Computational Sensorimotor Learning

Pulkit Agrawal

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2018-133
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-133.html

September 23, 2018

Computational Sensorimotor Learning

By

Pulkit Agrawal

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Jitendra Malik, Chair
Professor Alexei A. Efros
Professor Jack Gallant

Summer 2018

Computational Sensorimotor Learning

Abstract

Computational Sensorimotor Learning

by

Pulkit Agrawal

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Jitendra Malik, Chair

Our fascination with human intelligence has historically influenced AI research to directly build autonomous agents that can solve intellectually challenging problems such as chess and GO. The same philosophy of direct optimization has percolated in the design of systems for image/speech recognition or language translation. But, the AI systems of today are brittle and very different from humans in the way they solve problems as evidenced by their severely limited ability to adapt or generalize. Evolution took a very long time to evolve the necessary sensorimotor skills of an ape (approx. 3.5 billion years) and relatively very short amount of time to develop apes into present-day humans (approx. 18 million years) that can reason and make use of language. There is probably a lesson to be learned here: by the time organisms with simple sensorimotor skills evolved, they possibly also developed the necessary apparatus that could easily support more complex forms of intelligence later on. In other words, by spending a long time solving simple problems, evolution prepared agents for more complex problems. It is probably the same principle at play, wherein humans rely on what they already to know to find solutions to new challenges. The principle of incrementally increasing complexity as evidenced in evolution, child development and the way humans learn may, therefore, be vital to building human-like intelligence.

The current prominent theory in developmental psychology suggests that seemingly frivolous play is a mechanism for infants to conduct experiments for incrementally increasing their knowledge. Infant's experiments such as throwing objects, hitting two objects against each other or putting them in mouth help them understand how forces affect objects, how do objects feel, how different materials interact, etc. In a way, such play prepares infants for future life by laying down the foundation

of a high-level framework of experimentation to quickly understand how things work in new (and potentially non-physical/abstract) environments for constructing goal-directed plans.

I have used ideas from infant development to build mechanisms that allow robots to learn about their environment by experimentation. Results show that such learning allows the agent to adapt to new environments and reuse its past knowledge to succeed at novel tasks quickly.

To my parents – Sadhana & Ratan Agrawal

Contents

Acknowledgments

My dissertation would be impossible without the unconditional support, love, inspiration and encouragement of my parents Sadhana Agrawal, Ratan Agrawal and my sister Ishita Agarwal. They provided me with an invaluable platform to grow from and a philosophy to live by and pursue my dreams.

My grandfather, Shyam Agrawal was very influential in my shaping my interests in science during my childhood. Whenever he visited, we would have endless discussions on physics, chemistry, mathematics, engineering, and politics. We would often go on long walks to chat about my intellectual curiosities at the time. My grandmother, Shail Agrawal was a constant source of encouragement to pursue education. She took excellent care of me.

At a very young age, I found a partner in crime in my cousin Divyesh Gupta, and we spent countless summers building and discussing crazy things. It had a profound impact on me, and the invisible fingerprints of those times have left their mark on this dissertation. I would especially point out encouragement and thought leadership from Dinesh Gupta. I am thankful to have the company of Ranjana Gupta, Kalpana Agrawal, Prashant Agrawal, Neetu Agrawal, Nandini Agrawal, Tanesha Gupta, Divas Agrawal, and other extended members of my family who have always supported me.

I am grateful to Jonathan Manton for introducing me to computer vision during my first internship at the University of Melbourne.

Bhiksha Raj was pivotal in my decision to pursue a Ph.D. I met him first as an undergraduate student at a machine learning winter school organized by him in 2010. We have become friends since then, and I spent a wonderful summer at Carnegie Mellon University working on analyzing group conversations, EEG, and auditory signals. I owe the knowledge of basics of machine learning to him. He spent countless hours coaching me. He has been a pillar of support throughout my Ph.D., and I am grateful for his trust in my intuition and the advise to follow it. I wouldn't be in Berkeley, and this thesis would not have existed without his oversight. I would also like to thank Carolyn Rose and Rita Singh for the wonderful time at Pittsburgh.

This dissertation is the culmination of the freedom of running my Ph.D. as my *start-up* offered by my advisor Jitendra Malik. He ensured I had complete intellectual

freedom and stood like a giant wall to support me. I could not have wished for a better mentor. Jitendra is an ocean of knowledge: any research idea I took to him for discussion was met with an entire lineage of references often dating back to the early 20^{th} century. From him, I learned how to think about a problem at the big picture level, identify an appropriate line of attack and then concretely implement the solution. He helped me realize, that in research, many times the *devil is in the details.* He was instrumental in coming up with names for my various research papers (including the title of this thesis), influenced my research directions and from him I learned a great many things such as: how to choose research problems, how to be extremely pragmatic and judge every situation/idea in an unbiased way and a lot more. I am incredibly grateful for his patience and his trust in me for letting me pursue my own research projects and disappear for months. It has always been a great pleasure to debate and discuss research ideas with him.

I found a great friend and mentor in Alyosha Efros. There were multiple days I went into the lab, only in the hope that I would get to spend time with Alyosha. At the end of my Ph.D., it became a habit to drop by his office and chat about research or other things about life. He has been a pillar of support, very encouraging of all crazy ideas and always had an infinite amount of time to discuss anything, even at 3 am in the night. Alyosha was the life of SDH 7th floor when we were there. He would often ignite controversial discussions, and some of our best ideas came in post-mid-night chats while savoring dark chocolates. I learned from him how to communicate research ideas and identify the crux of a problem, how an okay job is very different from a great job, and he helped me to be focused on core research problems and avoid fluff. I have not learned more about research and life from anyone but Alyosha during my Ph.D. I was very fortunate to spend time in his company!

I spent a significant portion of the first two years at Berkeley in Jack Gallant's lab. His lab meetings were among the most intellectually stimulating experience at Berkeley. In his lab, I learned that *the only truth in science is questions.* It gave me a perspective on how to think deeply about a problem and tackle it scientifically. Jack taught me how to write a research paper, how to formulate experiments for testing various explanations of the results and quickly iterate. Through conversations with him, I got an insight into how the academic system functions. In him, I found a mentor for life.

I thoroughly enjoyed time spent at the Redwood Center of theoretical neuroscience and especially with Bruno Olshausen. I thank him for introducing me to the world of computational neuroscience. We often discussed whether it is possible to create an AI system with the same capabilities as a *fly.* These discussions forced me to think about what intelligence means. From Bruno, I learned how to express ideas in mathematics and to think about what an equation means/implies. It is beautiful

how simple theoretical concepts can explain a lot of experimental observations. I am grateful to Bruno for helping me get a taste of it!

I greatly benefited from Pieter Abbeel's class on Advanced Robotics. I am thankful for him for all the discussions during my Ph.D., providing resources and time to help me get started in robotics and supporting me throughout my time at Berkeley. Pieter is an inspiration. .

I thank Trevor Darrell for all the support, perspective on research problems and ideas. From him I learned how to make things happen given the institutional constraints; picked up advice on how to manage research groups and navigate the academic/industrial eco-system.

My research interests align closely with Abhinav Gupta. Our discussions have been fruitful, and I am grateful to him for his firm support and encouragement to pursue self-supervised robotics. His papers have profoundly influenced me and research in self-supervised robotics would be in dark-ages without his contribution.

I am grateful to Josh Tenenbaum for hosting me in his lab for a couple of weeks in July 2017. With him, I had deep, long and invigorating discussions on Artificial Intelligence that forced me to formulate my ideas more rigorously. He was very helpful in this process. We had many fun dinners, and I learned a lot about what we know about the intelligence of human babies and together we contemplated on good benchmark problems for AI. He played a critical role in introducing me to the AI ecosystem at MIT.

If Joao Carreira was not in Berkeley in 2015, I might have quit the Ph.D. program. Joao remains my favorite postdoc for a good reason – he was immensely supportive both intellectually and emotionally. He helped me write my first papers in computer vision, was always full of crazy ideas and it's still a pleasure to be in his company.

I am very grateful to Sergey Levine, whom I met while he was a post-doc at Berkeley. He helped me transition into robotics. I learned a lot from him about control, simulators and reinforcement learning. I found the technical rigor that Sergey brought to all our discussions to be invaluable, and we spent countless hours chatting about various topics. Sergey helped me run independent projects by playing a pivotal role in recruiting many amazing undergraduate students such as Ashvin Nair, Dian Chen and Fred Shentu whose contributions have been critical to this dissertation.

I worked very closely with Deepak Pathak during the last two years of my Ph.D. In terms of research, these were the most productive years of my student life. I thoroughly enjoyed countless amazing discussions about all aspects of artificial intelligence including learning loss functions, curiosity, hierarchical reinforcement learning, imitation, segmentation, robotics, constrained optimization methods, generative models, self-supervised learning and the list goes on an on. Our discussions would

often run up to 3-4am in the night, turn philosophical, and we would inevitably be the last two people hanging out on the 7th floor of SDH (sometimes along with Alyosha). In addition to research discussions, we enjoyed many delicious Indian meals together, gelato bets and random ramblings about life. My Ph.D. experience is incomplete and would have been very different without having Deepak as a peer.

With Mayur Mudigonda I had the most number of dinners discussing science. I met him in my first semester at Berkeley, and he has been a great friend, collaborator, and supporter. Our discussions played a significant role in biasing my interests in studying sensorimotor and active learning systems. We titled our first co-submission as *"Learning Simultaneous Sensory and Motor Representations (SESEMO)"*. We were too naive at the time to get it published, but thoughts developed in that paper ended up occupying a significant part of my Ph.D. I have continued to have stimulating discussions with Mayur, and he has colored my time at Berkeley in artistic ways.

With Shiry Ginosar I pursued some of the most unusual ideas. Alas! None of them came to fruition; nevertheless, I enjoyed the time spent with her. Her clarity of thought is impeccable. I thank her for all the discussions and the time.

I thank Brian Cheung for all the thought-provoking discussions during our interactions over the past six years. He was one of the few people who was already doing deep learning before it took off.

I thank Panna for the wonderful collaboration on forecasting sports moves. I learned a lot from her about sports in the United States among other things.

I thank Rachit Dubey for the enlightening discussions about cognitive science, curiosity and artificial intelligence. It was a pleasure collaborating with him.

I had great discussions with Shubham Tulsiani about the utility of 3D representations for control, extrapolative generalization, and many other applications. Shubham sat on the desk next to me for a substantial part of my Ph.D. and always got me thinking about something or the other. I was lucky to have him as a labmate.

I am grateful to be part of Jitendra's research group. In the initial years of my Ph.D., I was lucky to have Jon Barron, Bharath Hariharan and Saurabh Gupta around. Not only were our interactions intellectually stimulating, but I am also thankful to them for helping me with much mundane stuff such as running jobs on the cluster and navigating other non-research related Ph.D. stuff. I also had the opportunity to closely interact with many of Jitendra's and Alyosha's brilliant students: Subhranshu Maji, Chunhui Gu, Jon Barron, Bharath Hariharan, Georgia Gkioxari, Saurabh Gupta, Shiry Ginosar, Tinghui Zhou, Junyan Zhu, Carl Doersch, Panna Felsen, Shubham Tulsiani, Ke Li, Weicheng Kuo, Ashish Kumar, Edmund Ye. It is undoubtedly true that in a Ph.D. you learn the most from your labmates.

I had a fantastic opportunity to work with some remarkably brilliant undergraduate students, Jacob Huh, Ashvin Nair, Dian Chen, Fred Shentu, Jeffrey Zhang and

many others. Not only did we chase our intellectual curiosities, but they challenged my understanding of the field and research questions in a way very different than my peers. I am grateful for their contributions in making this thesis possible. With them, I was able to run the *"Improbable AI lab"*, which was among the most fun things I did at Berkeley.

I was lucky to be surrounded by exceptional Ph.D. students in the Berkeley eco-system. In addition to the people mentioned above, I would like to thank Alex Huth, Dustin Stansbury, Michael Oliver, James Gao, Natalia Bilenko, Anwar Nunez-Elizande, Tuomas Haarnoja, Avi Singh, Abhishek Gupta, Chelsea Finn, Karthik Narayan, Judy Hoffman, Jeff Donahue, Sergey Kayarev, Yangqing Jia, Evan Shelhamer, Bradley Stadie, George Netscher, Daniel Aranki, Somil Bansal, Nick Boyd, Michael Chang, Yusuf Erol, Yuansi Chen, Roel Dobbe, Mohit Bansal, Greg Durrett, Carlos Florensa, Animesh Garg, Erin Grant, Allan Jabri, Forrest Iandola, Gregory Kahn, Frederik Ebert, Jeff Mahler, Michael Lasky, Sanjay Krishnan, Alex Lee, Parsa Mahmoudieh, Taesung Park, Vitchyr Pong, Fereshteh Sadeghi, Nihar Shah, Rashmi Vinayak, Adam Stooke, Eric Tzeng, Richard Zhang, Ryan Zarcone, Tijana Zrnic, Jessica Hamrick, Amik Singh and many others. I have learned a lot from my interactions with them.

Berkeley Artificial Intelligence Research (BAIR) attracted excellent post-doctoral researchers. I had the pleasure of working/interacting with Pablo Arbelaez, Ross Girshick, Joao Carreira, Philipp Krahenbuhl, Katerina Fragkiadaki, Christian Hane, David Fouhey, Andrew Owens, Sergey Levine, Mark Lescroart, Dinesh Jayaraman, Roberto Calandra, Aviv Tamar, Amir Zamir, Phillip Isola, David Held, Sachin Patil, Angjoo Kanazawa, Yong Jae Lee and many others with whom I crossed paths with. At one point in my Ph.D., on every paper, I was a co-author with a different post-doc. Closely working with many of them helped me imbibe diverse perspectives on research methodologies and problems.

I spent an exciting summer at DeepMind where I developed a foundation in reinforcement learning, and it was an incredibly stimulating place to pursue general artificial intelligence. I found good friends and partners in crime in the AI dream. I owe my knowledge about physics simulators (especially Mujoco) to Tom Erez and Yuval Tassa. Volodymyr Mnih, Nicolas Hees, and Timothy Lillicrap were great mentors in helping me figure out the nuts and bolts of reinforcement learning; I had in-depth conversations with Nando Freitas, Tejas Kulkarni, Jon Scholz, Peter Battaglia, Misha Denil, Andrew Zisserman, Andrei Rusu, Raia Hadsell, Jacob Menick and many other colleagues. My time at DeepMind had a significant impact on the course of my future research.

I thank Chirs Meyers, other staff at Berkeley's invention lab, Ankur Agrawal and Dian Chen for helping me design and construct the arena for robot experiments.

Nothing in the Berkeley AI lab moves without Angie Abbatecola's nod. I was first introduced to her as Jitendra's assistant, but with time she rose up to managing the logistics entire BAIR. She along with her team comprising of Lena Lau Stewart and others ensured that I was adequately taken care of and no hurdles bothered me. Angie's kindness deeply touched me. I am also thankful to SDH building manager Domenico; Erin Skelly at the Berkeley International office; Shirley Salanio, Audrey Sillers, Susanne Kauer and other staff at the Department of Electrical Engineering and Computer Sciences and the greater University ecosystem for all the help.

For four years, I lived at the International House (I-House). Residing in I-House was an enriching experience, and I made many friends. I am thankful for many people I met at I-House for making my Ph.D. experience even more special.

The Ph.D. experience is only partly about the time spent in the lab. I found close friends in Nitesh Mor, Prachi Shah, Ivana Stradner, Emily Campbell, Ali Momin, Vishwanath Bulusu, BVV Srirajdutt, Anubhav Singla and many others outside of my research circle. I thank all of them for their support, encouragement, and understanding. I would especially like to thank Prachi for her support in the final year of my Ph.D.

I was lucky to have great teachers and mentors at IIT Kanpur: Ajit Chaturvedi, S.C. Srivastava, Amitabha Mukerjee, Manik Das, Surendra Baswana, Laxmidhar Behera, Aloke Dutta, A.R. Harish, S.S.K Iyer, Anil Joshi, Ram Potluri, Naren Naik, R.K Bansal, S. Umesh and Joseph John. In addition, I has the pleasure of spending time with Kshitij Deo, Biplab Deka, Sourabh Sankule, Kartik Venkat, Ashish Agrawal, Rahul Agrawal, Shantanu Agrawal, Shruti Agrawal, Ankur Agrawal, Anubhav Singla, Vikram Rastogi, Geetak Gupta, Mainak Chowdhury, Nitish Srivastava, BVV SrirajDutt, Sourav Khandelwal, Raghav Khanna, Gaurav Bhatele and many other fellow students who profoundly influenced my thinking and my scientific interests. A lot of good friendships were made I would like to specially mention the support of the science and technology council and the water polo team at IIT Kanpur. Our aquatics coach, Vivek Rao Vadi imparted invaluable life lessons.. ‘

Finally, I am indebted to my teachers in both my primary and high-school who shaped my curiosity and were of immense support right from the beginning. I am grateful to *Pathak Sir* (Mr. Manoj Pathak) for introducing me to music at a young age. My instrument has been a good companion throughout my Ph.D.

Chapter 1

Introduction

Humans made machines to assist them to perform either mundane tasks such as crunching numbers, washing clothes or physically challenging tasks such as harvesting wheat or for enabling new possibilities such as quickly transporting themselves and goods from one location to other. The guiding design principle behind building these machines has been the efficient and precise execution of one very particular task over and over again. Quite unsurprisingly, all the machines that we know of today are very good at performing one very specific task and utterly incapable of executing any other task. For instance, a robotic arm that can bolt a car door to the body will completely fail to bolt together two parts of a toaster or even the door of another car model. As humans, we understand the notion of bolting, and therefore it is trivial for us to generalize and (if possible) bolt almost any given object parts together. More generally, humans seem to possess what is colloquially referred to as "common sense" knowledge that enables them to perform a "general set of humans" task. Developing machines with such abilities has proven to be extremely challenging and is one of the core topics of research in Artificial Intelligence. In this thesis, I will present some ideas on how to go about building machines that can acquire "commonsense" knowledge about their environment via self-supervised interactions and minimal reliance on expert teachers.

1.1 Today's Artificial Intelligence

The past fifty years have seen impressive progress in artificial intelligence (AI). Today, there exist systems that surpass humans at playing sophisticated games such as chess [1], GO [2], the ATARI video games [3], can precisely segment objects in complex natural images [4], accurately identify the type of objects in images [5, 6], translate between languages [7], robots that can perform impressive maneuvers such

as back-flips or play table-tennis [8] and numerous robots automating industrial production lines [9]. In contrast to man's historic journey to the moon, best described in Neil Armstrong's words, *"that's one small step for [a] man, one giant leap for mankind."*, the great strides in artificial intelligence are only a small leap in our quest for understanding and creating truly intelligent machines (see [10] for a discussion). A closer look at current AI systems reveals why this is the case.

Game AI: Many of our successes have come in the domain of games such as Chess, GO, Checkers, Poker, Backgammon or the ATARI video games [1–3, 11–13], which run in simulation and is therefore possible to play the game millions of times to find a winning strategy. The state of art algorithms work under strong assumptions such as *closed world* (i.e., complete knowledge all possible game states and moves), rely on random search for finding good strategies which is impractical for most real-world problems and the learned strategies are extremely specific to one particular setup. For instance, a machine that can play chess on a 8×8 board is utterly incapable of playing on a 9×9 board, let alone playing a different game such as *tic-tac-toe*.

Computer Vision: In the recent years, computer vision underwent a deep learning revolution starting from the unprecedented performance of a deep neural network at the task of classifying images into one out of thousand categories [5] of the Imagenet dataset [14]. Soon many more neural network architectures were proposed [6, 15–17] and the best systems reached close to human performance at identifying one of thousand image categories in the Imagenet dataset [14]. Very significant performance improvements were reported on almost all computer vision tasks including the core problems of object detection [18–20], segmentation [4, 21] and human pose-estimation [22–24]. In addition to advances in computing power, these performance improvements relied on large datasets containing millions of human annotated images [25, 26]. In summary, today we have techniques that are adept at solving the mapping problem from images to a set of labels given enough annotated training data is provided.

While many attempts have been made to build computer vision systems that can transfer the representations learned by solving one task to another [18, 27–31], thousands (if not millions) of manually annotated examples are still required to accurately identify new concepts such as a person lying down on bed or person sitting. Our deep learning systems today have no abstraction of person, bed or the concept of "lying down". They appear to be making predictions based on some rather lower-level image statistics. A different line of work based on generative models [32, 33] is promising but is yet to be scaled to real-world visual data.

Robotics and Control: The story is not that different in the world of robotics where we have built systems that are very good at executing one precise task. Today there are robots like *Honda Asimo* or *Atlas* by Boston Dynamics that can walk on two legs, perform impressive maneuvers such as back-flips, and there is an unprecedented use of robotic automation in manufacturing and other industries. However, the performance of the best systems in the DARPA Robotics Challenge is a reminder that today's robots are not robust and do not have even a remote resemblance to the common sense abilities of humans.

In summary, today we have created AI systems that either require large amounts of human supervision, are very specific to one particular task, operate under closed world assumptions or work in simulated environments. How to create AI systems that can work in the real world, do not require complete knowledge of their environment (i.e., open world), are robust and don't rely heavily on human supervision is an open question.

1.2 Problem Formulation

Consider an agent that is capable of acting and observing its environment through possibly multiple sensory modalities such as vision, audition, touch, smell, etc. In general, we will assume that sensory observations might not contain enough information to entirely infer the full state of the environment (i.e., sensing abilities of the agent are possibly limited). The environment is also assumed to be large enough that it is not possible to completely observe it any one point in time. Our agent's environment also consists of other agents that are either friendly, adversarial or neutral. Our agent has no direct access to the state of the other agents but can observe them if they are in the field of view. Lets further assume that the environment is possibly stochastic and mostly stationary, except that the behavior of all agents is non-stationary and is subject to change with time.

The task of designing the agent is to come up with a mapping from observations to actions (called as a *policy* (π)), so that the agent can complete a task assigned to it. Any given point in time, let the agent's knowledge about its environment be k_t and what knowledge it starts off with (i.e., k_0) is a free variable to decided by the designer. The task provided to the agent might be solvable using k_t or the other agent might need to augment its knowledge by either exploring the environment or by communicating with other agents. Exploration is defined merely as the process of taking actions and observing the consequences. The strategy for exploration could either be pre-defined or the agent can come up with a strategy given its current knowledge about the world. It, however, must be the case that any task posed to

the agent is solvable – i.e., there exists a strategy to accumulate knowledge using the sensory observations of the agent and communication with other agents that is sufficient to solve the task. Note that we are not assuming that there already exists a language for communication. Assuming the existence of any language or leaving it to the agent to come up with its own language is again a design choice. The agent will only be evaluated by its success at solving the tasks given to it.

Grounding the abstract setup mentioned in the previous paragraph immediately presents multiple questions: (a) What is the initial knowledge that should be designed into the agent?; (b) How does the agent represent its current knowledge (k_t)?; (c) How does the agent update its knowledge given new observations?; (d) How are the tasks communicated to the agent? I will start by a possible answer to (d) and use this to analyze the dominant paradigms for building AI systems today.

1.2.1 Task Communication

A task is defined to be a feasible manipulation of the state of the environment. There are many ways to communicate the task:

1. **Goal Observation:** The observation corresponding to the desired state that the agent must achieve. For instance, if the task is to re-arrange objects on a table, the agent might be provided with a (target) image of objects in the target configuration. While this formulation is sufficient for some family of tasks, it has two drawbacks: (a) for some tasks such as pouring water from one cup into a second one, the initial and final image might look identical if the cups are not transparent and the image is taken from the side; (b) the environment needs to be set into the target configuration to generate the target configuration which can be cumbersome and defeats the purpose of designing an autonomous agent for completing the task.

2. **Goal State:** Instead of providing the observation, communicate the underlying ground truth state of the environment. While this solves the problem 1(a) mentioned above, it leads to two other problems: (a) Defining how to characterize the “state" of the environment is non-trivial. For e.g. in the example of pouring water described above, the state might be $y_2 = 4cm$, where y_1, y_2 are water-levels in both the cups. Not only is this representation very specific to this task, but it also needs to be communicated to the agent what water-level actually means (i.e. the *symbol grounding problem* [34]); (b) it requires an oracle to identify the state of the environment (i.e. *system identification* problem). Such oracle is not available in the real world.

3. **Observation of the task demonstration:** Instead of just providing an observation depicting a goal state, providing a sequence of observations is a more expressive way to communicate the task. While it can potentially address the problem mentioned in 1(a) by making the complete action of pouring observable, it intensifies another problem. If for instance the agent observes the video of an expert pouring the liquid, it needs to infer whether it is the locations of the expert's hands that is part of the task or whether it is the fact that liquid is transferred from one cup to another. If the agent was to match the task in a pixel-perfect manner it would be obsessed to exactly match the demonstrated trajectory of the hand movement. However, if an adult human were to observe a similar demonstration, she would immediately infer that the task is to pour the liquid and the exact trajectory followed by the hand in the demonstration is inconsequential. Making such inferences about what are the relevant and irrelevant parts of the demonstrations is non-trivial and is the core problem studied in *inverse reinforcement learning* and *inverse optimal control*. Note that this problem also affects 1, but to a lesser extent.

4. **Sequence of states from a task demonstration:** Same as the point above, but instead of observations the agent has access to the sequence of ground-truth states of the environment. The benefits and disadvantages of this way of expressing the goal can be easily inferred from points 2 and 3 above. Note that the problem of inferring the relevant parts of the demonstration can be bypassed in this scenario by choosing the state-representation to not include the location of the demonstrator's hands. However, such choices of state-representation are extremely task-specific and state-representation requires access to an oracle which is impractical in the real world.

5. **Observation Sequence + Actions:** This is similar to point 3 but in addition to observations of the demonstration, the agent is also provided with the sequence of actions that would complete the task. In general such a description of the goal can be achieved when the expert either physically moves the manipulators of the agent/robot to perform the task (i.e. kinesthetic teaching) or tele-operates it. Problems of *learning from demonstration (LfD)* or *behavior cloning* are concerned with using either a single or multiple demonstrations to learn a policy for completing the task. Note that a single demonstration is generally not enough, because simply replaying the actions might not sufficient for reaching the goal because the initial state of the agent might be different or something in the environment might have changed since the time the demonstration was recorded. Note that this formulation of expressing the task attempts to bypass the problem of explicitly inferring what are rele-

vant/irrelevant parts of the demonstration (see point 3) by directly providing actions to the agent. However, an agent with finite resources might not be able to perfectly learn the mapping from observations to actions for every observation in the demonstration. It might need to identify what are important parts of the demonstrated trajectory and focus its resources there. In such a realistic setting, the problem of inferring the intent of the demonstration is not bypassed. Additionally, this process of expressing the task is extremely tedious and the strategies learned by the agent to are likely to be very specific to the end task.

6. **State Sequence + Actions:** Similar to above, but instead of observation/action trajectory, the agent is assumed to have access to the ground truth state/action trajectory.

7. **Formal/Natural Language Instruction:** It is very tempting to express the task in terms of a language instruction such as *"pour liquid into the second cup"*. While language removes some of the issues of conveying the intent mentioned in points 3, 5, it doesn't get around the critical *symbol grounding problem* described in point 2 above. If we are able to develop a grounded representation of symbols, expressing goals in form of language is very expressive and opens doors for easily expressing and performing complex tasks by making use of compositional structure of language.

8. **Abstract (possibly learned) representation of the Goal:** Similar to points 1, 3 above but with the difference that goals are expressed in an abstract (feature) representation of the goal observation.The reason for expressing the goal in an abstract space is to represent things that are relevant to completing the task and ignoring the rest. For instance, in the running example of pouring liquid, it is not necessary to represent the hand of the demonstrator, but only the notion that liquid was poured from cut A into B. How to come up with or learn such representations is unclear. There are multiple lines of work of unsupervised learning, self-supervised learning or using pre-trained features (say from a network trained on Imagenet classification task) aimed at learning good representations of the data. However, what is *good general* representation is a tricky question.

   Unsupervised/self-supervised learning are closely related to each other and are useful paradigms under the premise that there is either insufficient amount of data or supervision available to directly optimize for the target task. The general idea is to learn features by optimizing a proxy task for which large amounts of data can be obtained and hope that these features will be useful

for one particular or a family of target tasks. Yes! You read it correctly, one merely *hopes* the features are going to be useful. One reason for simply being hopeful is that currently there are no good ways of choosing proxy tasks based on the set of target tasks that the features are being learned for. Research in this area typically relies on the intuition of the researcher to come up with different proxy tasks and experimentally validate them on a battery of target tasks.

One very popular such proxy task is to reconstruct the data itself. First, the data is transformed into a representation (say by processing it with multiple layers of neurons), and this representation is used to reconstruct the data. To prevent the trivial solution of identity mapping, this representation is constrained to be either low-dimensional [35–37] or sparse [38] or sometimes the different dimensions of the representation are forced to be independent [39,40] or change slowly [41, 42] or satisfy some other constraint [43]. Let's understand this with an example: suppose that we want to learn good features for recognizing what objects are present in an image. We set up the proxy task of taking a large number of images and reconstructing them by constraining the different dimensions of the representation to be independent of each other. A simple model of image generation process has four independent parameters: (a) the *shape* of the object; (b) the *albedo* of the object; (c) the angle at which the light impinges the surface of the object (i.e. *direction* of light) and; (d) the *spectrum* of light frequencies falling on the surface. One might hope that by imposing independence on reconstruction, one can learn the four factors mentioned above. However, it is improbable that this would be the case due to two reasons: firstly, there are multiple combinations of independent factors that can explain the data equally well; secondly, these factors are estimated from data captured in a structured world. For instance, there exist correlations between shapes of objects and their colors. E.g., spherically shaped fruits are more likely to have shades of yellow/orange/red/green instead of blue. This example suggests that from a finite amount of data it might not be possible to infer the truly independent factors. The above discussion suggests that while reconstruction can find a set of features, it is unknown what that set of features would be and whether that set would be useful for the target task at all. It is possible therefore that while reconstruction based methods have done very well on simpler datasets such as MNIST [44] and CIFAR [45], they are yet to find success on real-world data.

The recent wave of feature learning methods have posed proxy tasks different from reconstruction and have used the name “self-supervised" learning to

differentiate from the conventional notion of unsupervised learning by clustering/reconstruction. Some representative works include learning features by predicting relationship between different parts of the image [46, 47], correlating ego-motion with visual inputs [48, 49], tracking objects [50, 51], predicting sound from videos [52], using motion [53], inpainting [54], adversarial feature learning [55], predicting color from grayscale images [56] etc. While these works have enjoyed more success than the reconstruction based unsupervised learning works at their usefulness at target tasks, it is far from clear that they provide a good generic representation. The choice of a particular proxy task inevitably dictates the what target tasks would benefit from the learned features.

On the other end of feature learning methods spectrum is feature learning via supervision from manual annotations. One of the most successful instantiations of this idea is to learn features by optimizing for image classification accuracy on the Imagenet challenge and then using these features for other recognition tasks [18, 27, 57, 58]. However, as mentioned above the nature of the task (i.e., image classification) will inevitably bias the utility of these features for different tasks. For instance, to classify what objects an image contains the learner might find representations that are invariant to the pose of the object. Such representations are unlikely to be of much use for the manipulation task of rearranging objects in a target configuration.

The above discussion summarizes the difficulty of pre-determining a useful abstract representation for specifying goals. This difficulty also manifests as a central problem in hierarchical control/hierarchical reinforcement learning in the form of how should sub-goals be represented. It seems that instead of attempting to find a universally good representation, it is better to construct an agent that through the process of solving N tasks learns a representation that is sufficient to express the $(N+1)^{th}$ task.

9. **Reward Function:** Instead of expressing the goal as a description of a desired state that has numerous challenges as pointed above, reward functions is a seemingly ingenious way to define tasks where the agent receives a high-score if it reaches a desired state. In this setup, the agent is never communicated *what* a desired state is, but is simply rewarded *when* it stumbles upon that state.

   While at first it appears we might have bypassed the problem of state description, but it has simply been re-packaged into a different problem of reward function design. Consider the running example of pouring the liquid, and let the r_1 be a reward function that provides a reward of $+1$ if $200ml$ of liquid has been poured into cup B and a reward of 0 otherwise. When the agent starts off, it

is clueless about the task and might therefore simply perform random actions. It is extremely unlikely that by random chance it would pick up the first cup and pour at least $200ml$ of liquid into the second cup. This is referred to as the *exploration* problem.

One way to remedy the exploration problem is to modify the reward function (i.e. "shape" it) to the increase the likelihood the agent obtaining some reward by random chance. For instance, let the reward function r_2 be the amount of liquid poured into the second cup. Now this makes the task easier, because the agent immediately gets a reward when it starts pouring in contrast to waiting to pout $200ml$ before getting any reward. However this is still a tough exploration challenge and to make the task easier, the reward designer might also add an additional reward term that penalizes the distance of cup A from B. While at first it might appear this should make the task easier, but there are multiple considerations: (a) the optimizer might find a local optima and not explore beyond reaching the second cup; (b) the distance between cups need to be properly specified. For instance, if the reward is to align the center of cup A with cup B, then if cup A is tilted to pour then the liquid will fall out of B. The shaping reward possibly needs to align the edge of cup A with center of cup B. In general it is non-trivial to design reward functions to obtain desired behaviors [59–61].

In addition to the issues mentioned above, it is non-trivial to measure rewards. In the running example, one would need to develop a system to infer how much liquid has been poured into the cup and also to detect how far is cup A from cup B. This entails detecting from sensory observations what is the state of the environment, which is similar to describing the task as a sequence of observations (i.e., point 3). In summary, while expressing tasks in the language of reward functions is a viable option, it suffers from many of the same issues as other ways of expressing tasks discussed above.

10. **Other task-specific representation:** It is possible to construct alternate task descriptions such as providing the difference in observations/states from the initial instead of providing the desired state or some other niche way of expressing a specific task. However, in general any such representation can be reduced to one of the cases mentioned above.

The discussion above suggests that the main problem in communicating goals if of measuring progress towards the task from sensory observations. In points 1, 3, 5 it was hard to identify relevant parts of the observation, in points 2, 4 it was non-trivial to define the state, in point 7 it was unclear how to ground language into

observations and in point 8 how to learn or construct the appropriate representation of the observation. Resolving this issue is one of the core problems of building sensorimotor agents.

It is natural to be curious how do we humans get around this problem. The simple answer is that we are not optimized for performing a single task of pouring liquid from one cup to another. Instead, our experience of multiple tasks informs us that the exact trajectory of the demonstrator's hand is irrelevant and what matters is that the liquid was poured into the second cup. It seems that we have a prior over the family of possible tasks that we bring to bear to infer task-relevant information. How much of these priors are built in or learned is a separate question that we will briefly touch upon later in the thesis.

Whatever our belief might be regarding how much knowledge to build in, it is clear that at the starting of life the first unicellular organisms did not have any knowledge that even remotely resembles the kind of common sense knowledge we have today. How did evolution resolve the many problems we discussed above? One thing in favor of evolution is that organisms, the environment, and the tasks co-evolved. It was never the case that an organism started tabula-rasa and had to make sense of high-dimensional sensory observations of vision, touch, etc. The primordial organisms were attracted to sunlight or certain chemicals, and as their sensory systems increased in complexity with evolution, these complex systems were probably biased to replicate the same responses as the simpler versions of themselves. Such biases naturally restrict the exploration space which in turn might have helped the organism infer its more complex sensory signals much more easily as opposed to starting from scratch. In formal terms, the learner acquires a prior over the space of functions that it searches over (i.e., inductive bias) while solving more straightforward tasks that constrains the space of functions it searcher over for more complex tasks/sensory inputs.

It might, therefore, be the case that problems encountered in communicating tasks are a byproduct of the experimental setup of communicating a single complex task in a rich environment to an agent in a tabula-rasa state. Lets first review the dominant approaches for learning sensorimotor skills to analyze whether and how they resolve these issues.

1.3 Learning Sensorimotor Behavior

I will briefly discuss the dominant paradigms for learning sensorimotor behavior through a few sample tasks shown in Figure 1.1. In the first task, the agent is required to rearrange objects kept on a table from their current to target configuration shown

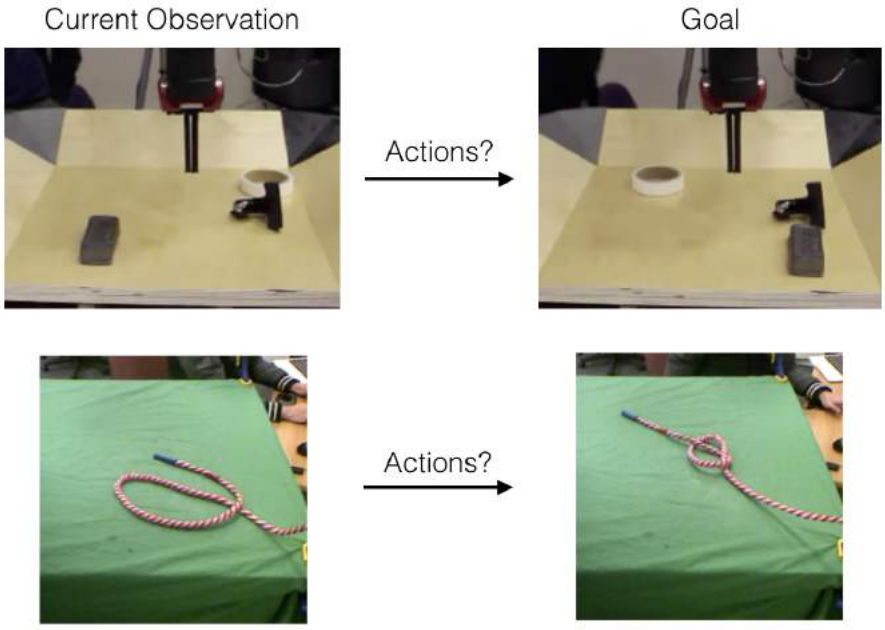


Figure 1.1: *Sample Tasks:* The agent visually observes the world and is tasked to either rearrange objects into the target configuration shown in the goal image or manipulate the rope from its current shape into a knot.

in the goal image. The agent only has access to visual observation. In the second task, the agent is required to re-arrange the rope from its initial shape into a knot.

1.3.1 Reinforcement Learning (RL)

The objective in reinforcement learning is to find which action to take in every state for maximizing the sum of future rewards. Let the current state/observation be x_t and let a_t be the action taken by the agent in response to x_t. The mapping between states and actions (i.e., policy) can be found by optimizing,

$$\max_{\theta} \mathbb{E}_{\pi(x_t;\theta)}[\Sigma_t r_t] \tag{1.1}$$

where θ denotes the parameters of the policy.

Lets consider the task of knot-tying (see Figure 1.1). Suppose the reward function takes the following form: $r_t = 1$ if the rope is configured into a knot and is set to 0 otherwise. Usually, at the start of training, the parameters θ are initialized with random values resulting in the agent executing random actions. Until the agent ties the knot atleast once by random chance (i.e., by taking a sequence of random actions), it will receive no reward and therefore its policy will remain unchanged (i.e. random). In this scenario, it is highly improbable that the agent will tie knots by

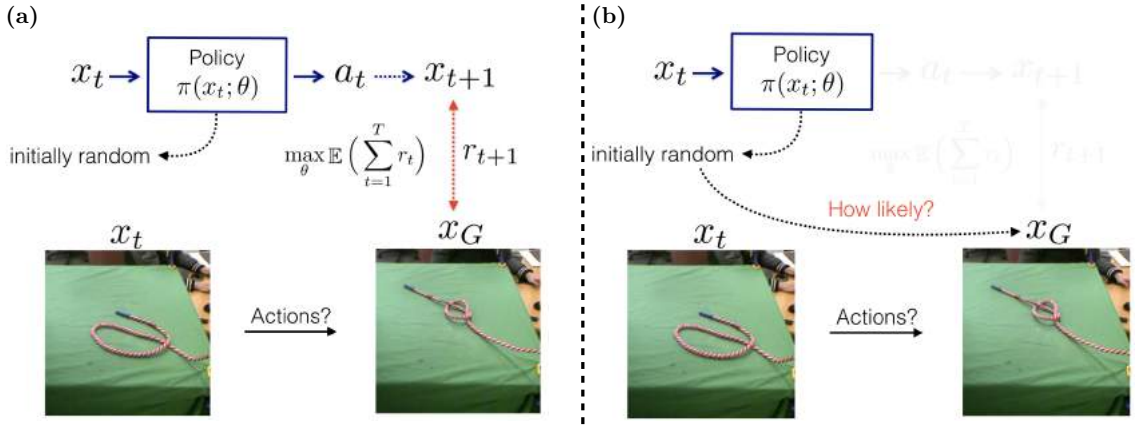


Figure 1.2: (a) In *Reinforcement Learning* the objective is to determine a mapping (i.e. a *policy*) from state (x_t) to action (a_t) that maximizes the sum of rewards ($\Sigma_t r_t$). The agent is rewarded when it reaches the goal (x_G). (b) At the start of training, the parameters θ are initialized to random values resulting in random actions. It is highly improbable that the agent will tie a knot by a taking a sequence of random actions and in absence of rewards its policy will continue to be random. The RL agent is very likely to fail at this task.

random chance. This is known as the exploration problem. One reason for success of RL in games/simulated environments is that one can simply wait for tens to thousands of millions interactions in the hope that the agent will stumble upon the reward. This is infeasible in real world setups.

Secondly, while in games the reward is available from the game itself, it is unclear how such reward can be obtained in real world setups. For instance, in order to calculate whether the rope is configured into a knot, one would either rely on a human expert that provides reward to the agent or build a computer vision classifier that seperates knots from other configurations. Relying on humans for measuring rewards doesnot scale and is tedious. Building computer vision system is very tedious too, because suppose if we did manage to gather enough data to train a classifer for knots, the moment we change to a different task (e.g., configuring the rope into a "S" or "L" shape), the data and training of the classifier would need to be repeated.

Thirdly, the policy learned for knot-tying will be very specific to this task and will fail to generalize to even configure the same rope into different shapes. One potential reason is optimization for a single task and there is no real incentive for the agent to learn any generalizable behaviors/skills.

In summary, RL algorithms of today require a ginormous number of interactions, learn policies that are specific to one task and it is unclear where do rewards come

from. These issues pose a challenge for scaling RL to real world environments.

1.3.2 Learning from Demonstration/Imitation Learning

he current dominant paradigm in learning from demonstration (LfD) [62–65] requires the expert to either manually move the robot joints (i.e., kinesthetic teaching) or teleoperate the robot to execute the desired task. The expert typically provides multiple demonstrations of a task at training time, and this generates data in the form of observation-action pairs from the agent's point of view. The agent then distills this data into a policy for performing the task of interest. Such a heavily supervised approach, where it is necessary to provide demonstrations by controlling the robot, is incredibly tedious for the human expert. Moreover, for every new task that the robot needs to execute, the expert is required to provide a new set of demonstrations.

Instead of communicating *how* to perform a task via observation-action pairs, a more general formulation allows the expert to communicate only *what* needs to be done by providing the observations of the desired world states via a video or a sparse sequence of images. This way, the agent is required to infer how to perform the task (i.e., actions) by itself. In psychology, this is known as *observational learning* [66]. While this is a harder learning problem, it is a more interesting setting, because the expert can demonstrate multiple tasks quickly and easily.

An agent without any prior knowledge will find it extremely hard to imitate a task by simply watching a visual demonstration in all but the simplest of cases. Thus, the natural question is: in order to imitate, what form of prior knowledge must the agent possess? A large body of work [67–72] has sought to capture prior knowledge by manually pre-defining the state that must be inferred from the observations. The agent then infers how to perform the task (i.e., plan for imitation) using this state. Unfortunately, computer vision systems are often unable to estimate the state variables accurately and it has proven non-trivial for downstream planning systems to be robust to such errors.

1.4 Classical Model Based Control

At the first thought, it might appear quite stupid to some readers as to why would one try to "learn policies" either via RL or through demonstrations/imitation when we know the Newtonian physics that should suffice for a lot of manipulation tasks. Why ignore all this knowledge that we have accumulated over the past centuries and force an agent to learn from scratch?

Indeed, if it is possible to create a construction that fully describes a problem, then such a construction should obviously should be exploited to find a solution. For instance, consider the game of *chess*, *GO* or *tic-tac-toe*. A tree whose every node denotes the current state of the game and the child nodes denote all possible next moves provides a full description of the game. In such setups it makes sense to to design algorithms for game play that exploit the knowledge of game trees [2]. This process of reducing a problem to a construction that is suitable for finding solutions is referred to as *system identification*. Even for manipulation tasks, ewtonian physics does not operate on high-dimensional sensory observations such as images, but it operates on quantities such as mass, friction, shape etc. For using physics, for every problem it is required to identify what parameters are required to conduct the simulation (see Figure 1.3).

System identification is only the first step and the agent must estimate these parameters from its sensory observations. This is referred to as *state estimation*. I will now discuss the challenges in system identification (section 1.4.1) and state estimation (section 1.4.2).

1.4.1 System Identification

The key component of system identification is the choice of the how the system is represented. For instance, a tabletop environment with objects could be represented by position of objects. While such a representation might allow an agent to push objects to desired locations, it is insufficient to push objects into desired poses. The solution is to augment the state representation with pose information. In case objects are of irregular geometry, it might be necessary to also include object shape in the representation. This simple example illustrates the intimate connection between the state representation and the nature of tasks an agent can perform.

Identifying an "appropriate" representation for solving a given family of tasks turns out to be quite non-trivial. I will present one more illustrative example to make the point. Consider the task of indoor navigation. One common way of approaching this problem is to represent indoor environments in terms of obstacles and free-space. The process of finding obstacles and free spaces is typically referred to as estimating the map and is widely studied in the SLAM literature [73]. The navigation problem then reduces to finding a path between two points so that every point in the path belongs to free space. Numerous motion planning [74] algorithms can be employed to find paths from a current to a target position.

It might appear that this representation of obstacles and free space is appropriate for navigation until the time the agent encounters a door, a fleet of stairs or an elevator. The knowledge that doors can be opened to explore new space, stairs/elevators can

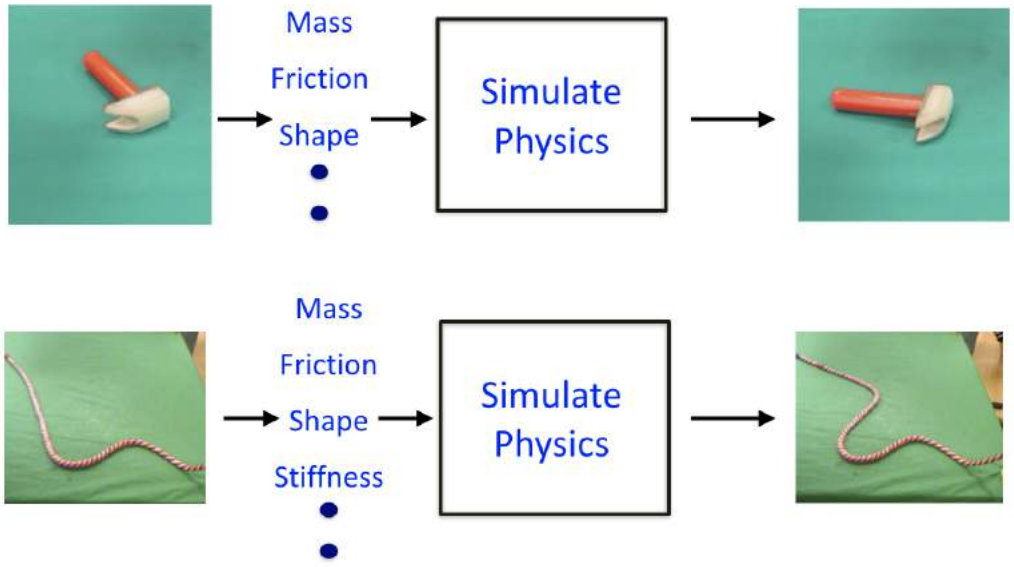


Figure 1.3: System Identification: Newtonian physics does not operate on high-dimensional sensory observations such as images, but it operates on quantities such as mass, friction, shape etc. While for pushing objects we donot require to estimate stiffness, for manipulating ropes it is critical. For using physics simulation, for every problem it is required to identify what parameters are required to conduct the simulation. This process is referred to as system identification and can often be quite tedious.

be used to move between floors or the fact that some corridors lead to a dead-end, while some lead to other corridors for further exploration is completely absent. Incorporating such knowledge into state representation becomes necessary to navigate. One might do so by constructing classifiers that identify these semantic entities in the map. Note that just semantic identification is not sufficient and how to interact with these entities must also be coded up and its ends up being non-trivial (see section 5 in [73]). The core problem is that it is hard to enumerate all possible interactions that the agent might require. For e.g., identifying the chair as a sittable surface might not be sufficient and chair might need to be displaced to navigate to the space behind it. Furthermore, our environments are not static and representing dynamic entities such as humans and other agents further complicates things.

This problem of identifying the appropriate representation is not specific to navigation or table top manipulation, but is a core problem at the root of design of all AI systems. For instance, in computer vision, the initial work sought to represent objects by relationships between edges that constitute it. Such representations proved to be brittle and were succeeded by the era of feature descriptors such as

SIFT [75] and HOG [76] that were designed to be invariant to nuisance factors to object categorization such as illumination, scale and rotation. While these features were more successful than the previous approaches, it was realized that hand-design of features was hard and deep learning based approaches [5, 6, 77] that sought to automatically learn task-specific features (aka representation) turned out to be significantly more effective. The general difficulty of finding an appropriate representation has been discussed in point 8 in section 1.2.1.

1.4.2 State Estimation

Deciding how to represent a given system is only part of the solution. In general, an agent doesnot has direct access to the state (say z) of the system but can only observe the environment through its sensors such as vision, audition, touch etc. Let the sensory observation of the agent be denoted as X. Inferring the state (z) from raw sensory observations (X) is referred to as state estimation.

State estimation is extremely challenging. Its been more that five decades of computer vision and we simply donot know how to accurately estimate properties such as mass, stiffness etc. directly from images. Even if were to use the latest advances in computer vision relying on supervised learning, it is unclear from where to obtain large datasets of annotations for such properties. Secondly, even if we could get such supervision, the estimates are likely to be noisy, which in turn will lead to inaccuracies in the output of the physics simulation (see Figure 1.3).

1.4.3 Is the engineering wisdom of modularization the way to go?

It is engineering wisdom to decompose the problem into simpler modules, work on these modules individually and compose them to solve the overall problem. Unless these modules are perfect, there is always a danger that errors can compound when these modules are connected together and thereby sacrificing the overall system performance. In developing intelligent agents, we must be careful of this peril [78]. For instance, in order to use newtonian physics to solve manipulation problems, one must decompose the system into atleast two parts: (a) State estimation from sensory observations; and (b) forward simulation of the state using physics (see Figure 1.3).

In the sections above, we have already discussed some problems associated with system identification and state estimation. Lets analyze these issues with one more example. The conclusion that objects with feather like appearance fall slower than objects with stone like appearance can be reached by either correlating visual texture to observed speed of falling objects, or by computing the drag force after estimating

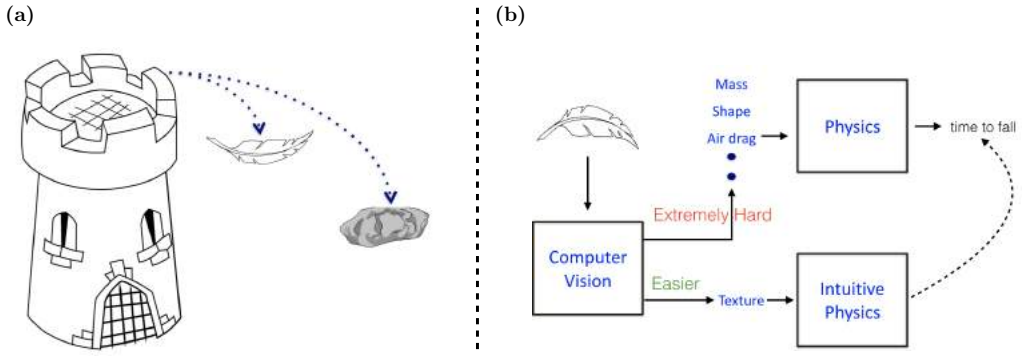


Figure 1.4: (a) Consider the problem of inferring whether the stone will fall slower than the feather when dropped from a tower. (b) In order to use (newtonian) physics for computing feather's descent time, from a single or multiple images of the feather one must estimate physical parameters such as the mass, the shape etc. These parameters are extremely hard to infer are their estimate will inevitably be noisy. On the other hand, correlating from past experience that objects with fluffy texture fall slower than solids is much easier. While such knowledge (i.e., intuitive physics) is not as general as newtonian physics, it might suffice for solving a wide-variety of common day tasks than donot require hyper-precision.

the cross section area of the object. Depending on whether estimation of visual texture or cross section area is more robust, one of these methods will be more accurate than other (see Figure 1.4).

This consideration is critical because estimating mass distribution, deformation and friction properties, contact points etc from sensory data is very challenging and it might just be the case that an alternate parameterization (such as the example of visual texture versus cross section area mentioned above) may perform as well, but is easier to estimate and more robust. Moreover, for many practical object manipulation tasks of interest, such as re-arranging objects, cutting vegetables, folding clothes, and so forth, small errors in execution are acceptable. The key challenge is robust performance in the face of varying environmental conditions. This suggests that a more robust but a somewhat imprecise model may in fact be preferred over a less robust and a more precise model (see Figure 1.5).

We call these as as "intuitive" physics models [79–81]. Intuitive physics is the simple understanding of how actions effect objects that is obtained from interaction data and is possibly different from the physics that we know from Newton's law. The intuitive physics approach is in the spirit of recent successful deep learning techniques in computer vision and speech processing that learn features directly from data. However, it is unclear how to build these models, because as opposed to

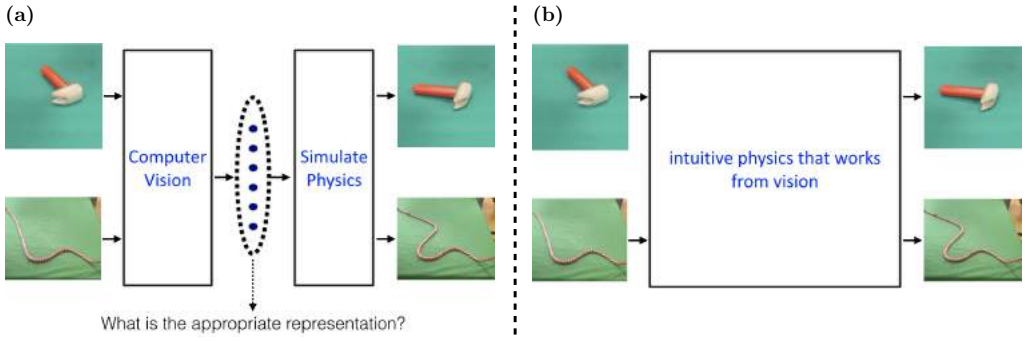


Figure 1.5: (a) The classical model based approach uses a two-staged pipeline to first infer the state representation from sensory observations and then using a physics simulator to infer the consequences of actions. It is not obvious what representation should form the interface between what the sensory system estimates and what the physics simulator operates upon. (b) Instead of separately solving the computer vision and physic simulation problems, it might be easier to jointly learn an intuitive physics models that operates on raw sensory inputs. See text for a more detailed discussion.

image classification or speech recognition system, there is no dataset for intuitive physics and the agent must explore its environment to collect such data. In human development, it is well known that infants spend years worth of time playing with objects in a seemingly random manner with no specific end goal [82]. It is possible that this experience is distilled into "intuitive" models of dynamics that are later used for inferring optimal actions for achieving specific goals.

1.5 Core problem of Artificial General Intelligence

As we have discussed above, a common characteristic of current AI systems is that they are specialized in performing one specific task. For instance a machine that can play chess on a 8×8 board is utterly incapable of playing on a 9×9 board, let alone playing a different game such as *tic-tac-toe*. Human intelligence or common sense reasoning on the other hand is characterized by the ability to use previous knowledge to solve new problems either more efficiently or significantly faster than a system that starts off with no prior knowledge.

The core problem of intelligence doesnot seems to be building agents that can solve some particular complex tasks, but agents that continuously increase their understanding of their environment. The issue with building agents to solve one particular task is that there is no incentive to learn any generalizable skills and

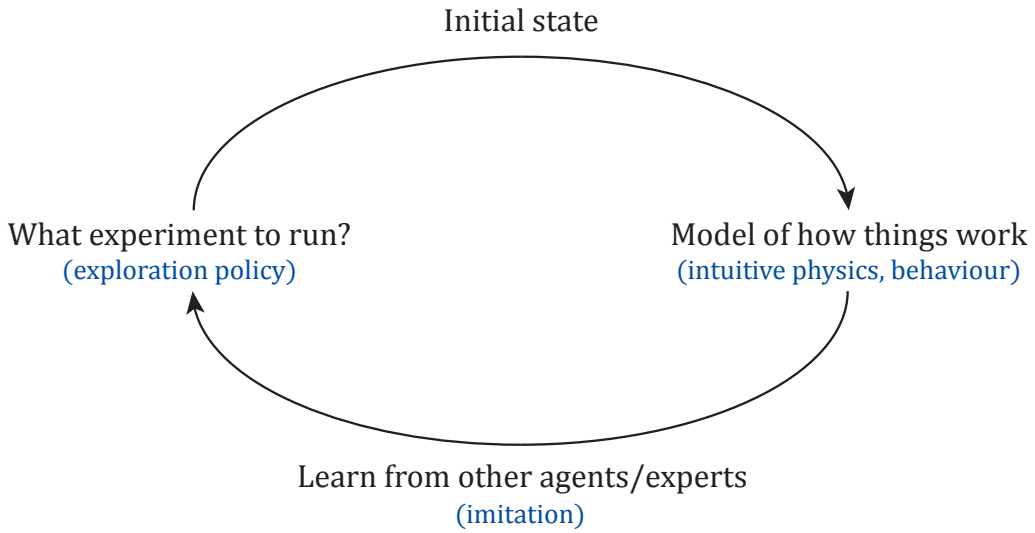


Figure 1.6: An overview of the proposed approach to general artificial intelligence.

when they are posed a new-task they start off almost from a tabula-rasa state. Furthermore, because such agents are not capable of exploring, they will be unable to seek solutions to problems that cannot be solved with the agent's current knowledge. An agent that has the ability to increment its knowledge can find solutions by either by communicating with other agents or by conducting experiments in its environment to learn about the necessary things.

Furthermore, as discussed extensively in section 1.2.1 that the major problem in communicating tasks might result as a by-product of trying to directly solve complex problems instead of building agents that can leverage knowledge gained by solving simpler problems to tackle more complex ones. It seems that if we direct efforts towards building agents that gradually increase in complexity/understanding of their environment, it might provide a way to deal with the challenges of working of sensory observations and agents that are not completely reliant on external supervision, but can find their own solutions.

1.6 Summary of the Proposed Solution

This dissertation takes inspiration from developmental psychology that studies how infants learn about their environment to construct a paradigm for building agents that can also continuously learn about their environment and bring to bear this knowledge to accomplish target tasks.

The prominent theory in developmental psychology suggests that seemingly

frivolous play is a *self-supervised* mechanism for infants to *conduct experiments* for incrementally increasing their knowledge [82, 83]. Experiments conducted by infants such as throwing objects, hitting two objects against each other or putting them in their mouth help them understand how forces affect objects, what happens when objects are touched, how different materials interact, etc. In a way, *play teaches infants to use the tool of experimentation* to learn a model of how things work, which in turn could be used to perform new tasks or adapt to new environments.

There four main components of building an agent that can continously learn by experimenting are (see Figure 1.6): (a) The agent must be able to explore its environment to collect data for incrementing its knowledge. [84] provides a formalization of curiosity based exploration that manages to overcome many pitfalls of previous works. (b) The interaction data gathered from exploration is used to learn models from raw sensory observations that are useful for performing previously unseen tasks [79, 85]. (c) There is just too much to learn about the world. Many times what we learn is biased by observing other agents in our environment or by taking cues from them. I will show how the models of the environment learned by the agent enable imitation by watching experts [86, 87], and bias the agent's exploration towards interesting visual states shown by an expert [86]. (d) Finally, what an agent explores and learns inevitably depends on the initial state of the agent. This dissertation provides some answers to these questions in the following chapters, and this paradigm is called *Computational Sensorimotor Learning*.

In addition to interacting with objects, agents also interact with humans/other agents in their environments. Interaction with other agents can be studied with a paradigm similar to interaction with objects. Results of some initial investigation in this direction are summarized in [24, 88].

I believe that answers to intelligence will not be found by agents in lab environments or simulation, but by robots that explore and conduct experiments in the real world. My dissertation is an undertaking of the challenge of putting this hypothesis to test.

The main challenge in constructing such an agent is of building a model that summarizes the experience of the agent. One possibility is to build a model that predicts the future sensory observations from the current observations and action. Lets assume that our agent has access to visual observations. Building a model that predicts in the visual space is very challenging because it requires predicting the image at the next time step. Predicting the value of every pixel in a future image is highly non-trivial in real world scenarios, and in most cases it is not the precise pixel values that are of interest, but the occurrence of a more abstract event. For example, predicting that a glass jar will break when pushed from the table onto the ground is of greater interest (and easier) than predicting exactly how every piece of shattered

glass will look. The difficulty, however, is that supervision for such abstract concepts or events is not readily available.

One solution to this problem is proposed in chapter 2 that shows how the agent can predict its own actions to learn visual representations for visual recognition tasks. This idea is expanded in form of co-learning of forward and inverse dynamics in chapter 3 that are shown to be useful for pushing objects kept on a table. A forward model predicts the next state from the current state and action, and an inverse model predicts the action given the initial and target state. In joint training,the inverse model objective provides supervision for transforming image pixels into an abstract feature space, which the forward model can then predict. The inverse model alleviates the need for the forward model to make predictions in the raw sensory space and the forward model in turn regularizes the feature space for the inverse model.

Chapters 4 and 5 show how the joint forward-inverse model enables the agent to easily imitate and bias its exploration by observing an expert. Chapter 6 makes use of the abstract feature space learned by joint forward-inverse models to learn a curiosity-driven exploration strategy. Further improvements to the model and the utility of data collected from curiosity-driven exploration for learning good models is presented in Chapter 5. Chapter 7 explores what might be good priors to build into an agent to bootstrap learning by conducting a study on how humans explore while playing video games. Chapter 8 describes a system for human pose estimation and building models that predict the activity of other humans in sports-games (i.e. intuitive behavior). Finally, chapter 9 contrasts the representations learned by deep neural networks optimized for end tasks against the human visual representation and presents an hypothesis about how deep learning systems might provide insights into the human neural systems.

Chapter 2

Learning to See by Moving

Recent advances in computer vision have shown that visual features learnt by neural networks trained for the task of object recognition using more than a million labelled images are useful for many computer vision tasks like semantic segmentation, object detection and action classification [5, 27, 58, 89]. However, object recognition is one among many tasks for which vision is used. For example, humans use visual perception for recognizing objects, understanding spatial layouts of scenes and performing actions such as moving around in the world. Is there something special about the task of object recognition or is it the case that useful visual representations can be learnt through other modes of supervision? Clearly, biological agents perform complex visual tasks and it is unlikely that they require external supervision in form of millions of labelled examples. Unlabelled visual data is freely available and in theory this data can be used to learn useful visual representations. However, until now unsupervised learning approaches [36, 90–92] have not yet delivered on their promise and are nowhere to be seen in current applications on complex real world imagery.

Biological agents use perceptual systems for obtaining sensory information about their environment that enables them to act and accomplish their goals [93, 94]. Both biological and robotic agents employ their motor system for executing actions in their environment. Is it also possible that these agents can use their own motor system as a source of supervision for learning useful perceptual representations? Motor theories of perception have a long history [93, 94], but there has been little work in formulating computational models of perception that make use of motor information. In this work we focus on visual perception and present a model based on egomotion (i.e. self motion) for learning useful visual representations. When we say useful visual representations [95], we mean representations that possess the following two characteristics - (1) ability to perform multiple visual tasks and (2)

ability of performing new visual tasks by learning from only a few labeled examples provided by an extrinsic teacher.

Mobile agents are naturally aware of their egomotion (i.e. self-motion) through their own motor system. In other words, knowledge of egomotion is "freely" available. For example, the vestibular system provides the sense of orientation in many mammals. In humans and other animals, the brain has access to information about eye movements and the actions performed by the animal [94]. A mobile robotic agent can estimate its egomotion either from the motor commands it issues to move or from odometry sensors such as gyroscopes and accelerometers mounted on the agent itself.

We propose that useful visual representations can be learnt by performing the simple task of correlating visual stimuli with egomotion. A mobile agent can be treated like a camera moving in the world and thus the knowledge of egomotion is the same as the knowledge of camera motion. Using this insight, we pose the problem of correlating visual stimuli with egomotion as the problem of predicting the camera transformation from the consequent pairs of images that the agent receives while it moves. Intuitively, the task of predicting camera transformation between two images should force the agent to learn features that are adept at identifying visual elements that are present in both the images (i.e. visual correspondence). In the past, features such as SIFT, that were hand engineered for finding correspondences were also found to be very useful for tasks such as object recognition [75, 96]. This suggests that egomotion based learning can also result in features that are useful for such tasks.

In order to test our hypothesis of feature learning using egomotion, we trained multilayer neural networks to predict the camera transformation between pairs of images. As a proof of concept, we first demonstrate the usefulness of our approach on the MNIST dataset [97]. We show that features learnt using our method outperform previous approaches of unsupervised feature learning when class-label supervision is available only for a limited number of examples (section 2.2.4) Next, we evaluated the efficacy of our approach on real world imagery. For this purpose, we used image and odometry data recorded from a car moving through urban scenes, made available as part of the KITTI [98] and the San Francisco (SF) city [99] datasets. This data mimics the scenario of a robotic agent moving around in the world. The quality of features learnt from this data were evaluated on four tasks (1) Scene recognition on SUN [100] (section 2.4.1), (2) Visual odometery (section 2.4.4), (3) Keypoint matching (section 2.4.3) and (4) Object recognition on Imagenet [14] (section 2.4.2). Our results show that for the same amount of training data, features learnt using egomotion as supervision compare favorably to features learnt using class-label as supervision. We also show that egomotion based pretraining outperforms a

previous approach based on slow feature analysis for unsupervised learning from videos [41,42,101]. To the best of our knowledge, this work provides the first effective demonstration of learning visual representations from non-visual access to egomotion information in real world setting.

The rest of this paper is organized as following: In section 2.1 we discuss the related work, in section 2.2, 2.3, 2.4 we present the method, dataset details and we conclude with the discussion in section 2.5.

2.1 Related Work

Past work in unsupervised learning has been dominated by approaches that pose feature learning as the problem of discovering compact and rich representations of images that are also sufficient to reconstruct the images [36,38,91,102–104]. Another line of work has focused on learning features that are invariant to transformations either from video [41,42,101] or from images [92,105]. [106] perform feature learning by modeling spatial transformations using boltzmann machines, but donot evaluate the quality of learnt features.

Despite a lot of work in unsupervised learning (see [90] for a review), a method that works on complex real world imagery is yet to be developed. An alternative to unsupervised learning is to learn features using intrinsic reward signals that are freely available to a system (i.e self-supervised learning). For instance, [107] used intrinsic reward signals available to a robot for learning features that predict path traversability, while [62] trained neural networks for driving vehicles directly from visual input.

In this work we propose to use non-visual access to egomotion information as a form of self-supervision for visual feature learning. Unlike any other previous work, we show that our method works on real world imagery. Closest to our method is the the work of transforming auto-encoders [37] that used egomotion to reconstruct the transformed image from an input source image. This work was purely conceptual in nature and the quality of learned features was not evaluated. In contrast, our method uses egomotion as supervision by predicting the transformation between two images using a siamese-like network model [108].

Our method can also be seen as an instance of feature learning from videos. [41,42,101] perform feature learning from videos by imposing the constraint that temporally close frames should have similar feature representations (i.e. slow feature analysis) without accounting for either the camera motion or the motion of objects in the scene. In many settings the camera motion dominates the motion content of the video. Our key observation is that knowledge of camera motion (i.e. egomotion)

is *freely* available to mobile agents and can be used as a powerful source of self-supervision.

2.2 A Simple Model of Motion-based Learning

Figure 2.1: Exploring the utility of egomotion as supervision for learning useful visual features. A mobile agent equipped with visual sensors receives a sequence of images as inputs while it moves in its environment. The movement of the agent is equivalent to the movement of a camera. In this work, egomotion based learning is posed as the problem of predicting camera transformation from image pairs. The top and bottom rows of the figure show some sample image pairs from the SF and KITTI datasets that were used for feature learning.

We model the visual system of the agent with a Convolutional Neural Network (CNN, [77]). The agent optimizes its visual representations (i.e. updating the weights of the CNN) by minimizing the error between the egomotion information (i.e. camera transformation) obtained from its motor system and egomotion predicted using its visual inputs only. Performing this task is equivalent to training a CNN with two streams (i.e. Siamese Style CNN or SCNN [108]) that takes two images as inputs and predicts the egomotion that the agent underwent as it moved between the two spatial locations from which the two images were obtained. In order to learn useful visual representations, the agent continuously performs this task as it moves around in its environment.

In this work we use the pretraining-finetuning paradigm for evaluating the utility of learnt features. Pretraining is the process of optimizing the weights of a randomly initialized CNN for an auxiliary task that is not the same as the target task. Finetuning is the process of modifying the weights of a pretrained CNN for the given

target task. Our experiments compare the utility of features learnt using egomotion based pretraining against class-label based and slow-feature based pretraining on multiple target tasks.

2.2.1 Two Stream Architecture

Each stream of the CNN independently computes features for one image. Both streams share the same architecture and the same set of weights and consequently perform the same set of operations for computing features. The individual streams have been called as Base-CNN (BCNN). Features from two BCNNs are concatenated and passed downstream into another CNN called as the Top-CNN (TCNN) (see figure 2.2). TCNN is responsible for using the BCNN features to predict the camera transformation between the input pair of images. After pretraining, the TCNN is removed and a single BCNN is used as a standard CNN for feature computation for the target task.

2.2.2 Shorthand for CNN architectures

The abbreviations Ck, Fk, P, D, Op represent a convolutional(C) layer with k filters, a fully-connected(F) layer with k filters, pooling(P), dropout(D) and the output(Op) layers respectively. We used ReLU non-linearity after every convolutional/fully-connected layer, except for the output layer. The dropout layer was always used with dropout of 0.5. The output layer was a fully connected layer with number of units equal to the number of desired outputs. As an example of our notation, C96-P-F500-D refers to a network with 96 filters in the convolution layer followed by ReLU non-linearity, a pooling layer, a fully-connected layer with 500 unit, ReLU non-linearity and a dropout layer. We used [109] for training all our models.

2.2.3 Slow Feature Analysis (SFA) Baseline

Slow Feature Analysis (SFA) is a method for feature learning based on the principle that useful features change slowly in time. We used the following contrastive loss formulation of SFA [101, 108],

$$L(x_{t_1}, x_{t_2}, W) = \begin{cases} D(x_{t_1}, x_{t_2}) & \text{if } |t_1 - t_2| \leq T \\ max\big(0, m - D(x_{t_1}, x_{t_2})\big) & \text{if } |t_1 - t_2| > T \end{cases} \tag{2.1}$$

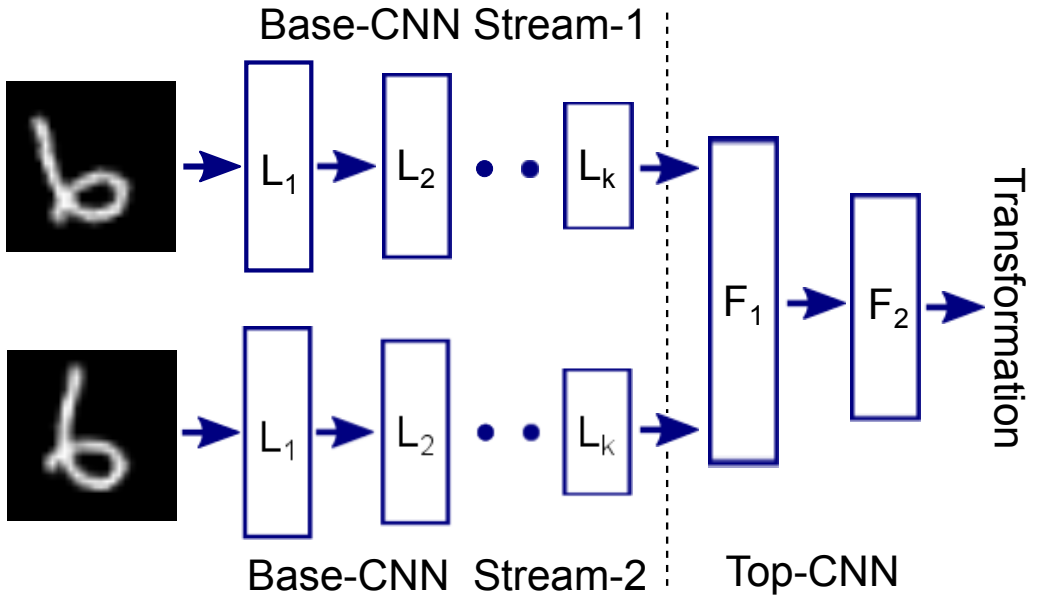


Figure 2.2: Description of the method for feature learning. Visual features are learnt by training a Siamese style Convolutional Neural Network (SCNN, [108]) that takes as inputs two images and predicts the transformation between the images (i.e. egomotion). Each stream of the SCNN (called as Base-CNN or BCNN) computes features for one image. The outputs of two BCNNs are concatenated and passed as inputs to a second multilayer CNN called as the Top-CNN (TCNN) (shown as layers F_1, F_2). The two BCNNs have the same architecture and share weights. After feature learning, TCNN is discarded and a single BCNN stream is used as a standard CNN for extracting features for performing target tasks like scene recognition.

where, L is the loss, x_{t_1}, x_{t_2} refer to feature representations of frames observed at times t_1, t_2 respectively, W are the parameters that specify the feature extraction process, D is a measure of distance with parameter, m is a predefined margin and T is a predefined time threshold for determining whether the two frames are temporally close or not. In this work, x_t are features computed using a CNN with weights W and D was chosen to be the L2 distance. SFA pretraining was performed using two stream architectures that took pairs of images as inputs and produced outputs x_{t_1}, x_{t_2} as outputs from the two streams respectively.

2.2.4 Proof of Concept using MNIST

On MNIST, egomotion was emulated by generating synthetic data consisting of random transformation (translations and rotations) of digit images. From the training set of 60K images, digits were randomly sampled and then transformed using two different sets of random transformations to generate image pairs. CNNs were trained for predicting the transformations between these image pairs.

Data

For egomotion based pretraining, relative translation between the digits was constrained to be an integer value in the range [-3, 3] and relative rotation θ was constrained to lie within the range [-30 30. The prediction of transformation was posed as a classification task with three separate soft-max losses (one each for translation along X, Y axes and the rotation about Z-axis). SCNN was trained to minimize the sum of these three losses. Translations along X, Y were separately binned into seven uniformly spaced bins each. The rotations were binned into bins of size 3ach resulting into a total of 20 bins (or classes). For SFA based pretraining, image pairs with relative translation in the range [-1, 1] and relative rotation within [-3 3 were considered to be temporally close to each other (see equation 2.1). A total of 5 million image pairs were used for both pretraining procedures.

Network Architectures

We experimented with multiple BCNN architectures and chose the optimal architecture for each pretraining method separately. For egmotion based pretraining, the two BCNN streams were concatenated using the TCNN: *F1000-D-Op*. Pretraining was performed for 40K iterations (i.e. 5M examples) using an initial learning rate of 0.01 which was reduced by a factor of 2 after every 10K iterations.

The following architecture was used for finetuning: *BCNN-F500-D-Op*. In order to evaluate the quality of BCNN features, the learning rate of all layers in the BCNN were set to 0 during finetuning for digit classification. Finetuning was performed for 4K iterations (which is equivalent to training for 50 epochs for the 10K labelled training examples) with a constant learning rate of 0.01.

Results

The BCNN features were evaluated by computing the error rates on the task of digit classification using 100, 300, 1K and 10K class-labelled examples for training. These sets were constructed by randomly sampling digits from the standard training

Table 2.1: Comparison of various pretraining methods on MNIST reveals that egomotion based pretraining outperforms many previous approaches for unsupervised learning. The performance is reported as the **error rate**.

Method	# examples for finetuning			
	100	**300**	**1000**	**10000**
Autoencoder [109]	24.1	12.2	7.7	4.8
Ranzato et al. [92]	-	7.18	3.21	**0.85**
Lee et al. [91]	-	-	2.62	-
Train from Scratch	20.1	8.3	4.5	1.6
SFA (m=10)	11.2	6.4	3.5	2.1
SFA (m=100)	11.9	6.4	4.8	4.7
Egomotion (ours)	**8.7**	**3.6**	**2.0**	0.9

set of 60K digits. For this part of the experiment, the original digit images were used (i.e. without any transformations or data augmentation). The standard test set of 10K digits was used for evaluation and error rates averaged across 3 runs are reported in table 2.1.

The BCNN architecture: C96-P-C256-P, was found to be optimal for egomotion and SFA based pretraining and also for training from scratch (i.e. random weight initialization). Results for other architectures are provided in the supplementary material. For SFA based pretraining, we experimented with multiple values of the margin m and found that $m = 10, 100$ led to the best performance. Our method outperforms convolutional deep belief networks [91], a previous approach based on learning features invariant to transformations [92] and SFA based pretraining.

2.3 Learning Visual Features From Egomotion in Natural Environments

We used two main sources of real world data for feature learning: the KITTI and SF datasets, which were collected using cameras and odometry sensors mounted on a car driving through urban scenes. Details about the data, the experimental procedure, the network architectures and the results are provided in sections 2.3.1, 2.3.2, 2.3.3 and 2.4 respectively.

2.3.1 KITTI Dataset

The KITTI dataset provided odometry and image data recorded during 11 short trips of variable length made by a car moving through urban landscapes. The total number of frames in the entire dataset was 23,201. Out of 11, 9 sequences were used for training and 2 for validation. The total number of images in the training set was 20,501.

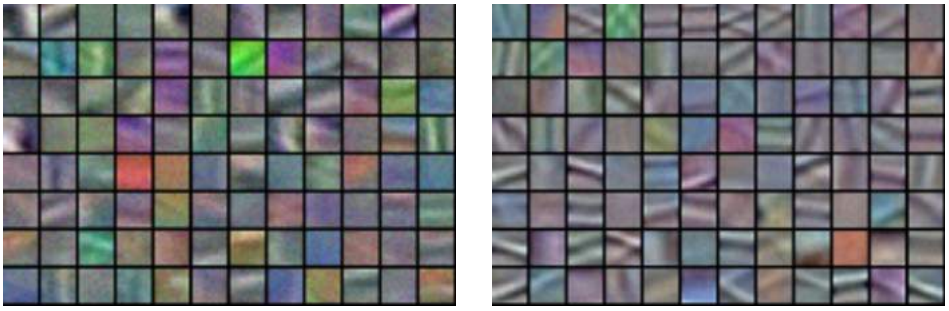
The odometry data was used to compute the camera transformation between pairs of images recorded from the car. The direction in which the camera pointed was assumed to be the Z axis and the image plane was taken to be the XY plane. X-axis and Y-axis refer to horizontal and vertical directions in the image plane. As significant camera transformations in the KITTI data were either due to translations along the Z/X axis or rotation about the Y axis, only these three dimensions were used to express the camera transformation. The rotation was represented as the euler angle about the Y-axis. The task of predicting the transformation between pair of images was posed as a classification problem. The three dimensions of camera transformation were individually binned into 20 uniformly spaced bins each. The training image pairs were selected from frames that were at most ± 7 frames apart to ensure that images in any given pair would have a reasonable overlap. For SFA based pretraining, pairs of frames that were separated by atmost ± 7 frames were considered to be temporally close to each other.

The SCNN was trained to predict camera transformation from pairs of 227×227 pixel sized image regions extracted from images of overall size 370×1226 pixels. For each image pair, the coordinates for cropping image regions were randomly chosen. Figure 2.1 illustrates typical image crops.

2.3.2 SF Dataset

SF dataset provides camera transformation between $\approx$ 136K pairs of images (constructed from a set of 17,357 unique images). This dataset was constructed using Google StreetView [99]. $\approx 130K$ image pairs were used for training and $\approx 6K$ pairs for validation.

Just like KITTI, the task of predicting camera transformation was posed as a classification problem. Unlike KITTI, significant camera transformation was found along all six dimensions of transformation (i.e. the 3 euler angles and the 3 translations). Since, it is unreasonable to expect that visual features can be used to infer big camera transformations, rotations between [-30 30 were binned into 10 uniformly spaced bins and two extra bins were used for rotations larger and smaller than 30nd -30espectively. The three translations were individually binned into 10



(a) KITTI-Net (b) SF-Net

Figure 2.3: Visualization of layer 1 filters learnt by egomotion based pretraining on (a) KITTI and (b) SF datasets. A large majority of layer-1 filters are color detectors and some of them are edge detectors. This is expected as color is a useful cue for determining correspondences between image pairs.

uniformly spaced bins each. Images were resized to a size of 360×480 and image regions of size 227×227 were used for training the SCNN.

2.3.3 Network Architecture

BCNN closely followed the architecture of first five AlexNet layers [5]: *C96-P-C256-P-C384-C384-C256-P*. TCNN architecture was: *C256-C128-F500-D-Op*. The convolutional filters in the TCNN were of spatial size 3×3. The networks were trained for 60K iterations with a batch size of 128. The initial learning rate was set to 0.001 and was reduced by a factor of two after every 20K iterations.

We term the networks pretrained using egomotion on KITTI and SF datasets as KITTI-Net and SF-Net respectively. The net pretrained on KITTI with SFA is called KITTI-SFA-Net. Figure 2.3 shows the layer-1 filters of KITTI-Net and SF-Net. A large majority of layer-1 filters are color detectors, while some of them are edge detectors. As color is a useful cue for determining correspondences between closeby frames of a video sequence, learning of color detectors as layer-1 filters is not surprising. The fraction of filters that detect edges is higher for the SF-Net. This is not surprising either, because higher fraction of images in the SF dataset contain structured objects like buildings and cars.

2.4 Evaluating Motion-based Learning

For evaluating the merits of the proposed approach, features learned using egomotion based supervision were compared against features learned using class-label and SFA based supervision on the challenging tasks of scene recognition, intra-class keypoint matching and visual odometry and object recognition. The ultimate goal of feature learning is to find features that can generalize from only a few supervised examples on a new task. Therefore it makes sense to evaluate the quality of features when only a few labelled examples for the target task are provided. Consequently, the scene and object recognition experiments were performed in the setting when only 1-20 labelled examples per class were available for finetuning.

The KITTI-Net and SF-Net (examples of models trained using egomotion based supervision) were trained using only only $\approx$ 20K unique images. To make a fair comparison with class-label based supervision, a model with AlexNet architecture was trained using only 20K images taken from the training set of ILSVRC12 challenge (i.e. 20 examples per class). This model has been referred to as AlexNet-20K. In addition, some experiments presented in this work also make comparison with AlexNet models trained with 100K and 1M images that have been named as AlexNet-100K and AlexNet-1M respectively.

2.4.1 Scene Recognition

SUN dataset consisting of 397 indoor/outdoor scene categories was used for evaluating scene recognition performance. This dataset provides 10 standard splits of 5 and 20 training images per class and a standard test set of 50 images per class. Due to time limitation of running 10 runs of the experiment, we evaluated the performance using only 3 train/test splits.

For evaluating the utility of CNN features produced by different layers, separate linear (SoftMax) classifiers were trained on features produced by individual CNN layers (i.e. BCNN layers of KITTI-Net, KITTI-SFA-Net and SF-Net). Table 2.2 reports recognition accuracy (averaged over 3 train/test splits) for various networks considered in this study. KITTI-Net outperforms SF-Net and is comparable to AlexNet-20K. This indicates that given a fixed budget of pretraining images, egomotion based supervision learns features that are almost as good as the features using class-based supervision on the task of scene recognition. The performance of features computed by layers 1-3 (abbreviated as L1, L2, L3 in table 2.2) of the KITTI-SFA-Net and KITTI-Net is comparable, whereas layer 4, 5 features of KITTI-Net significantly outperform layer 4, 5 features of KITTI-SFA-Net. This indicates that egomotion based pretraining results into learning of higher-level features, while SFA based

Table 2.2: Comparing the **accuracy** of neural networks pre-trained using motion-based and class-label based supervision for the task of scene recognition on the SUN dataset. The performance of layers 1-6 (labelled as L1-L6) of these networks was evaluated after finetuning the network using 5/20 images per class from the SUN dataset. The performance of the KITTI-Net (i.e. motion-based pretraining) fares favorably with a network pretrained on Imagenet (i.e. class-based pretraining) with the same number of pretraining images (i.e. 20K).

Method	**Pretrain Supervision**	**#Pretrain**	**#Finetune**	**L1**	**L2**	**L3**	**L4**	**L5**	**L6**	**#Finetune**	**L1**	**L2**	**L3**	**L4**	**L5**	**L6**
AlexNet-1M	Class-Label	1M	5	5.3	10.5	12.1	12.5	18.0	23.6	20	11.8	22.2	25.0	26.8	33.3	37.6
AlexNet-20K		20K	5	4.9	6.3	6.6	6.3	6.6	6.7	20	8.7	12.6	12.4	11.9	12.5	12.4
KITTI-SFA-Net	Slowness	20.5K	5	4.5	5.7	6.2	3.4	0.5	-	20	8.2	11.2	12.0	7.3	1.1	-
SF-Net	Egomotion	18K	5	4.4	5.2	4.9	5.1	4.7	-	20	8.6	11.6	10.9	10.4	9.1	-
KITTI-Net		20.5K	5	4.3	6.0	5.9	5.8	6.4	-	20	7.9	12.2	12.1	11.7	12.4	-
GIST [100]	Human	-	5	6.2						20	11.6					
SPM [100]	Human	-	5	8.4						20	16.0					

pretraining results into learning of lower-level features only.

The KITTI-Net outperforms GIST [110], which was specifically developed for scene classification, but is outperformed by Dense SIFT with spatial pyramid matching (SPM) kernel [96]. The KITTI-Net was trained using limited visual data ($\approx 20Kframes$) containing visual imagery of limited diversity. The KITTI data mainly contains images of roads, buildings, cars, few pedestrians, trees and some vegetation. It is in fact surprising that a network trained on data with such little diversity is competitive on classifying indoor and outdoor scenes with the AlexNet-20K that was trained on a much more diverse set of images. We believe that with more diverse training data for egomotion based learning, the performance of learnt features will be better than currently reported numbers.

The KITTI-Net outperformed the SF-Net except for the performance of layer 1 (L1). As it was possible to extract a larger number of image region pairs from the KITTI dataset as compared to the SF dataset (see section 2.3.1, 2.3.2), the result that KITTI-Net outperforms SF-Net is not surprising. Because KITTI-Net was found to be superior to the SF-Net in this experiment, the KITTI-Net was used for all other experiments described in this paper.

2.4.2 Object Recognition

If egomotion based pretraining learns useful features for object recognition, then a net initialized with KITTI-Net weights should outperform a net initialized with random weights on the task of object recognition. For testing this, we trained CNNs using 1, 5, 10 and 20 images per class from the ILSVRC-2012 challenge. As this

Table 2.3: Top-5 **accuracy** on the task of object recognition on the ILSVRC-12 validation set. AlexNet-Scratch refers to a net with AlexNet architecture initialized with randomly weights. The weights of KITTI-Net and KITTI-SFA-Net were learned using egomotion based and SFA based supervision on the KITTI dataset respectively. All the networks were finetuned using $1, 5, 10, 20$ examples per class. The KITTI-Net clearly outperforms AlexNet-Scratch and KITTI-SFA-Net.

Method	1	5	10	20
AlexNet-Scratch	1.1	3.1	5.9	14.1
KITTI-SFA-Net (Slowness)	1.5	3.9	6.1	14.9
KITTI-Net (Egomotion)	**2.3**	**5.1**	**8.6**	**15.8**

dataset contains 1000 classes, the total number of training examples available for training for these networks were 1K, 5K, 10K and 20K respectively. All layers of KITTI-Net, KITTI-SFA-Net and AlexNet-Scratch (i.e. CNN with random weight initialization) were finetuned for image classification.

The results of the experiment presented in table 2.3 show that egomotion based supervision (KITTI-Net) clearly outperforms SFA based supervision(KITTI-SFA-Net) and AlexNet-Scratch. As expected, the improvement offered by motion-based pretraining is larger when the number of examples provided for the target task are fewer. These result show that egomotion based pretraining learns features useful for object recognition.

2.4.3 Intra-Class Keypoint Matching

Identifying the same keypoint of an object across different instances of the same object class is an important visual task. Visual features learned using egomotion, SFA and class-label based supervision were evaluated for this task using keypoint annotations on the PASCAL dataset [111].

Keypoint matching was computed in the following way: First, ground-truth object bounding boxes (GT-BBOX) from PASCAL-VOC2012 dataset were extracted and re-sized (while preserving the aspect ratio) to ensure that the smaller side of the boxes was of length 227 pixels. Next, feature maps from layers 2-5 of various CNNs were computed for every GT-BBOX. The keypoint matching score was computed between all pairs of GT-BBOX belonging to the same object class. For given pair of GT-BBOX, the features associated with keypoints in the first image were used to predict the location of the same keypoints in the second image. The normalized pixel distance between the actual and predicted keypoint locations was taken as the

error in keypoint matching. More details about this procedure have been provided in the supp. materials.

It is natural to expect that accuracy of keypoint matching would depend on the camera transformation between the two viewpoints of the object(i.e. viewpoint distance). In order to make a holistic evaluation of the utility of features learnt by different pretraining methods on this task, matching error was computed as a function of viewpoint distance [112]. Figure 2.4 reports the matching error averaged across all keypoints, all pairs of GT-BBOX and all classes using features extracted from layers conv-3 and conv-4.

KITTI-Net trained only with 20K unique frames was superior to AlexNet-20K and AlexNet-100K and inferior only to AlexNet-1M. A net with AlexNet architecture initialized with random weights (AlexNet-Rand), surprisingly performed better than AlexNet-20K. One possible explanation for this observation is that with only 20K examples, features learnt by AlexNet-20K only capture coarse global appearance of objects and are therefore poor at keypoint matching. SIFT has been hand engineered for finding correspondences across images and performs as well as the best AlexNet-1M features for this task (i.e. conv-4 features). KITTI-Net also significantly outperforms KITTI-SFA-Net. These results indicate that features learnt by egomotion based pretraining are superior to SFA and class-label based pretraining for the task of keypoint matching.

2.4.4 Visual Odometry

Visual odometry is the task of estimating the camera transformation between image pairs. All layers of KITTI-Net and AlexNet-1M were finetuned for 25K iterations using the training set of SF dataset on the task of visual odometry (see section 2.3.2 for task description). The performance of various CNNs was evaluated on the validation set of SF dataset and the results are reported in table 2.4.

Performance of KITTI-Net was either superior or comparable to AlexNet-1M on this task. As the evaluation was made on the SF dataset itself, it was not surprising that on some metrics SF-Net outperformed KITTI-Net. The results of this experiment indicate that egomotion based feature learning is superior to class-label based feature learning on the task of visual odometry.

2.5 Discussion

In this work, we have shown that egomotion is a useful source of intrinsic supervision for visual feature learning in mobile agents. In contrast to class labels,

Table 2.4: Comparing the **accuracy** of various pretraining methods on the task of visual odometry.

Method	**Translation Acc.**			**Rotation Acc.**		
	δX	δY	δZ	$\delta\theta_1$	$\delta\theta_2$	$\delta\theta_3$
SF-Net	40.2	**58.2**	38.4	45.0	**44.8**	40.5
KITTI-Net	**43.4**	57.9	**40.2**	**48.4**	44.0	**41.0**
AlexNet-1M	41.8	58.0	39.0	46.0	44.5	40.5

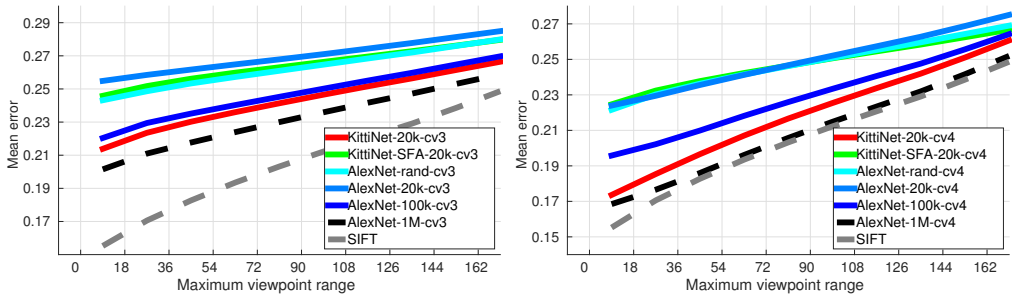


Figure 2.4: Intra-class keypoint matching error as a function of viewpoint distance averaged over 20 PASCAL objects using features from layers conv3 (left) and conv4 (right) of various CNNs used in this work. Please see the text for more details.

knowledge of egomotion is "freely" available. On MNIST, egomotion-based feature learning outperforms many previous unsupervised methods of feature learning. Given the same budget of pretraining images, on task of scene recognition, egomotion-based learning performs almost as well as class-label-based learning. Further, egomotion based features outperform features learnt by a CNN trained using class-label supervision on two orders of magnitude more data (AlexNet-1M) on the task of visual odometry and one order of magnitude more data on the task of intra-class keypoint matching. In addition to demonstrating the utility of egomotion based supervision, these results also suggest that features learnt by class-label based supervision are not optimal for all visual tasks. This means that future work should look at what kinds of pretraining are useful for what tasks.

One potential criticism of our work is that we have trained and evaluated high capacity deep models on relatively little data (e.g. only 20K unique images available on the KITTI dataset). In theory, we could have learnt better features by downsizing the networks. For example, in our experiments with MNIST we found that pretraining a 2-layer network instead of 3-layer results in better performance (table 2.1). In this

work, we have made a conscious choice of using standard deep models because the main goal of this work was not to explore novel feature extraction architectures but to investigate the value of egmotion for learning visual representations on architectures known to perform well on practical applications. Future research focused on exploring architectures that are better suited for egomotion based learning can only make a stronger case for this line of work. While egomotion is freely available to mobile agents, there are currently no publicly available datasets as large as Imagenet. Consequently, we were unable to evaluate the utility of motion-based supervision across the full spectrum of training set sizes.

In this work, we chose to first pretrain our models using a base task (i.e. egomotion) and then finetune these models for target tasks. An equally interesting setting is that of online learning where the agent has continuous access to intrinsic supervision (such as egomotion) and occasional explicit access to extrinsic teacher signal (such as the class labels). We believe that such a training procedure is likely to result in learning of better features. Our intuition behind this is that seeing different views of the same instance of an object (say) car, may not be sufficient to learn that different instances of the car class should be grouped together. The occasional extrinsic signal about object labels may prove useful for the agent to learn such concepts. Also, current work makes use of passively collected egomotion data and it would be interesting to investigate if it is possible to learn better visual representations if the agent can actively decide on how to explores its environment (i.e. active learning [113]).

Chapter 3

A Model for Intuitive Physics

In this chapter we investigate whether a robot can use it's own experience to learn an intuitive model of physics that is also effective for planning actions. In our setup (see Figure 3.1), a Baxter robot interacts with objects kept on a table in front of it by randomly poking them. The robot records the visual state of the world before and after it executes a poke in order to learn a mapping between its actions and the accompanying change in visual state caused by object motion. To date our robot has interacted with objects for more than 400 hours and in process collected more than 100K pokes on 16 distinct objects.

What kind of a model should the robot learn from it's experience? One possibility is to build a model that predicts the next visual state from the current visual state and the applied force (i.e forward dynamics model). This is challenging because predicting the value of every pixel in the next image is non-trivial in real world scenarios. Moreover, in most cases it is not the precise pixel values that are of interest, but the occurrence of a more abstract event. For example, predicting that a glass jar will break when pushed from the table onto the ground is of greater interest (and easier) than predicting exactly how every piece of shattered glass will look. The difficulty, however, is that supervision for such abstract concepts or events is not readily available in unsupervised settings such as ours. In this work, we propose one solution to this problem by jointly training forward and inverse dynamics models. A forward model predicts the next state from the current state and action, and an inverse model predicts the action given the initial and target state. In joint training, the inverse model objective provides supervision for transforming image pixels into an abstract feature space, which the forward model can then predict. The inverse model alleviates the need for the forward model to make predictions in the pixel space and the forward model in turn regularizes the feature space for the inverse model.

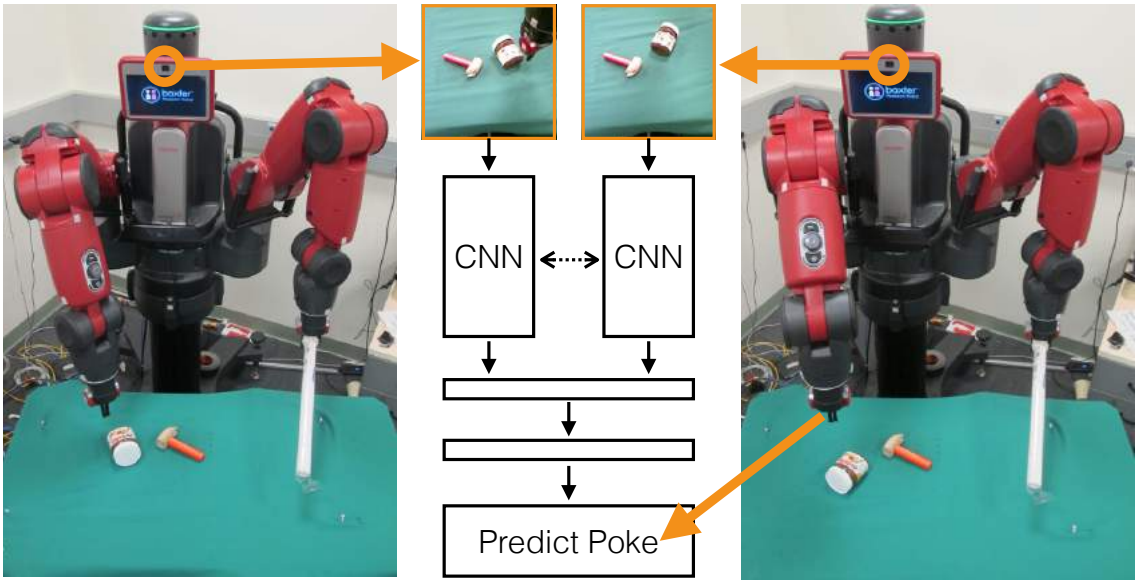


Figure 3.1: Infants spend years worth of time playing with objects in a seemingly random manner. They might use this experience to learn a model of physics relating their actions with the resulting motion of objects. Inspired by this hypothesis, we let a robot interact with objects by randomly poking them. The robot pokes objects and records the visual state before (left) and after (right) the poke. The triplet of before image, after image and the applied poke is used to train a neural network (center) for learning the mapping between actions and the accompanying change in visual state. We show that this learn model can be used to push objects into a desired configuration.

We empirically show that the joint model allows the robot to generalize and plan actions for achieving tasks with significantly different visual statistics as compared to the data used in the learning phase. Our model can be used for multi step decision making and displace objects with novel geometry and texture into desired goal locations that are much farther apart as compared to position of objects before and after a single poke. We probe the joint modeling approach further using simulation studies and show that the forward model regularizes the inverse model.

3.1 Data

Figure 1 shows our experimental setup. The robot is equipped with a Kinect camera and a gripper for poking objects kept on a table in front of it. At any given time there were 1-3 objects chosen from a set of 16 distinct objects present on the table. The robot's coordinate system was as following: X and Y axis represented the horizontal and vertical axes, while the Z axis pointed away from the robot. The robot poked objects by moving its finger along the XZ plane at a fixed height from the table.

Poke Representation: For collecting a sample of interaction data, the robot first selects a random target point in its field of view to poke. One issue with random poking is that most pokes are executed in free space which severely slows down collection of interesting interaction data. For speedy data collection, a point cloud from the Kinect depth camera was used to only chose points that lie on any object except the table. Point cloud information was only used during data collection and at test time our system only requires RGB image data. After selecting a random point to poke (p) on the object, the robot randomly samples a poke direction (θ) and length (l). Kinematically, the poke is defined by points p_1, p_2 that are $\frac{l}{2}$ distance from p in the directions $\theta^o, (180+\theta)^o$ respectively. The robot executes the poke by moving its finger from p_1 to p_2.

Our robot can run autonomously 24x7 without any human intervention. Sometimes when objects are poked they move as expected, but other times due to non-linear interaction between the robot's finger and the object they move in unexpected ways as shown in Figure 3.2. Any model of the poking data must deal with such non-linear interactions (see project website for more examples). A small amount of data in the early stages of the project was collected on a table with a green background, but most of our data was collected in a wooden arena with walls for preventing objects from falling down. All results in this paper are from data collected only from the wooden arena.

3.2 Method

The forward and inverse models can be formally described by equations 3.1 and 3.2, respectively. The notation is as following: x_t, u_t are the world state and action applied time step t, $\hat{x}_{t+1}, \hat{u}_{t+1}$ are the predicted state and actions, and W_{fwd} and W_{inv} are parameters of the functions F and G that are used to construct the forward and inverse models.

$$\hat{x}_{t+1} = F(x_t, u_t; W_{fwd}) \qquad (3.1) \qquad\qquad \hat{u}_t = G(x_t, x_{t+1}; W_{inv}) \qquad (3.2)$$

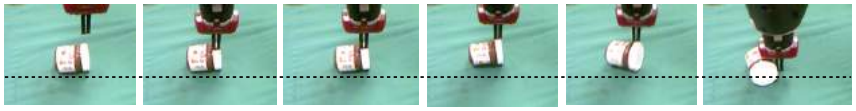


Figure 3.2: These images depict the robot in the process of displacing the bottle away from the indicated dotted line. In the middle of the poke, the object flips and ends up moving in the wrong direction. Such occurrences are common because the real world objects have complex geometric and material properties. This makes learning manipulation strategies without prior knowledge very challenging.

Given an initial and goal state, inverse models provide a direct mapping to actions required for achieving the goal state in one step (if feasible). However, multiple possible actions can transform the world from one visual state to another. For example, an object can appear in a certain part of the visual field if the agent moves or if the agent uses its arms to move the object. This multi-modality in the action space makes the learning hard. On the other hand, given x_t and u_t, there exists a next state x_{t+1} that is unique up to dynamics noise. This suggests that forward models might be easier to learn. However, learning forward models in image space is hard because predicting the value of each pixel in the future frames is a non-trivial problem with no known good solution. However, in most scenarios we are not interested in predicting every pixel, but predicting the occurrence of a more abstract event such as object motion, change in object pose etc.

The ability to learn an abstract task relevant feature space should make it easier to learn a forward dynamics model. One possible approach is to learn a dynamics model in the feature representation of a higher layer of a deep neural network trained to perform image classification (say on ImageNet) [114]. However, this is not a general way of learning task relevant features and it is unclear whether features adept at object recognition are also optimal for object manipulation. The alternative of adapting higher layer features of a neural network while simultaneously optimizing for the prediction loss leads to a degenerate solution of all the features reducing to zero, since the prediction loss in this case is also zero. Our key observation is that this degenerate solution can be avoided by imposing the constraint that it should be possible to infer the the executed action (u_t) from the feature representation of two images obtained before (x_t) and after (x_{t+1}) the action (u_t) is applied (i.e. optimizing the inverse model). This formulation provides a general mechanism for using general purpose function approximators such as deep neural networks for simultaneously learning a task relevant feature space and forecasting the future outcome of actions in this learned space.

A second challenge in using forward models is that inferring the optimal action

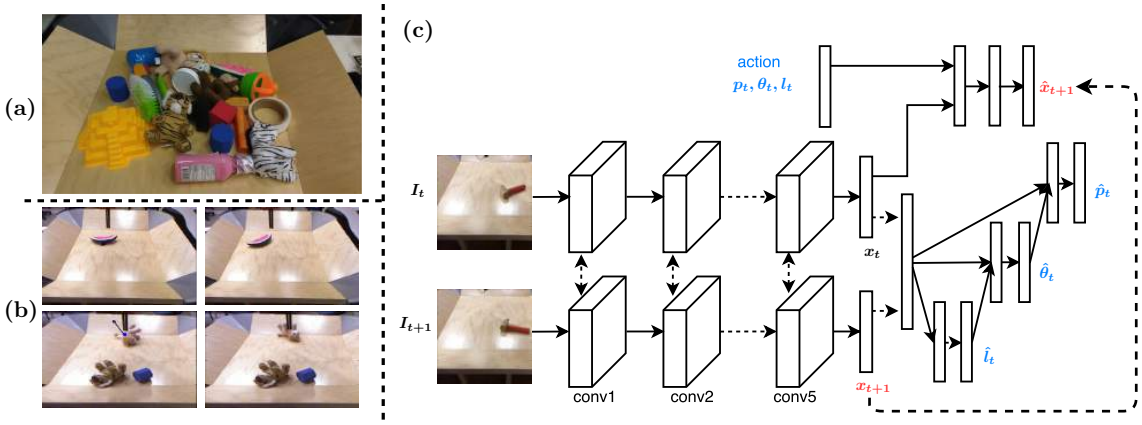


Figure 3.3: (a) The collection of objects in the training set poked by the robot. (b) Example pairs of before (I_t) and after images (I_{t+1}) after a single poke was made by the robot. (c) A Siamese convolutional neural network was trained to predict the poke location (p_t), angle (θ_t) and length (l_t) required to transform objects in the image at the t^{th} time step (I_t) into their state in I_{t+1}. Images I_t and I_{t+1} are transformed into their latent feature representations (x_t, x_{t+1}) by passing them through a series of convolutional layers. For building the inverse model, x_t, x_{t+1} are concatenated and passed through fully connected layers to predict the discretized poke. For building the forward model, the action $u_t = \{p_t, \theta_t, l_t\}$ and x_t are passed through a series of fully connected layers to predict x_{t+1}.

inevitably leads to finding a solution to non-convex problems that are subject to local optima. The inverse model does not suffers from this drawback as it directly outputs the required action. These considerations suggest that inverse and forward models have complementary strengths and therefore it is worthwhile to investigate training a joint model of inverse and forward dynamics.

3.2.1 Model

A deep neural network is used to simultaneously learn a model of forward and inverse dynamics (see Figure 3.3). A tuple of before image (I_t), after image (I_{t+1}) and the robot's action (u_t) constitute one training sample. Input images at consequent time steps (I_t, I_{t+1}) are transformed into their latent feature representations (x_t, x_{t+1}) by passing them through a series of five convolutional layers with the same architecture as the first five layers of AlexNet [5]. For building the inverse model, x_t, x_{t+1} are concatenated and passed through fully connected layers to conditionally predict the poke location (p_t), angle (θ_t) and length (l_t) separately. For modeling multimodal poke distributions, poke location, angle and length of poke are discretized into a

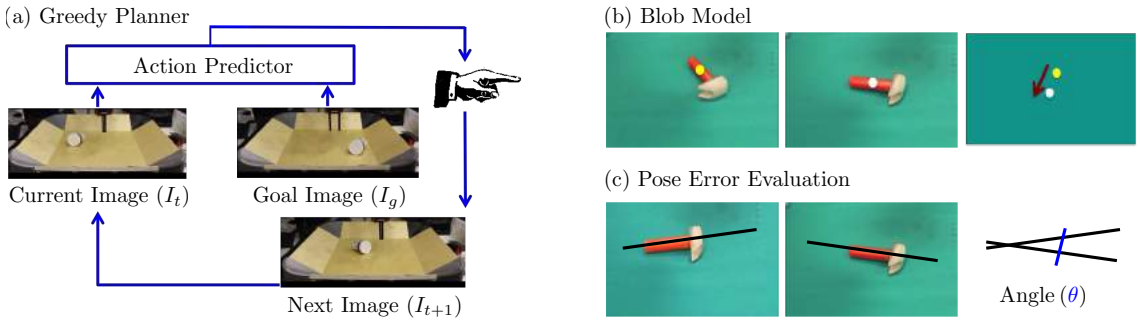


Figure 3.4: (a) Greedy planner is used to output a sequence of pokes to displace the objects from their configuration in initial to the goal image. (b) The blob model first detects the location of objects in the current and goal image. Based on object positions, location and angle of the poke is computed and then executed by the robot. The obtained next and goal image are used to compute the next poke and this process is repeated iteratively. (c) The error of the models in poking objects to their correct pose is measured as the angle between the major axis of the objects in the final and goal images.

20×20 grid, 36 bins and 11 bins respectively. The 11^{th} bin of the poke length is used to denote no poke. For building the forward model, the feature representation of the before image (x_t) and the action (u_t; real-valued vector without discretization) are passed into a sequence of fully connected layer that predicts the feature representation of the next image (x_{t+1}). Training is performed to optimize the loss defined in equation 3.3 below.

$$L_{joint} = L_{inv}(u_t, \hat{u_t}, W) + \lambda L_{fwd}(x_{t+1}, \hat{x}_{t+1}, W) \tag{3.3}$$

L_{inv} is a sum of three cross entropy losses between the actual and predicted poke location, angle and length. L_{fwd} is a L1 loss between the predicted ($\hat{x}_{t+1}$) and the ground truth (x_{t+1}) feature representation of the after image (I_{t+1}). W are the parameters of the neural network. We used $\lambda = 0.1$ in all our experiments. We call this the joint model and we compare its performance against the inverse only model that was trained by setting $\lambda = 0$ in equation 3.3. More details about model training are provided in the supplementary materials.

3.2.2 Evaluation Procedure

One way to test the learnt model is to provide the robot with an initial and goal image and task it to apply pokes that would displace objects into the configuration

shown in the goal image. If the robot succeeds at achieving the goal configuration when the visual statistics of the pair of initial and goal image is similar to before and after image in the training set, then this would not be a convincing demonstration of generalization. However, if the robot is able to displace objects into goal positions that are much farther apart as compared to position of objects before and after a single poke then it might suggest that our model has not simply overfit but has learnt something about the underlying physics of how objects move when poked. This suggestion would be further strengthened if the robot is also able to push objects with novel geometry and texture in presence of multiple distractor objects.

If the objects in the initial and goal image are farther apart than the maximum distance that can be pushed by a single poke, then the model would be required to output a sequence of pokes. We use a greedy planning method (see Figure 3.4(a)) to output a sequence of pokes. First, images depicting the initial and goal state are passed through the learnt model to predict the poke which is then executed by the robot. Then, the image depicting the current world state (i.e. the current image) and the goal image are fed again into the model to output a poke. This process is repeated iteratively unless the robot predicts a no-poke (see section 3.2.1) or a maximum number of 10 pokes is reached.

Error Metrics: In all our experiments, the initial and goal images differ in the position of only a single object. The location and pose of the object in the final image after the robot stops and the goal image are compared for quantitative evaluation. The location error is the Euclidean distance between the object locations. In order to account for different object distances in the initial and goal state, we use relative instead of absolute location error. Pose error is defined as the angle (in degrees) between the major axis of the objects in the final and goal images (see Figure 3.4(c)). Please see supplementary materials for further details.

3.2.3 Blob Model

We compared the performance of the learnt model against a baseline blob model. This model first estimates object locations in current and goal image using template based object detector. It then uses the vector difference between these to compute the location, angle and length of poke executed by the robot (see supplementary materials for details). In a manner similar to greedy planning with the learnt model, this process is repeated iteratively until the object gets closer to the desired location in the goal image by a pre-defined threshold or a maximum number of pokes is reached.

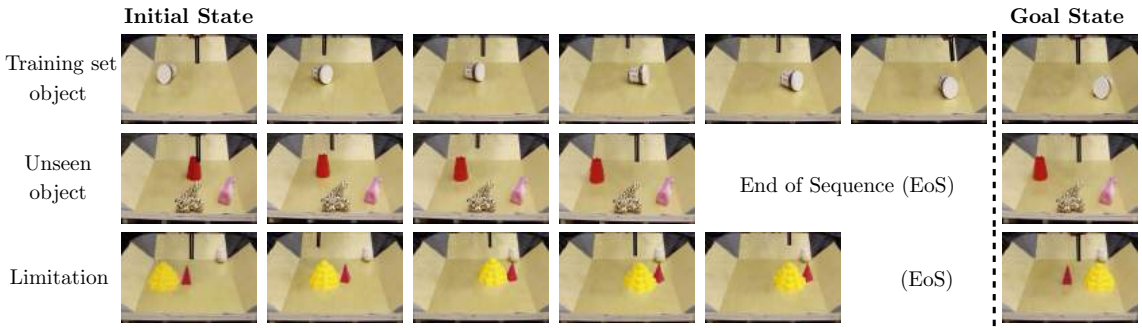


Figure 3.5: The robot is able to successfully displace objects in the training set (row 1; Nutella bottle) and objects with previously unseen geometry (row 2; red cup) into goal locations that are significantly farther than pair of before and after images used in the training set. The robot is unable to push objects around obstacles (row 3; limitation of greedy planning).

3.3 Results

The robot was tasked to displace objects in an initial image into their configuration depicted in a goal image (see Figure 3.5). The three rows in the figure show the performance when the robot is asked to displace an object (Nutella bottle) present in the training set, an object (red cup) whose geometry is different from objects in the training set and when the task is to move an object around an obstacle. These examples are representative of the robot's performance and more examples can be found on the project website. It can be seen that the robot is able to successfully poke objects present in the training set and objects with novel geometry and texture into desired goal locations that are significantly farther than pair of before and after images used in the training set.

Row 2 in Figure 3.5 also shows that the robot's performance in unaffected by the presence of distractor objects that occupy the same location in the current and goal images. These results indicate that the learnt model allows the robot to perform tasks that show generalization beyond the training set (i.e. poking object by small distances). Row 3 in Figure 3.5 depicts an example where the robots fails to push the object around an obstacle (yellow object). The robot acts greedily and ends up pushing the obstacle along with the object. One more side-effect of greedy planning is zig-zag instead of straight trajectories taken by the object between its initial and goal locations. Investigating alternatives to greedy planning, such as using the learnt forward model for planning pokes is a very interesting direction for future research.

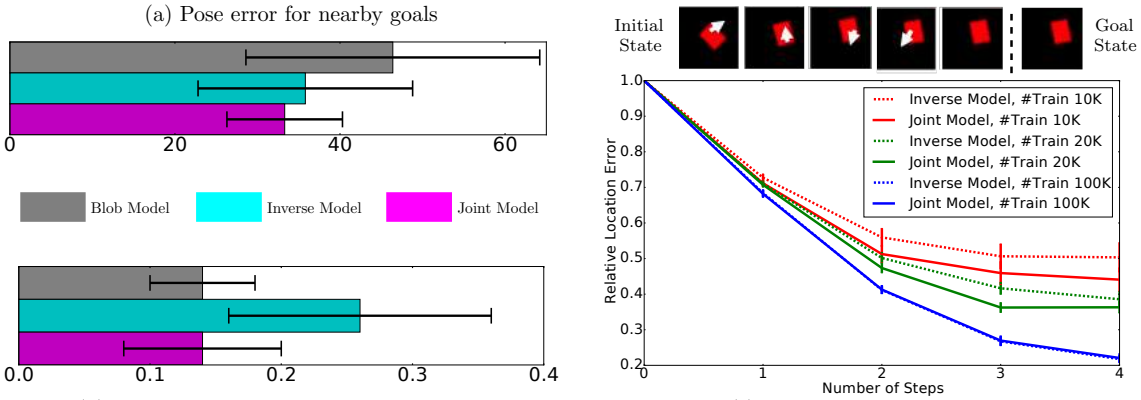
What representation could the robot have learnt that allows it to generalize?

One possibility is that the robot ignores the geometry of the object and only infers the location of the object in the initial and goal image and uses the difference vector between object locations to deduce what poke to execute. This strategy is invariant to absolute distance between the object locations and is therefore capable of explaining the observed generalization to large distances. While we cannot prove that the model has learnt to detect object location, nearest neighbor visualizations of the learnt feature space clearly suggest sensitivity to object location (see supplementary materials). This is interesting because the robot received no direct supervision to locate objects.

Because different objects have different geometries, they need to be poked at different places to move them in the same manner. For example, a Nutella bottle can be reliably moved forward without rotating the bottle by poking it on the side along the direction toward its center of mass, whereas a hammer is reliably moved by poking it where the hammer head meets the handle. Pushing an object to a desired pose is harder and requires a more detailed understanding of object geometry in comparison to pushing the object to a desired location. In order to test whether the learnt model represents any information about object geometry, we compared its performance against the baseline blob model (see section 3.2.3 and figure 3.4(b)) that ignores object geometry. For this comparison, the robot was tasked to push objects to a nearby goal by making only a single poke (see supplementary materials for more details). Results in Figure 3.6(a) show that both the inverse and joint model outperform the blob model. This indicates that in addition to representing information about object location, the learn models also represent some information about object geometry.

3.3.1 Forward model regularizes the inverse model

We tested the hypothesis whether the forward model regularizes the feature space learnt by the inverse model in a 2-D simulation environment where the agent interacted with a red rectangular object by poking it by small forces. The rectangle was allowed to freely translate and rotate (Figure 3.6(c)). Model training was performed using an architecture similar to the one described in section 3.2.1. Additional details about the experimental setup, network architecture and training procedure for the simulation experiments are provided in the supplementary materials. Figure 3.6(c) shows that when less training data (10K, 20K examples) is available the joint model outperforms the inverse model and reaches closer to the goal state in fewer steps (i.e. fewer actions). This shows that indeed the forward model regularizes the inverse model and helps generalize better. However, when the number of training examples is increased to 100K both models are at par. This is not surprising because



(a) Pose error for nearby goals

(b) Relative location error for far away goals

(c) Simulation experiments

Figure 3.6: (a) Inverse and Joint model are more accurate than the blob model at pushing objects towards the desired pose. (b) The joint model outperforms the inverse-only model when the robot is tasked to push objects by distances that are significantly larger than object distance in before and after images used in the training set (i.e. a test of generalization). (c) Simulation studies reveal that when less number of training examples (10K, 20K) are available the joint model outperforms the inverse model and the performance is comparable with larger amount of data (100K). This result indicates that the forward model regularizes the inverse model.

training with more data often results in better generalization and thus the inverse model is no longer reliant on the forward model for the regularization.

Evaluation on the real robot supports the findings from the simulation experiments. Figure 3.6(b) shows that in a test of generalization, when an object is required to be displaced by a long distance, the joint model outperforms the inverse model. Similar performance of joint and blob model at this task is not surprising because even if the pokes are somewhat inaccurate but generally in the direction from object's current to goal location, the object might traverse a zig-zag path but it would eventually reach the goal. The joint model is however more accurate at displacing objects into their correct pose as compared to the blob model (Figure 3.6(a)).

3.4 Related Work

Several recent works have proposed to learn visual control policies using reinforcement learning for tasks such as playing Atari games [3] and controlling robots in simulation [115] and in the real world [116]. However, these methods are model free and learn goal specific policies, which makes it difficult to repurpose the learned

policies for new tasks. In contrast, the aim of this work is to learn intuitive physical models of object interaction in an unsupervised manner. Such models can then be used to reach any target configuration.

A number of recent methods have also been proposed for learning representations for vision-based control using autoencoders to transform visual inputs into a low-dimensional state space [117–120]. However, these works have used model free methods as training auto encoders on complex real world imagery is difficult, and it is unclear that features obtained by optimizing pixelwise reconstruction are necessarily well suited for model based control. Recently [114] proposed to build prediction models in the space of features learnt by pretraining on image classification on Imagenet. Their approach assumes that invariances learnt for object recognition are also good for control. In contrast, our approach is entirely self-supervised, does not require human-provided labels and is capable of learning task specific features.

[121, 122] learn how to grasp objects by trial and error from a large number of attempts. These methods aim to acquire a policy for solving a single concrete task, while our work is concerned with learning a general predictive model that could be used to achieve a variety of goals at test time. Furthermore, poking is a type of nonprehensile manipulation (i.e. manipulation without grasping [74]). When an object is grasped, it is possible to fully control the state of the grasped object. With non prehensile manipulation, the state of the manipulated object is not directly controllable and thus less predictable. This makes the problem of achieving the goal state using non prehensile manipulation such as poking significantly more challenging than achieving the goal state by grasping [74, 123].

A good review of model based control can be found in [124] and [125, 126] provide interesting perspectives. [127] used deep learning based model predictive control for cutting vegetables. However, their system did not use vision and relied solely on the knowledge of the robotic state space and is thus limited in its generality. Only very recently, [79, 128–130] addressed the problem of model based control from vision in synthetic domains of manipulating two degree of freedom robotic arm, inverted pendulum, billiards and Atari games. In contrast to these works, we tackle manipulation of complex, compressible real world objects. [131–133] proposed using Newtonian physics in combination with neural networks to predict the dynamics of objects in the future. However, they do not test their models for goal directed actions. A second difference is that we use learn “intuitive" physics from data instead of relying on Newtonian physics for reasons mentioned in the first chapter.

In robotic manipulation, a number of prior methods have been proposed that use hand-designed visual features and known object poses or key locations to plan and execute pushes and other non-prehensile manipulations [134–136]. Unlike these methods, the goal in our work is to learn an intuitive physics model for pushing only

from raw images, thus allowing the robot to learn by exploring the environment on its own without human intervention.

3.5 Discussion

We presented a method for jointly learning the inverse and forward models for predicting the outcome of actions from raw visual inputs, as well as an empirical evaluation of an experiential approach to learning intuitive physics models in the context of robotic manipulation. In our experimental setup, we collected over 50 thousand interaction episodes to construct models that predict the outcome of robotic pokes. We then evaluated our learned models by using them to choose pokes that will push an object into a target configuration. Our results indicate that the proposed joint inverse and forward models attain higher accuracy than more standard methods, and also that active data collection, where the robot sets and attempts to reach its own goals using the latest learned model, produces more accurate predictive models.

Nonprehensile manipulation is a hard problem, and although our models perform better than the baseline methods, they are far from perfect. While poking, the robot does not have full control of the object state, which makes it harder to predict and plan for the outcome of the action. For example, depending on whether the poke is slightly on the left or slightly on the right of a simple cuboidal object, the resulting motion can be substantially different (i.e. clockwise or anti-clockwise). With real world objects having complex geometry, this learning problem becomes even harder. An interesting area of future investigation is to use continuous time control with smaller pokes that are likely to be more predictable than the large pokes used in this work.

Without any a priori knowledge and information about goals the robot has no incentive to learn pokes that are more reliable at displacing objects from one configuration to another. As there are multiple ways of displacing objects, this issue can lead to non-robust planning solutions that are likely to fail. For addressing this concern, we used active data sampling, which we show leads to better results. The intuition behind this is as following: the robot chooses a desired target image (x_{target}), and then uses the learned model to predict the poke (a) that will transform the objects in the current image (x_{before}) into the target configuration. The triplet of previous state (x_{before}), the poke (a), and the new state (x_{after}) that is obtained after executing the predicted poke is used as training data to improve the model. This process reinforces pokes that reliably transform x_{before} into x_{target} (i.e if $x_{after} \approx x_{target}$). However, if x_{after} is substantially different from x_{target}, there is no direct feedback that biases the model into believing that action a is unreliable for achieving x_{target}.

An interesting direction for future research is to either resolve this limitation by incorporating the predictions of the forward model into the active sampling scheme, or come up with alternative strategies or constraints that will bias the robot towards learning reliable actions.

A neural network formulation for simultaneously learning and forecasting in an abstract feature space is proposed in this work. We show that the forward model regularizes the inverse model, but it might be possible that alternative loss functions are even better regularizers. Also, using forward models just as regularizers is unsatisfactory because forward models equip the agent with the ability to perform rollouts, and consequently perform planning with longer horizons. This is in contrast to inverse models that can be used for multistep execution only by executing actions one after another, without any foresight. A detailed investigation of how forward models can be effectively used for planning in the proposed framework is currently pending and is an interesting avenue for future work.

Our experiments show generalization to the location and pose of objects, but we have not shown any generalization to new objects. The primary reason is that data collection on a real robot is slow and until now we have only collected data for four objects. However, this is not a limitation of our approach as data collection process can be easily parallelized across robots [122]. An open research question in data collection is whether we should collect large amounts of data for few objects or small amounts of data for a large number of objects. Future work addressing this question and showing generalization to new objects would be of great interest.

In this work we have proposed the use of intuitive models of physics learned from experience and simultaneous learning of forward and inverse models for vision based control. Although our approach is evaluated on a specific robotic manipulation task, there are no task specific assumptions, and the techniques are applicable to other tasks. In future, it would be interesting to see how the proposed approach scales with more complex environments, diverse object collections, different manipulation skills and to other non-manipulation based tasks, such as navigation.

In this work we propose to learn “intuitive" model of physics using interaction data. An alternative is to represent the world in terms of a fixed set of physical parameters such as mass, friction coefficient, normal forces etc and use a physics simulator for computing object dynamics from this representation [131, 132, 137, 138]. This approach is general because physics simulators inevitably use Newton’s laws that apply to a wide range of physical phenomenon ranging from orbital motion of planets to a swinging pendulum. Estimating parameters such as as mass, friction coefficient etc. from sensory data is subject to errors, and it is possible that one parameterization is easier to estimate or more robust to sensory noise than another. For example, the conclusion that objects with feather like appearance fall slower than

objects with stone like appearance can be reached by either correlating visual texture to the speed of falling objects, or by computing the drag force after estimating the cross section area of the object. Depending on whether estimation of visual texture or cross section area is more robust, one parameterization will result in more accurate predictions than the other. Pre-defining a set of parameters for predicting object dynamics, which is required by “simulator-based" approach might therefore lead to suboptimal solutions that are less robust.

For many practical object manipulation tasks of interest, such as re-arranging objects, cutting vegetables, folding clothes, and so forth, small errors in execution are acceptable. The key challenge is robust performance in the face of varying environmental conditions. This suggests that a more robust but a somewhat imprecise model may in fact be desirable over a less robust and a more precise model. While the arguments presented above suggest that intuitive physics models are likely to be more robust than simulator based models, quantifying the robustness of these models is an interesting direction for future work. Furthermore, it is non-trivial to use simulator based models for manipulating deformable objects such as clothes and ropes because simulation of deformable objects is hard and also also requires representing objects by heavily handcrafted features that are unlikely to generalize across objects. The intuitive physics approach does not make any object specific assumptions and can be easily extended to work with deformable objects. This approach is in the spirit of recent successful deep learning techniques in computer vision and speech processing that learn features directly from data, whereas the simulator based physics approach is more similar to using hand-designed features. Current methods for learning intuitive physics models, such as ours are data inefficient and it is possible that combining intuitive and simulator based approaches leads to better models than either approach by itself.

In poking based interaction, the robot does not have full control of the object state which makes it harder to predict and plan for the outcome of an action. The models proposed in this work generalize and are able to push objects into their desired location. However, performance on setting objects in the desired pose is not satisfactory, possibly because of the robot only executing pokes in large, discrete time steps. An interesting area of future investigation is to use continuous time control with smaller pokes that are likely to be more predictable than the large pokes used in this work. Further, although our approach is evaluated on a specific robotic manipulation task, there are no task specific assumptions, and the techniques are applicable to other tasks. In future, it would be interesting to see how the proposed approach scales with more complex environments, diverse object collections, different manipulation skills and to other non-manipulation based tasks, such as navigation. Other directions for future investigation include the use of forward model for planning

and developing better strategies for data collection than random interaction.

Chapter 4

Learning from Experts

In theory an autonomous continually learning agent should not rely on any expert supervision or communication from other agents to augment its knowledge. However our world is incredibly rich and diverse and there is simply too much to learn. In such a setup, an agent would greatly benefit if what it learns could be prioritized by an expert or be biased by observing other agents. Such prioritization/biasing would enable the agent to focus on learning relevant things.

There are multiple ways in which social learning or learning from experts can help. A family of algorithms under the umbrella term of "imitation" learning or learning by imitation have been widely used to teach agents to perform complex tasks. The current dominant paradigm in learning from demonstration (LfD) [62–65] requires the expert to either manually move the robot joints (i.e., kinesthetic teaching) or teleoperate the robot to execute the desired task. The expert typically provides multiple demonstrations of a task at training time, and this generates data in the form of observation-action pairs from the agent's point of view. The agent then distills this data into a policy for performing the task of interest. Such a heavily supervised approach, where it is necessary to provide demonstrations by controlling the robot, is incredibly tedious for the human expert. Moreover, for every new task that the robot needs to execute, the expert is required to provide a new set of demonstrations. Such approaches can be broadly categorized as:

- Behavior Cloning: Expert demonstrates multiple trajectories. A mapping from states (x_t) to actions (a_t) is then estimated using this data [139].
- Trajectory Learning: Multiple demonstrations are considered to be noisy samples of an ideal trajectory the a demonstrator wishes to communicate [140].
- Inverse Optimal Control: The demonstrated trajectories are used to estimate

a reward function that is optimized using a known/estimated model of the system.

- Inverse Reinforcement Learning: Similar to inverse optimal control, but the estimated reward function is optimized using reinforcement learning.

Using experts as a source of supervision for learning complex behaviors is only a small part of how an agent can learn from other agents. In developmental psychology it is well documented that infants engage in face-to-face interactions with their mothers and over a lifetime it is very common for humans to learn from their peers things about their environment that are not geared towards a particular end-task. There are broadly five ways in which an agent can learn from experts/other agents:

1. Demonstrations of complex tasks.
2. Learning by passive observation of other agents [66].
3. Guided Exploration: Another agent provides cues on what to explore. This can be considered as a case of providing weak supervision, where instead of teaching an agent how to perform the task, the teacher provides hints about what to explore. Such exploration may or may not be immediately useful to the agent's goals.
4. Active Imitation Learning: Instead of simply obtaining supervision from the expert, the agent can query the expert as needed [141].
5. Learning about other Agents: Instead of learning a particular task, the agent's goal is to learn as much as possible about another agent. This process can either enhance the knowledge of the agent or help in collaborative tasks with the other agent later in time.

While there has been substantial research in (1), research in other problem setups is much more limited. I will first present a framework for learning by observation that enables all these forms of learning and present results on the task of rope manipulation and indoor navigation. The formulations and results in this chapter have been taken from my previous publications [86, 87].

4.1 A Framework for Learning by Observation

Rizzolatti et al. [142, 143] serendipitously discovered that some neurons in the area F5 of the premotor cortex that fire when a macaque performs an action; also

fire when the macaque does not perform the action, but simply observes a similar action being performed by another macaque. Such neurons were termed as *mirror neurons*. It was hypothesized that *mirror neurons* enable an animal to interpret the actions of another animal by firing a group of neurons that would have fired if the animal itself performed the task. In a more abstract sense, it means that an agent can understand what another agent is doing by internally simulating how it would perform the same actions as the another agent. While there is some controversy in the research community about the specifics of mirror neurons [144], they do provide a powerful idea for action understanding.

In the most general formulation, let $\mathcal{S}^{\wr} : \{x_1^o, x_2^o, ..., x_T^o\}$ be the sequence of raw sensory observations of an agent makes of the other agent. If the agent possesses an internal model that outputs a sequence of actions (a_τ; called as *skill*) required to transition between a pair of observations $\{x_t^o, x_{t+k}^o\}$ (i.e., $a_\tau = \pi(x_t^o, x_{t+k}^o)$), then it could employ this model/*skill* to either repeat the demonstration (i.e. way #1, #2) or if the agent is uncertain of its ability to repeat (because either its model is imperfect or it was demonstrated a task that required skills it doesnot posses), it could either ask for help from the other agent (i.e. way #4) or autonomously explore its environment to become more capable of repeating what was demonstrated (i.e. way #3). Note that this framing of the problem enables the agent to either stitch skills it already possesses to perform new tasks or collect data by exploration to learn newer skills as a result of observing another agent.

There are two components of the above formulation: (a) skill learning; (b) using skills to imitate/explore. I will now describe them.

4.1.1 Learning a Model to Imitate

Let $\mathcal{S} : \{x_1, a_1, x_2, a_2, ..., x_T\}$ be the sequence of observations and actions generated by the agent as it explores its environment using the policy $a = \pi_E(s)$. This exploration data is used to learn the goal-conditioned skill policy (GSP) π takes as input a pair of observations (x_i, x_g) and outputs sequence of actions $(\vec{a}_\tau : a_1, a_2...a_K)$ required to reach the goal observation (x_g) from the current observation (x_i).

$$\vec{a}_\tau = \pi(x_i, x_g; \theta_\pi) \tag{4.1}$$

where states x_i, x_g are sampled from the $\mathcal{S}$. The number of actions, K, is also inferred by the model. We represent π by a deep network with parameters θ_π in order to capture complex mappings from visual observations (x) to actions. π can be thought of as a variable-step generalization of the inverse dynamics model [125], or as the policy corresponding to a universal value function [145, 146], with the difference that x_g need not be the end goal of a task but can also be an intermediate sub-goal.

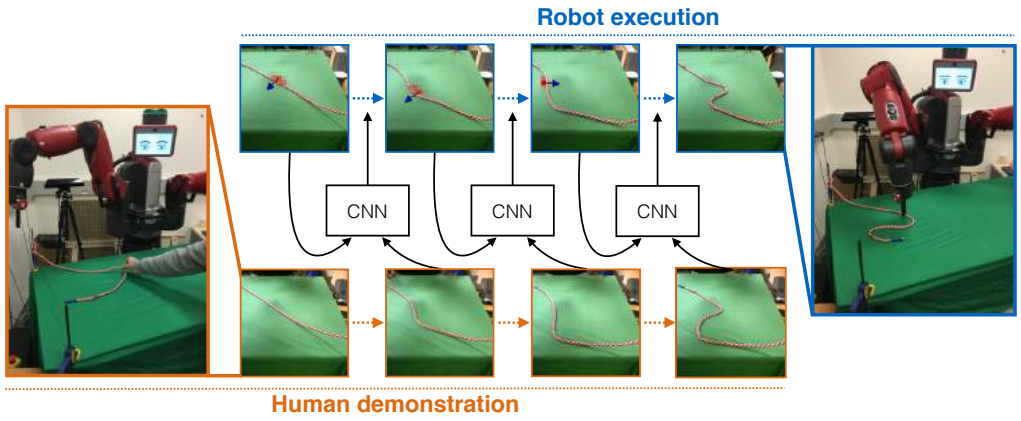


Figure 4.1: We present a system where the robot is capable of manipulating a rope into target configurations by combining a high-level plan provided by a human with a learned low-level model of rope manipulation. A human provides the robot with a sequence of images recorded while he manipulates the rope from an initial to goal configuration. The robot uses a learned inverse dynamics model to execute actions to follow the demonstrated trajectory. The robot uses a convolutional neural network (CNN) for learning the inverse model in a self-supervised manner using 60K interactions with the rope with no human supervision. The red heatmap on each image of the robot's execution trace shows the predicted location of the *pick* action and the blue arrow shows the direction of the action. This image is best seen in color.

4.2 Imitating Visual Demonstrations

Let the task to be imitated be provided as a sequence of images $\mathcal{D} : \{x_1^d, x_2^d, ..., x_N^d\}$ captured when the expert demonstrates the task. This sequence of images $\mathcal{D}$ could either be temporally dense or sparse. Our agent uses the learned GSP π to imitate the sequence of visual observations $\mathcal{D}$ starting from its initial state x_0 by following actions predicted by $\pi(x_0, x_1^d; \theta_\pi)$. Let the observation after executing the predicted action be x_0'. Since multiple actions might be required to reach close to x_1^d, the agent queries a separate *goal recognizer* network to ascertain if the current observation is close to the goal or not. If the answer is negative, the agent executes the action $a = \pi(x_0', x_1^d; \theta_\pi)$. This process is repeated iteratively until the *goal recognizer* outputs that agent is near the goal, or a maximum number of steps are reached. Let the observation of the agent at this point be $\hat{x}_1$. After reaching close to the first observation (x_1^d) in the demonstration, the agent sets its goal as (x_2^d) and repeats the process. The agent stops when all observations in the demonstrations are processed.

4.2.1 Goal Recognizer

We train a goal recognizer network to figure out if the current goal is reached and therefore allow the agent to take variable numbers of steps between goals. Goal recognition is especially critical when the agent has to transit through a sequence of intermediate goals, as is the case for visual imitation, as otherwise compounding error could quickly lead to divergence from the demonstration. This recognition is simple given knowledge of the true physical state, but difficult when working with visual observations. Aside from the usual challenges of visual recognition, the dependence of observations on the agent's own dynamics further complicates goal recognition, as the same goal can appear different while moving forward or turning during navigation.

We pose goal recognition as a binary classification problem that given an observation x_i and the goal x_g infers if x_i is close to x_g or not. Lacking expert supervision of goals, we draw goal observations at random from the agent's experience during exploration, since they are known to be feasible. For each such pseudo-goal, we consider observations that were only a few actions away to be positives (i.e., close to the goal) and the remaining observations that were more than a fixed number of actions (i.e., a margin) away as negatives. We trained the goal classifier using the standard cross-entropy loss. Like the skill policy, our goal recognizer is conditioned on the goal for generalization across goals. We found that training an independent goal recognition network consistently outperformed the alternative approach that augments the action space with a "stop" action. Making use of temporal proximity as supervision has also been explored for feature learning in the concurrent work of [147].

4.3 Evaluation Procedure

The performance of the robot was evaluated by tasking it to reconfigure the rope from a given initial configuration into target configurations of varying complexity depicting "L", "S", "W" shapes and a knot.

The performance was quantitatively measured by calculating the distance between the rope configurations in the sequence of images provided from the human demonstration and the sequence of images achieved by the robot after executing the actions from the inverse dynamics model. The distance between two rope configurations is computed by first segmenting the rope in each image, then aligning points in these two segmentations with thin plate spline robust point matching (TPS-RPM) [148] and calculating the mean pixel distance between the matched points. We compare the performance of our method against a hand-engineered baseline, a nearest neigh-

bor baseline, and our method without imitation (see Section 9.3.1). Videos of the self-supervised data collection, the demonstrations, and autonomous executions are available at https://ropemanipulation.github.io/

4.3.1 Baseline

Hand-Engineered baseline: The first baseline we compare against is a hand-engineered method that takes as input the sequence of images from the human demonstration. For inferring which action should be executed to transform the rope from the configuration in I_t into the configuration in I_{t+1}, we first segment the rope in both the images, and use TPS-RPM to register the segments. In the absence of a model of rope dynamics, a simple way to move the rope into a target configuration is to *pick* the rope at the point with the largest deformation in the first image relative to the second and then *drop* the rope at the corresponding point in the second image. As the point with largest distance may be an outlier, we use the point at the 90^{th} percentile of the deformation distances for the *pick* action.
Nearest Neighbor baseline: To evaluate whether the neural network simply memorized the training data, we compared our method to a nearest neighbor baseline. Given the current image (I_t) and the target image in the human demonstration (I'_{t+1}), a pair of images (I_k, I_{k+1}) in the training set that is closest to (I_t, I'_{t+1}) is determined and the ground truth action used to collect this training sample is executed by the robot. As the distance metric for nearest neighbor calculation, we used Euclidean distance in raw RGB space after down-sampling images to 32×32.
No Imitation baseline: For understanding how critical is imitation for manipulating the rope into desired shape, we evaluated the learned model by feeding in only the initial and goal image (I_1, I'_T) without any intermediate steps. The inverse model takes these images as inputs and the robot executes the predicted action and image I_2 is obtained. Next, the pair (I_2, I'_T) is fed into the inverse model to infer the next action to execute and this process is repeated iteratively for the same number of steps as in the human demonstration.

4.4 Results

Figure 4.2 qualitatively shows that using the learned inverse dynamics model, the robot is capable of re-configuring the rope into many different shapes. It can also be seen that when the rope is not bent sharply, the robot is more accurate at manipulation.

Figure 4.3 compares the performance of our method against the hand-engineered,

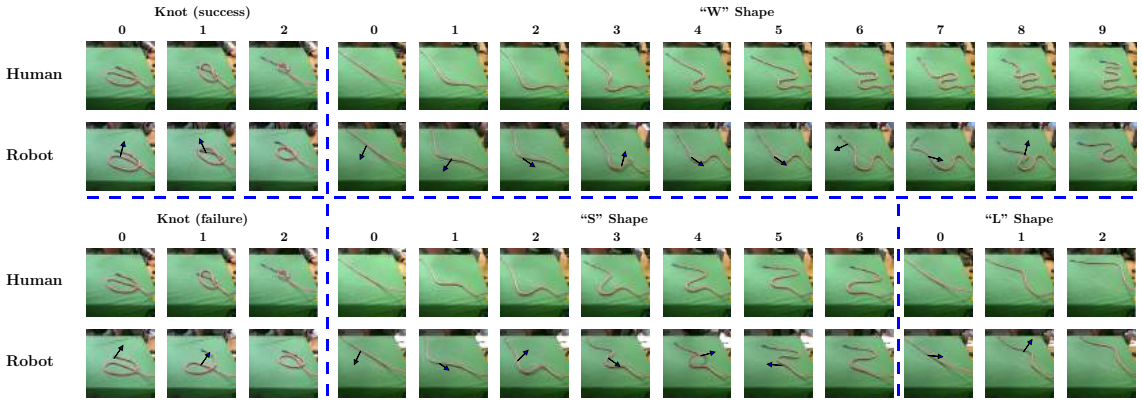


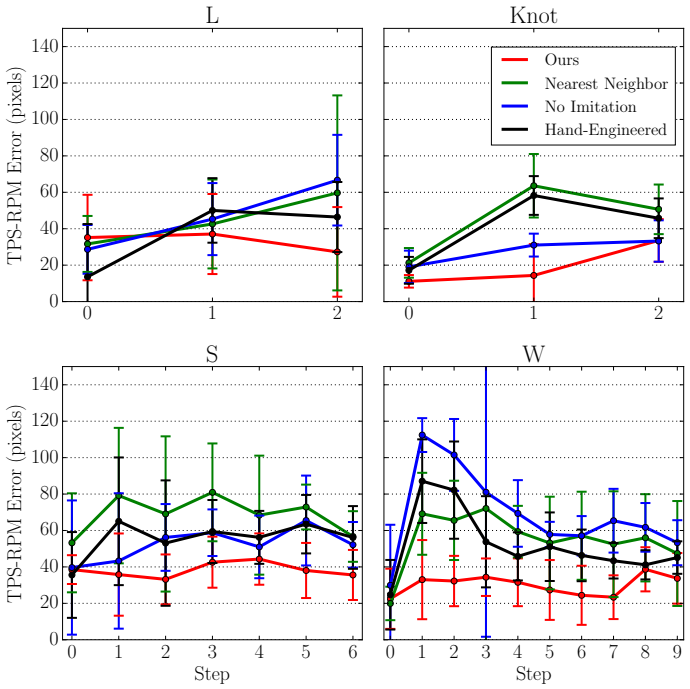
Figure 4.2: Qualitative comparison of the robot's performance in imitating the human demonstration for arranging the rope into *W, S, L* and *knot* configurations. The upper row in each example shows the sequence of demonstration images provided as input to the robot and the second (lower) row shows the states achieved by the robot as it tries to follow the demonstrated trajectory. The blue arrow on each image of the robot's execution trace shows the direction and the location of the pick point of the action performed by the robot. Please see the supplementary materials on the project website for more examples.

nearest neighbor and no-imitation baselines described in Section 9.3.1. Each subfigure shows the performance for a different target configuration. The x-axis in each subfigure corresponds to the number of intermediate images that were provided to the robot via demonstration. The y-axis corresponds to the TPS-RPM distance between the rope configuration achieved by the robot and the corresponding rope configuration in the actual demonstration. Lower values indicate better performance.

For every sequence, mean accuracy and standard deviation are reported across 10 different repeats of two human demonstration sequences. The results demonstrate that our method outperforms various baselines including a hand-engineered baseline (i.e. TPS-RPM baseline), indicating that through self-supervision, the robot has learned a dynamics model of rope motion that is useful for manipulation.

4.4.1 Importance of Imitation

How important are human demonstrations for manipulating the rope into a desired configuration? Results in Figure 4.3 show that when the robot was only provided the initial and final images instead of all the images in the human demonstration sequence, the performance is significantly worse. Figure 4.3 further shows that without imitation robot is able to tie the knot only 11/50 times instead of 19/50



Knot Tying Success

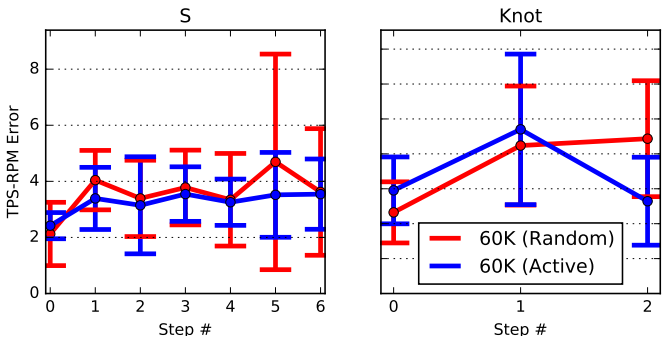
Imitation?	30K	60K
Yes	6/50	19/50
No	5/50	11/50

Figure 4.3: Performance comparison of the proposed method against the baselines described in Section 4.3.1. Performance is measured as the TPS-RPM distance metric between the rope configurations in the sequence of images achieved by the robot and the ones provided by human demonstration. Lower distance indicates better performance. Each subplot shows performance for a different target shape. Our method outperforms the baseline methods.

times with imitation.

4.4.2 Generalization to other ropes

We tested if the learned model is successfully able to manipulate new ropes by qualitatively evaluating performance on a white jute rope that was significantly stiffer than the red rope used in the training and a black cotton rope that was significantly thinner and less stiff than the red rope used in the training. We found that the robot was successfully able to configure these ropes into relatively simpler “L" and “S" shapes. Even though the model was trained using interaction data from a single rope, it generalizes to other ropes. This shows that instead of learning features specific to the particular rope used for data collection, our model learns features that generalize to other ropes.



Knot Tying Success

Random	Active
13/50	19/50

Figure 4.4: Using active data collection improves our capability in manipulating the rope into desired shapes and tying knots.

One possible reason that the robot is unsuccessful at manipulating the white rope into more curvy “W" and knot configurations is that the rope is too stiff to bend into curves necessary for forming the “W" and the knot. With the black rope the robot was able to successfully manipulate the rope until the penultimate step in tying a knot, but failed at completing the knot. We also ran experiments where we changed the background from green to white and found that our model was unsuccessful at rope manipulation. This result is not surprising because all our training data is collected using a single background and it is expected that with training data collected on diverse backgrounds our method would be able to generalize to novel backgrounds. The video demonstrations of these experiments are available at the project website.

4.5 Expert Guided Exploration

Instead of providing a complete demonstration of how to perform the task, sometimes a human/expert might simply provide a set of states which it deems would be useful for an agent to explore. Can the agent make use of such hints to improve its performance on a given end task? With randomized data collection it is unlikely that the robot would place the rope in interesting configurations and therefore the model may not be accurate at performing tasks such as knot-tying that require moving the rope into complex configurations. In order to bias data collection towards interesting configurations, we collected a set of 50 images of the rope when it was manually arranged in a random configuration (i.e. the goal buffer). We then used a model trained with 30K randomly collected data points and instead of randomly sampling an action, we randomly sampled a image from the goal buffer

and set that as the goal image. We passed the current and goal image into the inverse model and used the action predicted by the inverse model for data collection.

Results in Figure 4.4 show that performance on knot tying improves to 38% from 26% using the biased data, thus indicating that using the strategy mentioned above the agent was indeed able to bias its exploration in a manner that improved end-task performance.

4.6 Related Work

Imitation Learning The two main threads of imitation learning are behavioral cloning [62, 63], which directly supervises the mapping of states to actions, and inverse reinforcement learning [65, 122, 149–151], which recovers a reward function that makes the demonstration optimal (or nearly optimal). Inverse RL is most commonly achieved with state-actions, and is difficult to extend to fitting the reward to observations alone, though in principle state occupancy could be sufficient. Recent work in imitation learning [152–154] can generalize to novel goals, but require a wealth of demonstrations comprised of expert state-actions for learning. Our approach does not require expert actions at all.

Visual Demonstration The common scenario in LfD is to assume full knowledge of expert states and actions during demonstrations, but several papers have focused on relaxing this supervision to visual observations alone. [86] observe a sequence of images from the expert demonstration for performing rope manipulations. [147, 155] imitate humans with robots by self-supervised learning but require expert supervision at training time. Third person imitation learning [156] and the concurrent work of imitation-from-observation [157] learn to translate expert observations into agent observations such that they can do policy optimization to minimize the distance between the agent trajectory and the translated demonstration, but they require demonstrations for learning. Visual servoing is a standard problem in robotics [158] that seeks to take actions that align the agent's observation with a target configuration of carefully-designed visual features [159, 160] or raw pixel intensities [161]. Classical methods rely on fixed features or policies, but more recently end-to-end learning has improved results [162, 163].

Manipulation of deformable objects has been of great interest to the robotics community [164]. Prior works have considered problems such as surgical suturing [165, 166], towel folding [167], knot tying and rope manipulation among many others. Rope manipulation and knot tying are most closely related to our work. Inoue et al. [168] investigated the problem of knot tying and following works used motion planning [169], fixtures for open-loop execution [170] and robotic hands

with tactile feedback [171]. Morita et al. [172] developed a system for tying knots from visual observations that makes use of knot theory [173] to identify a sequence of knot states and then execute motor primitives to achieve these states. Wakamatsu et al. [174] chose a sequence of robotic hand motion primitives from rope cross states inferred by a planner to achieve a desired knot from a given initial state of the rope. In contrast to these works, our goal is not to tie knots but to manipulate rope into a general configuration by watching a human as she manipulates the rope. Our system does not require any rope-specific knowledge and is therefore generalizable to manipulating other deformable objects.

Chapter 5

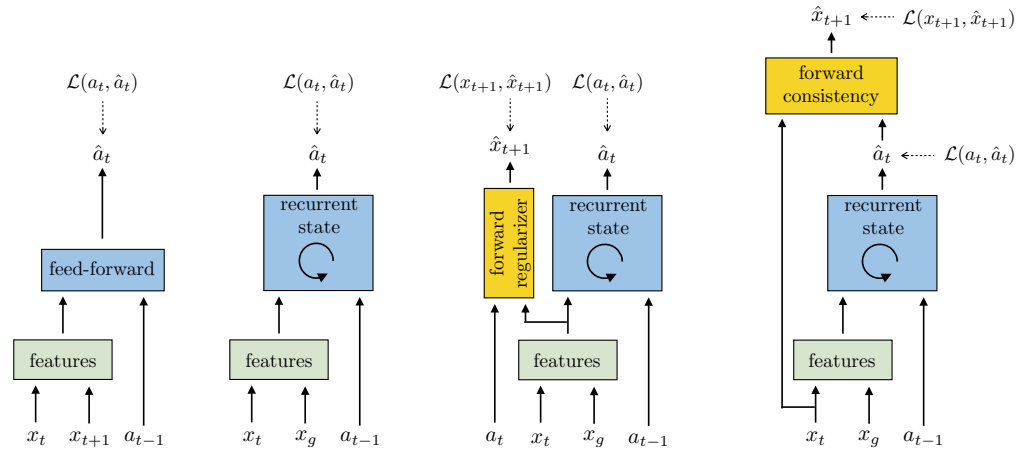
Revisting Forward and Inverse Models

In the previous chapter we used GSP for imitation. One critical challenge in learning the GSP is that, in general, there are multiple possible ways of going from one state to another: that is, the distribution of trajectories between states is multi-modal. We address this issue with our novel *forward consistency loss* based on the intuition that, for most tasks, reaching the goal is more important than how it is reached. To operationalize this, we first learn a forward model that predicts the next observation given an action and a current observation. We use the difference in the output of the forward model for the GSP-selected action and the ground truth next state to train the GSP. This loss has the effect of making the GSP-predicted action *consistent* with the ground-truth action instead of exactly matching the actions themselves, thus ensuring that actions that are different from the ground-truth—but lead to the same next state—are not inadvertently penalized.

In addition to improving the GSP model, in this chapter we also investigate how data collected from curiosity-driven exploration strategies affects the quality of learned models as compared to random exploration.

5.1 Forward Consistency Loss

One way to account for multi-modality is by employing the likes of variational auto-encoders [40, 175]. However, in many practical situations it is not feasible to obtain ample data for each mode. In this work, we propose an alternative based on the insight that in many scenarios, we only care about whether the agent reached the final state or not and the exact trajectory is of lesser interest. Instead of penalizing



(a) Inverse Model (b) Multi-step GSP (c) Forward-regularized GSP (d) Forward-consistent GSP (ours)

Figure 5.1: The goal-conditioned skill policy (GSP) takes as input the current and goal observations and outputs an action sequence that would lead to that goal. We compare the performance of the following GSP models: (a) Simple inverse model; (b) Mutli-step GSP with previous action history; (c) Mutli-step GSP with previous action history and a forward model as regularizer, but no forward consistency; (d) Mutli-step GSP with forward consistency loss proposed in this work.

the actions predicted by the GSP to match the ground truth, we propose to learn the parameters of GSP by minimizing the distance between observation $\hat{x}_{t+1}$ resulting by executing the predicted action $\hat{a}_t = \pi(x_t, x_{t+1}; \theta_\pi)$ and the observation x_{t+1}, which is the result of executing the ground truth action a_t being used to train the GSP. In this formulation, even if the predicted and ground-truth action are different, the predicted action will not be penalized if it leads to the same next state as the ground-truth action. While this formulation will not explicitly maintain all modes of the action distribution, it will reduce the variance in gradients and thus help learning. We call this penalty the *forward consistency loss*.

Note that it is not immediately obvious as to how to operationalize *forward consistency loss* for two reasons: (a) we need the access to a good *forward dynamics* model that can reliably predict the effect of an action (i.e., the next observation state) given the current observation state, and (b) such a dynamics model should be differentiable in order to train the GSP using the state prediction error. Both of these issues could be resolved if an analytic formulation of forward dynamics is known.

In many scenarios of interest, especially if states are represented as images, an

analytic forward model is not available. In this work, we learn the forward dynamics f model from the data, and is defined as $\tilde{x}_{t+1} = f(x_t, a_t; \theta_f)$. Let $\hat{x}_{t+1} = f(x_t, \hat{a}_t; \theta_f)$ be the state prediction for the action predicted by π. Because the forward model is not analytic and learned from data, in general, there is no guarantee that $\tilde{x}_{t+1} = \hat{x}_{t+1}$, even though executing these two actions, $a_t, \hat{a}_t$, in the real-world will have the same effect. In order to make the outcome of action predicted by the GSP and the ground-truth action to be *consistent* with each other, we include an additional term, $\|x_{t+1} - \hat{x}_{t+1}\|_2^2$ in our loss function and infer the parameters θ_f by minimizing $\|x_{t+1} - \tilde{x}_{t+1}\|_2^2 + \lambda \|x_{t+1} - \hat{x}_{t+1}\|_2^2$, where λ is a scalar hyper-parameter. The first term ensures that the learned forward model explains ground truth transitions (x_t, a_t, x_{t+1}) collected by the agent and the second term ensures consistency. The joint objective for training GSP with forward model consistency is:

$$\min_{\theta_\pi, \theta_f} \; \|x_{t+1} - \tilde{x}_{t+1}\|_2^2 + \lambda \|x_{t+1} - \hat{x}_{t+1}\|_2^2 \; + \; \mathcal{L}(a_t, \hat{a}_t) \tag{5.1}$$

$$\begin{aligned} \text{s.t.} \quad & \tilde{x}_{t+1} = f(x_t, a_t; \theta_f) \\ & \hat{x}_{t+1} = f(x_t, \hat{a}_t; \theta_f) \\ & \hat{a}_t = \pi(x_t, x_{t+1}; \theta_\pi) \end{aligned}$$

Note that learning θ_π, θ_f jointly from scratch is precarious, because the forward model f might not be good in the beginning, and hence could make the gradient updates noisier for π. To address this issue, we first pre-train the forward model with only the first term and GSP separately by blocking the gradient flow and then fine-tune jointly.

Generalization to feature space dynamics Past work has shown that learning forward dynamics in the feature space as opposed to raw observation space is more robust and leads to better generalization [85, 176]. Following these works, we extend the GSP to make predictions in feature representation $\phi(x_t), \phi(x_{t+1})$ of the observations x_t, x_{t+1} respectively learned through the self-supervised task of action prediction. The forward consistency loss is then computed by making predictions in this feature space ϕ instead of raw observations. The optimization objective for feature space generalization with mutli-step objective is shown in Equation (5.2).

Generalization to multi-step GSP We extend our one-step optimization to variable length sequence of actions in a straightforward manner by having a multi-step GSP π_m model with a step-wise forward consistency loss. The GSP π_m maintains an internal recurrent memory of the system and outputs actions conditioned on current observation x_t, starting from x_i to reach goal observation x_T. The forward consistency loss is computed at each time step, and jointly optimized with the action prediction loss over the whole trajectory. The final multi-step objective with feature

space dynamics is as follows:

$$\min_{\theta_\pi,\theta_f,\theta_\phi} \sum_{t=i}^{t=T} \Big(\|\phi(x_{t+1}) - \tilde{\phi}(x_{t+1})\|_2^2 \; + \; \lambda \|\phi(x_{t+1}) - \hat{\phi}(x_{t+1})\|_2^2 \; + \; \mathcal{L}(a_t, \hat{a}_t) \Big) \quad (5.2)$$

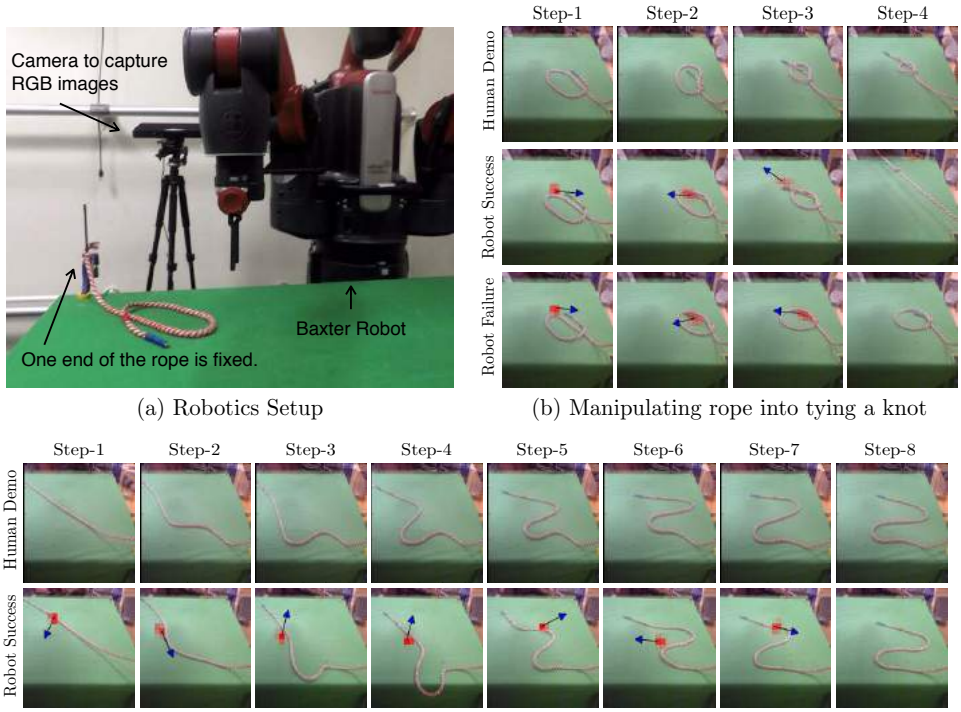
$$\begin{aligned} \text{s.t.} \quad & \tilde{\phi}(x_{t+1}) = f(\phi(x_t), a_t; \theta_f) \\ & \hat{\phi}(x_{t+1}) = f(\phi(x_t), \hat{a}_t; \theta_f) \\ & \hat{a}_t = \pi(\phi(x_t), \phi(x_T); \theta_\pi) \end{aligned}$$

where $\phi(.)$ is represented by a CNN with parameters θ_ϕ. The number of steps taken by the multi-step GSP π_m to reach the goal at inference is variable depending on the decision of goal recognizer; described in next subsection. Note that, in this objective, if ϕ is identity then the dynamics simply reduces to modeling in raw observation space. We analyze feature space prediction in *VizDoom* 3D navigation and stick to observation space in the rope manipulation and the office navigation tasks.

The multi-step *forward-consistent GSP* π_m is implemented using a recurrent network which at every step takes as input the feature representation of the current ($\phi(x_t)$) state, goal ($\phi(x_T)$) states, action at the previous time step (a_{t-1}) and the internal hidden representation h_{t-1} of the recurrent units and predicts $\hat{a}_t$. Note that inputting the previous action to GSP π_m at each time step could be redundant given that hidden representation is already maintaining a history of the trajectory. Nonetheless, it is helpful to explicitly model this history. This formulation amounts to building an auto-regressive model of the joint action that estimates probability $P(a_t|x_1, a_1, ...a_{t-1}, x_t, x_g)$ at every time step. It is possible to further extend our forward-consistent GSP π_m to build multi-step forward model, but we leave that direction of future work.

5.2 Experiments

We evaluate our model by testing its performance on: rope manipulation using Baxter robot, navigation of a wheeled robot in cluttered office environments, and simulated 3D navigation. The key requirements of a good skill policy are that it should generalize to unseen environments and new goals while staying robust to irrelevant distractors in the observations. For rope manipulation, we evaluate generalization by testing the ability of the robot to manipulate the rope into configurations such as knots that were not seen during random exploration. For navigation, both real-world and simulation, we check generalization by testing on a novel building/floor.



(a) Robotics Setup

(b) Manipulating rope into tying a knot

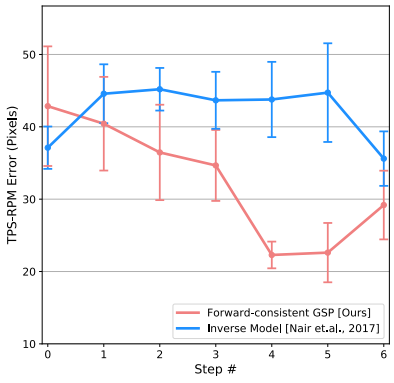
(c) Manipulating rope into 'S' shape

Figure 5.2: Qualitative visualization of results for rope manipulation task using Baxter robot. (a) Our robotics system setup. (b) The sequence of human demonstration images provided by the human during inference for the task of knot-tying (top row), and the sequences of observation states reached by the robot while imitating the given demonstration (bottom rows). (c) The sequence of human demonstration images and the ones reached by the robot for the task of manipulating rope into 'S' shape. Our agent is able to successfully imitate the demonstration.

5.2.1 Ablations and Baselines

Our proposed formulation of GSP composed of following components: (a) recurrent variable-length skill policy network, (b) explicitly encoding previous action in the recurrence, (c) goal recognizer, (d) forward consistency loss function, and (w) learning forward dynamics in the feature space instead of raw observation space. We systematically ablate these components of forward-consistent GSP, to quantitatively review the importance of each component and then perform comparisons to the prior approaches that could be deployed for the task of visual imitation.

The following methods will be evaluated and compared to in the subsequent experiments section: (1) Classical methods: In visual navigation, we attempted to compare against the state-of-the-art open source classical methods, namely, ORB-SLAM2 [177, 178] and Open-SFM [179]. (2) Inverse Model: [86] leverage vanilla



(a) TPS-RPM error for 'S' shape manipulation

Method	Success %
Inverse Model [Nair et.al. 2017]	36% ± 9.6%
Forward-regularized GSP	44% ± 9.9%
Forward-consistent GSP [Ours]	**60**% ± **9.8**%

(b) Success rate for Knot-tying

Figure 5.3: GSP trained using forward consistency loss significantly outperforms the baselines at the task of (a) manipulating rope into 'S' shape as measured by TPS-RPM error and (b) knot-tying where we report success rate with bootstrap standard deviation.

inverse dynamics to follow demonstration in rope manipulation setup. We compare to their method in both visual navigation and manipulation. (3) GSP-NoPrevAction-NoFwdConst is the ablation of our recurrent GSP without previous action history and without forward consistency loss. (4) GSP-NoFwdConst refers to our recurrent GSP with previous action history, but without forward consistency objective. (5) GSP-FwdRegularizer refers to the model where forward prediction is only used to regularize the features of GSP but has no role to play in the loss function of predicted actions. The purpose of this variant is to particularly ablate the benefit of consistency loss function with respect to just having forward model as feature regularizer. (6) GSP refers to our complete method with all the components. We now discuss the experiments and evaluate these baselines.

1) Goal Finding

We first tested if the GSP learned by the TurtleBot can enable it to find its way to a goal that is within the same room from just a *single image* of the goal. To test the extrapolative generalization, we keep the Turtlebot approximately 20-30 steps away from the target location in a way that current and goal observations have no overlap as shown in Figure **??**. We test the robot in an indoor office environment on a different floor that it has never encountered before. We judge the robot to be successful if it stops close to the goal and failure if it crashed into furniture or does not reach the goal within 200 steps. Since the initial and goal images have no overlap, classical techniques such as structure from motion that rely on feature matching cannot be used to infer the executed action. Therefore, in order to reach

Model Name	Run Id-1	Run Id-2	Run Id-3	Run Id-4	Run Id-5	Run Id-6	Run Id-7	Run Id-8	Num Success
Random Search	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	0
Inverse Model [Nair et. al. 2017]	Fail	Fail	Fail	Fail	Fail	Fail	Fail	Fail	0
GSP-NoPrevAction-NoFwdConst	39 steps	34 steps	Fail	Fail	Fail	Fail	Fail	Fail	2
GSP-NoFwdConst	22 steps	22 steps	39 steps	48 steps	Fail	Fail	Fail	Fail	4
GSP (Ours)	119 steps	66 steps	144 steps	67 steps	51 steps	Fail	100 steps	Fail	6

Table 5.1: Quantitative evaluation of various methods on the task of navigating using a *single image* of goal in an unseen environment. Each column represents a different run of our system for a different initial/goal image pair. Our full GSP model takes longer to reach the goal on average given a successful run but reaches the goal successfully at a much higher rate.

the goal, the robot must explore its surroundings. We find that our GSP model outperforms the baseline models in reaching the target location. Our model learns the exploratory behavior of rotating in place until it encounters an overlap between its current and goal image. Results are shown in Table 5.1 and videos are available at the website [1].

2) Visual Imitation In the previous paragraph, we saw that the robot can reach a goal that's within the same room. However, our agent is unable to reach far away goals such as in other rooms using just a single image. In such scenarios, an expert might communicate instructions like go to the door, turn right, go to the closest chair etc. Instead of language instruction, in our setup we provide a sequence of *landmark* images to convey the same high-level idea. These landmark images were captured from the robot's camera as the expert moved the robot from the start to a goal location. However, note that it is not necessary for the expert to control the robot to capture the images because we don't make use of the expert's actions, but only the images. Instead of providing the image after every action in the demonstration, we only provided every fifth image. The rationale behind this choice is that we want to sample the demonstration sparsely to minimize the agent's reliance on the expert. Such sub-sampling (as shown in Figure 5.4) provides an easy way to vary the complexity of the task.

We evaluate via multiple runs of two demonstrations, namely, maze demonstration where the robot is supposed to navigate through a maze-like path and perturbed loop demonstration, where the robot is supposed to make a complete loop as instructed by demonstration images. The loop demonstration is longer and more difficult than the maze. We start the agent from different starting locations and orientations with respect to that of demonstration. Each orientation is initialized such that no

[1] https://pathak22.github.io/zeroshot-imitation/

Figure 5.4: The performance of TurtleBot at following a visual demonstration given as a sequence of images (top row). The TurtleBot is positioned in a manner such that the first image in demonstration has no overlap with its current observation. Even under this condition the robot is able to move close to the first demo image (shown as Robot WayPoint-1) and then follow the provided demonstration until the end. This also exemplifies a failure case for classical methods; there are no possible keypoint matches between WayPoint-1 and WayPoint-2, and the initial observation is even farther from WayPoint-1.

part of the demonstration's initial frame is visible. Results are shown in Table 5.2. When we sample every frame, our method and classical structure from motion can both be used to follow the demonstration. However, at sub-sampling rate of five, SIFT-based feature matching approaches did not work and ORBSLAM2 [177] failed to generate a map, whereas our method was successful. Notice that providing sparse landmark images instead of dense video adds robustness to the visual imitation task. In particular, consider the scenario in which the environment has changed since the time the demonstration was recorded. By not requiring the agent to match every demonstration image frame-by-frame, it becomes less sensitive to changes in the environment.

5.2.2 3D Navigation in VizDoom

We have evaluated our approach on real-robot scenarios thus far. To further analyze the performance and robustness of our approach through large scale experiments, we setup the same navigation task as described in previous subsection in a simulated VizDoom environment. Our goal is to measure: (1) the robustness of each method with proper error bars, (2) the role of initial self-supervised data collection for performance on visual imitation, (3) the quantitative difference in modeling forward consistency loss in feature space in comparison to raw visual space.

In VizDoom, we collect data by deploying two types of exploration methods:

Model Name	Maze Demonstration			Loop Demonstration		
	Run-1	Run-2	Run-3	Run-1	Run-2	Run-3
SIFT	10%	5%	15%	—	—	—
GSP-NoPrevAction-NoFwdConst	60%	70%	100%	—	—	—
GSP-NoFwdConst	65%	90%	100%	0%	0%	0%
GSP (ours)	100%	60%	100%	0%	100%	100%

Table 5.2: Quantitative evaluation of TurtleBot's performance at following visual demonstrations in two scenarios: maze and the loop. We report the % of landmarks reached by the agent across three runs of two different demonstrations. Results show that our method outperforms the baselines. Note that 3 more trials of the loop demonstration were tested under significantly different lighting conditions and neither model succeeded. Detailed results are available in the supplementary materials.

random exploration and curiosity-driven exploration [176]. The hypothesis is that if the initial data collected by the robot is driven by a better strategy than just random, this should eventually help the agent follow long demonstrations better. Our environment consists of 2 maps in total. We train on one map with 5 different starting positions for collecting exploration data. For validation, we collect 5 human demonstrations in a map with the same layout as in training but with different textures. For zero-shot generalization, we collect 5 human demonstrations in a novel map layout with novel textures. Exact details for data collection and training setup are in the supplementary materials of the paper [87].

Metric

We report the median of maximum distance reached by the robot in following the given sequence of demonstration images. The maximum distance reached is the distance of farthest landmark point that the agent reaches contiguously, i.e., without missing any intermediate landmarks. Measuring the farthest landmark reached does not capture how efficiently it is reached. Hence, we further measure efficiency of the agent as the ratio of number of steps taken by the agent to reach farthest contiguous landmark with respect to the number of steps shown in human demonstrations.

Visual Imitation

The task here is same as the one in real robot navigation where the agent is shown a sparse sequence of images to imitate. The results are in Table 5.3. We

Model Name	Same Map, Same Texture		Same Map, Diff Texture		Diff Map, Diff Texture	
	Median %	Efficiency %	Median %	Efficiency %	Median %	Efficiency %
Random Exploration for Data Collection:						
GSP-NoFwdConst	63.2 ± 5.7	36.4 ± 3.3	32.2 ± 0.7	28.9 ± 4.0	34.5 ± 0.6	23.1 ± 2.4
GSP (ours pixels)	62.2 ± 5.1	43.0 ± 2.6	32.4 ± 0.8	30.9 ± 2.9	35.4 ± 1.1	29.3 ± 3.9
GSP (ours features)	68.9 ± 6.9	53.9 ± 4.0	32.4 ± 0.7	47.4 ± 7.6	39.1 ± 2.0	30.4 ± 2.5
Curiosity-driven Exploration for Data Collection:						
GSP-NoFwdConst	78.2 ± 2.3	63.0 ± 4.3	43.2 ± 2.6	33.9 ± 3.0	40.2 ± 4.0	27.3 ± 1.9
GSP-FwdRegularizer	78.4 ± 3.4	59.8 ± 4.1	50.6 ± 4.7	30.9 ± 3.0	37.9 ± 1.1	28.9 ± 1.7
GSP (ours pixels)	78.2 ± 3.4	65.2 ± 4.2	47.1 ± 4.7	32.4 ± 3.0	44.8 ± 4.0	29.5 ± 1.9
GSP (ours features)	78.2 ± 4.6	67.0 ± 3.3	49.4 ± 4.8	26.9 ± 1.5	47.1 ± 3.0	24.1 ± 1.7

Table 5.3: Quantitative evaluation of our proposed GSP and the baseline models at following visual demonstrations in VizDoom 3D Navigation. Medians and 95% confidence intervals are reported for demonstration completion and efficiency over 50 seeds and 5 human paths per environment type.

found that the exploration data collected via curiosity significantly improves the final imitation performance across all methods including the baselines with respect to random exploration. Our baseline GSP model with a forward regularizer instead of consistency loss ends up overfitting to the training layout. In contrast, our forward-consistent GSP model outperforms other methods in generalizing to new map with novel textures. This indicates that the forward consistency is possibly doing more than just regularizing the policy features. Training forward consistency loss in feature space further enhances the generalization even when both pixel and feature space models perform similarly on training environment.

5.3 Conclusions

The results demonstrate that the *forward-consistency* loss helps in learning better dynamics models that significantly improves performance on the tasks of knot-tying and visual navigation. Moreover, results in VizDoom (section 5.2.2) show that using data collected by curiosity-driven exploration leads to learning of better models as compared to data collected via random exploration. These results taken together suggest that better exploration strategies for collecting data and improvements in model can improve the accuracy of imitation.

In the experiments presented above we have shown how an agent can imitate by observing a single visual demonstration provided by the expert. An interesting

challenge for future research is how can an agent learn to perform the $(N+1)^{th}$ task without any demonstration after having received demonstration for N tasks.

Chapter 6

Exploration

As human agents, we are accustomed to operating with rewards that are so sparse that we only experience them once or twice in a lifetime, if at all. To a three-year-old enjoying a sunny Sunday afternoon on a playground, most trappings of modern life – college, good job, a house, a family – are so far into the future, they provide no useful reinforcement signal. Yet, the three-year-old has no trouble entertaining herself in that playground using what psychologists call intrinsic motivation [180] or curiosity [181]. Motivation/curiosity have been used to explain the need to explore the environment and discover novel states. The French word *flâneur* perfectly captures the notion of a curiosity-driven observer, the "deliberately aimless pedestrian, unencumbered by any obligation or sense of urgency" (Cornelia Otis Skinner). More generally, curiosity is a way of learning new skills which might come handy for pursuing rewards in the future.

Similarly, in reinforcement learning, intrinsic motivation/rewards become critical whenever extrinsic rewards are sparse. Most formulations of intrinsic reward can be grouped into two broad classes: 1) encourage the agent to explore "novel" states [182–184] or, 2) encourage the agent to perform actions that reduce the error/uncertainty in the agent's ability to predict the consequence of its own actions (i.e. its knowledge about the environment) [185–190].

Measuring "novelty" requires a statistical model of the distribution of the environmental states, whereas measuring prediction error/uncertainty requires building a model of environmental dynamics that predicts the next state (s_{t+1}) given the current state (s_t) and the action (a_t) executed at time t. Both these models are hard to build in high-dimensional continuous state spaces such as images. An additional challenge lies in dealing with the stochasticity of the agent-environment system, both due to the noise in the agent's actuation, which causes its end-effectors to move in a stochastic manner, and, more fundamentally, due to the inherent stochasticity in

the environment. To give the example from [186], if the agent receiving images as state inputs is observing a television screen displaying white noise, every state will be novel as it would be impossible to predict the value of any pixel in the future. This means that the agent will remain curious about the television screen because it is unaware that some parts of the state space simply cannot be modeled and thus the agent can fall into an artificial curiosity trap and stall its exploration. Other examples of such stochasticity include appearance changes due to shadows from other moving entities, presence of distractor objects, or other agents in the environment whose motion is not only hard to predict but is also irrelevant to the agent's goals. Somewhat different, but related, is the challenge of generalization across physically (and perhaps also visually) distinct but functionally similar parts of an environment, which is crucial for large-scale problems. One proposed solution to all these problems is to only reward the agent when it encounters states that are hard to predict but are "learnable" [185]. However, estimating learnability is a non-trivial problem [184].

This work belongs to the broad category of methods that generate an intrinsic reward signal based on how hard it is for the agent to predict the consequences of its own actions However, we manage to escape most pitfalls of previous prediction approaches with the following key insight: we only predict those changes in the environment that could possibly be due to the actions of our agent or affect the agent, and ignore the rest. That is, instead of making predictions in the raw sensory space (e.g. pixels), we transform the sensory input into a feature space where only the information relevant to the action performed by the agent is represented. We learn this feature space using self-supervision – training a neural network on a proxy inverse dynamics task of predicting the agent's action given its current and next states. Since the neural network is only required to predict the action, it has no incentive to represent within its feature embedding space the factors of variation in the environment that do not affect the agent itself. We then use this feature space to train a forward dynamics model that predicts the feature representation of the next state, given the feature representation of the current state and the action. We provide the prediction error of the forward dynamics model to the agent as an intrinsic reward to encourage its curiosity.

The role of curiosity has been widely studied in the context of solving tasks with sparse rewards. In our opinion, curiosity has two other fundamental uses. Curiosity helps an agent explore its environment in the quest for new knowledge (a desirable characteristic of exploratory behavior is that it should improve as the agent gains more knowledge). Further, curiosity is a mechanism for an agent to learn skills that might be helpful in future scenarios. In this paper, we evaluate the effectiveness of our curiosity formulation in all three of these roles.

We first compare the performance of an A3C agent [191] with and without

the curiosity signal on 3-D navigation tasks with sparse extrinsic reward in the *VizDoom* environment. We show that a curiosity-driven intrinsic reward is crucial in accomplishing these tasks (see Section 6.3.1). Next, we show that even in the absence of any extrinsic rewards, a curious agent learns good exploration policies. For instance, an agent trained only with curiosity as its reward is able to cross a significant portion of Level-1 in *Super Mario Bros.* Similarly in *VizDoom*, the agent learns to walk intelligently along the corridors instead of bumping into walls or getting stuck in corners (see Section 6.3.2). A question that naturally follows is whether the learned exploratory behavior is specific to the physical space that the agent trained itself on, or if it enables the agent to perform better in unseen scenarios too? We show that the exploration policy learned in the first level of *Mario* helps the agent explore subsequent levels faster, while the intelligent walking behavior learned by the curious *VizDoom* agent transfers to a completely new map with new textures (see Section 6.3.3). These results suggest that the proposed method enables an agent to learn generalizable skills even in the absence of an explicit goal.

6.1 Curiosity-Driven Exploration

Our agent is composed of two subsystems: a reward generator that outputs a curiosity-driven intrinsic reward signal and a policy that outputs a sequence of actions to maximize that reward signal. In addition to intrinsic rewards, the agent optionally may also receive some extrinsic reward from the environment. Let the intrinsic curiosity reward generated by the agent at time t be r_t^i and the extrinsic reward be r_t^e. The policy sub-system is trained to maximize the sum of these two rewards $r_t = r_t^i + r_t^e$, with r_t^e mostly (if not always) zero.

We represent the policy $\pi(s_t; \theta_P)$ by a deep neural network with parameters θ_P. Given the agent in state s_t, it executes the action $a_t \sim \pi(s_t; \theta_P)$ sampled from the policy. θ_P is optimized to maximize the expected sum of rewards,

$$\max_{\theta_P} \mathbb{E}_{\pi(s_t;\theta_P)}[\Sigma_t r_t] \tag{6.1}$$

Unless specified otherwise, we use the notation $\pi(s)$ to denote the parameterized policy $\pi(s; \theta_P)$. Our curiosity reward model can potentially be used with a range of policy learning methods; in the experiments discussed here, we use the asynchronous advantage actor critic policy gradient (A3C) [191] for policy learning. Our main contribution is in designing an intrinsic reward signal based on prediction error of the agent's knowledge about its environment that scales to high-dimensional continuous state spaces like images, bypasses the hard problem of predicting pixels

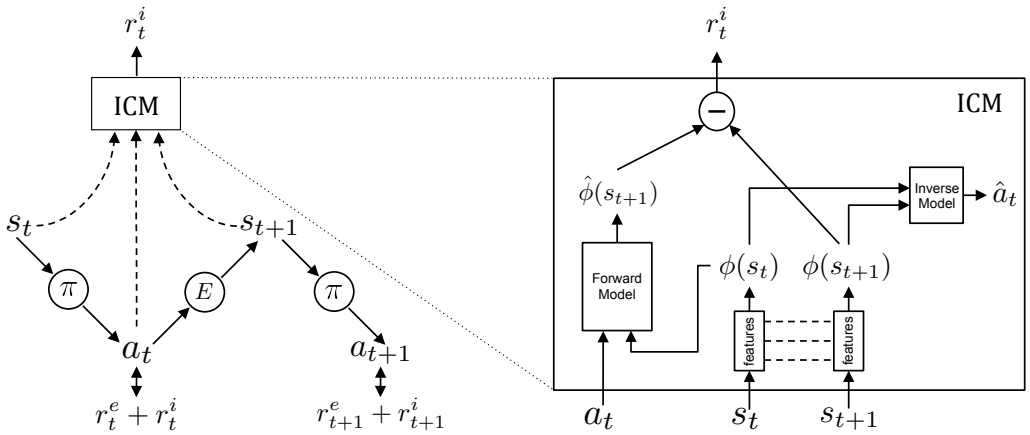


Figure 6.1: The agent in state s_t interacts with the environment by executing an action a_t sampled from its current policy π and ends up in the state s_{t+1}. The policy π is trained to optimize the sum of the extrinsic reward (r_t^e) provided by the environment E and the curiosity based intrinsic reward signal (r_t^i) generated by our proposed Intrinsic Curiosity Module (ICM). ICM encodes the states s_t, s_{t+1} into the features $\phi(s_t), \phi(s_{t+1})$ that are trained to predict a_t (i.e. inverse dynamics model). The forward model takes as inputs $\phi(s_t)$ and a_t and predicts the feature representation $\hat{\phi}(s_{t+1})$ of s_{t+1}. The prediction error in the feature space is used as the curiosity based intrinsic reward signal. As there is no incentive for $\phi(s_t)$ to encode any environmental features that can not influence or are not influenced by the agent's actions, the learned exploration strategy of our agent is robust to uncontrollable aspects of the environment.

and is unaffected by the unpredictable aspects of the environment that do not affect the agent.

6.1.1 Prediction error as curiosity reward

Making predictions in the raw sensory space (e.g. when s_t corresponds to images) is undesirable not only because it is hard to predict pixels directly, but also because some part of the input sensory space could be unpredictable and inconsequential to the agent, for e.g., the movement and location of tree leaves in a breeze in the environment.

For determining a good feature space for making future predictions, let's divide all sources that can influence the agent's observations into three cases: (1) things that can be controlled by the agent; (2) things that the agent cannot control but that can affect the agent (e.g. a vehicle driven by another agent), and (3) things

out of the agent's control and not affecting the agent (e.g. moving leaves). A good feature space for curiosity should model (1) and (2) and be unaffected by (3). This latter is because, if there is a source of variation that is inconsequential for the agent, then the agent has no incentive to know about it.

6.1.2 Self-supervised prediction for exploration

Instead of hand-designing features for every environment, our aim is to come up with a general mechanism for learning feature representations such that the prediction error in the learned feature space provides a good intrinsic reward signal. Given the raw state s_t, we encode it using a deep neural network into a feature vector $\phi(s_t; \theta_E)$, denoted as $\phi(s_t)$ for succinctness. We propose to learn the parameters of this feature encoder using two sub-modules described as follows.

The first sub-module is the neural network g which takes the feature encoding $\phi(s_t), \phi(s_{t+1})$ of two consequent states as input and predicts the action a_t taken by the agent to move from state s_t to s_{t+1}, defined as:

$$\hat{a}_t = g\Big(\phi(s_t), \phi(s_{t+1}); \theta_I\Big) \tag{6.2}$$

where, $\hat{a}_t$ is the predicted estimate of the action a_t. The neural network parameters θ_I, θ_E are trained to optimize,

$$\min_{\theta_I, \theta_E} L_I(\hat{a}_t, a_t) \tag{6.3}$$

where, L_I is the loss function that measures the discrepancy between the predicted and actual actions. L_I is modeled as soft-max loss across all possible actions when a_t is discrete, and euclidean regression loss for continuous actions a_t. This learned function g is also known as the inverse dynamics model and the tuple (s_t, a_t, s_{t+1}) required to learn g is obtained while the agent interacts with the environment using its current policy $\pi(s)$.

In addition to the inverse dynamics model, we train another sub-module that takes as inputs a_t and $\phi(s_t)$ to predict the feature encoding of the state at time step $t + 1$,

$$\hat{\phi}(s_{t+1}) = f\Big(\phi(s_t), a_t; \theta_F\Big) \tag{6.4}$$

where $\hat{\phi}(s_{t+1})$ is the predicted estimate of $\phi(s_{t+1})$. The function f is also known as the forward dynamics model and is trained to optimize the regression loss,

$$\min_{\theta_F, \theta_E} L_F\Big(\hat{\phi}(s_{t+1}), \phi(s_t)\Big) \tag{6.5}$$

Finally, the intrinsic reward signal r_t^i is computed as,

$$r_t^i = \frac{\eta}{2}\|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2 \tag{6.6}$$

where $\eta > 0$ is a scaling factor. In order to generate the curiosity-based intrinsic reward, we jointly optimize the inverse and forward dynamics loss described in equations (6.3) and (6.5) respectively. We refer to this proposed curiosity formulation as Intrinsic Curiosity Module (ICM). The inverse model helps learn a feature space that encodes information relevant for predicting the agent's actions only and the forward model error makes this learned feature representation more predictable as long as it is rich enough to capture inverse dynamics. As there is no incentive for this feature space to encode any environmental features that are not influenced by the agent's actions, our agent will receive no rewards for reaching environmental states that are inherently unpredictable and its exploration strategy will be robust to the presence of distractor objects, changes in illumination, or other nuisance sources of variation in the environment. See Figure 6.1 for illustration of the formulation.

The overall optimization problem for learning the agent is a composition of equations (6.1), (6.3) and (6.5), defined as

$$\min_{\theta_P,\theta_I,\theta_F,\theta_E} \Bigg[-\lambda \mathbb{E}_{\pi(s_t;\theta_P)}[\Sigma_t r_t] + (1-\beta)L_I + \beta L_F \Bigg] \tag{6.7}$$

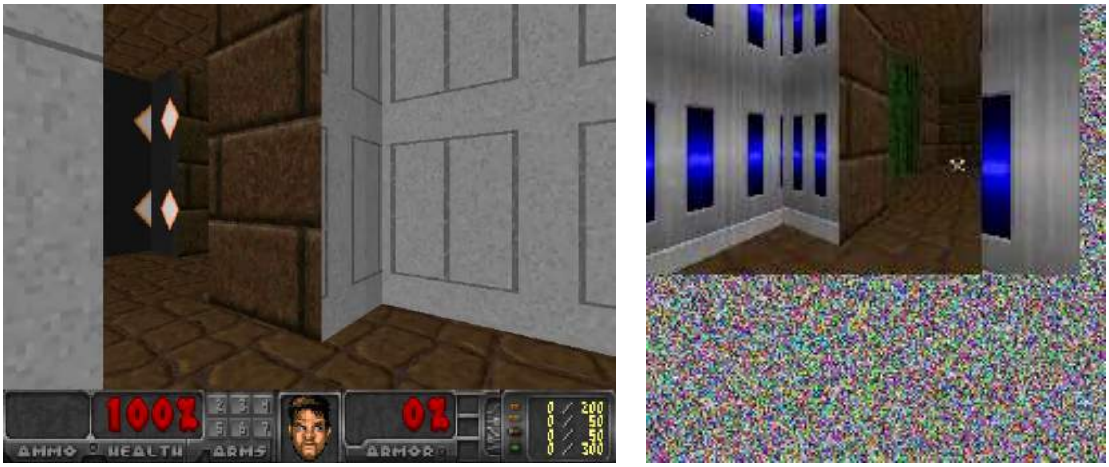
where $0 \leq \beta \leq 1$ is a scalar that weighs the inverse model loss against the forward model loss and $\lambda > 0$ is a scalar that weighs the importance of the policy gradient loss against the importance of learning the intrinsic reward signal.

The use of inverse models has been investigated to learn features for recognition tasks [48, 49, 85]. Agrawal et al. [85] also used the forward model as a regularizer for the features with no policy learning, while we use it to generate the curiosity reward for training our agent's policy.

6.2 Experimental Setup

To evaluate our curiosity module on its ability to improve exploration and provide generalization to novel scenarios, we use two simulated environments.

Environments The first environment is the VizDoom [192] game where we consider the 3-D navigation task with four discrete actions – forward, left, right and no-action. Our testing setup in all the experiments is the 'DoomMyWayHome-v0' environment

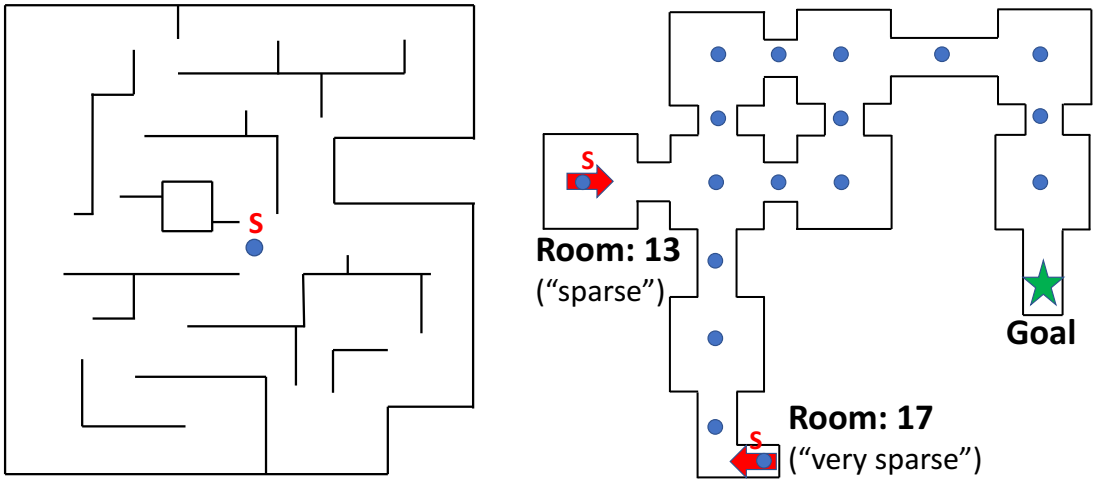


(a) Input snapshot in VizDoom (b) Input w/ noise

Figure 6.2: Frames from VizDoom 3-D environment which agent takes as input: (a) Usual 3-D navigation setup; (b) Setup when uncontrollable noise is added to the input.

which is available as part of OpenAI Gym [193]. The map consists of 9 rooms connected by corridors and the agent is tasked to reach some fixed goal location from its spawning location. Episodes are terminated either when the agent reaches fixed goal or if the agent exceeds a maximum of 2100 time steps. The agent is only provided a sparse terminal reward of +1 if it finds the vest and zero otherwise. For generalization experiments, we pre-train on a different map with different random textures from [194] with 2100 steps long episodes as there is no goal in pre-training. Sample frames from VizDoom are shown in Figure 6.2a, and maps are explained in Figure 6.3. It takes approximately 350 steps for an optimal policy to reach the vest location from the farthest room in this map (sparse reward).

Our second environment is the classic Nintendo game Super Mario Bros [195]. We consider four levels of the game: pre-training on the first level and showing generalization on the subsequent levels. In this setup, we reparametrize the action space of the agent into 14 unique actions following [195]. This game is played using a joystick allowing for multiple simultaneous button presses, where the duration of the press affects what action is being taken. This property makes the game particularly hard, e.g. to make a long jump over tall pipes or wide gaps, the agent needs to predict the same action up to 12 times in a row, introducing long-range dependencies. All our experiments on Mario are trained using curiosity signal only, without any reward from the game.



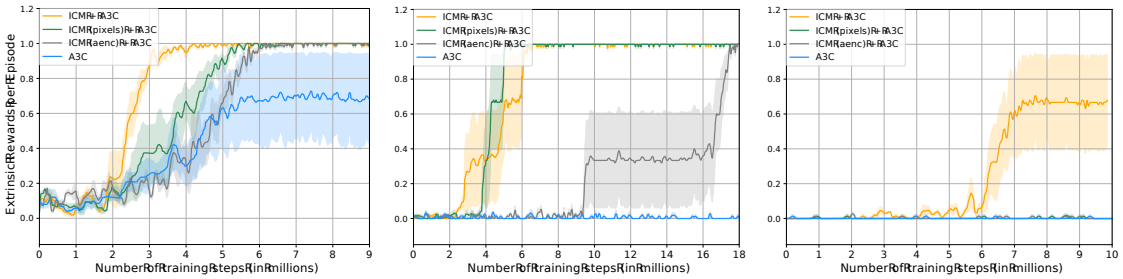
(a) Train Map Scenario (b) Test Map Scenario

Figure 6.3: Maps for VizDoom 3-D environment: (a) For generalization experiments (c.f. Section 6.3.3), map of the environment where agent is pre-trained only using curiosity signal without any reward from environment. 'S' denotes the starting position. (b) Testing map for VizDoom experiments. Green star denotes goal location. Blue dots refer to 17 agent spawning locations in the map in the "dense" case. Rooms 13, 17 are the fixed start locations of agent in "sparse" and "very sparse" reward cases respectively. Note that textures are also different in train and test maps.

Baseline Methods We compare our model (denoted as 'ICM + A3C') against (a) vanilla 'A3C' with ϵ-greedy exploration; (b) 'ICM-pixels + A3C' where we predict the next observation in the pixel space instead of the feature space of the inverse model (see supp. materials for implementation details). The performance comparison between 'ICM-pixels + A3C' and 'ICM + A3C' is indicative of pros/cons of our method over established methods of computing curiosity reward by making predictions in observation space [186, 189]; (c) comparison with state-of-the-art exploration methods based on variational information maximization (VIME) [190] criterion.

6.3 Experiments

Three broad settings are evaluated: a) sparse extrinsic reward on reaching a goal (Section 6.3.1); b) exploration with no extrinsic reward (Section 6.3.2); and c) generalization to novel scenarios (Section 6.3.3). Generalization is evaluated on a



(a) "dense reward" (b) "sparse reward" (c) "very sparse reward"

Figure 6.4: Comparing the performance of the A3C agent with no curiosity (blue) against the curiosity in pixel space agent (green) and the proposed curious ICM-A3C agent (orange) as the hardness of the exploration task is gradually increased from left to right. Exploration becomes harder with larger distance between the initial and goal locations: "dense", "sparse" and "very sparse". The results depict that succeeding on harder exploration task becomes progressively harder for the baseline A3C, whereas the curious A3C is able to achieve good score in all the scenarios. Pixel based curiosity works in dense and sparse but fails in very sparse reward setting. The protocol followed in the plots involves running three independent runs of each algorithm. Darker line represents mean and shaded area represents mean ± standard error of mean. We did not perform any tuning of random seeds.

novel map with novel textures in *VizDoom* and on subsequent game levels in *Mario*.

6.3.1 Sparse Extrinsic Reward Setting

In the 'DoomMyWayHome-v0' task (section 6.2) agent is provided with a sparse extrinsic reward only when it finds the goal located at a fixed location in the map. The episode terminates upon reaching goal or after a maximum of 2100 steps We systematically varied the difficulty of this task and constructed "dense", "sparse" and "very-sparse" reward (see Figure 6.3b) scenarios by varying the distance between the initial spawning location of the agent and the location of the goal. In the "dense" reward case, the agent is randomly spawned in any of the 17 possible spawning locations uniformly distributed across the map. This is not a hard exploration task because sometimes the agent is randomly initialized close to the goal and therefore by random ϵ-greedy exploration it can reach the goal with reasonably high probability. In the "sparse" and "very sparse" reward cases, the agent is always spawned in Room-13 and Room-17 respectively which are 270 and 350 steps away from the

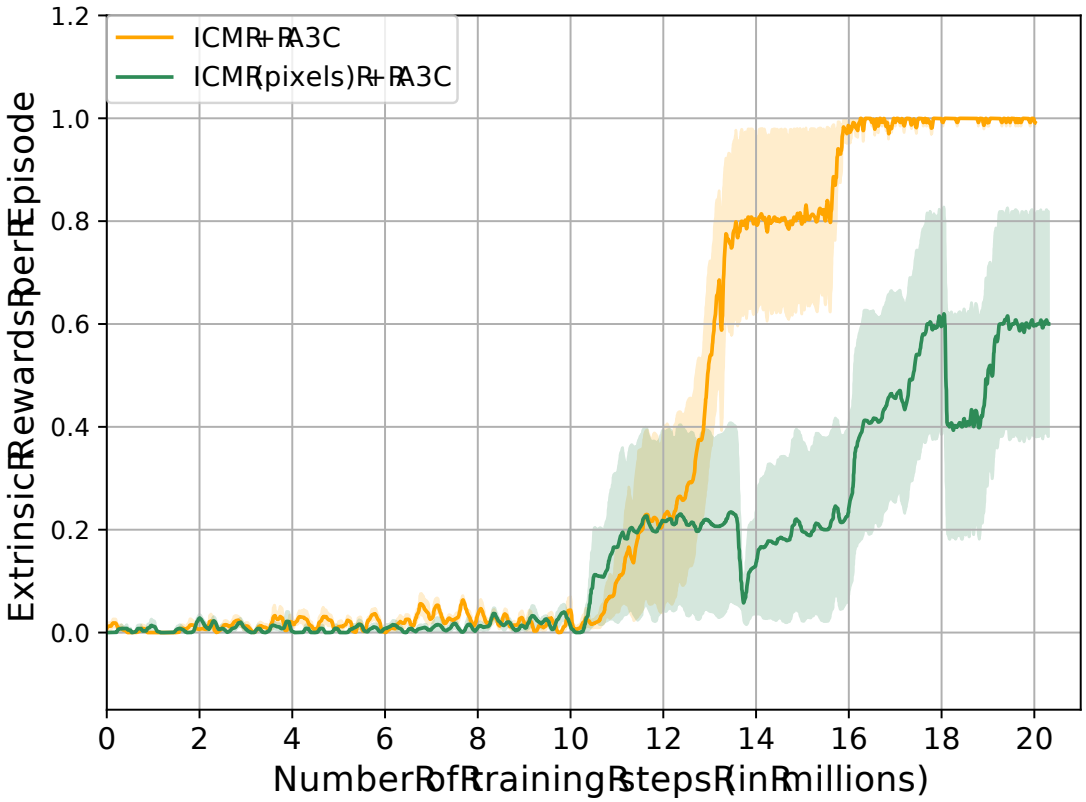


Figure 6.5: Evaluating the robustness of ICM to the presence of uncontrollable distractors in the environment. We created such a distractor by replacing 40% of the visual observation of the agent by white noise (see Figure 6.2b). The results show that while ICM succeeds most of the times, the pixel prediction model struggles.

goal under an optimal policy. A long sequence of directed actions is required to reach the goals from these rooms, making these settings hard goal directed exploration problems.

Results in Figure 6.4 show that curious agents learn much faster indicating that their exploration is more effective in compared to ϵ-greedy exploration of the baseline agent. One possible explanation of the inferior performance of ICM-pixels in comparison to ICM is that in every episode the agent is spawned in one out of seventeen rooms with different textures. It is hard to learn a pixel-prediction model as the number of textures increases.

In the "sparse" reward case, as expected, the baseline A3C agent fails to solve the

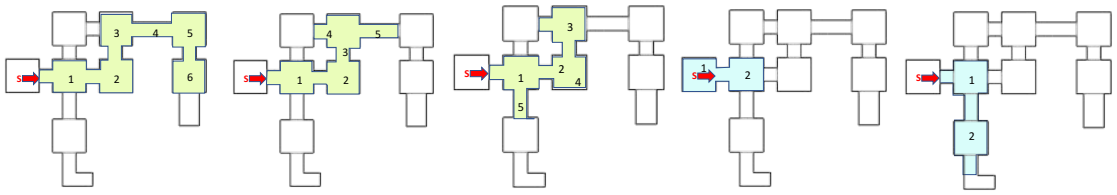


Figure 6.6: Each column in the figure shows the coverage of an agent by coloring the rooms it visits during 2100 steps of exploration. The red arrow shows the initial location and orientation of the agent at the start of the episode The first three (in green) and the last two columns (in blue) show visitation of curious (ICM) and randomly exploring agents respectively. The results clearly show that the curious agent trained with intrinsic rewards explores a significantly larger number of rooms as compared to a randomly exploring agent.

task, while the curious A3C agent is able to learn the task quickly. Note that ICM-pixels and ICM have similar convergence because, with a fixed spawning location of the agent, the ICM-pixels encounters the same textures at the starting of each episode which makes learning the pixel-prediction model easier as compared to the "dense" reward case. Finally, in the "very sparse" reward case, both the A3C agent and ICM-pixels never succeed, while the ICM agent achieves a perfect score in 66% of the random runs. This indicates that ICM is better suited than ICM-pixels and vanilla A3C for hard goal directed exploration tasks.

Robustness to uncontrollable dynamics For testing the robustness of the proposed ICM formulation to changes in the environment that do not affect the agent, we augmented the agent's observation with a fixed region of white noise which made up 40% of the image (see Figure 6.2b). In *VizDoom* 3-D navigation, ideally the agent should be unaffected by this noise as the noise does not affect the agent in anyway and is merely a nuisance. Figure 6.5 compares the performance of ICM against some baseline methods on the "sparse" reward setup described above. While, the proposed ICM agent achieves a perfect score, ICM-pixels suffers significantly despite having succeeded at the "sparse reward" task when the inputs were not augmented with any noise (see Figure 6.4b). This indicates that in contrast to ICM-pixels, ICM is insensitive to nuisance changes in the environment.

Comparison to other baselines One possible reason for superior performance of the curious agent is that the intrinsic reward signal is simply acting as a regularizer by providing random rewards that push the agent out of the local minima. We systematically tested this hypothesis using many different random reward distributions

on the “sparse VizDoom" task and found that with just random rewards the agents fail on sparse reward tasks. Please see supplementary materials for more details. Comparison to the state of the art TRPO-VIME [190] agent in the table below shows that the ICM agent is superior in performance. The hyper-parameters and accuracy for TRPO and VIME agents follow from the concurrent work [196].

Method (“sparse” reward setup)	Mean (Median) Score (at convergence)
TRPO	26.0 % (0.0 %)
A3C	0.0 % (0.0 %)
VIME + TRPO	46.1 % (27.1 %)
ICM + A3C	**100.0** % (100.0 %)

6.3.2 No Reward Setting

A good exploration policy is one which allows the agent to visit as many states as possible even without any goals. In order to test if our agent can learn a good exploration policy, we trained it on *VizDoom* and *Mario* without any rewards from the environment. We then evaluated what portion of the map was explore (for *VizDoom*), and how much progress it made (for *Mario*) in this setting. To our surprise, we have found that in both cases, the no-reward agent was able to perform quote well (see video at http://pathak22.github.io/noreward_rl/).

VizDoom: Coverage during Exploration. An agent trained with no extrinsic rewards was able to learn to navigate corridors, walk between rooms and explore many rooms in the 3-D Doom environment. On many occasions, the agent traversed the entire map and reached rooms that were farthest away from the room it was initialized in. Given that the episode terminates in 2100 steps and farthest rooms are over 250 steps away (for an optimally-moving agent), this result is quite remarkable, demonstrating that it is possible to learn useful skills without the requirement of any external supervision of rewards. Example explorations are shown in Figure 6.6. The first 3 maps show our agent explore a much larger state space without any extrinsic signal, compared to a random exploration agent (last two maps), which often has hard time getting around local minima of state spaces, e.g. getting stuck against a wall and not able to move (see video).

Mario: Learning to play with no rewards. Without any extrinsic reward from environment, our Mario agent can learn to cross over 30% of Level-1. The agent received no reward for killing or dodging enemies or avoiding fatal events, yet it automatically discovered these behaviors (see video). One possible reason is

Level Ids	Level-1	Level-2				Level-3			
Accuracy Iterations	Scratch 1.5M	Run as is 0	Fine-tuned 1.5M	Scratch 1.5M	Scratch 3.5M	Run as is 0	Fine-tuned 1.5M	Scratch 1.5M	Scratch 5.0M
Mean ± stderr	711 ± 59.3	31.9 ± 4.2	466 ± 37.9	399.7 ± 22.5	455.5 ± 33.4	319.3 ± 9.7	97.5 ± 17.4	11.8 ± 3.3	42.2 ± 6.4
% distance > 200	50.0 ± 0.0	0	64.2 ± 5.6	88.2 ± 3.3	69.6 ± 5.7	50.0 ± 0.0	1.5 ± 1.4	0	0
% distance > 400	35.0 ± 4.1	0	63.6 ± 6.6	33.2 ± 7.1	51.9 ± 5.7	8.4 ± 2.8	0	0	0
% distance > 600	35.8 ± 4.5	0	42.6 ± 6.1	14.9 ± 4.4	28.1 ± 5.4	0	0	0	0

Table 6.1: Quantitative evaluation of the agent trained to play Super Mario Bros. using only curiosity signal without any rewards from the game. The policy learnt on Level-1 is evaluated both when it is is run "as is", and further fine-tuned on subsequent levels. The results are compared to settings when Mario agent is trained from scratch in Level-2,3 using only curiosity without any extrinsic rewards. Evaluation metric is based on the distance covered by the Mario agent.

because getting killed by the enemy will result in only seeing a small part of the game space, making its curiosity saturate. In order to remain curious, it is in the agent's interest to learn how to kill and dodge enemies so that it can reach new parts of the game space. This suggests that curiosity provides indirect supervision for learning interesting behaviors.

To the best of our knowledge, this is the first demonstration where the agent learns to navigate in a 3D environment and discovers how to play a game by making use of relatively complex visual imagery directly from pixels, without any extrinsic rewards. There are several prior works that use reinforcement learning to navigate in 3D environments from pixel inputs or playing ATARI games such as [3, 191, 197], but they rely on intermediate external rewards provided by the environment.

6.3.3 Generalization to Novel Scenarios

In the previous section we showed that our agent learns to explore large parts of the space where its curiosity-driven exploration policy was trained. However it remains unclear, when exploring a space, how much of the learned behavior is specific to that particular space and how much is general enough to be useful in novel scenarios? To investigate this question, we train a no reward exploratory behavior in one scenario (e.g. Level-1 of Mario) and then evaluate the resulting exploration policy in three different ways: a) apply the learned policy "as is" to a new scenario; b) adapt the policy by fine-tuning with curiosity reward only; c) adapt the policy to maximize some extrinsic reward. Happily, in all three cases, we observe some promising generalization results:

Evaluate "as is": The distance covered by the agent on Levels 1, 2, and 3 when the policy learned by maximizing curiosity on Level-1 of *Mario* is executed without any adaptation is reported in Table 6.1. The agent performs surprisingly well on Level 3, suggesting good generalization, despite the fact that Level-3 has different structures and enemies compared to Level-1. However, note that running "as is" on Level-2 does not do well. At first, this seems to contradict the generalization results on Level-3. However, note that Level-3 has similar global visual appearance (day world with sunlight) to Level-1, whereas Level-2 is significantly different (night world). If this is indeed the issue, then it should be possible to quickly adapt the agent's exploration policy to Level-2 with a little bit of "fine-tuning". We will explore this below.

Fine-tuning with curiosity only: From Table 6.1 we see that when the agent pre-trained (using only curiosity as reward) on Level-1 is fine-tuned (using only curiosity as reward) on Level-2 it quickly overcomes the mismatch in global visual appearance and achieves a higher score than training from scratch with the same number of iterations. Interestingly, training "from scratch" on Level-2 is worse than the fine-tuned policy, even when training for more iterations than pre-training + fine-tuning combined. One possible reason is that Level-2 is more difficult than Level-1, so learning the basic skills such as moving, jumping, and killing enemies from scratch is harder than in the relative "safety" of Level-1. This result, therefore might suggest that first pre-training on an earlier level and then fine-tuning on a later one produces a form of curriculum which aids learning and generalization. In other words, the agent is able to use the knowledge it acquired by playing Level-1 to better explore the subsequent levels. Of course, the game designers do this on purpose to allow the human players to gradually learn to play the game.

However, interestingly, fine-tuning the exploration policy pre-trained on Level-1 to Level-3 deteriorates the performance, compared to running "as is". This is because Level-3 is very hard for the agent to cross beyond a certain point – the agent hits a curiosity blockade and is unable to make any progress. As the agent has already learned about parts of the environment before the hard point, it receives almost no curiosity reward and as a result it attempts to update its policy with almost zero intrinsic rewards and the policy slowly degenerates. This behavior is vaguely analogous to boredom, where if the agent is unable to make progress it gets bored and stops exploring.

Fine-tuning with extrinsic rewards: If it is the case that the agent has actually learned useful exploratory behavior, then it should be able to learn quicker than starting from scratch even when external rewards are provided by environment. We

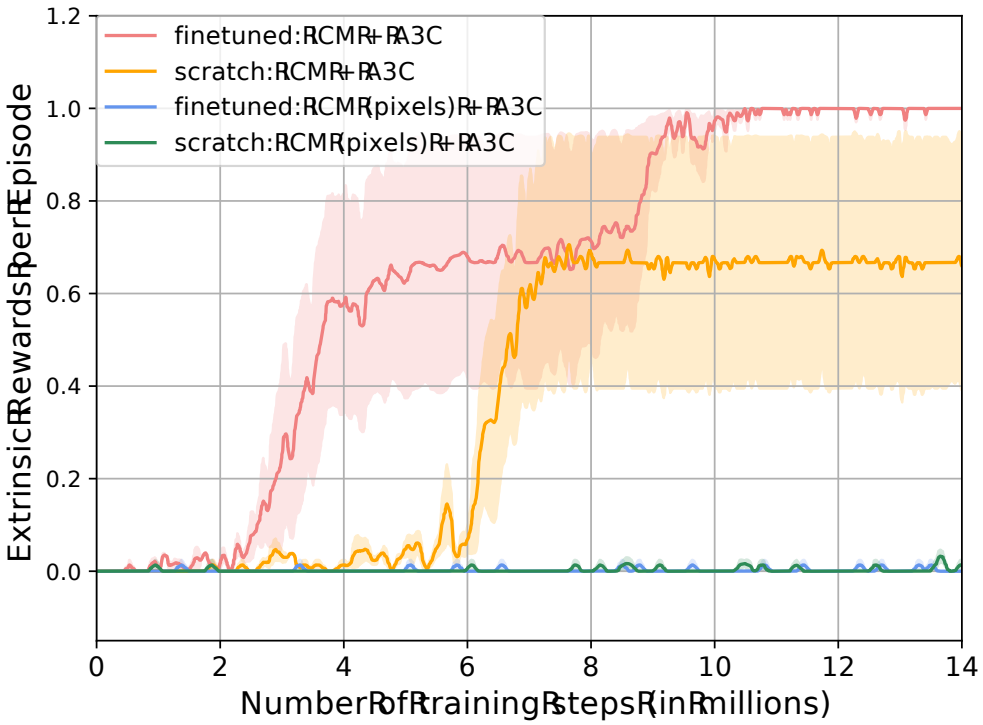


Figure 6.7: Curiosity pre-trained ICM + A3C agent when finetuned on the test map with environmental rewards outperforms ICM + A3C agent trained from scratch using both environmental and curiosity reward on the "very sparse" reward setting of *VizDoom*. The pixel prediction based ICM agent completely fails. These results indicate that our curiosity formulation is able to learn generalizable exploration policies.

perform this evaluation on *VizDoom* where we pre-train the agent using curiosity reward on a map showed in Figure 6.3a. We then test on the "very sparse" reward setting of 'DoomMyWayHome-v0' environment which uses a different map with novel textures (see Figure 6.3b) as described earlier in Section 6.3.1.

Results in Figure 6.7 show that the ICM agent pre-trained only with curiosity and then fine-tuned with external reward learns faster and achieves higher reward than an ICM agent trained from scratch to jointly maximize curiosity and the external rewards. This result confirms that the learned exploratory behavior is also useful when the agent is required to achieve goals specified by the environment. It is also worth noting that ICM-pixels does not generalize to this test environment. This indicates that the proposed mechanism of measuring curiosity is significantly better for learning skills that generalize as compared to measuring curiosity in the raw

sensory space.

6.4 Related Work

Curiosity-driven exploration is a well studied topic in the reinforcement learning literature and a good summary can be found in [198, 199]. Schmidhuber [185, 186] and Sun et al. [200] use surprise and compression progress as intrinsic rewards. Classic work of Kearns et al. [201] and Brafman et al. [202] propose exploration algorithms polynomial in the number of state space parameters. Others have used empowerment, which is the information gain based on entropy of actions, as intrinsic rewards [188, 203]. Stadie et al. [189] use prediction error in the feature space of an auto-encoder as a measure of interesting states to explore. State visitation counts have also been investigated for exploration [182, 204, 205]. Osband et al. [206] train multiple value functions and makes use of bootstrapping and Thompson sampling for exploration. Many approaches measure information gain for exploration [207–209]. Houthooft et al. [190] use an exploration strategy that maximizes information gain about the agent's belief of the environment's dynamics. Our approach of jointly training forward and inverse models for learning a feature space has similarities to [85, 125, 126], but these works use the learned models of dynamics for planning a sequence of actions instead of exploration. The idea of using a proxy task to learn a semantic feature embedding has been used in a number of works on self-supervised learning in computer vision [42, 46, 48–50, 54].

Concurrent work: A number of interesting related papers have appeared on Arxiv while the present work was in submission. Sukhbaatar et al. [210] generates supervision for pre-training via asymmetric self-play between two agents to improve data efficiency during fine-tuning. Several methods propose improving data efficiency of RL algorithms using self-supervised prediction based auxiliary tasks [211, 212]. Fu et al. [196] learn discriminative models, and Gregor et al. [213] use empowerment based measure to tackle exploration in sparse reward setups.

6.5 Discussion

In this work we propose a mechanism for generating curiosity-driven intrinsic reward signal that scales to high dimensional visual inputs, bypasses the difficult problem of predicting pixels and ensures that the exploration strategy of the agent is unaffected by nuisance factors in the environment. We demonstrate that our agent significantly outperforms the baseline A3C with no curiosity, a recently proposed VIME [190] formulation for exploration, and a baseline pixel-predicting formulation.

In *VizDoom* our agent learns the exploration behavior of moving along corridors and across rooms without any rewards from the environment. In *Mario* our agent crosses more than 30% of Level-1 without any rewards from the game. One reason why our agent is unable to go beyond this limit is the presence of a pit at 38% of the game that requires a very specific sequence of 15-20 key presses in order to jump across it. If the agent is unable to execute this sequence, it falls in the pit and dies, receiving no further rewards from the environment. Therefore it receives no gradient information indicating that there is a world beyond the pit that could potentially be explored. This issue is somewhat orthogonal to developing models of curiosity, but presents a challenging problem for policy learning.

It is common practice to evaluate reinforcement learning approaches in the same environment that was used for training. However, we feel that it is also important to evaluate on a separate "testing set" as well. This allows us to gauge how much of what has been learned is specific to the training environment (i.e. memorized), and how much might constitute "generalizable skills" that could be applied to new settings. In this paper, we evaluate generalization in two ways: 1) by applying the learned policy to a new scenario "as is" (no further learning), and 2) by fine-tuning the learned policy on a new scenario (we borrow the pre-training/fine-tuning nomenclature from the deep feature learning literature). We believe that evaluating generalization is a valuable tool and will allow the community to better understand the performance of various reinforcement learning algorithms. To further aid in this effort, we will make the code for our algorithm, as well as testing and environment setups freely available online.

An interesting direction of future research is to use the learned exploration behavior/skill as a motor primitive/low-level policy in a more complex, hierarchical system. For example, our *VizDoom* agent learns to walk along corridors instead of bumping into walls. This could be a useful primitive for a navigation system.

While the rich and diverse real world provides ample opportunities for interaction, reward signals are sparse. Our approach excels in this setting and converts unexpected interactions that affect the agent into intrinsic rewards. However our approach does not directly extend to the scenarios where "opportunities for interactions" are also rare. In theory, one could save such events in a replay memory and use them to guide exploration. However, we leave this extension for future work.

One potential concern with our formulation is that since the inverse model only needs to encode features relevant for predicting the agent's actions it might totally neglect representing interesting objects that the agent might interact with. For example, consider a tabletop manipulation setting where the goal of a robotic hand is to push a desired object into a target goal location. In the absence of extrinsic rewards, the agent must itself discover that pushing the object is an interesting event

that must be explored further. However, the inverse model loss would be perfectly satisfied if $\phi(s_t)$ only contains information about the location of the agent's hand and ignores all features describing the object. This is undesirable because it would imply that the agent has no information about the object and would therefore be unable to explore interesting pushing interactions. However, because we are also training the forward model simultaneously with the inverse model this issue will not happen in practice. To see why, consider this scenario: when the agent applies a 5N force in free space, its end effector moves by 1m. Now, if the agent's hand hits an object with the same amount of applied force, the hand would move by a smaller distance. This would lead to an error in the forward model prediction and force the feature space to represent the object that was hit to minimize the loss. However, if the object that the agent interacts with does not affect the agent at all then our method would fail to represent it. While this situation may often occur in simulated environments such as games, it is highly improbable in real world scenarios.

While following the above discussion, it is important to note that the feature space will learn to represent objects being manipulated only if the agent is able to make many interactions with objects in its environment. The real world is rich, diverse and complex. While it provides ample opportunities for interaction, reward signals are sparse. Our approach excels in this setting and converts unexpected interactions that affect the agent into intrinsic rewards. The hitting interactions are also not so rare, e.g. a navigating agent might keep issuing "down" command while walking on hard floor but is unable to go down. Similarly a robot arm interacting with multiple objects kept on a table is likely to hit at least one of them often enough. In scenarios where "opportunities for interactions" are actually rare, one could, in theory, save such events in a replay memory. However, we leave this extension for future work.

Chapter 7

Initial State

A very important question in the design of continual learning agents is the specification of the initial state. It has been fervently debated how much knowledge should be built in and how much of it should be left to the agent to figure out. It is obvious that starting tabula-rasa might require one to re-create evolution which is daunting and probably infeasible (note that neural networks learning from scratch also have carefully hand-crafted architecture). On the other extreme, building in too much structure runs the risk of furnishing the agent too inflexible.

This debate of innate v/s acquired knowledge is also at the core of study of human intelligence. No matter where one's belief lies on this spectrum, one thing is clear that humans are very proficient at acquiring new skills and learning about new things. As pointed out earlier, in my view, the core problem of intelligence is not how much knowledge an agent has, but how proficiently it can increase its knowledge. This view provides a cue – we should design the initial state that enables the agent to quickly increase its knowledge as opposed to optimizing for some particular end problem. To understand what forms of prior knowledge allow for quick exploration and learning, I will describe a human study investigating priors used by human subjects for efficiently learning about their environment and consequently solving sparse reward task [214]. I will then present details of a followup work that implements some of these findings to aid in downstream robotic tasks [215].

7.1 Investigating Human Priors for Playing Games

While deep Reinforcement Learning (RL) methods have shown impressive performance on a variety of video games [3], they remain woefully inefficient compared to human players, taking millions of action inputs to solve even the simplest Atari games. Much research is currently focused on improving sample efficiency of RL

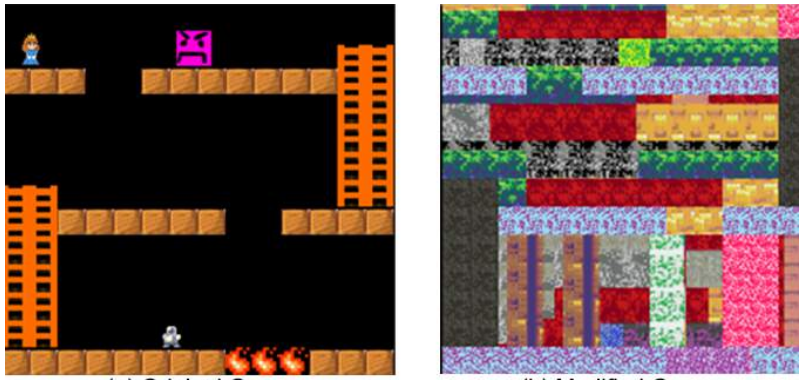


Figure 7.1: **Motivating example.** (a) A simple platformer game. (b) The same game modified by re-rendering the textures. Despite the two games being structurally the same, human players took twice as long to finish the second game as the first one. In comparison, the performance of an RL agent was approximately the same for the two games.

algorithms [216, 217]. However, there is an orthogonal issue that is often overlooked: RL agents attack each problem *tabula rasa*, whereas humans come in with a wealth of prior knowledge about the world, from physics to semantics to affordances.

Consider the following motivating example: you are tasked with playing an unfamiliar computer game shown in Figure 7.1(a). No manual or instructions are provided; you don't even know which game sprite is controlled by you. Indeed, the only feedback you are ever given is "terminal", i.e. once you successfully finish the game. Would you be able to successfully finish this game? How long would it take? We recruited forty human subjects to play this game and found that subjects finished it quite easily, taking just under 1 minute of game-play or 3000 action inputs. This is not overly surprising as one could easily guess that the game's goal is to move the robot sprite towards the princess by stepping on the brick-like objects and using ladders to reach the higher platforms while avoiding the angry pink and the fire objects.

Now consider a second scenario in which this same game is re-rendered with new textures, getting rid of semantic and affordance [218] cues, as shown in Figure 7.1(b). How would human performance change? We recruited another forty subjects to play this game and found that, on average, it took the players more than twice the time (2 minutes) and action inputs (6500) to complete the game. The second game is clearly much harder for humans, likely because it is now more difficult to guess the game structure and goal, as well as to spot obstacles.

For comparison, we can also examine how modern RL algorithms perform on

these games. This is not so simple, as most standard RL approaches expect very dense rewards (e.g. continuously updated game-score [3]), whereas we provide only a terminal reward, to mimic how most humans play video games. In such sparse reward scenarios, standard methods like A3C [191] are too sample-inefficient and were too slow to finish the games.

Hence, we used a curiosity-based RL algorithm specifically tailored to sparse-reward settings [84], which was able to solve both games. Unlike humans, RL did not show much difference between the two games, taking about 4 million action inputs to solve each one. This should not be surprising. Since the RL agent did not have any prior knowledge about the world, both these games carried roughly the same amount of information from the perspective of the agent.

This simple motivating experiment highlights the importance of prior knowledge that humans draw upon to quickly solve tasks given to them, as was also pointed out by several earlier studies [219–222]. Developmental psychologists have also been investigating the prior knowledge that children draw upon in learning about the world [223, 224]. However, these studies have not explicitly quantified the relative importance of the various priors for problem-solving. Some studies have looked into incorporating priors in RL agents via object representations [225, 226] or language grounding [227], but progress will be constrained until the field develops a better understanding of the kinds of prior knowledge humans employ.

In this particular study, we systematically quantify the importance of different types of priors humans bring to bear while solving one particular kind of problem – video games. We chose video games as the task for our investigation because it is relatively easy to methodically change the game to include or mask different kinds of knowledge and run large-scale human studies. Furthermore, video games, such as ATARI, are a popular choice in the reinforcement learning community.

This piece of work consists of a series of ablation studies on a specially-designed game environment, systematically masking out various types of visual information that could be used by humans as priors. The full game (unlike the motivating example above) was designed to be sufficiently complex and difficult for humans to easily measure changes in performance between different testing conditions.

We find that removal of some prior knowledge causes a drastic degradation in the performance of human players from 2 minutes to over 20 minutes. Another key finding of our investigation is that while specific knowledge, such as "ladders are to be climbed", "keys are used to open doors", "jumping on spikes is dangerous", is important for humans to quickly solve games, more general priors about the importance of objects and visual consistency are even more critical.

7.2 Method

To investigate the aspects of visual information that enable humans to efficiently solve video games, we designed a browser-based platform game consisting of an agent sprite, platforms, ladders, angry pink object that kills the agent, spikes that are dangerous to jump on, a key, and a door (see Figure 7.2 (a)). The agent sprite can be moved with the help of arrow keys. A terminal reward of +1 is provided when the agent reaches the door after having to taken the key, thereby terminating the game. The game is reset whenever the agent touches the enemy, jumps on the spike, or falls below the lowest platform. We made this game to resemble the exploration problems faced in the classic ATARI game of *Montezuma's Revenge* that has proven to be very challenging for deep reinforcement learning techniques [3, 182]. Unlike the motivating example, this game is too large-scale to be solved by RL agents, but provides the complexity we need to run a wide range of human experiments.

We created different versions of the video game by re-rendering various entities such as ladders, enemies, keys, platforms etc. using alternate textures (Figure 7.2). These textures were chosen to mask various forms of prior knowledge that are described in the experiments section. We also changed various physical properties of the game, such as the effect of gravity, and the way the agent interacts with its environment. Note that all the games were exactly the same in their underlying structure and reward, as well as the shortest path to reach the goal, thereby ensuring that the change in human performance (if any) is only due to masking of the priors.

We quantified human performance on each version of the game by recruiting 120 participants from Amazon Mechanical Turk. Each participant was instructed to finish the game as quickly as possible using the arrow keys as controls, but *no information* about the goals or the reward structure of the game was communicated. Each participant was paid $1 for successfully completing the game. The maximum time allowed for playing the game was set to 30 minutes. For each participant, we recorded the (x, y) position of the player at every step of the game, the total time taken by the participant to finish the game and the total number of deaths before finishing the game. We used this data to quantify the performance of each participant. Note that each participant was only allowed to complete a game once, and could not participate again (i.e. different 120 participants played each version of the game).

7.3 Quantifying the importance of object priors

[1]Different game manipulations can be played at

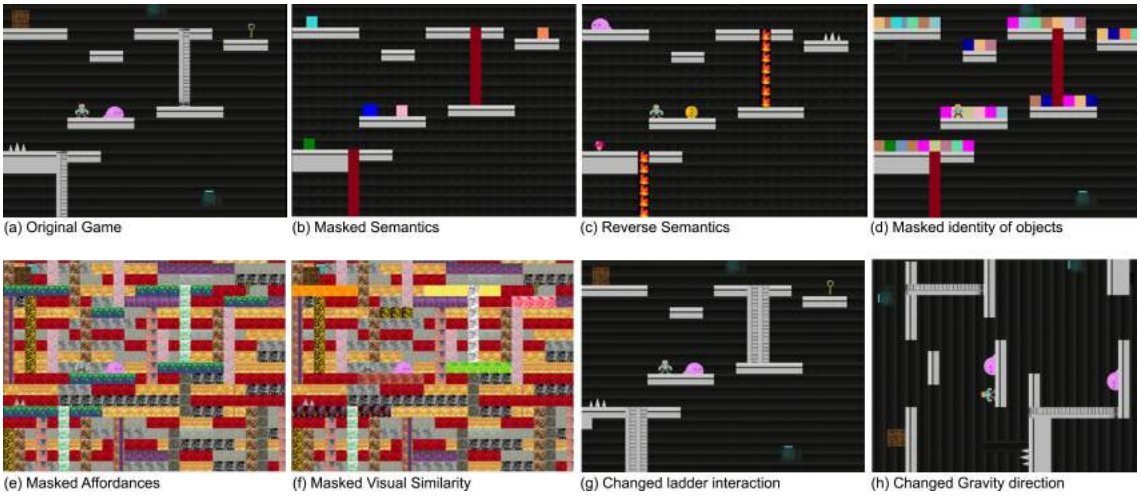


Figure 7.2: **Various game manipulations**. (a) Original version of the game. (b) Game with masked objects to ablate semantics prior. (c) Game with reversed associations as an alternate way to ablate semantics prior. (d) Game with masked objects and distractor objects to ablate the concept of object. (e) Game with background textures to ablate affordance prior. (f) Game with background textures and different colors for all platforms to ablate similarity prior. (g) Game with modified ladder to hinder participant's prior about ladder interactions. (h) Rotated game to change participant's prior about gravity. Readers are encouraged to play all these games online[1].

The original game (available to play at this link) is shown in Figure 7.2(a). A single glance at this game is enough to inform human players that the agent sprite has to reach the key to open the door while avoiding the dangerous objects like spikes and angry pink slime. Unsurprisingly, humans quickly solve this game. Figure 7.3(a) shows that the average time taken to complete the game is 1.8 minutes (blue bar) and the average number of deaths (3.3, orange bar) and unique game states visited (3011, yellow bar) are all quite small.

7.3.1 Semantics

To study the importance of prior knowledge about object semantics, we rendered objects and ladders with blocks of uniform color as shown in Figure 7.2(b). This game can be played at this link. In this version, the visual appearance of objects

[1] https://rach0012.github.io/humanRL_website/

conveys no information about their semantics. Results in Figure 7.3(b) show that human players take more than twice the time (4.3 minutes), have higher number of deaths (11.1), and explore significantly larger number of states (7205) as compared to the original game (p-value: $p < 0.01$). This clearly demonstrates that masking semantics hurts human performance.

A natural question is how do humans make use of semantic information? One hypothesis is that knowledge of semantics enables humans to infer the latent reward structure of the game. If this indeed is the case, then in the original game, where the key and the door are both visible, players should first visit the key and then go to the door, while in the version of the game without semantics, players should not exhibit such bias. We found that in the original game, nearly all participants reached the key first, while in the version with masked semantics only 42 out of 120 participants reached the key before the door (see Figure 7.4(a)). Moreover, human players took significantly longer to reach the door after taking the key as compared to the original game (see Figure 7.4(b)). This result provides further evidence that in the absence of semantics, humans are unable to infer the reward structure and consequently significantly increase their exploration. To rule out the possibility that increase in time is simply due to the fact players take longer to finish the game without semantics, the time to reach the door after taking the key was normalized by the total amount of time spent by the player to complete the game.

To further quantify the importance of semantics, instead of simply masking, we manipulated the semantic prior by swapping the semantics between different entities. As seen on Figure 7.4(c), we replaced the pink enemy and spikes by coins and ice-cream objects respectively which have a positive connotation; the ladder by fire, the key and the door by spikes and enemies which have negative connotations (see game link). As shown in Figure 7.3(c), the participants took longer to solve this game (6.1 minutes, $p < 0.01$). The average number of deaths (13.7) was also significantly more and the participants explored more states (9400) compared to the original version ($p < 0.01$ for both). Interestingly, the participants also took longer compared to the masked semantics version ($p < 0.05$) implying that when we reverse semantic information, humans find the game even tougher.

7.3.2 Objects as Sub-goals for Exploration

While blocks of uniform color in the game shown in Figure 7.2(b) convey no semantics, they are distinct from the background and seem to attract human attention. It is possible that humans infer these distinct entities (or objects) as sub-goals, which results in more efficient exploration than random search. That is, there is something special about objects that draws human attention compared to any random piece

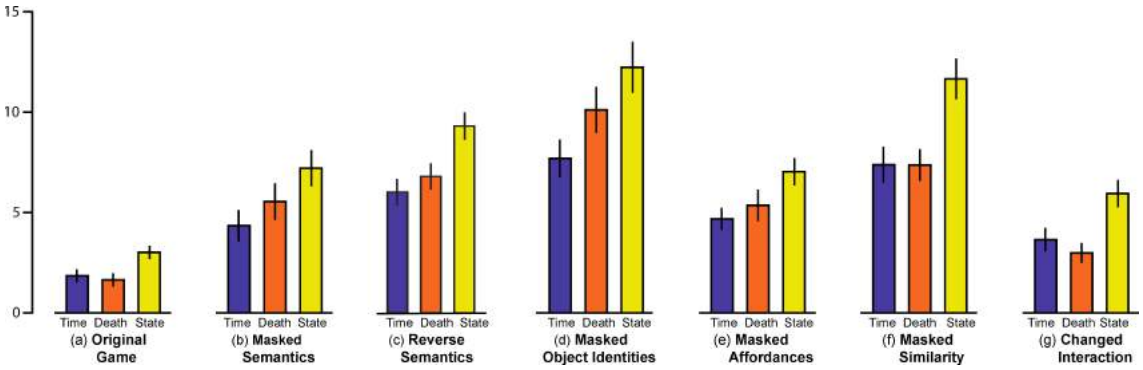


Figure 7.3: **Quantifying the influence of various object priors.** The blue bar shows average time taken by humans (in minutes), orange bar shows the average number of deaths, and yellow bar shows the number of unique states visited by players to solve the various games. For visualization purposes, the number of deaths is divided by 2, and the number of states is divided by 1000 respectively.

of texture. To test this, we modified the game to cover each space on the platform with a block of different color to hide where the objects are (see Figure 7.2(d), game link). Most colored blocks are placebos and do not correspond to any object and the actual objects have the same color and form as in the previous version of the game with masked semantics (i.e., Figure 7.2(b)). If the prior knowledge that visibly distinct entities are interesting to explore is critical, this game manipulation should lead to a significant drop in human performance.

Results in Figure 7.3(d) show that masking the concept of objects leads to drastic deterioration in performance. The average time taken by human players to solve the game is nearly four times longer (7.7 minutes), the number of deaths is nearly six times greater (20.2), and humans explore four times as many game states (12, 232) as compared to the original game. When compared to the game version in which only semantic information was removed (Figure 7.3(b)), the time taken, number of deaths and number of states are all significantly greater ($p < 0.01$). When only semantics are removed, after encountering one object, human players become aware of what possible locations might be interesting to explore next. However, when concept of objects is also masked, it is unclear what to explore next. This effect can be seen by the increase in normalized time taken to reach the door from the key as compared to the game where only semantics are masked (Figure 7.4(b)). All these results suggest that concept of objects i.e. knowing that visibly distinct entities are interesting and can be used as sub-goals for exploration, is a critical prior and perhaps more important than knowledge of semantics.

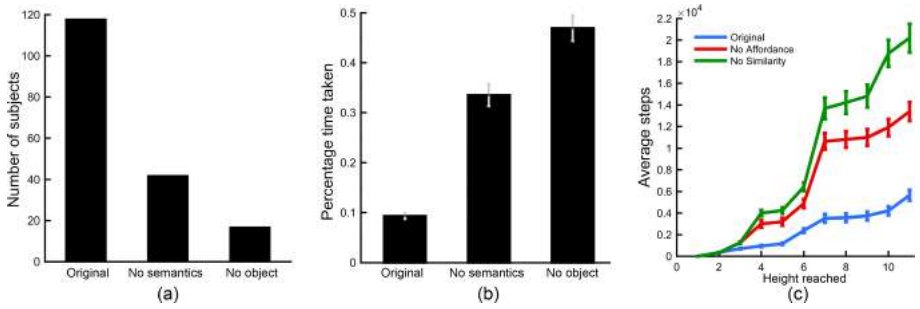


Figure 7.4: **Change in behavior upon ablation of various priors**. (a) Graph comparing number of participants that reached the key before the door in the original version, game without semantics, and game without object prior. (b) Amount of time taken by participants to reach the door once they obtained the key. (c) Average number of steps taken by participants to reach various vertical levels in original version, game without affordance, and game without similarity.

7.3.3 Affordances

Until now, we manipulated objects in ways that made inferring the underlying reward structure of the game non-trivial. However, in these games it was obvious for humans that *platforms* can support agent sprites, *ladders* could be climbed to reach different platforms (even when the ladders were colored in uniform red in games shown in Figure 7.2(b,c), the connectivity pattern revealed where the ladders were) and black parts of the game constitute *free space*. Here, the platforms and ladders *afford* the actions of walking and climbing [218], irrespective of their appearance. In the next set of experiments, we manipulated the game to mask the affordance prior.

One way to mask affordances is to fill free space with random textures, which are visually similar to textures used for rendering ladders and platforms (see Figure 7.2(e), game link). Note that in this game manipulation, objects and their semantics are clearly observable. When tasked to play this game, as shown in Figure 7.3(e), humans require significantly more time (4.7 minutes), die more often (10.7), and visit more states (7031) compared to the original game ($p < 0.01$). On the other hand, there is no significant difference in performance compared to the game without semantics, i.e., Figure 7.2(b), implying that the affordance prior is as important as the semantics prior in our setup.

7.3.4 Things that look similarly, behave similarly

In the previous game, although we masked affordance information, once the player realizes that it is possible to stand on a particular texture and climb a specific texture, it is easy to use color/texture similarity to identify other platforms and ladders in the game. Similarly, in the game with masked semantics (Figure 7.2(b)), visual similarity can be used to identify other enemies and spikes. These considerations suggest that a general prior of the form that things that look the same act the same might help humans efficiently explore environments where semantics or affordances are hidden.

We tested this hypothesis by modifying the masked affordance game in a way that none of the platforms and ladders had the same visual signature (Figure 7.2(f), game link). Such rendering prevented human players from using the similarity prior. Figure 7.3(f)) shows that performance of humans was significantly worse in comparison to the original game (Figure 7.2(a)), the game with masked semantics (Figure 7.2(b)) and the game with masked affordances (Figure 7.2(e)) ($p < 0.01$). When compared to the game with no object information (Figure 7.2(d)), the time to complete the game (7.6 minutes) and the number of states explored by players were similar ($11,715$), but the number of deaths (14.8) was significantly lower ($p < 0.01$). These results suggest that visual similarity is the second most important prior used by humans in gameplay after the knowledge of directing exploration towards objects.

In order to gain insight into how this prior knowledge affects humans, we investigated the exploration pattern of human players. In the game when all information is visible we expected that the progress of humans would be uniform in time. In the case when affordances are removed, the human players would initially take some time to figure out what visual pattern corresponds to what entity and then quickly make progress in the game. Finally, in the case when the similarity prior is removed, we would expect human players to be unable to generalize any knowledge across the game and to take large amounts of time exploring the environment even towards the end. We investigated if this indeed was true by computing the time taken by each player to reach different vertical distances in the game for the first time. Note that the door is on the top of the game, so the moving up corresponds to getting closer to solving the game. The results of this analysis are shown in Figure 7.4(c). The horizontal-axis shows the height reached by the player and the vertical-axis show the average time taken by the players. As the figure shows, the results confirm our hypothesis.

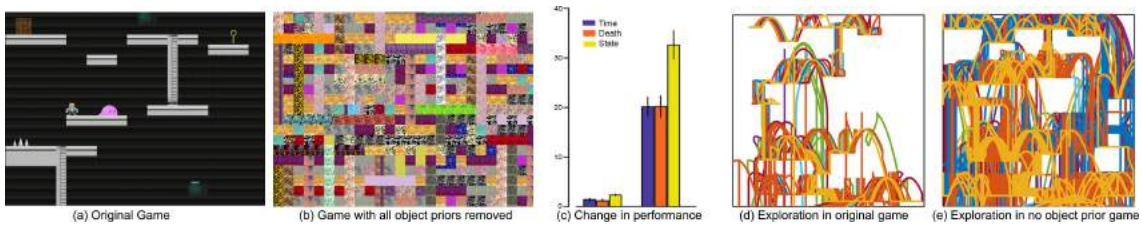


Figure 7.5: **Masking all object priors drastically affects human performance**. (a) Original game. (b) Version without any object priors. (c) Graph depicting difference in participant's performance for both the games. (d) Exploration trajectory for original version and (e) for no object prior version.

7.3.5 How to interact with objects

Until now we have analyzed the prior knowledge used by humans to interpret the visual structure in the game. However, interpretation of visual structure is only useful if the player understands what to do with the interpretation. Humans seem to possess prior knowledge about how to interact with different objects. For example, monsters can be avoided by jumping over them, ladders can be climbed by pressing the up key repeatedly etc. Deep reinforcement learning agents, on the other hand, do not possess such priors and must learn how to interact with objects by mere trial and error.

To test how critical such prior knowledge is, we created a version of the game in which the ladders couldn't be climbed by simply pressing the up key. Instead, the ladders were zigzag in nature and in order to climb the ladder players had to press the up key, followed by alternating presses between the right and left key. Note that the ladders in this version looked like normal ladders, so players couldn't infer the properties of the ladder by simply looking at them (see Figure 7.2(g), game link). As shown in Figure 7.3(g), changing the property of the ladder increases the time taken (3.6 minutes), number of deaths (6), and states explored (5942) when compared to the original game ($p < 0.01$).

7.4 Concealing all object priors

In previous sections, we studied how different priors about objects affect human performance one at a time. To quantify human performance when all object priors investigated so far are simultaneously masked, we created the game shown in Figure 7.5(b) that hid all information about objects, semantics, affordance, and similarity(game link). Results in Figure 7.5(c) show that humans found it extremely

hard to play this game. The average time taken to solve the game increased to 20 minutes and the average number of deaths rose sharply to 40. Remarkably, the exploration trajectory of humans is now almost completely random as shown in Figure 7.5(e) with the number of unique states visited by the human players increasing by a factor of 9 as compared to the original game. Due to difficulty in completing this game, we noticed a high dropout of human participants before they finished the game. We had to increase the pay to $2.25 to encourage participants not to quit. Many participants noted that they could solve the game only by memorizing it.

Even though we preserved priors related to physics (e.g., objects fall down) and motor control (e.g., pressing left key moves the agent sprite to the left), just by rendering the game in a way that makes it impossible to use prior knowledge about how to visually interpret the game screen makes the game extremely hard to play. To further test the limits of human ability, we designed a harder game where we also reversed gravity and randomly re-mapped the key presses to how it affect's the motion of agent's sprite. We, the creators of the game, having played a previous version of the game hundreds of times had an extremely hard time trying to complete this version of the game. This game placed us in the shoes of reinforcement learning (RL) agents that start off without the immense prior knowledge that humans possess. While improvements in the performance of RL agents with better algorithms and better computational resources is inevitable, our results make a strong case for developing algorithms that incorporate prior knowledge as a way to improve the performance of artificial agents.

7.5 Physics and motor control priors

In addition to prior knowledge about objects, humans also bring in rich prior knowledge about intuitive physics and strong motor control priors when they approach a new task [228–231]. Here, we have taken some initial steps to explore the importance of such priors in context of human gameplay.

7.5.1 Gravity

One of the most obvious forms of knowledge that we have about the physical world is with regards to gravity, i.e., things fall from up to down. To mask this prior, we created a version of the game in which the whole game window was rotated 90° (refer to Figure 7.2(h)). In this way, the gravity was reversed from left to right (as opposed to up to down). As shown in Figure 7.6, participants spent more time to

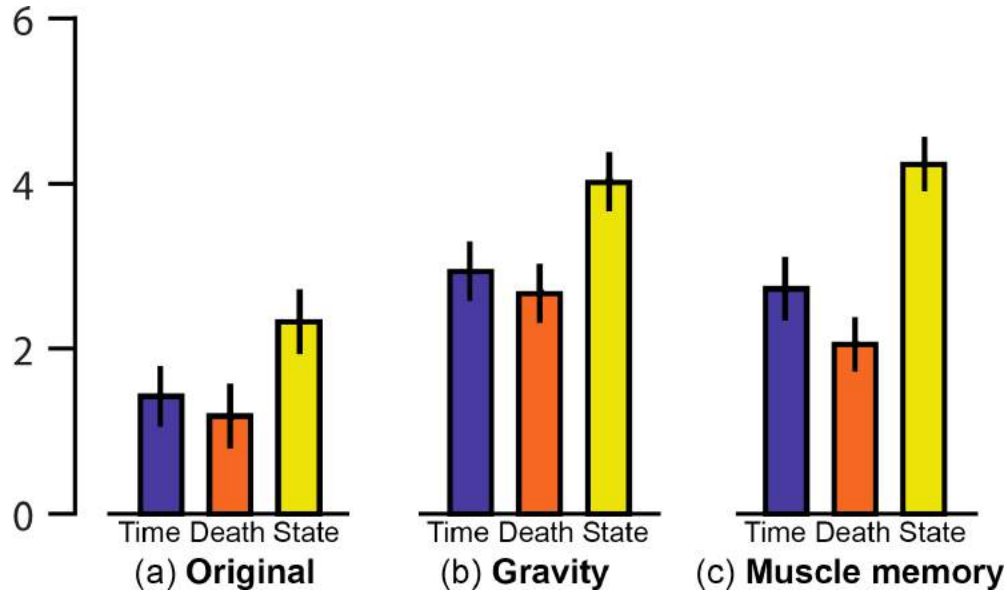


Figure 7.6: **Physics and motor control priors.** Performance of participants in original version, game with gravity reversed, and game with key controls reversed.

solve this game compared to the original version with average time taken close to 3 minutes ($p < 0.01$). The average number of deaths and number of states explored was also significantly larger than the original version ($p < 0.01$).

7.5.2 Muscle memory

Human players also come with knowledge about the consequences of actions such as pressing arrow keys moves the agent sprite in the corresponding directions (i.e., pressing up makes the agent sprite jump, pressing left makes the agent sprite go left and so forth). We created a version of the game in which we reversed the arrow key controls. Thus, pressing the left arrow key made the agent sprite go right, pressing the right key moved the sprite left, pressing the down key made the player jump (or go up the stairs), and pressing the up key made the player go down the stairs. Participants again took longer to solve this game compared to the original version with average time taken close to 3 minutes (refer to Figure 7.6). The average number of deaths and number of states explored was also significantly larger than the original version ($p < 0.01$). Interestingly, the performance of players when the gravity was reversed, and key controls were reversed is similar, with no significant difference between the two conditions.

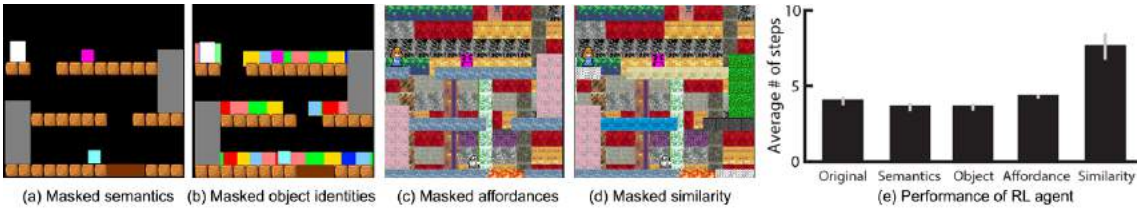


Figure 7.7: **Quantifying the performance of RL agent.** (a) Game without semantic information. (b) Game with masked and distractor objects to ablate concept of objects. (c) Game without affordance information. (d) Game without similarity information. (e) Performance of RL agent on various game manipulations (steps shown in order of million). Error bars indicate standard error of mean for the 5 random seeds. The RL agent performs similarly on all games except for the one without visual similarity.

7.6 Controlling for change in complexity

So far in this paper, we have manipulated various visual priors while keeping the underlying game and reward structure exactly the same. We have assumed that this will influence human performance while keeping RL agent performance unchanged, since RL does not have any priors to begin with. However, one possible confound is that the visual complexity of the modified games might have changed from the original game version, because masking out priors without changing visual complexity is extremely difficult.

To control for this confound, we investigated the performance of an RL agent on the various game manipulations. If RL agents are not affected by the game manipulations, then it would suggest that prior knowledge and not visual complexity is the main reason behind the change in human performance. Note that this confound is not present in the physics and motor control experiments as the visual input stays the same as the original game.

To this end, we systematically created different versions of the game in Figure 7.1(a) to ablate semantics, the concept of object, affordance, and similarity as shown in Figure 7.7. Note that the game used for human experiments shown in Figure 7.2 is more complex than the game used for RL experiments in Figure 7.7. This is because the larger game was simply too hard for state-of-the-art RL agents to solve. Apart from the difference in the game size, we tried to make the games as similar as possible. Even though this version of the game is simpler (regarding size, number of objects etc.), we note that this game is still non-trivial for an RL agent. For instance, due to the sparse reward structure of the game, both A3C [191] and

breadth-first search didn't come close to solving the game even after 10 million steps. Hence, for our purpose, we used an RL algorithm augmented with a curiosity based exploration strategy [84]. For each game version, we report the mean performance of five random seeds that succeeded.

As shown in Figure 7.7(e), the RL agent was unaffected by the removal of semantics, the concept of objects, as well as affordances – there is no significant difference between the mean score of the RL agent on these games when compared to the performance on the original game ($p > 0.05$). This suggests that the drop in human performance in these game manipulations is not due to the change in visual complexity, but it is rather due to the masking of the various priors. On the other hand, the performance of the RL agent does worsen when visual similarity is masked as it takes nearly twice as many interactions to complete the game compared to the original version. We believe this is due to to the use of *convolutional* neural networks that implicitly impose the prior of visual similarity rather than simply due to the change in visual complexity.

7.7 Discussion

While there is no doubt that the performance of deep RL algorithms is impressive, there is much to be learned from human cognition if our goal is to enable RL agents to solve sparse reward tasks with human-like efficiency. Humans have the amazing ability to use their past knowledge (i.e., priors) to solve new tasks quickly. Success in such scenarios critically depends on the agent's ability to explore its environment and then promptly learn from its successes [232,233]. In this vein, our results demonstrate the importance of prior knowledge in helping humans explore efficiently in these sparse reward environments [234, 235].

While the study presented above primarily investigated object priors (and physics priors to some extent), humans also possess rich prior knowledge about the world in the form of intuitive psychology and also bring in various priors about general video game playing such as that moving up and to the right in games is generally correlated with progress, games have goals, etc. Studying the importance of such priors will be an interesting future direction of research.

Building RL algorithms that require fewer interactions to reach the goal (i.e., sample efficient algorithms) is an active area of research, and further progress is inevitable. In addition to developing better optimization methods, we believe that instead of always initializing learning from scratch, either incorporating prior knowledge directly or constructing mechanisms for condensing experience into reusable knowledge (i.e., learning priors through *continual* learning) might be critical for building RL

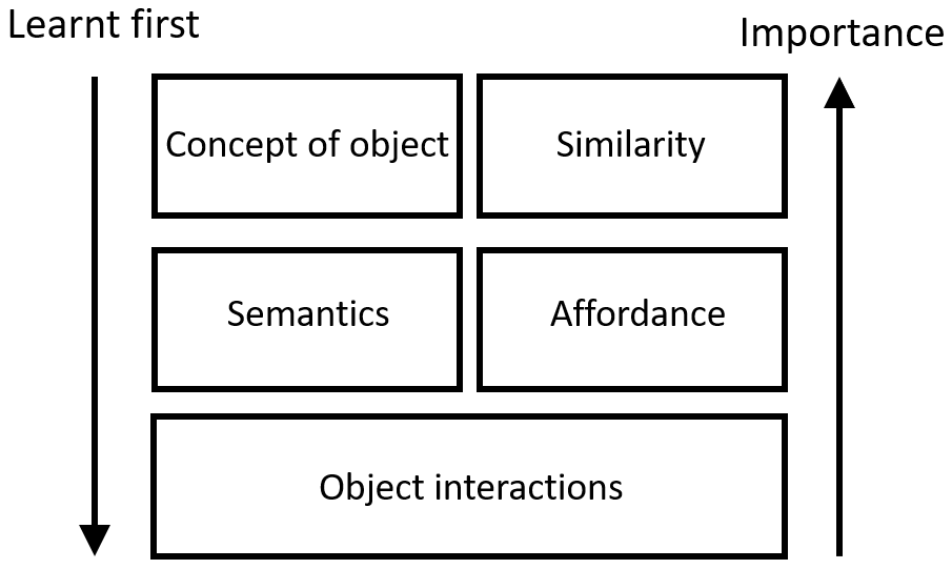


Figure 7.8: **Taxonomy of object priors.** The earlier an object prior is obtained during childhood, the more critical that object prior is in human problem solving in video games.

agents with human-like efficiency. Our work takes first steps toward quantifying the importance of various priors that humans employ in solving video games and in understanding how prior knowledge makes humans good at such complex tasks. We believe that our results will inspire researchers to think about different mechanisms of incorporating prior knowledge in the design of RL agents. We also hope that our experimental platform of video games, available in open-source, will fuel more detailed studies investigating human priors and a benchmark for quantifying the efficacy of different mechanisms of incorporating prior knowledge into RL agents.

While there are many possible directions on how to incorporate priors in RL and more generally AI agents, it is informative to study how humans acquire such priors. Studies in developmental psychology suggest that human infants as young as 2 months old possess a primitive notion of objects and expect them to move as connected and bounded wholes that allows them to perceive object boundaries and therefore possibly distinguish them from the background [223, 236]. At this stage, infants do not reason about object categories. By the age of 3-5 months, infants start exhibiting categorization behavior based on similarity and familiarity [237, 238]. The ability to recognize individual objects rapidly and accurately emerges comparatively late in development (usually by the time babies are 18-24 months old [239]). Similarly, while young infants exhibit some knowledge about affordances early during development,

the ability to distinguish a walkable step from a cliff emerges only by the time they are 18 months old [240].

These results in infant development suggest that starting with a primitive notion of objects, infants gradually learn about visual similarity and eventually about object semantics and affordances. It is quite interesting to note that the order in which infants increase their knowledge matches the importance of different object priors such as the existence of objects as sub-goals for exploration, visual similarity, object semantics, and affordances. Based on these results, we suggest a possible taxonomy and ranking of object priors in Figure 7.8. We put 'object interaction' at the bottom as in the context of our problem, knowledge about how to interact with specific objects can be only learned once recognition is performed.

Chapter 8

Intuitive Behavior

8.1 Forecasting Player Moves in Sports Videos

In 2002, Billy Beane defied conventional wisdom by performing meticulous statistical evaluations of undervalued players to assemble the Oakland Athletics baseball team on a negligible budget. His team made history with a record-setting 20-game win streak, and this tremendous feat is documented in the academy award nominated film *Moneyball*. Their success made an unprecedented case for competitive advantages gained by new analyses of individual players' game play. Now imagine if, in addition to knowing the shot success rate of Stephen Curry, the best basketball shooter, it is also possible to forecast that he is more likely to attempt a shot within zero, one, and two seconds of a pass when his teammates are in a diamond, ring, and triangle formation, respectively. Such forecasts are invaluable to the defending team in planning strategy. Billy Beane's analysis revolutionized strategic thinking in baseball, and similarly, we believe statistical methods for forecasting player moves have the potential to impact how teams plan their play strategies.

Predicting player moves in sports videos is an instance of a much grander research agenda to develop algorithms that can forecast future events directly from visual inputs. The ability to forecast is a key aspect of human intelligence, and as Kenneth Craik famously wrote in 1943, "*If the organism carries a 'small scale model' of external reality and its own possible actions within its head, it is able try out various alternatives, conclude which is the best of them, react to future situations before they arise and in every way react in much fuller, safer and more competent manner to emergencies which face it.*" While there has been a lot of interest in this problem [85, 104, 130, 241–251], we lack a good benchmark for comparing different forecasting algorithms.

For multiple reasons, it appears to us that team sports videos are a very good

benchmark for evaluating forecasting algorithms. Firstly, many human activities are social and team sports provide an opportunity to study forecasting in an adversarial multi-agent environment. Secondly, team sports are composed of a large and diverse set of discrete events, such as passing, shooting, dribbling, etc. The sequence of events reflects the game play strategies of the two teams, and thus forecasting requires game specific knowledge combined with other visual cues, such as player pose and game state. This implies that for any system to make accurate predictions directly from visual imagery, it must distill game specific knowledge by crunching large amounts of data. Representing such knowledge is a central problem in forecasting, which is put to test in this setup. Expert players and coaches gain such knowledge via experience gathered over long periods of time. An additional benefit of predicting discrete events is crisp and straightforward evaluation of the information of interest that avoids the problems associated with evaluating pixel-level predictions.

8.1.1 Related Work

Video analysis is an active research area. A large body of work has focused on action recognition [89, 252–256], people and object tracking [257–259]. In contrast to these works we are interested in the problem of forecasting. Predicting pedestrian trajectories [241, 260–263] and anticipating future human activities [241, 242, 246, 264–266] has seen considerable interest over the past few years. However, these works mostly consider predicting events related to a single human, while we attempt to forecast events in multi-agent environments involving adversarial human-human interaction. Other works have explored predicting future world states from single images [132, 247, 249, 267], but have been limited to simulation environments or involve a single agent. Predicting pixel values in future frames has also drawn considerable interest [104, 244] but is limited to very short term predictions.

Sport Video Analysis: Traditional work in computer vision analyzing sports videos [268] has focused on either tracking players [269] or balls [270]. Another body of work assumes the annotations of ball or player tracks to analyze game formations or skill level of individual players. For instance, [271] use tracks of ball trajectories in tennis games to predict where the ball would be hit, [272] analyze soccer matches using player tracks. [273] discover team formation and plays using player role representation instead of player identity. More recently techniques such as [274] have looked at the problem of identifying the key players and events in basketball game videos. Closest to our work is the work of [275] that proposes the use of hidden conditional random fields for predicting which player will receive the ball next in soccer games. They assume the knowledge of game state such as attack, defense, counter attack, free kick etc. and assume that identity of players is known.

In contrast, we present a forecasting system that works directly from visual inputs. It either uses images directly or converts them it into an overhead view representation using computer vision techniques. We do not require any external annotations of the game state.

8.1.2 Team Sports Datasets

We have focused our efforts on the most popular style of sport, *team goal sports*. We select water polo and basketball as two canonical examples because together they capture many diverse aspects of team goal sports: basketball is fast-moving and high-scoring like hurling and handball, while water polo is low-scoring like soccer and has man-up situations like hockey and lacrosse. Despite the different nuances of each team goal sport, they all share many common "events" during game play. For example, players advance the ball toward the goal themselves in a *drive*, and sometimes this results in a *goal* and other times in a *block* or a *missed shot*. Players *pass* the ball to their teammates, and sometimes the defense intercepts the pass.

Water polo

A water polo game naturally partitions into a sequence of alternating team possessions. During a possession, the attacking team's field players primarily spend time in their *front court*, which accounts for most of the interesting game play. The attacking team is required to take a shot within 30s, and failure to do so results in a *turnover*. Players of the two teams wear dark colored (typically blue/black) and light colored (typically white) caps. In the remainder of the paper we use dark-cap and light-cap to refer to two teams.

We collected a dataset of front court team possessions from video recordings of 5 water polo games between Division I National Collegiate Athletic Association varsity teams. Similar to the NBA for basketball, this represents the highest level of water polo play in the United States. We chose to focus only on front court possessions, as most interesting events happen during this period. The time intervals of the front court possessions were hand-marked by an expert water polo coach. All the games, four of which are men's games and the other a women's game, are played at the same outdoor pool on different days at times ranging from morning until night; the dataset exhibits a large range of lighting conditions. The games were recorded with a single freely moving camera that pans between each side of the pool with resolution 720p at 25-30fps. Often the camera is adjusted for a new front court possession, resulting in varied camera motions and zooms.

Player and Pool Track Annotations: Bounding box track annotations (Figure

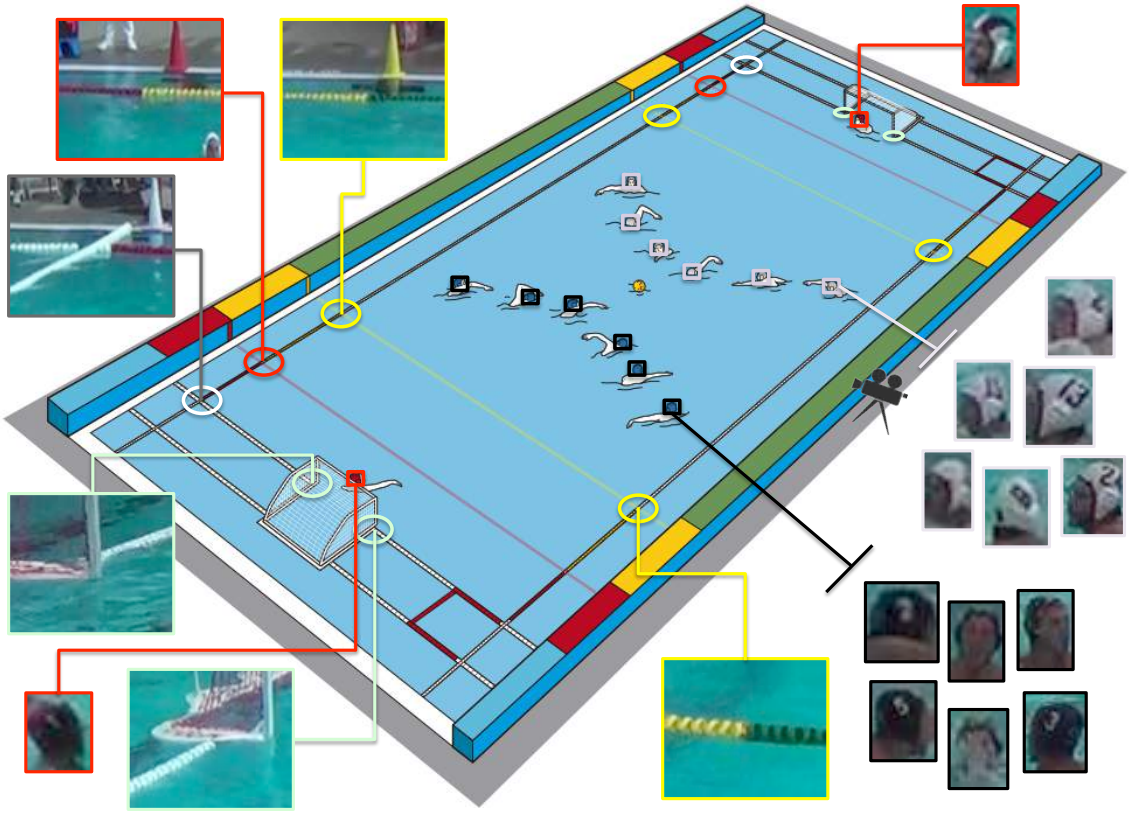


Figure 8.1: From single-camera video recordings of 5 water polo games, we collected bounding box track annotations of dark, light, and red-cap player heads. We also collected annotations of pool points marking the field of play: the 2m and 5m lines, the corner of the field, and the points where the cage and lane line meet.

8.1) of dark and light-cap player heads, goalkeepers, and the head of the player in possession of the ball were collected using the VATIC toolbox [276] and Amazon Mechanical Turk. Player possession is defined to begin at the moment a player grasps the ball and ends at the moment that same player passes/shoots the ball or another player steals the ball. Additional annotations of specific points marking the field of play: the 5m line, the 2m line, the pool corner, and the cage posts were obtained. These field markings provide necessary point correspondences between the image view and overhead view of the game, which enable the computation of the player trajectories in the overhead space from the player locations in the image view. For increased data diversity, annotations were collected for 11 quarters of play from 20 quarters available in the 5 games.

Train/Test Splits: The splits were as follows - *train*: 7 quarters, randomly sampled from the first 4 games; *validation*: light-capped team front court possessions in all 4 quarters of the fifth game; and *test*: dark-capped team front court possessions in all 4 quarters of the fifth game. In total, each split has 232, 134, and 171 respective examples of a player passing the ball in a team's front court.

Human Benchmark: Human subjects were shown every test image taken just before a player loses possession of the ball and were required to draw a bounding box around the head of the player which they thought would possess the ball next. Two sets of subjects: nine non-experts and four water polo experts were evaluated. Non-experts had never seen or played a water polo game. In order to account for their inexperience, non-experts were shown all examples used to train computer algorithms along with the the ground-truth answer before being asked to make predictions. The experts had all played competitive water polo for at least four years. Expert and non-expert humans accurately predicted the next ball possessor 73% and 55.3% of the time respectively.

Basketball

The dataset is comprised of ground truth (in contrast to water polo, where it is computed) 2D player and ball trajectories, sampled at 25 Hz, in 82 regular-season NBA games obtained using the STATS SportVU system [277], which is a calibrated six-camera setup in every NBA arena. The data includes labels for 16 basketball events, including free throw, field goal, pass, dribble, (player) possession, etc. that are detailed in the supplementary materials.

Train/Test Splits: A total of roughly $300k$ labeled events were randomly split into $180k$, $30k$, and $90k$ for train, validation, and test examples.

Human Benchmark: A set of 18 subjects familiar with basketball were shown a series of fifteen 5-second clips of basketball data, ending with a labeled event. The

ball and player trajectories were removed from the final n seconds of the clip, and the subjects were asked to predict the event at the end of the blanked portion. For each $n \in \{0.4, 1, 2\}$, each subject was shown 5 examples randomly sampled from a pool of 80 examples (5 examples of each of the 16 events). Humans were correct 13.5%, 20.6%, and 24.4% for $n = 2$, 1, and 0.4, respectively.

8.1.3 Methods: From Images to Overhead View

2D overhead view of the game where players are represented as dots at their (x, y) coordinate locations is often used by coaches because it provides immediate insight into player spacing, and distills player movement into a canonical, simple representation that is easy to compare across many game plays. We construct the overhead representation by first detecting players and ball. Using knowledge of playing field dimensions and locations of few landmarks, we estimate a homography to transform these detections into a canonical frame. We then link players across frames using tracking. Each step of this process is detailed below.

Player Detection: Off the shelf person detectors (like Fast-RCNN trained on COCO [19]) perform very poorly on the water polo data as most of the body is occluded by water and the size of fully visible player head is only about 30×30 pixels. We finetune VGG-16 network pre-trained on Imagenet for detecting light and dark cap players using Fast R-CNN and the annotations we collected described in section 8.1.2. The performance of dark and light color cap person detectors was 73.4% and 60.4%, respectively. We attribute the worse performance of the light-color cap detector to a few confounding factors: 1) many light-color caps were annotated, by one turker, with loose bounding boxes, 2) overhead lights produce specularities and water splashes can appear visually similar to light-color caps.

Player Tracking: We track by detection. The Hungarian matching algorithm [278] is used to link Fast-RCNN player detections to form player tracks. The pairwise affinity between detections in two sequential frames is a linear combination of Euclidean distance and bounding box overlap. Player tracking is essential for identifying who is the player in the current image that will receive the ball in the future. Additionally, tracking also helps prune spurious detections.

Overhead Projection: In the case of water polo (Figure 8.2) we used the annotations of 2m and 5m lines, the pool corner, and the cage posts to estimate the homography between the current image captured by the camera and a canonical 2D coordinate system representing the field of play using the normalized direct linear transformation (DLT) algorithm [279]. Next, we transform the midpoint of bottom edge of the player bounding box into a (x, y) location in the canonical frame. We use the bottom edge because that is the point of the player that is closest to the

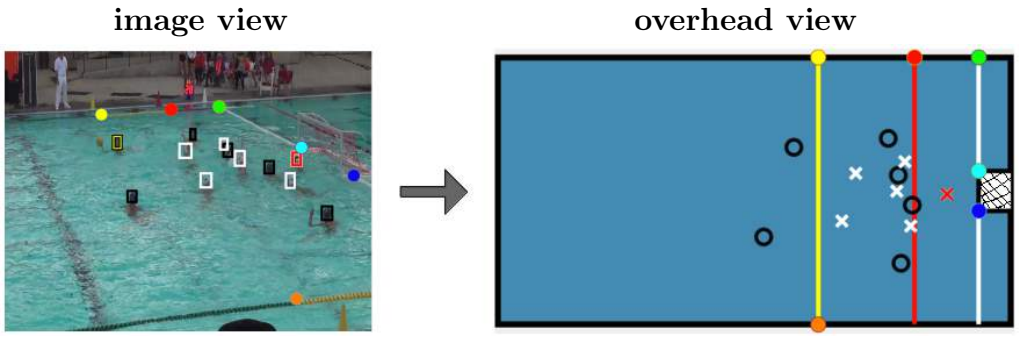


Figure 8.2: The image is converted into the overhead view by first estimating the homography between the image a canonical model of the playing field using field markings such as 2m/5m lines etc. The players are then detected and their locations are transformed using the estimated homography.

field of play, which in turn is mapped to the canonical frame by the homography transformation.

8.1.4 Forecasting Future Ball Position

The movement of the ball determines the game outcome, and therefore, it is the most important object in play at any moment of the game. We focus directly on the most important question during the game: where will the ball go next? We study two slightly different variants of this question: In the water polo dataset, we only consider the frame before which the ball possessor is about to lose of the possession of the ball, and we try to forecast which player will be in possession of ball next. In the basketball domain, we have access to much more data, and we additionally attempt the more general problem: where will the ball be in one or two seconds in the future?

Water polo: Who will possess the ball next?

In the typical front court water polo scene, there are 6 field players on the attack, defended by 1 goalkeeper and 6 field players on the opposing team. For example, in Figure 8.2, the dark-cap players are on the attack and the light-cap players are on defense. By definition, one of the attacking team players is in possession of the ball. Our system takes as input the frame just before the player loses ball possession by either making a pass to a teammate, shooting the ball, or committing a turnover. The task is to predict which player will possess the ball next.

A random choice of player from either team would be correct roughly $\frac{1}{12} \approx 8.3\%$

$\boldsymbol{F_{[1,2]}}$: (x,y) of player with ball

$\boldsymbol{F_{[3,4]}}$: (x,y) of player

$\boldsymbol{F_{[5,6]}}$: (x,y) of nearest defender

$\boldsymbol{F_7}$: same-team flag

$\boldsymbol{F_8}$: $||F_{[3,4]} - F_{[1,2]}||_2$

$\boldsymbol{F_9}$: $||F_{[3,4]} - F_{[5,6]}||_2$

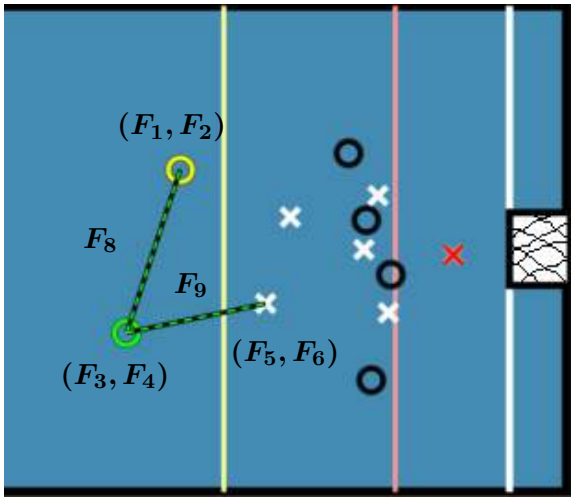


Figure 8.3: The features $F_{[1...9]}$ extracted from the 2D overhead view are used to train a random forest to classify players as either receiving the ball next or not.

of the time. As a player is more likely to pass the ball to his teammate, a random choice of player from the same team would be correct approximately 20% of the time (empirically validated on the test set). Such random guesses are very naive. Players often tend to pass the ball to nearer teammates, as shorter passes are easier to execute and minimize turnover risk. Predicting the nearest teammate as the next possessor is correct 28.1% of the time. Players also tend to pass the ball to open teammates, those who are not closely guarded by defenders. Predicting a pass to a teammate who is furthest from his nearest defender (i.e. most open) has accuracy of 36.7%. These baselines are considerably worse than an average human with no water polo expertise, who is correct 55.3% of the time.

In the next two sections, we describe how performance can be improved: (1) using additional player features estimated from the overhead representation, and (2) automatically learning feature representations directly from the input image. We operationalize these approaches in the following way: Let there be K players each with feature vector $F^i (i \in \{1, 2.., K\})$, let $b \in \{1, 2.., K\}$ be a discrete random variable that encodes the player in possession of the ball after a pass is made. The goal is to find the player who is most likley to receive the ball, i.e. $\text{argmax}_i \, P(b = i | F^1 .. F^K)$.

Hand designed features from overhead view

When deciding where to pass the ball, players consider which teammates are in good position to: score, advance the ball, and receive a pass. We formalize these insights and characterize each player using a 9-D feature vector extracted from the

overhead representation: the (x, y) player coordinates, the (x, y) coordinates of the nearest player on the opposite team, the (x, y) coordinates of the player in possession of the ball, an indicator flag for whether the player is on the same team as the player in possession of the ball, and the Euclidean distances of the player to the player with the ball and to his nearest defender. This player-centric feature vector is illustrated in Figure 8.3. We assume that features $F^1..F^k$ are mutually independent, and therefore computing $P(b = i|F^1..F^K)$ reduces to estimating $P(b = i|F^i)$.

We train a system to infer which player will possess the ball next in the following way: we used the pipeline described in section 8.1.3 to convert the raw image into it's corresponding overhead representation. Next, feature vector of each player was computed from the overhead representation. Finally, a random forest classifier was trained on these features using the training data to estimate $P(b = i|F^i)$. Five-fold cross-validation was performed to chose the optimal depth and number of trees. This system achieved a performance of 45.5% (see Table 8.1) and outperformed the baseline methods on the testing set. Analysis of the results revealed that this method is biased towards predicting the most open perimeter player as the one receiving the ball.

A common failure mode is predicting an open perimeter player, when he is not even facing the player in possession of the ball. These mistakes are not surprising as the overhead view has no access to visual appearance cues. Another possible reason for failures is that the pipeline for converting image data into overhead representation is inaccurate. To tease this apart, we re-ran the analysis using ground truth (instead of estimated) detections. As reported in Table 8.1, the accuracy gap with and without using ground truth detection is within the error bar of the performance on the testing set. This suggests that the pipeline for obtaining overhead representation is accurate and further performance improvements will be gained by building better forecasting models.

Forecasting directly from image space While the overhead view provides a good representation for analyzing game play, it loses subtle visual cues, such as the pose of the player and direction they are facing, that might be very relevant for forecasting. Instead of hand-designing such features, is it possible to automatically discover features that are useful for forecasting next ball possession?

The set of features $F^1..F^k$ is represented by image I_t and we compute $P(b = i|I_t)$ in the following manner: Let l_b, p^k be random variables denoting the future location of the ball and the k^{th} player respectively after the passed ball is received. Since only one player can receive the ball, we assume that if the ball is at location l_b it will be received by the player who has highest probability of presence at l_b (i.e. $\arg\max_k P(p^k = l_b)$.

Method	Ground Truth Heads	Detected Heads
Random, either team	9.5 ± 2.2	9.2 ± 2.2
Random teammate	19.1 ± 3.1	17.0 ± 2.8
Nearest neighbor teammate	28.1 ± 3.4	22.2 ± 3.2
Most open teammate	36.7 ± 3.7	28.7 ± 3.4
$F[8 \ldots 9]$	42.5 ± 3.8	35.2 ± 3.6
$F[7 \ldots 9]$	45.4 ± 3.4	38.4 ± 4.0
$F[3 \ldots 9]$	**48.8 ± 4.3**	44.1 ± 3.7
$F[1 \ldots 9]$	47.1 ± 3.8	**45.5 ± 3.5**
FCN, teammate	38.1 ± 3.5	35.2 ± 3.6
Human, Non-Expert	55.3 ± 7.9	-
Human, Expert	73.1 ± 2.0	-

Table 8.1: Each row reports accuracy of a different method for predicting which player will possess the ball next. The first four methods are baselines. The intermediate rows provide an ablation study of using various features defined above. The FCN is a deep learning based method and the last two rows report human performance. Performance metrics are reported for two circumstances: using ground truth player locations (column 1) and when detected instead of ground-truth locations (column 2) are used.

Let l_b^i denote the set of all locations at which $i = \arg\max_k P(p^k = l_b)$. With this,

$$P(b = i|I_t) = \int_{l_b \in l_b^i} P(p^k = l_b, l_b|I_t) \tag{8.1}$$

assuming conditional independence,

$$= \int_{l_b \in l_b^i} P(p^k = l_b|I_t) P(l_b|I_t) \tag{8.2}$$

We model $P(l_b|I_t)$ using a Fully convolutional neural network (FCN; [280]), that takes I_t as input and predicts a confidence mask of the same size as the image encoding $P(l_b|I_t)$. The ground truth value of mask is set to 1 in pixel locations corresponding to bounding box of the player who receives the ball and zero otherwise. The player bounding box is a proxy for future ball location. We finetuned Imagenet pre-trained VGG-16 network for this task.

As we only have 232 training examples, this vanilla system unsurprisingly did not perform well and overfit very easily even with standard data augmentation techniques such as image cropping and dropout regularization. One of our contributions is in

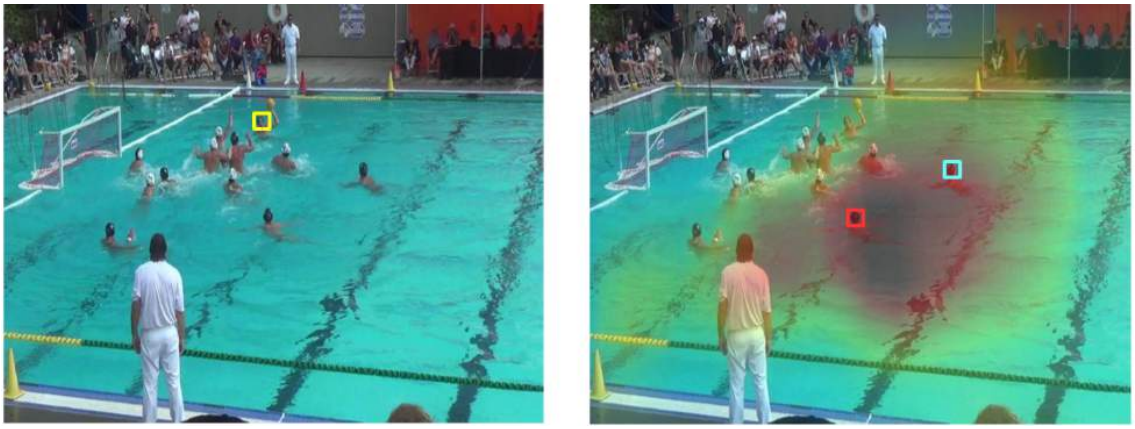


Figure 8.4: The FCN method (section 8.1.4) takes the left image as input and predicts a heatmap (shown overlaid on right) encoding probable ball locations after the ball is passed. The yellow, cyan and red squares indicate the player with the ball, the ground truth player who actually receive the ball next, and the player predicted to receive the ball by the FCN method respectively.

		Non-expert Humans	
		Correct	Incorrect
Random Forest	Correct	32.8	15.8
	Incorrect	22.8	28.6

Table 8.2: Comparing agreement between the predictions of next ball possessor made by humans and our best algorithm on the water polo data. Humans and the algorithm both make correct and incorrect predictions on the same examples more often than not.

showing that the performance can be significantly improved (from 10% to 38.1%) by requiring the FCN system to output the location of players in addition to which player will possess the ball next. Our hypothesis about why this modifications helps is that forcing the CNN to predict player locations results in representations that capture the important feature of player configurations and are thus more likely to generalize than other nuisance factors that the CNN can latch onto given the small size of the training set. This finding is interesting because it suggests that it might be possible to learn even better features by forecasting future player locations for which no additional annotation data is required once the detection and tracking pipeline described in the previous sections is setup.

To estimate $P(p^k = l_b|I_t)$ we first detect all the players in image I_t using the method described in section 8.1.3. We assume that players will be at the same location after the pass is made. In order to make the ball assignment among players to be mutually exclusive, we use the player locations to perform a Voronoi decomposition of the playing field. Let c^k be the voronoi cell corresponding to the k^{th} player. $P(p^k = l_b)$ is then to set to $\frac{1}{|c^k|}$ if $l_b \in c^k$ and zero otherwise. We then use equation (2) to compute $P(b = i|I_t)$.

This method performs comparably to the baseline that predicts the most open teammate. Visualization in Figure 8.4 shows a dominant pattern with FCN predictions: it consistently places higher likelihood around the perimeter of team in possession of the ball. This is a very sensible strategy to learn because players around the perimeter are often more open and statistical analysis reveals that there are more passes between perimeter players. Given the limited amount of data, the FCN based approach is unable to capture more nuanced aspects of player configurations or more fine grained visual cues such as the player pose.

Comparison to Human Performance Figure 8.5 compares the predictions of human non-experts against our best performing system. Some common trends are: Non-experts are more likely to incorrectly predict players near the cage. Table 8.2 reports agreement statistics between the predictions of our systems and non-expert humans. These numbers suggest that humans and our system have similar biases and are accurate/prone to errors on similar examples.

8.1.5 Basketball: Where will the ball go?

As more data was available for basketball, we attacked the more general problem of predicting where the ball will go next after one and two second respectively. We represented the overhead view as 64x64x3 images where the three channels corresponded to location of players of team 1, players of team 2 and the ball respectively. For capturing temporal context, we included 5 images from the past taken at times $\{t, t-1, ...t-4\}s$ respectively. The task was to predict the ball location at times $\{t+1, t+2\}s$ respectively. To account for multimodality in the output, we formulate this as a classification problem with the xy plane discretized into 256 bins.

We experiment with two different CNN training strategies: (a) early fusion of the temporal information by concatenating 5 images into a 15 channel image that was fed into a CNN or, (b) late fusion by using a LSTM on the output of CNN feature representation of the 5 images. The CNN architecture comprised of 4 convolutional layers containing 32 filters each of size 3x3, stride 2 and ReLU non-linearity. In case

Method	Error (1s in Future)				Error (2s in Future)			
	Distance (%)		Angle (^{o})		Distance (%)		Angle (^{o})	
	Mean	Median	Mean	Median	Mean	Median	Mean	Median
Last Position	11.7	10.4	-	-	20.0	18.3	-	-
Ball Velocity	100	100	89.3	88.7	100	100	88.8	85.9
CNN + LSTM	11.4	8.6	61.8	46.6	17.1	14.1	**53.1**	**38.1**
CNN (Early Fusion)	**10.8**	**8.3**	**60.2**	**44.1**	**16.8**	**13.8**	54.3	38.3

Table 8.3: The early fusion CNN outperforms Last Position and Ball Velocity baseline methods and a late fusion CNN based approach in predicting (basket)ball position 1s and 2s in the future. We report mean and median errors in the distance and angle of predicted ball positions.

of early fusion, the output of the last convolutional layer was fed into a 512-D fully connected layer which in turn fed into the prediction layer. In case of late fusion, the output of the last convolutional layer was fed into a 512-D LSTM layer which in turn fed into a prediction layer. The performance of these networks and some baseline methods is reported in Table 8.3.

We consider two baselines - one which predicts that the ball at time $t+1, t+2$ will remain at the same location as at time t (i.e. Last position). This is a reasonable baseline because in many frames the player is in possession of the ball and he does not move. The second baseline estimates the ball velocity at time t and uses it to forecast the future location. We report mean and median errors in the distance and the angle of prediction. The distance between the ground truth and predicted location is reported as the percentage of the length of the basketball court. The angular error is the angle between the vector 1 pointing from current position to ground truth position in the future and vector 2 pointing from current to predicted position. We find that the proposed methods outperform the baseline and the early fusion method performs slightly better than the late fusion method. As expected, the prediction errors in distance are larger when predicting for 2s as compared to 1s. However, the errors in angle follow the reverse trend. One explanation is that in a shorter period, the ball moves by small distances and therefore angle measures are not robust.

Method	Dataset	Δ T	FT made	FT miss	FG made	FG miss	Off. Rebound	Def. Rebound	Turnover	Foul	Time Out	Dribble	Pass	Possession	Block	Assist	Drive	Screen	mAP
Avg. Human	H	1s	100.0	0	60.0	12.5	11.1	16.7	0	0	33.3	66.7	0	0	0	0	0	28.6	20.6
Random Forest	H	1s	100.0	0	20.0	0	20.0	40.0	0	0	0	100.0	20.0	40.0	0	0	0	0	21.3
Image CNN	A	1s	46.0	20.3	3.9	4.8	2.0	5.5	0.9	1.6	0	61.7	16.5	22.9	0.9	1.8	1.6	3.7	11.9
Overhead CNN	A	1s	62.2	22.7	38.6	16.4	9.4	43.9	1.8	5.2	3.5	76.1	25.5	37.6	0.8	3.4	1.6	13.1	22.6
Random Forest	A	1s	75.5	41.4	41.3	15.7	11.8	61.2	2.3	5.6	4.5	80.5	26.7	40.9	1.0	3.5	1.2	8.5	26.4
Avg. Human	H	2s	33.3	20.0	14.3	0	0	0	0	0	37.5	75.0	0	0	0	0	16.7	20.0	13.5
Random Forest	H	2s	100.0	0	0	0	20.0	40.0	0	0	0	100.0	0	20.0	0	0	0	0	17.5
Image CNN	A	2s	32.5	7.8	1.9	2.5	0.9	2.7	0.5	0.8	0.2	53.8	14.7	19.9	0	0.6	0.6	2.9	8.8
Overhead CNN	A	2s	39.8	19.0	7.3	6.9	3.8	12.9	1.5	2.2	1.6	71.0	18.3	25.3	0.4	2.7	1.1	5.8	13.7
Random Forest	A	2s	66.9	29.7	11.8	7.3	5.0	35.4	1.5	2.6	2.7	76.4	21.4	30.2	0.3	2.5	0.9	5.0	18.7
Avg. Human	H	40ms	28.6	28.6	83.3	0	50.0	0	0	0	0	25.0	57.1	14.3	0	0	20.0	83.3	24.4
Random Forest	H	40ms	100.0	0	40.0	80.0	40.0	100.0	0	20.0	0	100.0	60.0	100.0	0	0	0	80.0	45.0
Random Forest	A	40ms	68.8	24.5	69.5	54.7	62.7	85.2	6.1	31.8	16.7	93.2	76.2	92.6	3.3	8.1	5.0	57.7	47.3

Table 8.4: Prediction accuracy ΔT seconds in the future of 16 basketball events: free throw (FT) made and missed, field goal (FG) made and missed, offensive (off) and defensive (def) rebound, etc. Methods were evaluated on the full (A) *test* split of $90k$ events, as well as a smaller, 80-example subset (H) for human performance evaluation and comparison.

Transferring from Basketball to Water polo

Basketball and water polo are both team sports that require scoring baskets/goals. This suggests that there maybe general game play strategies, e.g., pass to the most open player, that are shared between these two games. If this is indeed the case then a model trained on one of these sports should perform reasonably well on forecasting events in the other sport. In order to test this hypothesis we trained a random forest model on the basketball data (the larger dataset) for predicting which player will get the ball next using the same features as described in 8.1.4 and then tested it on the water polo testing set.

The accuracy of this model on basketball itself was 69.9% and 36.8% on water polo. The performance on water polo is worse than a model trained directly on water polo (which achieves 45.5%) but same as the most open teammate baseline with 36.7% accuracy (Table 8.1). One explanation of these results is that differences in game strategies arise from the differences in game rules, number of players, and field size. Therefore the basketball model is outperformed by a model trained on water polo itself. However, the transfer performance is significantly better than chance performance and nearest teammate baseline, suggesting that our method is capable of learning game-independent ball passing strategies. A more detailed analysis of the error modes is provided in the supplementary materials.

8.1.6 Forecasting Events in Basketball

Predicting the ball location is just one out of many events of interest. For example, whether a teammate would screen or whether dribble or a break would take place

are of great interest in basketball. In a manner similar to predicting where the ball will be at times $\{t+1, t+2\}s$, we predict which out 16 events of interest will happen in the future.

We evaluate random forest and neural network based approaches for this task. The input to the random forest are the following hand designed features, extracted from the last visible frame: player and ball coordinates and velocities, distances between each player and the ball, angles between each player and the ball, the time remaining on the shot clock, the remaining game time in the period, and the time since the most recent event for each event occurring in the visible history. In total, we used 92 features. We tested two different neural networks - (a) Overhead CNN that took as inputs the image representation of the overhead view (see Section 8.1.5) along with the hand designed features described above and (b) Image CNN that took as input raw RGB images. The neural network architectures and training procedure are detailed in the supplementary materials.

Table 8.4 reports the performance of humans and various methods described above at predicting player moves 1s, 2s and 40ms in advance. The two test splits,"H" and "A" correspond to 80 examples on which human subjects were tested and a set 90K examples on which the algorithm was evaluated. The purpose of reporting the accuracy when predicting 40ms in advance is to obtain an upper bound on performance. The results reveal that random forest outperforms CNN based approaches and both these approaches perform better than an average human. The Overhead CNN outperforms the Image CNN suggesting that extracting features relevant for forecasting from raw visuals is a hard problem. It is also noteworthy that humans are significantly better at identifying Field Goals (i.e. FG made), but worse at identifying other events.

8.1.7 Conclusion

In this work we present predicting next players' moves in basketball and water polo as benchmark tasks for measuring performance of forecasting algorithms. Instead of forecasting activities of a single human, sports require forecasting in adversarial multi-agent environments that are a better reflection of the real world. As the events we predict are discrete, our benchmark allows for a crisp and meaningful evaluation metric that is critical for measuring progress. We compare the performance of two general systems for forecasting player moves: 1) a hand-engineered system that takes raw visuals as inputs, then transforms them into an overhead view for feature extraction, and 2) an end-to-end neural network system. We find the hand-engineered system is close to (non-expert) human performance in water polo and outperforms humans in basketball. In both cases it outperforms the neural network system,

which raises a very interesting question - what auxiliary tasks/unsupervised feature learning mechanisms can be used to improve prediction performance. We find that a system trained on basketball data generalizes to water polo data, showing that our techniques are capable of extracting generic game strategies.

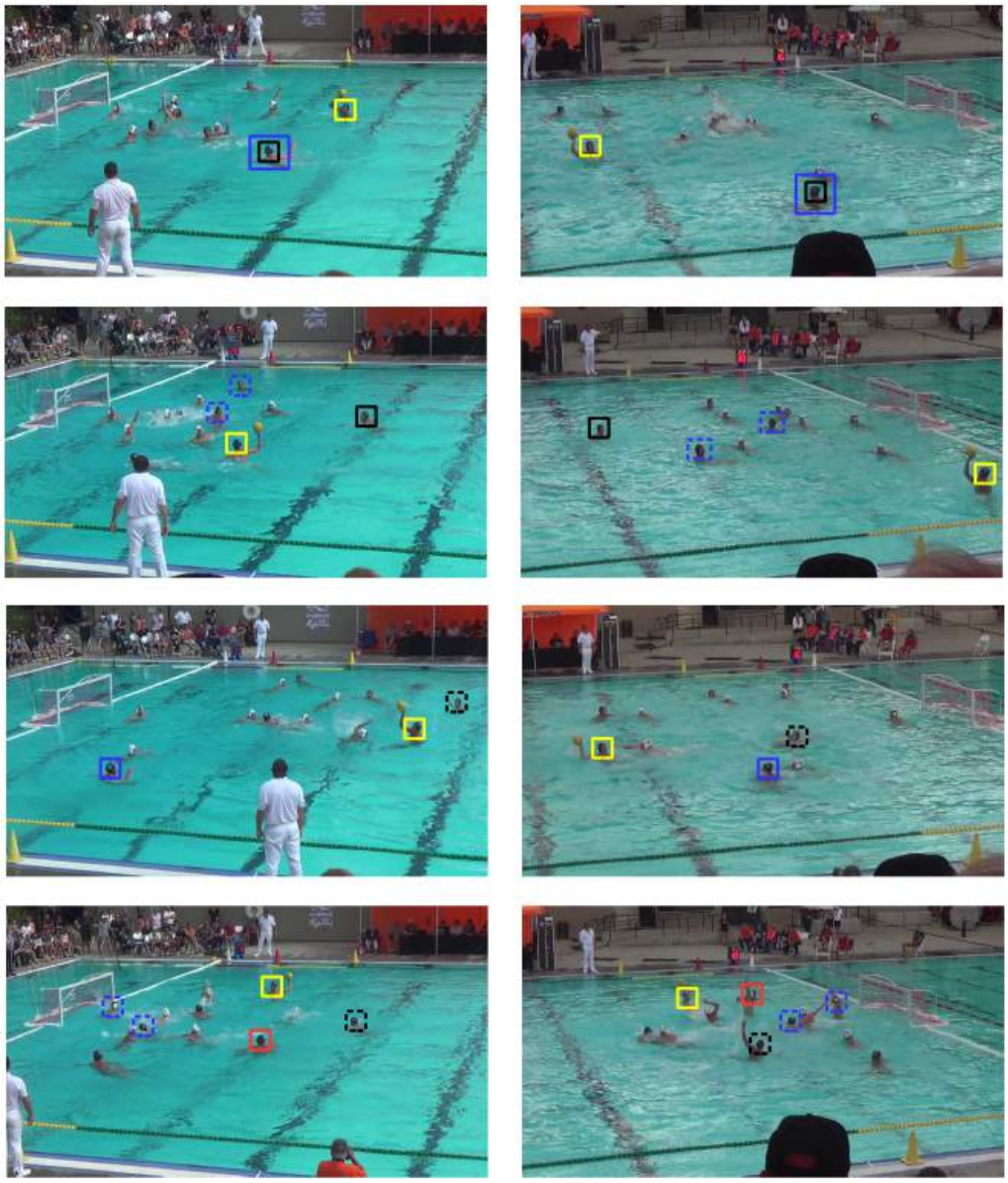


Figure 8.5: Sample predictions of our algorithm (black) and of water polo laymen (blue). The player in possession of the ball is marked in yellow, and in cases where both our algorithm and the humans made incorrect predictions, the player who actually received the ball is marked in red. A solid line indicates a correct prediction, whereas a dashed line indicates an incorrect prediction. Row 1 shows examples where both made the correct prediction. Row 2 shows examples where the algorithm is correct, but humans are incorrect. Row 3 shows examples where humans are correct, but our algorithm is incorrect. Finally, row 4 shows examples where both our algorithm and humans were incorrect.

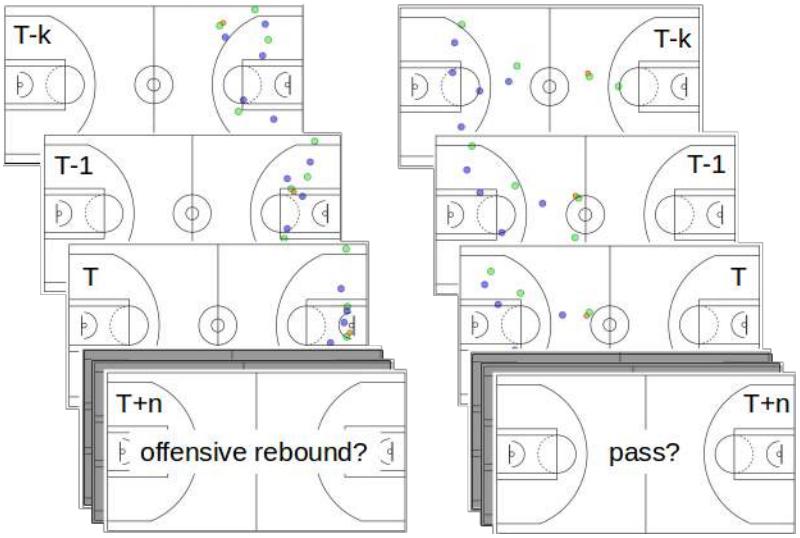


Figure 8.6: Examples of the basketball event prediction task: forecast an event n seconds in the future, provided a k-second history of the player and ball trajectories.

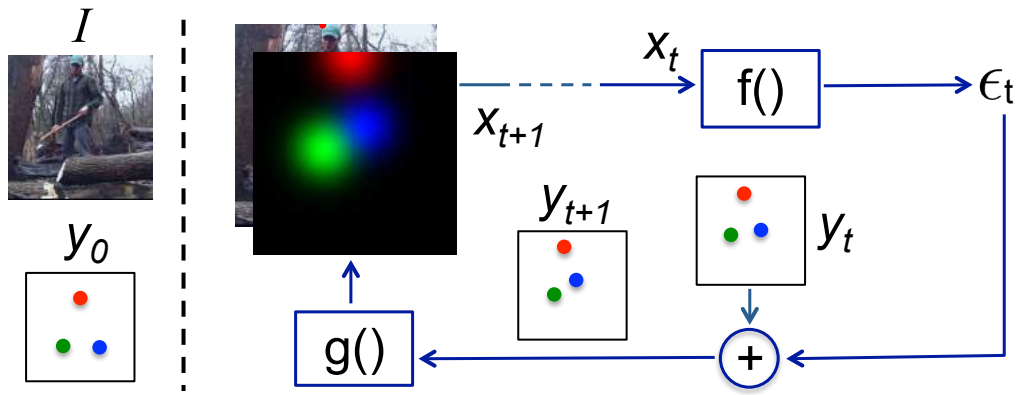


Figure 8.7: An implementation of Iterative Error Feedback (IEF) for 2D human pose estimation. The left panel shows the input image I and the initial guess of keypoints y_0, represented as a set of 2D points. For the sake of illustration we show only 3 out of 17 keypoints, corresponding to the right wrist (green), left wrist (blue) and top of head (red). Consider iteration t: predictor f receives the input x_t – image I stacked with a "rendering" of current keypoint positions y_t – and outputs a correction ϵ_t. This correction is added to y_t, resulting in new keypoint position estimates y_{t+1}. The new keypoints are rendered by function g and stacked with image I, resulting in x_{t+1}, and so on iteratively. Function f was modeled here as a ConvNet. Function g converts each 2D keypoint position into one Gaussian heatmap channel. For 3 keypoints there are 3 stacked heatmaps which are visualized as channels of a color image. In contrast to previous works, in our framework multi-layered hierarchical models such as ConvNets can learn rich models over the joint space of body configurations and images.

8.2 Human Pose Estimation

Feature extractors such as Convolutional Networks (ConvNets) [281] represent images using a multi-layered hierarchy of features and are inspired by the structure and functionality of the visual pathway of the human brain [282, 283]. Feature computation in these models is purely feedforward, however, unlike in the human visual system where feedback connections abound [284–286]. Feedback can be used to modulate and specialize feature extraction in early layers in order to model temporal and spatial context (e.g. *priming* [287]), to leverage prior knowledge about shape for segmentation and 3D perception, or simply for guiding visual attention to image regions relevant for the task under consideration.

Here we are interested in using feedback to build predictors that can naturally

handle complex, structured output spaces. We will use as running example the task of 2D human pose estimation [288–291], where the goal is to infer the 2D locations of a set of keypoints such as wrists, ankles, etc, from a single RGB image. The space of 2D human poses is highly structured because of body part proportions, left-right symmetries, interpenetration constraints, joint limits (e.g. elbows do not bend back) and physical connectivity (e.g. wrists are rigidly related to elbows), among others. Modeling this structure should make it easier to pinpoint the visible keypoints and make it possible to estimate the occluded ones.

Our main contribution is in providing a generic framework for modeling rich structure in both input and output spaces by learning hierarchical feature extractors over their joint space. We achieve this by incorporating top-down feedback – instead of trying to directly predict the target outputs, as in feedforward processing, we predict what is wrong with their current estimate and correct it iteratively. We call our framework Iterative Error Feedback, or IEF.

In IEF, a feedforward model f operates on the augmented input space created by concatenating (denoted by $\oplus$) the RGB image I with a visual representation g of the estimated output y_t to predict a "correction" (ϵ_t) that brings y_t closer to the ground truth output y. The correction signal ϵ_t is applied to the current output y_t to generate y_{t+1} and this is converted into a visual representation by g, that is stacked with the image to produce new inputs $x_{t+1} = I \oplus g(y_t)$ for f, and so on iteratively. This procedure is initialized with a guess of the output (y_0) and is repeated until a predetermined termination criterion is met. The model is trained to produce bounded corrections at each iteration, e.g. $||\epsilon_t||_2 < L$. The motivation for modifying y_t by a bounded amount is that the space of x_t is typically highly non-linear and hence local corrections should be easier to learn. The working of our model can be mathematically described by the following equations:

$$\epsilon_t = f(x_t) \tag{8.3}$$

$$y_{t+1} = y_t + \epsilon_t \tag{8.4}$$

$$x_{t+1} = I \oplus g(y_{t+1}), \tag{8.5}$$

where functions f and g have additional learned parameters Θ_f and Θ_g, respectively. Although we have used the predicted error to additively modify y_t in equation 8.4, in general y_{t+1} can be a result of an arbitrary non-linear function that operates on y_t, ϵ_t.

In the running example of human pose estimation, y_t is vector of retinotopic positions of all keypoints that are individually mapped by g into heatmaps (i.e. K heatmaps for K keypoints). The heatmaps are stacked together with the image and passed as input to f (see figure 8.7 for an overview). The "rendering" function g in

this particular case is not learnt – it is instead modelled as a 2D Gaussian having a fixed standard deviation and centered on the keypoint location. Intuitively, these heatmaps encode the current belief in keypoint locations in the image plane and thus form a natural representation for learning features over the joint space of body configurations and the RGB image.

The dimensionality of inputs to f is $H \times W \times (K + 3)$, where H, W represent the height and width of the image and $(K + 3)$ correspond to K keypoints and the 3 color channels of the image. We model f with a ConvNet with parameters Θ_f (i.e. ConvNet weights). As the ConvNet takes $I \oplus g(y_t)$ as inputs, it has the ability to learn features over the joint input-output space.

8.2.1 Learning

In order to infer the ground truth output (y), our method iteratively refines the current output (y_t). At each iteration, f predicts a correction (ϵ_t) that locally improves the current output. Note that we train the model to predict bounded corrections, but we do not enforce any such constraints at test time. The parameters (Θ_f, Θ_g) of functions f and g in our model, are learnt by optimizing equation 8.6,

$$\min_{\Theta_f, \Theta_g} \sum_{t=1}^{T} h(\epsilon_t, e(y, y_t)) \tag{8.6}$$

where, ϵ_t and $e(y, y_t)$ are predicted and target bounded corrections, respectively. The function h is a measure of distance, such as a quadratic loss. T is the number of correction steps taken by the model. T can either be chosen to be a constant or, more generally, be a function of ϵ_t (i.e. a termination condition).

We optimize this cost function using stochastic gradient descent (SGD) with every correction step being an independent training example. We grow the training set progressively: we start by learning with the samples corresponding to the first step for N epochs, then add the samples corresponding to the second step and train another N epochs, and so on, such that early steps get optimized longer – they get consolidated.

As we only assume that the ground truth output (y) is provided at training time, it is unclear what the intermediate targets (y_t) should be. The simplest strategy, which we employ, is to predefine y_t for every iteration using a set of fixed corrections $e(y, y_t)$ starting from y_0, obtaining $(y_0, y_1, ..y)$. We call our overall learning procedure *Fixed Path Consolidation (FPC)* which is formally described by algorithm 1.

The target bounded corrections for every iteration are computed using a function $e(y, y_t)$, which can take different forms for different problems. If for instance the

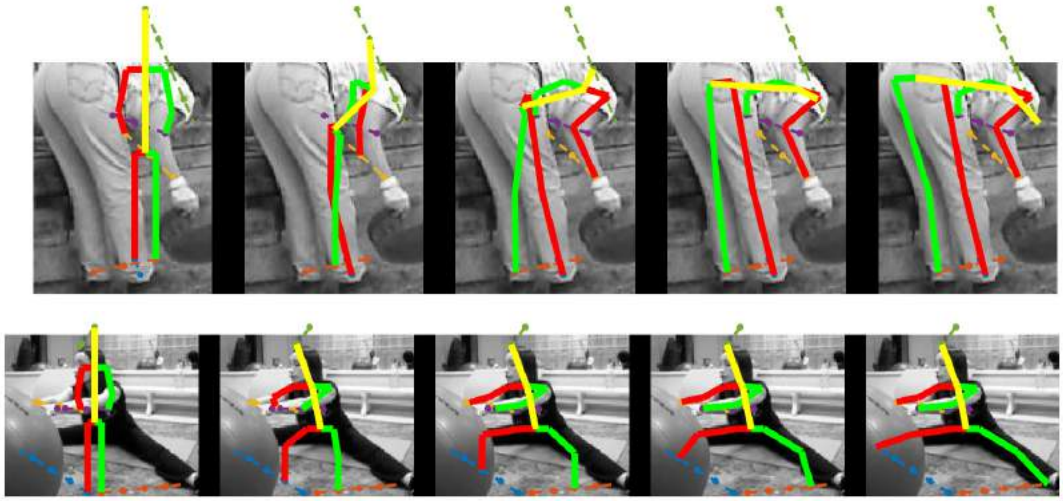


Figure 8.8: In our human pose estimation running example, the sequence of corrections ϵ_t moves keypoints along lines in the image, starting from an initial mean pose y_0 (left), all the way to the ground truth pose y (right), here shown for two different images. This simplifies prediction at test time, because the desired corrections to each keypoint are constant for each image, up to the last one which is a scaled version. Feedback allows the model to detect when the solution is close and to reduce "keypoint motion", as in a control system. Linear trajectories are shown for only a subset of the keypoints, to limit clutter.

output is 1D, then $e(y, y_t) = max(\text{sign}(y - y_t) \cdot \alpha, y - y_t)$ would imply that the target "bounded" error will correct y_t by a maximum amount of α in the direction of y.

Learning Human Pose Estimation

Human pose was represented by a set of 2D keypoint locations $y : \{y^k \in \Re^2, k \in [1, K]\}$ where K is the number of keypoints and y^k denotes the k^{th} keypoint. The predicted location of keypoints at the t^{th} iteration has been denoted by $y_t : \{y_t^k, k \in [1, K]\}$. The rendering of y_t as heatmaps concatenated with the image was provided as inputs to a ConvNet. The ConvNet was trained to predict a sequence of "bounded" corrections for each keypoint (ϵ_t^k) . The corrections were used to iteratively refine the keypoint locations.

Let $u = y^k - y_t^k$ and the corresponding unit vector be $\hat{u} = \frac{u}{||u||_2}$. Then, the target "bounded" correction for the t^{th} iteration and k^{th} keypoint was calculated as:

$$e(y^k, y_t^k) = \min(L, ||u||) \cdot \hat{u} \tag{8.7}$$

Algorithm 1 Learning Iterative Error Feedback with Fixed Path Consolidation

procedure FPC-LEARN
  Initialize y_0
  $E \leftarrow \{\}$
  for $t \leftarrow 1$ to (T_{steps}) **do**
    for all training examples (I, y) **do**
      $\epsilon_t \leftarrow e(y, y_t)$
    end for
    $E \leftarrow E \cup \epsilon_t$
    for $j \leftarrow 1$ to N **do**
      Update Θ_f and Θ_g with SGD, using loss h and target corrections E
    end for
  end for
end procedure

where L denotes the maximum displacement for each keypoint location. An interesting property of this function is that it is constant while a keypoint is far from the ground truth and varies only in scale when it is closer than L to the ground truth. This simplifies the learning problem: given an image and a fixed initial pose, the model just needs to predict a constant direction in which to move keypoints, and to "slow down" motion in this direction when the keypoint becomes close to the ground truth. See fig. 8.8 for an illustration.

The target corrections were calculated independently for each keypoint in each example and we used an L_2 regression loss to model h in eq. 8.6. We set L to 20 pixels in our experiments. We initialized y_0 as the median of ground truth 2D keypoint locations on training images and trained a model for $T = 4$ steps, using $N = 3$ epochs for each new step. We found the fourth step to have little effect on accuracy and used 3 steps in practice at test time.

ConvNet architecture. We employed a standard ConvNet architecture pre-trained on Imagenet: the very deep googlenet [16] [1]. We modified the filters in the first convolution layer (conv-1) to account for 17 additional channels due to 17 keypoints. In our model, the conv-1 filters operated on 20 channel inputs. The weights of the first three conv-1 channels (i.e. the ones corresponding to the image) were initialized using the weights learnt by pre-training on Imagenet. The weights corresponding to the remaining 17 channels were randomly initialized with Gaussian noise of variance 0.1. We discarded the last layer of 1000 units that predicted the Imagenet classes and

[1] The VGG-16 network [17] produced similar results, but required significantly more memory.

replaced it with a layer containing 32 units, encoding the continuous $2D$ correction [2] expressed in Cartesian coordinates (the 17th "keypoint" is the location of one point anywhere inside a person, marking her, and which is provided as input both during training and testing, see section 8.2.2). We used a fixed ConvNet input size of 224×224.

8.2.2 Results

We tested our method on the two most challenging benchmarks for 2D human pose estimation: the MPII Human Pose dataset [292], which features significant scale variation, occlusion, and multiple people interacting, and Leeds Sports Pose dataset (LSP) [293] which features complex poses of people in sports. For each person in every image, the goal is to predict the 2D locations of all its annotated keypoints.

MPII – Experimental Details. Human pose is represented as a set of 16 keypoints. An additional *marking-point* in each person is available both for training and testing, located somewhere inside each person's boundary. We represent this point as an additional channel and stack it with the other 16 keypoint channels and the 3 RGB channels that we feed as input to a ConvNet. We used the same publicly available train/validation splits of [290]. We evaluated the accuracy of our algorithm on the validation set using the standard PCKh metric [292], and also submitted results for evaluation on the test set once, to obtain the final score.

We cropped 9 square boxes centered on the marking-point of each person, sampled uniformly over scale, from $1.4\times$ to $0.3\times$ of the smallest side of the image and resized them to 256×256 pixels. Padding was added as necessary for obtaining these dimensions and the amount of training data was further doubled by also mirroring the images. We used the ground truth height of each person at training time, which is provided on MPII, and select as training examples the 3 boxes for each person having a side closest to $1.2\times$ the person height in pixels. We then trained googlenet models on random crops of 224×224 patches, using 6 epochs of consolidation for each of 4 steps. At test time, we predict which one of the 9 boxes is closest to $1.2\times$ the height of the person in pixels, using a shallower model, the VGG-S ConvNet [294], trained for that task using an L_2 regression loss. We then align our model to the center 224×224 patch of the selected window. The MatConvnet library [295] was employed for these experiments.

We train our models using keypoint positions for both visible and occluded keypoints, which MPII provides in many cases whenever they project on to the

[2] Again, we do not bound explicitly the correction at test time, instead the network is taught to predict bounded corrections.

image (the exception are people truncated by the image border). We zero out the backpropagated gradients for missing keypoint annotations. Note that often keypoints lie outside the cropped image passed to the ConvNet, but this poses no issues to our formulation – keypoints outside the image can be predicted and are still visible to the ConvNet as tails of rendered Gaussians.

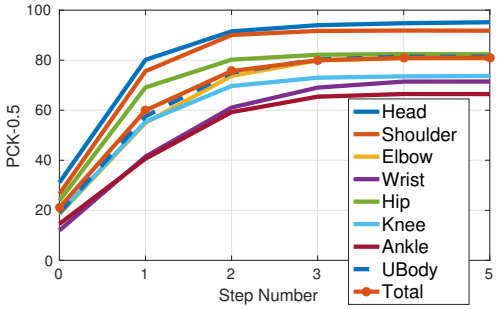


Figure 8.9: Evolution of PCKh at 0.5 overlap as function of correction step number on the MPII-human-pose validation set, using the finetuned googlenet network. The model aligns more accurately to parts like the head and shoulders, which is natural, because these parts are easier to discriminate from the background and have more consistent appearance than limbs.

Comparison with State-of-the-Art. The standard evaluation procedure in the MPII benchmark assumes ground truth scale information is known and images are normalized using this scale information. The current state-of-the-art is the sliding-window approach of Tompson et al [290] and IEF roughly matches this performance, as shown in table 8.5. In the more realistic setting of unknown scale information, the best previous result so far is from Tompson et al. [290] which was the first work to experiment with this setting and obtained 66.0 PCKh. IEF significantly improves upon this number to 81.3. Note however that the emphasis in Tompson et al's system was efficiency and they trained and tested their model using original image scales – searching over a multiscale image pyramid or using our automatic rescaling procedure should presumably improve their performance. See the MPII website for more detailed results.

LSP – Experimental Details. In LSP, differently from MPII, images are usually tight around the person whose pose is being estimated, are resized so people have a fixed size, and have lower resolution. There is also no marking point on the torsos so we initialized the 17th keypoints used in MPII to the center of the image. The same set of keypoints is evaluated as in MPII and we trained a model using the same hyper-parameters on the extended LSP training set. We use the standard

	Head	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	UBody	FBody
Yang & Ramanan [288]	73.2	56.2	41.3	32.1	36.2	33.2	34.5	43.2	44.5
Pischulin et al [296]	74.2	49.0	40.8	34.1	36.5	34.4	35.1	41.3	44.0
Tompson et al. [290]	96.1	91.9	83.9	77.8	80.9	72.3	64.8	84.5	82.0
IEF	95.7	91.6	81.5	72.4	82.7	73.1	66.4	82.0	81.3
Tompson et al. [290]	83.4	77.5	67.5	59.8	64.6	55.6	46.1	68.3	66.0
IEF	95.5	91.6	81.5	72.4	82.7	73.1	66.9	81.9	81.3

Table 8.5: MPII test set PCKh-0.5 results for Iterative Error Feedback (IEF) and previous approaches, when ground truth scale information at test time is provided (top) and in the more automatic setting when it is not available (bottom). UBody and FBody stand for upper body and full body, respectively.

	Torso	Upper Leg	Lower Leg	Upper Arm	Forearm	Head	Total
Pishchulin et al. [298]	88.9	64.0	58.1	45.5	35.1	85.1	58.0
Tompson et al. [299]	90.3	70.4	61.1	63.0	51.2	83.7	66.6
Fan et al. [300]	95.4	77.7	69.8	62.8	49.1	86.6	70.1
Chen and Yuille [297]	96.0	77.2	72.2	69.7	58.1	85.6	73.6
IEF	95.3	81.8	73.3	66.7	51.0	84.4	72.5

Table 8.6: Person-centric PCP scores on the LSP dataset test set for IEF and previous approaches.

LSP evaluation code supplied with the MPII dataset and report person-centric PCP scores in table 8.6. Our results are competitive with the current state-of-the-art of Chen and Yuille [297].

8.2.3 Analyzing IEF

In this section, we perform extensive ablation studies to validate four choices of the IEF model: 1) proceeding iteratively instead of in a single shot, 2) predicting bounded corrections instead of directly predicting the target outputs, 3) curriculum learning of our bounded corrections, and 4) modeling the structure in the full output space (all body joints in this case) over carrying out independent predictions for each label.

Iterative v/s Direct Prediction. For evaluating the importance of progressing towards solutions iteratively we trained models to directly predict corrections to the keypoint locations in a single shot (i.e. direct prediction). Table 8.7 shows that IEF that additively regresses to keypoint locations achieves PCKh-0.5 of 81.0 as

	Head	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	UBody	FBody
Iterative Error Feedback (IEF)	95.2	91.8	80.8	71.5	82.3	73.7	66.4	81.4	81.0
Direct Prediction	92.9	89.4	74.1	61.7	79.3	64.0	53.3	75.1	74.8
Iterative Direct Prediction	91.9	88.5	73.3	59.9	77.5	61.2	51.8	74.0	73.4

Table 8.7: PCKh-0.5 results on the MPII validation set for models finetuned from googlenet using Iterative Error Feedback (IEF), direct regression to the keypoint locations (direct prediction), and a model that was trained to iteratively predict human pose by regressing to the ground truth keypoint locations (instead of bounded corrections) in each iteration, starting from the pose in the previous iteration. The results show that our proposed approach results in significantly better performance.

compared to PCKh of 74.8 achieved by directly regressing to the keypoints.

Iterative Error Feedback v/s Iterative Direct Prediction. Is iterative prediction of the error important or iterative prediction of the target label directly (as in e.g., [301, 302]) performs comparably? In order to answer this question we trained a model from the pretrained googlenet to iteratively predict the ground truth keypoint locations (as opposed to predicting bounded corrections). For comparing performance, we used the same number of iterations for this baseline model and IEF. Table 8.7 shows that IEF achieves PCKh-0.5 of 81.0 as compared to PCKh of 73.4 by iterative direct prediction. This can be understood by the fact that the learning problem in IEF is much easier. In IEF, for a given image, the model is trained to predict constant corrections except for the last one which is a scaled version. In iterative direct prediction, because each new pose estimate ends up somewhere around the ground truth, the model must learn to adjust directions and magnitudes in all correction steps.

Importance of Fixed Path Consolidation (FPC). The FPC method (see algorithm 1) for training a IEF model makes N corrections is a curriculum learning strategy where in the $i^{th}(i \leq N)$ training stage the model is optimized for performing only the first i corrections. Is this curriculum learning strategy necessary or can all the corrections be simultaneously trained? For addressing this question we trained an alternative model that trains for all corrections in all epochs. We trained IEF with and without FPC for the same number of SGD iterations and the performance of both these models is illustrated in figure 8.10. The figure shows that without FPC, the performance drops by almost 10 PCKh points on the validation set and that there is significant drift when performing several correction steps.

Learning Structured Outputs. One of the major merits of IEF is supposedly that it can jointly learn the structure in input images and target outputs. For human

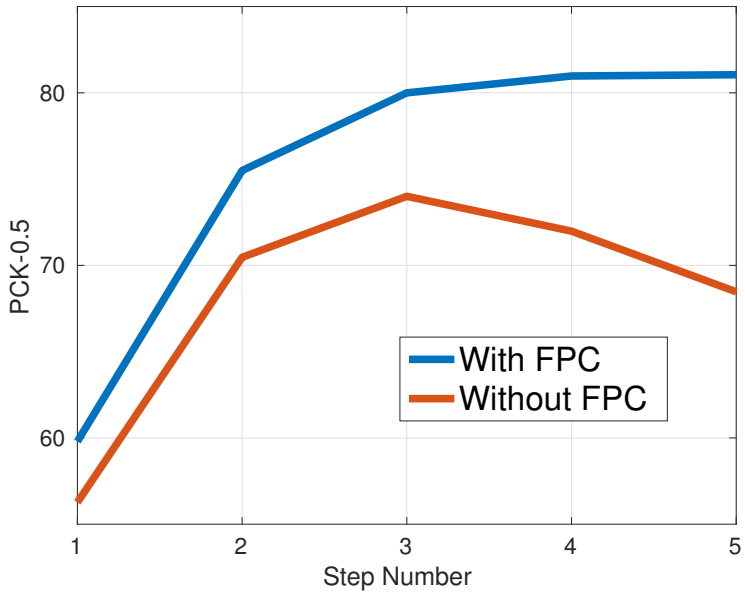


Figure 8.10: Validation PCKh-0.5 scores for different number of correction steps taken, when finetuning a IEF model from a googlenet base model using stochastic gradient descent with either Fixed Path Consolidation (*With FPC*), or directly over all training examples (*Without FPC*), for the same amount of time. FPC leads to significantly more accurate results, leading to models that can perform more correction steps without drifting. It achieves this by consolidating the learning of earlier steps and progressively increasing the difficulty of the training set by adding additional correction steps.

pose estimation, IEF models the space of outputs by augmenting the image with additional input channels having gaussian renderings centered around estimated keypoint locations . If it is the case that IEF learns priors over the appropriate relative locations of the various keypoints, then depriving the model of keypoints other than the one being predicted should decrease performance.

In order to evaluate this hypothesis we trained three different IEF models and tested how well each predicted the location of the “Left Knee" keypoint. The first model had only one input channel corresponding to the left knee, the second model had two channels corresponding to left knee and the left hip. The third model was trained using all keypoints in the standard IEF way. The performance of these three models is reported in table 8.8. As a baseline, regression gets 64.6, whereas the IEF model with a single additional input channel for the left knee gets PCKh of 69.2

	Direct Prediction of All Joints	IEF Left Knee	IEF Left Knee + Left Hip	IEF All Joints
Left Knee PCKh-0.5	64.6	69.2	72.8	73.8

Table 8.8: MPII validation PCKh-0.5 results for left knee localization when using IEF and both training and predicting different subsets of joints. We also show the result obtained using a direct prediction variant similar to plain regression on all joints (having the mean pose Gaussian maps in the input). Modeling global body structure jointly with the image leads to best results by "IEF All Joints". Interestingly, feedback seems to add value by itself and IEF on the left knee, in isolation, significantly outperforms the direct prediction baseline.

This shows that feeding back the current estimate of the left knee keypoint allows for more accurate localization by itself. Furthermore, the IEF model over both left knee and left hip gets PCKh of 72.8. This suggests that the relationship between neighboring outputs has much of the information, but modeling all joints together with the image still wins, obtaining a PCKh of 73.8.

8.2.4 Related Work

There is a rich literature on structured output learning [303, 304] (e.g. see references in [305]) but it is a relatively modern topic in conjunction with feature learning, for computer vision [290, 306, 307].

Here we proposed a feedback-based framework for structured-output learning. Neuroscience models of the human brain suggest that feedforward connections act as information carriers while numerous feedback connections act as modulators or competitive inhibitors to aid feature grouping [308], figure-ground segregation [309] and object recognition [310]. In computer vision, feedback has been primarily used so far for learning selective attention [311]; in [311] attention is implemented by estimating a bounding box in an image for the algorithm to process next, while in [312] attention is formed by selecting some convolutional features over others (it does not have a spatial dimension).

Stacked inference methods [301,302,313,314] are another related family of methods. Differently, some of these methods consider each output in isolation [289], all use different weights or learning models in each stage of inference [290] or they do not optimize for correcting their current estimates but rather attempt to predict the answer from scratch at each stage [302, 315]. In concurrent work, Oberweger et al [316] proposed a feedback loop for hand pose estimation from kinect data that is closely related to our approach. The autocontext work of [302] is also related and iteratively computes label heatmaps by concatenating the image with the heatmaps

previously predicted. IEF is inspired by this work and we show how this iterative computation can be carried out effectively with deep Convnet architectures, and with bounded error corrections, rather than aiming for the answer from scratch at each iteration.

Another line of work aims to inject class-specific spatial priors using coarse-to-fine processing, e.g. features arising from different layers of ConvNets were recently used for instance segmentation and keypoint prediction [317]. For pose inference, combining multiple scales [290, 318] aids in capturing subtle long-range dependencies (e.g. distinguishing the left and right sides of the body which depend on whether a person is facing the camera). The system in our human pose estimation example can be seen as closest to approaches employing "pose-indexed features" [319–321], but leveraging hierarchical feature learning. Graphical models can also encode dependencies between outputs and are still popular in many applications, including human pose estimation [297]. Classic spatial alignment and warping computer vision models, such as snakes, [322] and Active Appearance Models (AAMs) [323] have similar goals as the proposed IEF, but are not learned end-to-end – or learned at all – employ linear shape models and hand designed features and require slower gradient computation which often takes many iterations before convergence. They can get stuck in poor local minimas even for constrained variation (AAMs and small out-of-plane face rotations). IEF, on the other hand, is able to minimize over rich articulated human 3D pose variation, starting from a mean shape. Although extensions that use learning to drive the optimization have been proposed [324], typically these methods still require manually defined energy functions to measure goodness of fit.

8.2.5 Discussion

While standard ConvNets offer hierarchical representations that can capture the patterns of images at multiple levels of abstraction, the outputs are typically modeled as flat image or pixel-level 1-of-K labels, or slightly more complicated hand-designed representations. We aimed in this paper to mitigate this asymmetry by introducing Iterative Error Feedback (IEF), which extends hierarchical representation learning to output spaces, while leveraging at heart the same machinery. IEF works by, in broad terms, moving the emphasis from the problem of *predicting* the state of the external world to one of *correcting* the expectations about it, which is achieved by introducing a simple feedback connection in standard models.

In our pose estimation working example we opted for feeding pose information only into the first layer of the ConvNet for the sake of simplicity. This information may also be helpful for mid-level layers, so as to modulate not only edge detection, but also

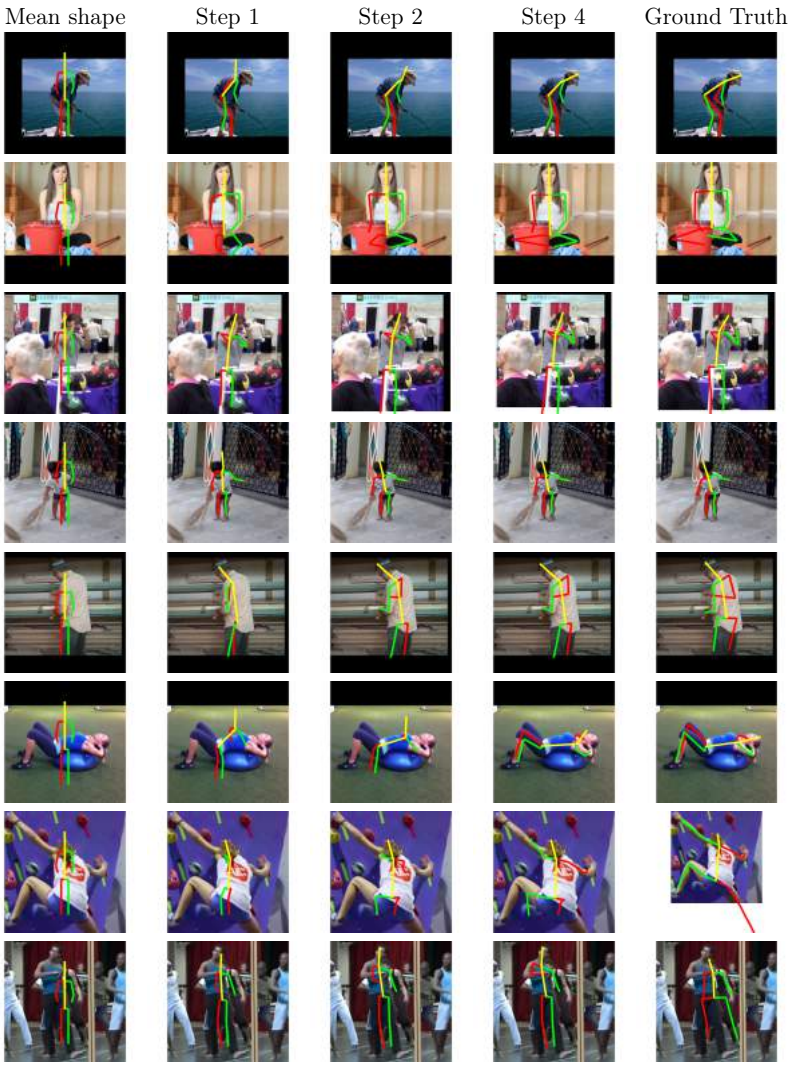


Figure 8.11: Example poses obtained using the proposed method IEF on the MPII validation set. From left to right we show the sequence of corrections the method makes – on the right is the ground truth pose, including annotated occluded keypoints, which are not evaluated. Note that IEF is robust to left-right ambiguities and is able to rotate the initial pose by up to 180^{o} (first and fifth row), can align across occlusions (second and third rows) and can handle scale variation (second, fourth and fifth rows) and truncation (fifth row). The bottom two rows show failure cases. In the first one, the predicted configuration captures the gist of the pose but is misaligned and not scaled properly. The second case shows several people closely interacting and the model aligns to the wrong person. The black borders show padding. Best seen in color and with zoom.

processes such as junction detection or contour completion which advanced feature extractors may need to compute. We also have only experimented so far feeding back "images" made up of Gaussian distributions. There may be more powerful ways to render top-down pose information using parametrized computational blocks (e.g. deconvolution) that can then be learned jointly with the rest of the model parameters using standard backpropagation. This is desirable in order to attack problems with higher-dimensional output spaces such as 3D human pose estimation [325, 326] or segmentation.

Chapter 9

Can Deep Learning Inform Neuroscience?

As discussed earlier in this thesis, in the recent years there have been many systems making use of deep learning that have exceeded or matched human performance at the game of GO [2], ATARI video games [3] and classifying images into one of the thousand imagenet categories [14]. A question of great interest is whether these super-human machine learning systems can inform how humans solve the same tasks. In this chapter, I will present a study comparing the similarities between a deep learning system for image classification and the human visual system [283].

9.1 What we know about the human visual system

Neuroscientists, psychologists and computer vision scientists have been fascinated by how does the human brain transform visual information captured by the retina into information useful for semantic tasks like object recognition and scene interpretation? We know that this transformation is performed by a hierarchically organized system of visual areas within the visual cortex. However, we do not know how the brain computes this transformation. Past studies have found that visual regions of interests (ROIs [327]) like V1, V2 located in posterior visual cortex appear to represent low-level visual features such as oriented edges [328], gabors [329] and local motion-energy features [330]. Visual ROIs like Fusiform Face Area (FFA [331, 332]), Extrastriate Body Area (EBA [333]) and Parahippocampal Place Area (PPA [334]) located in anterior visual cortex appear to represent high-level semantically meaningful features useful for detecting faces, bodies and understanding visual scenes. Further, it is believed that visual ROIs like V4 located in intermediate visual cortex represent

mid-level visual features that are useful grouping, figure-ground segmentation and representing contours [335]. However, the visual representations and computations performed in intermediate visual ROIs is poorly understood.

While we do understand to some extent the features represented by different parts of the visual cortex, to date there is no cogent theory that explains how the brain transforms retinal stimuli into high-level semantic information. The prominent computational theory in neuroscience put forth by Barlow [336] argues that the goal of the visual system is to reduce redundancy in information (i.e. compression). In his view, the representations in the visual cortex arise out of compression. Computational models based on this idea such as *independent component analysis* [39] and *sparse coding* [38] accurately predict the computations performed in the area V1 of the visual cortex. However, models based on the redundancy reduction hypothesis have not been able to explain visual representations in mid and other higher visual areas such as V4, PPA, FFA etc.

9.2 Framework for testing models of visual computation in the human brain

One way to address the question of understanding how the brain transforms low-level visual representations into high-level visual representation is to build a computational model that takes images/videos as inputs and outputs an accurate prediction of brain activity across the visual cortex. An accurate prediction of brain activity would imply that the constructed model and the brain represent visual information using similar features. The similarity of features by itself would be insufficient to make conclusions about the exact computations performed by the brain. However, such a finding would suggest that the constructed model is a plausible computational hypothesis for how the brain transforms low-level features into high-level features. Further, such a model could also be used to investigate the nature of visual representations in different parts of the visual cortex.

Given that brain is a complex non-linear processing system, it is unlikely that an analytical solution to the problem of constructing such a model would exist. Past studies have addressed this concern by breaking down the process of predicting brain activity elicited in response to stimulus images into two steps. In the first step, a feature space that provides a linearizing transformation between the stimulus images and measured brain activity is constructed. In the second step, regularized linear regression is used to find a set of weights that predict brain activity from the feature representation of images. This framework for predicting brain activity has been called the encoding model approach [329,337–341]. Past studies used manually constructed

feature spaces for predicting brain activity. For instance, [329, 330] predicted brain activity in visual ROIs like V1, V2 using Gabor features. [339, 340] used linguistically constructed feature spaces that indicated the presence or absence of multiple object categories in images. These studies were only only able to predict brain activity in anterior visual cortex (i.e. ROIs like FFA, EBA, PPA). Moreover, these studies were unsatisfying because they did not provide any explanation for how the brain converts stimulus images into semantically meaningful information. To date, there exists no model that can predict brain activity throughout the visual cortex starting from image pixels.

Instead of manually defining features, an alternative is to use machine learning techniques to learn features that are optimal for predicting brain activity. However, it is unlikely that these techniques would work because non-linear machine learning methods require large amounts of training data and brain activity recordings are not available in plenty. This is because, collecting brain activity data is both a tedious and a costly process. Another way to learn features is by training models for performing the same tasks that the human visual system performs. After all, it is reasonable to assume that visual processing and representations in the brain are optimized for the visual tasks it must perform. Moreover, large amounts of data are publically available for training models for performing tasks like object recognition that are also performed by humans [25].

Recently in the field of computer vision, a class of computational models called as Convolutional Neural Networks (ConvNets [281]) have been found to be very successful on the task of object recognition [342]. ConvNet were inspired by the organization of eurons in the visual cortex [328, 343] and multiple considerations suggest that the visual features of a ConvNet are a good candidate for studying visual features represented by the brain. Firstly, the brain and the ConvNet are both adept at the common task of object recognition. Secondly, the brain and the ConvNet both represent visual information hierarchically. For instance, the ConvNet architecture proposed by [342] represented images by a seven-layered hierarchy of visual features. Lastly, some past studies have shown that the lower layers of the ConvNet feature hierarchy represent visual features such as edges and corners whereas the higher layers represent visual features that are more useful for object recognition [15, 58]. These three facts taken together suggest that low and high-level visual features represented by the brain and the ConvNet are likely to be similar. If it is the case that low and high-level features represented by the brain and the ConvNet are similar, it is likely that mid-level features represented by the brain and the ConvNet are also similar.

In this chapter I will detail the experiment performed to test the above hypothesis. We investigated the relationship between the hierarchies of visual representations in the human brain and a ConvNet trained for the task of object recognition. The

method and results of our investigation are presented in section 9.3 and section 9.4 respectively. A discussion of the implication of the results and a comparison with related previous work in provided in section 9.5.

9.3 Method

For studying the relationship between the visual representations of the ConvNet and the human brain we constructed computational models for predicting brain activity from visual representations of the ConvNet (see figure 9.1). First, we trained a seven layered ConvNet with the architecture proposed by [342] for the task of classifying 1.2M million natural images into 1000 distinct object categories (ILSVRC-2012 challenge [25]) using the publically available software [109]. In the remainder of this paper, the term ConvNet refers to this particular ConvNet. This ConvNet transformed input images into seven set of features (one from each of the seven layers). These features were used to predict brain activity.

The brain activity data for this study were functional magnetic resonance imaging (fMRI [344]) recordings of human brain activity (specifically, the blood-oxygenation-level-dependent (BOLD) signal), recorded continuously while four subjects passively viewed a series of static photos of color natural scenes. These subjects have been referred to as S1, S2, S3 and S4 in the remainder of the paper. We used the fMRI data that was previously used by [339]. This study measured brain activity elicited by 1260 images shown twice each (train set of images), and another set of 126 images shown 12 times each (test set of images). Activity was measured in approximately 100,000 voxels (i.e., volumetric pixels) located in the cerebral cortex of each subject. We followed the same procedure for pre-processing the fMRI data as outlined in [339].

9.3.1 Constructing Models for Predicting Brain Activity

For every voxel, a separate model was constructed for predicting its BOLD activity from the given feature representation of the image. Ridge regression was used to find a set of weights that predicted voxel's BOLD activity using the training set of 1260 images. A single regularization parameter was chosen for all voxels, using five-fold cross-validation [340]. The accuracy of each model for each voxel was expressed as the correlation (r) between predicted and recorded voxel activity in response to images in the test set. The explained variance in each voxel's responses was calculated as the square of correlation coefficient (r2) [337]. Prediction accuracy was deemed statistically significant if the correlation had a p-value < 0.001 (see supplementary materials for more details).

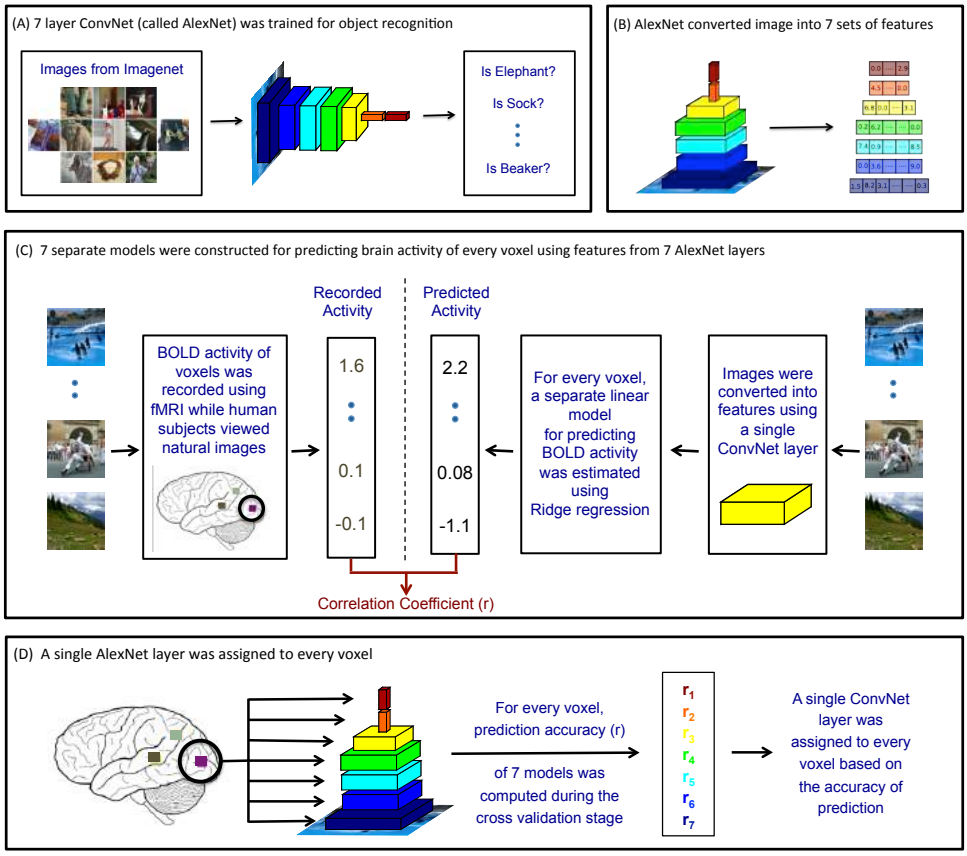


Figure 9.1: Description of the method for predicting brain activity using ConvNet features. First, a seven layered ConvNet was trained for the task of recognizing 1000 distinct object categories using a collection of 1.2M labelled images (panel A) [25,342]. This ConvNet extracted seven sets of features (from seven layers) for a given input image (panel B). ConvNet features were used to predict brain activity (i.e. BOLD activity measured using fMRI) of four human subjects while they passively viewed a separate collection of natural images (panel C). For every voxel, seven separate models for predicting BOLD activity were constructed using features extracted from seven layers of the ConvNet. Based on the accuracy of prediction (measured as correlation coefficient), an optimal ConvNet layer was assigned to every voxel (panel D).

The modelling framework described above implicitly assumes that voxel responses are stationary (i.e. the BOLD activity of a voxel is only a function of the input image and will be the same every time the same image is presented as stimulus).

However, the voxel responses can be non-stationary due to either the inherent non-stationarity in firing of individual neurons that constitute the voxel or due to noise in fMRI measurements. As our modelling framework is incapable of dealing with non-stationarity, we only fit models to voxels that are approximately stationary. The stationarity of a voxel can be estimated by calculating the repeatability in BOLD activity of the voxel expressed as the Signal to Noise Ratio (SNR). The method for computing the SNR is detailed in the supplementary materials. In this work, SNR has been expressed in terms of p-values (p_{SNR}). Note that this p_{SNR} is different measure than the p-value of the prediction accuracy. In this work, we have only considered voxels with $p_{SNR} < 0.001$.

Convolutional Neural Network (ConvNet) Model

After the ConvNet was trained for object recognition, it was used to transform all images used in the fMRI study into seven sets of features. The images used in the fMRI study were seperate from images used for training the ConvNet. Each set of feature corresponded to the feature representation of images produced by a single layer of the ConvNet. The first five layers of the ConvNet performed convolutions (denoted conv-1 through conv-5) and the last two layers were fully connected (fc-6, fc-7) (see supplementary materials for more details). For every voxel, seven separate sets of weights were estimated for predicting brain activity from these seven feature spaces. An optimal ConvNet layer was determined for every voxel based on the prediction accuracy of voxel activity measured during the cross-validation stage (see figure 9.1).

Baseline Model

In order to compare the prediction accuracy of the ConvNet with previously published models, a baseline model was constructed by combining the Gabor Wavelet model (GW; [329]) and the 19-Category model (19-Cat; [338, 339]). The GW and 19-Cat model have been shown to accurately predict brain activity in early (V1, V2) and late (PPA, FFA, EBA, OPA) visual areas respectively. More details on these two models has been provided in the supplementary materials. The Baseline model was constructed in the following way: For every voxel two sets of weights were independently estimated for predicting BOLD activity from GW and 19-Cat features. Each voxel was then assigned either to the GW or the 19-Cat model. This assignment was made based on the accuracy of the GW and the 19Cat models in predicting BOLD activity of the voxel measured during the cross validation stage. The model obtained after this assignment has been called the Baseline model.

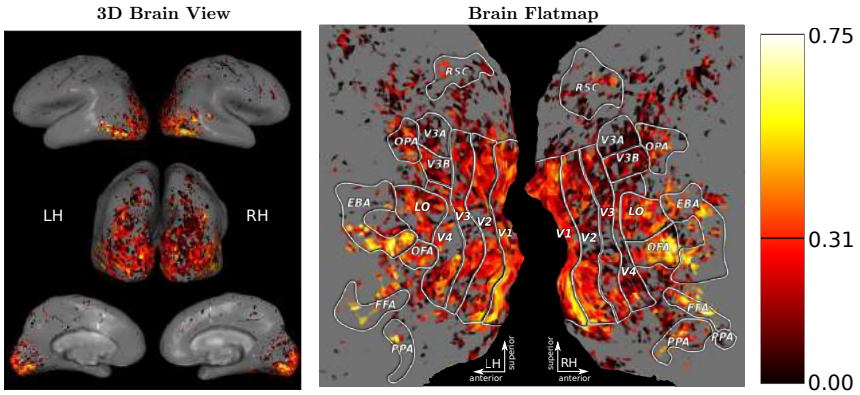


Figure 9.2: Accuracy of the ConvNet model in predicting brain activity of subject S1. The accuracy was measured as the correlation coefficient (Pearson's r) between the predicted and recorded brain activity in response to the test set of images. The color of each voxel reflects the prediction accuracy of the ConvNet model. Hotter colors reflect higher accuracy of prediction. The statistical significance of each voxel was evaluated individually and the mean cutoff value of r for the voxels with p-value < 0.001 was found to be 0.306 ± 0.008. The voxels with low SNR (i.e. p_{SNR}-value > 0.001) have been shown in gray.

9.4 Results

9.4.1 ConvNet predicts brain activity across the visual cortex

Any hypothesized feature space provides useful insights into brain representations only to the extent that it accurately predicts brain activity elicited under naturalistic conditions. The prediction accuracy of the ConvNet model was evaluated using the test set of 126 images and evoked BOLD activity (that were separate from set of images used for fitting the model). Figure 9.2 shows prediction accuracy of the ConvNet model fit to voxels distributed across visual cortex for subject S1. From this figure it can be concluded that the ConvNet model transforms image pixels into features that make significant predictions (i.e. p-value < 0.001) of BOLD activity across the visual cortex.

If it is the case that ConvNet model can provide insights into visual representations in the human brain beyond what is already known, then the ConvNet model must predict brain activity with accuracy higher than previously published models. We compared the accuracy of ConvNet model with the Baseline model across multiple visual ROIs using the following two metrics: The percentage of significantly predicted

Table 9.1: Comparing the accuracy of the ConvNet with the Baseline model for predicting brain activity across several visual ROIs. The accuracy was quantified using two metrics - the percentage of significantly predicted voxels (% Significant) and the mean explained variance (expressed as percentage, % Variance) in the BOLD activity of each ROI. The table reports the mean ± standard deviation of these metrics computed using 1000 bootstrap runs (see supplementary materials for details). The ConvNet model is as good or better than the baseline model in almost all ROIs and outperforms the baseline model in intermediate visual ROIs like V4, LO and OFA.

Measure	**Model**	**ROI**								
		V1	V2	V3	V4	LO	OFA	FFA	EBA	PPA
% Significant	ConvNet	32.8 ± 2.9	24.6 ± 1.9	16.3 ± 1.6	17.7 ± 1.6	41.7 ± 2.2	67.3 ± 4.3	69.2 ± 3.1	60.1 ± 3.4	47.7 ± 2.0
	Baseline	32.6 ± 2.9	26.6 ± 2.6	13.9 ± 1.9	11.0 ± 1.5	32.4 ± 1.9	53.3 ± 3.6	65.0 ± 3.5	57.6 ± 3.3	47.7 ± 2.6
% Variance	ConvNet	8.1 ± 0.6	6.4 ± 0.4	4.5 ± 0.3	4.7 ± 0.3	10.8 ± 0.8	17.4 ± 2.0	19.7 ± 2.2	15.5 ± 1.6	14.8 ± 1.3
	Baseline	7.5 ± 0.6	6.4 ± 0.5	4.0 ± 0.3	3.5 ± 0.2	8.4 ± 0.6	13.8 ± 1.3	17.8 ± 1.7	14.6 ± 1.4	14.2 ± 1.4

voxels in a ROI and the percentage explained variance in the BOLD response within a ROI.

For making this comparison, voxels belonging to the same ROI were grouped across all the subjects. The percentage of significantly predicted voxels was calculated as the percentage of voxels within a ROI for which BOLD responses were predicted with p-value < 0.001. The explained variance in BOLD response of each ROI was calculated as the mean explained variance in BOLD responses of voxels assigned to the ROI (see supplementary materials for more details). The results are reported in table 9.1 indicate that ConvNet and the Baseline model make comparable predictions in early and late visual areas. The ConvNet model outperforms the Baseline model in intermediate visual areas. This suggests that the ConvNet model might provide novel insights about visual features represented by the intermediate visual cortex.

9.4.2 The hierarchy of visual representations in the ConvNet mimics the hierarchy of visual representations in the human brain

Does the ConvNet model provide insights into how the brain transforms low-level visual features into high-level visual features? If it is the case that the ConvNet provides a plausible computational hypothesis for how the brain transforms low-level visual features into high-level visual features then the low, mid and high-level features represented in both the systems must match.

To investigate if this was the case, we plotted the ConvNet layer assigned to

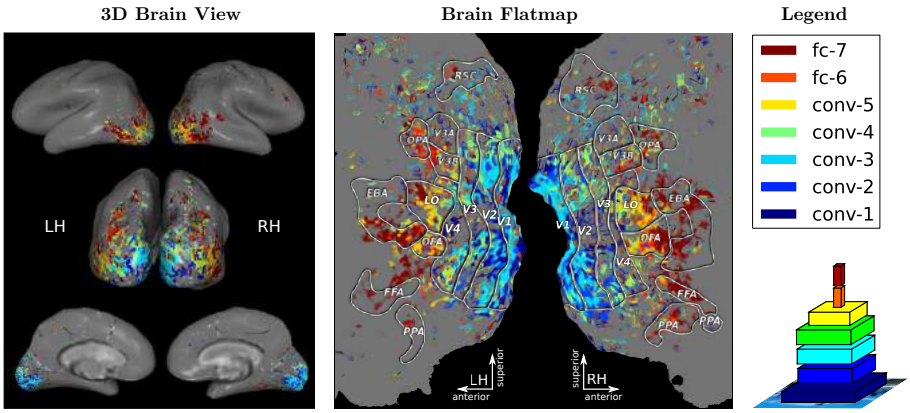


Figure 9.3: Relationship between the feature hierarchies of ConvNet and the human brain. Voxels have been color coded to show the ConvNet layer that was found to be optimal for predicting their activity. The voxels with p_{SNR}-value > 0.001 are shown in gray. The alpha channel of the voxel colors has been modulated to reflect the accuracy of the ConvNet model in predicting the BOLD activity of voxels. The alpha value for all voxels predicted with p-value < 0.001 has been set to 1. The alpha values for the remaining voxels has been set in proportion to r on linear scale ranging from 0 to 1. The lower (conv-1, conv-2, conv-3), intermediate (conv-4, conv-5) and higher (fc-6, fc-7) layer of the ConvNet were found to be optimal for voxels in posterior, intermediate and anterior areas of the visual cortex respectively. This shows that the hierarchy of visual representations in the ConvNet mimics the hierarchy of visual representations in the human brain.

every voxel on a flatmap of the brain (figure 9.3). Each voxel in this figure has been color-coded to reflect its corresponding ConvNet layer. The figure shows that lower, middle and higher layers of the ConvNet were optimal for predicting BOLD responses in posterior, intermediate and anterior parts of the visual cortex. This implies that low, mid and high-level visual features represented by the ConvNet are related to the low, mid and high-level visual features represented by the brain by a linear transformation. From this it can be concluded that the hierarchy of visual representations in the ConvNet mimics the hierarchy of visual representations in the human brain.

Figure 9.4: Using the ConvNet model to probe the stimulus tuning of individual voxels. Each row represents the tuning of a single voxel. The visual tuning of two voxels from the visual areas V1, V4, FFA and PPA is shown. These voxels were manually chosen from a set of voxels obtained after pooling voxels from all four subjects. The ConvNet model fit to each voxel was used to filter a set of 280K natural images. The six columns at left show the six images that the ConvNet model for each voxel predicts will most increase BOLD activity, and the six columns at right show the images that the model predicts will most decrease BOLD activity. The V1 voxels are predicted to increase activity when images consist of high texture and to decrease activity when images contain landscapes. The V4 voxels are predicted to increase activity when images contain a circular shapes and to decrease activity when they contain landscapes. The FFA voxels are predicted to increase activity when images contain human and animal faces and to decrease activity when they contain scenes. The PPA voxels are predicted to increase activity when images contain scenes and trains and to decrease activity when they contain animate objects.

9.4.3 Investigating Visual Representations in the Human Brain

Insights into visual representations of the human brain can be developed by visualizing the features represented by individual voxels. For this, we used the

ConvNet model to predict BOLD activity of individual voxels to a collection of more than 280K natural images (see supplementary material for a detailed description of this image collection). Then for every voxel independently, these images were rank-ordered according to the predicted BOLD activity. The top and bottom images within this ranking provide qualitative intuition about the features that are represented by a particular voxel (see supplementary material for more details). Figure 9.4 shows the top and bottom six images for two voxels in V1, V4, FFA and PPA. As there were too many voxels to visualize, only two sample voxels from each ROI were chosen in the following way: For each ROI, all voxels predicted with a p-value < 0.0001 were pooled across the four subjects. From this set, two voxels were manually chosen to illustrate the range of visual representations in a ROI. A random sample of voxels from V1, V4, FFA and PPA is shown in the supplementary materials.

The V1 voxels are predicted to increase activity when images consist of high texture and to decrease activity when images contain landscapes. This result is not surprising because V1 is known to contain neurons that respond to oriented edges and images with high texture are likely to excite a large number of V1 neurons. This in turn would cause the V1 voxels to elicit large responses to textured images. The FFA voxels are predicted to increase activity when images contain human and animal faces and to decrease activity when they contain scenes. These results are consistent with previous accounts of FFA [331, 332]. The PPA voxels are predicted to increase activity when images contain scenes and trains and to decrease activity when they contain animate objects. The geometric structure of trains is not very different from that of buildings. This suggests PPA voxels encode specific geometric structures useful for identifying scenes/places and are not likely to be selective for any object categories. This interpretation of features represented in PPA is consistent with the findings of [345] and previous accounts of PPA [334]. These results demonstrate that using the ConvNet model, results of multiple previous fMRI studies that investigated visual representations in individual ROIs [329, 331, 333, 334] can be reproduced using only a single fMRI study.

Despite several past studies, the understanding of visual representations in V4 is unsatisfactory [335, 346, 347]. Our analysis reveals that a subset of V4 voxels are predicted to increase BOLD activity when images contain a circular shapes and to decrease activity when they contain landscapes. This result is qualitatively consistent with neurophysiological reports that area V4 is selective for curvature and radial patterns [347] and shows that ConvNet can be used to investigate visual representations in intermediate visual ROIs.

9.5 Discussion

Understanding how the brain transforms low-level visual features into high-level visual features requires developing computational theories that make testable predictions about visual representations in the brain. In the past, such theories have either been based purely on the neurophysiological findings or have been inspired by Barlow's redundancy reduction hypothesis [336]. Hubel and Wiesel's finding of simple and complex cells [328] led to the computational hypothesis that the hierarchy of visual features in the brain was constructed by consequent stages of linear filtering, pooling and point-wise non-linearities. This idea was first championed by the Neocognitron model [282] and later by the HMAX model [348]. In a different line of work, past studies found that computational models based on Barlow's idea predicted what features were represented by neurons in V1 [38, 39]. Since then, several studies have attempted to use the redundancy reduction hypothesis for building computational models that explain features represented in visual areas beyond V1 [349, 350]. However, these studies have met with limited success and no prior study has been able to construct a computational model that provides plausible predictions about features represented across the visual cortex.

With help of experiments presented above, we demonstrated that hierarchy of visual representations in the ConvNet mimics the hierarchy of visual representations in the human brain. In contrast to past studies that proposed a similar model of computation [282, 348], the key difference is that the ConvNet model was trained for the task of object recognition. Models used in these previous studies were not optimized for performing any particular task and this suggests that that computational theories optimizing for end task performance such as image classification, might provide a better account of how brain represents visual information as compared to models capturing goal independent natural image statistics (i.e. unsupervised learning [36, 38, 39, 102]).

Some recent studies [351, 352] have provided evidence that ConvNets can explain visual representations in the Inferior Temporal (IT) cortex of macaques and humans. However, these results are not surprising because IT appears to represent semantically meaningful features such as faces and places and the ConvNet was trained for object recognition. What our results show is that the ConvNet mimics the hierarchy of visual representations across the visual cortex. This implies that not only is the ConvNet plausible model of visual processing but it can also be used to study visual representations throughout the visual cortex. Such claims cannot be made based on the results of any previous work.

One potential critique of our work is that unlike the brain, the ConvNet has no feedback or recurrent connections. How is it then that the ConvNet is able

to predict brain activity across the visual cortex? One explanation is provided by past studies that have shown that the brain can perform object recognition using feed-forward computations only [353]. Moreover, although the ConvNet model outperforms previously proposed models for predicting brain activity, there still is substantial amount of variance in brain activity that is not explained by the ConvNet model (see table 9.1).

Another potential critique of our work is that several architectural choices involved in designing the ConvNet (such as the number of layers) were simply made as a result of the fact that they led to good performance on the task of object recognition [342]. These choices may not be optimal for predicting brain activity and consequently the ConvNet model we used is probably sub-optimal. Modifying the ConvNet architecture to incorporate computational mechanisms like recurrence and feedback, and optimally choosing parameters such as the number of layers, the number of units in a layer, and the choice of specific non-linearity will lead to models that make more accurate predictions of brain activity. Future work on developing such models is likely to provide a more nuanced understanding of how the brain processes and represents visual stimuli.

9.6 Takeaways

The main result of our work is that the hierarchy of visual representations in the ConvNet mimics the hierarchy of visual representations in the human brain. This observation was either concurrently or later replicated by multiple other groups [351, 354–357]. This suggests that understanding visual representations in the ConvNet can help us understand the visual representations in the human brain. As evidence, we have shown that the ConvNet model reveals visual features represented by individual voxels. Our results also provide evidence that computational models optimized for executing ecologically relevant tasks (like object recognition) as opposed to models optimized solely for estimating natural image statistics can provide better hypothesis about how brain transforms low-level visual representations into high-level visual representations.

Similar to our results in the visual pathway, McDermott's group reported that hierarchy of auditory representations learned by deep networks optimized for speech and music recognition [358] resembled the hierarchy of features in the auditory cortex. Our and their findings taken together suggest that by optimizing for ecologically relevant tasks, deep neural networks learn representations similar to the sensory cortices in the human brain. This in turn means that we can understand the representations in the brain by understanding the representations of sensory data in

neural network which can be more easily manipulated. Furthermore, based on these results I also hypothesize that if we were to train neural networks that performed complex sensorimotor tasks such arranging objects, using a tool to manipulate another object, planning how to stack objects in a tower etc – these networks might provide a tool for understanding sensorimotor representations in the human brain. This is a very exciting avenue for future research.

Chapter 10

Conclusions

This dissertation is a work of a continually evolving agent that learned by experimenting. The only end is evolution. Check back with me and I will tell you how curious my agents are! Maybe it will be me, or my agents answering you. Shall you know? Think, what day that might be. On the path I walk, only a few words remain true, others evolve. Remember,

"Imagination is more important than knowledge. For knowledge is limited, whereas imagination embraces the entire world, stimulating progress, giving birth to evolution"

"I am enough of an artist to draw freely upon my imagination. Imagination is more important than knowledge. For knowledge is limited, whereas imagination encircles the world."

— Albert Einstein

"Knowledge takes you from A to B, Imagination takes you everywhere"

— Controversial

and you will find the answers you seek. As for me,

"Little by little, cell by cell, the truth shall evolve"

"I am neither special nor ordinary,
I am evolution decoding evolution,
From simplicity emerges complexity,
Embodied I am, the rest is a myth!"

— Pulkit Agrawal

Bibliography

[1] F.-H. Hsu, *Behind Deep Blue: Building the computer that defeated the world chess champion*. Princeton University Press, 2004.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, 2015.

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, pp. 2980–2988, IEEE, 2017.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[7] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[8] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

[9] J. G. C. Devol, "Programmed article transfer," June 13 1961. US Patent 2,988,237.

[10] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and Brain Sciences*, vol. 40, 2017.

[11] G. Tesauro, "Temporal difference learning and td-gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[12] J. Schaeffer, N. Burch, Y. Björnsson, A. Kishimoto, M. Müller, R. Lake, P. Lu, and S. Sutphen, "Checkers is solved," *science*, vol. 317, no. 5844, pp. 1518–1522, 2007.

[13] N. Brown and T. Sandholm, "Superhuman ai for heads-up no-limit poker: Libratus beats top professionals," *Science*, p. eaao1733, 2017.

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[15] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014*, pp. 818–833, Springer, 2014.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR*, 2015.

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 580–587, IEEE, 2014.

[19] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, 2015.

[20] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection.," in *CVPR*, 2017.

[21] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," *ICCV*, 2011.

[22] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *CVPR*, 2017.

[23] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *CVPR*, 2016.

[24] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, "Human pose estimation with iterative error feedback," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4733–4742, 2016.

[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR*, 2009.

[26] 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 *European conference on computer vision*, pp. 740–755, Springer, 2014.

[27] 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," *ICML*, 2014.

[28] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems*, pp. 3630–3638, 2016.

[29] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," *ICLR*, 2017.

[30] B. Hariharan and R. Girshick, "Low-shot visual object recognition," *ICCV*, 2017.

[31] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel, "Meta-learning for semi-supervised few-shot classification," *arXiv preprint arXiv:1803.00676*, 2018.

[32] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum, "One shot learning of simple visual concepts," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 33, 2011.

[33] J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman, "How to grow a mind: Statistics, structure, and abstraction," *science*, vol. 331, no. 6022, pp. 1279–1285, 2011.

[34] S. Harnad, "The symbol grounding problem," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1-3, pp. 335–346, 1990.

[35] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and helmholtz free energy," in *Advances in neural information processing systems*, pp. 3–10, 1994.

[36] R. Salakhutdinov and G. E. Hinton, "Deep boltzmann machines," in *International Conference on Artificial Intelligence and Statistics*, pp. 448–455, 2009.

[37] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *Artificial Neural Networks and Machine Learning–ICANN*, pp. 44–51, Springer, 2011.

[38] B. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, 1996.

[39] A. J. Bell and T. J. Sejnowski, "The "independent components" of natural scenes are edge filters.," *Vision research*, vol. 37, pp. 3327–38, Dec. 1997.

[40] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *ICLR*, 2014.

[41] L. Wiskott and T. J. Sejnowski, "Slow feature analysis: Unsupervised learning of invariances," *Neural computation*, vol. 14, no. 4, pp. 715–770, 2002.

[42] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun, "Unsupervised learning of spatiotemporally coherent metrics," *ICCV*, 2015.

[43] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," *ICML*, 2008.

[44] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[45] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," tech. rep., Citeseer, 2009.

[46] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *ICCV*, 2015.

[47] M. Noroozi, H. Pirsiavash, and P. Favaro, "Representation learning by learning to count," *arXiv preprint arXiv:1708.06734*, 2017.

[48] P. Agrawal, J. Carreira, and J. Malik, "Learning to see by moving," *arXiv preprint arXiv:1505.01596*, 2015.

[49] D. Jayaraman and K. Grauman, "Learning image representations equivariant to ego-motion," *arXiv preprint arXiv:1505.02206*, 2015.

[50] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *ICCV*, 2015.

[51] X. Wang, K. He, and A. Gupta, "Transitive invariance for selfsupervised visual representation learning," in *Proc. of Int'l Conf. on Computer Vision (ICCV)*, 2017.

[52] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba, "Ambient sound provides supervision for visual learning," *ECCV*, 2016.

[53] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan, "Learning features by watching objects move," *CVPR*, 2017.

[54] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *CVPR*, 2016.

[55] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial Feature Learning," *ICLR*, 2017.

[56] R. Zhang, P. Isola, and A. A. Efros, "Colorful Image Colorization," *ECCV*, 2016.

[57] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," *arXiv:1311.2901*, 2013.

[58] P. Agrawal, R. Girshick, and J. Malik, "Analyzing the performance of multilayer neural networks for object recognition," in *Computer Vision–ECCV 2014*, pp. 329–344, Springer, 2014.

[59] D. Hadfield-Menell, S. Milli, P. Abbeel, S. J. Russell, and A. Dragan, "Inverse reward design," in *Advances in Neural Information Processing Systems*, pp. 6765–6774, 2017.

[60] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[61] D. Amodei and J. Clark, "Faulty reward functions in the wild, 2016," *URL https://blog. openai. com/faulty-reward-functions*, 2016.

[62] D. A. Pomerleau, "ALVINN: An autonomous land vehicle in a neural network," in *NIPS*, 1989.

[63] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, 2009.

[64] S. Schaal, "Is imitation learning the route to humanoid robots?," *Trends in cognitive sciences*, 1999.

[65] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *ICML*, pp. 663–670, 2000.

[66] A. Bandura and R. H. Walters, *Social learning theory*, vol. 1. Prentice-hall Englewood Cliffs, NJ, 1977.

[67] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Teaching by showing: Generating robot programs by visual observation of human performance," in *International Symposium on Industrial Robots*, 1989.

[68] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by watching: Extracting reusable task knowledge from visual observation of human performance," *IEEE transactions on robotics and automation*, vol. 10, no. 6, pp. 799–822, 1994.

[69] K. Ikeuchi and T. Suehiro, "Toward an assembly plan from observation. i. task recognition with polyhedral objects," *IEEE Transactions on Robotics and Automation*, 1994.

[70] C. Breazeal and B. Scassellati, "Robots that imitate humans," *Trends in cognitive sciences*, 2002.

[71] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, 2004.

[72] Y. Yang, Y. Li, C. Fermüller, and Y. Aloimonos, "Robot learning manipulation action plans by" watching" unconstrained videos from the world wide web.," in *AAAI*, pp. 3686–3693, 2015.

[73] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[74] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[75] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2, pp. 1150–1157, Ieee, 1999.

[76] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.

[77] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, 1989.

[78] R. A. Brooks, "Intelligence without representation," *Artificial intelligence*, vol. 47, no. 1-3, pp. 139–159, 1991.

[79] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik, "Learning visual predictive models of physics for playing billiards," *ICLR*, 2016.

[80] A. Michotte, "The perception of causality," 1963.

[81] M. McCloskey, "Intuitive physics," *Scientific american*, 1983.

[82] L. Smith and M. Gasser, "The development of embodied cognition: Six lessons from babies," *Artificial life*, 2005.

[83] A. Gopnik, A. N. Meltzoff, and P. K. Kuhl, *The scientist in the crib: Minds, brains, and how children learn.* 1999.

[84] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International Conference on Machine Learning (ICML)*, vol. 2017, 2017.

[85] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," *NIPS*, 2016.

[86] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining self-supervised learning and imitation for vision-based rope manipulation," *ICRA*, 2017.

[87] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell, "Zero-shot visual imitation," in *ICLR*, 2018.

[88] P. Felsen, P. Agrawal, and J. Malik, "What will happen next? forecasting player moves in sports videos," *ICCV, Oct*, vol. 1, p. 2, 2017.

[89] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*, pp. 568–576, 2014.

[90] Y. Bengio, A. C. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," *CoRR, abs/1206.5538*, vol. 1, 2012.

[91] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 609–616, ACM, 2009.

[92] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–8, IEEE, 2007.

[93] J. J. Gibson, *The Ecological Approach to Visual Perception*. Houghton Mifflin, 1979.

[94] J. E. Cutting, *Perception with an eye for motion*, vol. 177.

[95] S. Soatto, "Visual scene representations: Sufficiency, minimality, invariance and approximations," *arXiv preprint arXiv:1411.7676*, 2014.

[96] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, pp. 2169–2178, IEEE, 2006.

[97] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[98] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[99] D. M. Chen, G. Baatz, K. Koser, S. S. Tsai, R. Vedantham, T. Pylvanainen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, *et al.*, "City-scale landmark identification on mobile devices," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 737–744, IEEE, 2011.

[100] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pp. 3485–3492, IEEE, 2010.

[101] H. Mobahi, R. Collobert, and J. Weston, "Deep learning from temporal coherence in video," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 737–744, ACM, 2009.

[102] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological cybernetics*, vol. 59, no. 4-5, pp. 291–294, 1988.

[103] H. Barlow, "Unsupervised learning," *Neural computation*, vol. 1, no. 3, pp. 295–311, 1989.

[104] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra, "Video (language) modeling: a baseline for generative models of natural videos," *arXiv preprint arXiv:1412.6604*, 2014.

[105] P. Fischer, A. Dosovitskiy, and T. Brox, "Descriptor matching with convolutional neural networks: a comparison to sift," *arXiv preprint arXiv:1405.5769*, 2014.

[106] R. Memisevic and G. E. Hinton, "Learning to represent spatial transformations with factored higher-order boltzmann machines," *Neural Computation*, vol. 22, no. 6, pp. 1473–1492, 2010.

[107] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, J. Han, B. Flepp, U. Muller, and Y. LeCun, "Online learning for offroad robots: Using spatial label propagation to learn long-range traversability," in *Proc. of Robotics: Science and Systems (RSS)*, vol. 11, p. 32, 2007.

[108] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 539–546, IEEE, 2005.

[109] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," *ACM Multimedia*, 2014.

[110] A. Oliva and A. Torralba, "Building the gist of a scene: The role of global image features in recognition," *Progress in brain research*, vol. 155, pp. 23–36, 2006.

[111] L. Bourdev, S. Maji, T. Brox, and J. Malik, "Detecting people using mutually consistent poselet activations," in *Computer Vision–ECCV 2010*, pp. 168–181, Springer, 2010.

[112] S. Vicente, J. Carreira, L. Agapito, and J. Batista, "Reconstructing pascal voc," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 41–48, IEEE, 2014.

[113] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 966–1005, 1988.

[114] C. Vondrick, H. Pirsiavash, and A. Torralba, "Anticipating the future by watching unlabeled video," *CVPR*, 2016.

[115] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *ICLR*, 2016.

[116] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *JMLR*, 2016.

[117] S. Lange, M. Riedmiller, and A. Voigtlander, "Autonomous reinforcement learning on raw visual input data in a real world application," in *IJCNN*, 2012.

[118] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," *ICRA*, 2016.

[119] S. Lange and M. A. Riedmiller, "Deep learning of visual control policies.," in *ESANN*, 2010.

[120] T. C. Kietzmann and M. Riedmiller, "The neuro slot car racer: Reinforcement learning in a real world setting," in *ICMLA*, 2009.

[121] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," *ICRA*, 2016.

[122] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with large-scale data collection," in *ISER*, 2016.

[123] M. R. Dogar and S. S. Srinivasa, "A planning framework for non-prehensile manipulation under clutter and uncertainty," *Autonomous Robots*, 2012.

[124] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, 2014.

[125] M. I. Jordan and D. E. Rumelhart, "Forward models: Supervised learning with a distal teacher," *Cognitive science*, 1992.

[126] D. M. Wolpert, Z. Ghahramani, and M. I. Jordan, "An internal model for sensorimotor integration," *Science-AAAS-Weekly Paper Edition*, 1995.

[127] I. Lenz, R. Knepper, and A. Saxena, "Deepmpc: Learning deep latent features for model predictive control," in *RSS*, 2015.

[128] N. Wahlström, T. B. Schön, and M. P. Deisenroth, "From pixels to torques: Policy learning with deep dynamical models," *arXiv:1502.02251*, 2015.

[129] 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.

[130] J. Oh, X. Guo, H. Lee, R. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in atari games," *NIPS*, 2015.

[131] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum, "Galileo: Perceiving physical object properties by integrating a physics engine with deep learning," in *NIPS*, 2015.

[132] R. Mottaghi, H. Bagherinezhad, M. Rastegari, and A. Farhadi, "Newtonian image understanding: Unfolding the dynamics of objects in static images," *CVPR*, 2016.

[133] A. Lerer, S. Gross, and R. Fergus, "Learning physical intuition of block towers by example," *ICML*, 2016.

[134] M. Kopicki, S. Zurek, R. Stolkin, T. Mörwald, and J. Wyatt, "Learning to predict how rigid objects behave under simple manipulation," in *ICRA*, 2011.

[135] M. Lau, J. Mitani, and T. Igarashi, "Automatic learning of pushing strategy for delivery of irregular-shaped objects," in *ICRA*, 2011.

[136] T. Meriçli, M. Veloso, and H. L. Akın, "Push-manipulation of complex passive mobile objects using experimentally acquired motion models," *Autonomous Robots*, 2015.

[137] S. Kolev and E. Todorov, "Physically consistent state estimation and system identification for contacts," in *International Conference on Humanoid Robots*, 2015.

[138] J. Hamrick, P. Battaglia, and J. B. Tenenbaum, "Internal physics models guide probabilistic judgments about object dynamics," in *Cognitive Science Society*, 2011.

[139] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007.

[140] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[141] A. P. Shon, D. Verma, and R. P. Rao, "Active imitation learning," in *AAAI*, pp. 756–762, 2007.

[142] G. Rizzolatti, L. Fadiga, V. Gallese, and L. Fogassi, "Premotor cortex and the recognition of motor actions," *Cognitive brain research*, vol. 3, no. 2, pp. 131–141, 1996.

[143] G. Rizzolatti and L. Craighero, "The mirror-neuron system," *Annu. Rev. Neurosci.*, vol. 27, pp. 169–192, 2004.

[144] G. Hickok, "Eight problems for the mirror neuron theory of action understanding in monkeys and humans," *Journal of cognitive neuroscience*, vol. 21, no. 7, pp. 1229–1243, 2009.

[145] D. Foster and P. Dayan, "Structure in the space of value functions," *Machine Learning*, 2002.

[146] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *ICML*, 2015.

[147] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, and S. Levine, "Time-contrastive networks: Self-supervised learning from video," in *ICRA*, 2018.

[148] H. Chui and A. Rangarajan, "A new algorithm for non-rigid point matching," in *CVPR*, 2000.

[149] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *ICML*, 2004.

[150] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI*, 2008.

[151] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *NIPS*, 2016.

[152] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," in *NIPS*, 2017.

[153] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," *CoRL*, 2017.

[154] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," *CVPR*, 2017.

[155] P. Sermanet, K. Xu, and S. Levine, "Unsupervised perceptual rewards for imitation learning," in *RSS*, 2017.

[156] B. C. Stadie, P. Abbeel, and I. Sutskever, "Third-person imitation learning," in *ICLR*, 2017.

[157] Y. Liu, A. Gupta, P. Abbeel, and S. Levine, "Imitation from observation: Learning to imitate behaviors from raw video via context translation," *ICRA*, 2018.

[158] H. Koichi and H. Tom, *Visual servoing: real-time control of robot manipulators based on visual sensory feedback*, vol. 7. World scientific, 1993.

[159] B. H. Yoshimi and P. K. Allen, "Active, uncalibrated visual servoing," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 156–161, IEEE, 1994.

[160] W. J. Wilson, C. W. Hulls, and G. S. Bell, "Relative end-effector control using cartesian position based visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 684–696, 1996.

[161] G. Caron, E. Marchand, and E. M. Mouaddib, "Photometric visual servoing for omnidirectional cameras," *Autonomous Robots*, vol. 35, no. 2-3, pp. 177–193, 2013.

[162] T. Lampe and M. Riedmiller, "Acquiring visual servoing reaching and grasping skills using neural reinforcement learning," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pp. 1–8, IEEE, 2013.

[163] A. X. Lee, S. Levine, and P. Abbeel, "Learning visual servoing with deep features and fitted q-iteration," *arXiv preprint arXiv:1703.11000*, 2017.

[164] J. E. Hopcroft, J. K. Kearney, and D. B. Krafft, "A case study of flexible object manipulation," *The International Journal of Robotics Research*, vol. 10, no. 1, pp. 41–50, 1991.

[165] H. Mayer, F. Gomez, D. Wierstra, I. Nagy, A. Knoll, and J. Schmidhuber, "A system for robotic heart surgery that learns to tie knots using recurrent neural networks," *Advanced Robotics*, vol. 22, no. 13-14, pp. 1521–1537, 2008.

[166] J. Schulman, A. Gupta, S. Venkatesan, M. Tayson-Frederick, and P. Abbeel, "A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario," in *IROS*, 2013.

[167] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2308–2315, IEEE, 2010.

[168] M. Inaba and H. Inoue, "Hand eye coordination in rope handling," *Journal of the Robotics Society of Japan*, vol. 3, no. 6, pp. 538–547, 1985.

[169] M. Saha and P. Isto, "Motion planning for robotic manipulation of deformable linear objects," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 2478–2484, IEEE, 2006.

[170] M. Bell, *Flexible object manipulation.* PhD thesis, Dartmouth College, Hanover, New Hampshire, 2010.

[171] Y. Yamakawa, A. Namiki, M. Ishikawa, and M. Shimojo, "One-handed knotting of a flexible rope with a high-speed multifingered hand having tactile sensors," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 703–708, IEEE, 2007.

[172] T. Morita, J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi, "Knot planning from observation," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 3, pp. 3887–3892, IEEE, 2003.

[173] R. H. Crowell and R. H. Fox, *Introduction to knot theory*, vol. 57. Springer Science & Business Media, 2012.

[174] H. Wakamatsu, E. Arai, and S. Hirai, "Knotting/unknotting manipulation of deformable linear objects," *The International Journal of Robotics Research*, vol. 25, no. 4, pp. 371–395, 2006.

[175] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," *arXiv preprint arXiv:1401.4082*, 2014.

[176] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *ICML*, 2017.

[177] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, 2017.

[178] A. J. Davison and D. W. Murray, "Mobile robot localisation using active vision," in *ECCV*, 1998.

[179] Mapillary, "Open source structure from motion pipeline," *https://github.com/mapillary/OpenSfM*, 2016.

[180] E. L. Ryan, Richard; Deci, "Intrinsic and extrinsic motivations: Classic definitions and new directions," *Contemporary Educational Psychology*, 2000.

[181] P. J. Silvia, "Curiosity and motivation," in *The Oxford Handbook of Human Motivation*, 2012.

[182] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," in *NIPS*, 2016.

[183] P. Poupart, N. Vlassis, J. Hoey, and K. Regan, "An analytic solution to discrete bayesian reinforcement learning," in *ICML*, 2006.

[184] M. Lopes, T. Lang, M. Toussaint, and P.-Y. Oudeyer, "Exploration in model-based reinforcement learning by empirically estimating learning progress," in *NIPS*, 2012.

[185] J. Schmidhuber, "A possibility for implementing curiosity and boredom in model-building neural controllers," in *From animals to animats: Proceedings of the first international conference on simulation of adaptive behavior*, 1991.

[186] J. Schmidhuber, "Formal theory of creativity, fun, and intrinsic motivation (1990–2010)," *IEEE Transactions on Autonomous Mental Development*, 2010.

[187] S. P. Singh, A. G. Barto, and N. Chentanez, "Intrinsically motivated reinforcement learning.," in *NIPS*, 2005.

[188] S. Mohamed and D. J. Rezende, "Variational information maximisation for intrinsically motivated reinforcement learning," in *NIPS*, 2015.

[189] B. C. Stadie, S. Levine, and P. Abbeel, "Incentivizing exploration in reinforcement learning with deep predictive models," *NIPS Workshop*, 2015.

[190] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, "Vime: Variational information maximizing exploration," in *NIPS*, 2016.

[191] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *ICML*, 2016.

[192] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "Vizdoom: A doom-based ai research platform for visual reinforcement learning," *arXiv:1605.02097*, 2016.

[193] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv:1606.01540*, 2016.

[194] A. Dosovitskiy and V. Koltun, "Learning to act by predicting the future," *ICLR*, 2016.

[195] P. Paquette, "Super mario bros. in openai gym," *github:ppaquette/gym-super-mario*, 2016.

[196] J. Fu, J. D. Co-Reyes, and S. Levine, "Ex2: Exploration with exemplar models for deep reinforcement learning," *arXiv:1703.01260*, 2017.

[197] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, *et al.*, "Learning to navigate in complex environments," *ICLR*, 2017.

[198] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner, "Intrinsic motivation systems for autonomous mental development," *Evolutionary Computation*, 2007.

[199] P.-Y. Oudeyer and F. Kaplan, "What is intrinsic motivation? a typology of computational approaches," *Frontiers in neurorobotics*, 2009.

[200] Y. Sun, F. Gomez, and J. Schmidhuber, "Planning to be surprised: Optimal bayesian exploration in dynamic environments," in *AGI*, 2011.

[201] M. Kearns and D. Koller, "Efficient reinforcement learning in factored mdps," in *IJCAI*, 1999.

[202] R. I. Brafman and M. Tennenholtz, "R-max-a general polynomial time algorithm for near-optimal reinforcement learning," *JMLR*, 2002.

[203] A. S. Klyubin, D. Polani, and C. L. Nehaniv, "Empowerment: A universal agent-centric measure of control," in *Evolutionary Computation*, 2005.

[204] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in atari games," in *NIPS*, 2015.

[205] H. Tang, R. Houthooft, D. Foote, A. Stooke, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, "# exploration: A study of count-based exploration for deep reinforcement learning," *arXiv:1611.04717*, 2016.

[206] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," in *NIPS*, 2016.

[207] S. Still and D. Precup, "An information-theoretic approach to curiosity-driven reinforcement learning," *Theory in Biosciences*, 2012.

[208] D. Y. Little and F. T. Sommer, "Learning and exploration in action-perception loops," *Closing the Loop Around Neural Systems*, 2014.

[209] J. Storck, S. Hochreiter, and J. Schmidhuber, "Reinforcement driven information acquisition in non-deterministic environments," in *ICANN*, 1995.

[210] S. Sukhbaatar, I. Kostrikov, A. Szlam, and R. Fergus, "Intrinsic motivation and automatic curricula via asymmetric self-play," in *ICLR*, 2018.

[211] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," *ICLR*, 2017.

[212] E. Shelhamer, P. Mahmoudieh, M. Argus, and T. Darrell, "Loss is its own reward: Self-supervision for reinforcement learning," *arXiv:1612.07307*, 2017.

[213] K. Gregor, D. J. Rezende, and D. Wierstra, "Variational intrinsic control," *ICLR Workshop*, 2017.

[214] R. Dubey, P. Agrawal, D. Pathak, T. L. Griffiths, and A. A. Efros, "Investigating human priors for playing video games," *International Conference on Machine Learning*, 2018.

[215] D. Pathak, Y. Shentu, D. Chen, P. Agrawal, T. Darrell, S. Levine, and J. Malik, "Learning instance segmentation by interaction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2042–2045, 2018.

[216] J. Oh, S. Singh, and H. Lee, "Value prediction network," in *NIPS*, 2017.

[217] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *ICML*, 2016.

[218] J. J. Gibson, *The ecological approach to visual perception: classic edition*. Psychology Press, 2014.

[219] C. G. D. Wasser, *An object-oriented representation for efficient reinforcement learning*. Rutgers The State University of New Jersey-New Brunswick, 2010.

[220] F. Doshi-Velez and Z. Ghahramani, "A comparison of human and agent reinforcement learning in partially observable domains," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 33, 2011.

[221] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and Brain Sciences*, pp. 1–101, 2016.

[222] P. A. Tsividis, T. Pouncy, J. L. Xu, J. B. Tenenbaum, and S. J. Gershman, "Human learning in atari."," in *The AAAI 2017 Spring Symposium on Science of Intelligence: Computational Principles of Natural and Artificial Intelligence*, 2017.

[223] E. S. Spelke and K. D. Kinzler, "Core knowledge," *Developmental science*, vol. 10, no. 1, pp. 89–96, 2007.

[224] S. Carey, *The origin of concepts*. Oxford University Press, 2009.

[225] C. Diuk, A. Cohen, and M. L. Littman, "An object-oriented representation for efficient reinforcement learning," in *Proceedings of the 25th international conference on Machine learning*, pp. 240–247, ACM, 2008.

[226] K. Kansky, T. Silver, D. A. Mély, M. Eldawy, M. Lázaro-Gredilla, X. Lou, N. Dorfman, S. Sidor, S. Phoenix, and D. George, "Schema networks: Zero-shot transfer with a generative causal model of intuitive physics," in *International Conference on Machine Learning*, pp. 1809–1818, 2017.

[227] K. Narasimhan, R. Barzilay, and T. Jaakkola, "Deep transfer in reinforcement learning by language grounding," *arXiv preprint arXiv:1708.00133*, 2017.

[228] S. J. Hespos, A. L. Ferry, and L. J. Rips, "Five-month-old infants have different expectations for solids and liquids," *Psychological Science*, vol. 20, no. 5, pp. 603–611, 2009.

[229] R. Baillargeon, "Infants' physical world," *Current directions in psychological science*, vol. 13, no. 3, pp. 89–94, 2004.

[230] R. Baillargeon, "How do infants learn about the physical world?," *Current Directions in Psychological Science*, vol. 3, no. 5, pp. 133–140, 1994.

[231] D. M. Wolpert and Z. Ghahramani, "Computational principles of movement neuroscience," *Nature neuroscience*, vol. 3, pp. 1212–1217, 2000.

[232] N. D. Daw, J. P. O'Doherty, P. Dayan, B. Seymour, and R. J. Dolan, "Cortical substrates for exploratory decisions in humans," *Nature*, vol. 441, no. 7095, pp. 876–879, 2006.

[233] J. D. Cohen, S. M. McClure, and J. Y. Angela, "Should I stay or should I go? How the human brain manages the trade-off between exploitation and exploration," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 362, no. 1481, pp. 933–942, 2007.

[234] W. B. Knox, A. R. Otto, P. Stone, and B. Love, "The nature of belief-directed exploratory choice in human decision-making," *Frontiers in psychology*, vol. 2, p. 398, 2012.

[235] S. J. Gershman and Y. Niv, "Novelty and inductive generalization in human reinforcement learning," *Topics in cognitive science*, vol. 7, no. 3, pp. 391–415, 2015.

[236] E. S. Spelke, "Principles of object perception," *Cognitive science*, vol. 14, no. 1, pp. 29–56, 1990.

[237] J. M. Mandler, "Representation," in *Cognition, perception, and language: Handbook of child psychology*, John Wiley & Sons Inc, 1998.

[238] D. Mareschal and P. C. Quinn, "Categorization in infancy," *Trends in cognitive sciences*, vol. 5, no. 10, pp. 443–450, 2001.

[239] A. F. Pereira and L. B. Smith, "Developmental changes in visual object recognition between 18 and 24 months of age," *Developmental science*, vol. 12, no. 1, pp. 67–80, 2009.

[240] K. S. Kretch and K. E. Adolph, "Cliff or step? Posture-specific learning at the edge of a drop-off," *Child Development*, vol. 84, no. 1, pp. 226–240, 2013.

[241] K. Kitani, B. Ziebart, J. Bagnell, and M. Hebert, "Activity forecasting," *Computer Vision–ECCV 2012*, pp. 201–214, 2012.

[242] C. Vondrick, H. Pirsiavash, and A. Torralba, "Anticipating the future by watching unlabeled video," *arXiv preprint arXiv:1504.08023*, 2015.

[243] A. Jain, H. S. Koppula, B. Raghavan, S. Soh, and A. Saxena, "Car that knows before you do: Anticipating maneuvers via learning temporal driving models," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3182–3190, 2015.

[244] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," *ICLR*, 2016.

[245] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *Advances In Neural Information Processing Systems*, pp. 613–621, 2016.

[246] H. S. Koppula and A. Saxena, "Anticipating human activities using object affordances for reactive robotic response," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 38, no. 1, pp. 14–29, 2016.

[247] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik, "Learning visual predictive models of physics for playing billiards," *arXiv preprint arXiv:1511.07404*, 2015.

[248] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros, "View synthesis by appearance flow," in *European Conference on Computer Vision*, pp. 286–301, Springer, 2016.

[249] J. Walker, A. Gupta, and M. Hebert, "Patch to the future: Unsupervised visual prediction," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 3302–3309, IEEE, 2014.

[250] J. Walker, C. Doersch, A. Gupta, and M. Hebert, "An uncertain future: Forecasting from static images using variational autoencoders," in *European Conference on Computer Vision*, pp. 835–851, Springer, 2016.

[251] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 961–971, 2016.

[252] P. V. K. Borges, N. Conci, and A. Cavallaro, "Video-based human behavior understanding: a survey," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 23, no. 11, pp. 1993–2008, 2013.

[253] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei, "Every moment counts: Dense detailed labeling of actions in complex videos," *arXiv preprint arXiv:1507.05738*, 2015.

[254] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, IEEE, 2008.

[255] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[256] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden markov models for complex action recognition," in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pp. 994–999, IEEE, 1997.

[257] R. Urtasun, D. J. Fleet, and P. Fua, "3d people tracking with gaussian process dynamical models," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, pp. 238–245, IEEE, 2006.

[258] S.-K. Weng, C.-M. Kuo, and S.-K. Tu, "Video object tracking using adaptive kalman filter," *Journal of Visual Communication and Image Representation*, vol. 17, no. 6, pp. 1190–1208, 2006.

[259] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *Acm computing surveys (CSUR)*, vol. 38, no. 4, p. 13, 2006.

[260] J. F. P. Kooij, N. Schneider, F. Flohr, and D. M. Gavrila, "Context-based pedestrian path prediction," in *Computer Vision–ECCV 2014*, pp. 618–633, Springer, 2014.

[261] H. Kretzschmar, M. Kuderer, and W. Burgard, "Learning to predict trajectories of cooperatively navigating agents," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 4015–4020, IEEE, 2014.

[262] V. Karasev, A. Ayvaci, B. Heisele, and S. Soatto, "Intent-aware long-term prediction of pedestrian motion," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 2543–2549, IEEE, 2016.

[263] E. Rehder and H. Kloeden, "Goal-directed pedestrian prediction," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 50–58, 2015.

[264] Y. Zhou and T. L. Berg, "Temporal perception and prediction in ego-centric video," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4498–4506, 2015.

[265] T. Lan, T.-C. Chen, and S. Savarese, "A hierarchical representation for future action prediction," in *Computer Vision–ECCV 2014*, pp. 689–704, Springer, 2014.

[266] D.-A. Huang and K. M. Kitani, "Action-reaction: Forecasting the dynamics of human interaction," in *Computer Vision–ECCV 2014*, pp. 489–504, Springer, 2014.

[267] D. Fouhey and C. Zitnick, "Predicting object dynamics in scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2019–2026, 2014.

[268] M. Beetz, N. von Hoyningen-Huene, B. Kirchlechner, S. Gedikli, F. Siles, M. Durus, and M. Lames, "Aspogamo: Automated sports game analysis models," *International Journal of Computer Science in Sport*, vol. 8, no. 1, pp. 1–21, 2009.

[269] IEEE, *Tracking multiple people under global appearance constraints*, 2011.

[270] A. Maksai, X. Wang, and P. Fua, "What players do with the ball: A physically constrained interaction modeling," *arXiv preprint arXiv:1511.06181*, 2015.

[271] X. Wei, P. Lucey, S. Morgan, and S. Sridharan, "Predicting shot locations in tennis using spatiotemporal data," in *Digital Image Computing: Techniques and Applications (DICTA), 2013 International Conference on*, pp. 1–8, IEEE, 2013.

[272] A. Bialkowski, P. Lucey, P. Carr, Y. Yue, S. Sridharan, and I. Matthews, "Large-scale analysis of soccer matches using spatiotemporal tracking data," in *Data Mining (ICDM), 2014 IEEE International Conference on*, pp. 725–730, IEEE, 2014.

[273] P. Lucey, A. Bialkowski, P. Carr, S. Morgan, I. Matthews, and Y. Sheikh, "Representing and discovering adversarial team behaviors using player roles," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2706–2713, 2013.

[274] V. Ramanathan, J. Huang, S. Abu-El-Haija, A. Gorban, K. Murphy, and L. Fei-Fei, "Detecting events and key actors in multi-person videos," *arXiv preprint arXiv:1511.02917*, 2015.

[275] X. Wei, P. Lucey, S. Vidas, S. Morgan, and S. Sridharan, "Forecasting events using an augmented hidden conditional random field," in *Computer Vision–ACCV 2014*, pp. 569–582, Springer, 2014.

[276] C. Vondrick, D. Patterson, and D. Ramanan, "Efficiently scaling up crowdsourced video annotation," *International Journal of Computer Vision*, vol. 101, no. 1, pp. 184–204, 2013.

[277] STATS, "https://www.stats.com/sportvu-basketball/," 2015.

[278] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.

[279] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[280] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[281] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, 1989.

[282] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[283] P. Agrawal, D. Stansbury, J. Malik, and J. L. Gallant, "Pixels to voxels: Modeling visual representation in the human brain," *arXiv preprint arXiv:1407.5104*, 2014.

[284] D. J. Felleman and D. C. Van Essen, "Distributed Hierarchical Processing in the Primate Cerebral Cortex," *Cerebral Cortex*, vol. 1, pp. 1–47, Jan. 1991.

[285] D. J. Kravitz, K. S. Saleem, C. I. Baker, L. G. Ungerleider, and M. Mishkin, "The ventral visual pathway: an expanded neural framework for the processing of object quality," *Trends in cognitive sciences*, vol. 17, no. 1, pp. 26–49, 2013.

[286] V. A. F. Lamme and P. R. Roelfaema, "the distinct modes of vision offered by feedforward and recurrent processing.," *Trends in Neurosciences*, vol. 23, p. 571, 2000.

[287] E. Tulving and D. L. Schacter, "Priming and human memory systems," *Science*, vol. 247, no. 4940, pp. 301–306, 1990.

[288] Y. Yang and D. Ramanan, "Articulated pose estimation with flexible mixtures-of-parts," in *CVPR*, 2011.

[289] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 1653–1660, IEEE, 2014.

[290] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 648–656, 2015.

[291] T. Pfister, J. Charles, and A. Zisserman, "Flowing convnets for human pose estimation in videos," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1913–1921, 2015.

[292] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 3686–3693, IEEE, 2014.

[293] S. Johnson and M. Everingham, "Clustered pose and nonlinear appearance models for human pose estimation," in *Proceedings of the British Machine Vision Conference*, 2010. doi:10.5244/C.24.12.

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

[295] A. Vedaldi and K. Lenc, "Matconvnet-convolutional neural networks for matlab," *arXiv preprint arXiv:1412.4564*, 2014.

[296] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele, "Poselet conditioned pictorial structures," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 588–595, IEEE, 2013.

[297] X. Chen and A. L. Yuille, "Articulated pose estimation by a graphical model with image dependent pairwise relations," in *Advances in Neural Information Processing Systems*, pp. 1736–1744, 2014.

[298] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele, "Strong appearance and expressive spatial models for human pose estimation," in *International Conference on Computer Vision (ICCV), 2013 IEEE Conference on*, pp. 3487–3494, IEEE, 2013.

[299] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *Advances in neural information processing systems*, pp. 1799–1807, 2014.

[300] X. Fan, K. Zheng, Y. Lin, and S. Wang, "Combining local appearance and holistic view: Dual-source deep neural networks for human pose estimation," *arXiv preprint arXiv:1504.07159*, 2015.

[301] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[302] Z. Tu, "Auto-context and its application to high-level vision tasks," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, IEEE, 2008.

[303] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *Proceedings of the twenty-first international conference on Machine learning*, p. 104, ACM, 2004.

[304] H. Daumé III, J. Langford, and D. Marcu, "Search-based structured prediction," *Machine learning*, vol. 75, no. 3, pp. 297–325, 2009.

[305] S. Nowozin and C. H. Lampert, "Structured learning and prediction in computer vision," *Foundations and Trends® in Computer Graphics and Vision*, vol. 6, no. 3–4, pp. 185–365, 2011.

[306] L.-C. Chen, A. G. Schwing, A. L. Yuille, and R. Urtasun, "Learning deep structured models," *arXiv preprint arXiv:1407.2538*, 2014.

[307] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep structured output learning for unconstrained text recognition," *ICLR 2015*, 2014.

[308] C. D. Gilbert and M. Sigman, "Brain states: Top-down influences in sensory processing," *Neuron*, vol. 54, no. 5, pp. 677 – 696, 2007.

[309] J. M. Hupe, A. C. James, B. R. Payne, S. G. Lomber, P. Girard, and J. Bullier, "Cortical feedback improves discrimination between figure and background by V1, V2 and V3 neurons," *Nature*, vol. 394, pp. 784–787, Aug. 1998.

[310] D. Wyatte, T. Curran, and R. C. O'Reilly, "The limits of feedforward vision: Recurrent processing promotes robust object recognition when objects are degraded," *J. Cognitive Neuroscience*, pp. 2248–2261, 2012.

[311] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds.), pp. 2204–2212, Curran Associates, Inc., 2014.

[312] M. F. Stollenga, J. Masci, F. Gomez, and J. Schmidhuber, "Deep networks with internal selective attention through feedback connections," in *Advances in Neural Information Processing Systems*, pp. 3545–3553, 2014.

[313] V. Ramakrishna, D. Munoz, M. Hebert, J. A. Bagnell, and Y. Sheikh, "Pose machines: Articulated pose estimation via inference machines," in *Computer Vision–ECCV 2014*, pp. 33–47, Springer International Publishing, 2014.

[314] D. Weiss, B. Sapp, and B. Taskar, "Structured prediction cascades," *arXiv preprint arXiv:1208.3279*, 2012.

[315] Q. Li, J. Wang, Z. Tu, and D. P. Wipf, "Fixed-point model for structured labeling," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 214–221, 2013.

[316] M. Oberweger, P. Wohlhart, and V. Lepetit, "Training a feedback loop for hand pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3316–3324, 2015.

[317] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," *arXiv preprint arXiv:1411.5752*, 2014.

[318] X. Fan, K. Zheng, Y. Lin, and S. Wang, "Combining local appearance and holistic view: Dual-source deep neural networks for human pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1347–1355, 2015.

[319] F. Fleuret and D. Geman, "Stationary features and cat detection," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2549–2578, 2008.

[320] P. Dollár, P. Welinder, and P. Perona, "Cascaded pose regression," in *CVPR*, 2010.

[321] C. Ionescu, J. Carreira, and C. Sminchisescu, "Iterated second-order label sensitive pooling for 3d human pose estimation," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 1661–1668, IEEE, 2014.

[322] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International journal of computer vision*, vol. 1, no. 4, pp. 321–331, 1988.

[323] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models-their training and application," *Computer vision and image understanding*, vol. 61, no. 1, pp. 38–59, 1995.

[324] X. Xiong and F. De la Torre, "Supervised descent method and its applications to face alignment," in *CVPR*, 2013.

[325] G. Shakhnarovich, P. Viola, and T. Darrell, "Fast pose estimation with parameter-sensitive hashing," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 750–757, IEEE, 2003.

[326] L. Sigal, M. Isard, H. Haussecker, and M. J. Black, "Loose-limbed people: Estimating 3D human pose and motion using non-parametric belief propagation," *International Journal of Computer Vision*, vol. 98, pp. 15–48, May 2011.

[327] M. Spiridon, B. Fischl, and N. Kanwisher, "Location and spatial profile of category-specific regions in human extrastriate cortex," *Human Brain Mapping*, vol. 27, no. 1, pp. 77–89, 2006.

[328] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology*, vol. 148, no. 3, pp. 574–591, 1959.

[329] K. N. Kay, T. Naselaris, R. J. Prenger, and J. L. Gallant, "Identifying natural images from human brain activity," *Nature*, vol. 452, no. 7185, pp. 352–355, 2008.

[330] S. Nishimoto, A. T. Vu, T. Naselaris, Y. Benjamini, B. Yu, and J. L. Gallant, "Reconstructing visual experiences from brain activity evoked by natural movies," *Current Biology*, vol. 21, no. 19, pp. 1641–1646, 2011.

[331] N. Kanwisher, J. McDermott, and M. M. Chun, "The fusiform face area: a module in human extrastriate cortex specialized for face perception," *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, vol. 17, no. 11, pp. 4302–4311, 1997.

[332] T. Çukur, A. G. Huth, S. Nishimoto, and J. L. Gallant, "Functional subdomains within human FFA.," *The Journal of neuroscience : the official journal of the Society for Neuroscience*, vol. 33, pp. 16748–66, Oct. 2013.

[333] I. Gauthier, M. J. Tarr, J. Moylan, P. Skudlarski, J. C. Gore, and A. W. Anderson, "The fusiform "face area" is part of a network that processes faces at the individual level," *Journal of Cognitive Neuroscience*, vol. 12, no. 3, pp. 495–504, 2000.

[334] R. Epstein and N. Kanwisher, "A cortical representation of the local visual environment," *Nature*, vol. 392, no. 6676, pp. 598–601, 1998.

[335] A. Pasupathy and C. E. Connor, "Responses to contour features in macaque area v4," *Journal of Neurophysiology*, vol. 82, no. 5, pp. 2490–2502, 1999.

[336] H. B. Barlow, "Possible principles underlying the transformations of sensory messages," 1961.

[337] S. V. David and J. L. Gallant, "Predicting neuronal responses during natural vision," *Network*, vol. 16, no. 2-3, pp. 239–260, 2005.

[338] T. Naselaris, K. N. Kay, S. Nishimoto, and J. L. Gallant, "Encoding and decoding in fMRI," *NeuroImage*, vol. 56, no. 2, pp. 400–410, 2011.

[339] D. Stansbury, T. Naselaris, and J. Gallant, "Natural scene statistics account for the representation of scene categories in human visual cortex.," *Neuron*, vol. 79, pp. 1025–34, Sept. 2013.

[340] A. G. Huth, S. Nishimoto, A. T. Vu, and J. L. Gallant, "A continuous semantic space describes the representation of thousands of object and action categories across the human brain.," *Neuron*, vol. 76, pp. 1210–24, Dec. 2012.

[341] T. M. Mitchell, S. V. Shinkareva, A. Carlson, K.-M. Chang, V. L. Malave, R. A. Mason, and M. A. Just, "Predicting human brain activity associated with the meanings of nouns," *science*, vol. 320, no. 5880, pp. 1191–1195, 2008.

[342] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks.," in *NIPS*, 2012.

[343] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of Physiology*, vol. 195, no. 1, pp. 215–243, 1968.

[344] R. B. Buxton, *Introduction to functional magnetic resonance imaging book pack: Principles and techniques*. Cambridge University Press, 2002.

[345] R. Rajimehr, K. J. Devaney, N. Y. Bilenko, J. C. Young, and R. B. Tootell, "The "parahippocampal place area" responds preferentially to high spatial frequencies in humans and monkeys," *PLoS biology*, vol. 9, no. 4, p. e1000608, 2011.

[346] A. Pasupathy and C. E. Connor, "Population coding of shape in area v4," *Nature neuroscience*, vol. 5, no. 12, pp. 1332–1338, 2002.

[347] J. L. Gallant, C. E. Connor, S. Rakshit, J. W. Lewis, and D. C. Van Essen, "Neural responses to polar, hyperbolic, and cartesian gratings in area v4 of the macaque monkey," *Journal of neurophysiology*, vol. 76, no. 4, pp. 2718–2739, 1996.

[348] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature neuroscience*, vol. 2, no. 11, pp. 1019–1025, 1999.

[349] C. F. Cadieu and B. A. Olshausen, "Learning intermediate-level representations of form and motion from natural movies," *Neural computation*, vol. 24, no. 4, pp. 827–866, 2012.

[350] Y. Karklin and M. S. Lewicki, "Learning higher-order structures in natural images," *Network: Computation in Neural Systems*, vol. 14, no. 3, pp. 483–499, 2003.

[351] S.-M. Khaligh-Razavi and N. Kriegeskorte, "Deep supervised, but not unsupervised, models may explain it cortical representation," *PLoS computational biology*, vol. 10, no. 11, p. e1003915, 2014.

[352] C. F. Cadieu, H. Hong, D. L. Yamins, N. Pinto, D. Ardila, E. A. Solomon, N. J. Majaj, and J. J. DiCarlo, "Deep neural networks rival the representation of primate it cortex for core visual object recognition," *PLoS computational biology*, vol. 10, no. 12, p. e1003963, 2014.

[353] S. Thorpe, D. Fize, C. Marlot, *et al.*, "Speed of processing in the human visual system," *nature*, vol. 381, no. 6582, pp. 520–522, 1996.

[354] U. Güçlü and M. A. van Gerven, "Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream," *The Journal of Neuroscience*, vol. 35, no. 27, pp. 10005–10014, 2015.

[355] R. M. Cichy, A. Khosla, D. Pantazis, A. Torralba, and A. Oliva, "Comparison of deep neural networks to spatio-temporal cortical dynamics of human visual object recognition reveals hierarchical correspondence," *Scientific reports*, vol. 6, p. 27755, 2016.

[356] D. L. Yamins and J. J. DiCarlo, "Using goal-driven deep learning models to understand sensory cortex," *Nature neuroscience*, vol. 19, no. 3, p. 356, 2016.

[357] M. Eickenberg, A. Gramfort, G. Varoquaux, and B. Thirion, "Seeing it all: Convolutional network layers map the function of the human visual system," *NeuroImage*, vol. 152, pp. 184–194, 2017.

[358] A. J. Kell, D. L. Yamins, E. N. Shook, S. V. Norman-Haignere, and J. H. McDermott, "A task-optimized neural network replicates human auditory behavior, predicts brain responses, and reveals a cortical processing hierarchy," *Neuron*, vol. 98, no. 3, pp. 630–644, 2018.