As shown in the bottom panel, the largest clusters correspond to the car's three primary egomotions: turning left, turning right, and going forward.

In the next few sections, we present our method assuming discrete egomotions mined as above. Later, in Section 4.1.5 we discuss a variant of our approach that operates with non-discrete motion patterns.

4.1.3 Definition of egomotion equivariance

Given $\mathcal{U}$, we wish to learn a feature mapping function $\mathbf{z}_{\boldsymbol{\theta}}(.) : \mathcal{X} \rightarrow \mathcal{R}^D$ parameterized by $\boldsymbol{\theta}$ that maps a single image to a D-dimensional vector space that is equivariant to egomotion.

To define equivariance, it is convenient to start with the notion of feature invariance, which is the standard property that visual representations for recognition are designed to exhibit. Invariant features are *unresponsive* to certain classes of so-called "nuisance transformations" such as observer egomotions, pose change, or illumination change. For images $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ with associated ego-poses $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ respectively, an egomotion-invariant feature mapping function $\mathbf{z}_{\boldsymbol{\theta}}$ satisfies:

$$\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_j) \approx \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_i). \tag{4.1}$$

Section 4.1.4 discusses tradeoffs involved in selecting G. We chose a small value for G for efficiency and did not vary it in experiments.

Recall that this is the form of representation sought by many existing feature learning methods, including those that learn representations from video [26, 48, 62, 102, 118, 168, 195].

Rather than being *unresponsive* as above, equivariant functions are *predictably* responsive to transformations, i.e., an egomotion-equivariant function $\mathbf{z}_{\boldsymbol{\theta}}$ must respond systematically and predictably to egomotions:

$$\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_j) \approx f(\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i, \boldsymbol{y}_j), \tag{4.2}$$

for some simple function $f \in \mathcal{F}$, where again $\boldsymbol{y}_i$ denotes the ego-pose meta-data associated with video frame $\boldsymbol{x}_i$. Note that f must be *simple*; as the space of allowed functions $\mathcal{F}$ grows larger, the requirement in Eq (4.2) above is satisfied by more feature mapping functions $\mathbf{z}_{\boldsymbol{\theta}}$. In other words, as $\mathcal{F}$ grows large, the equivariance constraint on $\mathbf{z}_{\boldsymbol{\theta}}$ grows weak.

We will first consider equivariance for *linear* functions $f(.)$, following [98]. Later, in Section 4.1.5, we will show how to extend this to the non-linear case. In the linear case, $\mathbf{z}_{\boldsymbol{\theta}}$ is said to be equivariant with respect to some transformation g if there exists a $D \times D$ matrix[4] M_g such that:

$$\forall \boldsymbol{x} \in \mathcal{X} : \mathbf{z}_{\boldsymbol{\theta}}(g\boldsymbol{x}) \approx M_g \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}). \tag{4.3}$$

Such an M_g is called the "equivariance map" of g on the feature space $\mathbf{z}_{\boldsymbol{\theta}}(.)$. It represents the affine transformation in the feature space that corresponds to transformation g in the pixel space. For example, suppose a motion pattern g corresponds

[4]bias dimension assumed to be included in D for notational simplicity

to a yaw turn of 20°, and $\boldsymbol{x}$ and $g\boldsymbol{x}$ are the images observed before and after the turn, respectively. Equivariance demands that there is some matrix M_g that maps the pre-turn image to the post-turn image, once those images are expressed in the feature space $\mathbf{z}_{\boldsymbol{\theta}}$. Hence, $\mathbf{z}_{\boldsymbol{\theta}}$ "organizes" the feature space in such a way that movement in a particular direction in the feature space (here, as computed by matrix-vector multiplication with M_g) has a predictable outcome. The linear case, as also studied in [98], ensures that the structure of the mapping has a simple form—the space $\mathcal{F}$ of possible equivariance maps is suitably restricted so that the equivariance constraint is significant, as discussed above. It is also convenient for learning since M_g can be encoded as a fully connected layer in a neural network. In Section 4.2, we experiment with both linear and simple non-linear equivariance maps.

4.1.3.1 Equivariance in dynamic 3D scenes

While prior work [88, 141] focuses on equivariance where g is a 2D image warp, we explore the case where $g \in \mathcal{P}$ is an egomotion pattern (cf. Section 4.1.2) reflecting the observer's 3D movement in the world. In theory, appearance changes of an image in response to an observer's egomotion are not determined completely by the egomotion alone. They also depend on the depth map of the scene and the motion of dynamic objects in the scene. One could easily augment either the frames $\boldsymbol{x}_i$ or the ego-pose $\boldsymbol{y}_i$ with depth maps, when available. Non-observer motion appears more difficult, especially in the face of changing occlusions and newly appearing objects. Even accounting for everything, a future frame may never be fully predictable purely from egomotion alone, due to changing occlusions or newly visible elements in the

scene. However, our experiments indicate we can learn effective representations even with dynamic objects and changing occlusions. In our implementation, we train with pairs relatively close in time, so as to avoid some of these pitfalls.

4.1.3.2 Equivariance to composite motions

While during training we target equivariance for the discrete set of G egomotions, if we use linear equivariance maps as above, the learned feature space will *not* be limited to preserving equivariance for pairs originating from the same egomotions. This is because the linear equivariance maps are composable. If we are operating in a space where every egomotion can be composed as a sequence of "atomic" motions, equivariance to those atomic motions is sufficient to guarantee equivariance to all motions.

To see this, suppose that the maps for "turn head right by 10°" (egomotion pattern r) and "turn head up by 10°" (egomotion pattern u) are respectively M_r and M_u, i.e, $\mathbf{z}(r\boldsymbol{x}) = M_r\mathbf{z}(\boldsymbol{x})$ and $\mathbf{z}(u\boldsymbol{x}) = M_u\mathbf{z}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$. Now for a novel diagonal motion d that can be composed from these atomic motions as $d = r \circ u$ ("turn head up by 10°, then right by 10°"), we have:

$$\begin{aligned}\mathbf{z}(d\boldsymbol{x}) &= \mathbf{z}((r \circ u)\boldsymbol{x}) \\ &= M_r\mathbf{z}(u\boldsymbol{x}) \\ &= M_rM_u\mathbf{z}(\boldsymbol{x}),\end{aligned} \tag{4.4}$$

so that, setting $M_d := M_rM_u$, we have:

$$\mathbf{z}(d\boldsymbol{x}) = M_d\mathbf{z}(\boldsymbol{x}). \tag{4.5}$$

Comparing this against the definition of equivariance in Eq (4.3), we see that $M_d = M_r M_u$ is the equivariance map for the novel egomotion $d = r \circ u$, even though d was not among $1, \ldots, G$. This property lets us restrict our attention to a relatively small number of discrete egomotion patterns during training, and still learn features equivariant with respect to new egomotions. Section 4.1.5 presents a variant of our method that operates without discretizing egomotions.

4.1.4 Equivariant feature learning objective

We now design a loss function that encourages the learned feature space $\mathbf{z}_{\boldsymbol{\theta}}$ to exhibit equivariance with respect to each egomotion pattern. Specifically, we would like to learn the optimal feature space parameters $\boldsymbol{\theta}^*$ jointly with its equivariance maps $\mathcal{M}^* = \{M_1^*, \ldots, M_G^*\}$ for the motion pattern clusters 1 through G (cf. Section 4.1.2).

To achieve this, a naive translation of the definition of equivariance in Eq (4.3) into a minimization problem over feature space parameters $\boldsymbol{\theta}$ and the $D \times D$ equivariance map candidate matrices $\mathcal{M}$ (assuming linear maps) would be as follows:

$$(\boldsymbol{\theta}^*, \mathcal{M}^*) = \arg\min_{\boldsymbol{\theta}, \mathcal{M}} \sum_{g} \sum_{\{(i,j):p_{ij}=g\}} d\left(M_g \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_j)\right), \tag{4.6}$$

where $d(.,.)$ is a distance measure. This problem can be decomposed into G independent optimization problems, one for each motion, corresponding only to the inner summation above, and dealing with disjoint data. The g-th such problem requires only that training frame pairs annotated with motion pattern $p_{ij} = g$ approximately satisfy Eq (4.3).

However, similar to the problem encountered before in Section 3.1.2, such a formulation admits problematic solutions that perfectly optimize it. For example, for the trivial all-zero feature space $\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathbf{0}, \forall \boldsymbol{x} \in \mathcal{X}$ with M_g set to the all-zeros matrix for all g, the loss above evaluates to zero. To avoid such solutions, and to force the learned M_g's to be different from one another (since we would like the learned representation to respond *differently* to different egomotions), we simultaneously account for the "negatives" of each motion pattern. Our learning objective is:

$$(\boldsymbol{\theta}^*, \mathcal{M}^*) = \arg\min_{\boldsymbol{\theta}, \mathcal{M}} \sum_{g,i,j} d_g \left(M_g \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_j), p_{ij} \right), \tag{4.7}$$

where $d_g(.,.,.)$ is a "contrastive loss" [64] specific to motion pattern g:

$$d_g(\boldsymbol{a}, \boldsymbol{b}, c) = \mathbb{1}(c = g) d(\boldsymbol{a}, \boldsymbol{b}) + \mathbb{1}(c \neq g) \max(\delta - d(\boldsymbol{a}, \boldsymbol{b}), 0), \tag{4.8}$$

where $\mathbb{1}(.)$ is the indicator function. This contrastive loss penalizes distance between $\boldsymbol{a}$ and $\boldsymbol{b}$ in "positive" mode (when $c = g$), and pushes apart pairs in "negative" mode (when $c \neq g$), up to a minimum margin distance specified by the constant δ. We use the ℓ_2 norm for the distance $d(.,.)$.

In our objective in Eq (4.7), the contrastive loss operates in the latent feature space. For pairs belonging to cluster g, the contrastive loss d_g penalizes feature space distance between the first image and its transformed pair, similar to Eq (4.6) above. For pairs belonging to clusters other than g, the loss d_g requires that the transformation defined by M_g must not bring the image representations close together. In this way, our objective learns the M_g's jointly. It ensures that distinct egomotions, when

applied to an input $\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x})$, map it to different locations in feature space. We discuss how the feature mapping function parameters are optimized below in Section 4.1.6.

Note that the objective of Eq (4.8) depends on the choice G of the number of discovered egomotion patterns from Section 4.1.2. As remarked earlier, for movement with d degrees of freedom, setting $G \approx d$ should suffice (cf. Section 4.1.3). There are several tradeoffs involved in selecting G: (i) The more clusters, the fewer the training samples in each. This could lead to overfitting of equivariance maps M_g, so that optimizing Eq (4.8) may no longer produce truly equivariant features. (ii) The more the clusters, the more the number of parameters to be held in memory during training — each cluster has a corresponding equivariance map module. (iii) The fewer the clusters, the more noisy the training sample labels. Fewer clusters lead to larger clusters with more lossy quantization of egomotions in the training data. This might adversely affect the quality of training. In practice, for our experiments in Section 4.2, we observed that this dependence on G is not a problem — a small value for G is both efficient and produces good features. We did not vary G in experiments.

We now highlight the important distinctions between our objective of Eq (4.8) and the "temporal coherence" objective of [118], which is representative of works learning representations from video through slow feature analysis [26, 48, 62, 102, 168, 195]. Written in our notation, the objective of [118] may be stated as:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i,j} d_1(\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_j), \mathbb{1}(|t_i - t_j| \leq T)), \tag{4.9}$$

where t_i, t_j are the video time indices of $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ and T is a temporal neighborhood

size hyperparameter. This loss encourages the representations of nearby frames to be similar to one another, learning invariant representations. To see this, note how this loss directly optimizes representations to exhibit the invariance property defined in Eq (4.1). Observe that this is nearly identical to the first-order temporal coherence loss of Equation (3.4) (the only difference is the use of the ℓ_1 norm here rather than the ℓ_2 norm) that our "steady" features formulation in Chapter 3 generalizes to achieve higher order temporal coherence. However, temporal coherence alone does not account for the nature of the egomotion between the frames. Accordingly, while temporal coherence helps learn invariance to small image changes, it does not target a (more general) equivariant space. Like the passive kitten from Hein and Held's experiment, the temporal coherence constraint watches video to passively learn a representation; like the active kitten, our method registers the *observer motion* explicitly with the video to learn more effectively, as we will demonstrate in results.

4.1.5 Equivariance in non-discrete motion spaces with non-linear equivariance maps

Thus far, we have dealt with a discrete set of motions $\mathcal{G}$. When using linear equivariance maps, due to the composability of the maps, equivariance to all motions is guaranteed by achieving equivariance to only the discrete set of motions in $\mathcal{G}$, so long as those discrete motions span the full motion space (Section 4.1.3).

Still, the discrete motion solution has two limitations. First, it only generalizes to all egomotions for the restricted notion of equivariance relying on linear

maps, defined in Eq (4.3). In particular, for non-linear equivariance mapping functions $f(.)$ in the more general definition of equivariance in Eq (4.2), it does not guarantee equivariance to all egomotions. While linear maps nonetheless may be preferable for injecting stronger equivariance regularization effects, it is worth considering more general function families. Secondly, this discrete solution is lossy, as it requires discretizing the continuous space of all motions into specific clusters. More specifically, image pairs assigned to the same cluster may be related by slightly different observer motions. This information is necessarily ignored by the motion discretization solution.

On the other hand, directly learning an infinite number of equivariance maps M_g, one corresponding to each motion g in the training set, is intractable. In this section, we develop a variant of our approach that implicitly learns these infinite equivariance maps and allows it to naturally transcend the linearity constraint on equivariance maps.

We now describe this non-discrete variant of our method. The set of egomotions $\mathcal{G}$ may now be an infinite, uncountable set of motions. As an example, we will assume the set of all motions in the training set:

$$\mathcal{G} = \{\boldsymbol{y}_i - \boldsymbol{y}_j;\ i, j \ \text{ are temporally nearby frames in training video}\}, \tag{4.10}$$

where $\boldsymbol{y}_i$ is the ego-pose associated with frame $\boldsymbol{x}_i$, as defined before.

Now, rather than attempting to learn separate equivariant maps M_g for each motion $g \in \mathcal{G}$, we may parameterize the entire family of M_g's through a single matrix

function $\mathbf{M}$, as: $M_g = \mathbf{M}(g)$. Substituting this in Eq (4.3), we now want:

$$\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_j) \approx \mathbf{M}(\boldsymbol{y}_i - \boldsymbol{y}_j)\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_i). \tag{4.11}$$

At a high level, this may be thought of as similar to forming $G = \infty$ egomotion clusters for use with the discrete egomotions approach developed above (or more precisely, as many clusters as the number of egomotion-labeled training pairs, i.e., $G = N_u$). Until this stage, our equivariance maps remain linear, as in Eq (4.3). However, since we are no longer restricted to a discrete set of motions $\mathcal{G}$, we need no longer rely on the composability of linear equivariance maps. Instead, we can further generalize our maps as follows:

$$\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_j) \approx \mathbf{M}(\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i - \boldsymbol{y}_j), \tag{4.12}$$

where $\mathbf{M}$ is now a function that produces a vector in the learned feature space as output. Note how this compares against the general notion of equivariance first defined in Eq (4.2).

Our general "non-discrete" equivariance objective may now be stated as:

$$(\boldsymbol{\theta}^*, \mathbf{M}^*) = \arg\min_{\boldsymbol{\theta}, \mathbf{M}} \sum_{i,j} d\left(\mathbf{M}(\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i - \boldsymbol{y}_j), \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_j)\right), \tag{4.13}$$

where $d(.,.)$ is a distance measure. Note that the objective in Eq (4.13) parallels the alternative objective in Eq (4.7) for the discrete motion case.[5] The architecture of the function $\mathbf{M}(.)$ and how it is trained, are specified in Section 4.1.6 and Section 4.2.1.

[5]However, while the loss of Eq (4.7) is contrastive, Eq (4.13) specifies a non-contrastive loss. To overcome this deficiency in our experiments, we optimize this non-discrete equivariance loss only in conjunction with an auxiliary contrastive loss, such as DRLIM [64].

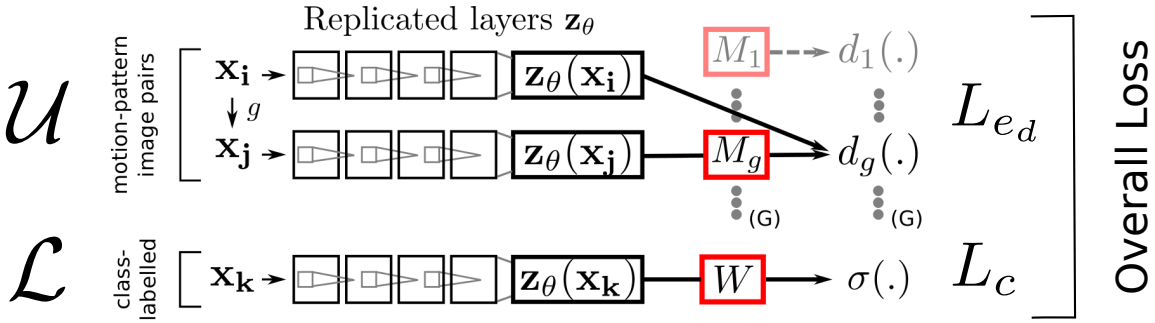


Figure 4.3: Training setup for discrete egomotions: (top) a two-stack "Siamese network" processes video frame pairs identically before optimizing the equivariance loss of Eq (4.7), and (bottom) a third layer stack simultaneously processes class-labeled images to optimize the supervised recognition softmax loss as in Eq (4.14). See Section 4.2.1 for exact network specifications.

This non-discretized motion and non-linear equivariance formulation allows an easy way to control the strength of the equivariance objective. The more complex the class of functions modeled by $\mathbf{M}(.)$, the weaker the notion of equivariance that is imposed upon the learned feature space. Moreover, it does not require discarding fine-grained information among the egomotion labels, as in the discrete motion case. We evaluate the impact of these conceptual differences in experiments (Section 4.2.4).

4.1.6 Form of the feature mapping function

For the mapping $\mathbf{z}_{\boldsymbol{\theta}}(.)$, we use a convolutional neural network architecture, so that the parameter vector $\boldsymbol{\theta}$ now represents the layer weights. We start with the discrete egomotions variant of our method. Let L_{e_d} denote the equivariance loss of Eq (4.7) based on discretized egomotions. L_{e_d} is optimized by sharing the weight parameters $\boldsymbol{\theta}$ among two identical stacks of layers in a "Siamese" network [23, 64, 118], as shown in the top two rows of Figure 4.3. Video frame pairs from $\mathcal{U}$ are

fed into these two stacks. Both stacks are initialized with identical random weights, and identical gradients are passed through them in every training epoch, so that the weights remain tied throughout. Each stack encodes the feature map that we wish to train, $\mathbf{z}_{\boldsymbol{\theta}}$.

To optimize Eq (4.7), an array of equivarance maps $\mathcal{M}$, each represented by a fully connected layer, is connected to the top of the second stack. Each such equivariance map then feeds into a motion-pattern-specific contrastive loss function d_g, whose other inputs are the first stack output and the egomotion pattern ID p_{ij}. This Siamese network architecture is depicted in the $\mathcal{U} \rightarrow L_{e_d}$ pipeline in Figure 4.3 (top).

Optimization is done through mini-batch stochastic gradient descent implemented through backpropagation with the Caffe package [82] (more details in Section 4.2 and the Appendix).

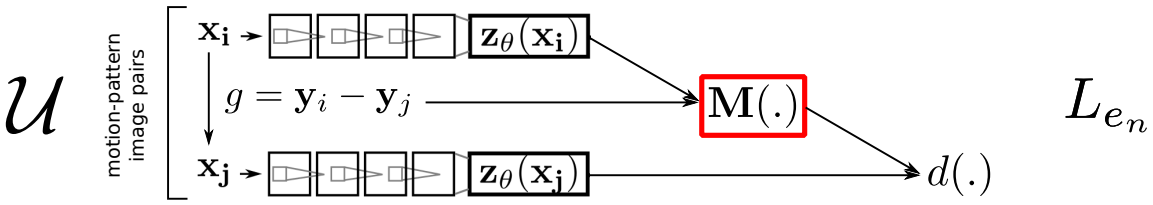


Figure 4.4: Unsupervised training setup for the "non-discrete" variant Eq (4.13) of the equivariance objective. First, layer stacks with tied weights, representing the feature mapping $\mathbf{z}_{\boldsymbol{\theta}}$ to be learned, process video frame pairs identically to embed them into a feature space. In this space, an equivariance mapping function $M(.)$ acts on the first frame and camera egomotion vector g to attempt to predict the second frame. See Figure 4.5 for the architecture of $\mathbf{M}(.)$, and Section 4.2.1 for further details. When used in a regularization setup with labeled data $\mathcal{L}$, a third stack of layers may be added as in Figure 4.3 to compute the classification loss.

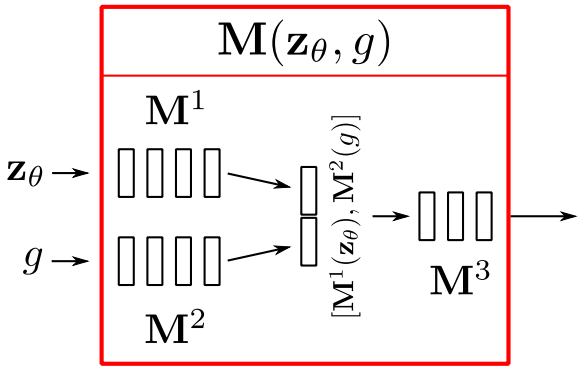


Figure 4.5: Architecture of the module $\mathbf{M}(.)$ used for optimizing the non-discrete equivariance objective of Eq (4.13). The feature vector $\mathbf{z}_{\boldsymbol{\theta}}$ and the continuous egomotion vector $g = \boldsymbol{y}_i - \boldsymbol{y}_j$ are processed through separate neural network modules $\mathbf{M}^1$ and $\mathbf{M}^2$ respectively, before appending and processing through a final module $\mathbf{M}^3$ that produces an output in the same domain as the input $\mathbf{z}_{\boldsymbol{\theta}}$. Figure 4.4 shows where this fits into the full Siamese network framework.

For the case of the non-discretized motions variant of our approach in Eq (4.13), let L_{e_n} denote the equivariance loss of Eq (4.13). L_{e_n} is optimized as follows. The array of equivariance maps $\mathcal{M}$ is replaced by a single module $\mathbf{M}(\mathbf{z}_{\boldsymbol{\theta}}, g)$, as shown in Figure 4.4. The architecture of $\mathbf{M}(\mathbf{z}_{\boldsymbol{\theta}}, g)$ is specified in Figure 4.5. The feature vector $\mathbf{z}_{\boldsymbol{\theta}}$ and the non-discrete egomotion $g = \boldsymbol{y}_i - \boldsymbol{y}_j$ are processed through separate neural network modules $\mathbf{M}^1$ and $\mathbf{M}^2$ before appending and processing through a final module $\mathbf{M}^3$. The specific architectures of these internal modules $\mathbf{M}^1, \mathbf{M}^2, \mathbf{M}^3$ are specified in Section 4.2.

It is worth reiterating that the architectures of equivariance maps define the nature of the desired equivariance, and control the strength of the equivariance objective: broadly, we expect that the more complex the architecture, the weaker the equivariance regularization is. An equivariance map architecture that is very simple

could lead to heavy regularization, but equally, complex architectures might back-propagate no regularizing gradients to the base learned features $\mathbf{z}_{\boldsymbol{\theta}}$, since they might be able to represented overfitted equivariance maps even for arbitrary input features, such as features from a randomly initialized neural network.

4.1.7 Applying learned equivariant representations to recognition tasks

While we have thus far described our formulation for generic equivariant image representation learning, our hypothesis is that representations trained as above will facilitate high-level visual tasks such as recognition. One way to see this is by observing that equivariant representations expose camera and object pose-related parameters to a recognition algorithm, which may then account for this critical information while making predictions. For instance, a feature space that embeds knowledge of how objects change under different viewpoints or manipulations may allow a recognition system to hallucinate new views (in that feature space) of an object to improve performance.

More generally, recall the intuitions gained from the view prediction task illustrated in Figure 1.4. As discussed in Section 1.2, acquiring the ability to hallucinate future views in severely underdetermined situations requires mastery of complex visual skills like depth, 3D geometry, semantics, and context. Therefore, our equivariance formulation of this view prediction task within the learned feature space induces the development of these ancillary high-level skills, which are transferable to other high-level tasks like object or scene recognition.

Suppose that in addition to the ego-pose annotated pairs $\mathcal{U}$ we are also given a

small set of N_l class-labeled static images, $\mathcal{L} = \{(\boldsymbol{x}_k, c_k)\}_{k=1}^{N_l}$, where $c_k \in \{1, \ldots, C\}$. We may now adapt our equivariance formulation to enable the training of a recognition pipeline on $\mathcal{L}$. In our experiments, we do this in two settings, purely unsupervised feature extraction (Section 4.2.4), and unsupervised *regularization* of the supervised recognition task (Section 4.2.5). We now describe the approaches for these two settings in detail.

In both of the scenarios below, note that neither the supervised training data $\mathcal{L}$ nor the testing data for recognition are required to have any associated sensor data. Thus, our features are applicable to standard image recognition tasks.

4.1.7.1 Adapting unsupervised equivariant features for recognition

In the unsupervised setting, we first train representations by optimizing the equivariance objective of Eq (4.7) (or Eq (4.13) for the non-discrete case). We then directly represent the class-labeled images from $\mathcal{L}$ in our learned equivariant feature space. These features may then be input to a generic machine learning pipeline, such as a k-nearest neighbor classifier, that is to be trained for recognition using labeled data $\mathcal{L}$. Alternatively, the weights learned in the network may be finetuned using the labeled data $\mathcal{L}$, producing a neural network classifier.

This setting allows us to test if optimizing neural networks *only* for equivariant representations, with no explicit *discriminative* component in the loss function, still produces discriminative representations. Aside from testing the power of our equivariant feature learning objective in isolation, this setting allows a nice modularity between the feature learning step and category learning step. In particular,

when learned prior to any recognition task, our features can be used for easy "off-the shelf" testing of the unsupervised neural network directly as a feature extractor for new tasks. The user does not need to simultaneously optimize the embedding parameters and classifier parameters specific to his task. Moreoever, it requires no more computational resources than for the Siamese paired network scheme described in Section 4.1.6 for learning equivariant representations.

4.1.7.2 Unsupervised equivariance regularization for recognition

Alternatively, we may *jointly* train representations for equivariance, as well as for discriminative ability geared towards a target recognition task, similar to the setup we used in the last chapter (see Section 3.1.1). Let L_{e_d} denote the unsupervised equivariance loss of Eq (4.7). We can integrate our unsupervised feature learning scheme with the recognition task, by optimizing a misclassification loss together with L_{e_d}. Let W be a $C \times D$ matrix of classifier weights. We solve jointly for W and the maps $\mathcal{M}$:

$$(\boldsymbol{\theta}^*, W^*, \mathcal{M}^*) = \underset{\boldsymbol{\theta}, W, \mathcal{M}}{\arg\min}\, L_c(\boldsymbol{\theta}, W, \mathcal{L}) + \lambda L_e(\boldsymbol{\theta}, \mathcal{M}, \mathcal{U}), \tag{4.14}$$

where L_c denotes the softmax loss over the learned features:

$$L_c(W, \mathcal{L}) = -\frac{1}{N_l} \sum_{i=1}^{N_l} \log(\sigma_{c_k}(W \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)), \tag{4.15}$$

and $\sigma_{c_k}(.)$ is the softmax probability of the correct class.

$$\sigma_{c_i}(\boldsymbol{p_i}) = \exp(p_{c_i}) / \sum_{c=1}^{C} \exp(p_c). \tag{4.16}$$

The regularizer weight λ in Eq (4.14) is a hyperparameter.

In this setting, the unsupervised egomotion equivariance loss encodes a prior over the feature space that can improve performance on the supervised recognition task with limited training examples.

To optimize Eq (4.14), in addition to the Siamese net that minimizes L_e as above, the supervised softmax loss is minimized through a third replica of the $\mathbf{z}_{\boldsymbol{\theta}}$ layer stack with weights tied to the two Siamese networks stacks. Labelled images from $\mathcal{L}$ are fed into this stack, and its output is fed into a softmax layer whose other input is the class label. So while this is a more complete framework for applying our equivariant representations to recognition tasks, it is also more computationally intensive; compared to Section 4.1.7.1, it requires more memory, more computation per iteration, and more iterations for convergence due to the more complex objective function. The complete scheme is depicted in Figure 4.3.

4.1.7.3 Equivariant representations for next-best view selection

Next, we model a situation where an agent equipped with equivariant visual representations has the ability to *act* on the real world at test time. Specifically, given one view of an object, the agent must decide how to move next to help recognize the object, i.e., which neighboring view would best reduce object category prediction uncertainty. This task is illustrated in Figure 4.6. Precisely because our features are equivariant (i.e., behave predictably) with respect to egomotion, we can exploit them to "envision" the next views that are possible and choose the most valuable one accordingly.

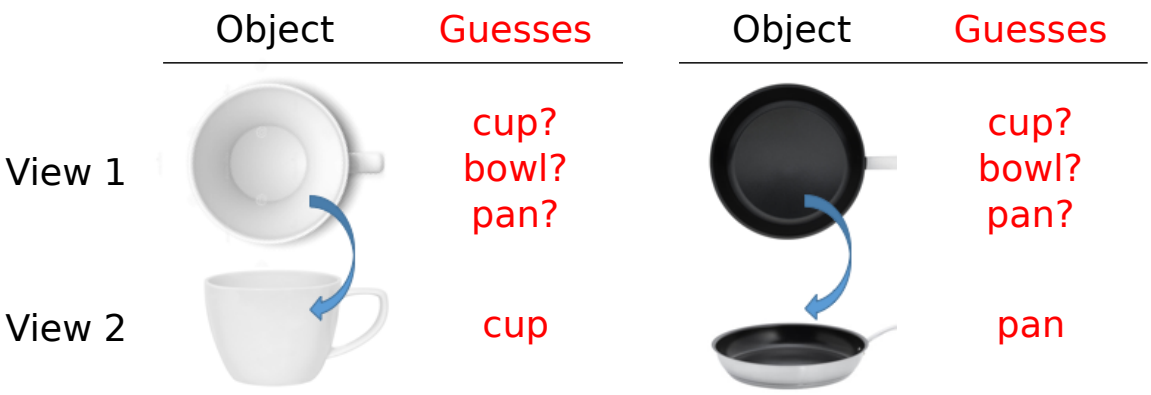


Figure 4.6: Illustration of "next-best-view" selection for recognition. Suppose that a robot, having observed one view of an object (top) is not immediately confident of the category of the object. In the next-best view setting, it can then select to move around the object (or manipulate the object) intelligently, to disambiguate among its top competing hypotheses.

We now describe a simple method for this task, similar in spirit to [176].[6] We limit the choice of next view g to { "up", "down", "up+right" and "up+left" } for simplicity. First, we build a k-nearest neighbor (k-NN) image-pair classifier for each possible g, using only training image pairs $(\boldsymbol{x}, g\boldsymbol{x})$ related by the egomotion g. This classifier C_g takes as input a vector of length $2D$, formed by appending the features of the image pair (each image's representation is of length D) and produces the output probability of each class. So, $C_g([\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}),\ \mathbf{z}_{\boldsymbol{\theta}}(g\boldsymbol{x})])$ returns class likelihood probabilities for all C classes. Output class probabilities for the k-NN classifier are computed from the histogram of class votes from the k nearest neighbors.

At test time, we first compute features $\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_0)$ on the given starting image

[6] Later, in Chapter 6, we will explore more sophisticated solutions for such active recognition problems.

$\boldsymbol{x}_0$. Next, we predict the feature $\mathbf{z}_{\boldsymbol{\theta}}(g\boldsymbol{x}_0)$ corresponding to each possible surrounding view g, as $M_g\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_0)$, per the definition of equivariance (cf. Eq (4.3)).

With these predicted transformed image features and the pair-wise nearest neighbor class probabilities $C_g(.)$, we may now pick the next-best view as:

$$g^* = \arg\min_g H(C_g([\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_0),\ M_g\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x_0})])), \tag{4.17}$$

where $H(.)$ is the information-theoretical entropy function. This selects the view that would produce the least predicted image pair class prediction uncertainty.

This simple mutual information-based approach to next-best view selection for exploiting egomotion-equivariant representations is intended only as a preliminary demonstration of the suitability of egomotion-equivariant representations for active recognition. In Chapter 6, we will go beyond this approach, showing how an egomotion-equivariance-based "lookahead" regularizer helps in the development of an end-to-end active recognition system.

4.2 Experiments

We validate our approach on three public datasets and compare to multiple existing methods. The main questions we address in the experiments are: (i) quantitatively, how well is equivariance preserved in our learned embedding? (Section 4.2.2); (ii) qualitatively, can we see the egomotion consistency of embedded image pairs? (Section 4.2.3); (iii) when learned entirely without supervision, how useful are our method's features for recognition tasks? (Section 4.2.4); (iv) when used as a regularizer for a classification loss, how effective are our method's fea-

tures for recognition tasks? (Section 4.2.5); and (v) how effective are the learned equivariant features for next-best view selection in an active recognition scenario? (Section 4.2.6).

Throughout, we compare the following methods:

- CLSNET: A neural network trained only from the supervised samples with a softmax loss.

- TEMPORAL: The *temporal coherence* approach of [118], which regularizes the classification loss with Eq (4.9) setting the distance measure $d(.)$ to the ℓ_1 distance in d_1. This method aims to learn invariant features by exploiting the fact that adjacent video frames should not change too much.

- DRLIM: The approach of [64], which also regularizes the classification loss with Eq (4.9), but setting $d(.)$ to the ℓ_2 distance in d_1.

- LSM: The "learning to see by moving" (LSM) approach of [3], proposed independently and concurrently with our method, which also exploits video with accompanying egomotion for unsupervised representation learning. LSM uses egomotion in an alternative approach to ours; it trains a neural network to predict the observer egomotion g, given views $\boldsymbol{x}$ and $g\boldsymbol{x}$, before and after the motion. In our experiments, we use the publicly available KITTI-trained model provided by the authors.

- EQUIV: Our egomotion equivariant feature learning approach, as defined by the objective of Eq (4.7).

- EQUIV+DRLIM: Our approach augmented with temporal coherence regularization ([64]).

- EQUIV+DRLIM (non-discrete): The non-discrete motion variant of our approach as defined by the objective of Eq (4.12), augmented with temporal coherence regularization.[7]

All of these baselines are identically augmented with a classification loss for the regularization-based experiments in Section 4.2.2, 4.2.5 and 4.2.6. TEMPORAL and DRLIM are the most pertinent baselines for validating our idea of exploiting egomotion for visual learning, because they, like us, use contrastive loss-based formulations, but represent the popular "slowness"-based family of techniques ([26, 48, 62, 102, 195]) for unsupervised feature learning from video, which, unlike our approach, are passive. In addition, our results against LSM evaluate the strength of our egomotion-equivariance formulation against the alternative approach of [3].

4.2.1 Experimental setup

Recall that in the fully unsupervised mode, our method trains with pairs of video frames annotated only by their ego-poses in $\mathcal{U}$. In the supervised mode, when applied to recognition, our method additionally has access to a set of class-labeled images in $\mathcal{L}$. Similarly, the baselines all receive a pool of unsupervised data and

[7]Note that we do not test EQUIV (non-discrete), i.e., the non-discrete formulation of Eq (4.13) in isolation. This is because Eq (4.13) specifies a non-contrastive loss that would result in collapsed feature spaces (such as $\mathbf{z}_{\boldsymbol{\theta}} = \mathbf{0} \forall \boldsymbol{x}$) if optimized in isolation. To overcome this deficiency, we optimize this non-discrete equivariance loss only in conjunction with the contrastive DRLIM loss in the EQUIV+DRLIM (non-discrete) approach.

supervised data. We now detail the data composing these two sets. Figure 4.7 shows samples from the various datasets used in our experiments in this chapter.

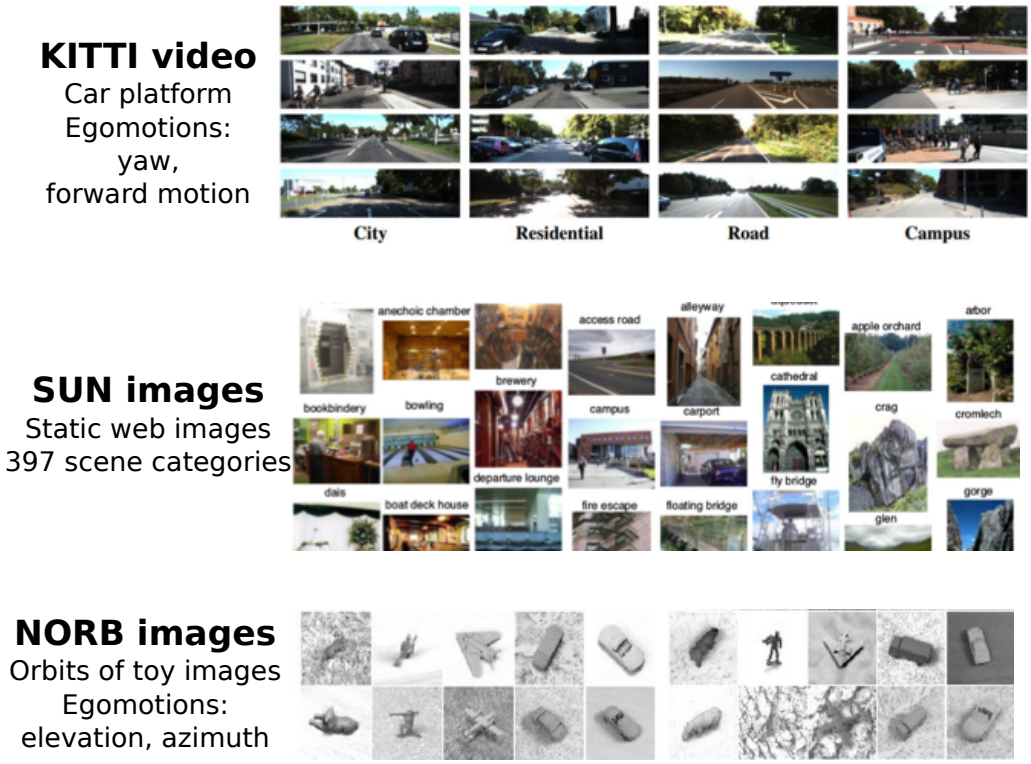


Figure 4.7: We test our method across diverse standard datasets: (Top) Figure from [52] showcasing images from the four KITTI location classes (shown here in color; we use grayscale images), (Middle) Figure from [177] showcasing images from a subset of the 397 SUN classes (shown here in color; see text for image pre-processing details). (Bottom) Figure from [96], showing images of toys captured from varying camera poses and with varying backgrounds. In some of our experiments, we learn representations from KITTI videos and apply them to SUN scene recognition. Note how these two datasets vary greatly in content. In KITTI, the camera always faces a road, and it has a fixed field of view and camera pitch, and the content is entirely street scenes around Karlsruhe. In SUN, the images are downloaded from the internet, and belong to 397 diverse indoor and outdoor scene categories—most of which have nothing to do with roads.

Unsupervised datasets We consider two unsupervised datasets, NORB and KITTI, to compose the unlabeled video pools $\mathcal{U}$ augmented with egomotion.

- **NORB** [96]: This dataset has 24,300 96×96-pixel images of 25 toys captured by systematically varying camera pose. We generate a random 67%-33% train-validation split and use 2D ego-pose vectors $\boldsymbol{y}$ consisting of camera elevation and azimuth. Because this dataset has discrete ego-pose variations, we consider two egomotion patterns, i.e, $G = 2$ (cf. Section 4.1.2): one step along elevation and one step along azimuth. For EQUIV, we use all available positive pairs for each of the two motion patterns from the training images, yielding a $N_u = 45,417$-pair training set. For DRLIM and TEMPORAL, we create a 50,000-pair training set (positives to negatives ratio 1:3). Pairs within one step (elevation and/or azimuth) are treated as "temporal neighbors", as in the turntable results of [64, 118].

- **KITTI** [51, 52]: This dataset contains videos with registered GPS/IMU sensor streams captured on a car driving around four types of areas (location classes): "campus", "city", "residential", "road". We generate a random 67%-33% train-validation split and use 2D ego-pose vectors consisting of "yaw" and "forward position" (integral over "forward velocity" sensor outputs) from the sensors. We discover egomotion patterns p_{ij} (cf. Section 4.1.2) on frame pairs ≤ 1 second apart. We compute 6 clusters and automatically retain the $G = 3$ with the largest motions, which upon inspection correspond to "forward motion/zoom", "right turn", and "left turn" (see Figure 4.2). For EQUIV, we create a $N_u =$

47,984-pair training set with 11,996 positives. For DRLIM and TEMPORAL, we create a 98,460-pair training set with 24,615 "temporal neighbor" positives sampled $\leq$2 seconds apart.[8] Of the various KITTI cameras that simultaneously capture video, we use the feed from "camera 0" (see [51] for details) in our expriments. For our unsupervised pretraining experiments, we reproduce the setting of [3] to allow fair comparison, cropping random 227×227 images from the original 370×1226 video frames to create the unlabeled dataset $\mathcal{U}$ ("KITTI-227"), before optimizing the objective of Eq (4.7). In Section 4.2.5, when testing the more computationally demanding regularization pipeline of 4.1.7.2 (cf. discussion in Section 4.1.7), we use grayscale, downsampled 32×32 pixel frames, so that (i) fast and thorough experiments are still possible, (ii) we can adopt CNN architecture choices known to be effective for tiny images [1], described below, and (iii) model complexity can be kept low enough so that our unsupervised training datasets are not too small.[9]

Supervised datasets In our recognition experiments, we consider three supervised datasets $\mathcal{L}$. These datasets allow us to test our approach's impact for three distinct recognition tasks for static images: object instance recognition, location recognition, and scene recognition. The supervised datasets are:

[8]For fairness, the training frame pairs for each method are drawn from the same starting set of KITTI training videos.

[9]Note that while the number of frame pairs N_u may be different for different methods, all methods have access to the same training videos, so this is a fair comparison. The differences in N_u are due to the methods themselves. For example, on KITTI data, EQUIV selectively uses frame pairs corresponding to large motions (Section 4.1.2), so even given the same starting videos, it is restricted to using a smaller number of frame pairs than DRLIM and TEMPORAL.

- **NORB**: We select six images from each of the $C = 25$ object training splits at random to create instance recognition training data.
- **KITTI**: We select four images from each of the $C = 4$ location class training splits at random to create location recognition training data.
- **SUN** [177]: We select six images for each of $C = 397$ scene categories at random from the standard training dataset to create scene recognition training data, unless otherwise stated. We preprocess them identically to the KITTI images above for all experiments. For the purely unsupervised experiments, we follow the setting of [3], resizing images to 256×256 before cropping random 227×227 regions ("SUN-227").

We keep all the supervised datasets small, since unsupervised feature learning should be most beneficial when labeled data is scarce. This corresponds to handling categorization problems in the "long tail". Note that while the video frames of the unsupervised datasets $\mathcal{U}$ are associated with ego-poses, the static images of $\mathcal{L}$ have no such auxiliary data.

Network architectures and optimization We now discuss the neural network architectures used for the base network $\mathbf{z}_{\boldsymbol{\theta}}$ and the equivariance maps in various experimental settings. These architectures are chosen to be appropriate for the size and complexity of images (or video frames) in each dataset.

For NORB, $\mathbf{z}_{\boldsymbol{\theta}}$ is a fully connected network: 20 full-ReLU$\rightarrow$ D =100 full feature units. M_g is a single fully connected layer Linear(100,100). These are schemat-

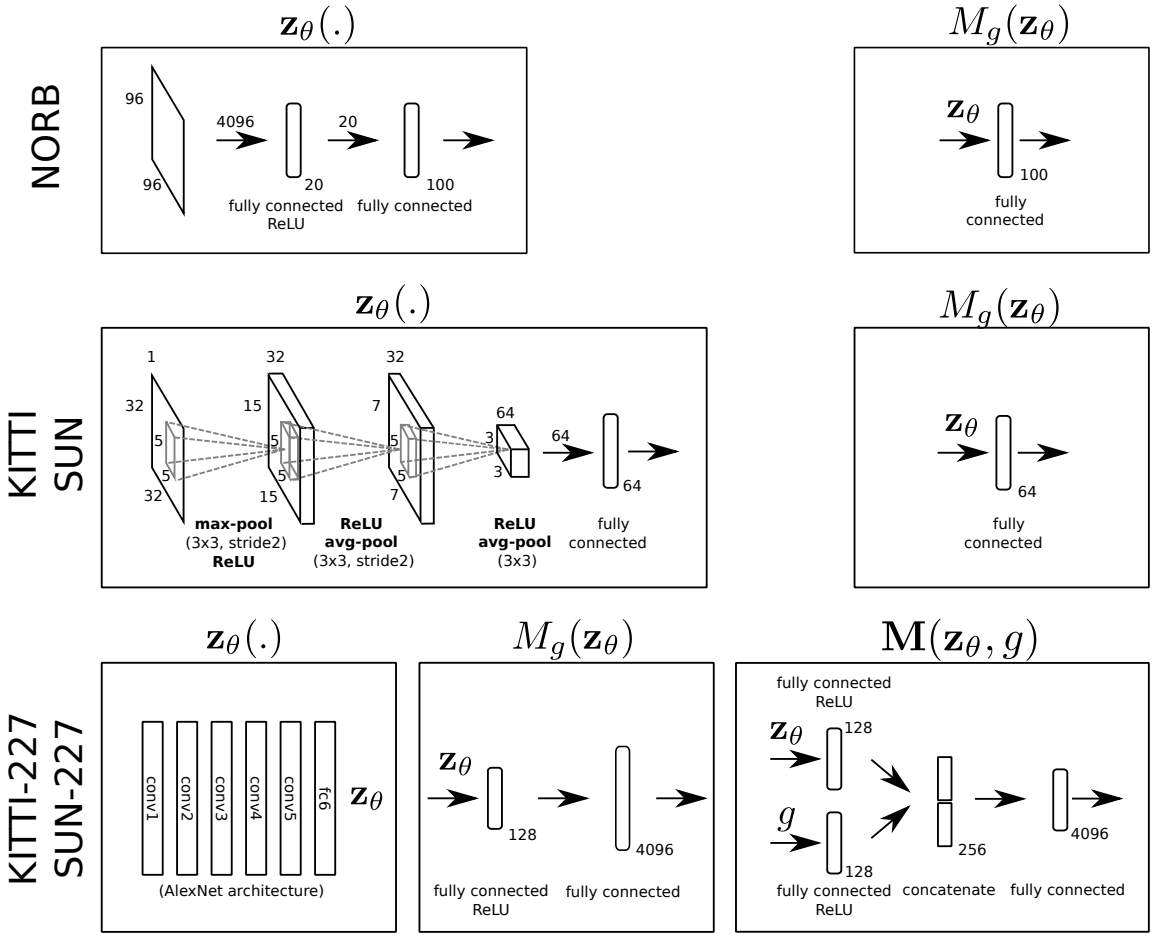


Figure 4.8: Schematics representing neural network architectures used in various experimental settings, for the base network $\mathbf{z}_{\boldsymbol{\theta}}$ and the equivariance maps M_g (for the discrete case) and $\mathbf{M}(\mathbf{z}_{\boldsymbol{\theta}},_g)$ (for the non-discrete case).

ically depicted in Figure 4.8 (top row).

For KITTI, the base neural network $\mathbf{z}_{\boldsymbol{\theta}}$ closely follows the cuda-convnet [1] recommended CIFAR-10 architecture: 32 Conv(5x5)-MaxPool(3x3)-ReLU $\rightarrow$ 32 Conv(5x5)-ReLU-AvgPool(3x3) $\rightarrow$ 64 Conv(5x5)-ReLU-AvgPool(3x3) $\rightarrow$ D =64 full feature units. The equivariance map M_g is a single fully connected layer Linear(64,64),

which takes in 64-dimensional $\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x})$ as input, and produces 64-dimensional $M_g\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x})$ as output. Figure 4.8 (middle row) presents schematics of these architectures.

For experiments with KITTI-227 and SUN-227, we follow the standard AlexNet architecture, augmented for fast training with batch normalization [74] (before every layer with learnable weights - *conv1-5*, *fc6*). We truncate the AlexNet architecture at the first fully connected layer, *fc6*, treating its output as the feature representation $\mathbf{z}_{\boldsymbol{\theta}}$. For EQUIV+DRLIM(discrete), the equivariance map modules M_g have the architecture: input $\rightarrow$ Linear(4096,128) $\rightarrow$ ReLU $\rightarrow$ Linear(128,4096), that produces a feature in the original 4096-dim feature space.[10] For EQUIV+DRLIM(non-discrete), the architecture of the equivariance map module $\mathbf{M}(.)$ follows the outline in Section 4.1.6 and Figure 4.5. Specifically,

- Module $\mathbf{M}^1$, which processes the 4096-dimensional output of *fc6*, has the architecture: input$\rightarrow$ Linear(4096,128)$\rightarrow$ReLU, producing a 128-dimensional output.

- Module $\mathbf{M}^2$, which processes the 2-dimensional continuous egomotion label, has the architecture: input$\rightarrow$ Linear(2,128)$\rightarrow$ReLU, producing a 128-dimensional output.

- Module $\mathbf{M}^3$, which processes the 256-dimensional concatenated outputs of the first two modules, has the architecture: input$\rightarrow$ Linear(256,4096), producing

[10]We do not use a straightforward fully connected layer Linear(4096,4096) as this would drastically increase the number of network parameters, and possibly cause overfitting of M_g, back-propagating poor equivariance regularization gradients through to the base network $\mathbf{z}_{\boldsymbol{\theta}}$. The M_g architecture we use in its place is non-linear due to the ReLU units.

a vector in the original 4096-dimensional feature space.

These architectures for KITTI-227 and SUN-227 experiments are shown in Figure 4.8 (bottom row).

We use Nesterov-accelerated stochastic gradient descent. The base learning rate and regularization λs are selected with greedy cross-validation. We report all results for all methods based on five repetitions.

4.2.2 Quantitative analysis: equivariance measurement

First, we test the learned features for equivariance. Equivariance is measured separately for each egomotion g through the normalized error ρ_g:

$$\rho_g = E\left[\frac{\|M_g^{'}\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \mathbf{z}_{\boldsymbol{\theta}}(g\boldsymbol{x})\|_2}{\|\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \mathbf{z}_{\boldsymbol{\theta}}(g\boldsymbol{x})\|_2}\right], \tag{4.18}$$

where $E[.]$ denotes the empirical mean, $M_g^{'}$ is the equivariance map, and $\rho_g = 0$ would signify perfect equivariance. To understand this error measure, we start by noting that the numerator is directly related to the definition of equivariance in Eq (4.3): $\mathbf{z}_{\boldsymbol{\theta}}(g\boldsymbol{x}) \approx M_g\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x})$. Thus, the numerator alone constitutes the most straightforward measure of equivariance error. However, this term depends on the *scale* of the feature representation, which may vary between methods. So, rather than measure the distance between the transformed and ground truth features directly, ρ_g measures the *ratio* by which $M_g^{'}$ *reduces* the distance between $\mathbf{z}_{\boldsymbol{\theta}}(g\boldsymbol{x})$ and $\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x})$. The denominator and numerator in Eq (4.18) are therefore the distance between the representations of original and transformed images, respectively *before* and *after*

applying the equivariance map. The normalized error ρ_g is the empirical mean of this distance reduction ratio across all samples.

We closely follow the equivariance evaluation approach of [98] to solve for the equivariance maps of features produced by each compared method on held-out validation data, before computing ρ_g. Such maps are produced explicitly by our method, but not the baselines. Thus, as in [98], we compute their maps[11] by solving a least squares minimization problem based on the definition of equivariance in Eq (4.3):

$$M'_g = \arg\min_M \sum_{m(\boldsymbol{y}_i, \boldsymbol{y}_j)=g} \|\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - M\mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}_j)\|_2. \tag{4.19}$$

The equivariance maps M'_g computed as above are used to compute the normalized errors ρ_g as in Eq (4.18). M'_g and ρ_g are computed on disjoint subsets of the validation image pairs.

We test both (i) "atomic" egomotions matching those provided in the training pairs (i.e, "up" 5°and "down" 20°) and (ii) composite egomotions ("up+right", "up+left", "down+right"). The latter lets us verify that our method's equivariance extends beyond those motion patterns used for training (cf. Section 4.1.3).

First, as a sanity check, we quantify equivariance for the unsupervised loss of Eq (4.7) in isolation, i.e, learning with only $\mathcal{U}$. Our EQUIV method's average ρ_g error is 0.0304 and 0.0394 for atomic and composite egomotions in NORB, respectively. In comparison, DRLIM—which promotes invariance, not equivariance—achieves $\rho_g =$

[11] For uniformity, we do the same recovery of M'_g for our method; our results are similar either way.

Motion types →	atomic			composite			
Methods ↓	"up (u)"	"right (r)"	avg.	"u+r"	"u+l"	"d+r"	avg.
random	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
CLSNET	0.9276	0.9202	0.9239	0.9222	0.9138	0.9074	0.9145
TEMPORAL [118]	0.7140	0.8033	0.7587	0.8089	0.8061	0.8207	0.8119
DRLIM [64]	0.5770	0.7038	0.6404	0.7281	0.7182	0.7325	0.7263
EQUIV	0.5328	0.6836	**0.6082**	0.6913	0.6914	0.7120	**0.6982**
EQUIV+DRLIM	**0.5293**	**0.6335**	**0.5814**	**0.6450**	**0.6460**	**0.6565**	**0.6492**

Table 4.1: The "normalized error" equivariance measure ρ_g for individual egomotions (Eq (4.18)) on NORB, organized as "atomic" (motions in the EQUIV training set) and "composite" (novel) egomotions. This metric captures how well equivariance is preserved in the embedding space. Lower values are better.

0.3751 and 0.4532. Thus, without class supervision, EQUIV tends to learn nearly completely equivariant features, even for novel composite transformations.

Next, we evaluate equivariance for all methods using features optimized for the NORB recognition task. Table 4.1 shows the results. As expected, we find that the features learned with EQUIV regularization are again easily the most equivariant. Normalized errors are lower for smaller motions than for larger motions, e.g., all methods do better on the atomic motion "u" (up by 5°) than on the other atomic motion "r" (right by 20°). Naturally, this also means error must be lower for atomic motions than for composite motions, since the latter are combinations of two atomic motions. This is confirmed by the results in Table 4.1.

Finally, we run similar experiments on the more challenging KITTI-227 data. Over the three egomotion clusters on KITTI-227, DRLIM *fc6* features achieved an average equivariance error ρ_g of 0.7791. In comparison, EQUIV produced significantly more equivariant features as expected, yielding average equivariance error 0.7315.

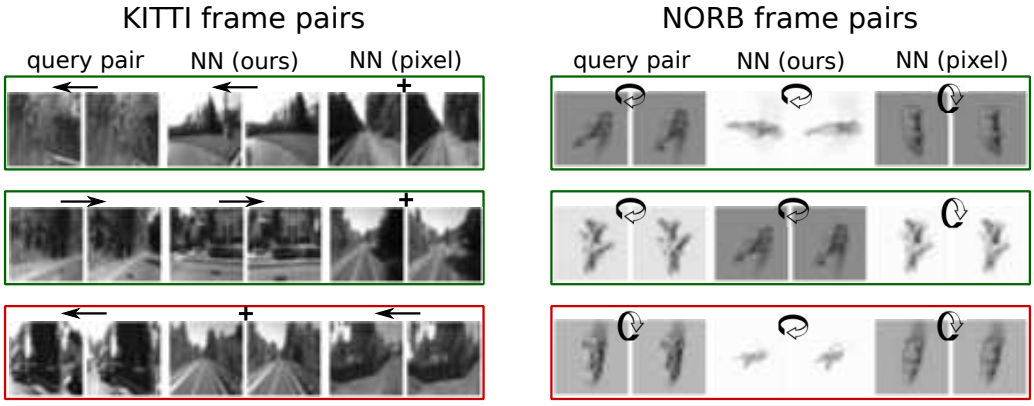


Figure 4.9: Nearest neighbor image pairs (columns 3 and 4 in each block) in pairwise equivariant feature difference space for various query image pairs (columns 1 and 2 per block). For comparison, columns 5 and 6 show pixel-wise difference-based neighbor pairs. The direction of egomotion in query and neighbor pairs (inferred from ego-pose vector differences) is indicated above each block. See text.

To estimate how much more egomotion-equivariance may be beneficial for generic visual features, we now compare these unsupervised models against a fully supervised model ("IMAGENET-SUP" [92]) with the same standard AlexNet architecture as our models, but trained on ImageNet [35], a large manually curated classification dataset with millions of labeled images. Features extracted from such models are among the most widely used representations for various computer vision tasks today [41]. Fully supervised IMAGENET-SUP *fc6* features achieve 0.6285 average error, indicating significant egomotion-equivariance. We view the equivariance of these standard, widely used neural network features trained on labeled classification datasets as validation that equivariance to egomotions may be a useful desideratum for learning good generic visual features in an unsupervised manner, as our method aims to do.

4.2.3 Qualitative analysis: detecting image pairs with similar motions

To qualitatively evaluate the impact of equivariant feature learning, we pose a nearest neighbor task in the *feature difference* space to retrieve image pairs related by similar egomotion to a query image pair.

Given a learned feature space $\mathbf{z}(.)$ and a query image pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$, we form the pairwise feature difference $\boldsymbol{d}_{ij} = \mathbf{z}(\boldsymbol{x}_i) - \mathbf{z}(\boldsymbol{x}_j)$. In an equivariant feature space, other image pairs $(\boldsymbol{x}_k, \boldsymbol{x}_l)$ with similar feature difference vectors $\boldsymbol{d}_{kl} \approx \boldsymbol{d}_{ij}$ would be likely to be related by similar egomotion to the query pair.[12] This can also be viewed as an analogy completion task, $\boldsymbol{x}_i : \boldsymbol{x}_j = \boldsymbol{x}_k :?$, where the right answer $\boldsymbol{x}_l$ must be computed by applying the unknown transformation $\boldsymbol{p}_{ij}$ to $\boldsymbol{x}_k$.

Figure 4.9 shows examples from KITTI (top) and NORB (bottom). For a variety of query pairs, we show the top neighbor pairs in the EQUIV space, as well as in pixel-difference space for comparison. Overall they visually confirm the desired equivariance property: neighbor-pairs in EQUIV's difference space exhibit a similar transformation (e.g., turning and zooming), whereas those in the original image space often do not. Consider the first NORB example (first row among NORB examples), where pixel distance, perhaps dominated by the lighting, identifies a wrong egomotion match, whereas our approach finds a correct match, despite the changed object identity, starting azimuth, lighting etc. The red boxes show failure cases. For instance, in the last KITTI example (third row), large foreground motion

[12]Note that in our model of equivariance, this is not strictly true, since the pair-wise difference vector $M_g \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x}) - \mathbf{z}_{\boldsymbol{\theta}}(\boldsymbol{x})$ need not actually be consistent across images $\boldsymbol{x}$. However, for small motions and linear maps M_g, this still holds approximately, as we show empirically.

of a truck in the query image pair causes our method to wrongly miss the rotational motion.

4.2.4 Evaluation of purely unsupervised feature learning for recognition

Next, we present experiments to test whether useful visual features may be trained in neural networks by minimizing *only* the unsupervised equivariance loss of Eq (4.7), using *no* labeled samples. We follow the approach described in Section 4.1.7.1.

As discussed before, this setting tests the power of our equivariant feature learning objective in isolation, and offers several advantages: (i) It has lower memory and computational requirements, since there is no need for a third stack of layers dedicated to classification (Section 4.1.6, Figure 4.3). (ii) It allows easy off-the-shelf testing of the unsupervised neural network either directly as a feature extractor for new tasks, or as a "pretrained" network to be fine-tuned for new tasks. (iii) It gets rid of the regularization λ of Eq (4.14), thus leaving fewer hyperparameters to optimize. These advantages allow relatively fast experimentation with large neural networks to test our purely supervised pipeline. We therefore perform experiments on large 227×227 images for most of the remainder of this section.

For these experiments, each layer stack follows the standard AlexNet architecture [92] for the layer stacks in these experiments, treating the output of the first fully connected layer, *fc6*, as the feature representation $\mathbf{z}_{\boldsymbol{\theta}}$ (as shown in Figure 4.8). Both the 227×227 image resolution and network architecture allow us to test our method with identical settings against a concurrent and independently proposed approach

for unsupervised representation learning from video+egomotion [3], LSM. We compare the features produced by our method against baselines under two conditions: nearest neighbor classification, and finetuning for classification. In the rest of this subsection, we first present both these settings in Section 4.2.4.1 and Section 4.2.4.2, before discussing their results together in Section 4.2.4.3.

4.2.4.1 Nearest neighbor classifiers with unsupervised features

We first test our unsupervised features for the task of k-nearest neighbor scene recognition on SUN images ("SUN-227" as described in Section 4.2.1). Nearest neighbor tasks are useful to directly analyze the effectiveness of the learned features; such tasks are also used in prior work for unsupervised feature learning [62, 168]. Our nearest neighbor training set has 50 class-labeled training samples per class ($50 \times 397 = 19850$ total training samples), and we set $k = 1$. To evaluate the effect of the equivariance loss on features learned at various layers in the neural network, we perform these nearest neighbor experiments separately on features from *conv3*, *conv4*, *conv5*, and *fc6* layers of the AlexNet architecture used in our experiments.

In addition to the passive slow feature analysis baseline DRLIM, we also compare against the egomotion-based feature learning baseline, LSM, trained with identical settings to our method. We also report the performance when (i) using the pixel space itself as the feature vector ("pixel"), and (ii) using a randomly initialized neural network with identical architecture to ours and baselines ("random weights"). Note that this "random weights" baseline benefits from inductive biases specifically designed and encoded into the architecture of neural networks, such as through con-

volutions and pooling, same as our methods, which should enable it to produce better representations than its input pixel space ("pixel"), even without any training.

4.2.4.2 Finetuning unsupervised network weights for classification

In our second setting for testing the effectiveness of purely unsupervised training with our approach, we finetune the unsupervised network weights for a classification task.

Specifically, we build a new neural network classifier from the unsupervised network by attaching a small neural network "TopNet" with random weights to the layer that is to be evaluated. The architecture for TopNet is Linear(D,500)-ReLU-Linear(500,C)-Softmax Loss, where D is the dimensionality of the output at the layer under evaluation, and $C = 397$ is the number of classes in SUN. We finetune all models on 5 class-labeled training samples per class ($5 \times 397 = 1985$ total training samples). We use identical, standard finetuning settings for all models: learning rate 0.001 and momentum 0.9 with minibatch size 128 for 100 epochs with standard stochastic gradient descent. As before, we test all networks at various layers: *conv3*, *conv4*, *conv5*, and *fc6*. Once again, we compare our methods against DRLIM and LSM.

4.2.4.3 Unsupervised feature evaluation results

Results for the nearest neighbor experiments in Section 4.2.4.1 are shown in Table 4.2, and those for the finetuning experiments are in Table 4.3.

Datasets→	KITTI-227→SUN-227 [397 cls]				
Methods↓/Layers→	conv3	conv4	conv5	fc6	best layer
random	0.25				
pixels	1.80				
random weights	3.66 ± 0.12	3.60 ± 0.06	3.68 ± 0.25	4.04 ± 0.07	4.04 ± 0.07
LSM [3]	≈ 4.9	≈ 4.3	≈ 4.3	-	≈ 4.9
DRLIM [64]	6.44 ± 0.17	6.42 ± 0.23	5.80 ± 0.21	3.46 ± 0.10	6.44 ± 0.17
EQUIV	$\mathbf{7.14 \pm 0.22}$	$\mathbf{7.62 \pm 0.17}$	$\mathbf{6.96 \pm 0.09}$	$\mathbf{4.48 \pm 0.22}$	$\mathbf{7.62 \pm 0.17}$
EQUIV+DRLIM	$\mathbf{7.38 \pm 0.08}$	$\mathbf{7.48 \pm 0.19}$	$\mathbf{6.88 \pm 0.22}$	3.84 ± 0.18	$\mathbf{7.48 \pm 0.19}$
EQUIV+DRLIM (non-discrete)	6.46 ± 0.17	6.16 ± 0.09	6.22 ± 0.20	3.40 ± 0.08	6.46 ± 0.17

Table 4.2: SUN scene recognition accuracies (mean ± standard error) with purely unsupervised feature learning, and nearest neighbor classification (k=1, 50 labeled training images per class). The columns correspond to different layers of the AlexNet architecture. Our EQUIV-based methods once again outperform all baselines. (LSM *fc6* results are not reported in [3], so that entry is left blank. It has only one publicly shared model, so its scores do not have error bars.)

Our method strongly outperforms the baselines in both settings. On nearest neighbor experiments, EQUIV and EQUIV+DRLIM earn best scores of 7.62% and 7.48% compared to the best two baselines, DRLIM and LSM, which score 6.44% and 4.9% respectively. Recall that this is a 397-way classification task, so chance performance is 0.25%. Finetuning experiments yield similar results (Table 4.3), with EQUIV and EQUIV+DRLIM yielding best results of 6.77% and 6.71%, with other methods far behind. These results establish that the equivariance formulation more effectively exploits egomotion than the motion-regression approach of [3] for this data, and succeeds in learning generic image features.[13]

As expected, the pixel space baseline "pixels", and the randomly initialized

[13]For fairness, Table 4.3 uses identical finetuning settings for all models (see Section 4.2.4.2). Compared to its results in Table 4.3, the LSM baseline achieves higher scores on a related experiment in [3], possibly due to differences in finetuning settings and train-test splits.

Datasets→	KITTI-227→SUN-227 [397 cls]				
Methods↓/Layers→	conv3	conv4	conv5	fc6	best layer
random			0.25		
LSM [3]	$\approx 0.26^*$	$\approx 1.04^*$	≈ 3.97	-	≈ 3.97
DRLIM [64]	6.16 ± 0.17	6.12 ± 0.08	5.61 ± 0.05	3.32 ± 0.05	6.16 ± 0.017
EQUIV	$\mathbf{6.77 \pm 0.05}$	$\mathbf{6.75 \pm 0.17}$	$\mathbf{6.77 \pm 0.07}$	$\mathbf{4.22 \pm 0.06}$	$\mathbf{6.77 \pm 0.05}$
EQUIV+DRLIM	$\mathbf{6.70 \pm 0.05}$	$\mathbf{6.71 \pm 0.08}$	6.36 ± 0.05	3.93 ± 0.06	$\mathbf{6.71 \pm 0.08}$
EQUIV+DRLIM (non-discrete)	6.10 ± 0.08	6.00 ± 0.03	5.37 ± 0.07	3.53 ± 0.06	6.10 ± 0.08

Table 4.3: SUN scene recognition accuracies (mean ± standard errors) with purely unsupervised feature learning, followed by finetuning for classification (5 labeled training images per class). The columns correspond to different layers of the AlexNet architecture. Our EQUIV-based methods once again outperform all baselines. (LSM *fc6* results are not reported in [3], so that entry is left blank. It has only one publicly shared model, so its scores do not have error bars.) * denotes models that failed to converge with finetuning.

neural network "random weights" perform significantly worse than all other methods on the nearest neighbor task in Table 4.2. The results also confirm the effect of the inductive bias of the network architecture; the "random weights" baseline builds stronger representations than its input pixel space.

Finally, we also test the non-discrete variant of our approach, EQUIV+DRLIM (non-discrete). Recall from Section 4.1.5, that this approach has an important advantage over EQUIV+DRLIM: it does not discretize the space of motions, so it may be able to more effectively exploit egomotion information. This variant therefore allows us to evaluate whether this advantage is important empirically for this dataset.

As shown in Table 4.2 and Table 4.3, EQUIV+DRLIM (non-discrete) is once again stronger than the baselines. However, it falls short of our standard EQUIV+DRLIM variant. This may be due to (i) the non-linear equivariance map (cf. Section 4.2.1) being too complex and thus weakening the constraint on the learned feature space

(cf. the discussion in Section 4.1.3), (ii) the formulation of Eq (4.13) lacking the *contrastive* property of the loss of Eq (4.7); the feature space is held up from collapsing only by the contrastive term within DRLIM (Eq (4.9)), and (iii) the difficulty of learning a single effective function $\mathbf{M}$ in Eq (4.13), to encode feature transformations corresponding to all possible motions. Given the success of the discretized motion variant of our approach, we focus on it for all subsequent experiments.

Interestingly, representations learned by all methods are most discriminative at *conv3* and *conv4*, and drop off at higher layers, especially *fc6*, where for nearest neighbor classification in Table 4.2, "random weights" performs better or on par with most methods. This suggests that model complexity of the networks may be too high in this setting (reproduced from [3]), given the relatively modest size of the KITTI dataset(≈20,500 video frames). Despite this, EQUIV features perform reasonably well even at *conv5* — in both Table 4.2 and Table 4.3, EQUIV conv5 features are better than all baseline features at any layer, suggesting that our egomotion-equivariance idea exploits unsupervised KITTI data more efficiently than the baselines. In particular, the temporal coherence objective of DRLIM appears to induce a loss of discriminativeness in feature layers close to *fc6*, the layer to which the loss is applied. DRLIM is best at *conv3*, and while EQUIV+DRLIM performs similarly to EQUIV at lower layers, its performance significantly drops at higher layers compared to EQUIV. This is in keeping with our intuitions outlined in Section 1.2; the DRLIM slow feature analysis objective targets invariance, too much of which can lead to a loss of useful information for class discrimination. LSM too shows similar trends, falling off steadily in feature discriminativeness from *conv3* to *conv5*, as seen in Table 4.2.

4.2.5 Equivariance as a regularizer for recognition

Having assessed the effectiveness of training purely with our unsupervised objective in Section 4.2.4, we now test the unsupervised regularization pipeline of Section 4.1.7.2 on three recognition tasks: NORB-NORB, KITTI-KITTI, and KITTI-SUN. The first dataset in each pairing is unsupervised, and the second is supervised. To allow the usage of less complex neural network architectures,[14] reduce computational requirements and enable faster experimentation, all these datasets have smaller images relative to the KITTI-227 and SUN-227 datasets in Section 4.2.4. NORB has 96×96 images while KITTI and SUN are composed of 32×32 images.

Table 4.4 (top, "Regularized") shows the results. On all three datasets, our method significantly improves classification accuracy, not just over the no-prior CLSNET baseline, but also over the closest previous unsupervised feature learning methods.[15]

All the unsupervised feature learning methods yield large gains over CLSNET on all three tasks. However, DRLIM and TEMPORAL are significantly weaker than the proposed method. Those methods are based on the "slow feature analysis" principle [174]—nearby frames must be close to one another in the learned feature space. We observe in practice that temporally close frames are mapped close to each other after only a few training epochs. This points to a possible weakness in these methods—even with parameters (temporal neighborhood size, regularization

[14] We observed in Section 4.2.4 that performance dropped at higher layers, indicating that AlexNet model complexity might be too high.

[15] To verify the CLSNET baseline is legitimate, we also ran a Tiny Image nearest neighbor baseline on SUN as in [177]. It achieves 0.61% accuracy (worse than CLSNET, which achieves 0.70%).

	Datasets→ Methods↓	NORB-NORB [25 cls]	KITTI-KITTI [4 cls]	KITTI-SUN [397 cls, top-1]	KITTI-SUN [397 cls, top-10]
Regularized	TEMPORAL [118]	35.47 ± 0.51	45.12 ± 1.21	1.21 ± 0.14	8.24 ± 0.25
	DRLIM [64]	36.60 ± 0.41	47.04 ± 0.50	1.02 ± 0.12	6.78 ± 0.32
	EQUIV	38.48 ± 0.89	50.64 ± 0.88	1.31 ± 0.07	8.59 ± 0.16
	EQUIV+DRLIM	$\mathbf{40.78 \pm 0.60}$	$\mathbf{50.84 \pm 0.43}$	$\mathbf{1.58 \pm 0.17}$	$\mathbf{9.57 \pm 0.32}$
Unsupervised	random	4.00	25.00	0.25	2.52
	CLSNET	25.11 ± 0.72	41.81 ± 0.38	0.70 ± 0.12	6.10 ± 0.67
	TEMPORAL [118]	42.97 ± 0.62	47.39 ± 0.53	0.79 ± 0.01	-
	DRLIM [64]	42.83 ± 0.76	46.83 ± 0.45	0.86 ± 0.03	-
	EQUIV	42.18 ± 0.32	43.25 ± 1.00	0.56 ± 0.01	-
	EQUIV+DRLIM	$\mathbf{44.08 \pm 0.31}$	$\mathbf{49.59 \pm 0.66}$	$\mathbf{0.87 \pm 0.01}$	-

Table 4.4: Recognition result for three datasets (mean ± standard error) of accuracy % over five repetitions. The last two columns are both results from the KITTI-SUN task, only with different accuracy metrics (top-1 and top-10). Each unsupervised method is tested in two configurations: unsupervised regularization for supervised learning as in Section 4.1.7.2 (top band), and purely unsupervised feature learning as in Section 4.1.7.1 followed by k-nearest neighbor classification (bottom band) with the same labeled training set as the regularization methods. We use small supervised training sets and small k, so it is not straightforward to measure top-10 accuracy with k-nearest neighbor classifiers. We leave those entries blank above.

λ) cross-validated for recognition, the slowness prior is too weak to regularize feature learning effectively, since strengthening it causes loss of discriminative information. In contrast, our method requires *systematic* feature space responses to egomotions, and offers a stronger prior.

The most exciting result is KITTI-SUN (the two rightmost columns in Table 4.4). The KITTI data itself is vastly more challenging than NORB due to its noisy ego-poses from inertial sensors, dynamic scenes with moving traffic, depth variations, occlusions, and objects that enter and exit the scene. Furthermore, the fact we can transfer EQUIV features learned without class labels on KITTI (street scenes from Karlsruhe, road-facing camera with fixed pitch and field of view) to be useful

for a supervised task on the very different domain of SUN ("in the wild" Web images from 397 categories mostly unrelated to streets) indicates the generality of our approach. Note that our method was also validated with larger images in this same KITTI-SUN setting in Section 4.2.4.

Our best recognition accuracy of 1.58% on SUN is achieved with only 6 labeled examples per class. It is ≈30% better than the nearest competing baseline TEMPORAL and over 6 times better than chance. We also compute "Top-10" accuracy (last column) for SUN. This corresponds to the likelihood of the true class being among the top-10 most likely classes according to the classifier. Top-10 accuracy trends closely follow the standard top-1 accuracy result, with EQUIV+DRLIM scoring 9.57% compared to 8.24% for the best baseline, TEMPORAL.

Comparison with purely unsupervised training: Finally, while the broad trends observed above are similar to those observed for the unsupervised training evaluation in Section 4.2.4, individual accuracies are not directly comparable with those reported in Table 4.2 and 4.3, due to the differences in image sizes and network architectures. To address this, we now repeat the experiments of Section 4.2.4.1 for these new datasets, performing nearest neighbor classification with purely unsupervised features. Networks are trained with purely unsupervised losses. Features are then extracted and used in a nearest neighbor classifier ($k = 1$) using the target task training set (6, 4, and 6 labeled training images per class respectively for NORB, KITTI, and SUN).

The results of these experiments, shown in Table 4.4 (bottom, "Unsuper-

vised"), allow us to observe several trends. In general, we should expect that regularization yields higher accuracies than purely unsupervised feature learning followed by nearest neighbors — the regularization setting allows target domain and target task knowledge to influence the learning of the features themselves. From Table 4.4, KITTI-SUN and KITTI-KITTI both follow these expected trends. Of these, KITTI-SUN results in particular are consistently significantly poorer with purely unsupervised training, compared to regularization. We believe this is due to the large domain differences between KITTI and SUN, which mean that networks produced by purely unsupervised training on KITTI may be less well-suited to processing SUN image inputs. This is supported by the fact that accuracies on KITTI-KITTI, where only the target dataset is changed and domains are matched, are only marginally better with regularization than with unsupervised training.

NORB-NORB accuracies are an exception in which purely unsupervised training consistently performs slightly better than regularization. We believe that the toy neural network architecture employed in our NORB experiments (see Section 4.2.1 and Figure 4.8) could prevent effective training, allowing a simple nearest neighbor classifier to be more effective. Finally, on both the "regularized" and "unsupervised" accuracies in Table 4.4, EQUIV+DRLIM features improve significantly over EQUIV. This trend is thus consistent among experiments with small image datasets, but not observed in our experiments with larger images in Section 4.2.4. This suggests that the performance of EQUIV+DRLIM may especially depend significantly on network architectures and/or feature dimensionality.

Varying training set sizes: We are especially interested in the impact of our feature learning idea when supervised training sets are relatively small. This corresponds to handling categorization problems in the "long tail", where training samples are scarce and priors are most useful. However, we do continue to see impact by our approach for larger training sets. For example, with N=20 samples for each of 397 classes on KITTI-SUN (7,940 total labeled training images), EQUIV scores 3.66+/-0.08% accuracy vs. 1.66+/-0.18 for CLSNET. Thus, our equivariance prior continues to boost recognition accuracy even at larger training set sizes.

Later in this thesis, in Section 5.2.3.3, we compare the egomotion-equivariance approach of this chapter directly against the slow and steady features of Chapter 3.

4.2.6 Next-best view selection for recognition

Finally, we show the results of a direct application of equivariant features to "next-best view selection" on NORB, as described in Section 4.1.7.3 and illustrated in Figure 4.6. Given one view of a NORB toy, the task is to tell a hypothetical robot how to move next, in order to best recognize the object, i.e, which neighboring view would best reduce object instance label prediction uncertainty.

To use the approach of Section 4.1.7.3 for the baselines, we first compute equivariance maps M'_g for all methods as described in Section 4.2.2. We set $k = 25$ for computing k-nearest neighbors, as per Section 4.1.7.3.

Table 4.5 shows the results. On this task too, EQUIV features easily outperform the baselines. Recall that our approach for this task is based on exploiting the predictability of feature responses to observer motions, i.e., feature equivariance.

Methods↓	1-view→ 2-view	accuracy gain
random	4.00 → 4.00	0
TEMPORAL [118]	29.60→ 31.90	2.30
DRLIM [64]	14.89→ 17.95	3.06
EQUIV	**38.52→43.86**	**5.34**
EQUIV+DRLIM	**38.46→43.18**	**4.72**

Table 4.5: Next-best view selection accuracy % on NORB. Our method EQUIV (and augmented with slowness in EQUIV+DRLIM) clearly outperforms all baselines.

This result thus highlights the potential for many such novel applications of our equivariant feature-learning formulation. Indeed, we use these insights in developing a more sophisticated active recognition system in Chapter 6.

4.2.7 Comparison against more recent methods

Finally, there has been much other work on unsupervised feature learning since the time this work was completed. We now present some results comparing our features against more recent work, and discuss the findings. Note also that while our baselines until now have been restricted to using the same dataset as ours for unsupervised learning, we now compare against methods trained on arbitrary datasets.

My co-authors and I evaluated our egomotion-equivariance approach trained on KITTI 227×227 images against other more recent methods in [48]. We evaluated all features on PASCAL VOC 2012 multi-label classification after finetuning. The results are shown in Table 4.6. Our method achieves an average precision score

Methods↓	Supervision	PASCAL VOC 2012
ImageNet	1.2M labeled images	73.9%
Wang et al [168]	4M visual tracks	47.4%
Agrawal et al [3]	20K image egomotion	40.2%
Pathak et al [127]	1M image spatial context	41.4%
Gao et al [48]	1M region proposal pairs	44.1%
Ours	20K image egomotion	40.7%

Table 4.6: PASCAL VOC 2012 multi-label classification mean average precision scores for our method versus various more recent unsupervised feature learning methods, and ImageNet-pretrained features.

of 40.7%. This is roughly on par with learning features by inpainting [127] which scores 41.4%, and learning features by predicting motion [3] which scores 40.2%. A method for learning from temporal continuity within precomputed visual tracks of objects [168] performs best, scoring 47.4%. These approaches however all fall significantly short of supervised ImageNet-pretrained AlexNet features, which score 73.9%.

We believe that a major limiting factor for the performance of our approach is the availability of appropriate unlabeled data. While camera egomotion metadata accompanying video is in principle free to acquire for an embodied agent, off-the-shelf video data with egomotion labels is scarce in practice, and this is a limiting factor in the practical applicability of egomotion-based learning. For instance, our KITTI models are trained with fewer than 20,000 images overall (which are also highly correlated since they are video frames). In contrast, the best performing method of [168] in the above evaluation was trained on 4 million visual tracks from video.

Finally, we cite a result from recent work that attempts to analyze the con-

cepts learned within various neural network models. Recently, the semantic concepts detected within various models were visualized in [12] and categorized into "object", "part", "scene", "material", "texture", and "color" concept detectors. They found that our egomotion-based model learned mainly texture, scene, object, and color detectors. This is largely similar to other unsupervised approaches evaluated in the paper. The tracking-based method of [168] once again performs qualitatively best in this evaluation.

4.3 Conclusion

Over the last decade, visual recognition methods have focused almost exclusively on learning from "bags of images". We argue that such "disembodied" image collections, though clearly valuable when collected at scale, deprive feature learning methods from the informative physical context of the original visual experience. We presented the first "embodied" approach to feature learning that generates features equivariant to egomotion. Our results on multiple datasets and on multiple tasks show that our approach successfully learns equivariant features, which are beneficial for many downstream tasks and hold great promise for novel future applications.

Our results are promising, but there is an important caveat currently limiting the ability to scale up our approach to be practically useful. As pointed out in Section 4.2.7, while egomotion-labeled data is in principle free for an agent exploring the world, off-the-shelf video data with egomotion labels is scarce in practice, and this is a limiting factor in the practical applicability of egomotion-based learning. I have considered but not explored alternative sources of data such as Google Streetview and

similar services from which large-scale datasets of egomotion-labeled image pairs may be mined, as in [187]. Such sources share many of the limitations of KITTI (limited to outdoor, day-time images from a car-mounted camera on a road), but it would be interesting to empirically investigate how well our method can exploit a much larger dataset collected from all around the world. An even more interesting source of data could be human-worn egocentric camera recordings with accompanying camera motion data. Such a dataset would exhibit greater variation in content together with more degrees of freedom for the camera egomotion, which might strengthen the representations learned by our approach.

There are other potential directions for follow-up work. Our experiments showed that the clustering-based discrete approach was more effective than the non-discrete variant. However, at a high level, these methods may be seen to lie at two ends of a spectrum over which the granularity of clusters may be varied continuously. It would be interesting to study if there is an optimal granularity. It would be interesting to probe other points in this spectrum.

Another drawback stems from the fact that appearance changes of an image in response to an observer's ego-motion are not determined by the current view and the egomotion alone. Several latent factors are involved, such as the depth map of the observed scene, dynamic moving objects within or even outside of the currently observed view, occlusions etc. One possibility for modeling these ambiguities is to allow the system to predict multiple alternative futures and penalize the error of only the most correct prediction for each sample.

Finally, to see a high-level shortcoming of this work, recall the two advan-

tages of the active kitten from Chapter 1, which proved to be key to its perceptual development: (1) proprioceptive knowledge, and (2) ability to *select* motions during development. While the work in this chapter captures the first of these advantages by substituting known camera motion for the active kitten's proprioception, it currently relies on pre-recorded video that is captured without the system's own direction. The second advantage is explored later within this dissertation, in Chapters 6 and 7.

Before that though, we take one intermediate step, asking: what if we were able to access arbitrary viewpoints within visual environments during training time, rather than having to rely on precaptured video? In Chapter 5, rather than concern ourselves with how to select motions to access those viewpoints, we will instead assume all possible viewpoints are available at training time.

Chapter 5

Unsupervised learning through one-shot image-based shape reconstruction

[1]In contrast to the setting of the last chapter where our egomotion equivariance approach exploited precaptured off-the-shelf video, an embodied learning agent has the option of exhaustively examining its visual environment during learning. This means that rather than having to rely on incidentally useful information such as a small number of overlapping views of a 3D object for learning, the agent could learn through *full* observation. For example rather than watch an object rotate in one direction, an agent might turn it around in all directions with its manipulators during training. In this chapter, I will describe an approach to effectively exploit such complete observations of 3D objects to learning good 2D image representations, which I introduced earlier in Section 1.3.

The work in this chapter may be seen as a logical extension of the preceding chapters as follows. In Chapters 3 and 4, we have been exploring the learning of visual representations from video. Whereas previous "slow feature analysis" methods for learning from video were restricted to learning invariances from the slowness of transitions between a pair of consecutive frames, Chapter 3 generalizes this notion to

[1]The work in this chapter was supervised by Prof. Kristen Grauman.

learn higher order "steadiness" dynamics by exploiting longer sequences of frames. Then, in Chapter 4, we showed how to exploit pairs of image observations related by known observer motions. In this chapter, we take one further step to ask: what if, rather than having access to only *pairs* of observations related by known agent motions as in Chapter 4, the agent could instead learn by *fully* inspecting objects from a grid of viewpoints? In this sense, this work may be thought of as extending the equivariance idea of Chapter 4 analogously to how the steady feature analysis notion developed in Chapter 3 extends prior slow feature analysis approaches—richer information may be mined from jointly exploiting known relationships among collections of images if we do not restrict those collections to only have two elements.

More specifically, in this component of my thesis, I explore the setting where, rather than one video in each training environment from which rare, useful view pairs must be mined in a preprocessing stage, the agent instead has the ability to acquire a full "viewgrid" consisting of all possibly useful views, related by all possible egomotions.

To recap the motivating ideas for this component from Section 1.3, while visual perception relies largely on flat 2D observations, the visual world and objects in it are inherently three dimensional. Inferring 3D geometry from 2D views is therefore basic to visual perception, as confirmed by evidence from cognitive psychology [146]. Inspired by this, we ask the question: can we learn image representations encoding geometry from 2D views, for use in computer vision systems? We propose to do this by training a system for the "mental rotation" or single view image-based shape reconstruction task. Given one 2D view of an object from an arbitrary viewpoint,

the system must output views corresponding to arbitrary rotations from the current viewpoint. This task is illustrated in Figure 5.1.

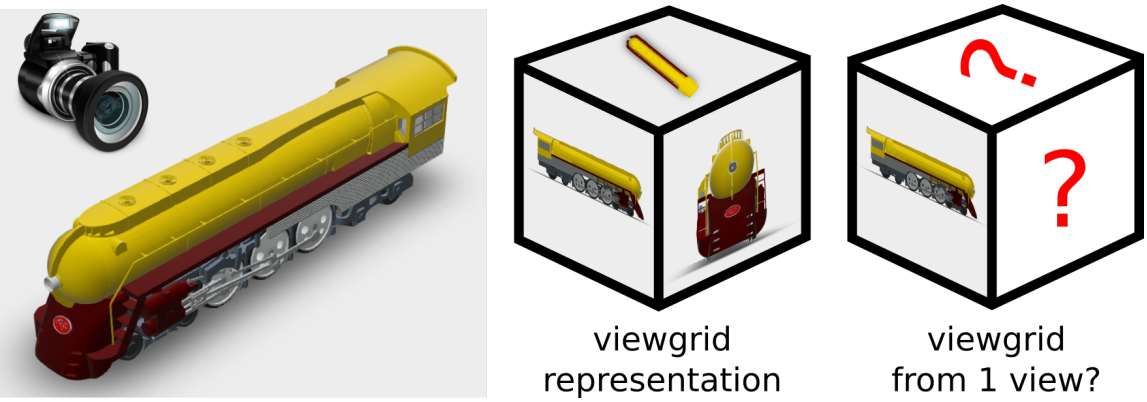


Figure 5.1: One-shot image-based shape reconstruction: A 3D shape is represented by its "viewgrid" — views from evenly spaced viewpoints. Given one 2D view of an arbitrary shape, we train a deep network to produce the remaining views in the viewgrid and ask: does this training induce transferable representations in the network?

With an infinite number of views from all around an object, classic geometric methods allow full 3D shape recovery [67] with little or no object understanding, as they are completely agnostic to the semantics of the object, and work for arbitrary shapes that might not even represent real-world objects. With real world objects however, 3D understanding is possible from much sparser observations by using cues such as semantics and shading. This suggests the use of learning-based approaches for reconstruction.

As reviewed in Section 1.3 and in more detail in Section 2.9, other vision researchers have also recently begun to investigate the use of deep learning methods for single view 3D reconstruction [31, 43, 56, 84, 135, 159, 175, 183, 193]. Our convolutional encoder-decoder [112] neural network architecture is similar to [159,

183, 193]. However, while we view reconstruction as a path to learning good image representations, these works differ from us in viewing reconstruction as an end task in itself. While we train one model to reconstruct all categories, prior approaches instead build category-specific models for some common categories like chairs and cars. Our main hypothesis is that once we have induced a latent representation within the model for "lifting" 2D views of generic objects of inferred 3D shapes, that representation then possesses useful properties for generic high-level vision tasks such as object category recognition.

The rest of this chapter will proceed as follows: Section 5.1 describes the technical details of our neural network-based system for one-shot image-based shape reconstruction. Section 5.2 then presents our key results validating that (i) our system successfully learns the training task of category-agnostic image-based shape reconstruction, generalizing even to categories that were not seen in the training set, and (ii) the representations learned in the process are generically useful, in particular transferring well to the semantic tasks of object recognition and retrieval.

5.1 Approach

Our goal is to train a system to recover a full image-based shape reconstruction of an object after observing one view of it from an arbitrary viewpoint. This task of "mentally rotating" the object from its observed viewpoint to arbitrary relative poses requires developing 3D understanding from single 2D views, which is crucial for many vision tasks. Through training on this image-based shape reconstruction task,

we want to eventually learn generically useful visual representations that embed this 3D understanding and apply those representations to other visual tasks.

5.1.1 Problem setup

During training, we first evenly sample views from the viewing sphere around each object. To do this, we begin by selecting a set S_{az} of M camera azimuths $S_{az} = \{360°/M, 720°/M, \ldots 360°\}$ centered around the object. Then we select a set S_{el} of N camera elevations $S_{el} = \{0°, \pm 180°/(N-1), \pm 360°/(N-1), \cdots \pm 90°\}$. We now sample all $M \times N$ views of every object corresponding to the cartesian product $S = S_{az} \times S_{el}$ of azimuth and elevation positions: $\{\boldsymbol{y}(\boldsymbol{\theta_i}) : \boldsymbol{\theta_i} \in S\}$.[2] Note that each $\boldsymbol{\theta_i}$ is an elevation-azimuth pair, and represents one position in the viewing grid S.

Now, with these evenly sampled views, the one-shot image-based shape reconstruction task can be formulated as follows. Suppose the observed view is at an unknown camera position $\boldsymbol{\theta}$ sampled from our viewing grid set S of camera positions. The system must learn to predict the views $\boldsymbol{y}(\boldsymbol{\theta'})$ at position $\boldsymbol{\theta'} = \boldsymbol{\theta} + \boldsymbol{\delta_i}$ for all $\boldsymbol{\delta_i} \in S$. Because of the even sampling over the full viewing sphere, $\boldsymbol{\theta'}$ is itself also in our original viewpoint set S, so we have already acquired supervision for all views that our system must learn to predict.

Our training phase may be thought of as representing a setting where a robotic visual agent learns visual perception from scratch by inspecting a large number of objects. It can move its camera to arbitrary viewpoints around each object as it does

[2] omitting object indices throughout to simplify notation.

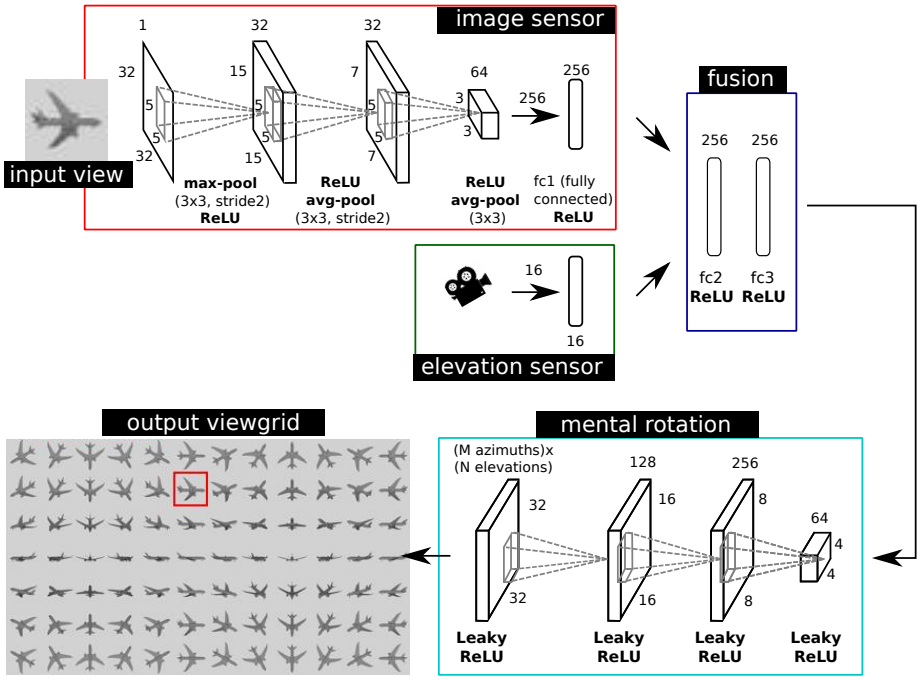


Figure 5.2: Architecture of our system. A single view of an object (top left) and the corresponding camera elevation (bottom left) are processed independently in "image sensor" and "elevation sensor" neural net modules, before fusion in a "fusion module". The output code, which embeds the 3D shape of the object aligned to the observed view, is now processed in a deconvolutional "mental rotation" decoder. The final output is a sequence of images representing systematically shifted viewpoints relative to the observed view.

so, to acquire its own supervision. At test time, as a first goal, it must be able to observe only one view and hallucinate the effects of all camera displacements from that view. Eventually, the goal is to transfer knowledge acquired from training on this unsupervised reconstruction task to recognition tasks.

5.1.2 Shape reconstruction system architecture

To tackle this one-shot shape reconstruction task, we model the system as a deep feed-forward neural network. Our network architecture naturally splits into four

modular sub-networks with different functions: an elevation sensor module, an image sensor module, a fusion module, and finally, a mental rotation module. Together, the elevation sensor, image sensor, and fusion modules process the observation and proprioceptive camera elevation information to produce a single feature vector that encodes the full object model. The mental rotation module then acts as a decoder — it processes this code through a series of learned deconvolutions to produce the desired image-based shape reconstruction at its output. Figure 5.2 presents this architecture in detail.

Encoder: First, the image sensor module embeds the observed view through a series of convolutions and a fully connected layer into a vector. In parallel, information about the camera elevation angle is processed through the elevation sensor module.Note that the object pose is not fully known — while camera elevation can be determined from gravity cues, there is no way to determine the azimuth.

The outputs of the image and elevation sensor modules are concatenated and passed through a fusion module which jointly processes the information from the image and elevation sensors to produce a $D = 256$-dimensional output "code" vector, which embeds knowledge of object shape. To sum up, the function of the encoder is to "lift" a 2D view to a single vector representation of the full 3D object shape.

Decoder: The output of the fusion module is now processed through another fully connected layer to increase its dimensionality before reshaping into a sequence of

small 4×4 feature maps. These maps are then iteratively upsampled through a series of learned deconvolutional layers. The final output of the mental rotation module is a sequence of MN output maps $\{\hat{\boldsymbol{y}}_{\boldsymbol{i}} : i = 1, \ldots M \times N\}$ of same height and width as the input image. Together these MN maps represent the system's output viewgrid.

The complete architecture, together with more detailed specifications, is visualized in Figure 5.2. Our convolutional encoder-decoder [112] neural network architecture is similar to [159, 183, 193]. The primary focus of our work is, however, very different. We view one-shot reconstruction as a path to good image representations that lift 2D views to 3D.

There is one final detail of the pipeline to be dealt with — what is the correspondence between the individual views in the target viewgrid and the system's output maps? Recall that at test time, the system will be presented with a single view of a novel object, from an unknown viewpoint (again elevation known, azimuth unknown). How then can it know the correct viewpoint coordinates for the viewgrid it must produce? It instead produces viewgrids aligned with the *observed* viewpoint at the azimuthal coordinate origin. Azimuthal rotations of a given viewgrid all form an equivalence class. In other words, circularly shifting the 7×12 viewgrid in Fig 1 by one column will produce a different, but entirely valid viewgrid representation of the same airplane object.

To optimize the entire pipeline, we regress to the target viewgrid $\boldsymbol{y}$, which is available for each training object. Since our output viewgrids are aligned by the observed view, we must accordingly shift the target viewgrid before performing

regression. This leads to the following minimization objective:

$$\mathcal{L} = \sum_{i=1}^{M \times N} \|\hat{\boldsymbol{y}}_{\boldsymbol{i}} - \boldsymbol{y}(\boldsymbol{\theta} + \boldsymbol{\delta}_{\boldsymbol{i}})\|^2, \tag{5.1}$$

where we omit the summation over the training set to keep the notation simple. Each output map $\hat{\boldsymbol{y}}_{\boldsymbol{i}}$ is thus penalized for deviation from a specific *relative* shift from the observed viewpoint $\boldsymbol{\theta}$.[3]

At test time, the full image-based shape representation is recovered in one shot, i.e., in a single forward pass through the neural network. This is different from iterative mental rotation by learning to directly hallucinate only some elemental rotations, and then composing them together, as in [31]. Directly producing the full viewgrid has several advantages. In particular, it is efficient, and it avoids "drift" degradation from iterated forward passes. Most critically to our end goal of unsupervised feature learning, this one-shot reconstruction task enforces that the encoder must capture the *full* 3D object shape from observing just one 2D view.

At this point, it is appropriate to step back to once again observe connection between this one-shot reconstruction approach and the egomotion-equivariance approach of Chapter 4. In both cases, we exploit known camera pose relationships, i.e., egomotions among images for unsupervised image representation learning. As pointed out earlier, while Chapter 4 is restricted to *pairs* of views from precaptured unlabeled video, the one-shot reconstruction approach of this chapter instead exploits relationships among *all* views of an object jointly.

[3]Since our system has access to elevation, we zero out the elevation coordinate of $\boldsymbol{\theta}$, i.e., the system is trained to produce viewgrids that are perfectly aligned with the target viewgrid in elevation.

Finally, aside from high-level motivational connections, the two approaches are also *technically* related. Egomotion-equivariance targets view prediction in the latent feature space as a proxy task for feature learning. The one-shot reconstruction task of this chapter also involves view prediction, but in the pixel space, and of all unobserved views simultaneously. Keeping the prediction task in the pixel space has the technical advantage of not relying on contrastive formulations using negative samples, as was necessary in Chapter 4 (and Chapter 3 before that).

5.1.3 Optimization

Models are initialized with parameters drawn from a uniform distribution between -0.1 and +0.1. The mean squared error loss is then optimized through standard minibatch stochastic gradient descent (batch size 32, momentum 0.9) via backpropagation. For our method and all baselines, we optimize the learning rate hyperparameter on validation data. Training terminates when the loss on the validation set begins to rise.

5.1.4 Unsupervised features for recognition

While we have thus far described our formulation for training a deep neural network for image-based shape reconstruction from single views, our hypothesis is that representations trained in this manner will facilitate high-level visual recognition tasks. This is motivated by the fact that in order to solve the reconstruction task effectively, the network must implicitly learn to "lift" 2D views of objects to inferred 3D shapes. A full 3D shape representation has many attractive properties for generic

visual tasks. For instance, pose invariance is desirable for recognition, but is a hard problem in 2D views. This becomes trivial in a 3D representation, since different poses correspond to simple transformations in the 3D space.

Suppose that the visual agent has learned a model as above for image-based shape reconstruction by inspecting 3D shapes. If it is now presented with new dataset of class-labeled training images for a categorization task, we may transfer the 3D knowledge acquired in the one-view reconstruction task to this new task.

Specifically, for each new class-labeled image, we directly represent it in the feature space by an intermediate layer in our deep neural network trained for reconstruction. These features are then input to a generic machine learning pipeline, such as a k-nearest neighbor classifier, that is to be trained for the categorization task.

Recall that the output of the fusion module in Figure 5.2, which is the fc3 feature vector, is trained to encode 3D shape. In our experiments, we test the usefulness of features from fc3 and its two immediate preceding layers, fc2, and fc1, for solving object classification and retrieval tasks.

5.2 Experiments

We evaluate our approach for two tasks. First, we assess its performance on the image-based shape completion task that it is trained on. Next, we evaluate the strength of the features learned within the model for semantic recognition tasks.

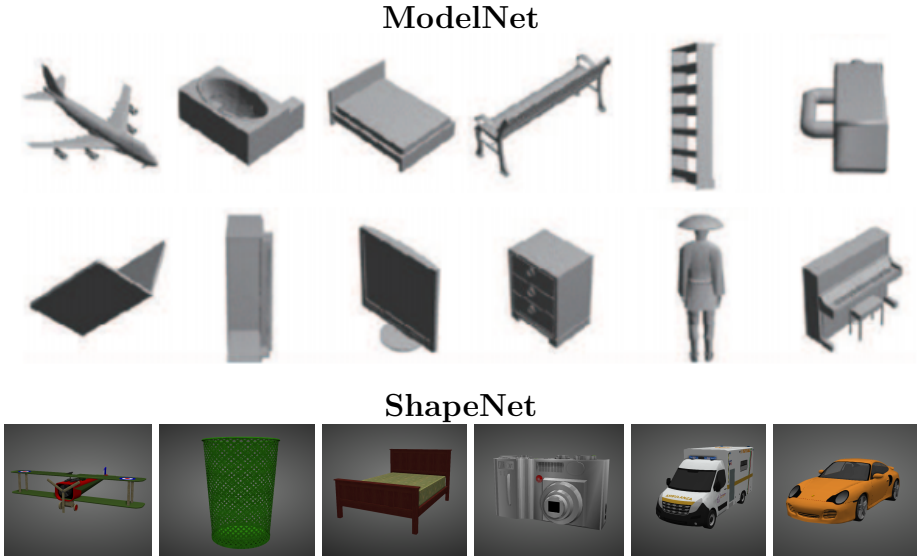


Figure 5.3: Examples of 3D object models from ModelNet (top) and ShapeNet (bottom), the two datasets we use in our experiments. Note: we use grayscale renderings for both datasets.

5.2.1 Experimental setup

We test our method on two publicly available datasets: ModelNet [176] and ShapeNet [28]. Both of these datasets provide large numbers of manually generated 3D models, with class labels. For each object model, we render 32×32 grayscale views from a grid of viewpoints that is evenly sampled over the viewing sphere centered on the object. Example object models from the two datasets are shown in Figure 5.3.

ModelNet [176] has 3D CAD models that are downloaded from the Web, and then manually aligned and categorized. ModelNet comes with two standard subsets: ModelNet-10 and ModelNet-40, with 10 and 40 object classes respectively. The 40 classes in ModelNet-40 include the 10 classes in ModelNet-10. We use the 10 ModelNet-10 classes as our unseen classes, and the other 30 ModelNet-40 classes

Dataset→	ModelNet		ShapeNet	
Methods↓/ Data→	seen classes	unseen classes	seen classes	unseen classes
Camera elevations	0,±30°,±60°,±90°		0,±30°,±60°	
Camera azimuths	0,30°,60°,...,330°		0,45°,±315°	
View size	32×32		32×32	
Categories	30	10	30	25
Training models	5,852	-	11,532	-
Validation models	312	-	1,681	-
Testing models	1,248	726	3,569	641

Table 5.1: Dataset statistics

as seen classes. We use the standard train-test split, and set aside 20% of seen class test set models as validation data. Table 5.1 shows more details.

ShapeNet [28] contains a large number of models organized into semantic categories under the WordNet taxonomy. All models are consistently aligned to fixed canonical viewpoints. We use the standard ShapeNetCore-v2 subset which contains models from 55 diverse categories. Of these, we select the 30 largest categories as seen categories, and the remaining 25 categories are unseen. We use the standard train-test split. Further, since different categories have highly varying numbers of object instances, we limit each category in our seen class training set to 500 models at most, to prevent the training from being dominated by models of a small number of very common categories. Table 5.1 shows more details.

5.2.2 Image-based shape reconstruction

First, we train and test our method on the one-shot image-based shape recon-

struction task. For both datasets, the system is trained on the seen classes training set. To set the learning rate and determine termination criteria during deep network training, we employ the seen classes validation set. The trained model is subsequently tested on both seen and unseen class test sets.

To quantitatively evaluate our method, we measure the per-pixel mean squared deviation of the viewgrid produced by our method from the ground truth viewgrid. We compare our method on both seen and unseen classes, against several baselines:

- **Avg view:** This baseline simply predicts, at each viewpoint in the viewgrid, the average of all views observed in the training set *over all viewpoints*.
- **Avg viewgrid:** Both ModelNet and ShapeNet have consistently aligned models, so there are significant biases that can be exploited by a method that has access to this canonical alignment information. This baseline aims to exploit this bias by predicting, at each viewpoint in the viewgrid, the average of all views observed in the training set *at that viewpoint*. Note that our system does not actually have access to this alignment information, so it cannot exploit this bias.
- **GT category avg view:** This baseline represents a model that has perfectly learned classification. Given an arbitrary object from some ground truth category, this baseline predicts, at each viewpoint, the average of all views observed in the training set for that category.
- **GT category avg viewgrid:** This baseline is the same as GT category avg view, but has knowledge of canonical alignments too, so it produces the average

of views observed at each viewpoint over all models in that category in the training set.

- **Ours w. canonical alignment:** Our method does not see canonical viewgrid alignments during training, since it is trained to produce views at various *relative* displacements from the observed view, i.e., its target is a viewgrid that has the current view at its origin. This is realistic, since such knowledge is unlikely to be available to an agent observing objects in the real world. However, for the sake of evaluation, we also evaluate a baseline that has access to canonical viewpoints. Concretely, at training time, this baseline is trained to produce a canonically aligned viewgrid from observing one view. Rather than optimizing the objective in Eq (5.1), this baseline optimizes:

$$\mathcal{L} = \sum_{i=1}^{M \times N} (\hat{\boldsymbol{y}}_{\boldsymbol{i}} - \boldsymbol{y}(\boldsymbol{\delta}_{\boldsymbol{i}}))^2, \tag{5.2}$$

  so that each output map $\hat{\boldsymbol{y}}_{\boldsymbol{i}}$ of the system is now assigned to a specific coordinate in the canonical viewgrid axes, rather than a specific relative shift from the observed viewpoint $\boldsymbol{\theta}$, as in Eq (5.1).

Table 5.2 shows the mean squared error results. "Avg viewgrid" and "GT category avg viewgrid" improve by large margins over "Avg view" and "GT category avg view" respectively. This shows that viewgrid alignment biases can be useful for reconstruction in ModelNet and ShapeNet.

Despite this, our approach not only outperforms these methods by large margins but even outperforms its variant "Ours w. canonical alignment" that does use

Dataset→	ModelNet		ShapeNet	
Methods↓/ Data→	seen classes	unseen classes	seen classes	unseen classes
	MSE×1000	MSE×1000	MSE×1000	MSE×1000
Avg view	13.514	15.956	14.793	16.394
Avg viewgrid	12.954	15.725	14.334	15.942
GT category avg view	11.006	-	12.279	-
GT category avg viewgrid	8.891	-	9.374	-
Ours w. canonical alignment	4.689	10.440	5.879	9.021
Ours	**3.718**	**7.005**	**4.656**	**6.811**

Table 5.2: Quantitative results for one-shot image-based shape completion. Results are reported in MSE on images normalized to lie in $[0, 1]$. Lower is better. Per-category results are shown in Figure 5.4 and Figure 5.5. "GT category" methods do not work on unseen classes since they rely on knowledge of the category, so those entries are left blank.

alignment biases in the data. The margin is especially large on unseen class subsets of both ModelNet and ShapeNet, where using alignment biases of seen class categories strongly hurts the performance of "Ours w. canonical alignment". Why is "Ours w. canonical alignment" weaker? Recall that it is trained to produce canonically aligned viewgrids from observing one view, but like our method, it also does not have access to the absolute azimuth of the observed view. If "Ours w. canonical alignment" were learning to do something simple like produce the average viewgrid for an inferred category (similar to "GT category avg viewgrid"), then targeting canonically aligned viewgrids would be a clear advantage.[4] But to exploit instance-specific details from the observed view effectively, it must perform the additional step of inferring the position of the observed view in the canonical viewgrid, which can be difficult. Even if it infers shape correctly, it may struggle to produce correct canonically aligned viewgrids. It is therefore easier and more natural to represent

[4]assuming seen categories. For unseen categories, such a strategy is infeasible.

shape with respect to the observed viewpoint, as in "Ours".

Figure 5.4 and 5.5 show errors for each seen and unseen category in sorted order. While the distribution of errors among unseen classes is shifted towards higher errors from the seen classes, there is a significant overlap, i.e., many unseen classes have lower reconstruction errors than many seen classes, indicating significant generalization from seen to unseen classes.

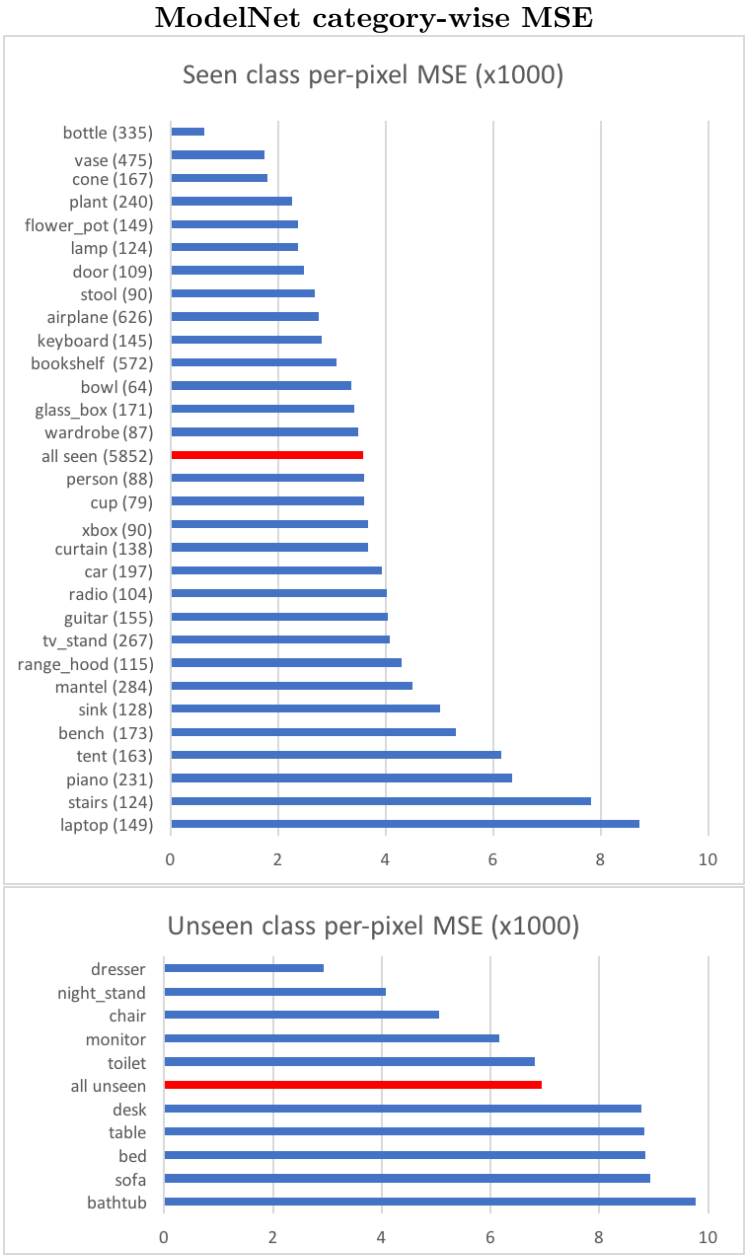


Figure 5.4: ModelNet seen (top) and unseen (bottom) class reconstruction performance in MSE per class (lower is better). Seen classes are not evenly represented in the training set, so the number of training samples per class is indicated in parentheses next to the corresponding class name.

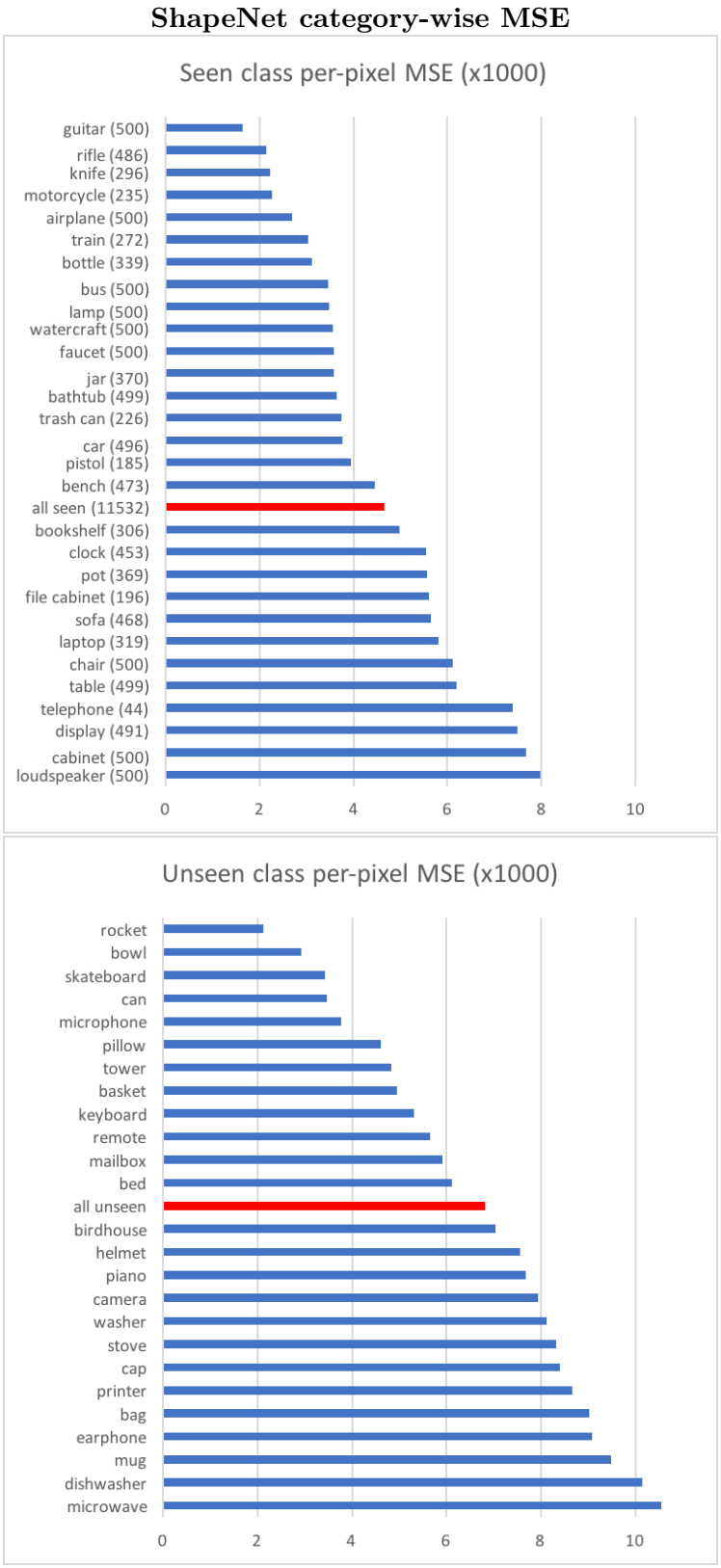


Figure 5.5: ShapeNet seen (top) and unseen (bottom) class reconstruction performance in MSE per class (lower is better). Seen classes are not evenly represented in the training set, so number of training samples per class are indicated in parentheses next to the class names.

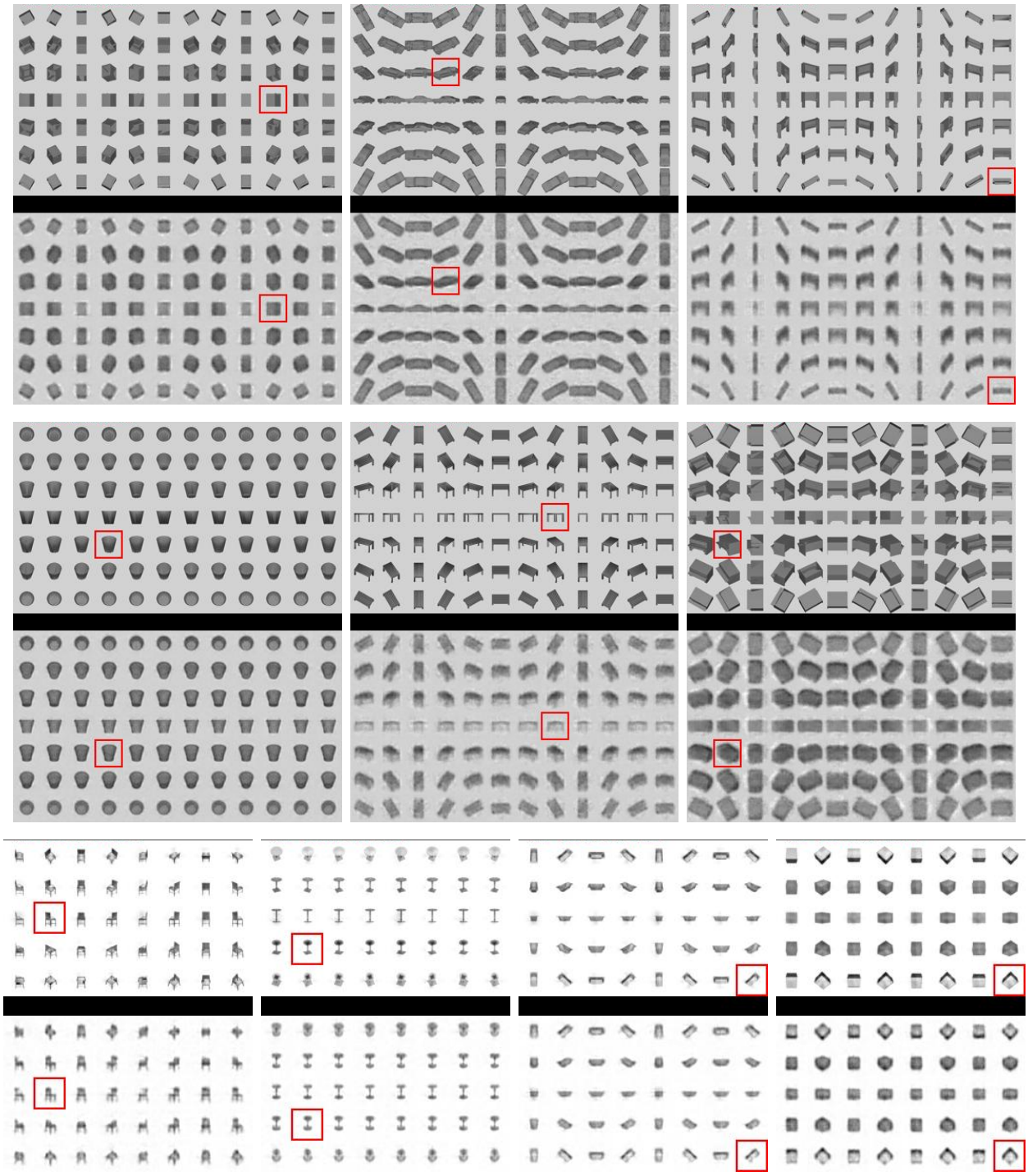


Figure 5.6: One-shot image-based shape reconstructions from single view. In each panel, ground truth viewgrids are shown at the top, the observed view is marked with a red box, and our method's reconstructions are shown at the bottom. Top two rows show ModelNet results, and bottom row shows ShapeNet results. (Best seen in pdf at high resolution.) In (row 1, panel 1) from ModelNet and (row 3, panel 4) from ShapeNet, our method exhibits the ability to infer shape from shading cues for simple objects. In (row 1, panel 3), the system manages to reconstruct an object shape from what appears to be an impossible viewpoint to fully infer the shape, indicating that it effectively exploits the semantic structure in ModelNet to make educated guesses. In (row 2, panel 2), the system observes a very ambiguous viewpoint that could be any one of four different views at the same azimuth. In response to this ambiguity, it attempts to play it safe to minimize MSE loss by averaging over possible outcomes, producing blurry views.

Figure 5.6 shows some example viewgrids generated by our method. In (row 1, panel 1) from ModelNet and (row 3, panel 4) from ShapeNet, our method exhibits the ability to infer shape from shading cues for simple objects. In (row 1, panel 3), the system manages to reconstruct an object shape from what appears to be an impossible viewpoint to fully infer the shape, indicating that it effectively exploits the semantic structure in ModelNet to make educated guesses. In (row 2, panel 2), the system observes a very ambiguous viewpoint that could be any one of four different views at the same azimuth. In response to this ambiguity, it attempts to play it safe to minimize MSE loss by averaging over possible outcomes, producing blurry views.

Overall, these results establish that our approach successfully learns a unified category-agnostic one-shot shape reconstruction model that handles not only a large number of generic categories that are represented in its training set, but even completely novel unseen categories.

Influence of observed view position on viewgrid reconstruction error

We now break down the errors summarized in Table 5.2 on the basis of the dependence of those errors on the observed view. Specifically, in both ModelNet and ShapeNet, models are manually aligned to some canonical starting positions (which are deliberately withheld from our approach). This means that we can meaningfully attempt to understand which positions in the viewgrid, when observed, led to higher or lower reconstruction errors for the full viewgrid for each category. In other words, which views are more or less informative to our one-shot image-based shape

reconstruction approach?

Figure 5.7 show results for a few ModelNet seen classes. For each class, a heatmap of mean-square errors is overlaid over the average viewgrid for that class. The first map corresponds to all 30 ModelNet seen classes, and the accompanying colorbar illustrates how lower MSEs correspond to darker, colder, blue colors, and higher MSEs to lighter, warmer, yellow colors.

Studying these heatmaps reveals some intuitive and interesting trends. Across all datasets, perfectly aligned viewing positions from where only a single or a small number of faces of an object are visible have yellowish, lighter highlights indicating high reconstruction errors conditioned on those views, i.e., lower informativeness. See for instance, the "bench" heatmap in Figure 5.7, which has characteristic yellowish horizontal and vertical stripes running across the viewgrid corresponding to elevations and azimuths that only reveal a small number of faces of the object. Top and bottom views, sampled at -90°and +90°in ModelNet, are consistently uninformative, since very different shapes can have very similar overhead projections. Shapes that have narrow linear projections along some directions tend to present very little information from the corresponding views. For example, see the horizontal view of the airplane and the keyboard (middle row in the corresponding viewgrids, corresponding to zero azimuth) in Figure 5.7.

Overall, these trends largely agree with our intuitive notions of which views are most informative for 3D understanding, and serve as evidence that our method learns to perform reconstruction by observing meaningful and appropriate cues.

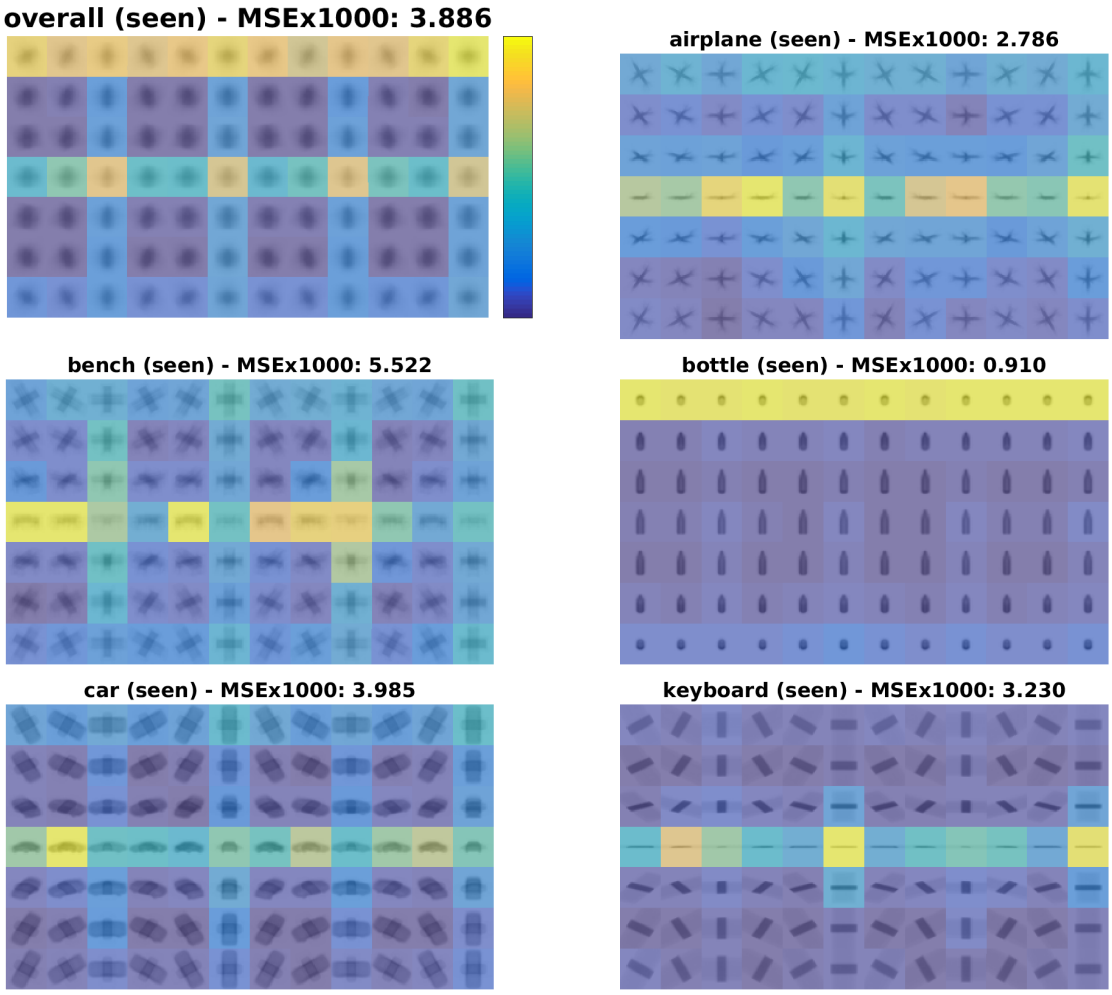


Figure 5.7: ModelNet reconstruction MSE for a few sample seen classes, conditioned on observed view (best observed in pdf at high resolution). See Section 5.2.2.

5.2.3 Unsupervised feature evaluation

Our system is trained end-to-end from scratch for the task of predicting all unobserved viewpoints given only one view of an object. We now test whether it

learns visual representations useful for recognition in the process.

5.2.3.1 Nearest neighbor classification

First, as described in Section 5.1.4, we extract features from various layers in the network (fc1, fc2, fc3 in Figure 5.2) and use them as inputs in a k-nearest neighbor classifier trained for categorization of individual object views. We run this experiment on both seen and unseen class subsets on both ModelNet and ShapeNet. In each case, we use 1000 samples per class in the training set, and set $k = 5$.

We compare our features against a variety of baselines:

- **Pixels:** For this baseline the 32×32 image is vectorized and used directly as a feature vector.
- **Random weights:** A network with identical architecture to ours and initialized with the same scheme is used to extract features with no training.
- **DrLIM [64]:** This is a commonly used "slow feature analysis"-based unsupervised feature learning approach, which we also used as a baseline for work in Chapters 3 and 4. During training, DrLIM attempts to learn an invariant feature space by mapping features of views of the same training object close to each other, and pushing features of views of different training objects far apart from one another.
- **Autoencoder [14, 72, 112]:** A network is trained to observe an input view from an arbitrary viewpoint and produce exactly that same view as its output

(compared to our method which produces the full viewgrid, including views from *other* viewpoints too). A long line of work has attempted to learn unsupervised visual representations through autoencoders [14, 72, 112]. For this method, we use an architecture identical to ours except at the very last deconvolutional layer, where, rather than producing $N \times M$ output maps, it predicts just one map corresponding to the observed view itself.

- **Egomotion [3]:** Like our method, this baseline also exploits camera motion to learn unsupervised representations. While our method is trained on the task of predicting all rotated views given a starting view, this method trains on the task of predicting the camera rotation between a given pair of images. In our implementation of this baseline, we trained a model to predict eight classes of rotations, corresponding to the immediately adjacent viewpoints in the viewgrid for a given view (3×3 neighborhood).

- **Ours w. canonical alignment:** Also used in the previous experiment, this baseline is a variant of our method that is trained to produce canonically aligned viewgrids at training time, rather than views that are *relatively* displaced from the observed view.

All models are trained with identical architectures to ours until the fc3 layer, to allow fair comparison. Figure 5.8 shows detailed specifications of network architectures used in our method and baselines.

Recall that our models are trained to observe camera elevations together with views, as shown in Figure 5.2. While this is plausible in a real world setting where

an agent may know its camera elevation angle from gravity cues, for fair comparison with our baselines, we do not use location inputs when evaluating our unsupervised features. Instead, we feed in camera elevation 0° to the location module for all views.

Table 5.3 and Table 5.4 show the results for ModelNet and ShapeNet respectively. Trends across fc1, fc2, and fc3 are all very similar, and all three layers' representations appear to be approximately equally discriminative for this task for each method. Among the baselines, all unsupervised learning methods outperform "Pixels" and "Random weights", as expected. The two strongest baselines are "Egomotion" [3] and "DrLIM" [64]. Recall that "Egomotion" is especially relevant to our approach as it also has access to relative camera motion information during training. However, our approach exploits this information much more effectively. "Ours" and "Ours w. canonical alignment" both strongly outperform all prior approaches, and of the two, "Ours" marginally outperforms "Ours w. canonical alignment".

As seen from our results, "Autoencoder" features perform very poorly. Interestingly, our method — which can be thought of as a generalized autoencoder in 3D that maps one view to a full image-based shape reconstruction — learns much stronger representations. This suggests that our system benefits strongly from the incentive to learn not just an identity mapping, as in the autoencoder, but to acquire 3D understanding.

Comparison against ImageNet-pretrained features: While our baselines in the above experiments are mainly unsupervised approaches, it is interesting to ask:

how do our features compare against standard image representations learned with supervision? We use VGG features [149] trained on ImageNet. Note that to allow this comparison, we use higher-resolution ModelNet views (224×224) compared to our inputs (32×32). So, not only are these features learned with one million labeled images as supervision, they also enjoy the advantage of higher-resolution data. With these advantages, VGG fc6 features yield 60.3% accuracy for nearest neighbor classification on ModelNet-10. Comparing against the results shown in Table 5.3, these supervised features are better than our baselines, but significantly weaker than our approach. We believe that this is due in large part to the domain gap between ImageNet images and ModelNet views.

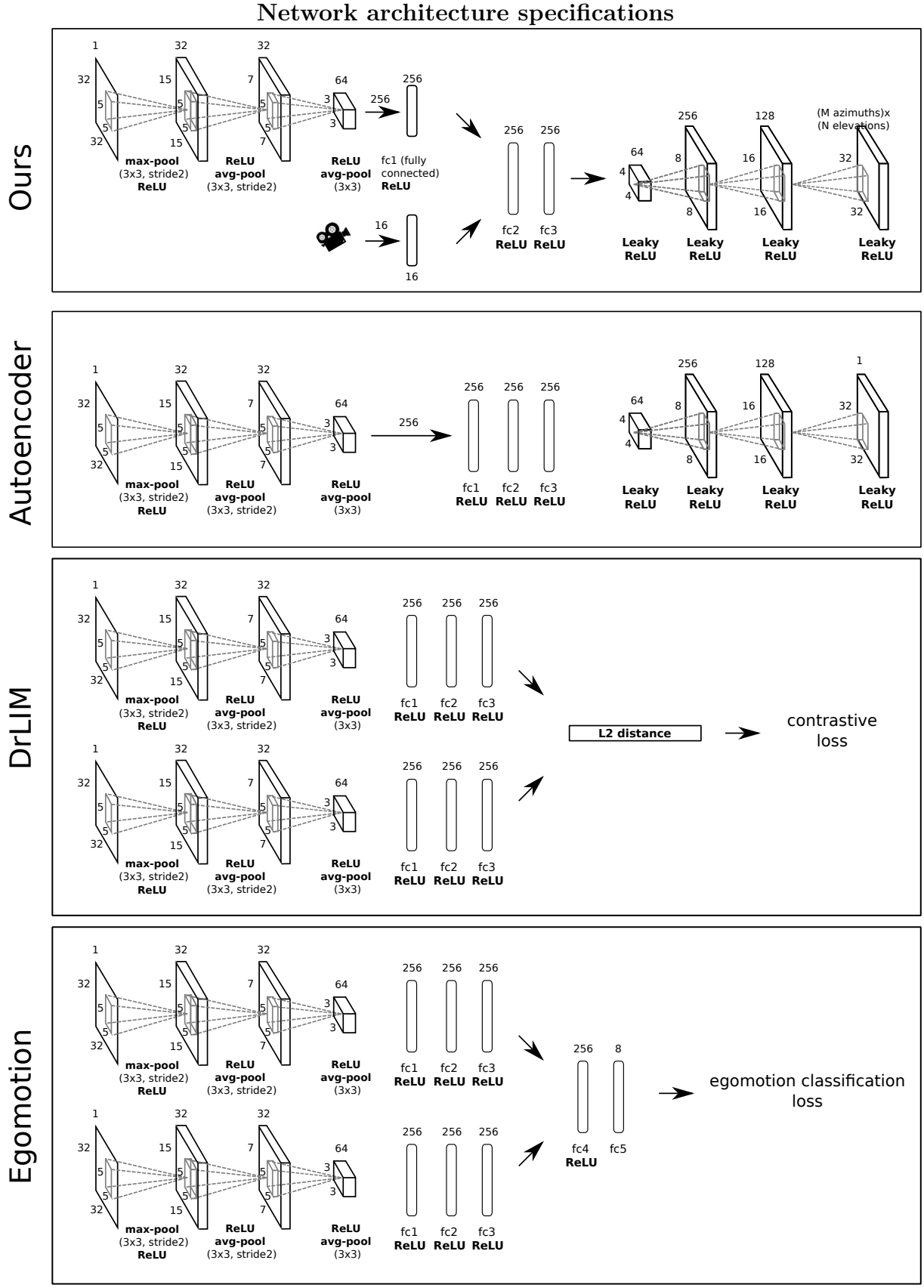


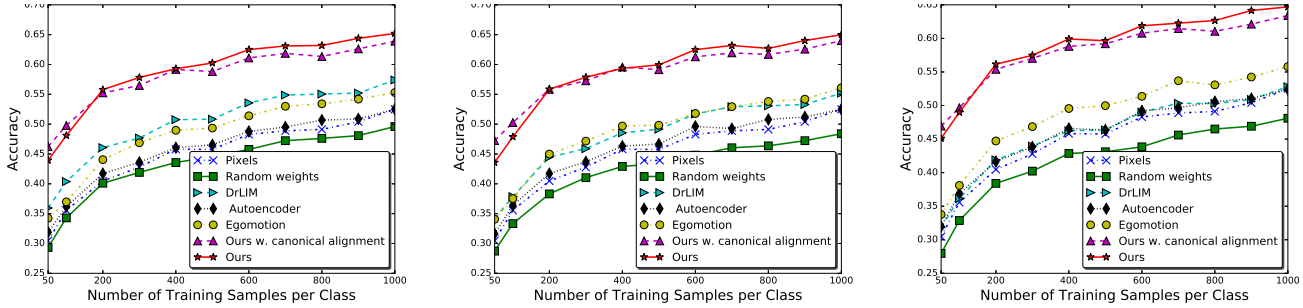
Figure 5.8: Architecture of our system and various baselines (best observed in pdf at high resolution).

Varying training set size Nearest neighbor classifiers can be sensitive to the size and specific constitution of the training set. We test the stability of the results in Table 5.3 and 5.4. To do this, we sample multiple training sets of varying sizes ranging from 50 to 1000 samples per class (50, 100, 200, 300, …1000), and report accuracies for k-nearest neighbor classification (k=5) with each training set.

These are presented for both seen and unseen class subsets of both ModelNet and ShapeNet, in Figure 5.9. We observe that curves corresponding to different methods rarely overlap as the training set is varied, establishing the stability of the results in Table 5.3 and 5.4. In particular, our one-shot reconstruction-based approaches continue to produce the most discriminative features at all training set sizes. fc1, fc2, and fc3 trends all continue to be similar.

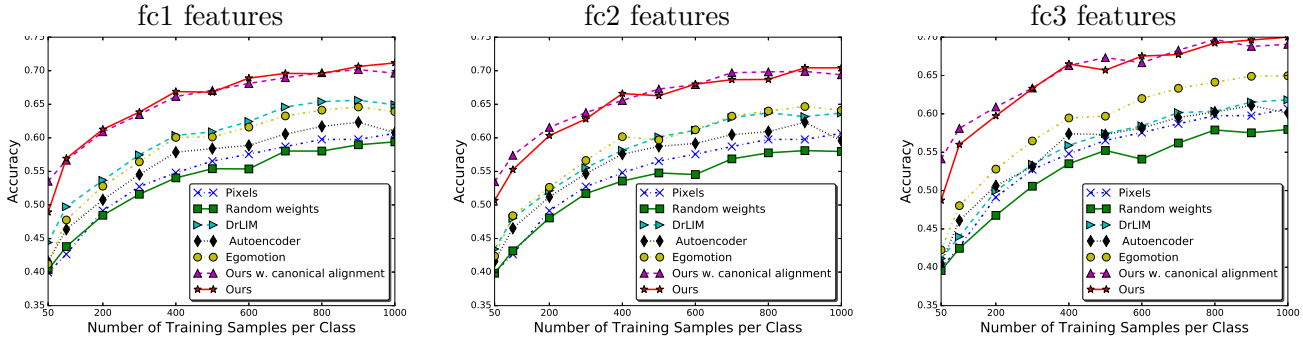
ModelNet seen classes: k-NN classification results with varying training set sizes

fc1 features | fc2 features | fc3 features



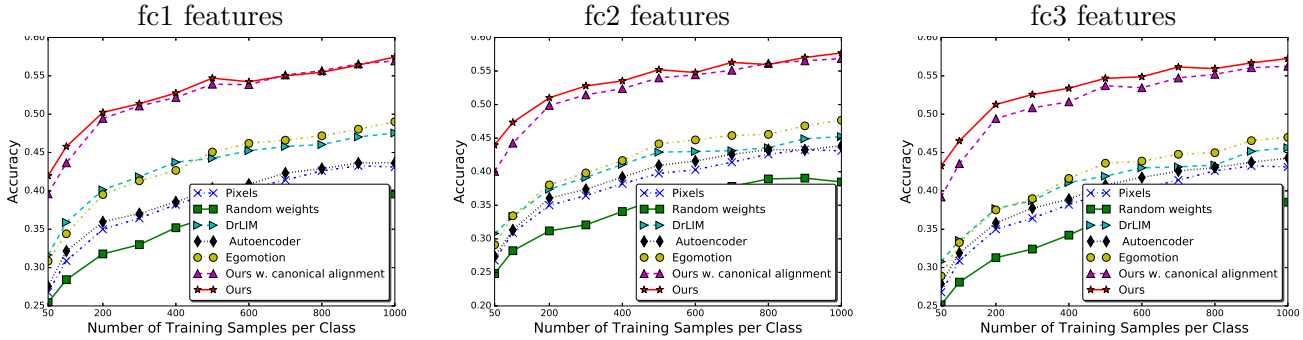
ModelNet unseen classes: k-NN classification results with varying training set sizes

fc1 features | fc2 features | fc3 features



ShapeNet seen classes: k-NN classification results with varying training set sizes

fc1 features | fc2 features | fc3 features



ShapeNet unseen classes: k-NN classification results with varying training set sizes

fc1 features | fc2 features | fc3 features

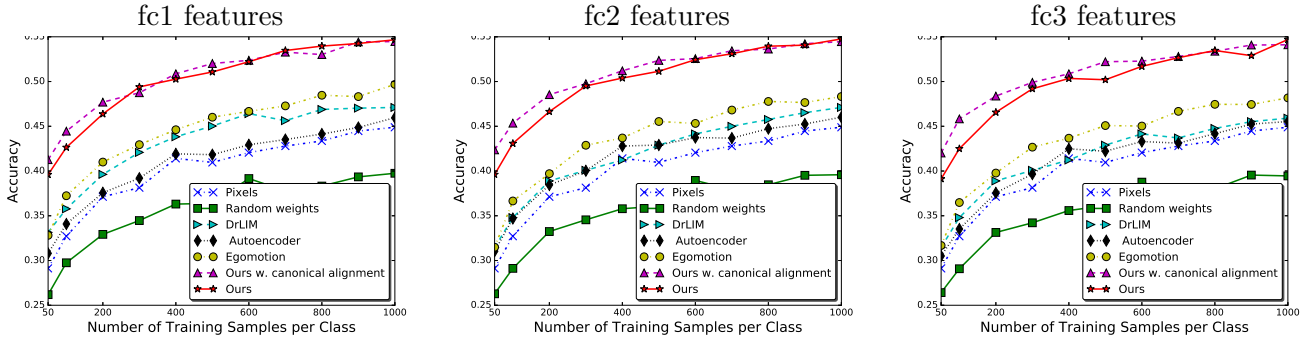


Figure 5.9: ModelNet and ShapeNet seen and unseen classes k-NN classification, varying training set size.

Datasets→	ModelNet-seen classes (30 cls)						ModelNet-unseen classes (10 cls)					
Layers→	fc1		fc2		fc3		fc1		fc2		fc3	
Methods↓/Metrics→	accuracy	mAP	accuracy	mAP	accuracy	mAP	accuracy	mAP	accuracy	mAP	accuracy	mAP
Pixels	52.5	58.4	52.5	58.4	52.5	58.4	60.7	67.1	60.7	67.1	60.7	67.1
Random weights	49.6	55.9	48.4	55.1	48.1	54.9	59.4	65.8	58.0	65.0	58.0	64.7
DrLIM [64]	57.4	63.3	55.2	61.5	52.8	59.1	64.9	70.9	63.7	69.7	61.8	68.1
Autoencoder [14, 72, 112]	52.5	58.5	52.5	58.6	52.4	58.6	60.8	67.7	59.6	66.9	60.1	67.0
Egomotion [3]	55.3	61.4	56.1	62.1	55.8	62.1	63.9	70.1	64.1	70.4	65.0	70.9
Ours w. canonical alignment	63.9	69.2	64.0	69.3	63.4	68.6	69.6	74.9	69.4	74.7	69.1	74.4
Ours	**65.2**	**70.6**	**65.0**	**70.4**	**64.7**	**69.9**	**71.2**	**76.1**	**70.4**	**75.8**	**70.0**	**75.0**

Table 5.3: Nearest neighbor classification with features from our model vs. baselines using identical architectures, on the ModelNet dataset. Results are reported as % accuracy and % mean average precision (mAP). Results with varying training dataset size are reported in Figure 5.9.

Datasets→	ShapeNet-seen classes (30 cls)						ShapeNet-unseen classes (25 cls)					
Layers→	fc1		fc2		fc3		fc1		fc2		fc3	
Methods↓/Metrics→	accuracy	mAP	accuracy	mAP	accuracy	mAP	accuracy	mAP	accuracy	mAP	accuracy	mAP
Pixels	43.1	49.3	43.1	49.3	43.1	49.3	44.9	51.1	44.9	51.1	44.9	51.1
Random weights	39.6	46.5	38.4	45.7	38.5	45.5	39.7	46.1	39.6	46.5	39.5	46.1
DrLIM [64]	47.5	52.2	45.2	50.9	45.6	49.9	47.1	52.1	47.2	53.1	45.9	53.1
Autoencoder [14, 72, 112]	43.7	50.0	43.9	50.1	44.3	50.3	46.0	51.8	46.0	52.0	45.5	51.7
Egomotion [3]	49.0	54.7	47.7	53.5	47.0	53.0	49.7	55.4	48.3	54.2	48.2	54.1
Ours w. canonical alignment	56.9	62.0	56.9	62.1	56.3	61.5	54.5	59.6	54.5	59.5	54.1	59.3
Ours	**57.5**	**62.4**	**57.7**	**62.7**	**57.3**	**62.3**	**54.7**	**59.9**	**54.8**	**60.1**	**54.7**	**59.9**

Table 5.4: Nearest neighbor classification with features from our model vs. baselenes using identical architectures, on the ShapeNet dataset. Accuracy and mean AP reported in %. Results with varying training dataset size reported in Figure 5.9.

5.2.3.2 Object category retrieval

Aside from these nearest neighbor classification experiments, we also perform object category retrieval experiments to test the usefulness of our features for a different high-level task. A retrieval system is presented with an individual object view as a query, and the task is to fetch the closest views from the training set in the learned feature space. We compare against the same baselines as for the nearest neighbor experiments. For this task, we present a query image from the test set, and retrieve the closest images in the training set, as measured by Euclidean distance in the learned feature space. For representations that encode semantics, these closest images would belong to the same category as the query, so we evaluate the various representations on their ability to retrieve images from the same class as the query.[5]

We measure top-1, top-5, and top-20 accuracies. We separately test fc1, fc2, and fc3 features from our method and the baselines, as in the classification experiments in Sec 4.3. Results are shown in Table 5.5, for ModelNet and ShapeNet for both seen and unseen classes.

Trends are very similar to those observed for classification. Ours and Ours w. canonical alignment once again easily outperform all baselines. Performance for all methods remains roughly similar at all three feature layers, except DrLIM [64]. DrLIM is strongest at fc1, and weakens at fc2 and still further at fc3, indicating that the invariance enforced by DrLIM at fc3 (See Figure 5.8) leads to loss of discriminativeness.

[5]For the unseen class experiments, images are retrieved from the corresponding unseen class training set, disjoint from the test set from which queries were drawn.

An interesting trend related to generalization to unseen classes is observed most clearly in the ShapeNet experiments, where the number of classes among seen and unseen classes is comparable (30 and 25 respectively), so that the numbers are roughly comparable among seen and unseen class experiments. Note how features from the Autoencoder baseline have much lower accuracies on unseen classes than on seen classes, demonstrating lack of generalization. Our one-shot reconstruction-based approaches have very similar accuracies on seen and unseen classes, thus establishing that they learn generalizable features. DrLIM [64] and Egomotion [3] accuracies are also comparable for seen and unseen classes, but significantly lower than our method.

ModelNet retrieval results

Datasets→	ModelNet-seen classes (30 cls)								
Layers→	fc1			fc2			fc3		
Methods↓/Metrics→	top-1	top-5	top-20	top-1	top-5	top-20	top-1	top-5	top-20
Pixels	55.2	46.9	36.6	55.2	46.9	36.6	55.2	46.9	36.6
Random weights	51.6	44.2	34.9	50.0	42.9	34.2	50.4	42.8	34.0
DrLIM [64]	58.8	51.1	41.7	56.7	48.8	39.7	53.8	46.3	37.1
Autoencoder [14, 72, 112]	54.8	46.5	37.5	55.1	46.7	37.5	55.5	47.3	37.7
Egomotion [3]	56.6	48.8	39.6	57.6	49.6	39.9	57.4	49.6	39.7
Ours w. canonical alignment	64.3	57.9	49.5	64.4	58.2	**50.2**	63.5	57.7	**50.0**
Ours	**65.6**	**59.2**	**49.7**	**64.9**	**58.3**	49.5	**65.5**	**58.4**	49.7
Datasets→	ModelNet-unseen classes (10 cls)								
Layers→	fc1			fc2			fc3		
Methods↓/Metrics→	top-1	top-5	top-20	top-1	top-5	top-20	top-1	top-5	top-20
Pixels	60.3	53.9	44.5	60.3	53.9	44.5	60.3	53.9	44.5
Random weights	60.0	52.6	44.3	57.9	51.7	43.5	57.9	51.1	43.0
DrLIM [64]	64.7	58.5	49.5	63.9	56.8	47.6	60.7	54.0	44.8
Autoencoder [14, 72, 112]	60.5	54.7	45.5	60.0	54.3	45.5	60.6	54.0	45.1
Egomotion [3]	63.4	57.4	48.2	64.2	57.3	48.2	63.8	57.4	48.3
Ours w. canonical alignment	69.0	63.9	**56.3**	68.8	63.8	**56.5**	68.4	63.3	**56.3**
Ours	**69.8**	**64.7**	55.9	**69.2**	**63.9**	55.5	**68.5**	**64.0**	55.2

ShapeNet retrieval results

Datasets→	ShapeNet-seen classes (30 cls)								
Layers→	fc1			fc2			fc3		
Methods↓/Metrics→	top-1	top-5	top-20	top-1	top-5	top-20	top-1	top-5	top-20
Pixels	42.8	36.5	28.6	42.8	36.5	28.6	42.8	36.5	28.6
Random weights	38.6	32.9	26.4	37.5	31.9	25.8	36.4	32.0	25.7
DrLIM [64]	47.1	40.2	31.3	45.9	39.6	30.2	43.3	37.1	28.9
Autoencoder [14, 72, 112]	42.5	36.8	29.6	42.6	37.1	29.8	42.3	37.4	29.5
Egomotion [3]	48.9	42.1	33.3	47.5	40.6	32.1	47.2	40.1	31.7
Ours w. canonical alignment	**56.1**	**50.8**	**44.0**	**57.3**	**51.6**	**45.2**	**56.9**	**51.4**	**43.6**
Ours	56.0	50.2	42.4	55.8	50.5	42.8	55.3	49.6	42.2
Datasets→	ShapeNet-unseen classes (25 cls)								
Layers→	fc1			fc2			fc3		
Methods↓/Metrics→	top-1	top-5	top-20	top-1	top-5	top-20	top-1	top-5	top-20
Pixels	44.6	40.4	33.6	44.6	40.4	33.6	44.6	40.4	33.6
Random weights	26.6	20.8	15.4	25.9	21.0	15.3	25.7	20.4	15.2
DrLIM [64]	48.1	41.5	36.1	48.3	41.8	34.7	44.9	41.1	34.1
Autoencoder [14, 72, 112]	30.7	23.9	17.0	31.1	23.9	17.3	30.5	23.9	17.2
Egomotion [3]	49.9	43.6	37.1	49.4	42.8	36.3	48.7	42.5	36.1
Ours w. canonical alignment	53.7	49.1	**42.9**	53.4	**49.6**	**43.5**	52.8	49.1	**43.6**
Ours	**54.9**	**49.3**	42.7	**55.1**	49.5	42.7	**54.7**	**49.2**	42.4

Table 5.5: Above: Retrieval experiments on ModelNet (1000 training samples per class). Seen class (top) and unseen class (bottom) results. Results reported as top-1, top-5 and top-20 accuracies. (Higher is better.) **Below:**.Retrieval experiments on ShapeNet (1000 training samples per class). Seen class (top) and unseen class (top) results. Results reported as top-1, top-5 and top-20 accuracies. (Higher is better.)

5.2.3.3 Comparison against our SSFA and egomotion-equivariant features

The one-shot image-based shape reconstruction method of this chapter is the third unsupervised feature learning approach we have proposed in this thesis. We now compare this method against the slow and steady features ("SSFA") of Chapter 3 and the egomotion-equivariant features ("Ego-equiv") of Chapter 4. To avoid confusion, since all three methods are ours, in this section, we will refer to the method of this chapter as "One-shot".

We perform the comparison on ModelNet-30. We first train all three methods on unsupervised data from ModelNet-30 training images. The second-order temporal coherence "SSFA" method of Chapter 3 exploits triplets of consecutive frames in video. We construct such triplets from ModelNet-30 training data viewgrids by starting at a random view (first view), selecting one of eight possible motions to an adjacent view (second view) and then repeating that same motion (third view). The "Ego-equiv" method of Chapter 4 uses camera egomotion cluster membership labels associated with pairs of images (we use the discrete egomotion-equivariance variant).

For "One-shot", we use the model trained for the experiments in previous sections. For "SSFA" and "Ego-equiv", we use identical architectures to "One-shot", up to fc3. After training, we evaluate the learned features by constructing a nearest neighbor classifier in the fc3 feature space learned by each method, as in Table 5.3.

The accuracies are reported in Table 5.6. Comparing against Table 5.3, all three methods perform well in comparison to standard unsupervised feature learning

Methods↓	accuracy
SSFA (Chapter 3)	62.3%
Ego-equiv (Chapter 4)	70.0%
One-shot (this chapter)	64.7%

Table 5.6: ModelNet-30 nearest neighbor classification accuracies using fc3 features, comparing the methods of Chapters 3 and Chapter 4 against the one-shot approach of this chapter.

baselines such as DrLIM [64], and Autoencoder [14, 72, 112]. First, the second-order temporal coherence approach of SSFA strongly outperforms the first-order approach of DrLIM (62.3 vs 57.4). Second, exploiting egomotion-equivariance in our Ego-equiv formulation yields a substantial further gain (62.3 to 70.0). Finally, One-shot is significantly better than SSFA, but falls well short of Ego-equiv (64.28 vs 70.0).

This is interesting as it points to the advantage of the feature-space formulation of Ego-equiv over the pixel-space formulation of this chapter. Specifically, optimizing the fidelity of a pixel-space reconstruction as in this chapter could discourage the development of invariances in the learned feature space as the network is forced to preserve all the information in its input. Further, the one-shot reconstruction loss of this chapter relies on Euclidean distance, which is a poor distance metric for the pixel space.

We therefore expect that a one-shot reconstruction formulation that relies on learned losses such as a GAN might succeed in learning better features. A formulation that optimizes for viewgrid reconstruction directly in the learned feature space rather than in the pixel space (similar to the feature-space "next view predictability" idea of Ego-equiv) might conceivably learn even better features by entirely avoiding the

drawbacks associated with pixel space outputs.

Pixel space prediction problems also enjoy some advantages over feature space problems. In particular, feature space prediction problems are typically formulated as regression to moving targets since the prediction targets are within the feature space that is currently being learned. Such formulations carry the risk that the network may learn to simplify the target to minimize the loss, rendering the learned feature space uninteresting. For instance, our feature space prediction losses of both Chapter 3 and Chapter 4 require contrastive losses and careful training hyperparameter selection to avoid uninteresting solutions. In contrast, for prediction problems in the pixel space, the target is fixed and losses can be much simpler regression losses as demonstrated in this chapter, so that optimization is straightforward and tractable.

5.3 Conclusion

In this chapter, we proposed an approach to mentally rotate a view of any object to arbitrary viewpoints, recovering a full image-based shape reconstruction in one shot, from a single view. Through experiments on two recent, widely used and publicly available shape datasets, we have validated that our approach learns a *single model* that can produce good shape reconstructions for a variety of objects from across many categories including categories that are not seen during training time.

Further, our approach is well-suited to learn unsupervised image features useful for recognition that can represent 3D shape information from observing just a single 2D view, since it is trained on the task of producing a full viewgrid in one shot.

Through experimental comparison against various unsupervised learning techniques, we have validated that our approach learns generic visual representations that transfer well to semantic tasks like object recognition and image retrieval, outperforming representative state of the art approaches. Our results establish the promise of explicitly targeting 3D understanding as a means of learning generically useful visual representations.

This work points to some interesting directions for follow-up research. Firstly, as pointed out above in Section 5.2.3.3, mean squared error (MSE) is a poor measure of inter-image distance— we use this both in the objective to train our approach to reconstruct viewgrids, and also to evaluate at test time how good our output reconstructions are. There is scope for improvement in both these applications: (i) For the training objective, it has recently become common practice to replace MSE and other regression-style losses (for pixel-domain output tasks like reconstruction) with a learned "true/fake" generative adversarial networks(GAN)-style loss function [61, 115, 127]. Whereas the MSE loss induces the network to average over its beliefs, the GAN loss is unforgiving of unrealistic-looking outputs. Consequently, we would expect the reconstruction results to improve significantly with such learned losses, but it is unclear whether there would be a corresponding improvement in the quality of the learned representations. (ii) While reconstruction performance is not our primary goal in this work, our evaluation of the output reconstructions using MSE distances from ground truth in the pixel space can be improved. In particular, it may be interesting to measure distances in a learned feature space instead, to judge whether high-level semantic properties are appropriately reconstructed.

Secondly, it would be interesting to recover 3D models using geometric methods from the reconstructed viewgrid as in [159], and further, to use the 3D reconstructions in turn to refine the viewgrid reconstructions.

Thirdly, we have compared our features against a previously proposed egomotion approach in this chapter [3]. However, in Chapter 4, we learned egomotion-based features that performed better than features learned using the concurrently proposed method of [3]. It would therefore be instructive to directly compare the one-shot reconstruction approach of this chapter against the egomotion-equivariance approach of Chapter 4.

Before moving to the next chapter, it is appropriate at this stage to step back and reflect on the progress in the last few chapters. In Chapter 3, we showed how higher-order temporal coherence in natural video can be exploited for unsupervised learning. In Chapter 4, we showed that agents can exploit proprioceptive egomotion knowledge as additional metadata when learning representations of the visual world from precaptured video. And finally, in this chapter, we showed how, when we remove the restriction of precaptured video and assume access to any and all possible observations of an object at training time, an unsupervised one-shot shape reconstruction task enables visual learning. All of this work until now has been focused on learning a feature mapping for images. Going forward, in Chapters 6 and 7, we will be interested in learning exploratory *behaviors* or policies for embodied agents that will dictate both the data they are exposed to during training, and the data they acquire in testing environments.

Chapter 6

End-to-end active visual category recognition

[1]In Chapters 4 and 5, we explored the idea that motion enables improved visual *learning*. However, the uses of motion in vision extend far beyond proprioceptive knowledge as a supervisory signal during learning; for embodied agents, motion plays a key role in visual *inference*. On the one hand, such agents may face the problem of unconstrained (e.g. ill-framed or poorly focused) visual input which is difficult to recognize effectively, one frame at a time. However, like a human might walk over to a window to better judge the weather, embodied agents have the opportunity to move within or act upon their environments to improve their performance. Indeed, as Gibson would say [55], the complete animal visual system consists not only of the eyes, but also the head to which those eyes are attached, and further still, the body to which that head is attached.

In the introductory chapter of this dissertation, I motivated and introduced the key concepts of this work in detail in Section 1.4. To recap, we now step into an "active vision" setting, where the agent has the ability to voluntarily and intelligently

[1]The work in this chapter was supervised by Prof. Kristen Grauman and originally published in: "Look-ahead before you leap: active vision by forecasting the effects of motion". Dinesh Jayaraman and Kristen Grauman. In Proceedings of the European Conference on Computer Vision, Amsterdam, the Netherlands, October 2016.

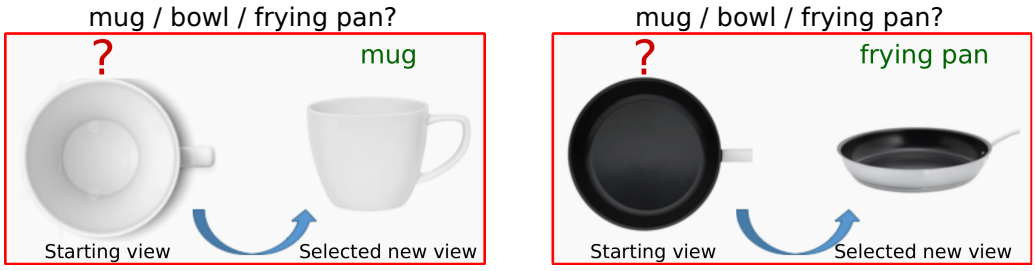


Figure 6.1: A schematic illustrating the active categorization of two objects. A moving vision system may not recognize objects after just one view, but may intelligently choose to acquire new views to disambiguate amongst its competing hypotheses.

acquire new information *at test time*. This setting is illustrated in Fig 6.1. In contrast, recent recognition research has almost exclusively focused on static image recognition tasks, where the system takes a single snapshot as input. The implicit assumption is that the input snapshot is already appropriately captured. This neglects a key challenge for embodied visual agents: intelligence is required to obtain proper inputs in the first place.

There are three technical challenges for an active vision system—control, per-view recognition, and evidence fusion. I contend that these are closely intertwined, and must be tailored to work together. In particular, as the first contribution of this component of my thesis, I develop an approach to learn all three modules of an active vision system simultaneously and end-to-end, in a deep recurrent neural network. Given an initial view and a set of possible agent motions, our approach learns how to move in the 3D environment to produce accurate categorization results.

Additionally, I hypothesize that motion planning for active vision requires an agent to internally "look before it leaps". That is, it ought to simultaneously reason about the effect of its motions on future inputs. To demonstrate this, as a second

contribution of this active recognition component of my dissertation, we jointly train our active vision system to have the ability to predict *how its internal representation of its environment will evolve* conditioned on its choice of motion. This idea builds upon the egomotion-conditioned view prediction idea of Chapter 4, but now in the context of reinforcement learning for motion policies.

Going forward, I describe our approach in Section 6.1, and present results that empirically validate its performance in Section 6.2.

6.1 Approach

First, we define the setting and data flow for active recognition (Section 6.1.1). Then we define our basic system architecture (Section 6.1.2). Finally, we describe our look-ahead module (Section 6.1.3).

6.1.1 Problem setup

We first describe our active vision setting at test time, using a 3D object category recognition scenario as a running example. Our results consider both object and scene category recognition tasks.

In the 3D object setting, the active recognition system can issue motor commands to move a camera within a viewing sphere around the 3D object X of interest. Each point on this viewing sphere is indexed by a corresponding 2D camera pose vector $\boldsymbol{p}$ indexing elevation and azimuth. An agent's manipulations of a 3D object in front of it can now be represented as a trajectory over the elevation and azimuth.

The system is allowed T timesteps to recognize every object instance X. At every timestep $t = 1, 2, \ldots T$:

- The system issues a motor command $\boldsymbol{m}_t$, e.g., "increase camera elevation by 20°, azimuth by 10°", from a set $\mathcal{M}$ of available camera motions. In our experiments, $\mathcal{M}$ is a discrete set consisting of small camera motions to points on an elevation-azimuth grid centered at the previous camera pose $\boldsymbol{p}_{t-1}$. At time $t = 1$, the "previous" camera pose p_0 is set to some random unknown vector, corresponding to the agent initializing its recognition episode at some arbitrary position with respect to the object.

- Next, the system is presented a new 2D view $\boldsymbol{x}_t = P(X, \boldsymbol{p}_t)$ of X captured from the new camera pose $\boldsymbol{p}_t = \boldsymbol{p}_{t-1} + \boldsymbol{m}_t$, where $P(.,.)$ is a projection function. This new evidence is now available to the system while selecting its next action $\boldsymbol{m}_{t+1}$.

At the final timestep $t = T$, the system must additionally predict a category label $\hat{y}$ for X, e.g., the object category it believes is most probable. In our implementation, the number of timesteps T is fixed, and all valid motor commands have uniform cost. The system is evaluated only on the accuracy of its prediction $\hat{y}$. However, the framework generalizes to the case of variable-length episodes.

6.1.2 Active recognition system architecture

Our basic active recognition system is modeled on the recurrent architecture first proposed in [117] for visual attention. Our system is composed of four basic mod-

ules: ACTOR, SENSOR, AGGREGATOR and CLASSIFIER, with weights W_a, W_s, W_r, W_c respectively. At each step t, ACTOR issues a motor command $\boldsymbol{m}_t$, which updates the camera pose vector to $\boldsymbol{p}_t = \boldsymbol{p}_{t-1} + \boldsymbol{m}_t$. Next, a 2D image $\boldsymbol{x}_t$ captured from this pose is fed into SENSOR together with the motor command $\boldsymbol{m}_t$. SENSOR produces a view-specific feature vector $\boldsymbol{s}_t = \text{SENSOR}(\boldsymbol{x}_t, \boldsymbol{m}_t)$, which is then fed into AGGREGATOR to produce aggregate feature vector $\boldsymbol{a}_t = \text{AGGREGATOR}(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_t)$. The cycle is completed when, at the next step $t+1$, ACTOR processes the aggregate feature from the previous timestep to issue $\boldsymbol{m}_{t+1} = \text{ACTOR}(\boldsymbol{a}_t)$. Finally, after T steps, the category label beliefs are predicted as $\hat{y}(W, X) = \text{CLASSIFIER}(\boldsymbol{a}_t)$, where $W = [W_a, W_s, W_r, W_c]$ is the vector of all learnable weights in the network, and for a C-class classification problem, $\hat{y}$ is a C-dimensional multinomial probability density function representing the likelihoods of the 3D object X belonging to each of the C classes. See Figure 6.2 (top) for a schematic showing how the modules are connected. Procedure block 1 lists the steps involved in the forward pass during training/inference.

In our setup, AGGREGATOR is a recurrent neural network, CLASSIFIER is a simple fully-connected hidden layer followed by a log-softmax and SENSOR separately processes the view $\boldsymbol{x}_t$ and the motor signal $\boldsymbol{m}_t$ in disjoint neural network pipelines before merging them through more layers of processing to produce the per-instance view feature $\boldsymbol{s}_t = \text{SENSOR}(\boldsymbol{x}_t, \boldsymbol{m}_t)$. ACTOR has a non-standard neural net architecture involving stochastic units: at each timestep, it internally produces an $|\mathcal{M}|$-dimensional multinomial density function $\pi(\boldsymbol{m}_t)$ over all candidate camera motions in $\mathcal{M}$, from which it samples one motion. For more details on the internal architectures of these

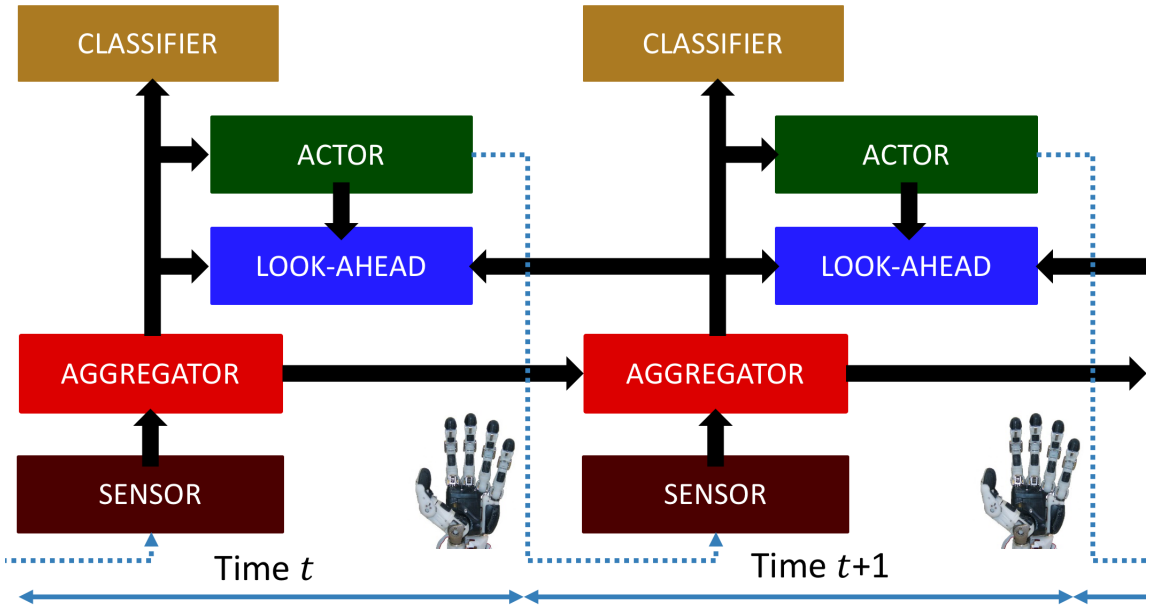


Figure 6.2: A schematic of our system architecture depicting the interaction between ACTOR, SENSOR and AGGREGATOR and CLASSIFIER modules, unrolled over timesteps. This schematic depicts an unrolled version of our network architecture, where each module is repeated once for each timestep. At training time, LOOKAHEAD acts across two timesteps, learning to predict the evolution of the output of AGGREGATOR conditional on the selected motion. See Section 6.1.2 for details.

modules, see Figure 6.3 (bottom).

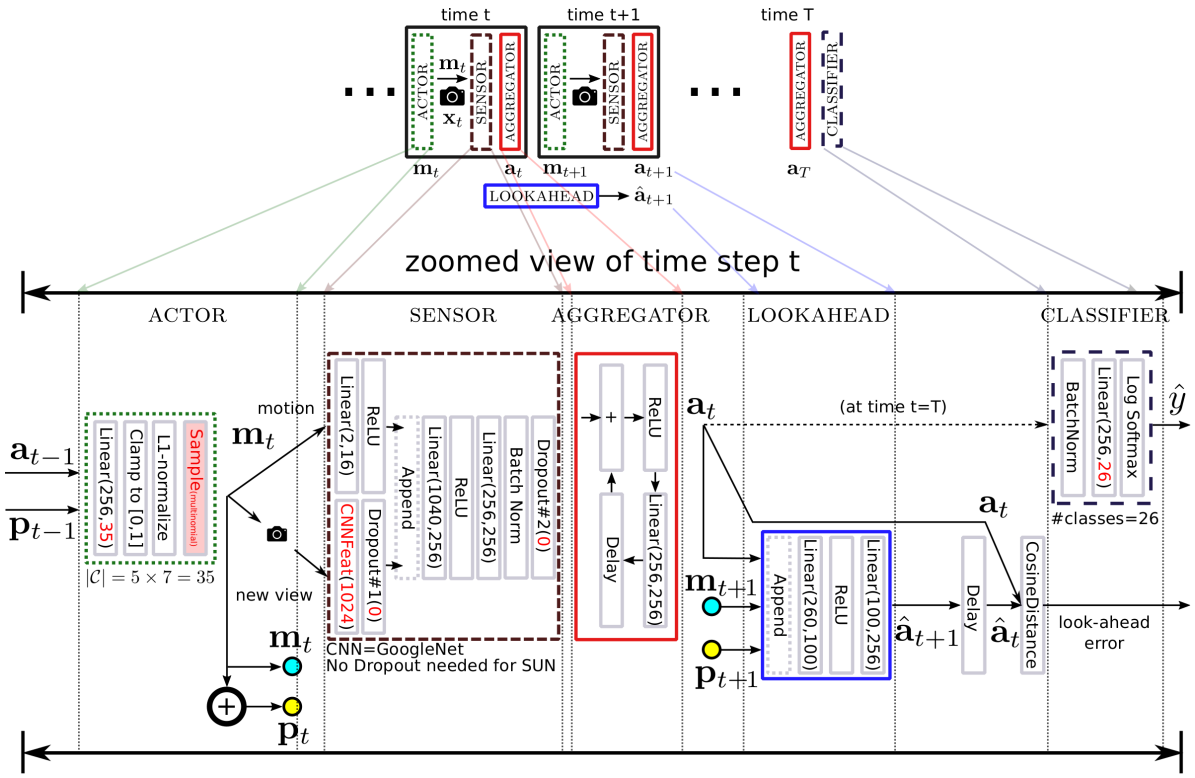


Figure 6.3: (Top) A high-level schematic of our system architecture depicting the interaction between ACTOR, SENSOR and AGGREGATOR and CLASSIFIER modules, unrolled over timesteps. Information flows from left to right. At training time, the additional LOOKAHEAD acts across two timesteps, learning to predict the evolution of the aggregate feature $\boldsymbol{a}_t$ into $\boldsymbol{a}_{t+1}$ conditional on the selected motion $\boldsymbol{m}_t$. (Bottom) A detailed schematic diagram showing the architectures of and connections amongst our active vision system modules. The small schematic at the top presents a succinct bird's-eye view of information flow within as well as between time steps, and the large schematic below zooms into the operations at some given time step t in more detail. Processing proceeds from left to right, with arrows to disambiguate where necessary. In the bottom schematic, "Linear(a,b)" denotes fully connected layers which transform a-length vector inputs to b-length vector outputs. The "Clamp" operator in ACTOR is a squashing function that sets both upper and lower limits on its inputs. The red "Sample" layer in ACTOR takes the weights of a multinomial pdf as input and samples stochastically from the distribution to produce its output (gradients cannot be backpropagated through this layer; it is trained through REINFORCE [172] instead of SGD from the classification loss). "Delay" layers store inputs internally for one time-step and output them at the next time-step. Other layer names in the schematic are self-explanatory. Input and output sizes of some layers are marked in red to denote that these are parameters derived from dataset-related choices — these are set for our SUN360 experiments in this schematic, and explanations are shown below each module. Note that AGGREGATOR is a recurrent neural network, and LOOKAHEAD may be considered a "predictive" autoencoder, that reduces its input features (appended together with the current agent motion $\boldsymbol{m}_t$) to a more compact representation in its bottleneck layer before producing its prediction of its next time-step input.

Procedure 1 Forward propagation (training/inference time)

Input: 3D instance X, together with:

- projection function $P(X, \boldsymbol{p})$ denoting the view captured from camera pose $\boldsymbol{p}$
- proprioception function $f(\boldsymbol{p})$ of the current position, which is known to the active vision system, e.g., wrist position, or gravity direction.

Output: $\hat{y}$, the predicted label for instance X.

procedure ForwardOneStep($t, \boldsymbol{a}_{t-1}, \boldsymbol{p}_{t-1}, \hat{\boldsymbol{a}}_t$) ▷ 1 forward propagation step
  $\boldsymbol{m}_t \leftarrow$ actor($\boldsymbol{a}_{t-1}, f(\boldsymbol{p}_{t-1})$) ▷ motor command sampled from $\mathcal{C}$ to adjust camera
  $\boldsymbol{p}_t \leftarrow \boldsymbol{p}_{t-1} + \boldsymbol{m}_t$ ▷ camera pose update
  $\boldsymbol{x}_t \leftarrow P(X, \boldsymbol{p}_t)$ ▷ capture new view
  $\boldsymbol{s}_t \leftarrow$ sensor($\boldsymbol{x}_t, \boldsymbol{m}_t$) ▷ per-view processing
  $\boldsymbol{a}_t \leftarrow$ aggregator($\boldsymbol{a}_{t-1}, \boldsymbol{s}_t$) ▷ evidence fusion
  if $t > 1$ **then** ▷ relevant only at training time
    $\hat{\boldsymbol{a}}_t \leftarrow$ lookahead($\boldsymbol{a}_{t-1}, \boldsymbol{m}_{t-1}, f(\boldsymbol{p}_t)$) ▷ look-ahead prediction of current time-step feature
    look-ahead error $\zeta_t \leftarrow d(\boldsymbol{a}_t, \hat{\boldsymbol{a}}_t)$
  return $\boldsymbol{a}_t, \boldsymbol{p}_t, \zeta_t$

$\boldsymbol{a}_0 \leftarrow \mathbf{0}$ ▷ initialization
$\boldsymbol{p}_0 \leftarrow$ random position
for t =1,2,...T **do** ▷ move, observe, aggregate in a loop
  $\boldsymbol{a}_t, \boldsymbol{p}_t, \zeta_t \leftarrow$ ForwardOneStep($t, \boldsymbol{a}_t, \boldsymbol{p}_{t-1}$)
$\hat{y} \leftarrow$ classifier($\boldsymbol{a}_t$) ▷ final class prediction
return $\hat{y}$

Training At training time, the network weights W are trained jointly to maximize classifier accuracy at time T. Following [117], training W follows a hybrid procedure involving both standard backpropagation and "connectionist reinforcement learning" [172]. The modules with standard deterministic neural network connections

(CLASSIFIER, AGGREGATOR and SENSOR) can be trained directly by backpropagating gradients from a softmax classification loss, while the ACTOR module which contains stochastic units can only be trained using the REINFORCE procedure of [172].

Roughly, REINFORCE treats the ACTOR module as a Partially Observable Markov Decision Process (POMDP), with the pdf $\pi(\boldsymbol{m}_t|\boldsymbol{a}_{t-1}, W)$ representing the policy to be learned. In a reinforcement learning (RL)-style approach, REINFORCE iteratively increases weights in the pdf $\pi(\boldsymbol{m})$ on those candidate motions $\boldsymbol{m} \in \mathcal{M}$ that have produced higher "rewards", as defined by a reward function. A simple REINFORCE reward function to promote classification accuracy could be $R_c(\hat{y}) = 1$ when the most likely label in $\hat{y}$ is correct, and 0 when not. To speed up training, we use a variance-reduced version of this loss $R(\hat{y}) = R_c(\hat{y}) - R_c(z)$, where z is set to the most commonly occuring class. Beyond the stochastic units, the REINFORCE algorithm produces gradients that may be propagated to non-stochastic units through standard backpropagation. In our hybrid training approach, these REINFORCE gradients from ACTOR are therefore added to the softmax loss gradients from CLASSIFIER before backpropagation through AGGREGATOR and SENSOR.

More formally, given a training dataset of instance-label pairs $\{(X^i, y^i) : 1 \leq i \leq N\}$, the gradient updates are as follows. Let $W_{\setminus c}$ denote $[W_a, W_s, W_r]$, i.e., all the weights in W except the CLASSIFIER weights W_c, and similarly, let $W_{\setminus a}$ denote

$[W_c, W_r, W_s]$. Then:

$$\Delta W^{RL}_{\setminus c} \approx \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{W_{\setminus c}} \log \pi(m^i_t | \boldsymbol{a}^i_{t-1}; W_{\setminus c}) R^i, \tag{6.1}$$

$$\Delta W^{SM}_{\setminus a} = -\sum_{i=1}^{N} \nabla_{W_{\setminus a}} L_{\text{softmax}}(\hat{y}^i(W, X), y^i), \tag{6.2}$$

where indices i in the superscripts denote correspondence to the i^{th} training sample X^i. Equation (6.1) and (6.2) show the gradients computed from the REINFORCE rewards (RL) and the softmax loss (SM) respectively, for different subsets of weights. The notation in the RHS of Equation (6.1) makes explicit the fact that the conditional action probability $\pi(m^i_t | \boldsymbol{a}^i_{t-1})$ is influenced by all the parameters in the network, except classifier weights. The REINFORCE gradients ΔW^{RL} are computed using the approximation proposed in [172]. Final gradients with respect to the weights of each module used in weight updates are given by: $\Delta W_a = \Delta W^{RL}_a$, $\Delta W_s = \Delta W^{RL}_s + \Delta W^{SM}_s$, $\Delta W_r = \Delta W^{RL}_r + \Delta W^{SM}_r$, $\Delta W_c = \Delta W^{SM}_c$. Training is through standard stochastic gradient descent with early stopping based on a validation set.

6.1.3 Look-ahead: predicting the effects of motions

Active recognition systems select the next motion based on some expectation of the next view. Though non-trivial even in the traditional instance recognition setting [27, 38, 140, 171], with instances one can exploit the fact that pose estimation in some canonical pose space is sufficient in itself to estimate properties of future views. In other words, with enough prior experience seeing the object instance, it is largely a 3D (or implicit 3D) geometric model formation problem.

In contrast, as discussed in Section 2.2, this problem is much harder in active

categorization with realistic categories—the domain we target. Predicting subsequent views in this setting is severely under-constrained, and requires reasoning about semantics and geometry together. In other words, next view planning requires some element of learning about how 3D objects *in general* change in their appearance as a function of observer motion. Indeed, we presented a proof-of-concept experiment in Chapter 4 that showed that representations that facilitate future view prediction can be exploited effectively in a simple next-best view selection task. We build upon this insight in this chapter.

Concretely, we hypothesize that the ability to predict the next view conditional on the next camera motion is closely tied to the ability to select optimal motions. Thus, rather than learn separately the model of view transitions and model of motion policies, we propose a unified approach to learn them jointly. Our idea is that knowledge transfer from a view prediction task will benefit active categorization. In this formulation, we retain the system from Section 6.1.2, but simultaneously learn to predict, at every timestep t, the impact on aggregate features $\boldsymbol{a}_{t+1}$ at the next timestep, given $\boldsymbol{a}_t$ and any choice of motion $\boldsymbol{m}_t \in \mathcal{M}$. In other words, we simultaneously learn how the *accumulated history* of learned features—not only the current view—will evolve as a function of our candidate motions.

For this auxiliary task, we introduce an additional module, LOOKAHEAD, with weights W_l into the setup of Section 6.1.2 at training time. At timestep t, LOOKAHEAD takes as input the previous timestep aggregate feature vector $\boldsymbol{a}_{t-1}$ and the last motion command issued by ACTOR $\boldsymbol{m}_t$, and produces $\hat{\boldsymbol{a}}_t$ as an estimate of the current timestep aggregate features, i.e.,$\hat{\boldsymbol{a}}_t = \text{LOOKAHEAD}(\boldsymbol{a}_{t-1}, \boldsymbol{m}_t)$. This

module may be thought of as a "predictive auto-encoder" in the space of aggregate features $\boldsymbol{a}_t$ output by AGGREGATOR. A look-ahead error loss is computed at every timestep between the predicted and actual aggregate features: $d(\hat{\boldsymbol{a}}_t, \boldsymbol{a}_t | \boldsymbol{a}_{t-1}, \boldsymbol{m}_t)$. We use the cosine distance to compute this error. This per-timestep look-ahead loss provides a third source of training gradients $\Delta W^{LA}_{\setminus ca}$ for the network weights, as it is backpropagated through AGGREGATOR and SENSOR:

$$\Delta W^{LA}_{\setminus ca} = \sum_{i=1}^{N} \sum_{t=2}^{T} \nabla_{W_{\setminus ca}} d(\hat{\boldsymbol{a}}_t, \boldsymbol{a}_t | \boldsymbol{a}_{t-1}, \boldsymbol{m}_t), \tag{6.3}$$

where W now includes W_l and LA denotes lookahead. The LOOKAHEAD module itself is trained solely from this error, so that $\Delta W_l = \Delta W^{LA}_l$. The final gradients used to train SENSOR and AGGREGATOR change to include this new loss: $\Delta W_s = \Delta W^{RL}_s + \Delta W^{SM}_s + \lambda \Delta W^{LA}_s$, $\Delta W_r = \Delta W^{RL}_r + \Delta W^{SM}_r + \lambda \Delta W^{LA}_r$. λ is a new hyperparameter that controls how much the weights in the core network are influenced by the look-ahead error loss.

The look-ahead error loss of Equation 6.3 may also be interpreted as an unsupervised regularizer on the classification objective of Equation 6.1 and 6.2. This regularizer encodes the hypothesis that good features for the active recognition task must respond in learnable, systematic ways to camera motions.

This is related to the role of "equivariant" image features in Chapter 4, where we showed that regularizing image features to respond predictably to observer ego-motions improves performance on standard static image categorization tasks. The look-ahead module in this chapter differs from the equivariant feature idea of Chapter 4 in several important ways. First, we explore the utility of look-ahead for the

active categorization problem, not recognition of individual static images. Second, the proposed look-ahead module is conceptually distinct. In particular, we propose to regularize the aggregate features from a sequence of activity, not simply per-view features. Whereas in Chapter 4 the effect of a discrete egomotion on one image is estimated by linear transformations in the embedding space, the proposed look-ahead module takes as input both the history of views and the selected motion when estimating the effects of hypothetical motions.

Proprioceptive knowledge Another useful feature of our approach is that it allows for easy modeling of proprioceptive knowledge such as the current position $\boldsymbol{p}_t$ of a robotic arm. Since the ACTOR module is trained purely through REINFORCE rewards, all other modules may access its output $\boldsymbol{m}_t$ without having to backpropagate extra gradients from the softmax loss. For instance, while the sensor module is fed $\boldsymbol{m}_t$ as input, it does not directly backpropagate any gradients to train ACTOR. Since $\boldsymbol{p}_t$ is a function solely of $(\boldsymbol{m}_1 ... \boldsymbol{m}_t)$, this knowledge is readily available for use in other components of the system without any changes to the training procedure described above. We append appropriate proprioceptive information to the inputs of ACTOR and LOOKAHEAD, detailed in experiments.

Greedy softmax classification loss We found it beneficial at training time to inject softmax classification gradients after every timestep, rather than only at the end of T timesteps. To achieve this, the CLASSIFIER module is modified to contain a bank of T classification networks with identical architectures (but different weights,

since in general, AGGREGATOR outputs $\boldsymbol{a}_t$ at different timesteps may have domain differences). Note that the REINFORCE loss is still computed only at $t = T$. Thus, given that softmax gradients do not pass through the ACTOR module, it remains free to learn non-greedy motion policies.

6.2 Experiments

We evaluate our approach for object and scene categorization. In both cases, the system must choose how it will move in its 3D environment such that the full sequence of its actions leads to the most accurate categorization results.

6.2.1 Experimental setup

While active vision systems have traditionally been tested on custom robotic setups [132] (or simple turn-table-style datasets [140]), we aim to test our system on realistic, off-the-shelf datasets in the interest of benchmarking and reproducibility. To this end, we work with two publicly available datasets, SUN360 [178] and GERMS [110]. We additionally perform direct comparison against two recently proposed active recognition methods on a synthetic 3D model dataset, ModelNet10 [176].

Our **SUN360 [178]** experiments test a scenario where the agent is exploring a 3D scene and must intelligently turn to see new parts of the scene that will enable accurate scene categorization (bedroom, living room, etc.). SUN360 consists of spherical panoramas of various indoor and outdoor scenes together with scene category labels. We use the 26-category subset (8992 panoramic images) used in [178]. Each panorama by itself represents a 3D scene instance, around which an agent

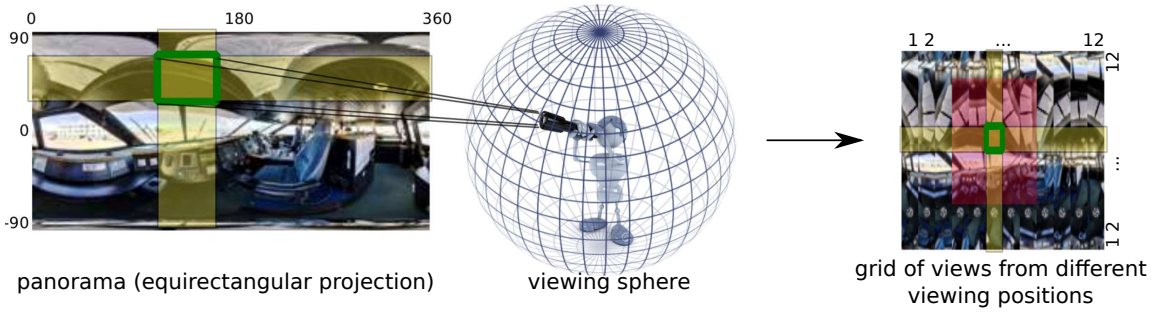


Figure 6.4: (Best seen in color) An "airplane interior" class example showing how SUN360 spherical panoramas (equirectangular projection on the left) are converted into 12×12 45°FOV view grid. As an illustration, the view at grid coordinates $x = 4$, $y = 6$ outlined in green in the view grid on the right corresponds approximately to the overlap region (also outlined in green) on the left (approximate because of panorama distortions—rectangles in the panorama are not rectangles in the rectified views present in the grid). The 5×7 red shaded region in the view grid (right) shows the motions available to ACTOR when starting from the highlighted view.

"moves" by rotating its head, as shown in Figure 6.4. For our experiments, the agent has a limited field of view (45°) at each timestep. We sample discrete views in a 12 elevations (camera pitch) $\times$ 12 azimuths (camera yaw) grid. The pitch and yaw steps are both spaced 30°apart (12×30=360), so that the entire viewing sphere is uniformly sampled on each axis. Starting from a full panorama of size 1024×2048, each 45° FOV view is represented first as a 224×224 image, from which 1024-dim. GoogLeNet [158] features are extracted from the penultimate layer. At each timestep, the agent can choose to move to viewpoints on a 5×7 grid centered at the current position. We set $T = 5$ timesteps.[2] Proprioceptive knowledge in the form of the current camera elevation angle is fed into ACTOR and LOOKAHEAD. We use a random 80-20 train-test split. Our use of SUN360 to simulate an active agent in

[2]Episode lengths were set based on learning time for efficient experimentation.

a 3D scene is new and offers a realistic scenario that we can benchmark rigorously; note that previous work on the dataset does a different task, i.e., recognition with the full panorama in hand at once [178], and results are therefore not comparable to our setting.

Our **GERMS [110]** experiments consider the scenario where a robot is holding an object and must decide on its next best motion relative to that object, e.g., to gain access to an unseen facet of the object, so as to recognize its instance label. GERMS has 6 videos each (3 train, 3 test) of 136 objects being rotated around different fixed axes, against a television screen displaying moving indoor scenes (see Figure 6.5). Each video frame is annotated by the angle at which the robotic arm is holding the object. Each video provides one collection of views that our active vision system can traverse at will, for a total of $136 \times 6 = 816$ train/test instances (compared to 8992 on SUN360). While GERMS is small and targets *instance* rather than category recognition, aside from SUN360 it is the most suitable prior dataset facilitating active recognition. Each frame is represented by a 4096-dim. VGG-net feature vector [149], provided by the authors [110]. We set episode lengths to $T = 3$ steps. As proprioceptive knowledge, we feed the current position of the robotic hand into ACTOR and LOOKAHEAD. We use the train-test subsets specified by the dataset authors.

Finally, our **ModelNet10 [176]** experiments also consider an active object recognition setup, but with synthetic 3D object models. ModelNet10 contains 4899 synthetic models of 10 household furniture categories (such as "bathtub", "bed", and "chair"). We use the pre-specified train-test split (3991 training and 908 testing

Figure 6.5: The GERMS active object instance recognition dataset [110] contains videos of a single-axis robotic hand rotating 136 toys against a moving background.

models). In our experimental setup, the active agent views one 2D projection of the 3D model at each timestep, and can choose to rotate the object to access neighboring 2D views, replicating the setup of [83] for direct comparison. Specifically, we capture 84 views of each model (7 elevations $\times$ 12 azimuths) in a viewing grid surrounding the object. At each timestep, the agent can choose to rotate the object to access one of the eight "adjacent" views within this grid. As in SUN360, elevation is available to the agent as proprioceptive information. To represent each view, we use the publicly available ImageNet-pretrained VGG16 model [149] and finetune it on ModelNet10 single view classification, before extracting fc7 features. While this dataset is synthetic unlike SUN360 and GERMS, we use it for direct comparison against two very recently proposed deep learning-based active recognition approaches [83, 176]. Some examples of 3D models from ModelNet10 are shown in Figure 6.6.

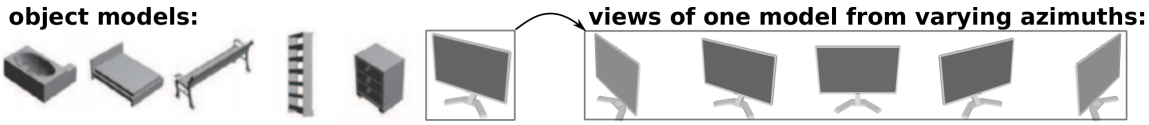


Figure 6.6: Examples of synthetic 3D models from ModelNet10, used in our active object recognition experiments.

Baselines: We extensively evaluate our "Look-ahead active RNN" (Section 6.1.3) and simpler "Active RNN" (Section 6.1.2) against eight baselines, including pas-

sive single-view methods, random view sampling, and traditional prior active vision approaches upgraded to be competitive in our setting, and recent work in the literature [83, 176].

- **`single view (neural net)`**: has access to only one view, like the starting view provided to the active systems. A feed-forward neural network is used for this baseline, composed from the appropriate components of the SENSOR and CLASSIFIER modules of our system. This baseline is entirely pose-agnostic, i.e., the same classifier is applied to views from all object poses.

- **`random views (average)`**: uses the same architecture as "`single view (neural net)`", but has access to T views, with successive views being related by randomly selected motions from the same motion set $\mathcal{M}$ available to the active systems. Its output class likelihood at $t = T$ is the average of its independent estimates of class likelihood for each view.

- **`random views (recurrent)`**: uses the same core architecture as our Active RNN method, except for the ACTOR module. In its place, random motions (from $\mathcal{M}$) are selected. Note that this should be a strong baseline, having nearly all aspects of the proposed approach except for the active view selection module. In particular, it has access to its selected motions in its SENSOR module, and can also learn to intelligently aggregate evidence over views in its AGGREGATOR RNN module.

- **`transinformation`**: is closely based on [140], in which views are selected greedily to reduce the information-theoretic uncertainty of the category hypothesis.

We make modifications for our setting, such as using 1024-D CNN features in place of the original receptive field histogram features, and using Monte Carlo sampling to approximate information gain. Each view is classified with pose-specific classifiers. When the class hypothesis is identical between consecutive views, it is emitted as output and view selection terminates. Like most prior approaches, this method relies on a canonical world coordinate space in which all object instances can be registered. Since this is infeasible in the active categorization setting, we treat each instance's coordinates as world coordinates.

- **`seqDP`**: is closely based on [36], and extends [140] using a sequential decision process with Bayesian aggregation of information between views. It runs to a fixed number of views.

- **`transinformation` + `seqDP`**: combines the strengths of [140] and [36]; it uses Bayesian information aggregation across views, and terminates early when the predicted class remains unchanged at consecutive timesteps.

- **`Depth-ShapeNets`** [176]: assumes access to depth information for all observed views, unlike our method, which only sees RGB information. It hallucinates unobserved entries in the voxel grid using 3D convolutional deep belief networks, and uses a mutual-information-based metric inspired by `seqDP` [36] to select next-best views. We compare against this method on ModelNet10 data using published numbers from [83].

- **`RGBD-Pairwise`** [83]: decomposes the sequence of observed views into pairs, classifies each pair using a CNN, and averages those pairwise classifications

Method↓/Dataset→ Performance measure→		SUN360 T=2 acc.	SUN360 T=3 acc.	SUN360 T=5 acc.	GERMS T=2 acc.	GERMS T=3 acc.
Passive approaches	Chance	14.08	14.08	14.08	0.74	0.74
	`single view (neural net)`	40.12±0.45	40.12±0.45	40.12±0.45	40.31±0.23	40.31±0.23
Random view (ablation)	`random views (average)`	45.71±0.29	50.47±0.37	54.21±0.57	45.71±0.30	46.97±0.43
	`random views (recurrent)`	47.74±0.27	51.29±0.21	55.64±0.28	44.85±0.40	44.24±0.24
Prior active approaches	`transinformation` [140]	40.69	40.69	44.86	28.83	31.02
	`seqDP` [36]	42.41	42.91	42.08	28.83	28.10
	`transinformation` + `seqDP`	44.69	46.91	48.19	29.93	29.56
`ours`	Active RNN	50.76±0.41	57.52±0.46	65.32±0.42	47.30±0.73	46.86±0.97
	Look-ahead active RNN	**51.72±0.29**	**58.12±0.43**	**66.01±0.34**	**48.02±0.68**	**47.99±0.79**
	Look-ahead active RNN+average	49.62±0.43	55.43±0.38	62.61±0.33	47.00±0.45	**48.31±0.72**

Table 6.1: Recognition accuracy on SUN360 and GERMS (neural net-based methods' scores are reported as mean ± standard error over 5 runs with different initializations)

over the full sequence to perform information fusion. For view selection, it trains a second CNN in supervised manner to directly map the current view to the best next viewpoint. Aside from the RGB information that our method accesses, `RGBD-Pairwise` assumes additional access to depth information. We compare directly against this method's published results on ModelNet-10.

Hyperparameters for all methods were optimized for overall accuracy on a validation set through iterative search over random combinations [16].

6.2.2 Results

Table 6.1 shows the recognition accuracy results for scene categorization (SUN360) and object instance recognition (GERMS). Figure 6.7 and Figure 6.8 plot the results as a function of timesteps, comparing against ablated variants of our approach and against classic prior approaches (`transinformation`, `seqDP` and `transinformation`+`seqDP`) respectively. Both variants of our method outperform the baselines on both datasets, confirming that our active approach successfully

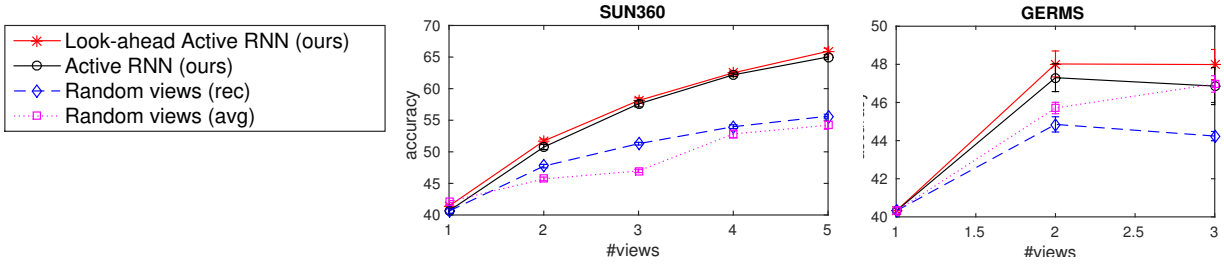


Figure 6.7: Evolution of accuracy over time for various ablated variants of our method, on SUN360 (left) and GERMS (right). Our methods show steady improvement with additional views, and easily outperform the best baselines. Also see Table 6.1.

learns intelligent view selection strategies. Passive baselines, representative of the current standard approaches to visual categorization that classify Web photos, perform uniformly poorly, highlighting the advantages of the active setting. In addition, our Look-ahead active RNN outperforms our Active RNN variant on both datasets, showing the value in simultaneously learning to predict action-conditional next views at the same time we learn the active vision policy. By "looking before leaping" our look-ahead module facilitates beneficial knowledge transfer for the active vision task.

On SUN360, even though it represents a much harder active *category recognition* problem, the margins between our method and the random view baselines are pronounced. Furthermore, while the traditional active baselines do show significant improvements from observing multiple views, they fall far short of the performance of our method despite upgrading them in order to be competitive, such as by using CNN features, as described above.

On GERMS, our method is once again easily superior to prior active methods. The margins of our gains over random-view baselines are smaller than on SUN360.

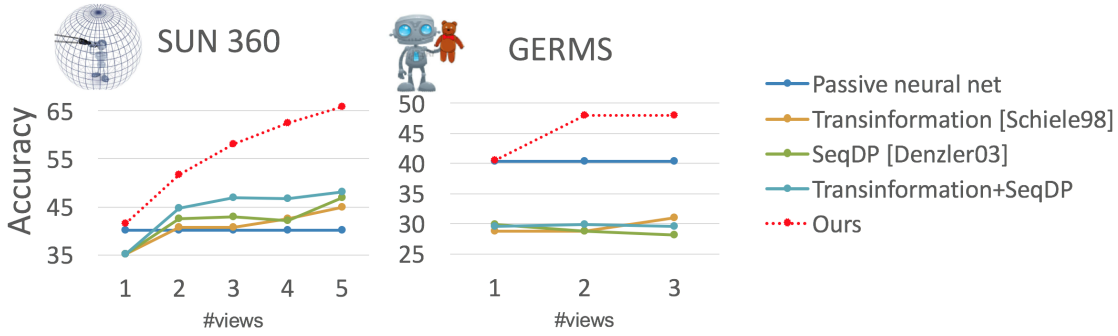


Figure 6.8: Evolution of accuracy over time for our method, vs. `transinformation` [140], `seqDP` [36], and `transinformation+seqDP`. Our integrated end-to-end solution strongly outperforms these well-known and widely used classic information-theoretical approaches to active recognition implemented with state-of-the-art CNN features, identical to our method. Also see Table 6.1.

Upon analysis, it becomes clear that this is due to GERMS being a relatively small dataset. Not only is (1) the number of active recognition instances small compared to SUN360 (816 vs. 8992), but (2) different views of the same object instance are naturally closer to each other than different views from a SUN360 panorama view-grid (see Figure 6.4 and Fig 6.5) so that even single view diversity is low, and (3) there is only a single degree of motion compared to two in SUN360. As a result, the number of possible reinforcement learning episodes is also much smaller. Upon inspection, we found that these factors can lead our end-to-end network to overfit to training data (which we countered with more aggressive regularization). In particular, it is problematic if our method achieves zero training error from just single views, so that the network has no incentive to learn to aggregate information across views well. Our active results are in line with those presented as a benchmark in the paper introducing the dataset [110], and we expect more training data is necessary

to move further with end-to-end learning on this challenge. This lack of data affects our prior active method baselines even more since they rely on *pose-specific* instance classifiers, so that each classifier's training set is very small. This explains their poor performance.

As an interesting upshot, we see further improvements on GERMS by averaging the CLASSIFIER modules' outputs, i.e., class likelihoods estimated from the aggregated features at each timestep $t = 1, .., T$ ("Look-ahead active RNN + average"). Since the above factors make it difficult to learn the optimal AGGREGATOR in an end-to-end system like ours, a second tier of aggregation in the form of averaging over the outputs of our system can yield improvements. In contrast, since SUN offers much more training data, averaging over per-timestep CLASSIFIER outputs significantly *reduces* the performance of the system, compared to directly using the last timestep output. This is exactly as one would hope for a successful end-to-end training. This reasoning is further supported by the fact that "`random views (average)`" shows slightly poorer performance than "`random views (recurrent)`" on GERMS, but is much better on SUN360.

Indeed, the significant gains of "`random views (recurrent)`" over "`random views (average)`" on SUN360 points to an important advantage of treating object/scene categorization as a grounded, sequence-based decision process. The ability to intelligently fuse observations over timesteps based on both the views themselves and the camera motions relating them offers substantial rewards. In contrast, the current computer vision literature in visual categorization is largely focused on categorization strategies that process individual images outside the context of any agent

motion or sequential data, much like the "Single view" or "`random views (average)`" baselines. We see our empirical results as an exciting prompt for future work in this space. They also suggest the need for increased efforts creating large 3D and video benchmark datasets (in the spirit of SUN360 and GERMS and beyond) to support such vision research, allowing us to systematically study these scenarios outside of robot platforms.

The result on SUN360 in particular is significant since no prior active recognition approach has been shown to successfully handle any comparably complex dataset. While active categorization is technically challenging compared to instance recognition as discussed in Section 2.2, datasets like SUN360 that contain complex visual data with ambiguous views may actually be most suited to showing the advantages of the active recognition paradigm.

Figure 6.9 and Figure 6.10 show some qualitative examples of the view selection behavior of our approach on SUN360 and GERMS respectively.

Figure 6.9: (Best viewed in color on pdf with zoom) Views selected using our approach on SUN360. Each row, corresponding to a scene, contains three red panels corresponding to the selected views at $t = 1, 2, 3$. Each panel shows the current view (left) and position on view grid (pink highlight is current position). In the top row, given the first view, our method makes reasonable but wrong guesses, but corrects itself within two moves, by observing the crowd and following their gaze.

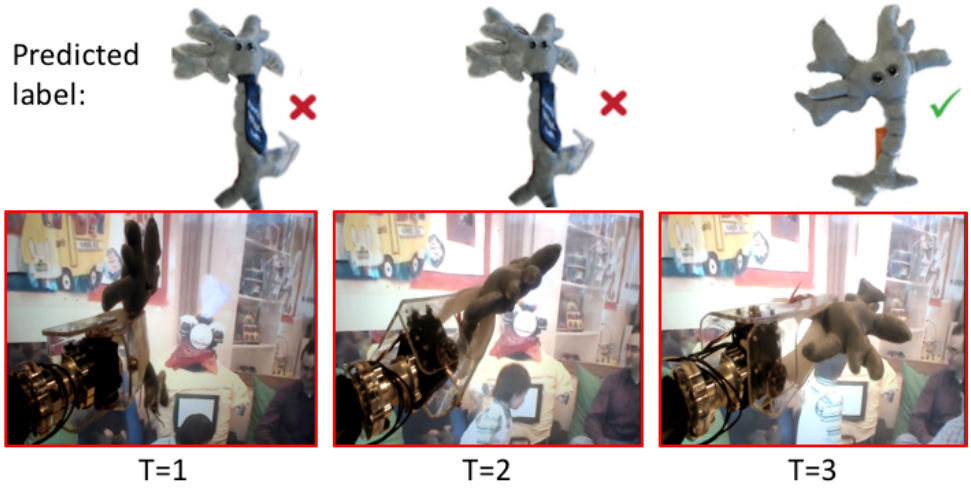


Figure 6.10: Views selected using our approach for a GERMS object. The three red panels correspond to the selected views at $t = 1, 2, 3$. The predicted instance at each timestep is depicted along the top. In this case, our approach manages to manipulate from poorly lit, side-on views to a more frontal, well-lit view to correctly recognize the object instance.

Active object recognition with synthetic CAD models: The above experiments establish the advantages of our method over traditional active recognition approaches in realistic settings using panoramic scenes (SUN360) and object manipulation videos (GERMS). Next, we use synthetic ModelNet10 object models in an active object recognition experiment to allow direct comparison against two recently published deep learning-based active recognition approaches: `Depth-ShapeNets` [176] and `RGBD-Pairwise` [83], both of which use the VGG-M CNN architecture, same as `ours`, to represent input images. We exactly reproduce the settings described in [83] and compare against the `Depth-ShapeNets` and `RGBD-Pairwise` results presented there. Figure 6.11 plots the performance of our method against these baselines. Both these methods use 2.5D information, i.e., they have additional access to depth while our method uses only 2D projections. Despite this handicap, our method significantly outperforms these baselines.

Table 6.2 presents the precise accuracy results and includes a comparison against other baselines: `random views (recurrent)`, `single-action`, and `single-action+`. As before, `random views (recurrent)` uses the same architecture as out method, except that it selects random motions at each timestep. We define `single-action` and `single-action+` below:

- **`single-action`**: This represents a policy that rotates the object by a fixed amount along a fixed direction at every time-step. We exhaustively test all valid rotation actions and only present results for the action that works best on the test set.

Method↓/Dataset→ Performance measure→	ModelNet10 T=3 acc.	 T=6 acc.
Chance	11.02	11.02
`single view (neural net)`	85.65±0.09	85.65±0.09
`random views (recurrent)`	89.23±0.03	90.10±0.08
`single-action`	89.81±0.05	90.99±0.08
`single-action+`	90.25±0.06	91.77±0.04
`Depth-ShapeNets` [176]	78.7	81.0
`RGBD-Pairwise` [83]	88.8	91.6
`ours`	**91.01±0.14**	**92.50±0.07**

Table 6.2: Recognition accuracy on ModelNet, including recently published benchmarks.

- **`single-action+`**: In our setup, since elevation is limited to $[-90°, +90°]$, a `single-action` policy that moves upwards could get stuck at the highest elevation, for instance. At this point, `single-action+` avoids this by switching directions to move downwards.

`random views (recurrent)`, `single-action`, and `single-action+` are all ablated variants of our method which replace the ACTOR module of our system with heuristic action policies while retaining the rest of our pipeline. As Table 6.2 shows, our learned action policy does significantly better than these heuristic policy baselines. At the same time, it is worth noting that even these ablated variants outperform the best previously published approaches [83, 176], suggesting that aside from the learned action policy, our approach's sensing and aggregation pipelines are also key to its superior performance

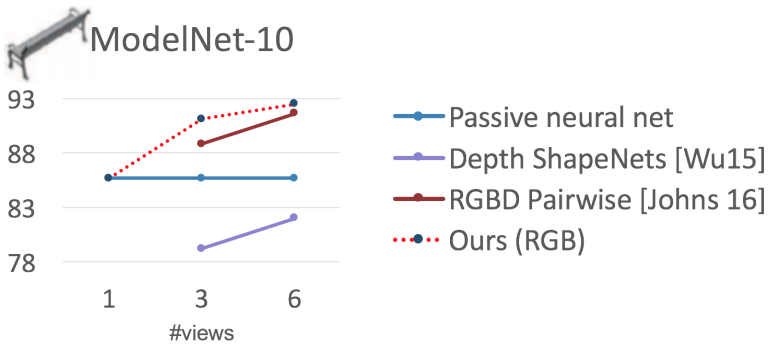


Figure 6.11: Evolution of accuracy over time for our method vs. two other recently proposed deep learning-based approaches, `Depth-ShapeNets` [176] and `RGBD-Pairwise` [83], both of which use depth information. Our integrated end-to-end solution strongly outperforms both approaches with significant margins despite using only RGB information (no depth). While RGBD Pairwise is the strongest baseline, our method's performance after seeing only three views ($T = 3$) is nearly on par with the strongest baseline's performance after selecting 6 views ($T = 6$). Also see Table 6.2.

Comparison against an agent with access to all views at once: We now compare against an imaginary agent `all-views`, that sees all possible views at the same time (during training as well as testing), and thus has access to all the information about its environment by definition. This is not practical or even possible in most active settings, but it is of value to study this imaginary agent as a useful point of comparison. Our active recognition systems sequentially select and aggregate information from a few carefully chosen views. Comparing against `all-views` helps us answer: how close do these limited views get to capturing the discriminative information in the *entire* environment?

For SUN360 scene recognition, our `all-views` model is as follows. We train a

feedforward convolutional neural network with the GoogLeNet architecture to classify full panoramas of the scenes into the 26 SUN-360 categories. All panoramas are represented as equirectangular projections. At training time, the network is initialized with ImageNet-pretrained weights and trained for panorama classification until validation error increases. Besides having access to all views, `all-views` also implicitly knows both the elevation and azimuth of each view, which is an additional advantage over our approach, which can only access elevations. The training splits are identical to those used in the experiments above. We study two variants:

- `all-views-pretrained`: For this variant, we freeze ImageNet-pretrained weights until the penultimate layer, and only train a linear classifier on top using a softmax loss. This corresponds to using the pretrained network as a feature extractor. This is in keeping with the settings used in all our previous experiments, where we extracted ImageNet-pretrained GoogLeNet features from the penultimate layer.

- `all-views-finetuned`: For this variant, we finetune ImageNet-pretrained weights with reduced learning with reduced learning rate (0.001). Unlike the network in `all-views-pretrained`, the network in `all-views-finetuned` is allowed to adapt to the equirectangular projections. The ImageNet-pretrained GoogLeNet features also adapt to the SUN360 domain, thus putting methods that use only pretrained GoogLeNet features at a disadvantage.

Recall from Table 6.1 that our active agent "Look-ahead active RNN" using ImageNet-pretrained GoogLeNet features achieves 66.01% accuracy at $T = 5$,

having seen merely 5 out of 144 possible views (12 elevations, 12 azimuths). This is on par with the performance of `all-views-pretrained`, which achieves 66.29% accuracy using the same features but with access to *all 144 views at the same time*. `all-views-finetuned` also has access to all 144 viewpoints, but uses different features that are more finetuned for SUN360 data, allowing improved scene recognition. It achieves 71.71% accuracy. These results establish that the policies learned by our approach allow the selection of nearly all the discriminative information in the scene within just 5 views.

In similar spirit to these SUN360 scene recognition experiments, we report results for ModelNet-10 active object recognition from [83] for a method that sees all 12 views at zero elevation, both at training and at test time. This method uses the multi-view convolutional neural network (MVCNN) architecture of [156] to fuse information across all views. With these 12 canonical views, MVCNN achieves 92.2% accuracy for classification. In comparison, recall from Table 6.2 that our approach achieves 92.5% accuracy with only $T = 6$ views selected using a learned policy. Similar to SUN360 scene recognition, these results suggest that our method is able to strategically select discriminative viewpoints for object recognition and fuse information across them so that only a small number of views suffice to capture nearly all of the discriminative information about the object.

Visualizing the impact of motion policies: Next we analyze the motion policies learned by our method. Recall that in our setup, the first view is selected at random and presented to the agent. Some views of an object or scene are less dis-

criminative than others, but good active recognition agents should be able to recover from poor starting views very quickly, by steering the camera or manipulating the object intelligently to reach better viewpoints.

To assess whether our approach does in fact demonstrate speedy recovery, we visualize the impact of the starting viewpoint on active recognition accuracy as a function of time on SUN360 and ModelNet10 in Figure 6.12. At $T = 1$, accuracy is a strong function of starting position on both SUN360 and ModelNet. On SUN360, azimuth has no discernible impact as expected since panoramas in scenes have no canonical forward direction. Interestingly however, elevations near the horizontal (middle band in the plots of Figure 6.12) produce the highest accuracies, suggesting that the maximum discriminative information for scene categorization exists at approximately eye level. On ModelNet10, there is a grid-like structure to the position dependence at $T = 1$, every 90 degrees along both elevation and azimuth. Upon examination, this has an interesting explanation. ModelNet10 has several approximately cuboidal object categories like table, bed, and dresser. For such objects, at every 90°rotation, a majority of the faces of the object are occluded from view, and only one or two faces are visible, which makes recognition difficult.

There are thus strong position dependencies in both datasets for individual view discriminativeness. For both `ours` and `random views (recurrent)`, the spread of accuracies is therefore large at $T = 1$, and accuracies consistently improve over time. However, the key observation from this visualization is that for `ours`, the accuracies converge to an approximately uniform distribution over starting position (i.e., negligible dependence on starting position) much more quickly on both datasets.

Our intelligent action selection approach is better able to overcome the disadvantage of bad starting positions.

6.3 Conclusion

We presented a new end-to-end approach for active visual categorization. Our framework simultaneously learns (1) how the system should move to improve its sequence of observations, and (2) how a sequence of future observations is likely to change conditioned on its possible actions. We show the impact on object and scene recognition, where our active approach makes sizeable strides over single view and passively moving systems. Furthermore, we establish the positive impact in treating all components of the active recognition system simultaneously. All together, the results are encouraging evidence that modern visual recognition algorithms can venture further into unconstrained, sequential data, moving beyond the static image snapshot labeling paradigm.

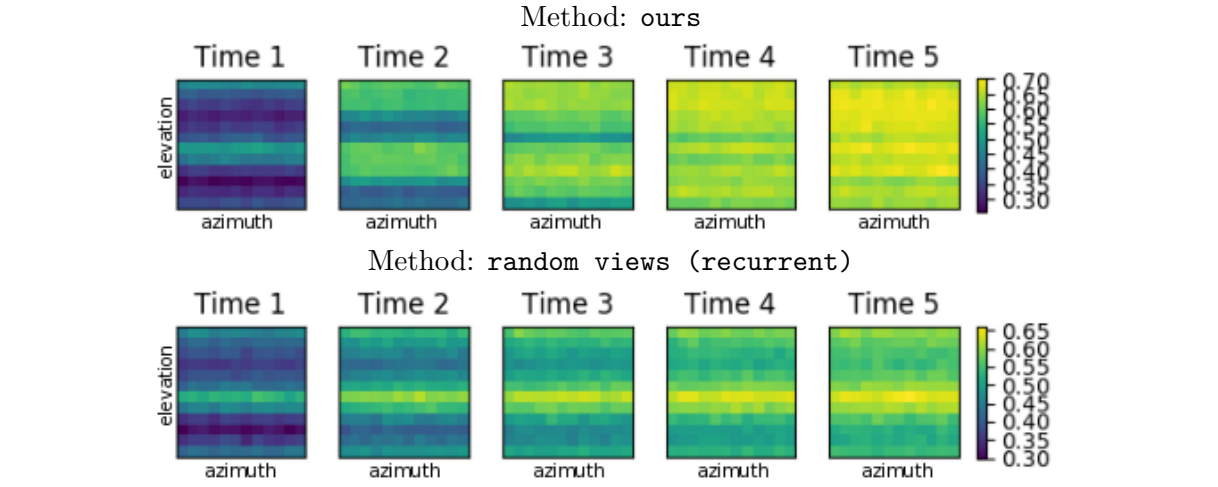
Our treatment of the active recognition problem leaves open a few directions for follow-up work. While our model refreshes its category beliefs after every observation, it does not have the option of terminating. Instead, we trained fixed time-budgeted models, and in our experiments, we tested accuracy as a function of time with fixed numbers of observations. An important future challenge is for the agent to decide when to observe another view and when that is unnecessary. This could be accomplished by setting a confidence threshold on the agent's current beliefs at inference time. A more interesting option would be to include a termination action to learn when to terminate the observation of an object or scene. A small negative

reward with each elapsed timestep could provide an agent the incentive to terminate observation early. Other situations may demand movement or energy-budgeted exploration, which could be modeled by negative rewards that are proportional to the magnitude of the selected motion.

As observed in our experiments, the look-ahead regularizer proved substantially useful only when training data was limited. This raises the question: are there better ways to utilize the look-ahead ability within an active recognition system rather than as just a regularizer? The look-ahead module is at its essence a model of state transitions within the system (considering the aggregator output as the "state"), since it models the next state conditional on current state and action. This view of the look-ahead module brings into focus the connection to model-based reinforcement learning [157], which aims to more explicitly use models of the state transition and reward dynamics of the environment in reinforcement learning systems. This is a promising direction for potential future investigation.

In the context of this dissertation's larger focus on embodied vision, this chapter represents a first step towards learning not only how to infer predictions (like category labels) based on observations but also how to intelligently acquire those observations in the first place. In the next chapter, we will continue along this direction, seeking to learn exploratory "look around" behaviors useful for general visual perception tasks.

SUN360 Plots (Starting position dependency as a function of time)



ModelNet10 Plots (Starting position dependency as a function of time)

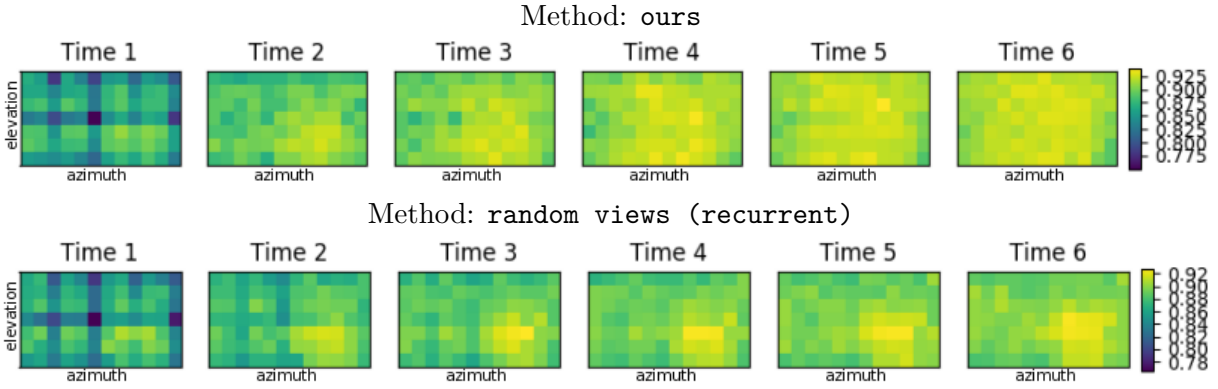


Figure 6.12: (Best seen in color) Accuracy as a function of starting position, over time. For each timestep t, a color-coded viewgrid (12×12 on SUN360, 7×12 on ModelNet) presents the accuracy after t views, when starting from each position in the viewgrid. At early timesteps, accuracy is strongly dependent on starting position since some views are more discriminative than others. However, good action policies must quickly recover from poor starting views, so that active recognition accuracies become less dependent on starting position over time. On both SUN360 active scene recognition and ModelNet10 active object category recognition, our approach achieves this much more efficiently than `random views (recurrent)`.

Chapter 7

Learning to look around

[1]As we saw in Chapter 6, visual perception requires not only making inferences from observations, but also making decisions about *what to observe*. Exploratory behaviors can be useful for perception in many varied contexts: an agent with a view of a television screen in front of it may not know if it is in a living room or a bedroom. An agent observing a mug from the side may have to move to see it from above to know if it is empty. An agent surveying a rescue site may need to explore at the onset to get its bearings. As introduced before in Section 1.5, in this final component of my thesis, I ask: can an agent learn generic exploratory "look around" behaviors *in the absence of any supervision*, and which are *not specific to any one task*, but instead generically useful?

More concretely, in the last chapter, we were concerned with making decisions about what to observe given a specific end-goal—object or scene categorization for a known lexicon of categories. Since this is a supervised problem, the agent at training time simultaneously uses the category label supervision to learn both the classifier as well as the observation policy tuned for that one specific task. In the real world, the task of an active visual agent is itself usually defined by previous observations—an

[1]The work in this chapter was supervised by Prof. Kristen Grauman.

agent observing its environment for the most part may have no specific end goal other than to gather as much information as possible about it, and new tasks may present themselves dynamically. How may such an agent *efficiently observe* its environment to acquire a correct internal model of it?

In this chapter, I pose observation selection as an independent problem in itself, with the aim of learning generic exploratory policies that gather information potentially useful to many perception tasks. An agent ought to be able to enter a new environment or pick up a new object and intelligently (non-exhaustively) look around it. This capability would be valuable in both task-driven scenarios (e.g., a drone searches for signs of a particular activity) as well as scenarios where the task itself unfolds simultaneously with the agent's exploratory actions (e.g., a search-and-rescue robot enters a burning building and dynamically decides its mission).

I present our solution to this problem of learning to actively "look around". We formulate this as the "active observation completion" problem: an agent must intelligently acquire a small set of observations optimally so that it can then hallucinate all other possible observations. We then develop a reinforcement learning solution for active visual completion. Our approach uses recurrent neural networks to aggregate information over a sequence of views. The agent is rewarded based on its predictions of unobserved views.

We explore two applications of our idea. See Figure 7.1. In the first, the agent scans an omnidirectional natural scene through its limited field of view camera; here the goal is to select efficient camera motions so that after a few glimpses, it has modeled unobserved portions of the scene well. In the second, the agent manipulates

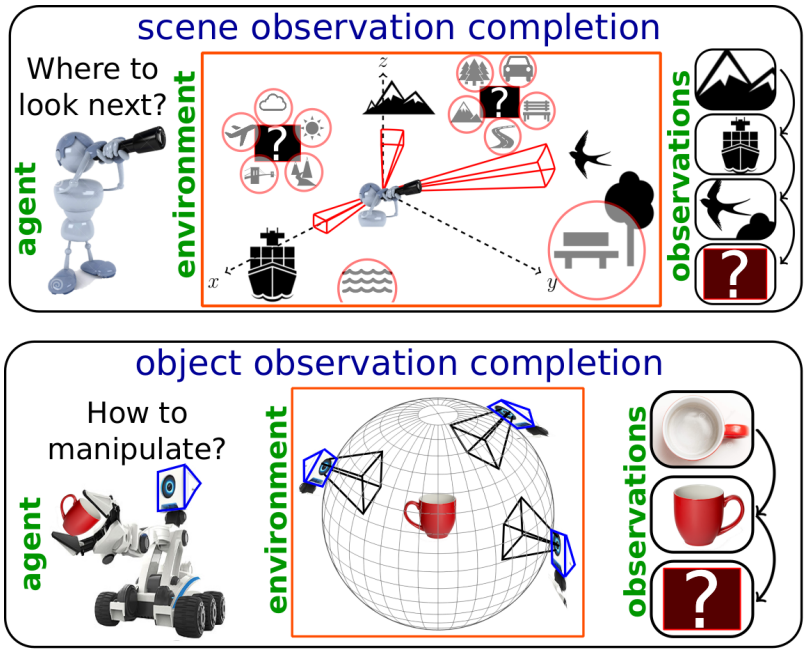


Figure 7.1: Looking around efficiently is a complex task requiring the ability to reason about regularities in the visual world using cues like context and geometry. (Left) An agent that has observed limited portions of its environment can reasonably hallucinate some unobserved portions (e.g., water near the ship), but is much more uncertain about other portions. Where should it look next? (Right) An agent inspecting a mug. Having seen a top view and a side view, how must it rotate the mug now to get maximum new information?

a 3D object to inspect it; here the goal is to select efficient manipulations so that after only a small number of actions, it has a full model of the object's 3D shape. In both cases, the system must learn to leverage visual regularities (such as shape primitives and context) that suggest the likely contents of unseen views, focusing actions on portions that are difficult to hallucinate.

In the rest of this chapter, Section 7.1 presents our approach, and Section 7.2 presents our experiments and results.

7.1 Approach

For ease of presentation, we present the problem setup as applied to a 3D object understanding task. With minor modifications (detailed in Section 7.2) our framework applies also to the panoramic scene understanding setting depicted in Figure 7.1. Both will be tested in our experiments.

7.1.1 Problem setup

The problem setting is as follows: At timestep $t = 1$, an active agent is presented with an object X in a random, unknown pose[2]. At every timestep, it can perform one action to rotate the object and observe it from the resulting viewpoint. Its objective is to make efficient exploratory rotations to understand the object's shape. It maintains an internal representation of the object shape, which it updates after every new observation. After a budget of T timesteps of exploration, it should have learned a model that can produce a view of the object as seen from any specified new viewing angle.

In practice, we discretize the space of all viewpoints into a "viewgrid" $V(X)$. To do this, we evenly sample M azimuths from 0° to 360° and N elevations from -90° to +90° and form all MN possible pairings. Each pairing of an azimuth and an elevation corresponds to one viewpoint $\boldsymbol{\theta}_i$ on a viewing sphere focused on the object. Let $\boldsymbol{x}(X, \boldsymbol{\theta}_i)$ denote the 2D image corresponding to the view of object X

[2]As in Chapter 6, we assume the elevation angle alone is known, since this is true of real-world settings due to gravity.

from viewpoint $\boldsymbol{\theta}_i$. The viewgrid $V(X)$ is the table of views $\boldsymbol{x}(X, \boldsymbol{\theta_i})$ for $1 \leq i \leq MN$. During training, the full viewgrid of each object is available to the agent as supervision. During testing, the system must predict the complete viewgrid, having seen only a few views within it.

Observe that for the special case of 3D objects, the viewgrid completion problem is identical to the image-based shape reconstruction problem tackled in Chapter 5. However, whereas Chapter 5 restricted itself to 1 randomly selected view, we now tackle the problem of view acquisition over a sequence of timesteps. Further, while Chapter 5 only deals with 3D object models, the more general active observation completion framework of this chapter also applies to other settings, like panoramic scene completion. Finally, while Chapter 5 focuses on image representation learning, the work in this chapter uses pretrained image representations and focuses instead on exploratory policy learning.

At each timestep t, the agent observes a new view $\boldsymbol{x}_t$ and updates its *prediction* for the viewgrid $\hat{V}_t(\boldsymbol{x_1}, \cdots, \boldsymbol{x_t})$. Simplifying notation a little, the problem now reduces to sequentially exploring the viewgrid V to improve $\hat{V}_t$ — in other words, actively *completing the observation* of the viewgrid $V(X)$ of object X. Given the time budget $T << MN$, the agent can see a maximum of T views out of all MN views. We choose to complete the viewgrid in the pixel-space so as to maintain generality. Our formulation is easily adaptable to more specialized settings—e.g., if the end goal only requires perceiving poses of people, the predictions could be in the keypoint space instead.

This active observation completion task poses three major challenges. Firstly,

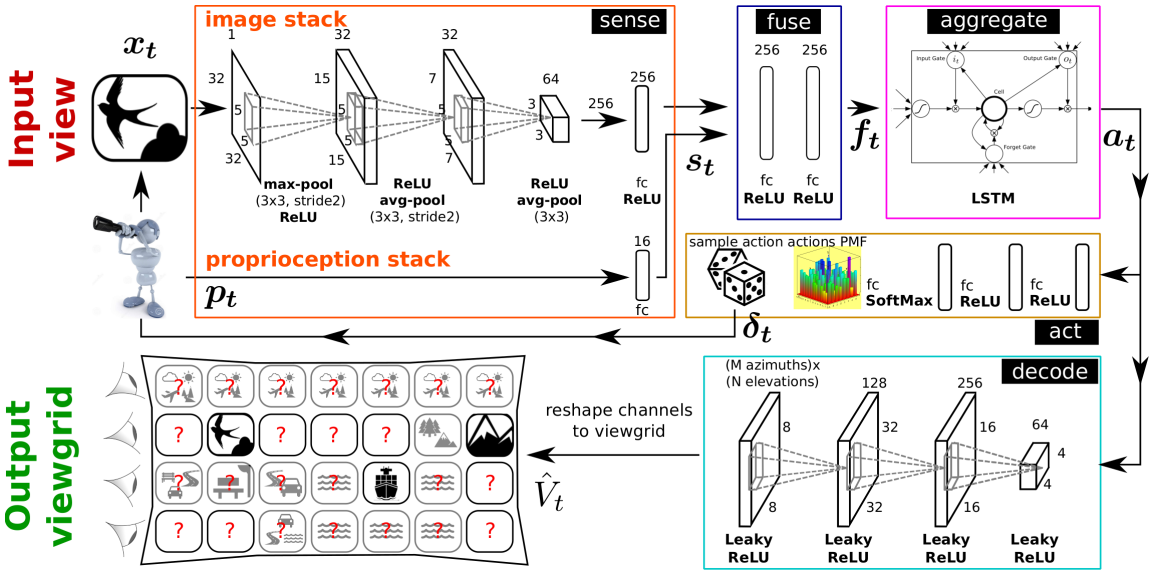


Figure 7.2: Architecture of our active observation completion system. While the input-output pair shown here is for the case of 360° scenes, we use the same architecture for ModelNet 3D objects. See Section 7.1.2 for details.

to predict unobserved views well, the agent must learn to understand 3D from very few views. Classic geometric solutions (structure-from-motion/SLAM) do not work under these conditions. Instead, reconstruction must draw on semantic and contextual cues. Secondly, intelligent action is critical to this task. Given a set of past observations, the system must act based on which new views are likely to be most informative, i.e., determine which views would most improve its model of the full viewgrid. Finally, the task is highly underconstrained—after only a few observations, there are typically many possibilities, and the agent must be able to handle this multimodality.

7.1.2 Active observation completion framework

Our solution to these challenges is a recurrent neural network, whose architecture naturally splits into five modules with distinct functions: SENSE, FUSE, AGGREGATE, DECODE, and ACT. We first present these modules and their connections; Section 7.1.3 below defines the learning objective and optimization. Architecture details for all modules are given in Figure 7.2.

Encoding to an internal model of the target First we define the core modules with which the agent encodes its internal model of the current environment. At each step t, the agent is presented with a 2D view $\boldsymbol{x}_t$ captured from a new viewpoint $\boldsymbol{\theta}_t$. We stress that absolute viewpoint coordinates $\boldsymbol{\theta}_t$ are *not* fully known, and objects/scenes are *not* presented in any canonical orientation. All viewgrids inferred by our approach treat the first view's azimuth as the origin. We assume only that the absolute elevation can be sensed using gravity, and that the agent is aware of the relative motion from the previous view. Let $\boldsymbol{p}_t$ denote this proprioceptive metadata (elevation, relative motion).

The SENSE module processes these inputs in separate neural network stacks to produce two vector outputs, which we jointly denote as $\boldsymbol{s}_t = \text{SENSE}(\boldsymbol{x}_t, \boldsymbol{p}_t)$ (see Figure 7.2, top left). FUSE combines information from both input streams and embeds it into $\boldsymbol{f}_t = \text{FUSE}(\boldsymbol{s}_t)$ (Fig 7.2, top center). Then this combined sensory information $\boldsymbol{f}_t$ from the current observation is fed into AGGREGATE, which is a long short term memory module (LSTM) [73]. AGGREGATE maintains an encoded internal model $\boldsymbol{a}_t$

of the object/scene under observation to "remember" all relevant information from past observations. At each timestep, it updates this code, combining it with the current observation to produce $\boldsymbol{a}_t = \text{AGGREGATE}(\boldsymbol{f}_1, \cdots, \boldsymbol{f}_t)$ (Fig 7.2, top right).

SENSE, FUSE, and AGGREGATE together may be thought of as performing the function of "encoding" observations into an internal model. This code $\boldsymbol{a}_t$ is now fed into two modules, for producing the output viewgrid and selecting the action, respectively.

Decoding to the inferred viewgrid DECODE translates the aggregated code into the predicted viewgrid $\hat{V}_t(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t) = \text{DECODE}(\boldsymbol{a}_t)$. To do this, it first reshapes $\boldsymbol{a}_t$ into a sequence of small 2D feature maps (Figure 7.2, bottom right), before upsampling to the target dimensions using a series of learned up-convolutions. The final up-convolution produces MN maps, one for each of the MN views in the viewgrid. For color images, we produce $3MN$ maps, one for each color channel of each view. This is then reshaped into the target viewgrid (Fig 7.2, bottom center). Seen views are pasted directly from memory to the appropriate viewgrid positions.

Acting to select the next viewpoint to observe Finally, ACT processes the aggregate code $\boldsymbol{a}_t$ to issue a motor command $\boldsymbol{\delta}_t = \text{ACT}(\boldsymbol{a}_t)$ (Figure 7.2, middle right). For objects, the motor commands rotate the object (i.e., agent manipulates the object or peers around it); for scenes, the motor commands move the camera (i.e., agent turns in the 3D environment). Upon execution, the observation's pose updates for the next timestep to $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{\delta}_t$. For $t = 1$, $\boldsymbol{\theta}_1$ is randomly sampled.

Internally, ACT first produces a distribution over all possible actions, and then samples $\boldsymbol{\delta}_t$ from this distribution. Motions in the real world are constrained to be continuous, so we restrict ACT to select "small" actions (details below). Due to the sampling operation, ACT is a *stochastic* neural network [122]. Once the new viewpoint $\boldsymbol{\theta}_{t+1}$ is set, a new view is captured and the whole process repeats. This happens until T timesteps have passed, involving $T-1$ actions.

Observe that our architectural choices are largely similar to the active recognition architecture choices in Chapter 6, with the primary difference being in the decoder module. The DECODE module in this chapter involves upconvolutions to produce a full viewgrid, whereas for the active categorization task of Chapter 6, it suffices to use a small classifier network with many fewer parameters for decoding the aggregated feature vector.

7.1.3 Objective function and model optimization

All modules are jointly optimized end-to-end to improve the final reconstructed viewgrid $\hat{V}_T$, which contains predicted views $\hat{\boldsymbol{x}}_{\boldsymbol{T}}(X, \boldsymbol{\theta}_j)$ for all viewpoints $\boldsymbol{\theta}_j, 1 \leq j \leq MN$.

A simple objective would be to minimize the distance between predicted and target views at the same viewpoint coordinate at time T: for each training object X, $L_T(X) = \sum_i d(\hat{\boldsymbol{x}}_{\boldsymbol{T}}(X, \boldsymbol{\theta}_i), \boldsymbol{x}(X, \boldsymbol{\theta}_i))$, where $d(.)$ is a distance function. However, this loss function requires viewpoint coordinates to be registered exactly in the input and target viewgrids, whereas the agent is given no external knowledge of the object's pose and thus must output viewgrids assuming the azimuth coordinate of the first

view to be the origin. Therefore, output viewgrids are shifted by an angle Δ_0 from the target viewgrid, and Δ_0 must be included in the loss function:

$$L_T(X) = \sum_{i=1}^{MN} d(\hat{\boldsymbol{x}}_{\boldsymbol{T}}(X, \boldsymbol{\theta}_i + \Delta_0), \boldsymbol{x}(X, \boldsymbol{\theta}_i)). \tag{7.1}$$

We set $d(.)$ to be the per-pixel squared $\mathcal{L}_2$ distance. With this choice, the agent expresses its uncertainty by averaging over the modes of its beliefs about unseen views. In principle, $d(.)$ could be replaced with other metrics. In particular, a GAN-style loss [61] would force the agent to select one belief mode to produce a photorealistic viewgrid, but the selected mode might not match the ground truth. Rather than one *plausible* photorealistic rendering, we aim to resolve uncertainty over time to converge to the *correct* model.

To minimize this loss, we employ a combination of stochastic gradient descent (using backpropagation through time to handle recurrence) and REINFORCE [172], as in [117]. Specifically, the gradient of the loss in Equation 7.1 is backpropagated via the DECODE, AGGREGATE, FUSE, and SENSE modules. If ACT were a standard deterministic neural network module, it could receive gradients from SENSE. However, ACT is stochastic as it involves a sampling operation. To handle this, we use the REINFORCE technique: we compute reward $R(X) = -L_T(X)$, and apply it to the outputs of ACT at all timesteps, backpropagating to encourage ACT behaviors that led to high rewards. To backpropagate through time (BPTT) to the previous timestep, the reward gradient from ACT is now passed to AGGREGATE for the previous timestep. BPTT for the LSTM module inside AGGREGATE proceeds normally with incoming gradients from the various timesteps—namely, the DECODE loss gradient for $t = T$, and the ACT reward gradients for previous timesteps.

In practice, we find it beneficial to penalize errors in the predicted viewgrid at *every* timestep, rather than only at $t = T$, so that the loss $L_T(X)$ of Equation 7.1 changes to:

$$L(X) = \sum_{t=1}^{T} \sum_{i=1}^{MN} d(\hat{\boldsymbol{x}}_{\boldsymbol{t}}(X, \boldsymbol{\theta}_i + \Delta_0), \boldsymbol{x}(X, \boldsymbol{\theta}_i)). \tag{7.2}$$

Note that this loss $L(X)$ would reduce to the loss $L_T(X)$ of Equation 7.1 if, instead of the summation over t, t were held fixed at T. Since there are now incoming loss gradients to DECODE at every timestep, BPTT involves adding reward gradients from ACT to per-timestep loss gradients from DECODE before passing through AGGREGATE. BPTT through AGGREGATE is unaffected. Our approach learns a non-myopic policy to best utilize the budget T, meaning it can learn behaviors more complex than simply choosing the next most promising observation. Accordingly, we retain the reward $R(X) = -L_T(X)$ for REINFORCE updates to ACT, based only on the final prediction; per-timestep rewards would induce greedy short-term behavior and disincentivize actions that yield gains in the long term, but not immediately.

Further, we find it useful to pretrain the entire network with $T = 1$, before training AGGREGATE and ACT with more timesteps, while other modules are frozen at their pretrained configurations. This helps avoid poor local minima and enables much faster convergence.

7.2 Experiments

7.2.1 Experimental setup

For benchmarking and reproducibility, we evaluate active settings with two widely used datasets, SUN360 and ModelNet. Recall that these datasets were used in previous chapters too. Examples from the datasets are depicted in Figure 6.4 and Figure 5.3.

On **SUN360** [178], our limited field-of-view (45°) agent attempts to complete an omnidirectional scene. SUN360 has spherical panoramas of diverse categories. We use the 26-category subset used in [76, 178]. The viewgrid has 32x32 views from 5 camera elevations (-90,-45,…,90°) and 8 azimuths (45,90,…,360°). At each timestep, the agent moves within a 3 elevations×5 azimuths neighborhood from the current position. We set training episode length $T = 6$ for training speed.

On **ModelNet** [176], our agent manipulates a 3D object to complete its image-based shape model of the object. ModelNet has two pre-specified subsets of object CAD models: ModelNet-40 (40 categories) and ModelNet-10 (10 category-subset of ModelNet-40). To help test our ability to generalize to previously unseen categories, we train on categories in ModelNet-40 that are not in ModelNet-10. We then test both on new instances from the seen categories, and on the unseen categories from ModelNet-10. The viewgrid has 32x32 views from 7 camera elevations (0,±30,±60,±90) and 12 azimuths (30,60,…,360°). Per-timestep motions are allowed within the 5×5 neighboring angles of the current viewing angle. The training episode length is $T = 4$.

Baselines We test our active completion approach "`ours`" against a variety of baselines:

- `1-view` is our method trained with $T = 1$. No information aggregation or action selection is performed by this baseline.
- `random` is identical to our approach, except that the action selection module is replaced by uniformly randomly selected actions from the pool of all possible actions.
- `large-action` chooses the largest action repeatedly. This tests if "informative" views are just far-apart views. Since there is no one largest action, we test all actions along the perimeter of the grid of allowable actions, and report results for the best-performing action on the test set.
- `large-action+` is the same as `large-action`, except that it chooses a different action at the highest/lowest elevation to avoid getting stuck. For example, if `large-action` repeatedly chose to go up by 30°, it would get stuck at 90°. `large-action+` would instead change direction to go down.
- `peek-saliency` moves to the most salient view within reach at each timestep, using a popular saliency metric [66]. To avoid getting stuck in a local saliency maximum, it does not revisit seen views. `peek-saliency` tests if salient views are informative for observation completion. Note that this baseline "peeks" at neighboring views prior to action selection to measure saliency, giving it an unfair and impossible advantage over `ours` and the other baselines.

Table 7.1: Per-pixel mean-squared error (MSE×1000) with episode length set to training length T (6 on SUN360, 4 on ModelNet), and corresponding improvement over `1-view` baseline. Lower error and higher improvement is better. RGB (luminance) values in color (gray) images are normalized to [0,1], so error values are on scale of 0 to 1000.

Dataset→	SUN360		ModelNet (seen classes)		ModelNet (unseen classes)	
Method↓ | Metric→	MSE(x1000)	Improvement	MSE(x1000)	Improvement	MSE(x1000)	Improvement
`1-view`	39.40	-	3.83	-	7.38	-
`random`	31.88	19.09%	3.46	9.66%	6.22	15.72%
`large-action`	30.76	21.93%	3.44	10.18%	6.16	16.53%
`large-action+`	28.91	26.62%	3.38	11.75%	6.00	18.70%
`peek-saliency` [66]	27.00	31.47%	3.47	9.40%	6.35	13.96%
`ours`	**23.16**	**41.22%**	**3.25**	**15.14%**	**5.65**	**23.44%**

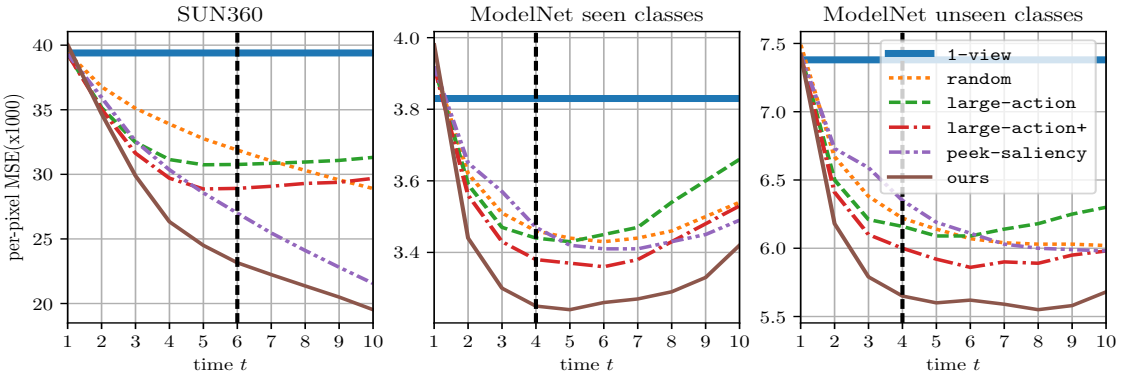


Figure 7.3: Active observation completion: per-pixel mean-squared error versus time for the three test datasets. Vertical black dotted line signifies the "look around budget" T targeted during training.

These baselines all use the same network architecture as `ours`, differing only in the exploration policy which we seek to evaluate.

7.2.2 Active observation completion results

Table 7.1 shows the scene and object completion mean-squared error on SUN360 and ModelNet (seen and unseen classes). For these results, episode lengths are held constant to T timesteps (6 on SUN360, 4 on ModelNet), same as during

training. While all the multi-view methods improve over `1-view`, our method outperforms all baselines by large margins. To isolate the impact of view selection, we report improvement over `1-view` for all methods. Compared to `random`, `ours` consistently yields approximately 2x improvement; our gains over `large-action` are also substantial in all cases, meaning that simply looking at well-spaced views is not enough. Both outcomes highlight the major value in learning to intelligently look around. Improvements are larger on more difficult datasets, where errors are larger (SUN360 > ModelNet unseen > ModelNet seen). This is as expected, since additional views are most critical where one view produces very poor results. On SUN360, `peek-saliency`, which has unfair access to neighboring views for action selection, is the strongest baseline, but still falls short of `ours`. On ModelNet data, `peek-saliency` performs poorly, likely because saliency fails to differentiate well between the synthetic CAD model views; what is informative about an object's shape is much more complex than what low-level unsupervised saliency can measure.

The plots in Figure 7.3 further show how error behaves as a function of time, including at time $t > T$, never experienced in training. With perfect information aggregation, all methods should asymptotically approach zero error at high t, which diminishes the value of intelligent exploration.[3] In practice, all methods show consistent improvement up to $t = T$, with sharpest error drops for `ours`. For $t > T$, behavior is more unpredictable since the LSTM cell in AGGREGATE is operating outside its training range, and may not effectively aggregate information. Despite

[3]This fact, together with training speed considerations, is why we limit the training time budget to $T = 4$.

this, `ours` shows strong improvement on SUN360. On ModelNet unseen classes, error flattens, and on ModelNet-seen classes, error begins to rise. In all cases, `ours` outperforms all baselines significantly at *all* t.

Figure 7.4 and Fig 7.5 present some completion episodes. As our system explores, the rough "shape" of the target scene or object emerges in its viewgrid predictions. We stress that the goal of our work is not to obtain photorealistic images. Rather, the goal is to learn policies for looking around that efficiently resolve model uncertainty in novel environments; the predicted viewgrids visualize the agent's beliefs over time. The key product of our method is a *policy*, not an image.

7.2.3 Unsupervised policy evaluation

In Chapter 6, we learned supervised policies for end-to-end active categorization (in this section, we will refer to this method as "`ours-sup-policy`"). In this chapter, we trained unsupervised exploratory policies for efficient generic information acquisition using our active observation completion framework (in this section, "`ours-unsup-policy`"). We now test how well `ours-unsup-policy` transfers to a new task with new data from unseen categories: the ModelNet-10 active categorization setting of `ours-sup-policy`.

The experimental setup is identical to the setup described in Section 6.2.1 of the last chapter, except that we use $T = 4$ and neighborhood size 5×5 to be consistent with the settings of this chapter, described in Section 7.2.1. The methods compared are as follows:

- `ours-sup-policy` is a full end-to-end active categorization system trained us-

ing our "Lookahead active RNN" approach of Chapter 6 on ModelNet-10 training data.

- `1-view` is a passive feed-forward neural network which only processes one randomly presented view and predicts its category. Its architecture is identical to `ours-sup-policy` minus the action selection and information aggregation modules.

- `ours-random-policy` is an active categorization system that selects random actions (same as baseline "`random views (recurrent)`" of Chapter 6). This uses the same core architecture as `ours-sup-policy`, except for the action selection module. In place of learned actions, it selects random legal motions from the same motion neighborhood as `ours-sup-policy`. This is trained on ModelNet-10 training data.

- `ours-unsup-policy` plugs in our unsupervised active observation completion policies into the active categorization system of Chapter 6. We then train an active observation completion model on *ModelNet-30* training data, disjoint from the target ModelNet-10 dataset classes. At test time, we run forward passes through two models simultaneously: (i) the *random actions model* trained for baseline `ours-random-policy` above (model "A"), and (ii) the observation completion model (model "B"). At every timestep, both models observe the same input view. They then communicate as follows: the ModelNet-30 observation completion model B selects actions to complete its internal model of the ModelNet-10 samples. At each timestep, this action is transmitted to

model A, in place of the randomly sampled actions that it was trained with. Recall that Model A is trained to aggregate information and make predictions on ModelNet-10 data, so it outputs ModelNet-10 class labels at each timestep. Observe that `ours-unsup-policy` thus effectively transfers policies trained on an unsupervised task, to a supervised active task on disjoint data from novel classes.

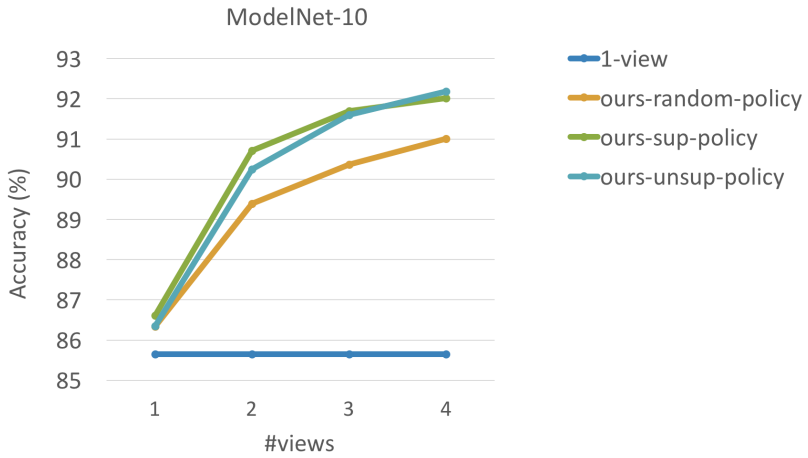


Figure 7.6: Evolution of ModelNet-10 active object categorization accuracy over time. `ours-unsup-policy` represents our unsupervised generic exploratory policy applied to the active categorization task. As seen here, it performs on par with `ours-sup-policy`, which is our end-to-end supervised policy trained specifically for active categorization using the approach of Chapter 6.

Figure 7.6 shows the results. Our unsupervised policy transfer approach `ours-unsup-policy` performs on par with our earlier end-to-end active categorization policy on this task, easily outperforming `ours-random-policy` and `1-view`. This is remarkable because the policy being employed in `ours-unsup-policy` is not only trained for the separate, unsupervised active observation completion task but

it was also trained on data from disjoint classes. This suggests that unsupervised exploratory tasks such as active observation completion could facilitate policy learning on massive unlabeled datasets in the future. Since policy learning is expensive in terms of data, computation and time, such exploratory policies once trained could be transferred to arbitrary new tasks with much smaller datasets. Performance may further improve if instead of directly transferring the policy, the policy could be finetuned for the new task, analogous to feature finetuning which is widely employed in the passive setting.

7.3 Conclusion

In this chapter, we posed and tackled a new problem: how can a visual agent learn to look around, independent of a recognition task? We present a new active observation completion framework for general exploratory behavior learning. Our reinforcement learning-based solution builds on the approach of Chapter 6 and demonstrates consistently strong results across very different settings for realistic scene and object completion, compared to multiple strong baselines.

Where Chapter 6 was concerned with learning *supervised* behaviors learned with supervision using category labels, in this chapter, we instead focus on learning unsupervised exploratory behaviors. Our results show the promise of general curiosity-driven exploration, an important step towards autonomous embodied visual agents.

Many of the potential improvements of the approach pointed out in Chapter 6 carry over to this chapter, such as learning termination policies or working in

movement-budgeted settings. Aside from this, another common observation about the work in both chapters is that there is in general no one *optimal* or *correct* policy that our approach converges on, over multiple runs. Instead, every new initialization produces an equivalent policy with near-similar performance, and it is usually difficult to understand or evaluate what is learned within these policies. In the vein of recent work on visualizing the rationales behind predictions of feed-forward neural networks [109, 188, 191], there is an opportunity for fruitful follow-up work on policy visualization to expose the rationales behind action decisions made by RNN-based active agents such as ours to allow better understanding of the training process and final model behavior.

Finally, our results in Section 7.2.3 show the promise of unsupervised exploratory policy transfer to new tasks. While this is exciting, these results may only be treated as preliminary since they have been demonstrated on only one synthetic image dataset, and when testing policy transfer to only one task (object categorization). It would be of great value to reproduce this result on more realistic datasets, testing transfer to a suite of diverse exploratory active tasks.

This chapter marks the end of my exposition of the completed technical work of my PhD. Throughout this dissertation, I have demonstrated how various avenues exist for embodied agents to acquire much of visual intelligence without manual supervision. Chapters 3, 4, and 5 showed how it is possible to learn static image representations without manual supervision through constant observation and knowledge of proprioception. Chapters 6 and 7 have shown that it is possible to employ end-to-end deep learning to acquire exploratory behaviors with or without

a specified and supervised target task. At the end of each of these chapters, I have pointed out potential directions for immediate follow-up work and improvements. Next, in Chapter 8, I will zoom out to conclude my dissertation, highlighting some promising research themes that could build upon and extend the ideas described in this manuscript in the next several years.

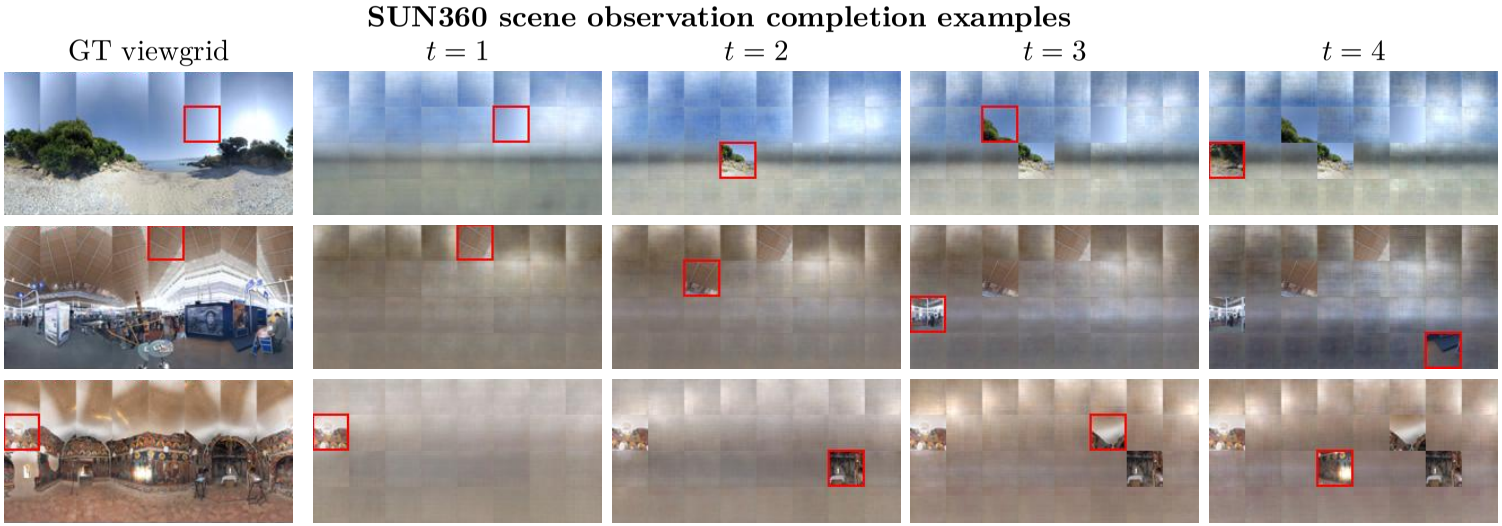


Figure 7.4: (Best viewed on pdf) Each row is one episode of SUN360 panorama observation. Column 1 shows the ground truth viewgrid with a red square around the random starting view. Columns 2-5 show our method's viewgrid completions for $t = 1, \ldots, 4$ with red squares around selected views. As the model's beliefs evolve, the space of possibilities grows more constrained, and the shape of the ground truth viewgrid begins to emerge. **Row 1:** The system correctly estimates a flat outdoor scene at $t = 1$, inferring the position of a horizon and even the sun from just one view of a gradient in the sky. At $t = 2$, it sees rocks and sand, and updates the viewgrid to begin resembling a beach. It then continues to focus on the most interesting (and unpredictable) region of the scene containing the rocks and shrubs. **Row 2:** The bright white light in the windows is never directly observed, yet the system has inferred its presence and rough shape by $t = 4$. **Row 3:** Observing a view naturally improves nearby view predictions, but more remarkably, sometimes even improves views that are far away. Here, at $t = 2$, after having seen an eye-level view, the agent appears to re-evaluate the implications of its first observation and correctly draw ceiling lights. The light background in the view observed at $t = 1$ could plausibly have been the color of a wall but, after the second view of the dark wall, the light background of the first view must be attributed to lighting.

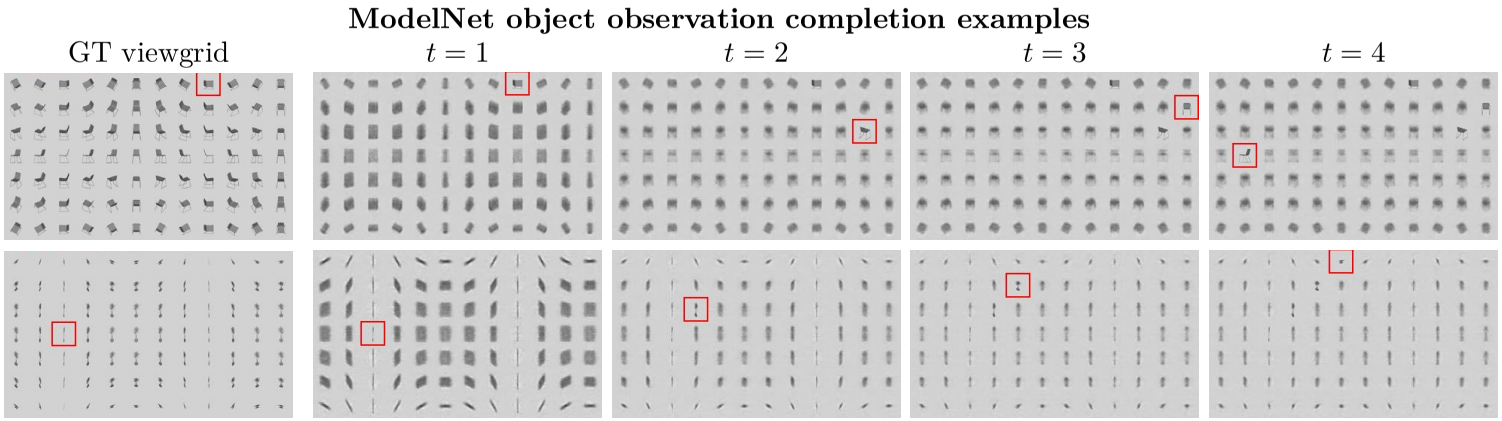


Figure 7.5: (Best viewed on pdf with zoom) Each row represents one episode for a ModelNet object. Formatting is as in Figure 7.4. **Row 1:** The first view is overhead, and azimuthally aligned with one of the sides of an unseen category object (chair). Our agent chooses to move as far from this view as possible at $t = 2$, instantly forming a much more chair-like predicted viewgrid, which continues to improve afterwards. **Row 2:** Views of narrow faces are very uninformative, and at $t = 1$, the system guesses some a flat board-like surface. After successfully moving to better views, its predictions start resembling the ground truth tree object.

Chapter 8

Conclusions and Future Work

In the preceding chapters, I have presented my thesis on embodied learning methods for recognition, in five stages:

- *Learning steady representations encoding visual dynamics from video*, in Chapter 3 [77]
- *Learning image representations tied to observer motion*, in Chapter 4 [75, 78]
- *Unsupervised learning through one-shot image-based shape reconstruction*, in Chapter 5 [80]
- *End-to-end active visual category recognition*, in Chapter 6 [76]
- *Learning to look around*, in Chapter 7 [79]

Through working on my thesis and thinking about these ideas, I have grown convinced that this nascent idea of embodied, exploratory, self-taught learning by a machine could have great long-term impact. Embodied machines of the future will have the capacity to acquire their knowledge of most aspects of the visual world, such as depth, geometry, context, objectness, saliency, and intuitive physical dynamics, entirely through unsupervised exploration. Eventually, such machines will

use supervision merely as a bridge between their self-acquired concepts and human language. These general ideas may even apply to domains other than vision.

In fact, replacing the continuous *human-provided* external supervision requirement of standard learning methods with *self-acquired* internal supervision (for example, the motor signals on a robot in my research) could prove critical to any independent artificial intelligences that may be deployed in real world situations where continuous supervision is impractical, so that, say, a robot dropped onto an unknown, unexplored planet may still have the ability to adapt to its own environment by learning through self-initiated exploration.

Having described these long term objectives, in the rest of this chapter, I describe directions for future work that I believe will be critical in making progress towards them.

Inter-task relationships Much of the work in this dissertation has been motivated by the idea that the quest for artificial intelligence must not limit itself to matching human performance on select pre-specified tasks for which large manually supervised datasets exist. Instead, we should attempt to also match biological vision in the efficiency of the learning process itself, and the ability to generalize easily to a wide variety of tasks. With this in mind, I believe that research questions surrounding *inter-task relationships* are important to future progress.

In particular, the computer vision community is working to make progress on many types of tasks, which have been identified and evolved over the years. All of our research is motivated by some "end goal" applications, such as web search,

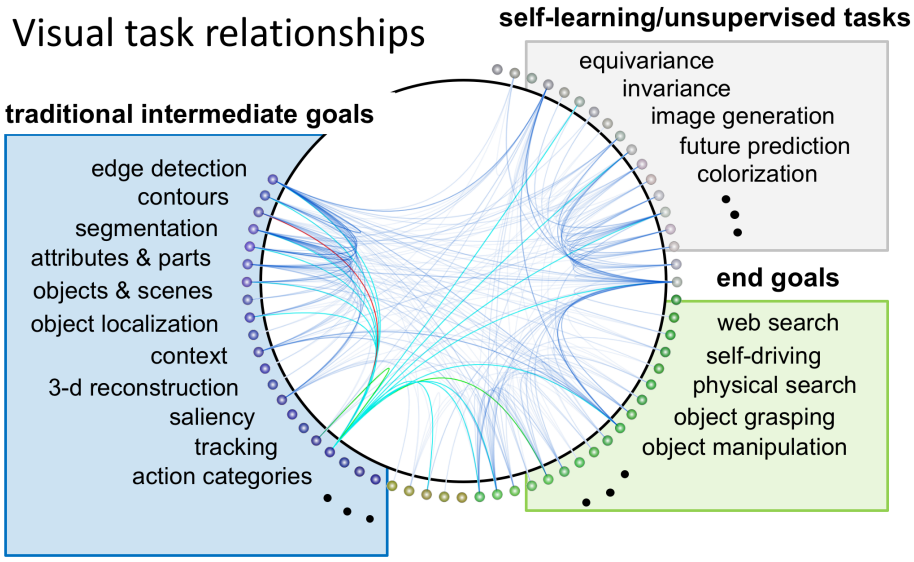


Figure 8.1: What is the nature of relationships among various types of computer vision tasks? If each task is a node in a graph, then edges between nodes correspond to relationships between tasks. Several interesting questions about these relationships remain to be investigated.

self-driving, physical object search, object grasping and manipulation. However, the community has also identified a long list of potentially useful "intermediate goal" tasks. Examples include: edge detection, segmentation, attribute recognition, object and scene categorization, and so on. Solutions for these tasks may in theory be composed together into solutions for end goal tasks. And finally, many of the tasks introduced in this dissertation fall into a third category of tasks, best classified as "self-supervised/unsupervised" tasks, which are designed to induce the ability to perform various intermediate and/or end goal tasks while not requiring any manual supervision. A system trained on these tasks must be able to learn other tasks more easily than a system that is starting from scratch. This category includes egomotion-equivariance (Chapter 4), future prediction (Chapter 3), one-shot reconstruction

(Chapter 5), jigsaw puzzle solving, and grayscale image colorization.

Having broadly classified visual tasks in this manner, we can ask many basic questions about the relationships between them. For instance, which unsupervised tasks induce the abilities to perform which supervised tasks? We could even ask broader questions about where researchers should invest their efforts: are all the intermediate goal tasks actually useful to our end goal tasks, and should we even be trying to solve all of them? For instance, which end goal tasks are enabled by modeling context, say, or saliency? Finally, is there some subset of these tasks that if a system can perform well, then it will be able to perform all the other tasks of interest? Or are there perhaps many such independent subsets? For a system that is to be trained on one such subset of tasks one after another in some ordering, are some task orderings better than others? Figure 8.1 schematically depicts our task classification within a graphical view of inter-task relationships, where each node is a task and edges correspond to relationships between pairs of tasks.

Simulated environments for sophisticated visual tasks In Chapters 6 and 7, we saw how simulated environments allow repeatable and benchmarkable evaluation of active embodied systems. While our simulated environments were set up to allow simple interactions that were sufficient for our tasks (e.g., moving a normal field of view camera over a panoramic scene or rotating a synthetic 3D object model for active scene or object recognition), more complex environments could allow simulated agents many more degrees of interaction with the environment and subsequently permit the study of more sophisticated interactive tasks. Other researchers have

recently begun efforts to create such simulated environments [131, 145, 194] or use pre-existing video game environments [136, 144].

My interest in simulated environments stems from the variety of opportunities they offer for researchers. Firstly, they can provide training sandboxes for active agents for sophisticated visual tasks such as semantic object search or driving. Unlike in the real world, hardware-level details are abstracted away, and there is virtually no limit on the amount of training experience that is possible. Secondly, they open up avenues for investigating questions about inter-task relationships, such as those introduced above. For instance, since the environment is completely human-designed and therefore contains rich annotations, it may be possible in a simulated car racing game to simulate 90% semantic segmentation and assess its impact on self-driving performance. Finally, these environments allow systematic perturbations of the underlying elements of image formation (such as object presence, occlusions, shading, camera positioning, and lighting) to better understand perception algorithms and address their shortcomings.

Figure 8.2: Which long-term unsupervised end goals would induce visual intelligence in computational agents, analogous to survival and reproduction for biological agents?

Unsupervised and self-supervised learning Finally, a common thread running through this dissertation is the idea of learning without manual labels. Including the work described in this thesis, there has recently been much activity on this topic, and I believe that many opportunities remain for further investigation.

In particular, my optimism for unsupervised learning stems from the observation that animal intelligence is the by-product of an evolutionary process that optimizes for unsupervised, long-term end goals like survival and reproduction. The key shortcoming in today's unsupervised learning approaches including those introduced in this thesis appears to be that they focus on myopic, short-term visual prediction tasks for unsupervised learning, which may not require sufficient visual reasoning to effectively induce visual abilities. I believe that for successful unsupervised learning, we must instead attempt to formulate long-term end goals that could induce visual intelligence in computational agents (See Figure 8.2 for a schematic representation). In this context, embodied agents have a critical advantage over passive agents in that they can voluntarily probe their environments in many diverse ways to acquire supervisory signals. Through the work described in my thesis and the concurrent efforts of other researchers, I have come to believe that any solution for truly effective unsupervised visual learning must involve *embodied* visual agents.

Bibliography

[1] Cuda-convnet. `https://code.google.com/p/cuda-convnet/`. 61, 70, 102, 104

[2] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk. Frequency-tuned salient region detection. In *CVPR*, 2009. 36

[3] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *ICCV*, 2015. 31, 33, 38, 40, 61, 98, 99, 102, 103, 112, 114, 115, 116, 123, 151, 152, 157, 159, 160, 165

[4] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. In *IJCV*, 1988. 23, 33

[5] P. Ammirato, P. Poirson, E. Park, J. Kosecka, and A. C. Berg. A dataset for developing and benchmarking active vision. In *ICRA*, 2017. 27, 36

[6] A. Andreopoulos and J. Tsotsos. A theory of active object localization. In *ICCV*, 2009. 34

[7] A. Andreopoulos and J. Tsotsos. 50 years of object recognition: Directions forward. In *CVIU*, 2013. 23, 33

[8] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention. In *ICLR*, 2015. 36, 37

[9] R. Bajcsy. Active perception. In *Proceedings of the IEEE*, 1988. 33

[10] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos. Revisiting active perception. In *arXiv preprint arXiv:1603.02729*, 2016. 33

[11] D. Ballard. Animate vision. In *Artificial Intelligence*, 1991. 33

[12] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition*, 2017. 124

[13] L. Bazzani, H. Larochelle, V. Murino, J.-A. Ting, and N. d. Freitas. Learning attentional policies for tracking and recognition in video with deep networks. In *ICML*, 2011. 36

[14] Y. Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009. 150, 151, 157, 160, 162

[15] J. Bergstra and Y. Bengio. Slow, decorrelated features for pretraining complex cell-like networks. In *NIPS*, 2009. 9, 15, 41, 53, 72

[16] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. In *JMLR*, 2012. 185

[17] P. Berkes and L. Wiskott. Slow feature analysis yields a rich repertoire of complex cell properties. In *Journal of vision*, volume 5, 2005. 8, 49

[18] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. Sukhatme. Interactive perception: Leveraging action in perception and perception in action. In *arXiv preprint arXiv:1604.03670*, 2016. 33

[19] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. In *arXiv preprint arXiv:1604.07316*, 2016. 31

[20] H. Borotschnig, L. Paletta, M. Prantl, A. Pinz, et al. Active object recognition in parametric eigenspace. In *BMVC*, 1998. 34

[21] M. Bowling, A. Ghodsi, and D. Wilkinson. Action respecting embedding. In *ICML*, 2005. 31, 32

[22] F. Brentano. *Psychologie vom empirischen Standpunkte*. 1874. 33

[23] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. Signature verification using a "Siamese" time delay neural network. In *IJPRAI*. World Scientific, 1993. 89

[24] Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. In *ICLR*, 2016. 38, 39

[25] N. Butko and J. Movellan. Optimal scanning for faster object detection. In *CVPR*, 2009. 36

[26] C. F. Cadieu and B. A. Olshausen. Learning intermediate-level representations of form and motion from natural movies. In *Neural computation*. MIT Press, 2012. 80, 85, 99

[27] F. Callari and F. Ferrie. Active object recognition: Looking for differences. In *IJCV*, 2001. 23, 34, 175

[28] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. In *arXiv preprint arXiv:1512.03012*, 2015. 138, 139

[29] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *ICCV*, 2015. 31, 32

[30] C.-Y. Chen and K. Grauman. Inferring unseen views of people. In *CVPR*, 2014. 45

[31] C. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3D-R2N2: A unified approach for single and multi-view 3d object reconstrution. In *ECCV*, 2016. 19, 44, 129, 135

[32] T. S. Cohen and M. Welling. Transformation Properties of Learned Visual Representations. In *ICLR*, 2015. 31, 32

[33] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 40

[34] A. J. Davison and D. W. Murray. Simultaneous localization and map-building using active vision. In *TPAMI*, 2002. 37

[35] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2, 8, 38, 49, 70, 109

[36] J. Denzler and C. M. Brown. Information theoretic sensor data selection for active object recognition and state estimation. In *TPAMI*, 2002. 34, 35, 184, 185, 187

[37] S. S. Dhillon and K. Chakrabarty. Sensor placement for effective coverage and surveillance in distributed sensor networks. In *WCNC 2003*, 2003. 38

[38] S. Dickinson, H. Christensen, J. Tsotsos, and G. Olofsson. Active object recognition integrating attention and viewpoint control. In *CVIU*, 1997. 23, 34, 35, 175

[39] W. Ding and G. W. Taylor. Mental rotation by optimizing transforming distance. In *NIPS DL Workshop*, 2014. 44, 45

[40] C. Doersch, A. Gupta, and A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 38

[41] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *arXiv preprint arXiv:1310.1531*, 2013. 69, 109

[42] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative Unsupervised Feature Learning with Convolutional Neural Networks. In *NIPS*, 2014. 15, 40

[43] A. Dosovitskiy, J. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015. 19, 44, 45, 129

[44] S. Edelman and H. H. Bülthoff. Orientation dependence in the recognition of familiar and novel views of three-dimensional objects. In *Vision research*, 1992. 27

[45] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. In *IJCV*, 2010. 59

[46] A. Fathi, A. Farhadi, and J. M. Rehg. Understanding egocentric activities. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 407–414. IEEE, 2011. 47

[47] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *CVPR*, 2015. 44, 45

[48] R. Gao, D. Jayaraman, and K. Grauman. Object-centric representation learning from unlabeled videos. In *ACCV*, 2016. 38, 41, 80, 85, 99, 122, 123

[49] A. G. Garcia, A. Vezhnevets, and V. Ferrari. An active search strategy for efficient object detection. In *CVPR*, 2015. 34

[50] L. Gatys, A. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *NIPS*, 2015. 48

[51] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012. 16, 79, 101, 102

[52] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets Robotics: The KITTI Dataset. In *IJRR*, 2013. 16, 60, 79, 100, 101

[53] R. W. Gibbs Jr. *Embodiment and cognitive science.* Cambridge University Press, 2005. 2

[54] J. J. Gibson. *The perception of the visual world.* Houghton Mifflin, 1950. 2

[55] J. J. Gibson. *The ecological approach to visual perception: classic edition.* Psychology Press, 2014. 75, 166

[56] R. Girdhar, D. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016. 19, 44, 129

[57] R. Girshick, J. Donahue, T. Darrell, and J.Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 69

[58] A. M. Glenberg, B. Jaworski, M. Rischal, and J. Levin. What brains are for: Action, meaning, and reading comprehension. In *Reading comprehension strategies: Theories, interventions, and technologies*, 2007. 2

[59] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010. 61

[60] B. Gong, W.-L. Chao, K. Grauman, and F. Sha. Diverse sequential subset selection for supervised video summarization. In *Advances in Neural Information Processing Systems*, pages 2069–2077, 2014. 47

[61] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014. 164, 209

[62] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun. Unsupervised Learning of Spatiotemporally Coherent Metrics. In *ICCV*, 2015. 9, 15, 16, 41, 53, 72, 80, 85, 99, 112

[63] R. Goroshin, M. F. Mathieu, and Y. LeCun. Learning to linearize under uncertainty. In *NIPS*, 2015. 38, 39

[64] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality Reduction by Learning an Invariant Mapping. In *CVPR*, 2006. 9, 11, 38, 41, 53, 54, 60, 64, 66, 72, 84, 88, 89, 98, 99, 101, 108, 114, 115, 118, 122, 150, 152, 157, 158, 159, 160, 162

[65] C. Häne, S. Tulsiani, and J. Malik. Hierarchical surface prediction for 3d object reconstruction. In *CVPR*, 2017. 44

[66] J. Harel, C. Koch, and P. Perona. Graph-based visual saliency. In *NIPS*, 2006. 36, 212, 213

[67] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision.* Cambridge university press, 2003. 19, 27, 44, 129

[68] J. Hays and A. A. Efros. Scene completion using millions of photographs. In *ACM Graphics (TOG)*, 2007. 48

[69] R. Held and A. Hein. Movement-produced stimulation in the development of visually guided behavior. In *Journal of comparative and physiological psychology*, 1963. 2, 5, 6, 74

[70] S. Helmer, D. Meger, P. Viswanathan, S. McCann, M. Dockrey, P. Fazli, T. Southey, M. Muja, M. Joya, J. Little, et al. Semantic robot vision challenge: Current state and future directions. In *IJCAI workshop*, 2009. 34

[71] G. Hinton, A. Krizhevsky, and S. Wang. Transforming auto-encoders. In *ICANN*, 2011. 16, 43, 45, 56

[72] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006. 61, 150, 151, 157, 160, 162

[73] S. Hochreiter and J. Schmidhuber. Long short-term memory. In *Neural computation*, 1997. 206

[74] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 105

[75] D. Jayaraman and K. Grauman. Learning image representations tied to egomotion. In *ICCV*, 2015. 7, 14, 17, 56, 223

[76] D. Jayaraman and K. Grauman. Look-ahead before you leap: active vision by forecasting the effects of motion. In *ECCV*, 2016. 7, 25, 211, 223

[77] D. Jayaraman and K. Grauman. Slow and steady feature analysis: higher order temporal coherence in video. In *CVPR*, 2016. 7, 10, 11, 223

[78] D. Jayaraman and K. Grauman. Learning egomotion-tied image representations from unlabeled video. In *IJCV Special Issue of Best Papers from ICCV 2015*, 2017. 7, 14, 17, 56, 223

[79] D. Jayaraman and K. Grauman. Learning to look around. In *(Under review)*, 2017. 7, 29, 223

[80] D. Jayaraman and K. Grauman. Unsupervised learning through one-shot image-based shape reconstruction. In *(Under review)*, 2017. 7, 20, 223

[81] D. Ji, J. Kwon, M. McFarland, and S. Savarese. Deep view morphing. In *arXiv*, 2017. 45

[82] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *arXiv*, 2014. 58, 61, 90

[83] E. Johns, S. Leutenegger, and A. J. Davison. Pairwise decomposition of image sequences for active multi-view recognition. In *CVPR*, 2016. 182, 183, 184, 191, 192, 193, 195

[84] A. Kar, S. Tulsiani, J. Carreira, and J. Malik. Category-specific object reconstruction from a single image. In *CVPR*, 2015. 19, 44, 129

[85] S. Karayev, M. Trentacoste, H. Han, A. Agarwala, T. Darrell, A. Hertzmann, and H. Winnemoeller. Recognizing image style. In *BMVC*, 2014. 69

[86] P. J. Kellman and E. S. Spelke. Perception of partly occluded objects in infancy. In *Cognitive psychology*, 1983. 26

[87] A. Kim and R. M. Eustice. Perception-driven navigation: Active visual slam for robotic area coverage. In *ICRA*, 2013. 37

[88] J. J. Kivinen and C. K. Williams. Transformation equivariant boltzmann machines. In *ICANN*, 2011. 16, 42, 81

[89] T. Kollar and N. Roy. Trajectory optimization using reinforcement learning for map exploration. In *IJRR*, 2008. 37

[90] A. Krause and C. Guestrin. Near-optimal observation selection using submodular functions. In *AAAI*, 2007. 38

[91] A. Krizhevsky. Learning multiple layers of features from tiny images, 2009. 2, 38, 70

[92] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 2, 8, 27, 38, 49, 109, 111

[93] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. 60

[94] T. Kulkarni, W. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *NIPS*, 2015. 16, 43, 44, 45

[95] G. Larsson, M. Maire, and G. Shakhnarovich. Colorization as a proxy task for visual understanding. In *CVPR*, 2017. 38, 48

[96] Y. LeCun, F. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR*, 2004. 58, 100, 101

[97] Y. J. Lee, J. Ghosh, and K. Grauman. Discovering important people and objects for egocentric video summarization. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1346–1353. IEEE, 2012. 22, 47

[98] K. Lenc and A. Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *CVPR*, 2015. 16, 43, 56, 80, 81, 107

[99] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-End training of deep visuomotor policies. In *ICRA*, 2015. 31, 32

[100] N. Li and J. DiCarlo. Unsupervised natural experience rapidly alters invariant object representation in visual cortex. In *Science*, volume 321, 2008. 8, 49

[101] Y. Li, A. Fathi, and J. M. Rehg. Learning to predict gaze in egocentric video. In *ICCV*, 2013. 47

[102] J.-P. Lies, R. M. Häfner, and M. Bethge. Slowness and sparseness have diverging effects on complex cell learning. In *PLoS computational biology*, 2014. 80, 85, 99

[103] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 2, 21, 38, 49

[104] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H.-Y. Shum. Learning to detect a salient object. In *PAMI*, 2011. 36

[105] S. Liwicki, S. Zafeiriou, and M. Pantic. Incremental slow feature analysis with indefinite kernel for online temporal video segmentation. In *ACCV*, 2012. 41, 72

[106] R. R. Llinás. *I of the vortex: From neurons to self*. MIT press Cambridge, MA, 2001. 2

[107] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999. 40

[108] Z. Lu and K. Grauman. Story-driven summarization for egocentric video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2714–2721, 2013. 47

[109] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015. 219

[110] M. Malmir, K. Sikka, D. Forster, J. Movellan, and G. W. Cottrell. Deep Q-learning for active recognition of GERMS. In *BMVC*, 2015. 35, 36, 179, 181, 182, 187

[111] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. A. Castellanos. Active policy learning for robot planning and exploration under uncertainty. In *Robotics: Science and Systems*, pages 321–328, 2007. 37

[112] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, pages 52–59. Springer, 2011. 129, 134, 150, 151, 157, 160, 162

[113] R. Memisevic. Learning to relate images. In *PAMI*, 2013. 44

[114] V. Michalski, R. Memisevic, and K. Konda. Modeling Deep Temporal Dependencies with Recurrent Grammar Cells. In *NIPS*, 2014. 16, 44

[115] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 164

[116] A. Mishra, Y. Aloimonos, and C. Fermuller. Active segmentation for robotics. In *IROS*, 2009. 33

[117] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu. Recurrent models of visual attention. In *NIPS*, 2014. 36, 37, 169, 173, 209

[118] H. Mobahi, R. Collobert, and J. Weston. Deep Learning from Temporal Coherence in Video. In *ICML*, 2009. 9, 11, 15, 16, 38, 41, 53, 59, 60, 64, 66, 72, 80, 85, 89, 98, 101, 108, 118, 122

[119] H. Moravec. Locomotion, vision and intelligence. In *Tech Report*, 1984. 2

[120] T. Nakamura and M. Asada. Motion sketch: Acquisition of visual motion guided behaviors. In *IJCAI*, 1995. 47

[121] F. Nater, H. Grabner, and L. V. Gool. Temporal relations in videos for unsupervised activity analysis. In *BMVC*, 2011. 41, 72

[122] R. M. Neal. Learning stochastic feedforward networks. In *Tech Report*, 1990. 208

[123] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016. 38

[124] M. Oquab, L. Bottou, I. Laptev, and J.Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014. 69

[125] L. Paletta and A. Pinz. Active object recognition by view integration and reinforcement learning. In *RAS*, 2000. 35

[126] S. Palmer, E. Rosch, and P. Chase. Canonical perspective and the perception of objects. In *Attention and performance IX*, 1981. 27

[127] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 48, 123, 164

[128] F. Perazzi, P. Krähenbühl, Y. Pritch, and A. Hornung. Saliency filters: Contrast based filtering for salient region detection. In *CVPR*, 2012. 36

[129] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2847–2854. IEEE, 2012. 47

[130] L. Proteau and D. Elliott. *Vision and motor control*. Elsevier, 1992. 2

[131] W. Qiu and A. Yuille. Unrealcv: Connecting computer vision to unreal engine. *arXiv preprint arXiv:1609.01326*, 2016. 227

[132] V. Ramanathan and A. Pinz. Active object categorization on a humanoid robot. In *VISAPP*, 2011. 34, 35, 179

[133] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos. In *arXiv preprint arXiv:1412.6604*, 2014. 16, 44

[134] X. Ren and C. Gu. Figure-Ground Segmentation Improves Handled Object Recognition in Egocentric Video. In *CVPR*, 2010. 47

[135] D. Rezende, S. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess. Unsupervised learning of 3d structure from images. In *NIPS*, 2016. 19, 44, 129

[136] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. *ECCV*, 2016. 227

[137] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. In *IJCV*, 2015. 21, 22, 70

[138] M. S. Ryoo and L. Matthies. First-person activity recognition: What are they doing to me? In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2730–2737, 2013. 47

[139] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3d scene structure from a single still image. In *TPAMI*, 2009. 27

[140] B. Schiele and J. Crowley. Transinformation for active object recognition. In *ICCV*, 1998. 23, 34, 35, 175, 179, 183, 184, 185, 187

[141] U. Schmidt and S. Roth. Learning rotation-aware features: From invariant priors to equivariant descriptors. In *CVPR*, 2012. 16, 42, 56, 81

[142] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 72

[143] P. Sermanet, A. Frome, and E. Real. Attention for fine-grained categorization. In *arXiv*, 2014. 36, 37

[144] A. Shafaei, J. J. Little, and M. Schmidt. Play and learn: Using video games to train computer vision models. *BMVC*, 2017. 227

[145] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles, 2017. 227

[146] R. N. Shepard and J. Metzler. Mental rotation of three-dimensional objects. 1971. 18, 128

[147] P. Simard, Y. LeCun, J. Denker, and B. Victorri. Transformation Invariance in Pattern Recognition - Tangent distance and Tangent propagation. In *Tech Report*, 1998. 15, 40

[148] P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, 2003. 15, 40

[149] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2014. 8, 49, 153, 181, 182

[150] S. Soatto. Actionable information in vision. In *ICCV*, 2009. 34

[151] K. Sohn and H. Lee. Learning invariant representations with local transformations. In *ICML*, 2012. 15, 40

[152] K. C. Soska and S. P. Johnson. Development of three-dimensional object completion in infancy. In *Child development*, 2008. 26

[153] K. C. Soska, K. E. Adolph, and S. P. Johnson. Systems in development: motor skill acquisition facilitates three-dimensional object completion. In *Developmental psychology*, 2010. 26, 28

[154] R. Spica, P. R. Giordano, and F. Chaumette. Active structure from motion: application to point, sphere, and cylinder. 2014. 37

[155] J. Stober, R. Miikkulainen, and B. Kuipers. Learning geometry from sensorimotor experience. In *ICDL*, 2011. 31, 32

[156] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multiview convolutional neural netowrks for 3d shape recognition. In *ICCV*, 2015. 195

[157] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998. 198

[158] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 180

[159] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Multi-view 3d models from single images with a convolutional network. In *ECCV*, 2016. 19, 45, 129, 134, 165

[160] A. Torralba. Neurobiology of attention, chapter contextual influences on saliency. 2005. 36

[161] A. Torralba, A. Oliva, M. S. Castelhano, and J. M. Henderson. Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search. In *Psychological review*, 2006. 26

[162] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. In *TPAMI*, 2008. 2, 38

[163] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: a survey. In *Foundations and trends in computer graphics and vision*, 2008. 16, 42

[164] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008. 15, 40

[165] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating the future by watching unlabeled video. In *CVPR*, 2016. 41, 44

[166] J. Walker, A. Gupta, and M. Hebert. Dense optical flow prediction from a static image. In *ICCV*, 2015. 38, 39, 44

[167] B. Wang. Coverage problems in sensor networks: A survey. In *ACM CSUR*, 2011. 38

[168] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *CVPR*, 2015. 9, 15, 16, 38, 41, 72, 80, 85, 112, 123, 124

[169] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 1992. 35

[170] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *NIPS*, 2015. 31, 32

[171] D. Wilkes and J. Tsotsos. Active object recognition. In *CVPR*, 1992. 23, 33, 34, 175

[172] R. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *JMLR*, 1992. 172, 173, 174, 175, 209

[173] M. Wilson. Six views of embodied cognition. In *Psychonomic bulletin & review*, 2002. 2

[174] L. Wiskott and T. J. Sejnowski. Slow feature analysis: unsupervised learning of invariances. In *Neural computation*, 2002. 9, 41, 53, 117

[175] J. Wu, T. Xue, J. Lim, Y. Tian, J. Tenenbaum, A. Torralba, and W. Freeman. Single image 3d interpreter network. In *ECCV*, 2016. 19, 44, 129

[176] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A Deep Representation for Volumetric Shapes. In *CVPR*, 2015. 34, 44, 96, 138, 179, 181, 182, 183, 184, 191, 192, 193, 211

[177] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 17, 59, 100, 103, 117

[178] J. Xiao, K. Ehinger, A. Oliva, A. Torralba, et al. Recognizing scene viewpoint using panoramic place representation. In *CVPR*, 2012. 179, 181, 211

[179] B. Xiong and K. Grauman. Detecting snap points in egocentric video with a web photo prior. In *ECCV*, 2014. 27

[180] C. Xu, J. Liu, and B. Kuipers. Moving object segmentation using motor signals. In *ECCV*, 2012. 47

[181] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015. 36, 37

[182] K. Yamada, Y. Sugano, T. Okabe, Y. Sato, A. Sugimoto, and K. Hiraki. Attention prediction in egocentric video using motion and visual saliency. In *PSIVT*, 2012. 47

[183] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *NIPS*, 2016. 19, 44, 129, 130, 134

[184] J. Yang, S. Reed, M.-H. Yang, and H. Lee. Weakly supervised disentangling with recurrent transformations in 3d view synthesis. In *NIPS*, 2015. 45

[185] X. Yu, C. Fermuller, C. L. Teo, Y. Yang, and Y. Aloimonos. Active scene recognition with vision and language. In *CVPR*, 2011. 34

[186] L. Zafeiriou, M. Nicolaou, S. Zafeiriou, S. Nikitidis, and M. Pantic. Learning slow features for behaviour analysis. In *ICCV*, 2013. 41, 72

[187] A. R. Zamir, T. Wekel, P. Agrawal, C. Wei, J. Malik, and S. Savarese. Generic 3d representation via pose estimation and matching. In *European Conference on Computer Vision*, pages 535–553. Springer, 2016. 125

[188] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 219

[189] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *CVPR*, 2016. 38, 48

[190] Z. Zhang and D. Tao. Slow feature analysis for human action recognition. In *PAMI*, 2012. 41, 72

[191] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014. 219

[192] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014. 2, 38

[193] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. Efros. View synthesis by appearance flow. In *ECCV*, 2016. 19, 44, 45, 129, 130, 134

[194] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. *ICRA*, 2017. 227

[195] W. Zou, S. Zhu, K. Yu, and A. Y. Ng. Deep learning of invariant features via simulated fixations in video. In *NIPS*, 2012. 9, 15, 41, 53, 72, 80, 85, 99

[196] W. Y. Zou, A. Y. Ng, and K. Yu. Unsupervised learning of visual invariance with temporal coherence. In *NIPS Workshop on Deep Learning*, 2011. 41

Vita

Dinesh Jayaraman was born in Chennai, India to Mr. R. Jayaraman and Mrs. J. Padmavathy on December 29, 1989. He went to Kendriya Vidyalaya Malad (Mumbai), The Modern High School (Dubai), SRDF Vivekananda Vidyalaya (Chennai), and DAV Senior Secondary School (Chennai) before receiving a Bachelor's degree in Electrical Engineering from the Indian Institute of Technology Madras (Chennai) in 2011. He then began graduate studies at The University of Texas at Austin, where he has been studying computer vision under the supervision of Prof. Kristen Grauman since December 2012. His research interests span visual recognition and machine learning. In the last few years, he has worked on visual learning and active recognition in embodied agents, unsupervised representation learning from unlabeled video, visual attribute prediction, and zero-shot categorization. During his PhD, he has received the Best Application Paper Award at the Asian Conference on Computer Vision 2016 for work on automatic cinematography, the Samsung PhD Fellowship for 2016-17, a UT Austin Microelectronics and Computer Development Fellowship, and a Graduate Dean's Prestigious Fellowship Supplement Award for 2016-17. In August 2017, he plans to begin a postdoc at The University of California, Berkeley to continue to investigate problems at the intersection of vision and robotics.

Permanent address: 91 Basha Street, Choolaimedu, Chennai-600094,
India

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.