BUILDING AGENTS THAT CAN SEE, TALK, AND ACT

A Dissertation
Presented to
The Academic Faculty

By

Abhishek Das

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Computer Science in the
School of Interactive Computing

Georgia Institute of Technology

May 2020

BUILDING AGENTS THAT CAN SEE, TALK, AND ACT

Committee members:

Dr. Dhruv Batra, Advisor
Georgia Institute of Technology &
Facebook AI Research

Dr. Devi Parikh
Georgia Institute of Technology &
Facebook AI Research

Dr. James Hays
Georgia Institute of Technology &
Argo AI

Dr. Joelle Pineau
McGill University &
Facebook AI Research

Dr. Jitendra Malik
UC Berkeley &
Facebook AI Research

Date Approved: March 9, 2020

Acknowledgements

I have had the time of my life in the past four years as a PhD student and there are several people I must thank for having played pivotal roles towards that experience.

First and foremost, Dhruv Batra, for having taken a chance on me. I did not have a background in machine learning or research, and was at best a half-decent engineer when I started. Over the course of numerous exchanges, Dhruv has groomed me as a researcher, and I have come to learn and appreciate his ability to connect dots between seemingly disparate domains (and realizing that is often where the magic lies), the value of aesthetics and rhetoric in science ('why should *anyone* care about this?'), the importance of an accurate world model (an accurate world model begets accurate predictions), and his unwavering encouragement to deep-dive into bold ideas outside comfort zone. I am grateful to have gotten the opportunity to be his student, and it is in large part my effort to vindicate his decision to advise me that drives me to do the best work I can.

I must also thank Devi Parikh, who I have had the privilege of working closely with. Devi's clarity of thought (and hence, clarity in communication), ability to reason from first principles, and devotion to her time management system – a system I continue to attempt at adopting – are infectious. Her feedback on my ideas, writing, communication, slides, talks have helped shape not only my research but also taught me how to think.

I am also grateful to Dhruv and Devi for fostering a collegial culture in our lab, and taking care of every student. I have benefited immensely from being insulated from funding concerns and having the freedom to pursue the directions I was most passionate about.

During my PhD, I have been fortunate to have had the chance to intern thrice at Facebook AI Research, and once each at DeepMind and Tesla Autopilot. From my time at Facebook AI Research, I would particularly like to thank Joelle Pineau for being generous with her time, patiently walking me through research directions I was eager but mostly confused about, pushing me on the basics of science (strong baselines, reproducible results, high scientific standards) and teaching me how to zoom out and place ideas in wider context, and Georgia Gkioxari for her useful perspectives that helped shape and navigate the tra-

jectory of the embodied question answering project through all its ups and downs. From my time at DeepMind, I would like to thank Felix Hill for many insightful discussions on grounded language and what a path towards solving it might encompass, and for his constant support and optimism, Rosalia Schneider, Josh Abramson, and Tim Harley for their attention to detail and hand-holding me through my seemingly primitive engineering skills, Laura Rimell for introducing me to key ideas in linguistics, and Federico Carnevale for being amazingly meticulous and teaching me lessons in how to organize experiments. It was inspiring to interact with and learn from Greg Wayne and Tim Lillicrap; their taste in big research questions is unparalleled. I am also thankful to Yuxin Wu, Angela Fan, Oleksandr Maksymets, Abhishek Kadian, Lisa Hendricks, Latha Pemula, Mike Rabbat, Theophile Gervet, Joshua Romoff, Amy Zhang, Ahmed Touati, Harsh Satija, Stephen Clark, Tomas Kocisky, Angeliki Lazaridou, Mateusz Malinowski, Aida Nematzadeh, Karl Moritz Hermann, Seb Ruder, Andrew Trask, Chris Dyer, Kris Cao, Phil Blunsom, Dani Yogatama, Gábor Melis, Cyprien de Masson d'Autume, DJ Strouse, Ali Eslami, Adhi Kuncoro, Sumanth Dathathri, Gefen Kohavi, Eugene Vinitsky, Lane McIntosh, and Andrej Karpathy for adding to these wonderfully positive experiences.

Thank you to my thesis committee – Dhruv Batra, Devi Parikh, James Hays, Joelle Pineau, and Jitendra Malik – for insightful feedback and discussions. And also to the generous fellowships from Facebook, Adobe, and Snap Inc. for supporting my research.

I owe special mentions to Andrej Karpathy, Greg Shakhnarovich, John Platt, Jeff Dean, Yoshua Bengio, Felix Hill, Ani Kembhavi, Greg Wayne, Oren Etzioni, Jianfeng Gao, Larry Zitnick, Karl Moritz Hermann for inspiring conversations that had a disproportionate impact on my research and career trajectory. I must also thank Nassim Parvin, with whom I happened to take two courses on design ethics and methods in the final year of my PhD. Her research and teachings have left a lasting imprint on me, and while I cannot stop wishing I had taken her courses earlier in my career, they are sure to influence future journeys I embark on. Thank you!

Over the course of my time at Virginia Tech (where I spent my first semester) and Georgia Tech, I have come to call CVMLP Lab home, and its members, past and present, my family – Ramprasaath Selvaraju, Ramakrishna Vedantam, Arjun Chandrasekaran, Aishwarya Agrawal, Yash Goyal, Harsh Agrawal, Michael Cogswell, Satwik Kottur, Prithvijit Chattopadhyay, Jiasen Lu, Deshraj Yadav, Jianwei Yang, Viraj Prabhu, Erik Wijmans, Nirbhay

Modhe, Arjun Majumdar, Joanne Truong, Akrit Mohapatra, Ayush Shrivastava, Rishabh Jain, Mohit Sharma, Vishvak Murahari, Samyak Datta, Karan Desai, Purva Tendulkar, Avi Singh, Khushi Gupta, Stefan Lee, Peter Anderson, and Zhile Ren. I owe special thanks to Ramprasaath for being my roommate for ∼4 years and consistently cooking the best chicken curry ever, which happens to be surprisingly robust to availability (or lack) of spices, to Harsh for hand-holding me through the VQA-HAT project, to Stefan for his keen sense for aesthetics (the canonical Q-BOT, A-BOT, M-BOT figures are all thanks to him) and being my go-to person for dumb questions early on in my PhD, to Ramakrishna, Aishwarya, Jiasen, and Peter, who with their regular stream of success stories served as role models through grad school, to Michael who is by far the clearest thinker I know, to Erik who has the unique ability to power through massive codebases in no time, and to Satwik for being one of the best co-authors I have had the privilege of working with. To my mentees – Vishvak Murahari, Vivek Vanga, Joel Ye, Sandeep Kumar – thank you for being wonderful collaborators to work with, and I hope you gained as much out of working together as I did. To the Caliper team – Deshraj, Devi, Dhruv – what a fun little experiment it was! To my late-night 8-ball crew – Rishabh, Ayush, Prithvi – I hope you practice hard enough to be able to beat me fair and square some day :) And to my conference crew – Satwik, Dipan Pal, Stefan, Devi, Dhruv – thank you for making conferences unforgettably fun!

I am also grateful to my friends outside of work for being constant pillars of support through all my ups and downs – Rushil Nagda, Vasudha Khurana, Aman Tripathi, Abhishek Nayak, Piyush Bharti, Rishika Sinha, Jayant Jain, Shivam Mangla, Chhavi Sahni, Sidharth Sadani, Shashank Mehta, Chetty Arun, Shubhangi Gupta, Varunprasaath Selvaraju, Keyur Shah, Radhika Pasari, Subu Avadai, Abhilash Kulkarni, Nisha Chandramoorthy, Manisha Jain, Harshil Mathur, Abhay Rana, Shagun Sodhani, and Vikalp Gupta.

To my parents – Saswati Das and Devashish Das – who provided constant love, encouragement and support to pursue my dreams (regardless of how outrageous they sounded), and made significant sacrifices to enable my move to the United States to pursue a PhD.

And finally, to Chandana Rajanna, for everything.

Table of Contents

Acknowledgments v

List of Tables x

List of Figures xvii

Summary 1

Chapter 1: Introduction 2
1.1 Outline 2
1.2 Related Work 4

I Agents That Can See and Talk **7**

Chapter 2: Visual Dialog 8
2.1 Introduction 8
2.2 Related Work 12
2.3 The Visual Dialog Dataset (VisDial) 13
2.4 VisDial Dataset Analysis 15
2.4.1 Analyzing VisDial Questions 15
2.4.2 Analyzing VisDial Answers 17
2.4.3 Analyzing VisDial Dialog 19
2.4.4 VisDial Evaluation Protocol 23
2.5 Neural Visual Dialog Models 24
2.6 Experiments 26
2.7 Conclusions 31

Chapter 3: Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning 32
3.1 Introduction 32
3.2 Related Work 35
3.3 Cooperative Image Guessing Game:
In Full Generality and a Specific Instantiation 36

3.4 Reinforcement Learning for Dialog Agents . . . 37
3.4.1 Policy Networks for Q-BOT and A-BOT . . . 39
3.4.2 Joint Training with Policy Gradients . . . 40
3.5 Emergence of Grounded Dialog . . . 42
3.6 Experiments . . . 43
3.7 Conclusions . . . 48

II Agents That Can See, Talk, and Act 49

Chapter 4: Embodied Question Answering . . . 50
4.1 Introduction . . . 50
4.2 Related Work . . . 53
4.3 EQA Dataset: Questions In Environments . . . 55
4.3.1 House3D: Interactive 3D Environments . . . 55
4.3.2 Question-Answer Generation . . . 56
4.4 A Hierarchical Model for EmbodiedQA . . . 59
4.4.1 Imitation Learning and Reward Shaping . . . 61
4.5 Experiments and Results . . . 62
4.6 Conclusion . . . 65

Chapter 5: Neural Modular Control for Embodied Question Answering . . . 66
5.1 Introduction . . . 66
5.2 Related Work . . . 68
5.3 Neural Modular Control . . . 69
5.3.1 Hierarchical Policy . . . 70
5.3.2 Hierarchical Behavior Cloning from Expert Trajectories . . . 72
5.3.3 Asynchronous Advantage Actor-Critic (A3C) Training . . . 74
5.4 Experiments and Results . . . 76
5.5 Conclusion . . . 78

III Multi-Agent Populations That Can See, Talk, and Act 80

Chapter 6: TarMAC: Targeted Multi-Agent Communication . . . 81
6.1 Introduction . . . 81
6.2 Related Work . . . 83
6.3 Technical Background . . . 84
6.4 TarMAC: Targeted Multi-Agent Communication . . . 85
6.5 Experiments . . . 88
6.5.1 SHAPES . . . 88
6.5.2 Traffic Junction . . . 90

6.5.3 House3D . . . 92
6.6 Conclusions and Future Work . . . 93

IV Language as a Task-Agnostic Probe of World Knowledge 94

Chapter 7: Probing Emergent Semantics in Predictive Agents via Question Answering . . . 95
7.1 Introduction . . . 95
7.2 Background and related work . . . 98
7.3 Environment & Tasks . . . 100
7.4 Approach . . . 100
7.4.1 Auxiliary Predictive Losses . . . 102
7.5 Experiments & Results . . . 104
7.5.1 Question-Answering Performance . . . 104
7.5.2 Compositional Generalization . . . 106
7.5.3 Robustness of the results . . . 107
7.6 Discussion . . . 107

Chapter 8: Conclusion . . . 109

List of Publications . . . 111

Appendix A: Embodied Question Answering . . . 115
A.1 Question-Answer Generation Engine . . . 116
A.2 CNN Training Details . . . 119
A.3 Question Answering Module . . . 121
A.4 Human Navigation + Machine QA . . . 122

Appendix B: Probing Emergent Semantics in Predictive Agents via Question Answering . . . 129
B.1 Network architectures and Training setup . . . 129
B.1.1 Importance Weighted Actor-Learner Architecture . . . 129
B.1.2 Agents . . . 129
B.1.3 QA network architecture . . . 130
B.1.4 Hyper-parameters . . . 130
B.1.5 Negative sampling strategies for CPC|A . . . 131
B.2 Effect of QA network depth . . . 131
B.3 Answering accuracy during training for all questions . . . 132
B.4 Environment . . . 132

Bibliography . . . 153

List of Tables

2.1 Comparison of existing image question answering datasets with VisDial . . 18

2.2 Comparison of sequences in VisDial, VQA, and Cornell Movie-Dialogs corpus in their original ordering *vs*. permuted 'shuffled' ordering. Lower is better for perplexity while higher is better for classification accuracy. *Left:* Cornell corpus has the highest absolute increase in perplexity followed by VisDial and VQA, which indicates the degree of linguistic sequential structure in these datasets. *Right:* The classifier on VisDial achieves the highest accuracy, followed by Cornell, thus indicating a strong temporal continuity in the conversation. 22

2.3 Performance of methods on VisDial v0.9, measured by mean reciprocal rank (MRR), recall@k and mean rank. Higher is better for MRR and recall@k, lower is better for mean rank. 29

2.4 Human-machine performance comparison on VisDial v0.5, measured by mean reciprocal rank (MRR), recall@k for $k = \{1, 5\}$ and mean rank. Note that higher is better for MRR and recall@k, while lower is better for mean rank. 30

3.1 Selected examples of Q-BOT-A-BOT interactions for `SL-pretrained` and `RL-full-QAf`. `RL-full-QAf` interactions are diverse, less prone to repetitive and safe exchanges ("can't tell", "don't know" *etc*.), and more image-discriminative. . . 44

4.1 Quantitative evaluation of EmbodiedQA agents on navigation and answering metrics for the EQA v1 test set. Ill-defined cells are marked with '-' because 1) reactive models don't have a stopping action, 2) humans pick a single answer from a drop-down list, so mean rank is not defined, 3) most distance metrics are trivially defined for shortest paths since they always end at the target object by design. 64

5.1 Subgoals and conditions used to automatically extract them from expert trajectories. . . . 71

5.2 Evaluation of EmbodiedQA agents on navigation and answering metrics for the EQA v1 test set. . . . 77

6.1 Comparison with prior work on cooperative multi-agent continuous communication. . . . 84

6.2 Success rates on 3 different settings of cooperative navigation in the SHAPES environment. . . . 88

6.3 Success rates on traffic junction. Our targeted 2-stage communication architecture gets a success rate of 97.1% on the 'hard' variant of the task, significantly outperforming [1]. Note that 1- and 2-stage refer to the number of rounds of communication between actions ((6.4)). . . . 90

6.4 Success rates on a 4-agent cooperative `find[fireplace]` navigation task in House3D. . . . 93

7.1 QA task templates. In every episode, objects and their configurations are randomly generated, and these templates get translated to QA pairs for all unambiguous <`shape, color`> combinations. There are 50 `shapes` and 10 `colors` in total. See B.4 for details. . . . 98

7.2 Top-1 accuracy on question-answering tasks. . . . 103

A.1 Functional forms of all question types in the EQA dataset . . . 115

A.2 (L-R) Quantitative results for the segmentation, depth estimation, and autoencoder heads of our multi-task perception network. All metrics are reported on a held out validation set. . . . 118

B.1 Hyperparameters. . . . 134

B.2 Randomization of objects in the Unity room. 50 different types, 10 different colors and 3 different scales. . . . 136

B.3 Environment action set. . . . 136

List of Figures

1.1 Previous work on agents that can see, talk, and/or act can be arranged along the axes of vision (from a single-frame to video), language (from captioning and single-shot question answering to dialog), and actions (from passive observers to active agents). 5

2.1 We introduce a new AI task – Visual Dialog, where an AI agent must hold a dialog with a human about visual content. We introduce a large-scale dataset (VisDial), an evaluation protocol, and novel encoder-decoder models for this task. 9

2.2 Differences between image captioning, Visual Question Answering (VQA) and Visual Dialog. Two (partial) dialogs are shown from our VisDial dataset, which is curated from a live chat between two Amazon Mechanical Turk workers (Sec. 2.3). 10

2.3 Collecting visually-grounded dialog data on Amazon Mechanical Turk via a live chat interface where one person is assigned the role of 'questioner' and the second person is the 'answerer'. We show the first two questions being collected via the interface as Turkers interact with each other in Fig. 2.3a and Fig. 2.3b. Remaining questions are shown in Fig. 2.3c. 11

2.4 Detailed instructions for Amazon Mechanical Turkers on our interface . . . 13

2.5 Examples from VisDial 13

2.6 Distribution of lengths for questions and answers (2.6a); and percent coverage of unique answers over all answers from the train dataset (2.6b), compared to VQA. For a given coverage, VisDial has more unique answers indicating greater answer diversity. 16

2.7 17

2.8 Distribution of first n-grams for (left to right) VQA questions, VisDial questions and VisDial answers. Word ordering starts towards the center and radiates outwards, and arc length is proportional to number of questions containing the word. 18

2.10 Most frequent answer responses except for 'yes'/'no' 20

2.11 Architecture of HRE encoder with attention. At the current round R_t, the model attends to relevant history from previous rounds, based on the current question. This attention-over-history feeds into a dialog-RNN along with question to generate joint representation E_t for the decoder. 27

2.12 Memory Network (MN) encoder 28

3.1 A cooperative image guessing game between two agents – Q-BOT and A-BOT– who communicate through a natural language dialog so that Q-BOT can select a particular unseen image from a lineup. We model these agents as deep neural networks and train them end-to-end with reinforcement learning. 33

3.2 Policy networks for Q-BOT and A-BOT. At each round t of dialog, (1) Q-BOT generates a question q_t from its question decoder conditioned on its state encoding S^Q_{t-1}, (2) A-BOT encodes q_t, updates its state encoding S^A_t, and generates an answer a_t, (3) both encode the completed exchange as F^Q_t and F^A_t, and (4) Q-BOT updates its state to S^Q_t, predicts an image representation $\hat{y}_t$, and receives a reward. 37

3.3 Emergence of grounded dialog: (a) Each 'image' has three attributes, and there are six tasks for Q-BOT (ordered pairs of attributes). (b) Both agents interact for two rounds followed by attribute pair prediction by Q-BOT. (c) Example 2-round dialog where grounding emerges: *color*, *shape*, *style* have been encoded as X, Y, Z respectively. (d) Improvement in reward while policy learning. 41

3.4 **a) Guessing Game Evaluation.** Plot shows the rank in percentile (higher is better) of the 'ground truth' image (shown to A-BOT) as retrieved using fc7 predictions of Q-BOT *vs*. rounds of dialog. Round 0 corresponds to image guessing based on the caption alone. We can see that the RL-full-QAf bots significantly outperforms the SL-pretrained bots (and other ablations). Error bars show standard error of means. **(c)** shows qualitative results on this predicted fc7-based image retrieval. Left column shows true image and caption, right column shows dialog exchange, and a list of images sorted by their distance to the ground-truth image. The image predicted by Q-BOT is highlighted in red. We can see that the predicted image is often semantically quite similar. **b) VisDial Evaluation.** Performance of A-BOT on VisDial v0.5 test, under mean reciprocal rank (MRR), recall@k for $k = \{5, 10\}$ and mean rank metrics. Higher is better for MRR and recall@k, while lower is better for mean rank. We see that our proposed Frozen-Q-multi outperforms all other models on VisDial metrics by 3% relative gain. This improvement is entirely 'for free' since no additional annotations were required for RL. 46

4.1 Embodied Question Answering – EmbodiedQA– tasks agents with navigating rich 3D environments in order to answer questions. These embodied agents must jointly learn language understanding, visual reasoning, and navigation to succeed. 51

4.2 We place our work in context by arranging prior work along the axes of vision (from a single-frame to video), language (from single-shot question answering to dialog), and action (from passive observers to active agents). When viewed from this perspective, EmbodiedQA presents a novel problem configuration – single-shot QA about videos captured by goal-driven active agents. We refer the reader to Section 1.2 for an overview of prior work along various 2D slices in this space. 54

4.3 The EQA dataset is built on a subset of the environments and objects from the SUNCG [2] dataset. We show (a) sample environments and the (b) rooms and (c) objects that are asked about in the EmbodiedQA task. 55

4.4 Overview of the EQA v1 dataset including dataset split statistics (left) and question type breakdown (right). 59

4.5 Our Adaptive Computation Time (ACT) navigator splits the navigation task between a planner and a controller module. The planner selects actions and the controller decides to continue performing that action for a variable number of time steps – resulting in a decoupling of direction (*'turn left'*) and velocity (*'5 times'*) and strengthening the long-term gradient flows of the planner module. 59

4.6 Sample trajectories from ACT+Q-RL agent projected on a floor plan (white areas are unoccupiable) and on-path egocentric views. The agent moves closer to already visible objects – potentially improving its perception of the objects. Note that the floor plan is shown only for illustration and not available to the agents. 63

5.1 We introduce a hierarchical policy for Embodied Question Answering. Given a question ("What color is the sofa in the living room?") and observation, our master policy predicts a sequence of subgoals – `Exit-room`, `Find-room[living]`, `Find-object[sofa]`, `Answer` – that are then executed by specialized sub-policies to navigate to the target object and answer the question ("Grey"). 67

5.2 We extract expert subgoal trajectories from shortest paths by dividing paths on room transition boundaries (circled in (a))and following the rules in Tab. 5.1. 73

5.3 (a,b,c) Success rate over training iterations for each sub-policy task using behavior cloning (BC), reinforcement learning from scratch (A3C), and reinforcement finetuning after behavior cloning (BC+A3C) training regimes. We find BC+A3C significantly outperforms either BC or A3C alone. Each of these is averaged over 5 runs. (d) Losses for master policy during behavior cloning *i.e.* assuming access to perfect sub-policies. 76

6.1 Overview of our multi-agent architecture with targeted communication. Left: At every timestep, each agent policy gets a local observation ω_i^t and aggregated message c_i^t as input, and predicts an environment action a_i^t and a targeted communication message m_i^t. Right: Targeted communication between agents is implemented as a signature-based soft attention mechanism. Each agent broadcasts a message m_i^t consisting of a signature k_i^t, which can be used to encode agent-specific information and a value v_i^t, which contains the actual message. At the next timestep, each receiving agent gets as input a convex combination of message values, where the attention weights are obtained by a dot product between sender's signature k_i^t and a query vector q_j^{t+1} predicted from the receiver's hidden state. 86

6.2 Visualizations of learned targeted communication in SHAPES. Best viewed in color. 89

6.3 Success rates for 1 *vs*. 2-stage *vs*. message size on Hard. Performance does not decrease significantly even when the message vector is a single scalar, and 2 rounds of back-and-forth communication before taking an environment action leads to a significant improvement over 1-stage. 90

6.4 Results on the traffic junction environment. 91

6.5 Agents navigating to the `fireplace` in House3D (marked in yellow). Note in particular that agent 4 is spawned facing away from it. It communicates with others, turns to face the `fireplace`, and moves towards it. 93

7.1 We train predictive agents to explore a visually-rich 3D environment with an assortment of objects of different shapes, colors and sizes. As the agent navigates (trajectory shown in white on the top-down map), an auxiliary network learns to simulate representations of future observations (labeled 'Simulation Network') k steps into the future, self-supervised by a loss against the ground-truth egocentric observation at $t + k$. Simultaneously, another decoder network is trained to extract answers to a variety of questions about the environment, conditioned on the agent's internal state but without affecting it (notice 'stop gradient' – gradients from the QA decoder are not backpropagated into the agent). We use this question-answering paradigm to decode and understand the internal representations that such agents develop. Note that the top-down map is only shown for illustration and not available to the agent. 96

7.2 Approach: at every timestep t, the agent receives an RGB observation x_t as input, processes it using a convolutional neural network to produce z_t, which is then processed by an LSTM to select action a_t. The agent learns to explore – it receives a reward of 1.0 for navigating to each new object. As it explores the environment, it builds up an internal representation h_t, which receives pressure from an auxiliary predictive module to capture environment semantics so as to accurately predict consequences of its actions multiple steps into the future. We experiment with a vanilla LSTM agent and two recent predictive approaches – CPC|A [3] and SimCore [4]. The internal representations are then probed via a question-answering decoder whose gradients are not backpropagated into the agent. The QA decoder is an LSTM initialized with h_t and receiving the question at every timestep. 101

7.3 L – Reward in an episode. R – Top-1 QA accuracy. Averaged over 3 seeds. Shaded region is 1 SD. 103

7.4 (Left): Sample trajectory ($1 \rightarrow 4$) and QA decoding predictions (for top 5 most probable answers) for the 'What shape is the green object?' from SimCore. Note that top-down map is not available to the agent. (Right): QA accuracy on disjoint train and test splits. 104

7.5 (Left) DeepMind Lab environment [5]: Rectangular-shaped room with 6 randomly selected objects out of a pool of 20 different objects of different colors. (Right) QA accuracy for `color` questions (What is the color of the <`shape`>?) in DeepMind Lab. Consistent with results in the main paper, internal representations of the SimCore agent lead to the highest accuracy while CPC|A and LSTM perform worse and similar to each other. 105

A.1 Some qualitative results from the hybrid CNN. Each row represents an input image. For every input RGB image, we show the reconstruction from the autoencoder head, ground truth depth, predicted depth as well as ground truth segmentation and predicted segmentation maps. 120

A.2 Conditioned on the navigation frames and question, the question answering module computes dot product attention over the last five frames, and combines attention-weighted combination of image features with question encoding to predict the answer. 121

A.3 Visualizations of queryable objects from the House3D renderer. Notice that instances within the same class differ significantly in shape, size, and color. 123

A.4 Visualizations of queryable rooms from the House3D renderer. 124

A.5 Examples of last five frames from human navigation vs. shortest path. . . . 125

A.6 The answer distribution for `location` template questions. Each bar represents a question of the form *'what room is the <OBJ> located in?'* and shows a distribution over the answers across different environments. The blank spaces in A.6b represent the questions that get pruned out as a result of the entropy+count based filtering. 126

A.7 The answer distribution for `preposition` template questions. Each bar represents a question of the form *'what is next to the <OBJ>?'* and shows a distribution over the answers across different environments. The blank spaces in A.7b represent the questions that get pruned out as a result of the entropy+count based filtering. 127

A.8 The answer distribution for `color` template questions. Each bar represents a question of the form *'what color is the <OBJ>?'* and shows a distribution over the answers across different environments (the color of each section on a bar denotes the possible answers). The blank spaces in A.8b represent the questions that get pruned out as a result of the entropy+count based filtering. 128

B.1 131

B.2 Answer accuracy over training for increasing QA decoder's depths. Left subplot shows the results for the SimCore agent and right subplot for the LSTM baseline. For SimCore, the QA accuracy increases with the decoder depth, up to 12 layers. For the LSTM agent, QA accuracy is not better than chance regardless of the capacity of the QA network. 132

B.3 QA accuracy over training for all questions and all models. 135

Summary

A long-term goal in AI is to build general-purpose intelligent agents that simultaneously possess the ability to *perceive* the rich visual environment around us (through vision, audition, or other sensors), *reason* and infer from perception in an interpretable and actionable manner, *communicate* this understanding to humans and other agents (*e.g.*, hold a natural language dialog grounded in the environment), and *act* on this understanding in physical worlds (*e.g.*, aid humans by executing commands in an embodied environment). To be able to make progress towards this grand goal, we must explore new multimodal AI tasks, move from datasets to physical environments, and build new kinds of models.

In this dissertation, we combine insights from different areas of AI – computer vision, language understanding, reinforcement learning – and present steps to connect the underlying domains of vision and language to actions towards such general-purpose agents.

In Part I, we develop agents that can *see* and *talk* – capable of holding free-form conversations about images – and reinforcement learning-based algorithms to train these visual dialog agents via self-play. In Part II, we extend our focus to agents that can *see*, *talk*, and *act* – embodied agents that can actively perceive and navigate in partially-observable simulated environments, to accomplish tasks such as question-answering. In Part III, we devise techniques for training populations of agents that can communicate with each other, to coordinate, strategize, and utilize their combined sensory experiences and act in the physical world. These agents learn both what messages to send and who to communicate with, solely from downstream reward without any communication supervision. Finally, in Part IV, we use question-answering as a task-agnostic probe to ask a self-supervised embodied agent what it knows about its physical world, and use it to quantify differences in visual representations agents develop when trained with different auxiliary objectives.

Chapter 1

Introduction

1.1 Outline

A long-term goal in AI is to build general-purpose intelligent agents that simultaneously possess the ability to *perceive* the rich visual environment around us (through vision, audition, or other sensors), *reason* and infer from perception in an interpretable and actionable manner, *communicate* this understanding to humans and other agents (*i.e.*, hold a natural language dialog grounded in the environment), and *act* on this understanding in physical worlds (*e.g.*, aid humans by executing API calls or commands in a virtual or embodied environment). In addition to being a fundamental scientific goal in artificial intelligence (AI), even a small advance towards such intelligent systems can *fundamentally change our lives* – from assistive chatbots for the visually impaired, to educational tools for children, to natural language interaction with self-driving cars and in-home physical mobile robots!

Towards this grand goal, in this dissertation, we combine insights from different areas of AI – computer vision, language understanding, reinforcement learning – and develop tasks and techniques that span and connect the domains of vision and language to actions.

In Part I, we develop agents that can *see* and *talk*. Chapter 2 introduces the task of Visual Dialog, which requires an AI agent to hold a meaningful dialog with humans in natural, conversational language about visual content. Specifically, given an image, a dialog history, and a question about the image, the agent has to ground the question in image, infer context from history, and answer the question accurately. Visual Dialog is disentangled enough from a specific downstream task so as to serve as a general test of machine intelligence, while being sufficiently grounded in vision to allow objective evaluation of individual responses and benchmark progress. We develop a novel two-person chat data-collection protocol to curate a large-scale Visual Dialog dataset (VisDial). VisDial consists of ∼1.2M dialog question-answer pairs from 10-round, human-human dialogs grounded in ∼120k images from COCO. We then introduce a family of neural encoder-

decoder models for Visual Dialog with 3 encoders – Late Fusion, Hierarchical Recurrent Encoder and Memory Network – and 2 decoders (generative and discriminative), which outperform a number of sophisticated baselines. We propose a retrieval-based evaluation protocol for Visual Dialog where the AI agent is asked to sort a set of candidate answers and evaluated on metrics such as mean-reciprocal-rank of human response. Putting it all together, we demonstrate the first 'visual chatbot'! Then in Chapter 3, we explore algorithms to train these visual dialog agents without exhaustively collecting human-annotated datasets (via simulation or self-play). Specifically, we pose a cooperative 'image guessing' game between two agents – Q-BOT and A-BOT– who communicate in natural language dialog so that Q-BOT can select an unseen image from a lineup of images. We use deep reinforcement learning (RL) to learn the policies of these agents end-to-end – from pixels to multi-agent multi-round dialog to game reward. We demonstrate two experimental results. First, as a 'sanity check' demonstration of pure RL (from scratch), we show results on a synthetic world, where the agents communicate in ungrounded vocabulary, *i.e.*, symbols with no pre-specified meanings (X, Y, Z). We find that two bots *invent their own communication protocol* and start using certain symbols to ask/answer about certain visual attributes (shape/color/style). Thus, we demonstrate the *emergence of grounded language* among 'visual' dialog agents with no human supervision. Second, we conduct large-scale real-image experiments on the VisDial dataset, where we pretrain with supervised dialog data and show that the RL 'fine-tuned' agents significantly outperform SL agents. Interestingly, the RL Q-BOT learns to ask questions that A-BOT is good at, ultimately resulting in more informative dialog and a better team.

In Part II, we extend our focus to agents that can *see*, *talk*, and *act* – embodied agents that can actively perceive and navigate in partially-observable simulated environments, to accomplish tasks such as question-answering. Specifically, in Chapter 4, we introduce the task of EmbodiedQA, release a dataset of programmatically-generated visual questions and answers grounded in 3D environments, evaluation metrics, and develop a hierarchical architecture for navigation and question-answering in these environments trained with imitation and reinforcement learning. And then in Chapter 5, we present a modular approach for learning policies over long planning horizons for EmbodiedQA. Our policy operates at multiple timescales, where the higher-level master policy proposes subgoals to be executed by specialized sub-policies. Our choice of subgoals is compositional and semantic, *i.e.* they can be sequentially combined in arbitrary orderings, and assume human-interpretable descriptions (*e.g.* 'exit room', 'find kitchen', 'find refrigerator', *etc.*).

In Part III, we devise techniques for training populations of agents that can comunicate with each other, to coordinate, strategize, and utilize their combined sensory experiences and act in the physical world. Specifically, we develop TarMAC, an architecture for targeted multi-agent communication, where agents learn both what messages to send and who to communicate with, solely from downstream task-specific reward without any communication supervision, which is then evaluated on a diverse set of cooperative multi-agent navigation tasks, of varying difficulties, with varying number of agents, in a variety of environments ranging from 2D grids of shapes and simulated traffic junctions to complex 3D indoor environments.

In Part IV, we propose question-answering as a *task-agnostic* probe to ask a self-supervised embodied agent what it knows about its physical world. We apply our method to two recent approaches to predictive modeling – action-conditional CPC [3] and SimCore [4]. After training agents with these predictive objectives in a visually-rich, 3D environment with an assortment of objects, colors, shapes, and spatial configurations, we probe their internal state representations with synthetic (English) questions, without backpropagating gradients from the question-answering decoder into the agent (unlike EmbodiedQA, where agents are trained end-to-end to answer questions). The performance of different agents when probed this way reveals that they learn to encode factual, and seemingly compositional, information about objects, properties and spatial relations from their physical environment. Our approach is intuitive, *i.e*. humans can easily interpret responses of the model as opposed to inspecting continuous vectors, and model-agnostic, *i.e*. applicable to any modeling approach. By revealing the implicit knowledge of objects, quantities, properties and relations acquired by agents as they learn, *question-conditional agent probing* can stimulate the development of stronger predictive learning objectives.

1.2 Related Work

Vision + Language: VQA → Visual Dialog. A number of problems at the intersection of vision and language have gained prominence – image captioning [6–9], video/-movie description [10–12], text-to-image coreference/grounding [13–18], visual storytelling [19, 20], and of course, visual question answering (VQA) [21–34]. However, all of these involve (at most) a single-shot natural language interaction – there is no dialog. More recently, and concurrent with our work in Part I, Vries *et al*. [35] and Mostafazadeh *et al*. [36] have also begun studying visually-grounded dialog.

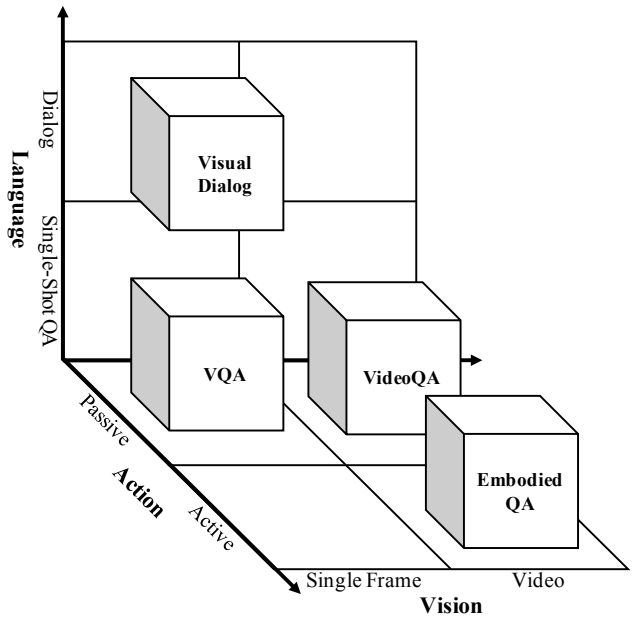


Figure 1.1: Previous work on agents that can see, talk, and/or act can be arranged along the axes of vision (from a single-frame to video), language (from captioning and single-shot question answering to dialog), and actions (from passive observers to active agents).

Visual Dialog is the visual analogue of text-based dialog and conversation modeling. While some of the earliest developed chatbots were rule-based [37], end-to-end learning based approaches are now being actively explored [38–44]. A recent large-scale conversation dataset is the Ubuntu Dialogue Corpus [45], which contains about 500K dialogs extracted from the Ubuntu channel on Internet Relay Chat (IRC). Liu *et al*. [46] perform a study of problems in existing evaluation protocols for free-form dialog. One important difference between free-form textual dialog and VisDial is that in VisDial, the two participants are not symmetric – one person (the 'questioner') asks questions about an image *that they do not see*; the other person (the 'answerer') sees the image and only answers the questions (in otherwise unconstrained text, but no counter-questions allowed). This role assignment gives a sense of purpose to the interaction (why are we talking? To help the questioner build a mental model of the image), and allows objective evaluation of responses.

Vision + Language → Action: Embodied Question Answering. Although aforementioned image and video question answering and dialog tasks require reasoning about natural language questions posed about visual content, a crucial limitation is the *lack of control* – these tasks present answering agents with a fixed view of the environment (*i.e*. one or more images from some fixed trajectory through the world) from which the agent must answer the question, never allowing the agents to *actively* perceive (what if I can't

answer a question from my current view?). In contrast, in our work on Embodied Question Answering in Part II, agents control their trajectory and fate, for good or ill. The task is significantly harder than VQA (*i.e.* most random paths are useless) but the agent has the flexibility to avoid confusing viewpoints and seek informative visual input.

Vision + Action: Visual Navigation. The problem of navigating in an environment based on visual perception has long been studied in vision and robotics (see [47] for an extensive survey). Classical techniques divide navigation into two distinct phases – mapping (where visual observations are used to construct a 3D model of the environment), and planning (which selects paths based on this map). Recent developments in deep RL have proposed fused architectures that go directly from egocentric visual observations to navigational actions [48–54]. The key distinction between these and our work in Part II is how the goals are specified. Visual navigation typically specifies agent goals either implicitly via the reward function [50, 51] (thus training a separate policy for each goal/reward), or explicitly by conditioning on goal state representations [55] including images of target objects [49]. In contrast, EmbodiedQA specifies agent goals via language, which is inherently compositional and renders training a separate policy for every question infeasible.

Language + Action: Situated Language Learning. Inspired by the classical work of Winograd [56], a number of recent works have revisited grounded language learning by situating agents in simple globally-perceived environments and tasking them with goals specified in natural language. The form and structure of these goal specifications range from declarative programs [57], to simple templated commands [58, 59], to free-form natural language instructions [60, 61].

Vision + Language + Action: Embodiment. Most relevant to Part II are recent works that extend the situated language learning paradigm to settings where agents' perceptions are local, purely visual, and change based on their actions – a setting we refer to as embodied language learning.

In concurrent work to ours, Hermann *et al*. [52] and Chaplot *et al*. [48] both develop embodied agents in simple game-like environments consisting of 1-2 rooms and a handful of objects with variable color and shape. In both settings, agents were able to learn to understand simple *'go to X'*/*'pick up X'* style commands where *X* would specify an object (and possibly some of its attributes). Similarly, Oh *et al*. [54] present embodied agents in a simple maze-world and task them to complete a series of instructions.

Part I

Agents That Can See and Talk

Chapter 2

Visual Dialog

2.1 Introduction

We believe that the next generation of visual intelligence systems will need to posses the ability to hold a meaningful dialog with humans in natural language about visual content. Applications:

- Aiding visually impaired users in understanding their surroundings [62] or social media content [63] (AI: *'John just uploaded a picture from his vacation in Hawaii'*, Human: *'Great, is he at the beach?'*, AI: *'No, on a mountain'*).
- Aiding analysts in making decisions from large volumes of surveillance data (Human: *'Did anyone enter this room last week?'*, AI: *'Yes, 27 instances logged on camera'*, Human: *'Were any of them carrying a black bag?'*),
- Interacting with an AI assistant (Human: *'Alexa – can you see the baby in the baby monitor?'*, AI: *'Yes, I can'*, Human: *'Is he sleeping or playing?'*).
- Robotics applications where the operator may be 'situationally blind' and operating via language [64] (Human: *'Is there smoke in any room around you?'*, AI: *'Yes, in one room'*, Human: *'Go there and look for people'*).

Despite rapid progress at the intersection of vision and language – in particular, in image captioning and visual question answering (VQA) – it is clear that we are far from this grand goal of an AI agent that can 'see' and 'communicate'. In captioning, the human-machine interaction consists of the machine simply *talking at* the human (*'Two people are in a wheelchair and one is holding a racket'*), with no dialog or input from the human. While VQA takes a significant step towards human-machine interaction, it still represents only *a single round of a dialog* – unlike in human conversations, there is no scope for follow-up questions, no memory in the system of previous questions asked by the user nor consistency with respect to previous answers provided by the system (Q: *'How many people on wheelchairs?'*, A: *'Two'*; Q: *'How many wheelchairs?'*, A: *'One'*).

As a step towards conversational visual AI, we introduce a novel task – **Visual Dialog** – along with a large-scale dataset, an evaluation protocol, and novel deep models.

Figure 2.1: We introduce a new AI task – Visual Dialog, where an AI agent must hold a dialog with a human about visual content. We introduce a large-scale dataset (VisDial), an evaluation protocol, and novel encoder-decoder models for this task.

Task Definition. The concrete task in Visual Dialog is the following – given an image I, a history of a dialog consisting of a sequence of question-answer pairs (Q1: *'How many people are in wheelchairs?'*, A1: *'Two'*, Q2: *'What are their genders?'*, A2: *'One male and one female'*), and a natural language follow-up question (Q3: *'Which one is holding a racket?'*), the task for the machine is to answer the question in free-form natural language (A3: *'The woman'*). This task is the visual analogue of the Turing Test.

Consider the Visual Dialog examples in Fig. 2.2. The question *'What is the gender of the one in the white shirt?'* requires the machine to selectively focus and direct attention to a relevant region. *'What is she doing?'* requires co-reference resolution (whom does the pronoun 'she' refer to?), *'Is that a man to her right?'* further requires the machine to have visual memory (which object in the image were we talking about?). Such systems also need to be consistent with their outputs – *'How many people are in wheelchairs?'*, *'Two'*, *'What are their genders?'*, *'One male and one female'* – note that the number of genders being specified should add up to two. Such difficulties make the proposed problem a highly interesting and challenging one.

Why do we talk to machines? Prior work in language-only (non-visual) dialog can be arranged on a spectrum with the following two end-points:

goal-driven dialog (*e.g.* booking a flight for a user)

$$\updownarrow$$

goal-free dialog (or casual 'chit-chat' with chatbots)

The two ends have vastly differing purposes and conflicting evaluation criteria. Goal-driven dialog is typically evaluated on task-completion rate (how frequently was the user

VQA

Q: How many people on wheelchairs ?
A: Two

Q: How many wheelchairs ?
A: One

Captioning

Two people are in a wheelchair and one is holding a racket.

Visual Dialog

Q: How many people are on wheelchairs ?
A: Two
Q: What are their genders ?
A: One male and one female
Q: Which one is holding a racket ?
A: The woman

Visual Dialog

Q: What is the gender of the one in the white shirt ?
A: She is a woman
Q: What is she doing ?
A: Playing a Wii game
Q: Is that a man to her right
A: No, it's a woman

Figure 2.2: Differences between image captioning, Visual Question Answering (VQA) and Visual Dialog. Two (partial) dialogs are shown from our VisDial dataset, which is curated from a live chat between two Amazon Mechanical Turk workers (Sec. 2.3).

able to book their flight) or time to task completion [41,65] – clearly, the shorter the dialog the better. In contrast, for chit-chat, the longer the user engagement and interaction, the better. For instance, the goal of the 2017 $2.5 Million Amazon Alexa Prize is to "create a socialbot that converses coherently and engagingly with humans on popular topics for 20 minutes."

We believe our instantiation of Visual Dialog hits a sweet spot on this spectrum. It is *disentangled enough* from a specific downstream task so as to serve as a general test of machine intelligence, while being *grounded enough* in vision to allow objective evaluation of individual responses and benchmark progress. The former discourages task-engineered bots for 'slot filling' [66] and the latter discourages bots that put on a personality to avoid answering questions while keeping the user engaged [37].

Contributions. We make the following contributions:

- We propose a new task: Visual Dialog, where a machine must hold dialog with a human about visual content.
- We develop a novel two-person chat data-collection protocol to curate a large-scale Visual Dialog dataset (VisDial). Upon completion[1], VisDial will contain 1 dialog

[1]VisDial data on COCO-`train` (~83k images) and COCO-`val` (~40k images) is already available for download at visualdialog.org. Since dialog history contains the ground-truth caption, we will not be collecting dialog data on COCO-`test`. Instead, we will collect dialog data on 20k extra images from COCO distribution (which will be provided to us by the COCO team) for our `test` set.

(a) What the 'questioner' sees. (b) What the 'answerer' sees. (c) Example dialog from VisDial dataset.

Figure 2.3: Collecting visually-grounded dialog data on Amazon Mechanical Turk via a live chat interface where one person is assigned the role of 'questioner' and the second person is the 'answerer'. We show the first two questions being collected via the interface as Turkers interact with each other in Fig. 2.3a and Fig. 2.3b. Remaining questions are shown in Fig. 2.3c.

each (with 10 question-answer pairs) on ~140k images from the COCO dataset [67], for a total of ~1.4M dialog question-answer pairs.

- We introduce a family of neural encoder-decoder models for Visual Dialog
  - Late Fusion: that embeds the image, history, and question into vector spaces separately and performs a 'late fusion' of these into a joint embedding.
  - Hierarchical Recurrent Encoder: that contains a dialog-level Recurrent Neural Network (RNN) sitting on top of a question-answer (QA)-level recurrent block. In each QA-level recurrent block, we also include an attention-over-history mechanism to choose and attend to the round of the history relevant to the current question.
  - Memory Network: that treats each previous QA pair as a 'fact' in its memory and learns to 'poll' the stored facts and the image to develop a context vector.

  We train all these encoders with 2 decoders (generative and discriminative) – all settings outperform a number of sophisticated baselines, including our adaption of state-of-the-art VQA models to VisDial.
- We propose a retrieval-based evaluation protocol for Visual Dialog where the AI agent is asked to sort a list of candidate answers and evaluated on metrics such as mean-reciprocal-rank of the human response.
- We conduct studies to quantify human performance.
- Putting it all together, on the project page we demonstrate the first visual chatbot!

2.2 Related Work

Vision and Language. Our work is related to prior work in vision and language, such as those on image captioning [6–9], video/movie description [10–12], text-to-image coreference/grounding [13–18], visual storytelling [19, 20], and of course, visual question answering (VQA) [21–27, 29, 30]. However, all of these involve (at most) a single-shot natural language interaction – there is no dialog. Concurrent with our work, two recent works [35, 36] also study visual dialog.

Visual Turing Test. Closely related to our work is that of Geman *et al*. [68], who proposed a fairly restrictive 'Visual Turing Test' – a system that asks templated, binary questions. In comparison, 1) our dataset has *free-form, open-ended* natural language questions collected via two subjects chatting on Amazon Mechanical Turk (AMT), resulting in a more realistic and diverse dataset (see Fig. 2.8). 2) The dataset in [68] only contains street scenes, while our dataset has considerably more variety since it uses images from COCO [67]. Moreover, our dataset is *two orders of magnitude larger* – 2,591 images in [68] vs ∼140k images, 10 question-answer pairs per image, total of ∼1.4M QA pairs.

Text-based Question Answering. Our work is related to text-based question answering or 'reading comprehension' tasks studied in the NLP community. Some recent large-scale datasets in this domain include the 30M Factoid Question-Answer corpus [69], 100K SimpleQuestions dataset [70], DeepMind Q&A dataset [71], the 20 artificial tasks in the bAbI dataset [72], and the SQuAD dataset for reading comprehension [73]. VisDial can be viewed as a *fusion* of reading comprehension and VQA. In VisDial, the machine must comprehend the history of the past dialog and then understand the image to answer the question. By design, the answer to any question in VisDial is not present in the past dialog – if it were, the question would not be asked. The history of the dialog *contextualizes* the question – the question *'what else is she holding?'* requires a machine to comprehend the history to realize who the question is talking about and what has been excluded, and then understand the image to answer the question.

Conversational Modeling and Chatbots. Visual Dialog is the visual analogue of text-based dialog and conversation modeling. While some of the earliest developed chatbots were rule-based [37], end-to-end learning based approaches are now being actively explored [38–44]. A recent large-scale conversation dataset is the Ubuntu Dialogue Corpus [45], which contains about 500K dialogs extracted from the Ubuntu channel on Internet Relay Chat (IRC). Liu *et al*. [46] perform a study of problems in existing evaluation protocols for free-form dialog. One important difference between free-form textual dia-

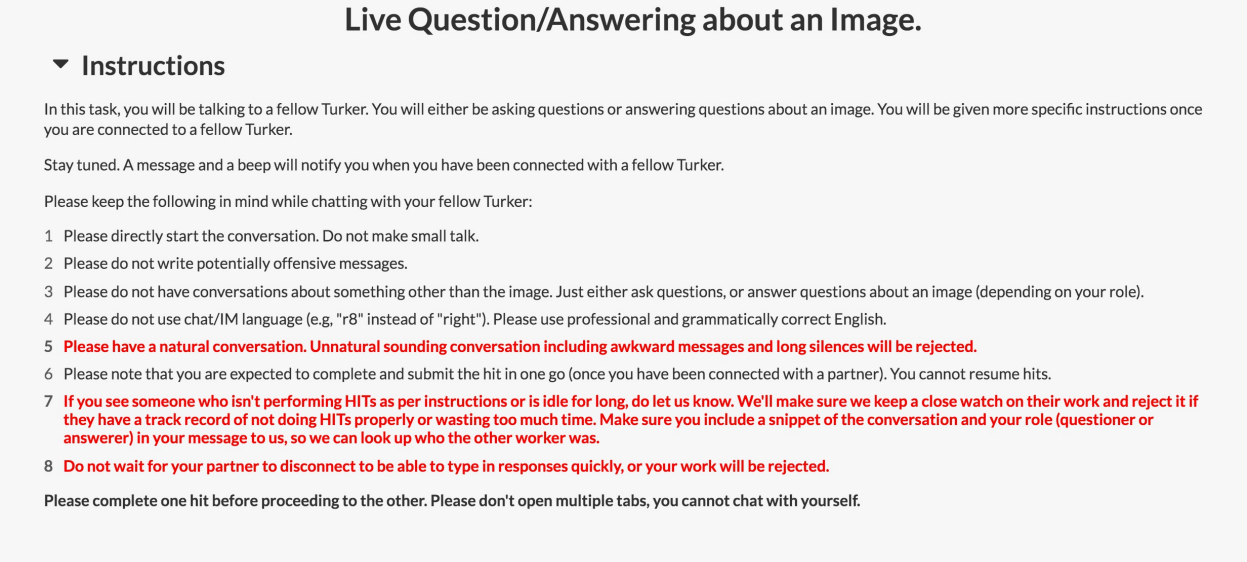
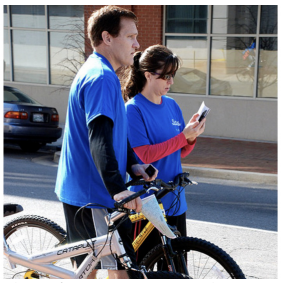


Figure 2.4: Detailed instructions for Amazon Mechanical Turkers on our interface

Caption: The skiers stood on top of the mountain.
Person A (1): how many skiers are there ?
Person B (1): hundreds
Person A (2): are they getting ready to go downhill ?
Person B (2): i think so my view is at end of line
Person A (3): is it snowing ?
Person B (3): no, there is lot of snow though
Person A (4): can you see anybody going downhill ?
Person B (4): no my view shows people going up small hill on skis i can't see what's going on
Person A (5): do you see lift ?
Person B (5): no
Person A (6): can you tell if they are male or female ?



Caption: A man and woman on bicycles are looking at a map.
Person A (1): where are they located ?
Person B (1): in city
Person A (2): are they on road ?
Person B (2): sidewalk next to 1
Person A (3): any vehicles ?
Person B (3): 1 in background
Person A (4): any other people ?
Person B (4): no
Person A (5): what color bikes ?
Person B (5): 1 silver and 1 yellow
Person A (6): do they look old or new ?

Figure 2.5: Examples from VisDial

log and VisDial is that in VisDial, the two participants are not symmetric – one person (the 'questioner') asks questions about an image *that they do not see*; the other person (the 'answerer') sees the image and only answers the questions (in otherwise unconstrained text, but no counter-questions allowed). This role assignment gives a sense of purpose to the interaction (why are we talking? To help the questioner build a mental model of the image), and allows objective evaluation of individual responses.

2.3 The Visual Dialog Dataset (VisDial)

We now describe our VisDial dataset. We begin by describing the chat interface and data-collection process on AMT, analyze the dataset, then discuss the evaluation protocol.

Consistent with previous data collection efforts, we collect visual dialog data on images from the Common Objects in Context (COCO) [67] dataset, which contains multiple objects in everyday scenes. The visual complexity of these images allows for engaging and

diverse conversations.

Live Chat Interface. Good data for this task should include dialogs that have (1) temporal continuity, (2) grounding in the image, and (3) mimic natural 'conversational' exchanges. To elicit such responses, we paired 2 workers on AMT to chat with each other in real-time (Fig. 2.3). Each worker was assigned a specific role. One worker (the 'questioner') sees only a single line of text describing an image (caption from COCO); the image remains hidden to the questioner. Their task is to ask questions about this hidden image to 'imagine the scene better'. The second worker (the 'answerer') sees the image and caption. Their task is to answer questions asked by their chat partner. Unlike VQA [22], answers are not restricted to be short or concise, instead workers are encouraged to reply as naturally and 'conversationally' as possible. Fig. 2.3c shows an example dialog.

This process is an unconstrained 'live' chat, with the only exception that the questioner must wait to receive an answer before posting the next question. The workers are allowed to end the conversation after 20 messages are exchanged (10 pairs of questions and answers).

To ensure quality of data, we provide detailed instructions on our interface as shown in Fig. 2.4. Since the workers do not know their roles before starting the study, we provide instructions for both questioner and answerer roles.

We also piloted a different setup where the questioner saw a highly blurred version of the image, instead of the caption. The conversations seeded with blurred images resulted in questions that were essentially 'blob recognition' – *'What is the pink patch at the bottom right?'*. For our full-scale data-collection, we decided to seed with just the captions since it resulted in more 'natural' questions and more closely modeled the real-world applications discussed in Sec. 2.1 where no visual signal is available to the human.

Building a 2-person chat on AMT. Despite the popularity of AMT as a data collection platform in computer vision, our setup had to design for and overcome some unique challenges – the key issue being that AMT is simply not designed for multi-user Human Intelligence Tasks (HITs). Thus, to host a live two-person chat on AMT to collect our dataset, we had to go beyond Amazon tools that are available and develop our own back-end messaging infrastructure based on Redis messaging queues and Node.js. To support data quality, we ensured that a worker could not chat with themselves (using say, two different browser tabs) by maintaining a pool of worker IDs paired. To minimize wait time for one worker while the second was being searched for, we ensured that there was always a significant pool of available HITs. If one of the workers abandoned a HIT (or was

disconnected) midway, automatic conditions in the code kicked in asking the remaining worker to either continue asking questions or providing facts (captions) about the image (depending on their role) till 10 messages were sent by them. Workers who completed the task in this way were fully compensated, but our backend discarded this data and automatically launched a new HIT on this image so a real two-person conversation could be recorded. Our entire data-collection infrastructure (front-end UI, chat interface, backend storage and messaging system, error handling protocols) is publicly available[2]. Fig. 2.5 shows random samples of dialogs from the VisDial dataset.

2.4 VisDial Dataset Analysis

2.4.1 Analyzing VisDial Questions

Visual Priming Bias. One key difference between VisDial and previous image question-answering datasets (VQA [22], Visual 7W [74], Baidu mQA [24]) is the lack of a 'visual priming bias' in VisDial. Specifically, in all previous datasets, subjects saw an image while asking questions about it. As analyzed in [27, 28, 30], this leads to a particular bias in the questions – people only ask *'Is there a clocktower in the picture?'* on pictures actually containing clock towers. This allows language-only models to perform remarkably well on VQA and results in an inflated sense of progress [27, 30]. As one particularly perverse example – for questions in the VQA dataset starting with *'Do you see a ...'*, blindly answering *'yes'* without reading the rest of the question or looking at the associated image results in an average VQA accuracy of 87%! In VisDial, questioners *do not* see the image. As a result, this bias is reduced.

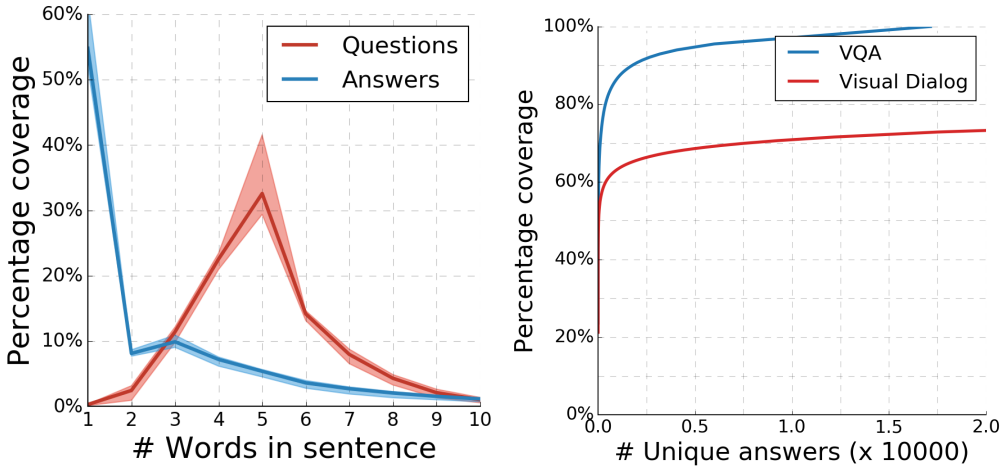
Question Statistics. Fig. 2.6a shows the distribution of question lengths in VisDial – we see that most questions range from four to ten words.

Fig. 2.7a shows question lengths by type and round. Average length of question by type is consistent across rounds. Questions starting with 'any' (*'any people?'*, *'any other fruits?'*, *etc.*) tend to be the shortest.

Fig. 2.7b shows round-wise coverage by question type. We see that as conversations progress, 'is', 'what' and 'how' questions reduce while 'can', 'do', 'does', 'any' questions occur more often. Questions starting with 'is' are the most popular in the dataset.

Fig. 2.8 shows 'sunbursts' visualizing the distribution of questions (based on the first four words) in VisDial *vs.* VQA. While there are a lot of similarities, some differences immedi-

[2] github.com/batra-mlp-lab/visdial-amt-chat



(a) (b)

Figure 2.6: Distribution of lengths for questions and answers (2.6a); and percent coverage of unique answers over all answers from the train dataset (2.6b), compared to VQA. For a given coverage, VisDial has more unique answers indicating greater answer diversity.

ately jump out. There are more binary questions[3] in VisDial as compared to VQA – the most frequent first question-word in VisDial is 'is' *vs*. 'what' in VQA. A detailed comparison of the statistics of VisDial *vs*. other datasets is available in Tab. 2.1.

Finally, there is a stylistic difference in the questions that is difficult to capture with the simple statistics above. In VQA, subjects saw the image and were asked to stump a smart robot. Thus, most queries involve specific details, often about the background (*'What program is being utilized in the background on the computer?'*). In VisDial, questioners did not see the original image and were asking questions to build a mental model of the scene. Thus, the questions tend to be open-ended, and often follow a pattern:

- Generally starting with the entities in the caption:

  'An elephant walking away from a pool in an exhibit',
  'Is there only 1 elephant?',

- digging deeper into their parts or attributes:

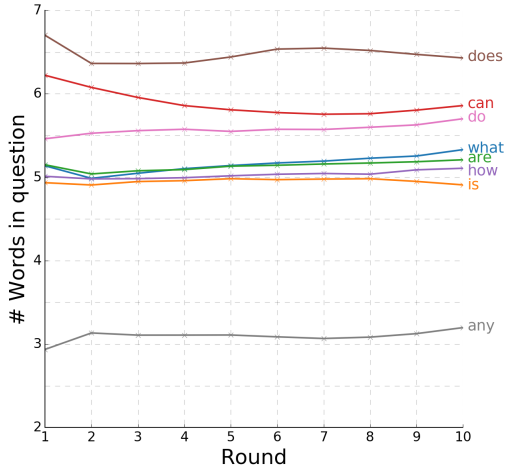
  'Is it full grown?', *'Is it facing the camera?'*,

- asking about the scene category or the picture setting:

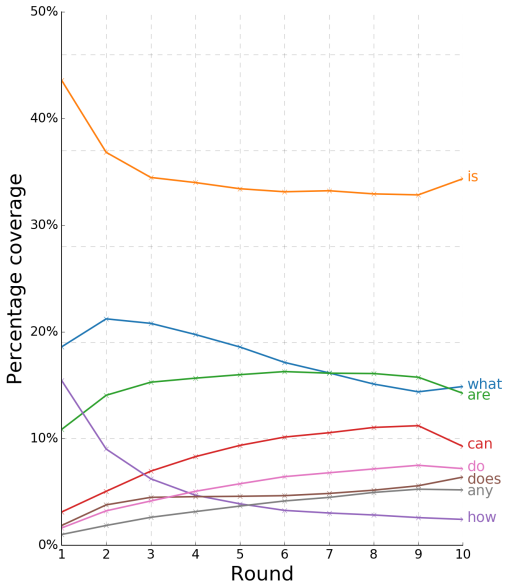
  'Is this indoors or outdoors?', *'Is this a zoo?'*,

- the weather:

[3]Questions starting in 'Do', 'Did', 'Have', 'Has', 'Is', 'Are', 'Was', 'Were', 'Can', 'Could'.



(a) Question lengths by type and round. Average length of question by type is fairly consistent across rounds. Questions starting with 'any' ('any people?', 'any other fruits?', etc.) tend to be the shortest.



(b) Percentage coverage of question types per round. As conversations progress, 'Is', 'What' and 'How' questions reduce while 'Can', 'Do', 'Does', 'Any' questions occur more often. Questions starting with 'is' are the most popular in the dataset.

Figure 2.7

'Is it snowing?', *'Is it sunny?'*,

- simply exploring the scene:

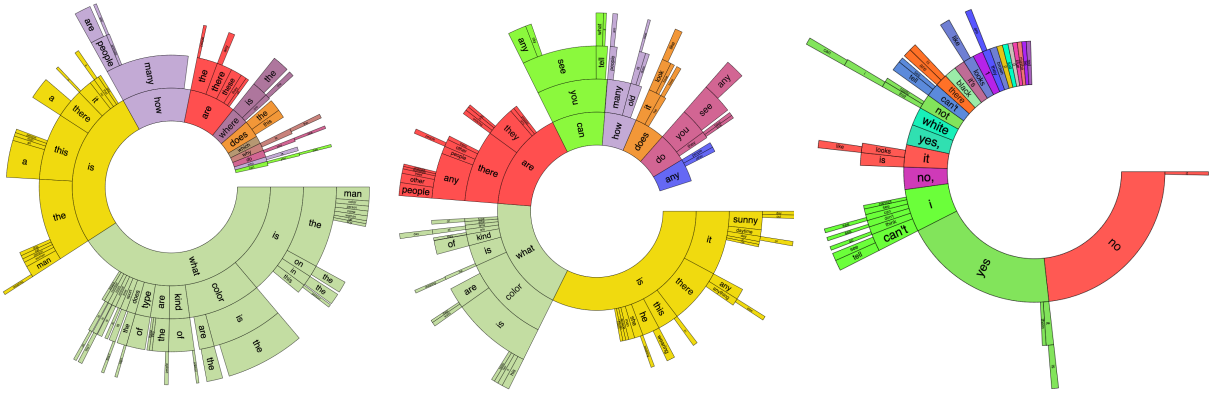
'Are there people?', *'Is there shelter for elephant?'*,

- and asking follow-up questions about the new visual entities discovered from these explorations:

'There's a blue fence in background, like an enclosure',
'Is the enclosure inside or outside?'.

2.4.2 Analyzing VisDial Answers

Answer Lengths. Fig. 2.6a shows the distribution of answer lengths. Unlike previous datasets, answers in VisDial are longer and more descriptive – mean-length 2.9 words (VisDial) vs 1.1 (VQA), 2.0 (Visual 7W), 2.8 (Visual Madlibs). Moreover, 37.1% of answers in VisDial are longer than 2 words while the VQA dataset has only 3.8% answers longer than 2 words (Tab. 2.1).



(a) VQA Questions (b) VisDial Questions (c) VisDial Answers

Figure 2.8: Distribution of first n-grams for (left to right) VQA questions, VisDial questions and VisDial answers. Word ordering starts towards the center and radiates outwards, and arc length is proportional to number of questions containing the word.

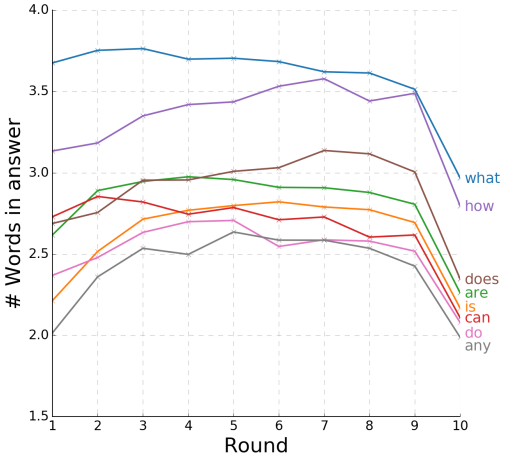
Table 2.1: Comparison of existing image question answering datasets with VisDial

	# QA Pairs	# Images	Q Length	A Length	A Length > 2	Top-1000 A	Human Accuracy
DAQUAR [21]	12,468	1,447	11.5 ± 2.4	1.2 ± 0.5	3.4%	96.4%	-
Visual Madlibs [75]	56,468	9,688	4.9 ± 2.4	2.8 ± 2.0	47.4%	57.9%	-
COCO-QA [23]	117,684	69,172	8.7 ± 2.7	1.0 ± 0	0.0%	100%	-
Baidu [24]	316,193	316,193	-	-	-	-	-
VQA [22]	614,163	204,721	6.2 ± 2.0	1.1 ± 0.4	3.8%	82.7%	✓
Visual7W [74]	327,939	47,300	6.9 ± 2.4	2.0 ± 1.4	27.6%	63.5%	✓
VisDial (Ours)	1,232,870	123,287	5.1 ± 0.0	2.9 ± 0.0	37.1%	63.2%	✓

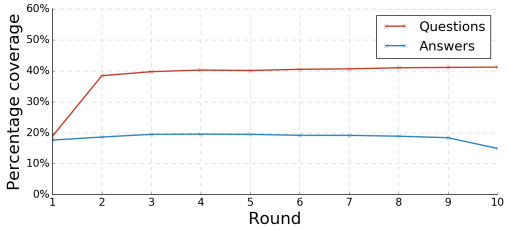
Fig. 2.9a shows answer lengths by type of question they were said in response to and round. In contrast to questions, there is significant variance in answer lengths. Answers to binary questions (*'Any people?'*, *'Can you see the dog?'*, *etc.*) tend to be short while answers to 'how' and 'what' questions tend to be more explanatory and long. Across question types, answers tend to be the longest in the middle of conversations.

Fig. 2.6b shows the cumulative coverage of all answers (y-axis) by the most frequent answers (x-axis). The difference between VisDial and VQA is stark – the top-1000 answers in VQA cover ∼83% of all answers, while in VisDial that figure is only ∼63%. There is a significant heavy tail in VisDial – most long strings are unique, and thus the coverage curve in Fig. 2.6b becomes a straight line with slope 1. In total, there are 337,527 unique answers in VisDial (out of 1,232,870 total).

Answer Types. Since the answers in VisDial are longer strings, we can visualize their distribution based on the starting few words (Fig. 2.8c). An interesting category of answers emerges – *'I think so'*, *'I can't tell'*, or *'I can't see'* – expressing doubt, uncertainty, or lack of information. This is a consequence of the questioner not being able to see the image –



(a) Answer lengths by question type and round. Across question types, average response length tends to be longest in the middle of the conversation.



(b) Percentage of QAs with pronouns for different rounds. In round 1, pronoun usage in questions is low (in fact, almost equal to usage in answers). From rounds 2 through 10, pronoun usage is higher in questions and fairly consistent across rounds.

they are asking contextually relevant questions, but not all questions may be answerable with certainty from that image. We believe this is rich data for building more human-like AI that refuses to answer questions it doesn't have enough information to answer. See [76,77] for a related, but complementary effort on question relevance in VQA.

Binary Questions *vs*. Binary Answers. In VQA, binary questions are simply those with 'yes', 'no', 'maybe' as answers [22]. In VisDial, we must distinguish between binary questions[3] and binary answers. Answers to such questions can (1) contain only 'yes' or 'no', (2) begin with 'yes', 'no', and contain additional information or clarification, (3) involve ambiguity (*'It's hard to see'*, *'Maybe'*), or (4) answer the question without explicitly saying 'yes' or 'no' (Q: *'Is there any type of design or pattern on the cloth?'*, A: *'There are circles and lines on the cloth'*). We call answers that contain 'yes' or 'no' as binary answers – 149,367 and 76,346 answers in subsets (1) and (2) from above respectively. Binary answers in VQA are biased towards 'yes' [22,27] – 61.40% of yes/no answers are 'yes'. In VisDial, the trend is reversed. Only 46.96% are 'yes' for all yes/no responses. This is understandable since workers did not see the image, and were more likely to end up with negative responses.

2.4.3 Analyzing VisDial Dialog

In Section 2.4.1, we discussed a typical flow of dialog in VisDial. We analyze two quantitative statistics here.

Coreference in dialog. Since language in VisDial is the result of a sequential conversation, it naturally contains pronouns – 'he', 'she', 'his', 'her', 'it', 'their', 'they', 'this', 'that',

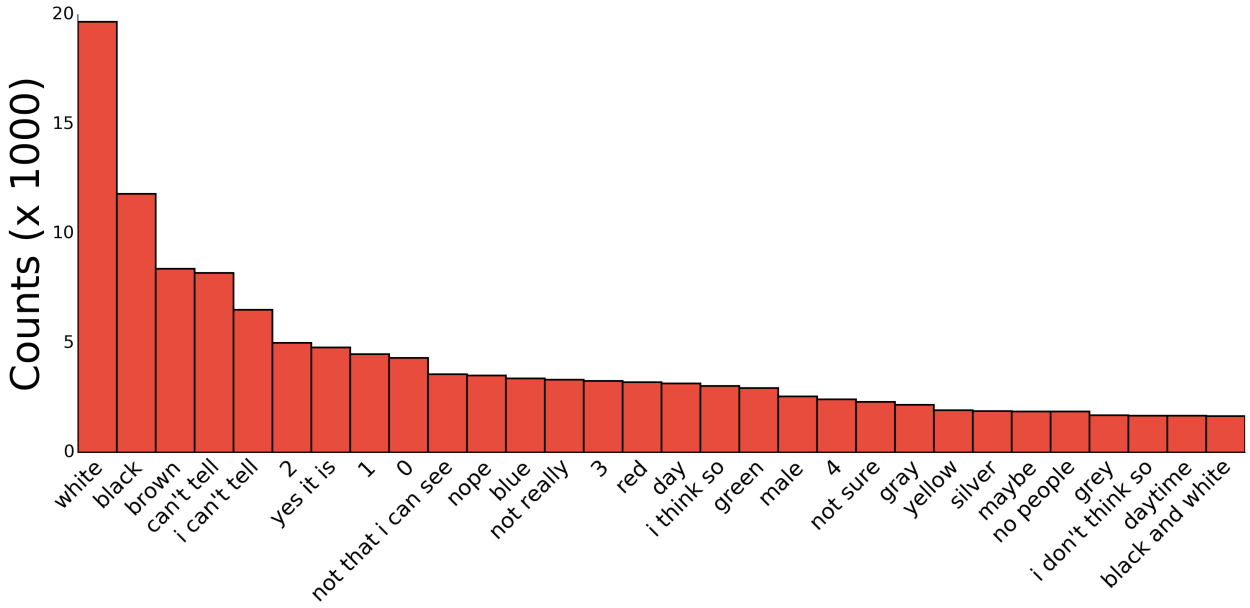


Figure 2.10: Most frequent answer responses except for 'yes'/'no'

'those', *etc*. In total, 38% of questions, 19% of answers, and *nearly all* (98%) dialogs contain at least one pronoun, thus confirming that a machine will need to overcome coreference ambiguities to be successful on this task.

In Fig. 2.9b, we see that pronoun usage is lower in the first round compared to other rounds, which is expected since there are fewer entities to refer to in the earlier rounds. The pronoun usage is also generally lower in answers than questions, which is also understandable since the answers are generally shorter than questions and thus less likely to contain pronouns. In general, the pronoun usage is fairly consistent across rounds (starting from round 2) for both questions and answers.

Temporal Continuity in Dialog Topics. *(A) Counting the Number of Topics:* In order to quantify the qualitative differences between VisDial and VQA, we performed a human study where we manually annotated question 'topics' for 40 images (a total of 400 questions), chosen randomly from the `val` set. The topic annotations were based on human judgement with a consensus of 4 annotators, with topics such as: asking about a particular object (*'What is the man doing?'*), the scene (*'Is it outdoors or indoors?'*), the weather (*"Is the weather sunny?'*), the image (*'Is it a color image?'*), and exploration (*'Is there anything else?"*). We performed similar topic annotation for questions from VQA for the same set of 40 images, and compared topic continuity in questions.

Across 10 rounds, VisDial questions have 4.55 ± 0.17 topics on average, confirming that these are not 10 independent questions. Recall that VisDial has 10 questions per image as opposed to 3 for VQA. Therefore, for a fair comparison, we compute average number of

topics in VisDial over all 'sliding windows' of 3 successive questions. For 500 bootstrap samples of batch size 40, VisDial has 2.14 ± 0.05 topics while VQA has 2.53 ± 0.09. Lower mean number of topics suggests there is more continuity in VisDial because questions do not change topics as often.

(B) Transition Probabilities over Topics: We can take this analysis a step further by computing topic transition probabilities over topics as follows. For a given sequential dialog exchange, we now count the number of topic transitions between consecutive QA pairs, normalized by the total number of possible transitions between rounds (9 for VisDial and 2 for VQA). We compute this 'topic transition probability' (how likely are two successive QA pairs to be about two different topics) for VisDial and VQA in two different settings – (1) in-order and (2) with a permuted sequence of QAs. Note that if VisDial were simply a collection of 10 independent QAs as opposed to a dialog, we would expect the topic transition probabilities to be similar for in-order and permuted variants. However, we find that for 1000 permutations of 40 topic-annotated image-dialogs, in-order-VisDial has an average topic transition probability of 0.61, while permuted-VisDial has 0.76 ± 0.02. In contrast, VQA has a topic transition probability of 0.80 for in-order *vs*. 0.83 ± 0.02 for permuted QAs.

There are two key observations: (1) In-order transition probability is lower for VisDial than VQA (*i.e*. topic transition is less likely in VisDial), and (2) Permuting the order of questions results in a larger increase for VisDial, around 0.15, compared to a mere 0.03 in case of VQA (*i.e*. in-order-VQA and permuted-VQA behave significantly more similarly than in-order-VisDial and permuted-VisDial).

Both these observations establish that there is smoothness in the temporal order of topics in VisDial, which is indicative of the narrative structure of a dialog, rather than independent question-answers.

Statistics of an NLP dialog dataset. In this analysis, our goal is to measure whether VisDial *behaves like a dialog dataset*. In particular, we compare VisDial, VQA, and Cornell Movie-Dialogs Corpus [78]. The Cornell Movie-Dialogs corpus is a text-only dataset extracted from pairwise interactions between characters from 617 movies, and is widely used as a standard corpus in the natural language processing (NLP) and dialog communities.

One popular evaluation criteria used in the dialog research community is the *perplexity* of language models trained on dialog datasets – the lower the perplexity of a model, the better it has learned the structure in the dialog dataset.

For the purpose of our analysis, we pick the popular sequence-to-sequence (Seq2Seq)

Table 2.2: Comparison of sequences in VisDial, VQA, and Cornell Movie-Dialogs corpus in their original ordering *vs*. permuted 'shuffled' ordering. Lower is better for perplexity while higher is better for classification accuracy. *Left:* Cornell corpus has the highest absolute increase in perplexity followed by VisDial and VQA, which indicates the degree of linguistic sequential structure in these datasets. *Right:* The classifier on VisDial achieves the highest accuracy, followed by Cornell, thus indicating a strong temporal continuity in the conversation.

Dataset	**Perplexity Per Token**		**Classification**
	Orig	Shuffled	
VQA	7.83	8.16 ± 0.02	52.8 ± 0.9
Cornell (10)	82.31	85.31 ± 1.51	61.0 ± 0.6
VisDial (Ours)	6.61	7.28 ± 0.01	73.3 ± 0.4

language model [79] and use the perplexity of this model trained on different datasets as a measure of temporal structure in a dataset.

As is standard in the dialog literature, we train the Seq2Seq model to predict the probability of utterance U_t given the previous utterance U_{t-1}, *i.e.* $\mathbf{P}(U_t \mid U_{t-1})$ on the Cornell corpus. For VisDial and VQA, we train the Seq2Seq model to predict the probability of a question Q_t given the previous question-answer pair, *i.e.* $\mathbf{P}(Q_t \mid (Q_{t-1}, A_{t-1}))$.

For each dataset, we used its `train` and `val` splits for training and hyperparameter tuning respectively, and report results on `test`. At test time, we only use conversations of length 10 from Cornell corpus for a fair comparison to VisDial (which has 10 rounds of QA).

For all three datasets, we created 100 permuted versions of `test`, where either QA pairs or utterances are randomly shuffled to disturb their natural order. This allows us to compare datasets in their natural ordering w.r.t. permuted orderings. Our hypothesis is that since dialog datasets have linguistic structure in the sequence of QAs or utterances they contain, this structure will be significantly affected by permuting the sequence. In contrast, a collection of independent question-answers (as in VQA) will not be significantly affected by a permutation. Tab. 2.2 compares the original, unshuffled `test` with the shuffled test-sets on two metrics:

(A) Perplexity: We compute the standard metric of *perplexity per token*, *i.e.* exponent of the normalized negative-log-probability of a sequence (where normalized is by the length of the sequence). Tab. 2.2 shows perplexities for the original unshuffled `test` and permuted `test` sequences.

We notice a few trends. First, we note that the absolute perplexity values are higher for the Cornell corpus than QA datasets. We hypothesize that this is due to the broad, unrestrictive dialog generation task in Cornell corpus, which is a more difficult task than

question prediction about images, which is in comparison a more restricted task.

Second, in all three datasets, the shuffled `test` has statistically significant higher perplexity than the original `test`, which indicates that shuffling does indeed break the linguistic structure in the sequences.

Third, the absolute increase in perplexity from natural to permuted ordering is highest in the Cornell corpus (3.0) followed by our VisDial with 0.7, and VQA at 0.35, which is indicative of the degree of linguistic structure in the sequences in these datasets. Finally, the relative increases in perplexity are 3.64% in Cornell, 10.13% in VisDial, and 4.21% in VQA – VisDial suffers the highest relative increase in perplexity due to shuffling, indicating the existence of temporal continuity that gets disrupted.

(B) Classification: As our second metric to compare datasets in their natural *vs*. permuted order, we test whether we can reliably classify a given sequence as natural or permuted.

Our classifier is a simple threshold on perplexity of a sequence. Specifically, given a pair of sequences, we compute the perplexity of both from our Seq2Seq model, and predict that the one with higher perplexity is the sequence in permuted ordering, and the sequence with lower perplexity is the one in natural ordering. The accuracy of this simple classifier indicates how easy or difficult it is to tell the difference between natural and permuted sequences. A higher classification rate indicates existence of temporal continuity in the conversation, thus making the ordering important.

Tab. 2.2 shows the classification accuracies achieved on all datasets. We can see that the classifier on VisDial achieves the highest accuracy (73.3%), followed by Cornell (61.0%). Note that this is a binary classification task with the prior probability of each class by design being equal, thus chance performance is 50%. The classifiers on VisDial and Cornell significantly outperform chance, but not the one on VQA (52.8%), indicating a lack of general temporal continuity.

To summarize this analysis, our experiments show that VisDial is significantly more dialog-like than VQA, and *behaves* more like a standard dialog dataset, the Cornell Movie-Dialogs corpus.

2.4.4 VisDial Evaluation Protocol

One fundamental challenge in dialog systems is evaluation. Similar to the state of affairs in captioning and machine translation, it is an open problem to automatically evaluate the quality of free-form answers. Existing metrics such as BLEU, METEOR, ROUGE are known to correlate poorly with human judgement in evaluating dialog responses [46].

Instead of evaluating on a downstream task [42] or holistically evaluating the entire conversation (as in goal-free chit-chat [80]), we evaluate *individual responses* at each round ($t = 1, 2, \ldots, 10$) in a retrieval or multiple-choice setup.

Specifically, at test time, a VisDial system is given an image I, the 'ground-truth' dialog history (including the image caption) $C, (Q_1, A_1), \ldots, (Q_{t-1}, A_{t-1})$, the question Q_t, *and* a list of $N = 100$ candidate answers, and asked to return a sorting of the candidate answers. The model is evaluated on retrieval metrics – (1) rank of human response (lower is better), (2) recall@k, *i.e.* existence of the human response in top-k ranked responses, and (3) mean reciprocal rank (MRR) of the human response (higher is better).

The evaluation protocol is compatible with both discriminative models (that simply score the input candidates, *e.g.* via a softmax over the options, and cannot generate new answers), and generative models (that generate an answer string, *e.g.* via Recurrent Neural Networks) by ranking the candidates by the model's log-likelihood scores.

Candidate Answers. We generate a candidate set of correct and incorrect answers from four sets:
(A) Correct: Ground-truth human response to the question.
(B) Plausible: Answers to 50 most similar questions. Similar questions are those that start with similar tri-grams and mention similar semantic concepts. To capture this, all questions are embedded into a vector space by concatenating the GloVe vectors of the first three words with the averaged GloVe embeddings of the remaining words in the questions. Euclidean distances are used to compute neighbors. Since these neighboring questions were asked on different images, their answers serve as 'hard negatives'.
(C) Popular: The 30 most popular answers from the dataset – *e.g.* 'yes', 'no', '2', '1', 'white', '3', 'grey', 'gray', '4', 'yes it is'. The inclusion of popular answers forces the machine to pick between likely *a priori* responses and plausible responses for the question, thus increasing the task difficulty.
(D) Random: The remaining are answers to random questions in the dataset. To generate 100 candidates, we first find the union of the correct, plausible, and popular answers, and include random answers until a unique set of 100 is found.

2.5 Neural Visual Dialog Models

In this section, we develop a number of neural Visual Dialog answerer models. Recall that the model is given as input – an image I, the 'ground-truth' dialog history (including

the image caption) $H = (\underbrace{C}_{H_0}, \underbrace{(Q_1, A_1)}_{H_1}, \ldots, \underbrace{(Q_{t-1}, A_{t-1})}_{H_{t-1}})$, the question Q_t, and a list of 100 candidate answers $\mathcal{A}_t = \{A_t^{(1)}, \ldots, A_t^{(100)}\}$ – and asked to return a sorting of $\mathcal{A}_t$.

At a high level, all our models follow the encoder-decoder framework, *i.e.* factorize into two parts – (1) an *encoder* that converts the input (I, H, Q_t) into a vector, and (2) a *decoder* that converts this embedded input into an output. We describe choices for each component next and present experiments with all encoder-decoder combinations.

Decoders: We use two types of decoders:

- **Generative** (LSTM) decoder: where the input encoding is set as the initial state of the Long Short-Term Memory (LSTM) language model. During training, we maximize the log-likelihood of the ground-truth answer given its corresponding input encoding (trained end-to-end). To evaluate, we use the model's log-likelihood scores and rank candidate answers.

  Note that this decoder does not need to score options during training. As a result, such models do not exploit the biases in option creation and typically underperform models that do [81], but it is debatable whether exploiting such biases is really indicative of progress. Moreover, generative decoders are more practical in that they can actually be deployed in real applications.
- **Discriminative** (softmax) decoder: computes dot product between input encoding and an LSTM encoding of each of the answer options. These dot products are fed into a softmax to compute the posterior probability over options. During training, we maximize the log-likelihood of the correct option. During evaluation, options are ranked based on their posterior probabilities.

Encoders: We develop 3 different encoders (listed below) that convert inputs (I, H, Q_t) into a joint representation. In all cases, we represent I via the $\ell 2$-normalized activations from the penultimate layer of VGG-16 [82]. For each encoder E, we experiment with all possible ablated versions: $E(Q_t), E(Q_t, I), E(Q_t, H), E(Q_t, I, H)$ (for some encoders, not all combinations are 'valid'; details below).

- **Late Fusion (LF) Encoder**: In this encoder, we treat H as a long string with the entire history $(H_0, \ldots, H_{t-1})$ concatenated. Q_t and H are separately encoded with 2 different LSTMs, and individual representations of participating inputs (I, H, Q_t) are concatenated and linearly transformed to a joint representation.
- **Hierarchical Recurrent Encoder (HRE)**: In this encoder, we capture the intuition that there is a hierarchical nature to our problem – each question Q_t is a sequence of words that need to be embedded, and the dialog as a whole is a sequence of

question-answer pairs (Q_t, A_t). Thus, similar to [43], we propose an HRE model that contains a dialog-RNN sitting on top of a recurrent block (R_t). The recurrent block R_t embeds the question and image jointly via an LSTM (early fusion), embeds each round of the history H_t, and passes a concatenation of these to the dialog-RNN above it. The dialog-RNN produces both an encoding for this round and a dialog context to pass onto the next round. We also add an attention-over-history ('Attention' in Fig. 2.11) mechanism allowing the recurrent block R_t to choose and attend to the round of the history relevant to the current question. This attention mechanism consists of a softmax over previous rounds $(0, 1, \ldots, t-1)$ computed from the history and question+image encoding.

- **Memory Network (MN) Encoder**: We develop a MN encoder that maintains each previous question and answer as a 'fact' in its memory bank and learns to refer to the stored facts and image to answer the question. Specifically, we encode Q_t with an LSTM to get a 512-d vector, encode each previous round of history $(H_0, \ldots, H_{t-1})$ with another LSTM to get a $t \times 512$ matrix. We compute inner product of question vector with each history vector to get scores over previous rounds, which are fed to a softmax to get attention-over-history probabilities. Convex combination of history vectors using these attention probabilities gives us the 'context vector', which is passed through an fc-layer and added to the question vector to construct the MN encoding. In the language of Memory Network [42], this is a '1-hop' encoding (see Fig. 2.12). We further extend this in the MNA encoder, where we compute attention over VGG-16 [82] convolutional layer image features (in a manner similar to Yang *et al*. [83]) using the question+history embedding from Memory Network.
  We use a '[encoder]-[input]-[decoder]' convention to refer to model-input combinations. For example, 'LF-QI-D' has a Late Fusion encoder with question+image inputs (no history), and a discriminative decoder.

2.6 Experiments

Splits. VisDial v0.9 contains 83k dialogs on COCO-`train` and 40k on COCO-`val` images. We split the 83k into 80k for training, 3k for validation, and test on the 40k COCO-`val`.

Baselines. We compare to a number of baselines:
(A) Simple Baselines: **Answer Prior:** Answer options to a test question are encoded with an LSTM and scored by a linear classifier. This captures ranking by frequency of answers in our training set without resolving to exact string matching. **NN-Q:** Given a test ques-

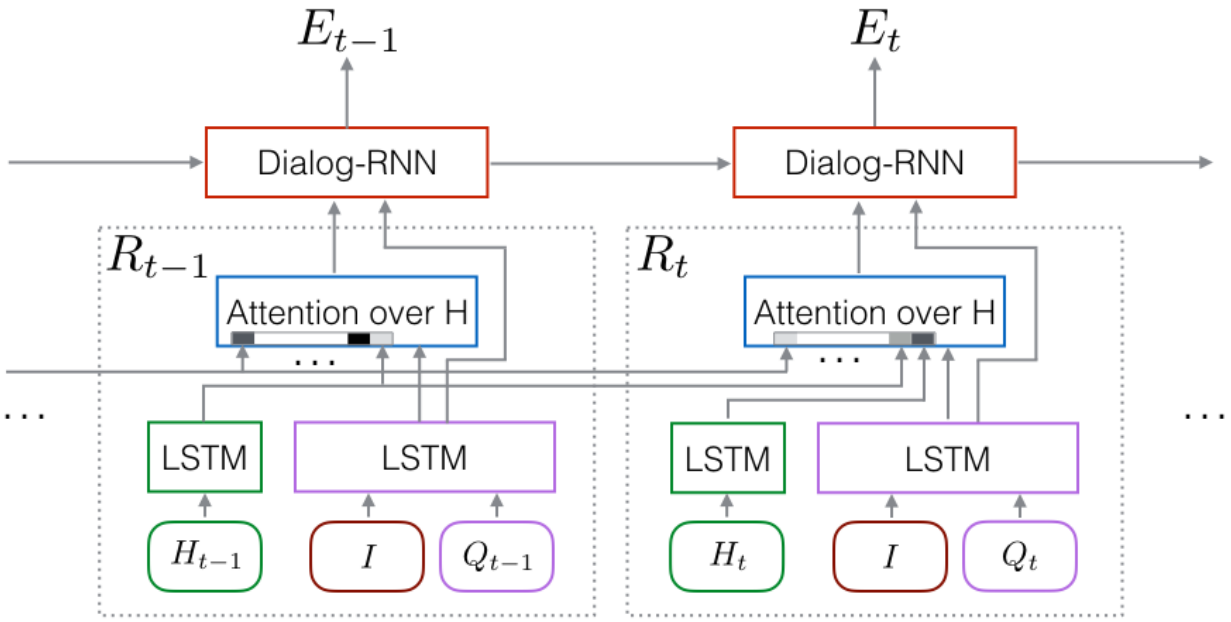


Figure 2.11: Architecture of HRE encoder with attention. At the current round R_t, the model attends to relevant history from previous rounds, based on the current question. This attention-over-history feeds into a dialog-RNN along with question to generate joint representation E_t for the decoder.

tion, we find k nearest neighbor questions (in GloVe space) from train, and score answer options by their mean-similarity with these k answers. **NN-QI:** First, we find K nearest neighbor questions for a test question. Then, we find a subset of size k based on image feature similarity. Finally, we rank options by their mean-similarity to answers to these k questions. We use $k = 20, K = 100$.

(B) Adapting VQA models to VisDial: Finally, we adapt several (near) state-of-art VQA models (SAN [83], HieCoAtt [26]) to Visual Dialog. Since VQA is posed as classification, we 'chop' the final VQA-answer softmax from these models, feed these activations to our discriminative decoder (Section 2.5), and train end-to-end on VisDial. Note that our LF-QI-D model is similar to that in [84]. Altogether, these form fairly sophisticated baselines.

Training Details. *(A) Preprocessing:* We spell-correct VisDial data using the Bing API [85]. Following VQA, we lowercase all questions and answers, convert digits to words, and remove contractions, before tokenizing using Python NLTK [86]. We then construct a dictionary of words that appear at least five times in the train set, giving us a vocabulary of around 7.5k.

(B) Hyperparameters: All our models are implemented in Torch [87]. Model hyperparameters are chosen by early stopping on `val` based on the Mean Reciprocal Rank (MRR) metric. All LSTMs are 2-layered with 512-dim hidden states. We learn 300-dim embed-

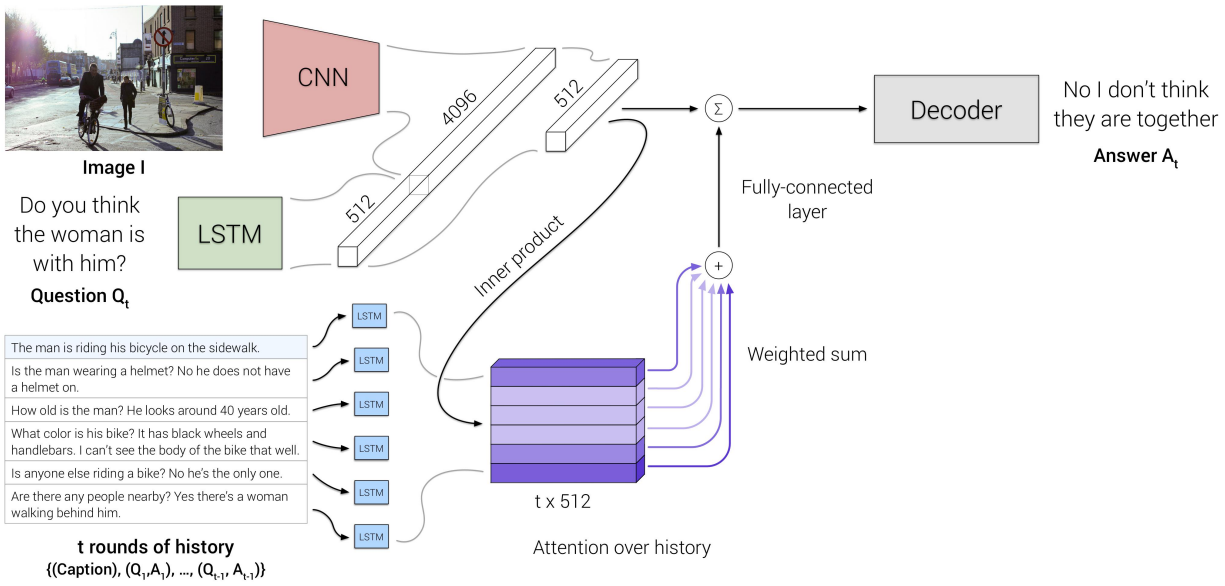


Figure 2.12: Memory Network (MN) encoder

dings for words and images. These word embeddings are shared across question, history, and decoder LSTMs. We use Adam [88] with a learning rate of 10^{-3} for all models. Gradients at each iterations are clamped to $[-5, 5]$ to avoid explosion. Our code and trained models are available at visualdialog.org.

Results. Tab. 2.3 shows results for our models and baselines on VisDial v0.9 (evaluated on 40k from COCO-`val`).

A few key takeaways – 1) As expected, all learning based models significantly outperform non-learning baselines. 2) All discriminative models significantly outperform generative models, which as we discussed is expected since discriminative models can tune to the biases in the answer options. 3) Our best generative and discriminative models are MNA-QIH-G with 0.534 MRR, and MNA-QIH-D with 0.609 MRR. 4) We observe that naively incorporating history doesn't help much (LF-Q *vs*. LF-QH and LF-QI *vs*. LF-QIH) or can even hurt a little (LF-QI-G *vs*. LF-QIH-G). However, models that better encode history (MN/HRE) perform better than corresponding LF models with/without history (*e.g.* LF-Q-D vs. MN-QH-D). 5) Models looking at I ({LF,HRE,MN,MNA }-QIH) outperform corresponding blind models (without I). 6) Attention over image features further boosts performance (MN-QIH vs. MNA-QIH).

Human Studies. We conducted studies on AMT to quantitatively evaluate human performance on this task for all combinations of {with image, without image}$\times${with history, without history} on 100 random images at each of the 10 rounds. Specifically, in each setting, we show human subjects a jumbled list of 10 candidate answers for a question –

Table 2.3: Performance of methods on VisDial v0.9, measured by mean reciprocal rank (MRR), recall@k and mean rank. Higher is better for MRR and recall@k, lower is better for mean rank.

	Model	MRR	R@1	R@5	R@10	Mean
Baseline	Answer prior	0.3735	23.55	48.52	53.23	26.50
	NN-Q	0.4570	35.93	54.07	60.26	18.93
	NN-QI	0.4274	33.13	50.83	58.69	19.62
Generative	LF-Q-G	0.5048	39.78	60.58	66.33	17.89
	LF-QH-G	0.5055	39.73	60.86	66.68	17.78
	LF-QI-G	0.5204	42.04	61.65	67.66	16.84
	LF-QIH-G	0.5199	41.83	61.78	67.59	17.07
	HRE-QH-G	0.5102	40.15	61.59	67.36	17.47
	HRE-QIH-G	0.5237	42.29	62.18	67.92	17.07
	HREA-QIH-G	0.5242	42.28	62.33	68.17	16.79
	MN-QH-G	0.5115	40.42	61.57	67.44	17.74
	MN-QIH-G	0.5259	42.29	62.85	68.88	**17.06**
	MNA-QIH-G	**0.5343**	**43.61**	**63.13**	**68.91**	17.08
Discriminative	LF-Q-D	0.5508	41.24	70.45	79.83	7.08
	LF-QH-D	0.5578	41.75	71.45	80.94	6.74
	LF-QI-D	0.5759	43.33	74.27	83.68	5.87
	LF-QIH-D	0.5807	43.82	74.68	84.07	5.78
	HRE-QH-D	0.5695	42.70	73.25	82.97	6.11
	HRE-QIH-D	0.5846	44.67	74.50	84.22	5.72
	HREA-QIH-D	0.5868	44.82	74.81	84.36	5.66
	MN-QH-D	0.5849	44.03	75.26	84.49	5.68
	MN-QIH-D	0.5965	45.55	76.22	85.37	5.46
	MNA-QIH-D	**0.6097**	**47.17**	**77.39**	**86.45**	**5.17**
VQA	SAN1-QI-D	0.5764	43.44	74.26	83.72	5.88
	HieCoAtt-QI-D	0.5788	43.51	74.49	83.96	5.84

top-9 predicted responses from our 'LF-QIH-D' model and the 1 ground truth answer – and ask them to rank the responses. Each task was done by 3 human subjects.

Results of this study are shown in the top-half of Tab. 2.4. We find that without access to the image, humans perform better when they have access to dialog history – compare the Human-QH row to Human-Q (R@1 of 30.31 *vs*. 25.10). As perhaps expected, this gap narrows down when humans have access to the image – compare Human-QIH to

Table 2.4: Human-machine performance comparison on VisDial v0.5, measured by mean reciprocal rank (MRR), recall@k for $k = \{1, 5\}$ and mean rank. Note that higher is better for MRR and recall@k, while lower is better for mean rank.

	Model	MRR	R@1	R@5	Mean
Human	Human-Q	0.441	25.10	67.37	4.19
	Human-QH	0.485	30.31	70.53	3.91
	Human-QI	0.619	46.12	82.54	2.92
	Human-QIH	0.635	48.03	83.76	2.83
Machine	HREA-QIH-G	0.477	31.64	61.61	4.42
	MN-QIH-G	0.481	32.16	61.94	4.47
	MN-QIH-D	0.553	36.86	69.39	3.48

Human-QI (R@1 of 48.03 *vs*. 46.12).

Note that these numbers are not directly comparable to machine performance reported in the main paper because models are tasked with ranking 100 responses, while humans are asked to rank 10 candidates. This is because the task of ranking 100 candidate responses would be too cumbersome for humans.

To compute comparable human and machine performance, we evaluate our best discriminative (MN-QIH-D) and generative (HREA-QIH-G, MN-QIH-G)[4] models on the same 10 options that were presented to humans. Note that in this setting, both humans and machines have R@10 = 1.0, since there are only 10 options.

Tab. 2.4 bottom-half shows the results of this comparison. We can see that, as expected, humans with full information (*i.e*. Human-QIH) perform the best with a large gap in human and machine performance (compare R@5: Human-QIH 83.76% *vs*. MN-QIH-D 69.39%). This gap is even larger when compared to generative models, which unlike the discriminative models are not actively trying to exploit the biases in the answer candidates (compare R@5: Human-QIH 83.76% *vs*. HREA-QIH-G 61.61%).

Furthermore, we see that humans outperform the best machine *even when not looking at the image*, simply on the basis of the context provided by the history (compare R@5: Human-QH 70.53% *vs*. MN-QIH-D 69.39%).

Perhaps as expected, with access to the image but not the history, humans are significantly better than the best machines (R@5: Human-QI 82.54% *vs*. MN-QIH-D 69.39%). With access to history humans perform even better.

[4]We use both HREA-QIH-G, MN-QIH-G since they have similar accuracies.

From in-house human studies and worker feedback on AMT, we find that dialog history plays the following roles for humans: (1) provides a context for the question and paints a picture of the scene, which helps eliminate certain answer choices (especially when the image is not available), (2) gives cues about the answerer's response style, which helps identify the right answer among similar answer choices, and (3) disambiguates amongst likely interpretations of the image (*i.e.*, when objects are small or occluded), again, helping identify the right answer among multiple plausible options.

2.7 Conclusions

To summarize, we introduce a new AI task – Visual Dialog, where an AI agent must hold a dialog with a human about visual content. We develop a novel two-person chat data-collection protocol to curate a large-scale dataset (VisDial), propose retrieval-based evaluation protocol, and develop a family of encoder-decoder models for Visual Dialog. We quantify human performance on this task via human studies. Our results indicate that there is significant scope for improvement, and we believe this task can serve as a testbed for measuring progress towards visual intelligence.

Chapter 3

Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning

3.1 Introduction

Chapter 2 and concurrent work by Vries *et al*. [35] both study the task of visually-grounded dialog. Perhaps somewhat counterintuitively, both these works treat dialog as a *static* supervised learning problem, rather than an *interactive* agent learning problem that it naturally is. Specifically, both works [35, 89] first collect a dataset of human-human dialog, *i.e.*, a sequence of question-answer pairs about an image $(q_1, a_1), \ldots, (q_T, a_T)$. Next, a machine (a deep neural network) is provided with the image I, the human dialog recorded till round $t-1$, $(q_1, a_1), \ldots, (q_{t-1}, a_{t-1})$, the follow-up question q_t, and is supervised to generate the human response a_t. Essentially, at each round t, the machine is artificially 'injected' into the conversation between two humans and asked to answer the question q_t; but the machine's answer $\hat{a}_t$ is thrown away, because at the next round $t+1$, the machine is again provided with the 'ground-truth' human-human dialog that includes the human response a_t and not the machine response $\hat{a}_t$. Thus, the machine is *never allowed to steer the conversation* because that would take the dialog out of the dataset, making it non-evaluable.

In this paper, we generalize the task of Visual Dialog beyond the necessary first stage of supervised learning – by posing it as a cooperative 'image guessing' game between two dialog agents. We use deep reinforcement learning (RL) to learn the policies of these agents end-to-end – from pixels to multi-agent multi-round dialog to the game reward.

Our setup is illustrated in Fig. 3.1. We formulate a game between a questioner bot (Q-BOT) and an answerer bot (A-BOT). Q-BOT is shown a 1-sentence description (a caption) of an unseen image, and is allowed to communicate in natural language (discrete symbols) with the answering bot (A-BOT), who is shown the image. The objective of this fully-cooperative game is for Q-BOT to build a mental model of the unseen image purely from the natural language dialog, and then retrieve that image from a lineup of images.

Notice that this is a challenging game. Q-BOT must ground the words mentioned in the

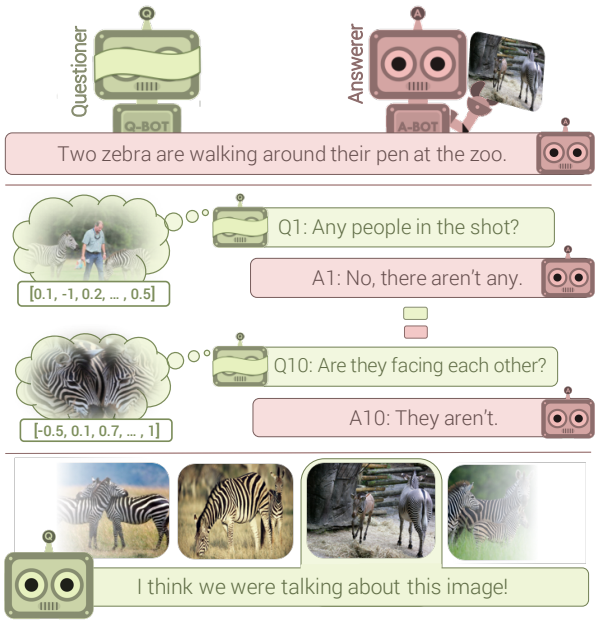


Figure 3.1: A cooperative image guessing game between two agents – Q-BOT and A-BOT– who communicate through a natural language dialog so that Q-BOT can select a particular unseen image from a lineup. We model these agents as deep neural networks and train them end-to-end with reinforcement learning.

provided caption (*'Two zebra are walking around their pen at the zoo.'*), estimate which images from the provided pool contain this content (there will typically be many such images since captions describe only the salient entities), and ask follow-up questions (*'Any people in the shot? Are there clouds in the sky? Are they facing each other?'*) that help it identify the correct image.

Analogously, A-BOT must build a mental model of what Q-BOT understands, and answer questions (*'No, there aren't any. I can't see the sky. They aren't.'*) in a precise enough way to allow discrimination between similar images from a pool (that A-BOT does not have access to) while being concise enough to not confuse the imperfect Q-BOT.

At every round of dialog, Q-BOT listens to the answer provided by A-BOT, updates its beliefs, and makes a prediction about the visual representation of the unseen image (specifically, the fc7 vector of I), and receives a reward from the environment based on how close Q-BOT's prediction is to the true fc7 representation of I. The goal of Q-BOT and A-BOT is to communicate to maximize this reward. One critical issue is that both the agents are imperfect and noisy – both 'forget' things in the past, sometimes repeat themselves, may not stay consistent in their responses, A-BOT does not have access to an external knowledge-base so it cannot answer all questions, *etc*. Thus, to succeed at the task, they must learn to play to each other's strengths.

An important question to ask is – why force the two agents to communicate in discrete symbols (English words) as opposed to continuous vectors? The reason is twofold. First, discrete symbols and natural language is interpretable. By forcing the two agents to communicate and understand natural language, we ensure that humans can not only inspect the conversation logs between two agents, but more importantly, communicate with them. After the two bots are trained, we can pair a human questioner with A-BOT to accomplish the goals of visual dialog (aiding visually/situationally impaired users), and pair a human answerer with Q-BOT to play a visual 20-questions game. The second reason to communicate in discrete symbols is to prevent cheating – if Q-BOT and A-BOT are allowed to exchange continuous vectors, then the trivial solution is for A-BOT to ignore Q-BOT's question and directly convey the fc7 vector for I, allowing Q-BOT to make a perfect prediction. In essence, discrete natural language is an interpretable low-dimensional "bottleneck" layer between these two agents.

Contributions. We introduce a novel goal-driven training for visual question answering and dialog agents. Despite significant popular interest in VQA, all previous approaches have been based on supervised learning, making this the first instance of *goal-driven* training for visual question answering / dialog. We demonstrate two experimental results.

First, as a 'sanity check' demonstration of pure RL (from scratch), we show results on a diagnostic task where perception is perfect – a synthetic world with 'images' containing a single object defined by three attributes (shape/color/style). In this synthetic world, for Q-BOT to identify an image, it must learn about these attributes. The two bots communicate via an ungrounded vocabulary, *i.e.*, symbols with no pre-specified human-interpretable meanings ('X', 'Y', '1', '2'). When trained end-to-end with RL on this task, we find that the two bots *invent their own communication protocol* – Q-BOT starts using certain symbols to query for specific attributes ('X' for color), and A-BOT starts responding with specific symbols indicating the value of that attribute ('1' for red). Essentially, we demonstrate the *automatic emergence of grounded language and communication* among 'visual' dialog agents with no human supervision!

Second, we conduct large-scale real-image experiments on the VisDial dataset [89]. With imperfect perception on real images, discovering a human-interpretable language and communication strategy from scratch is both tremendously difficult and an unnecessary re-invention of English. Thus, we pretrain with supervised dialog data in VisDial before 'fine tuning' with RL; this alleviates a number of challenges in making deep RL converge to something meaningful. We show that these RL fine-tuned bots significantly outperform the supervised bots. Most interestingly, while the supervised Q-BOT attempts to

mimic how humans ask questions, the RL trained Q-BOT *shifts strategies* and asks questions that the A-BOT is better at answering, ultimately resulting in more informative dialog and a better team.

3.2 Related Work

Vision and Language. Most related to this paper is our previous work on Visual Dialog (Chapter 2) and concurrent work by Vries *et al*. [35]. We proposed the task of Visual Dialog in [89], collected the VisDial dataset by pairing two subjects on Amazon Mechanical Turk to chat about an image (with assigned roles of 'Questioner' and 'Answerer'), and trained neural visual dialog answering models. De Vries *et al*. [35] extended the Referit game [90] to a 'GuessWhat' game, where one person asks questions about an image to guess which object has been 'selected', and the second person answers questions in 'yes'/'no'/NA (natural language answers are disallowed). One disadvantage of GuessWhat is that it requires bounding box annotations for objects; our image guessing game does not need any such annotations and thus an unlimited number of game plays may be simulated. Moreover, as described in Sec. 3.1, both these works unnaturally treat dialog as a static supervised learning problem. Although both datasets contain thousands of human dialogs, they still only represent an incredibly sparse sample of the vast space of visually-grounded questions and answers. Training robust, visually-grounded dialog agents via supervised techniques is still a challenging task.

In our work, we take inspiration from the AlphaGo [91] approach of supervision from human-expert games and reinforcement learning from self-play. Similarly, we perform supervised pretraining on human dialog data and fine-tune in an end-to-end goal-driven manner with deep RL.

20 Questions and Lewis Signaling Game. Our proposed image-guessing game is naturally the visual analog of the popular 20-questions game. More formally, it is a generalization of the Lewis Signaling (LS) [92] game, widely studied in economics and game theory. LS is a cooperative game between two players – a *sender* and a *receiver*. In the classical setting, the world can be in a number of finite discrete states $\{1, 2, \ldots, N\}$, which is known to the sender but not the receiver. The sender can send one of N discrete symbols/signals to the receiver, who upon receiving the signal must take one of N discrete actions. The game is perfectly cooperative, and one simple (though not unique) Nash Equilibrium is the 'identity mapping', where the sender encodes each world state with a bijective signal, and similarly the receiver has a bijective mapping from a signal to an action.

Our proposed 'image guessing' game is a generalization of LS with Q-BOT being the receiver and A-BOT the sender. However, in our proposed game, the receiver (Q-BOT) is not passive. It actively solicits information by asking questions. Moreover, the signaling process is not 'single shot', but proceeds over multiple rounds of conversation.

Text-only or Classical Dialog. Li *et al*. [44] have proposed using RL for training dialog systems. However, they hand-define what a 'good' utterance/dialog looks like (non-repetition, coherence, continuity, *etc*.). In contrast, taking a cue from adversarial learning [93, 94], we set up a cooperative game between two agents, such that we do not need to hand-define what a 'good' dialog looks like – a 'good' dialog is one that leads to a successful image-guessing play.

Emergence of Language. There is a long history of work on language emergence in multi-agent systems [95]. The more recent resurgence has focused on deep RL [96–99]. The high-level ideas of these concurrent works are similar to our synthetic experiments. For our large-scale real-image results, we do not want our bots to invent their own uninterpretable language and use pretraining on VisDial [89] to achieve 'alignment' with English.

3.3 Cooperative Image Guessing Game:

In Full Generality and a Specific Instantiation

Players and Roles. The game involves two collaborative agents – a questioner bot (Q-BOT) and an answerer bot (A-BOT) – with an information asymmetry. A-BOT sees an image I, Q-BOT does not. Q-BOT is primed with a 1-sentence description c of the unseen image and asks 'questions' (sequence of discrete symbols over a vocabulary V), which A-BOT answers with another sequence of symbols. The communication occurs for a fixed number of rounds.

Game Objective in General. At each round, in addition to communicating, Q-BOT must provide a 'description' $\hat{y}$ of the unknown image I based only on the dialog history and both players receive a reward from the environment inversely proportional to the error in this description under some metric $\ell(\hat{y}, y^{gt})$. We note that this is a general setting where the 'description' $\hat{y}$ can take on varying levels of specificity – from image embeddings (or fc7 vectors of I) to textual descriptions to pixel-level image generations.

Specific Instantiation. In our experiments, we focus on the setting where Q-BOT is tasked with estimating a vector embedding of the image I. Given some feature extractor (*i.e.*, a

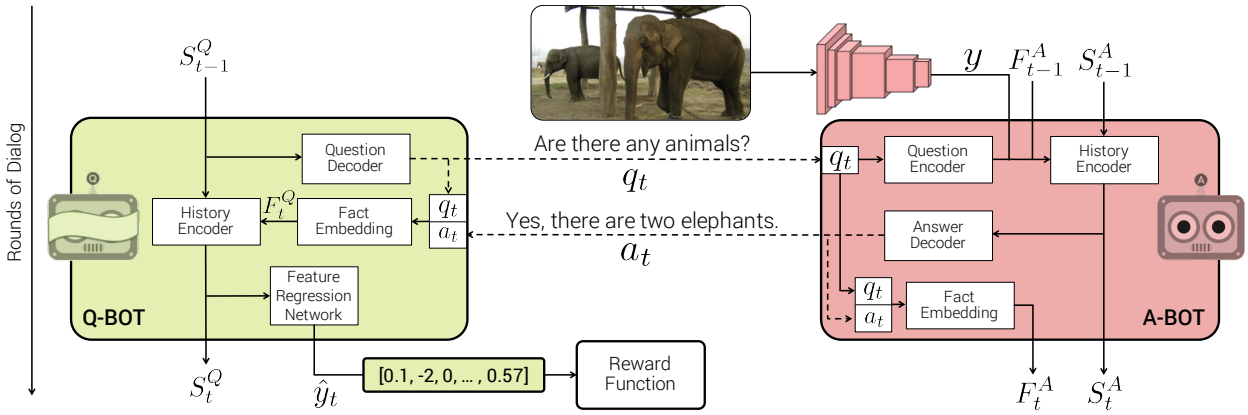


Figure 3.2: Policy networks for Q-BOT and A-BOT. At each round t of dialog, (1) Q-BOT generates a question q_t from its question decoder conditioned on its state encoding S^Q_{t-1}, (2) A-BOT encodes q_t, updates its state encoding S^A_t, and generates an answer a_t, (3) both encode the completed exchange as F^Q_t and F^A_t, and (4) Q-BOT updates its state to S^Q_t, predicts an image representation $\hat{y}_t$, and receives a reward.

pretrained CNN model, say VGG-16), no human annotation is required to produce the target 'description' $\hat{y}^{gt}$ (simply forward-prop the image through the CNN). Reward/error can be measured by simple Euclidean distance, and any image may be used as the visual grounding for a dialog. Thus, an unlimited number of 'game plays' may be simulated.

3.4 Reinforcement Learning for Dialog Agents

In this section, we formalize the training of two visual dialog agents (Q-BOT and A-BOT) with Reinforcement Learning (RL) – describing formally the *action*, *state*, *environment*, *reward*, *policy*, and training procedure. We begin by noting that although there are two agents (Q-BOT, A-BOT), since the game is perfectly cooperative, we can without loss of generality view this as a single-agent RL setup where the single "meta-agent" comprises of two "constituent agents" communicating via a natural language bottleneck layer.

Action. Both agents share a common action space consisting of all possible output sequences under a token vocabulary V. This action space is discrete and in principle, infinitely-large since arbitrary length sequences q_t, a_t may be produced and the dialog may go on forever. In our synthetic experiment, the two agents are given different vocabularies to coax a certain behavior to emerge (details in Sec. 3.5). In our VisDial experiments, the two agents share a common vocabulary of English tokens. In addition, at each round of the dialog t, Q-BOT also predicts $\hat{y}_t$, its current guess about the visual representation of the unseen image. This component of Q-BOT's action space is continuous.

State. Since there is information asymmetry (A-BOT can see the image I, Q-BOT cannot), each agent has its own observed state. For a dialog grounded in image I with caption c, the state of Q-BOT at round t is the caption and dialog history so far $s_t^Q = [c, q_1, a_1, \ldots, q_{t-1}, a_{t-1}]$, and the state of A-BOT also includes the image $s_t^A = [I, c, q_1, a_1, \ldots, q_{t-1}, a_{t-1}, q_t]$.

Policy. We model Q-BOT and A-BOT operating under stochastic policies $\pi_Q(q_t \mid s_t^Q; \theta_Q)$ and $\pi_A(a_t \mid s_t^A; \theta_A)$, such that questions and answers may be sampled from these policies conditioned on the dialog/state history. These policies will be learned by two separate deep neural networks parameterized by θ_Q and θ_A. In addition, Q-BOT includes a feature regression network $f(\cdot)$ that produces an image representation prediction *after listening to the answer at round t, i.e.*, $\hat{y}_t = f(s_t^Q, q_t, a_t; \theta_f) = f(s_{t+1}^Q; \theta_f)$. Thus, the goal of policy learning is to estimate the parameters $\theta_Q, \theta_A, \theta_f$.

Environment and Reward. The environment is the image I upon which the dialog is grounded. Since this is a purely cooperative setting, both agents receive the same reward. Let $\ell(\cdot, \cdot)$ be a distance metric on image representations (Euclidean distance in our experiments). At each round t, we define the reward for a state-action pair as:

$$r_t\Big(\underbrace{s_t^Q}_{\text{state}}, \underbrace{(q_t, a_t, y_t)}_{\text{action}}\Big) = \underbrace{\ell\left(\hat{y}_{t-1}, y^{gt}\right)}_{\text{distance at } t\text{-1}} - \underbrace{\ell\left(\hat{y}_t, y^{gt}\right)}_{\text{distance at } t} \tag{3.1}$$

i.e., the *change in distance* to the true representation before and after a round of dialog. In this way, we consider a question-answer pair to be low quality (*i.e.*, have a negative reward) if it leads the questioner to make a *worse* estimate of the target image representation than if the dialog had ended.

Note that the total reward summed over all time steps of a dialog is a function of only the initial and final states due to the cancellation of intermediate terms, *i.e.*,

$$\sum_{t=1}^{T} r_t\Big(s_t^Q, (q_t, a_t, y_t))\Big) = \underbrace{\ell\left(\hat{y}_0, y^{gt}\right) - \ell\left(\hat{y}_T, y^{gt}\right)}_{\text{overall improvement due to dialog}} \tag{3.2}$$

This is again intuitive – 'How much do the feature predictions of Q-BOT improve due to the dialog?' The details of policy learning are described in Sec. 3.4.2, but before that, let us describe the inner working of the two agents.

3.4.1 Policy Networks for Q-BOT and A-BOT

Fig. 3.2 shows an overview of our policy networks for Q-BOT and A-BOT and their interaction within a single round of dialog. Both the agent policies are modeled via Hierarchical Recurrent Encoder-Decoder neural networks, which have recently been proposed for dialog modeling [38, 43, 89].

Q-BOT consists of the following four components:

- **Fact Encoder**: Q-BOT asks a question q_t: *'Are there any animals?'* and receives an answer a_t: *'Yes, there are two elephants.'*. Q-BOT treats this concatenated (q_t, a_t)-pair as a 'fact' it now knows about the unseen image. The fact encoder is an LSTM whose final hidden state $F_t^Q \in \mathbb{R}^{512}$ is used as an embedding of (q_t, a_t).
- **State/History Encoder** is an LSTM that takes the encoded fact F_t^Q at each time step to produce an encoding of the prior dialog including time t as $S_t^Q \in \mathbb{R}^{512}$. Notice that this results in a two-level hierarchical encoding of the dialog $(q_t, a_t) \rightarrow F_t^Q$ and $(F_1^Q, \ldots, F_t^Q) \rightarrow S_t^Q$.
- **Question Decoder** is an LSTM that takes the state/history encoding from the previous round S_{t-1}^Q and generates question q_t by sequentially sampling words.
- **Feature Regression Network** $f(\cdot)$ is a single fully-connected layer that produces an image representation prediction $\hat{y}_t$ from the current encoded state $\hat{y}_t = f(S_t^Q)$.

Each of these components and their relation to each other are shown on the left side of Fig. 3.2. We collectively refer to the parameters of the three LSTM models as θ_Q and those of the feature regression network as θ_f.

A-BOT has a similar structure to Q-BOT with slight differences since it also models the image I via a CNN:

- **Question Encoder**: A-BOT receives a question q_t from Q-BOT and encodes it via an LSTM $Q_t^A \in \mathbb{R}^{512}$.
- **Fact Encoder**: Similar to Q-BOT, A-BOT also encodes the (q_t, a_t)-pairs via an LSTM to get $F_t^A \in \mathbb{R}^{512}$. The purpose of this encoder is for A-BOT to remember what it has already told Q-BOT and be able to understand references to entities already mentioned.
- **State/History Encoder** is an LSTM that takes as input at each round t – the encoded question Q_t^A, the image features from VGG [82] y, and the previous fact encoding F_{t-1}^A – to produce a state encoding, *i.e.* $\Big((y, Q_1^A, F_0^A), \ldots, (y, Q_t^A, F_{t-1}^A)\Big) \rightarrow S_t^A$. This allows the model to contextualize the current question w.r.t. the history while looking at the image to seek an answer.

- **Answer Decoder** is an LSTM that takes the state encoding S_t^A and generates a_t by sequentially sampling words.

To recap, a dialog round at time t consists of 1) Q-BOT generating a question q_t conditioned on its state encoding S_{t-1}^Q, 2) A-BOT encoding q_t, updating its state encoding S_t^A, and generating an answer a_t, 3) Q-BOT and A-BOT both encoding the completed exchange as F_t^Q and F_t^A, and 4) Q-BOT updating its state to S_t^Q based on F_t^Q and making an image representation prediction $\hat{y}_t$ for the unseen image.

3.4.2 Joint Training with Policy Gradients

In order to train these agents, we use the REINFORCE [100] algorithm that updates policy parameters $(\theta_Q, \theta_A, \theta_f)$ in response to experienced rewards. In this section, we derive the expressions for the parameter gradients for our setup.

Recall that our agents take actions – communication (q_t, a_t) and feature prediction $\hat{y}_t$ – and our objective is to maximize the expected reward under the agents' policies, summed over the entire dialog:

$$\min_{\theta_A, \theta_Q, \theta_g} J(\theta_A, \theta_Q, \theta_g) \quad \text{where,} \tag{3.3}$$

$$J(\theta_A, \theta_Q, \theta_g) = \mathop{\mathbb{E}}_{\pi_Q, \pi_A} \left[\sum_{t=1}^{T} r_t\big(s_t^Q, (q_t, a_t, y_t)\big) \right] \tag{3.4}$$

While the above is a natural objective, we find that considering the entire dialog as a single RL *episode* does not differentiate between individual good or bad exchanges within it. Thus, we update our model based on per-round rewards,

$$J(\theta_A, \theta_Q, \theta_g) = \mathop{\mathbb{E}}_{\pi_Q, \pi_A} \Big[r_t\big(s_t^Q, (q_t, a_t, y_t)\big) \Big] \tag{3.5}$$

Following the REINFORCE algorithm, we can write the gradient of this expectation as an expectation of a quantity related to the gradient. For θ_Q, we derive this explicitly:

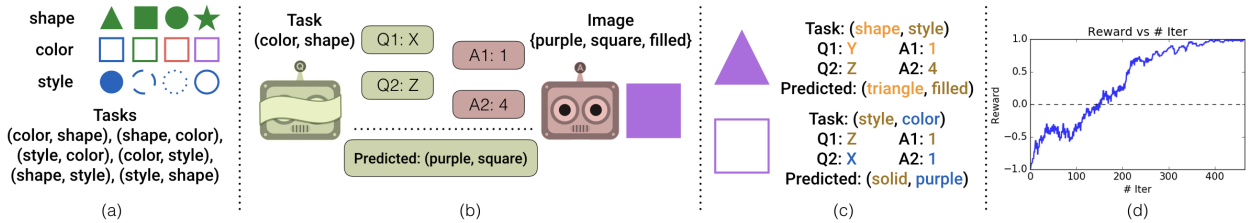


Figure 3.3: Emergence of grounded dialog: (a) Each 'image' has three attributes, and there are six tasks for Q-BOT (ordered pairs of attributes). (b) Both agents interact for two rounds followed by attribute pair prediction by Q-BOT. (c) Example 2-round dialog where grounding emerges: *color*, *shape*, *style* have been encoded as X, Y, Z respectively. (d) Improvement in reward while policy learning.

$$
\begin{aligned}
\nabla_{\theta_Q} J &= \nabla_{\theta_Q} \left[\mathop{\mathbb{E}}_{\pi_Q, \pi_A} \left[r_t \left(\cdot \right) \right] \right] \quad (r_t \text{ inputs hidden to avoid clutter}) \\
&= \nabla_{\theta_Q} \left[\sum_{q_t, a_t} \pi_Q \left(q_t | s_{t-1}^Q \right) \pi_A \left(a_t | s_t^A \right) r_t \left(\cdot \right) \right] \\
&= \sum_{q_t, a_t} \pi_Q \left(q_t | s_{t-1}^Q \right) \nabla_{\theta_Q} \log \pi_Q \left(q_t | s_{t-1}^Q \right) \pi_A \left(a_t | s_t^A \right) r_t \left(\cdot \right) \\
&= \mathop{\mathbb{E}}_{\pi_Q, \pi_A} \left[r_t \left(\cdot \right) \nabla_{\theta_Q} \log \pi_Q \left(q_t | s_{t-1}^Q \right) \right] \qquad (3.6)
\end{aligned}
$$

Similarly, gradient w.r.t. θ_A, *i.e.*, $\nabla_{\theta_A} J$ can be derived as

$$
\nabla_{\theta_A} J = \mathop{\mathbb{E}}_{\pi_Q, \pi_A} \left[r_t \left(\cdot \right) \nabla_{\theta_A} \log \pi_A \left(a_t | s_t^A \right) \right]. \qquad (3.7)
$$

As is standard practice, we estimate these expectations with sample averages. Specifically, we sample a question from Q-BOT (by sequentially sampling words from the question decoder LSTM till a stop token is produced), sample its answer from A-BOT, compute the scalar reward for this round, multiply that scalar reward to gradient of log-probability of this exchange, propagate backward to compute gradients w.r.t. all parameters θ_Q, θ_A. This update has an intuitive interpretation – if a particular (q_t, a_t) is *informative* (*i.e.*, leads to positive reward), its probabilities will be pushed up (positive gradient). Conversely, a poor exchange leading to negative reward will be pushed down (negative gradient).

Finally, since the feature regression network $f(\cdot)$ forms a deterministic policy, its parameters θ_f receive 'supervised' gradient updates for differentiable $\ell(\cdot, \cdot)$.

3.5 Emergence of Grounded Dialog

To succeed at our image guessing game, Q-BOT and A-BOT need to accomplish a number of challenging sub-tasks – they must learn a common language (*do you understand what I mean when I say 'person'?*) and develop mappings between symbols and image representations (*what does 'person' look like?*), *i.e.*, A-BOT must learn to ground language in visual perception to answer questions and Q-BOT must learn to predict plausible image representations – all in an end-to-end manner from a distant reward function. Before diving in to the full task on real images, we conduct a 'sanity check' on a synthetic dataset with perfect perception to ask – *is this even possible?*

Setup. As shown in Fig. 3.3, we consider a synthetic world with 'images' represented as a triplet of attributes – 4 *shapes*, 4 *colors*, 4 *styles* – for a total of 64 unique images. A-BOT has perfect perception and is given direct access to this representation for an image. Q-BOT is tasked with deducing two attributes of the image in a particular order – *e.g.*, if the task is *(shape, color)*, Q-BOT would need to output *(square, purple)* for a *(purple, square, filled)* image seen by A-BOT (see Fig. 3.3b). We form all 6 such tasks per image.

Vocabulary. We conducted a series of pilot experiments and found the choice of the vocabulary size to be crucial for coaxing non-trivial 'non-cheating' behavior to emerge. For instance, we found that if the A-BOT vocabulary V_A is large enough, say $|V_A| \geqslant 64$ (#images), the optimal policy learnt simply ignores what Q-BOT asks and A-BOT conveys the entire image in a single token (*e.g.* token $1 \equiv$ *(red, square, filled)*). As with human communication, an impoverished vocabulary that cannot possibly encode the richness of the visual sensor is necessary for non-trivial dialog to emerge. To ensure at least 2 rounds of dialog, we restrict each agent to only produce a single symbol utterance per round from 'minimal' vocabularies $V_A = \{1, 2, 3, 4\}$ for A-BOT and $V_Q = \{X, Y, Z\}$ for Q-BOT. Since $|V_A|^{\#\text{rounds}} <$ #images, a non-trivial dialog is necessary to succeed at the task.

Policy Learning. Since the action space is discrete and small, we instantiate Q-BOT and A-BOT as fully specified tables of Q-values (state, action, future reward estimate) and apply tabular Q-learning with Monte Carlo estimation over $10k$ episodes to learn the policies. Updates are done alternately where one bot is frozen while the other is updated. During training, we use ϵ-greedy policies [101], ensuring an action probability of 0.6 for the greedy action and split the remaining probability uniformly across other actions. At test time, we default to greedy, deterministic policy obtained from these ϵ-greedy policies. The task requires outputting the correct attribute value pair based on the task and image. Since there are a total of $4 + 4 + 4 = 12$ unique values across the 3 attributes, Q-BOT's final

action selects one of 12×12=144 attribute-pairs. We use $+1$ and -1 as rewards for right and wrong predictions.

Results. Fig. 3.3d shows the reward achieved by the agents' policies *vs*. number of RL iterations (each with 10k episodes/dialogs). We can see that the two quickly learn the optimal policy. Fig. 3.3b,c show some example exchanges between the trained bots. We find that the two invent their own communication protocol – Q-BOT consistently uses specific symbols to query for specific attributes: $X \rightarrow$ *color*, $Y \rightarrow$ *shape*, $Z \rightarrow$ *style*. And A-BOT consistently responds with specific symbols to indicate the inquired attribute, *e.g.*, if Q-BOT emits X (asks for *color*), A-BOT responds with: $1 \rightarrow$ *purple*, $2 \rightarrow$ *green*, $3 \rightarrow$ *blue*, $4 \rightarrow$ *red*. Similar mappings exist for responses to other attributes. Essentially, we find the *automatic emergence of grounded language and a communication protocol* among 'visual' dialog agents without any human supervision!

3.6 Experiments

Our synthetic experiments in the previous section establish that when faced with a cooperative task where information must be exchanged, two agents with perfect perception are capable of developing a complex communication protocol.

In general, with imperfect perception on real images, discovering human-interpretable language and communication strategy from scratch is both tremendously difficult and an unnecessary re-invention of English. We leverage the recently introduced VisDial dataset [89] that contains (as of the publicly released v0.5) human dialogs (10 rounds of question-answer pairs) on 68k images from the COCO dataset, for a total of 680k QA-pairs. Example dialogs from the VisDial dataset are shown in Tab. 3.1.

Image Feature Regression. We consider a specific instantiation of the visual guessing game described in Sec. 3.3 – specifically at each round t, Q-BOT needs to regress to the vector embedding $\hat{y}_t$ of image I corresponding to the fc7 (penultimate fully-connected layer) output from VGG-16 [82]. The distance metric used in the reward computation is ℓ_2, *i.e.* $r_t(\cdot) = \left\|y^{gt} - \hat{y}_{t-1}\right\|_2^2 - \left\|y^{gt} - \hat{y}_t\right\|_2^2$.

Training Strategies. We found two training strategies to be crucial to ensure/improve the convergence of the RL framework described in Sec. 3.4, to produce any meaningful dialog exchanges, and to ground the agents in natural language.

1) **Supervised Pretraining.** We first train both agents in a supervised manner on the `train` split of VisDial [89] v0.5 under an MLE objective. Thus, conditioned on human dialog history, Q-BOT is trained to generate the follow-up question by human1, A-BOT is trained to

Table 3.1: Selected examples of Q-BOT-A-BOT interactions for `SL-pretrained` and `RL-full-QAf`. `RL-full-QAf` interactions are diverse, less prone to repetitive and safe exchanges ("can't tell", "don't know" *etc.*), and more image-discriminative.

Image + Caption	Human-Human dialog [89]	`SL-pretrained` Q-BOT-A-BOT dialog	`RL-full-QAf` Q-BOT-A-BOT dialog
	What are their genders? 1 man 1 woman Are they both adults? Yes Do they wear goggles? Looks like sunglasses Do they have hats on? Man does Are there any other people? No What color is man's hat? Black Is it snowing now? No What is woman wearing? Blue jacket and black pants Are they smiling? Yes Do you see trees? Yes	How many people? 2 Male or female? I can't tell What are they wearing? I can't tell, they are far away What color are skis? I can't tell Are they wearing goggles? Yes Are they wearing goggles? Yes Are they wearing goggles? Yes Are they wearing goggles? Yes Are they wearing hats? Yes Are they wearing goggles? Yes	Are people male or female or both? I think 1 is male, can't see other 1 ... What are they wearing?They are all wearing snow pants and jackets Is it snowing? It does not appear to be snowing at moment Can you see any trees? Yes, there are lot of trees in background Can you see any buildings in background? No, I can not Can you see any trees? Yes, there are lot of trees in background Can you see any buildings in background? No, I do not see any buildings Can you see any trees? Yes, there are lot of trees in background Can you see any buildings in background? No , I do not see any buildings Does it look like they are in park? It does not appear to be
A couple of people in the snow on skis.			
	How old are men? They look to be 30's Are they in restaurant? I can't tell, its close up Do men look they might be related? Maybe Are they each holding slice of pizza in 1 hand and giving thumbs up with other? 1 holding slice, other holding box... What kind of pizza do they have? cheese Does box have pizza company name on it? No Are there any pictures or graphics on box? No Are men slender or heavy set? Slender Do either of men wear glasses? Yes 1 or both? 1	How old are men? They look like they are in their 20's Are they male or female? 1 male and 1 female What are they wearing? T shirt and shorts What kind of pizza is it? Looks like pepperoni Does it look good? Yes it does Can you tell what kind of pizza it is? Looks like pepperoni Are there any toppings on pizza? Pepperoni Are there any toppings on pizza? Pepperoni Are there any toppings on pizza? Looks like cheese, but I can't tell... Are there any other toppings on pizza? No	How old are men? They look like they are in their 20's Are they wearing chef's hat? No Do they have long hair? No What kind of pizza is it? Looks like pepperoni Does it look good? Yes Are there any other people around? No Are they inside or outside? Inside Can you tell what time of day it is? No Are there any other people around? No What color are walls? White
Two men giving thumbs up holding large slices of pizza.			
	How old is man? In his teens Is he wearing helmet? No Is he wearing any protective gear? No What color is his hair? Dirty blonde or dark brown What color is his shirt? Black with red letters Is he wearing shorts? Yes What color are his shorts? Denim Are there other people visible? No What color are his shoes? I can't see them Is this at skate park? Yes	How old is man? I can't see his face, I don't know What is he wearing? shorts and t shirt What color is his shirt? white what color are his pants? black what color is his shirt? blue What color are his shoes? Black What color are his shoes? Black What color are his shoes? Black What color are his shoes? Black What color are his shoes? Black	What color is skateboard? It is hard to tell but I think it's black Is he wearing helmet? Yes Is he wearing knee pads? No, no knee pads Is it day or night? It is daytime Is it sunny? It is bright out, but I see no shadows, so I can not Can you see any other people? There are 2 other people in background Is it day or night? It is daytime Is it sunny? Yes, it is sunny What color is skateboard? It is hard to tell but I think it's black Is he wearing helmet? Yes, he is wearing helmet
A man riding a skateboard down the side of a ramp.			
	What color is plane? White and blue Is there writing on plane? Yes it says lufthansa What color is ground? Grey tarmac Any people? No people visible Can you see airport? Yes What color is it? Grey Is there sunshine? No it's kind of overcast Any other planes? No other planes Any grass? No grass Any trees? No tree	What color is plane? White Is it large or small? Large Any people? No Any writing on plane? Yes Any writing on plane? Yes Any writing on plane? Yes Any writing on plane? Yes What does it say? UNK What color is plane? White Any writing on it? Yes	What color is airplane? White with red and blue stripes Can you tell what airline it is? No, it's too far away What time of day is it? It is daytime Can you see any people? No Can you see any buildings? Yes, in background Is it day or night time? It is daytime Can you see sky? Yes Are there clouds in sky? Yes, few Does it look like it's going to rain? Yes it does Does it look like it's going to rain? Yes it does
An airplane sitting on an asphalt landing strip.			
	What color is dog? Brown and white What breed is dog? Look like brittany spaniel Is dog wearing collar? Not able to see Can you tell what is cooking? Look like beans Is oven stainless steel? Yes Does it have see through glass? Yes How many racks in oven? 2 Can you see knobs? No Does oven have wooden handle? Not able to see Are beans in pot? Glass dish	What kind of dog is it? I can't tell What color is it? White Is it large or small? Large What kind of food is it? I can't tell Is it large or small? Small What color is it? White Is it large or small? Large What color are walls? White What kind of floor? Can't see floor What kind of floor? Can't see floor	What kind of dog is it? Terrier of some kind What color is it? White and grey Can you tell what kind of food it is? No Are there people around? No people Is it inside or outside? Inside Can you tell what time of day it is? No I can't What color is room? I can't see walls Do you see any windows? No Do you see any windows? No Do you see any windows? No
A dog watching food cook in an oven.			
	How old is this man? He looks like late teens Does he have short hair or long hair? Medium What color is his hair? Brown Is he wearing glasses? No Does he have facial hair? No How is he dressed? He's white dress shirt with red tie and matching red vest over it looks like school uniform Is he larger man or is he skinny? He's skinny Is this picture taken outside? Inside Can you tell if it's daytime? I'm not sure Can you tell what room he is standing in? He's in hallway	How old is man? He looks like he's in his 20s What is he wearing? T shirt and jeans What color is his hair? Black What is he wearing? T shirt and jeans What color is his hair? Black What is he wearing? T shirt and jeans What color is his hair? Black Is it short or long? Short What color is his hair? Black Is it short or long? Short	Where is man located? Looks like classroom of some sort What is man wearing? Black t shirt and jeans Are there any other people in picture? Yes, quite few people in background What color is table? Picture is black and white, but it's wood table Is there anything else on table? Not that I can see What are other people doing? They are standing in front of him, but I don't know what they Are there any windows? Not that I can see What color are walls? I can't see walls What time of day do you think it is? I can't tell What is man wearing? Black t shirt and jeans
A man making the live long and prosper sign from star trek.			

generate the response by human2, and the feature network $f(\cdot)$ is optimized to regress to y. The CNN in A-BOT is pretrained on ImageNet. This pretraining ensures that the agents can generally recognize some objects/scenes and emit English questions/answers. The space of possible (q_t, a_t) is tremendously large and without pretraining most exchanges result in no information gain about the image.

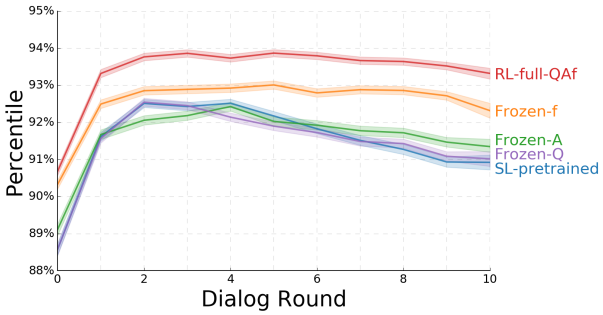
2) **Curriculum Learning.** After supervised pretraining, we 'smoothly' transition the agents to RL training according to a curriculum. Specifically, we continue supervised training for the first K (say 9) rounds of dialog and transition to policy-gradient updates for the remaining $10 - K$ rounds. We start at $K = 9$ and gradually anneal to 0. This curriculum ensures that the agent team does not suddenly diverge off policy, if one incorrect q or a is generated.

Models are pretrained for 15 epochs on VisDial, after which we transition to policy-gradient training by annealing K down by 1 every epoch. All LSTMs are 2-layered with 512-d hidden states. We use Adam [88] with a learning rate of 10^{-3}, and clamp gradients to $[-5, 5]$ to avoid explosion. All our code will be made publicly available. There is no explicit state-dependent baseline in our training as we initialize from supervised pretraining and have zero-centered reward, which ensures a good proportion of random samples are both positively and negatively reinforced.

Model Ablations. We compare to a few natural ablations of our full model, denoted `RL-full-QAf`. First, we evaluate the purely supervised agents (denoted `SL-pretrained`), *i.e.*, trained only on VisDial data (no RL). Comparison to these agents establishes how much RL helps over supervised learning. Second, we fix one of Q-BOT or A-BOT to the supervised pretrained initialization and train the other agent (and the regression network f) with RL; we label these as `Frozen-Q` or `Frozen-A` respectively. Comparing to these partially frozen agents tell us the importance of coordinated communication. Finally, we freeze the regression network f to the supervised pretrained initialization while training Q-BOT and A-BOT with RL. This measures improvements from language adaptation alone.

We quantify performance of these agents along two dimensions – how well they perform on the image guessing task (*i.e.* image retrieval) and how closely they emulate human dialogs (*i.e.* performance on VisDial dataset [89]).

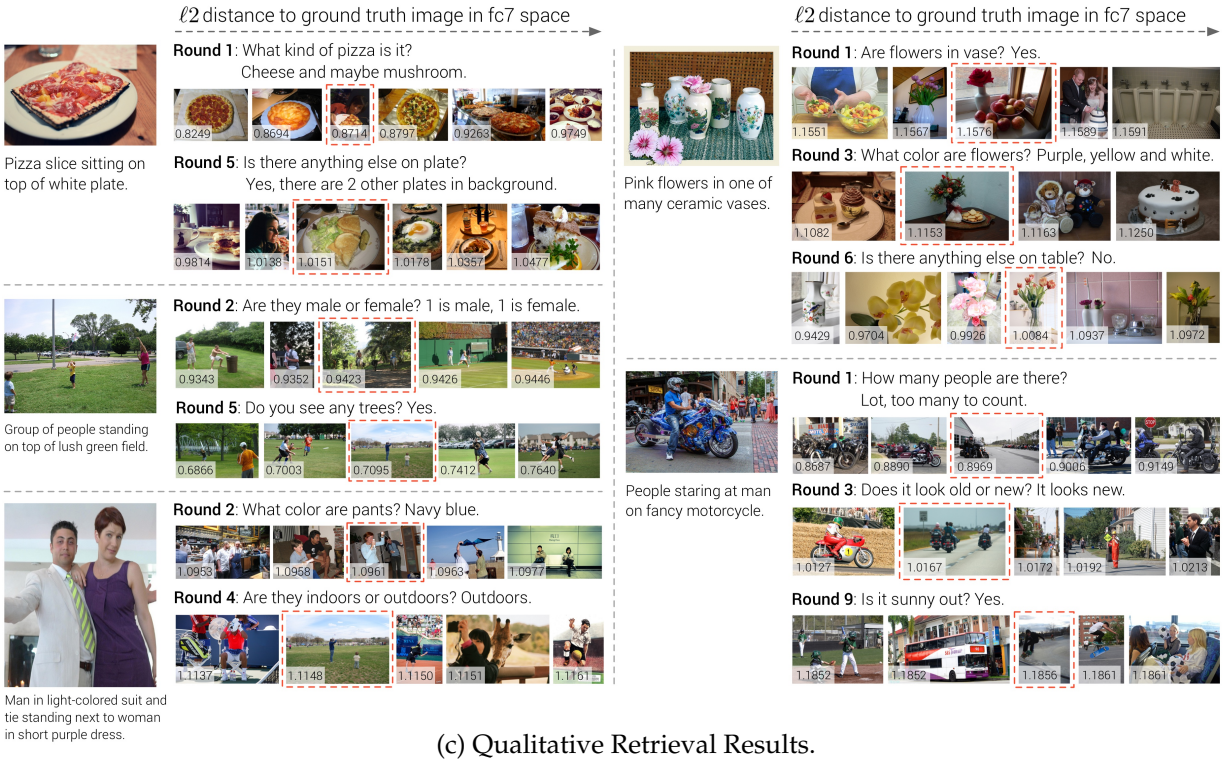
Evaluation: Guessing Game. To assess how well the agents have learned to cooperate at the image guessing task, we setup an image retrieval experiment based on the `test` split of VisDial v0.5 ($\sim$9.5k images), which were never seen by the agents in RL training. We present each image + an automatically generated caption [8] to the agents, and allow them to communicate over 10 rounds of dialog. After each round, Q-BOT predicts a feature



(a) Guessing Game Evaluation.

(b) Visual Dialog Answerer Evaluation.

Model	MRR	R@5	R@10	Mean Rank
SL-pretrain	0.436	53.41	60.09	21.83
Frozen-Q	0.428	53.12	60.19	21.52
Frozen-f	0.432	53.28	60.11	21.54
RL-full-QAf	0.428	53.08	60.22	21.54
Frozen-Q-multi	**0.437**	**53.67**	**60.48**	**21.13**



(c) Qualitative Retrieval Results.

Figure 3.4: **a) Guessing Game Evaluation.** Plot shows the rank in percentile (higher is better) of the 'ground truth' image (shown to A-BOT) as retrieved using fc7 predictions of Q-BOT *vs*. rounds of dialog. Round 0 corresponds to image guessing based on the caption alone. We can see that the `RL-full-QAf` bots significantly outperforms the `SL-pretrained` bots (and other ablations). Error bars show standard error of means. **(c)** shows qualitative results on this predicted fc7-based image retrieval. Left column shows true image and caption, right column shows dialog exchange, and a list of images sorted by their distance to the ground-truth image. The image predicted by Q-BOT is highlighted in red. We can see that the predicted image is often semantically quite similar. **b) VisDial Evaluation.** Performance of A-BOT on VisDial v0.5 `test`, under mean reciprocal rank (MRR), recall@k for $k = \{5, 10\}$ and mean rank metrics. Higher is better for MRR and recall@k, while lower is better for mean rank. We see that our proposed `Frozen-Q-multi` outperforms all other models on VisDial metrics by 3% relative gain. This improvement is entirely 'for free' since no additional annotations were required for RL.

representation $\hat{y}_t$. We sort the entire `test` set in ascending distance to this prediction and compute the rank of the source image.

Fig. 3.4a shows the mean percentile rank of the source image for our method and the baselines across the rounds (shaded region indicates standard error). A percentile rank of 95% means that the source image is closer to the prediction than 95% of the images in the set. Tab. 3.1 shows example exchanges between two humans (from VisDial), the `SL-pretrained` and the `RL-full-QAf` agents. We make a few observations:

- **RL improves image identification.** We see that `RL-full-QAf` significantly outperforms `SL-pretrained` and all other ablations (*e.g.*, at round 10, improving percentile rank by over 3%), indicating that our training framework is indeed effective at training these agents for image guessing.
- **All agents 'forget'; RL agents forget less.** One interesting trend we note in Fig. 3.4a is that all methods significantly improve from round 0 (caption-based retrieval) to rounds 2 or 3, but beyond that all methods with the exception of `RL-full-QAf` get *worse*, even though they have strictly more information. As shown in Tab. 3.1, agents will often get stuck in infinite repeating loops but this is much rarer for RL agents. Moreover, even when RL agents repeat themselves, it is after longer gaps (2-5 rounds). We conjecture that the goal of helping a partner over multiple rounds encourages longer term memory retention.
- **RL leads to more informative dialog.** SL A-BOT tends to produce 'safe' generic responses (*'I don't know'*, *'I can't see'*) but RL A-BOT responses are much more detailed (*'It is hard to tell but I think it's black'*). These observations are consistent with recent literature in text-only dialog [44]. Our hypothesis for this improvement is that human responses are diverse and SL trained agents tend to 'hedge their bets' and achieve a reasonable log-likelihood by being non-committal. In contrast, such 'safe' responses do not help Q-BOT in picking the correct image, thus encouraging an informative RL A-BOT.

Evaluation: Emulating Human Dialogs. To quantify how well the agents emulate human dialog, we evaluate A-BOT on the retrieval metrics proposed in our previous work [89]. Specifically, every question in VisDial is accompanied by 100 candidate responses. We use the log-likehood assigned by the A-BOT answer decoder to sort these candidates and report the results in Tab. 3.4b. We find that despite the RL A-BOT's answer being more informative, the improvements on VisDial metrics are minor. We believe this is because while the answers are correct, they may not necessarily mimic human responses (which

is what the answer retrieval metrics check for). In order to dig deeper, we train a variant of `Frozen-Q` with a multi-task objective – simultaneous (1) ground truth answer supervision and (2) image guessing reward, to keep A-BOT close to human-like responses. We use a weight of 1.0 for the SL loss and 10.0 for RL. This model, denoted `Frozen-Q-multi`, performs better than all other approaches on VisDial answering metrics, improving the best reported result on VisDial by 0.7 mean rank (relative improvement of 3%). Note that this gain is entirely 'free' since no additional annotations were required for RL.

Human Study. We conducted a human interpretability study to measure (1) whether humans can easily understand the Q-BOT-A-BOT dialog, and (2) how image-discriminative the interactions are. We show human subjects a pool of 16 images, the agent dialog (10 rounds), and ask humans to pick their top-5 guesses for the image the two agents are talking about. We find that mean rank of the ground-truth image for `SL-pretrained` agent dialog is 3.70 *vs*. 2.73 for `RL-full-QAf` dialog. In terms of MRR, the comparison is 0.518 *vs*. 0.622 respectively. Thus, under both metrics, humans find it easier to guess the unseen image based on `RL-full-QAf` dialog exchanges, which shows that agents trained within our framework (1) successfully develop image-discriminative language, and (2) this language is interpretable; they do not deviate off English.

3.7 Conclusions

To summarize, we introduce a novel training framework for visually-grounded dialog agents by posing a cooperative 'image guessing' game between two agents. We use deep reinforcement learning to learn the policies of these agents end-to-end – from pixels to multi-agent multi-round dialog to game reward. We demonstrate the power of this framework in a completely ungrounded synthetic world, where the agents communicate via symbols with no pre-specified meanings (X, Y, Z). We find that two bots invent their own communication protocol without any human supervision. We go on to instantiate this game on the VisDial [89] dataset, where we pretrain with supervised dialog data. We find that the RL 'fine-tuned' agents not only significantly outperform SL agents, but learn to play to each other's strengths, all the while remaining interpretable to human observers.

Part II

Agents That Can See, Talk, and Act

Chapter 4

Embodied Question Answering

4.1 Introduction

> The embodiment hypothesis is the idea that intelligence emerges in the interaction of an agent with an environment and as a result of sensorimotor activity.
>
> *Smith and Gasser [102]*

In Part I, we developed architectures and techniques for agents that can see and talk (*i.e.* visual dialog agents). While visual dialog agents can hold conversations about a static image, they cannot make any change to the state of their environment (what if I can't answer a question from my current view, or I am asked to make coffee?). Essentially, current visual dialog 'agents' have no agency! Towards this goal of building agents that can perceive, communicate in natural language, and execute actions in physical environments, we present a new AI task – ***Embodied Question Answering*** (EmbodiedQA), along with virtual environments, evaluation metrics, and a novel deep reinforcement learning (RL) model for this task.

Concretely, the EmbodiedQA task is illustrated in Fig. 4.1 – an agent is spawned at a random location in an environment (a house or building) and asked a question (*e.g.* *'What color is the car?'*). The agent perceives its environment through first-person vision (a single RGB camera) and can perform a few atomic actions: move-{forward, backward, right, left} and turn-{right, left}. The goal of the agent is to intelligently navigate the environment and gather the visual information necessary to answer the question.

EmbodiedQA is a challenging task that subsumes several fundamental AI problems as sub-tasks. Clearly, the agent must understand language (*what is the question asking?*) and vision (*what does a car look like?*), but a successful agent must also learn to perform:

Active Perception: The agent may be spawned anywhere in the environment and may not immediately 'see' the pixels containing the answer to the visual question (*i.e.* the car may not be visible). Thus, the agent *must* move to succeed – controlling the pixels that it will perceive. The agent must learn to map its visual input to the correct action based on its perception of the world, the underlying physical constraints, and its understanding

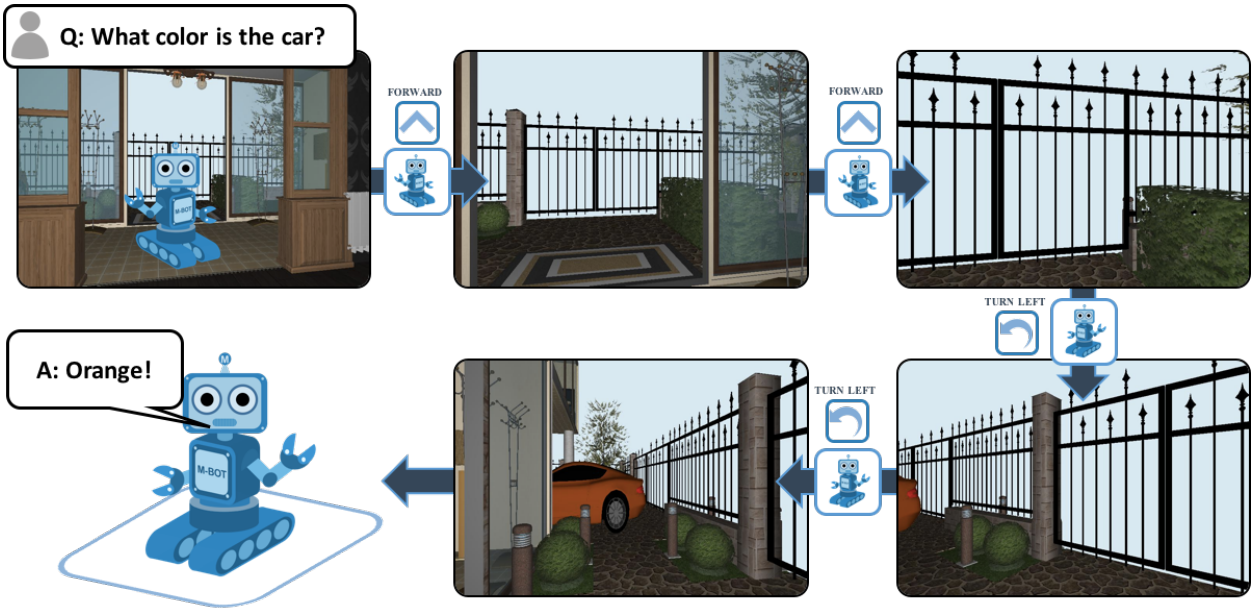


Figure 4.1: Embodied Question Answering – EmbodiedQA– tasks agents with navigating rich 3D environments in order to answer questions. These embodied agents must jointly learn language understanding, visual reasoning, and navigation to succeed.

of the question.

Common Sense Reasoning: The agent is not provided a floor-plan or map of the environment, and must navigate from egocentric views alone. Thus, it must learn common sense (*where am I? where are cars typically found in a housing compound? and where is the garage with respect to me?*) similar to how humans may navigate in a house they have never visited (*the car is probably in the garage outside, so I should find a door that leads out*).

Language Grounding: One commonly noted shortcoming of modern vision-and-language models is their lack of grounding – these models often fail to associate entities in text with corresponding image pixels, relying instead on dataset biases to respond seemingly intelligently even when attending to irrelevant regions [29, 30]. In EmbodiedQA, we take a goal-driven view of grounding – our agent grounds a visual question not into pixels but into a sequence of actions ('garage' *means* to navigate towards the house exterior where the 'car' is usually parked).

Credit Assignment: From a reinforcement learning perspective, EmbodiedQA presents a particularly challenging learning problem. Consider the question *'How many rooms contain chairs?'*. How does an agent discover that this question involves exploring the environment to visit 'rooms', detecting 'chairs', incrementing a count every time a 'chair' is in the view (except while the agent is in the same 'room'), and stopping when no more 'rooms' can be found? All without knowing what a 'room' is or how to find it, what a

'chair' looks like, or what counting is. To succeed, the agent must execute a somewhat precise sequence of hundreds of inter-dependent actions (forward, forward, turn-right, forward, forward, . . . , turn-left, '5') – all to be learned from a reward signal that says '4' is the right answer and anything else is incorrect. The task is complex enough that most random action sequences result in negative reward, and when things do go wrong its difficult for the agent to know why – was the question misunderstood? Can the agent not detect chairs? Did the agent navigate incorrectly? Was the counting incorrect?

As the first step in this challenging space, we judiciously scope out a problem space – environments, question types, learning paradigm – that allow us to augment the sparse RL rewards with imitation learning (showing the agent example trajectories) and reward shaping [103] (giving intermediate 'getting closer or farther' navigation rewards). Specifically, our approach follows the recent paradigm from robotics and deep RL [104, 105] – that the training environments are assumed to be sufficiently *instrumented* – *i.e.*, provide access to the agent location, depth and semantic annotations of the environment, and allow for computing obstacle-avoiding shortest paths from the agent to any target location.

Crucially, at test time, our agents operate entirely from egocentric RGB vision alone – no structured representation of the environments, no access to a map, no explicit localization of the agent or mapping of the environment, no A* or any other heuristic planning, and no pre-processing or hand-coded knowledge about the environment or the task of any kind. The agent in its entirety – vision, language, navigation, answering modules – is trained completely end-to-end – from raw sensory input (pixels and words) to goal-driven multi-room indoor navigation to visual question answering!

Contributions. We make the following contributions:

- We propose a new AI task: EmbodiedQA, where an agent spawned in an environment must intelligently navigate from an egocentric view to gather the necessary information to answer visual questions about its environment.
- We introduce a novel Adaptive Computation Time [106] navigator – that decomposes navigation into a 'planner' that selects actions, and a 'controller' that executes these primitive actions a variable number of times before returning control to the planner. When the agent decides it has seen the required visual information to answer the question, it stops navigating and outputs an answer.
- We initialize our agents via imitation learning and show that agents can answer questions more accurately after fine-tuning with reinforcement learning – that is, when allowed to control their own navigation *for the express purpose* of answering questions accurately. Unlike some prior work, we explicitly test and demonstrate

generalization of our agents to unseen environments.

- We evaluate our agents in *House3D* [107] a rich, interactive environment based on human-designed 3D indoor scenes from the SUNCG dataset [2]. These diverse virtual environments enable us to test generalization of our agent across floor-plans, objects, and room configurations – without the concerns of safety, privacy, and expense inherent to real robotic platforms.
- We introduce the EQA dataset of visual questions and answers grounded in House3D. The different question types test a range of agent abilities – scene recognition (`location`), spatial reasoning (`preposition`), color recognition (`color`). While the EmbodiedQA task definition supports free-fom natural language questions, we represent each question in EQA as as a *functional program* that can be programmatically generated and executed on the environment to determine the answer. This gives us the ability to control the distribution of question-types and answers in the dataset, deter algorithms from exploiting dataset bias [27, 30], and provide fine-grained breakdown of performance by skill.
- We integrated House3D renderer with Amazon Mechanical Turk (AMT) allowing subjects to *remotely operate the agent*. These expert demonstrations of question-based navigation serve as a benchmark to compare our proposed and future algorithms. All our code and data will be made publicly available.

4.2 Related Work

Closest to EmbodiedQA are recent works that extend the situated language learning paradigm to settings where agents' perceptions are local, purely visual, and change based on their actions – a setting we refer to as embodied language learning.

In concurrent and unpublished work, Hermann *et al*. [52] and Chaplot *et al*. [48] both develop embodied agents in simple game-like environments consisting of 1-2 rooms and a handful of objects with variable color and shape. In both settings, agents were able to learn to understand simple *'go to X'*/*'pick up X'* style commands where *X* would specify an object (and possibly some of its attributes). Similarly, Oh *et al*. [54] present embodied agents in a simple maze-world and task them to complete a series of instructions. In contrast to these approaches, our EmbodiedQA environments consist of multi-room homes (~8 per home) that are densely populated by a variety of objects (~54 unique objects per home). Furthermore, the instructions and commands in these works are low-level and more closely relate to actions than the questions presented in EmbodiedQA.

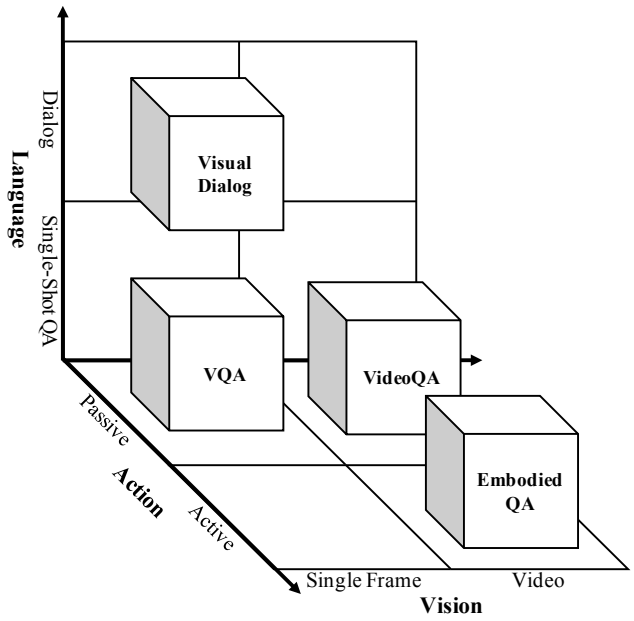


Figure 4.2: We place our work in context by arranging prior work along the axes of vision (from a single-frame to video), language (from single-shot question answering to dialog), and action (from passive observers to active agents). When viewed from this perspective, EmbodiedQA presents a novel problem configuration – single-shot QA about videos captured by goal-driven active agents. We refer the reader to Section 1.2 for an overview of prior work along various 2D slices in this space.

Interactive Environments. There are a number of interactive environments commonly used in the community, ranging from simple 2D grid-worlds (*e.g.* XWORLD [58]), to 3D game-like environments with limited realism (*e.g.* DeepMind Lab [5] or Doom [48]), to more complex, realistic environments (*e.g.* AI2-THOR [50] or Stanford 2D-3D-S [108]). While realistic environments provide rich representations of the world, most consist of only a handful of environments due to the high difficulty of their creation. On the other hand, large sets of synthetic environments can be programmatically generated; however, they typically lack realism (either in appearance or arrangement). In this work, we use the House3D [107] environment as it strikes a useful middle-ground between simple synthetic and realistic environments. See Sec. 4.3.1 for more details.

Hierarchical Agents. We model our EmbodiedQA agents as deep hierarchical agents that decompose the overall control problem such that a higher-level planner invokes lower-level controls to issue primitive actions. Such hierarchical modeling has recently shown promise in the deep reinforcement learning setting [54, 59, 109]. Our model also draws inspiration from the work on Adaptive Computation Time models of Graves [106].

(a) Sample Environments

gym	dining room
patio	living room
office	bathroom
lobby	bedroom
garage	elevator
kitchen	balcony

(b) Queryable Rooms

rug	piano	dryer	computer	fireplace	whiteboard	bookshelf	wardrobe cabinet
pan	toilet	plates	ottoman	fish tank	dishwasher	microwave	water dispenser
bed	table	mirror	tv stand	stereo set	chessboard	playstation	vacuum cleaner
cup	xbox	heater	bathtub	shoe rack	range oven	refrigerator	coffee machine
sink	sofa	kettle	dresser	knife rack	towel rack	loudspeaker	utensil holder
desk	vase	shower	washer	fruit bowl	television	dressing table	cutting board
ironing board	food processor						

(c) Queryable Objects

Figure 4.3: The EQA dataset is built on a subset of the environments and objects from the SUNCG [2] dataset. We show (a) sample environments and the (b) rooms and (c) objects that are asked about in the EmbodiedQA task.

4.3 EQA Dataset: Questions In Environments

Having placed EmbodiedQA in context, we now dive deeper by outlining the environments in which our agents are embodied and the questions they must answer. We will publicly release the environments, our curated EQA dataset, and our code to aid research in this nascent area.

4.3.1 House3D: Interactive 3D Environments

We instantiate EmbodiedQA in House3D [107], a recently introduced rich, interactive environment based on 3D indoor scenes from the SUNCG dataset [2]. Concretely, SUNCG consists of synthetic 3D scenes with realistic room and furniture layouts, manually designed using an online interior design interface (Planner5D [110]). Scenes were also further 'verified' as realistic by majority vote of three human annotators. In total, SUNCG contains over 45k environments with 49k valid floors, 404k rooms containing 5 million object instances of 2644 unique objects from 80 different categories. House3D converts SUNCG from a static 3D dataset to a set of virtual environments, where an agent (approximated as a cylinder 1 meter high) may navigate under simple physical constraints (not being able to pass through walls or objects). Fig. 4.3a shows top-down views of sample environments. Full details may be found in [107].

We build the EQA dataset on a pruned subset of environments from House3D. First, we only consider environments for which all three SUNCG annotators consider the scene layout realistic. Next, we filter out atypical environments such as those lacking ground or those that are too small or large (only keeping houses with an internal area of 300-800m^2 covering at least 1/3 the total ground area). Finally, we exclude non-home environments by requiring at least one kitchen, living room, dining room, and bedroom.

4.3.2 Question-Answer Generation

We would like to pose questions to agents that test their abilities to ground language, use common sense, reason visually, and navigate the environments. For example, answering the question *'What color is the car?'* ostensibly requires grounding the symbol *'car'*, reasoning that cars are typically outside, navigating outside and exploring until the car is found, and visually inspecting its color.

We draw inspiration from the CLEVR [111] dataset, and programmatically generate a dataset (EQA) of grounded questions and answers. This gives us the ability to control the distribution of question-types and answers in the dataset, and deter algorithms from exploiting dataset bias.

Queryable Rooms and Objects. Figs. 4.3c, 4.3b show the queryable rooms (12) and objects (50) in EQA. We exclude objects and rooms from SUNCG that are obscure (*e.g.* loggia rooms) or difficult to resolve visually (*e.g.* very small objects like light switches). We merge some semantically similar object categories (*e.g.* teapot, coffee_kettle) and singular vs plural forms of the same object type (*e.g.* (books, book)) to reduce ambiguity.

Questions as Functional Programs. Each question in EQA is represented as a functional program that can be executed on the environment yielding an answer[1]. These functional programs are composed of a small set of elementary operations ($select(\cdot)$, $unique(\cdot)$, $query(\cdot)$, *etc.*) that operate on sets of room or object annotations.

The number and the order of evaluation of these elementary operations defines a *question type* or template. For instance, one question type in EQA is the `location` template:

`location`: *'What room is the <OBJ> located in?'*

where <OBJ> refers to one of the queryable objects. The sequence of elementary operations for this question type is:

$$select(objects) \rightarrow unique(objects) \rightarrow query(location).$$

The first function, $select(objects)$, gets all the object names from the environment. The second, $unique(objects)$, retains only the objects that have a single instance in the entire house. The third, $query(location)$, generates a question (by filling in the appropriate template) for each such object. The 2nd operation, $unique(objects)$, is particularly important to generate unambiguous questions. For instance, if there are two air conditioners in the house, the question *'What room is the air conditioner located in?'* is ambiguous, with poten-

[1] or a response that the question is inapplicable (*e.g.* referring to objects not in the environment) or ambiguous (having multiple valid answers).

tially two different answers depending on which instance is being referred to.

Question Types. Associated with each question type is a template for generating a question about the rooms and objects, their attributes and relationships. We define nine question types and associated templates in EQA:

EQA v1

- `location`: *'What room is the <OBJ> located in?'*
- `color`: *'What color is the <OBJ>?'*
- `color_room`: *'What color is the <OBJ> in the <ROOM>?'*
- `preposition`: *'What is <on/above/below/next-to> the <OBJ> in the <ROOM>?'*

- `existence`: *'Is there a <OBJ> in the <ROOM>?'*
- `logical`: *'Is there a(n) <OBJ1> and a(n) <OBJ2> in the <ROOM>?'*
- `count`: *'How many <OBJs> in the <ROOM>?'*
- `room_count`: *'How many <ROOMs> in the house?'*
- `distance`: *'Is the <OBJ1> closer to the <OBJ2> than to the <OBJ3> in the <ROOM>?'*

The <ROOM> and <OBJ> tags above can be filled by any valid room or object listed in Fig. 4.3b and Fig. 4.3c respectively. Given these question templates, the possible answers are room names (`location`), object names (`preposition`), yes/no (`existence`, `logical` and `distance`), color names (`color`) or numbers (`counts`).

These questions test a range of agent abilities including object detection (`existence`), scene recognition (`location`), counting (`count`), spatial reasoning (`preposition`), color recognition (`color`), and logical operators (`logic`). Moreover, many of these questions require multiple capabilities: *e.g.*, answering a `distance` question requires recognizing the room and objects as well as reasoning about their spatial relation. Furthermore, the agent must do this by navigating the environment to find the room, looking around the room to find the objects, and possibly remembering their positions through time (if all three objects are not simultaneously visible).

Different question types also require different degrees of navigation and memory. For instance, *'How many bedrooms in the house?'* requires significant navigation (potentially exploring the entire environment) and long-term memory (keeping track of the count), while a question like *'What color is the chair in the living room?'* requires finding a single room, the living room, and looking for a chair.

EQA is easily extensible to include new elementary operations, question types, and templates as needed to increase the difficulty of the task to match the development of new models. As a first step in this challenging space, our experiments focus on EQA v1, which consists of 4 question types – `location`, `color`, `color_room`, `preposition`. One virtue of these

questions is that there is a single target queried object (<OBJ>), which enables the use of shortest paths from the agent's spawn location to the target as expert demonstrations for imitation learning (details in Section 4.4.1). We stress that EQA is not a static dataset, rather a curriculum of capabilities that we would like to achieve in embodied communicating agents.

Question-Answer Generation and Dataset Bias. In principle, we now have the ability to automatically generate all valid questions and their associated answers for each environment by executing the functional programs on the environment's annotations provided by SUNCG. However, careful consideration is needed to make sure the developed dataset is balanced over question types and answers.

For each filled question template (*e.g.* *'What room is the refrigerator located in?'*), we execute its functional form on all associated environments in the dataset (*i.e.* those containing refrigerators) to compute the answer distribution for this question. We exclude questions for which the normalized entropy of the answer distribution is below 0.5 – *e.g.*, an agent can simply memorize that refrigerators are almost always in kitchens, so this question would be discarded. We also exclude questions occurring in fewer than four environments as the normalized entropy estimates are unreliable.

Finally, in order to benchmark performance of agents vs human performance on EQA, it is important for the questions to not be tedious or frustrating for humans to answer. We do not ask `count` questions for objects with high counts (>=5) or `distance` questions between object triplets without clear differences in distance. We set these thresholds and room / object blacklists manually based on our experience performing these tasks.

Complete discussion of the question templates, functional programs, elementary operations, and various checks-and-balances can be found in the supplement.

EQA v1 Statistics. The EQA v1 dataset consists of over 5000 question across over 750 environments, referring to a total of 45 unique objects in 7 unique room types. The dataset is split into `train`, `val`, `test` such that there is no overlap in environments across splits. Fig. 4.4 shows the dataset splits and question type distribution. Approximately 6 questions are asked per environment on average, 22 at most, and 1 at fewest. There are relatively few `preposition` questions as many frequently occurring spatial relations are too easy to resolve without exploration and fail the entropy thresholding. We will make EQA v1 and the entire generation engine publicly available.

	Environments	Unique Questions	Total Questions
train	643	147	4246
val	67	104	506
test	57	105	529

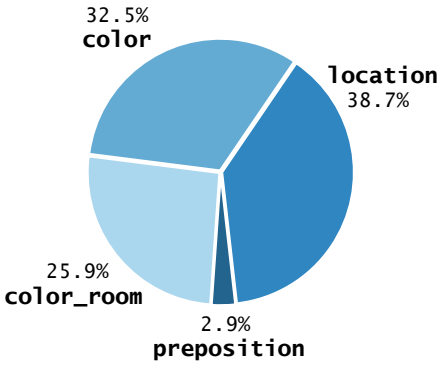


Figure 4.4: Overview of the EQA v1 dataset including dataset split statistics (left) and question type breakdown (right).

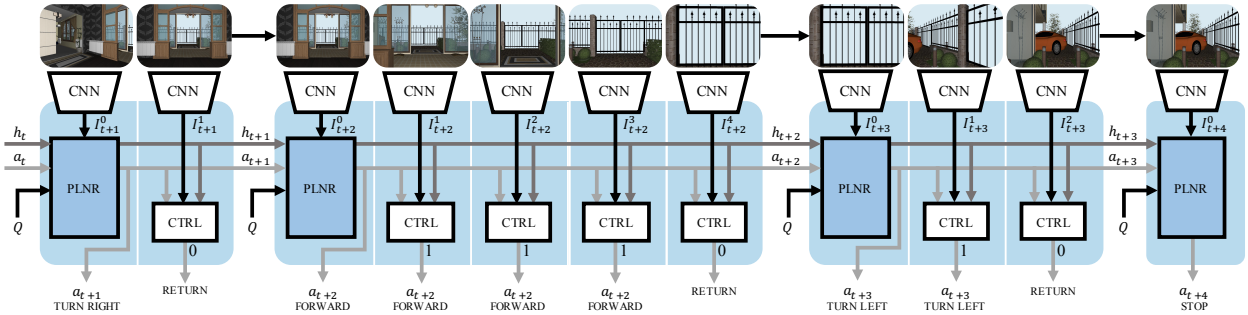


Figure 4.5: Our Adaptive Computation Time (ACT) navigator splits the navigation task between a planner and a controller module. The planner selects actions and the controller decides to continue performing that action for a variable number of time steps – resulting in a decoupling of direction (*'turn left'*) and velocity (*'5 times'*) and strengthening the long-term gradient flows of the planner module.

4.4 A Hierarchical Model for EmbodiedQA

We now introduce our proposed neural architecture for an EmbodiedQA agent. Recall that the agent is spawned at a random location in the environment, receives a question, and perceives only through a single egocentric RGB camera. Importantly, unlike some prior work [57, 59, 61, 112, 113], in EmbodiedQA, the agent does not receive any global or structured representation of the environment (map, location, objects, rooms), or of the task (the functional program that generated the question).

Overview of the Agent. The agent has 4 natural modules – vision, language, navigation, answering – and is trained from raw sensory input (pixels and words) to goal-driven multi-room indoor navigation to visual question answering. The modules themselves are built up largely from conventional neural building blocks – Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). One key technical novelty in our model is the use of Adaptive Computation Time (ACT) RNNs by Graves [106], which is an elegant approach for allowing RNNs to learn how many computational steps

to take between receiving an input and emitting an output by back-propagating through a 'halting' layer. We make use of this idea in our navigation module to cleanly separate the decision between – direction (where to move, decided by a 'planner') and velocity (how far to move, decided by a 'controler'). Fig. 4.5 illustrates the different modules in our agent, which we describe next.

Vision. Our agent takes egocentric RGB images from the House3D renderer as input, which we process with a small CNN consisting of 4 {5×5 Conv, ReLu, BatchNorm, 2×2 Max-Pool} blocks, producing a fixed-size representation.

A strong visual system for EmbodiedQA should encode information about object attributes (*i.e.* colors and textures), semantics (*i.e.* object categories), and environmental geometry (*i.e.* depth). As such, we pretrain the CNN under a multi-task pixel-to-pixel prediction framework – treating the above CNN as an encoder network, we train multiple network heads to decode the 1) original RGB values, 2) semantic class, and 3) depth of each pixel (which can be obtained from the House3D renderer).

Language. Our agents also receive questions which we encode with 2-layer LSTMs with 128-dim hidden states. Note that we learn *separate* question encoders for the navigation and answering modules – as each may need to focus on different parts of the question. For instance, in the question *'What color is the chair in the kitchen?'*, *'color'* is irrelevant for navigation and *'kitchen'* matters little for question answering (once in the kitchen).

Navigation. We introduce a novel Adaptive Computation Time (ACT) navigator that decomposes navigation into a 'planner', that selects actions (forward, left, right), and a 'controller', that executes these primitive actions a variable number of times (1,2, ...) before returning control back to the planner. Intuitively, this structure separates the intention of the agent (*i.e.* get to the other end of the room) from the series of primitive actions required to achieve this directive (*i.e. 'forward, forward, forward, ...'*), and is reminiscent of hierarchical RL approaches [54, 59, 109]. This division also allows the planner to have variable time steps between decisions, strengthening long-term gradient flows.

Formally, let $t = 1, 2, \ldots, T$ denote planner timestamps, and $n = 0, 1, 2, \ldots N(t)$ denote the variable number of controller steps. Let I_t^n denote the encoding of the image observed at t-th planner-time and n-th controller-step. We instantiate the planner as an LSTM. Thus, the planner maintains a hidden state h^t (that is updated only at planner timesteps), and samples an action $a_t \in$ {forward, turn-left, turn-right, stop-navigation}:

$$a_t,\ h_t \leftarrow \texttt{PLNR}\left(h_{t-1},\ I_t^0,\ Q,\ a_{t-1}\right), \tag{4.1}$$

where Q is the question encoding. After taking this action, the planner passes control to the controller, which considers the planner's state and the current frame to decide to continue performing a_t or to return control to the planner, *i.e.*

$$\{0, 1\} \ni c_t^n \leftarrow \texttt{CTRL}\left(h_t, a_t, I_t^n\right) \tag{4.2}$$

If $c_t^n = 1$ then the action a_t repeats and `CTRL` is applied to the next frame. Else if $c_t^n = 0$ or a max of 5 controller-steps has been reached, control is returned to the planner. We instantiate the controller as a feed-forward multi-layer perceptron with 1 hidden layer. Intuitively, the planner encodes 'intent' into the state encoding h_t and the chosen action a_t, and the controller keeps going until the visual input I_t^n aligns with the intent of the planner.

Question Answering. After the agent decides to stop (or a max number of actions have been taken), the question answering module is executed to provide an answer based on the sequence of frames $I_1^1, \ldots, I_T^n$ the agent has observed throughout its trajectory. The answering module attends to each of the last five frame, computes an attention pooled visual encoding based on image-question similarity, combines these with an LSTM encoding of the question, and outputs a softmax over the space of 172 possible answers.

4.4.1 Imitation Learning and Reward Shaping

We employ a two-stage training process. First, the navigation and answering modules are independently trained using imitation/supervised learning on automatically generated expert demonstrations of navigation. Second, the entire architecture is jointly fine-tuned using policy gradients.

Independent Pretraining via Imitation Learning. Most questions that could be asked in EmbodiedQA do not have a natural 'correct' navigation required to answer them. As mentioned in Section 4.3.2, one virtue of EQA v1 questions is that they contain a single target queried object (<OBJ>). This allows us to use the shortest path from the agent's spawn location to the target as an expert demonstration.

The navigation module is trained to mimic the shortest path actions in a teacher forcing setting - *i.e.*, given the history encoding, question encoding, and the current frame, the model is trained to predict the action that would keep it on the shortest path. We use a cross-entropy loss and train the model for 15 epochs. We find that even in this imitation learning case, it is essential to train the navigator under a distance-based curriculum. In the first epoch, we backtrack 10 steps from the target along the shortest path and initialize the agent at this point with the full history of the trajectory from the spawned location.

We step back an additional 10 steps at each successive training epoch. We train for 15 epochs total with batch size ranging from 5 to 20 questions (depending on path length due to memory limitations).

The question answering module is trained into predict the correct answer based on the question and the frames seen on the shortest path. We apply standard cross-entropy training over 50 epochs with a batch size of 20.

Target-aware Navigational Fine-tuning. While the navigation and answering modules that result from imitation learning perform well on their independent tasks, they are poorly suited to dealing with each other. Specifically, both modules are used to following the provided shortest path, but when in control the navigator may generalize poorly and provide the question answerer with unhelpful views of target (if it finds it at all). Rather than try to force the answering agent to provide correct answers from noisy or absent views, we freeze it and fine-tune the navigator.

We provide two types of reward signals to the navigator: the ultimate question answering accuracy achieved at the end of the navigation and a reward shaping [103] term that gives intermediate rewards for getting closer to the target. Specifically, the answering reward is 5 if the agent answers correctly and 0 otherwise. The navigational reward for each forward action is 0.005 times the change in distance to the target object (there is no reward or penalty for turning).

We train the agent with REINFORCE [100] policy gradients with a running average baseline for the answer reward. As in the imitation learning setting, we follow a curriculum of increasing distance between spawn and target locations. Our entire codebase is publicly available[2].

4.5 Experiments and Results

The ultimate goal of an EmbodiedQA agent is to answer questions accurately. However, it is important to disentangle success/failure at the intermediate task of navigation from the ultimate downstream task of question answering.

Question Answering Accuracy. Our agent (and all baselines) produce a probability distribution over 172 possible answers (colors, rooms, objects). We report the mean rank (**MR**) of the ground-truth answer in the answer list sorted by the agent's beliefs, where the mean is computed over all `test` questions and environments.

[2] github.com/facebookresearch/EmbodiedQA

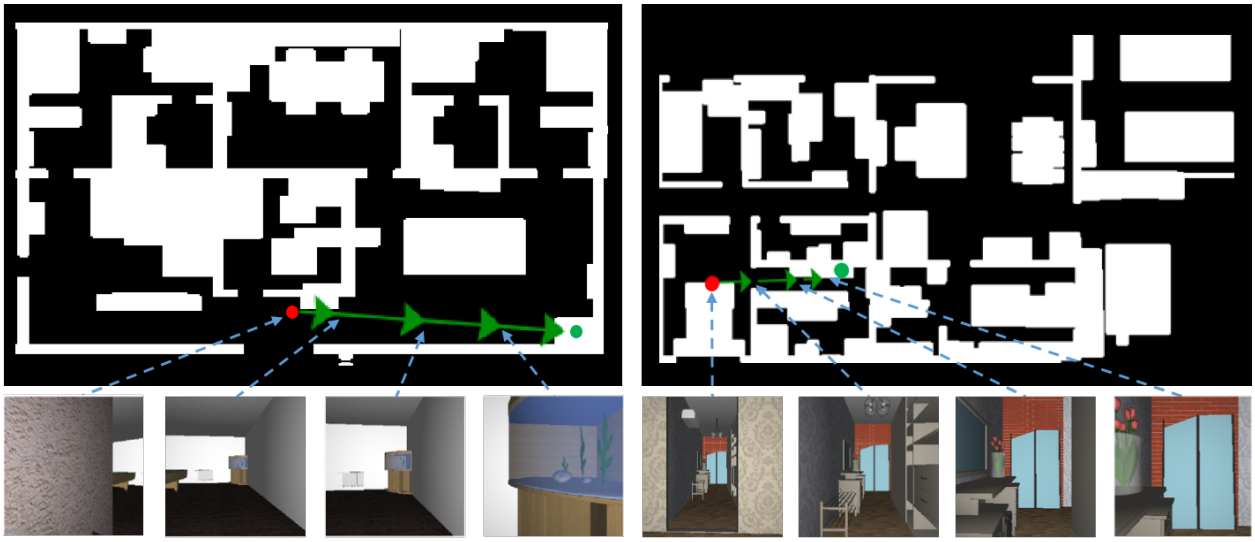


Figure 4.6: Sample trajectories from ACT+Q-RL agent projected on a floor plan (white areas are unoccupiable) and on-path egocentric views. The agent moves closer to already visible objects – potentially improving its perception of the objects. Note that the floor plan is shown only for illustration and not available to the agents.

Navigation Accuracy. We evaluate navigation performance on EQA v1 by reporting the distance to the target object at navigation termination ($\mathbf{d_T}$), change in distance to target from initial to final position ($\mathbf{d_\Delta}$), and the smallest distance to the target at any point in the episode ($\mathbf{d_{min}}$). All distances are measured in meters along the shortest path to the target. We also record the percentage of questions for which an agent either terminates in ($\%\mathbf{r_T}$) or ever enters ($\%\mathbf{r}_{\hookleftarrow}$) the room containing the target object(s). Finally, we also report the percent of episodes in which agents choose to terminate navigation and answer before reaching the maximum episode length ($\%\mathbf{stop}$). To sweep the difficulty of the task at test time, we spawn the agent 10, 30, or 50 actions away from the target and report each metric for T_{-10}, T_{-30}, T_{-50} settings.

Navigation Baselines. We compare our ACT navigator with a number of sophisticated baselines and ablations.

- **Reactive CNN.** This is a feedforward network that uses the last-n frames to predict the next action. We tried $n = \{1, 3, 5, 10\}$ and report $n = 5$, which worked best. Note that this is a target-agnostic baseline (*i.e.*, is not aware of the question). The purpose of this baseline is to check whether simply memorizing frames from training environments generalizes to test (it does not).
- **Reactive CNN+Question.** This combines the frame representation (as above) with an LSTM encoding of the question to predict the next action. This is similar to the approach of [49], with the difference that the goal is specified via a question encoding instead of

Table 4.1: Quantitative evaluation of EmbodiedQA agents on navigation and answering metrics for the EQA v1 test set. Ill-defined cells are marked with '-' because 1) reactive models don't have a stopping action, 2) humans pick a single answer from a drop-down list, so mean rank is not defined, 3) most distance metrics are trivially defined for shortest paths since they always end at the target object by design.

		Navigation																		**QA**		
		$\mathbf{d_T}$			$\mathbf{d_\Delta}$			$\mathbf{d_{min}}$			$\%\mathbf{r_T}$			$\%\mathbf{r_{↵}}$			$\%\mathbf{stop}$			**MR**		
		T_{-10}	T_{-30}	T_{-50}	T_{-10}	T_{-30}	T_{-50}	T_{-10}	T_{-30}	T_{-50}	T_{-10}	T_{-30}	T_{-50}	T_{-10}	T_{-30}	T_{-50}	T_{-10}	T_{-30}	T_{-50}	T_{-10}	T_{-30}	T_{-50}
Baselines	Reactive	2.09	2.72	3.14	-1.44	-1.09	-0.31	0.29	1.01	1.82	50%	49%	**47%**	52%	53%	48%	-	-	-	3.18	3.56	3.31
	LSTM	1.75	2.37	2.90	-1.10	-0.74	-0.07	0.34	1.06	2.05	55%	53%	44%	59%	57%	50%	80%	75%	80%	3.35	3.07	3.55
	Reactive+Q	1.58	2.27	2.89	-0.94	-0.63	-0.06	0.31	1.09	1.96	52%	51%	45%	55%	57%	**54%**	-	-	-	3.17	3.54	3.37
	LSTM+Q	1.13	2.23	2.89	-0.48	-0.59	-0.06	0.28	0.97	1.91	**63%**	53%	45%	64%	59%	**54%**	80%	71%	68%	3.11	3.39	3.31
Us	ACT+Q	**0.46**	**1.50**	**2.74**	**0.16**	**0.15**	**0.12**	0.42	1.42	2.63	58%	54%	45%	60%	56%	46%	**100%**	**100%**	**100%**	**3.09**	3.13	3.25
	ACT+Q-RL	1.67	2.19	2.86	-1.05	-0.52	0.01	**0.24**	**0.93**	**1.94**	57%	**56%**	45%	**65%**	**62%**	52%	32%	32%	24%	3.13	**2.99**	**3.22**
Oracle	HumanNav*	0.81	0.81	0.81	0.44	1.62	2.85	0.33	0.33	0.33	86%	86%	86%	87%	89%	89%	-	-	-	-	-	-
	ShortestPath+VQA	-	-	-	0.85	2.78	4.86	-	-	-	-	-	-	-	-	-	-	-	-	3.21	3.21	3.21

a target image.

- **LSTM+Question.** The above two are memoryless navigators. This LSTM navigator takes as input the encodings of the question, current frame, and previous action, and predicts the next action. Note that these are identical inputs/outputs as our ACT navigator. The purpose of comparing to this ablation of our approach is to establish the benefit of our proposed planner-controller architecture.

Navigation Oracles. We compare against two oracles:

- **HumanNav*** denotes goal-driven navigations by AMT workers remotely operating the agent (* denotes that data human studies were conducted on a subset set of `test`).
- **ShortestPaths+VQA** denotes the question answering performance achieved by our answering module when fed in shortest path at *test time*.

Table 4.1 shows the results of all baselines compared with our approach trained with just imitation learning (ACT+Q) and our approach fine-tuned with RL (ACT+Q-RL). We make a few key observations:

- **All baselines are poor navigators.** All baselines methods have *negative* $\mathbf{d_\Delta}$, *i.e.* they end up *farther* from the target than where they start. This confirms our intuition that EmbodiedQA is indeed a difficult problem.

- **Memory helps.** All models start equally far away from the target. Baselines augmented with memory (LSTM vs Reactive and LSTM-Q vs Reactive-Q) end closer to the target, *i.e.* achieve smaller $\mathbf{d_T}$, than those without.

- **ACT Navigators performs best.** Our proposed navigator (ACT+Q) achieves the smallest distance to target at the end of navigation ($\mathbf{d_T}$), and the RL-finetuned navi-

gator (ACT+Q-RL) achieves the highest answering accuracy.

- **RL agent overshoots.** Interestingly, we observe that while our RL-finetuned agent (ACT+Q-RL) gets closest to the target in its trajectory (*i.e.*, achieves least $\mathbf{d_{min}}$) and enters the target room most often (*i.e.*, achieves highest $\%\mathbf{r}_{\hookleftarrow}$), it does *not* end closest to the target (*i.e.*, does not achieve highest $\mathbf{d_{\Delta}}$). These statistics and our qualitative analysis suggests that this is because RL-finetuned agents learn to explore, with a lower stopping rate ($\%\mathbf{stop}$), and often overshoot the target. This is consistent with observations in literature [105]. In EmbodiedQA, this overshooting behavior does not hurt the question answering accuracy because the answering module can attend to frames along the trajectory. This behavior can be corrected by including a small negative reward for each action.

- **Shortest paths are not optimal for VQA.** A number of methods outperform ShortestPath+VQA in terms of answering accuracy. This is because while the shortest path clearly takes an agent to the target object, it may not provide the best vantage to answer the question. In future work, these may be improved by ray tracing methods to appropriately frame of the target object at termination.

4.6 Conclusion

We present Embodied Question Answering (EmbodiedQA) – a new AI task where an agent is spawned at a random location in a 3D environment and asked a question. In order to answer, the agent must first intelligently navigate to explore the environment, gather information through first-person (egocentric) vision, and then answer the question. We develop a novel neural hierarchical model that decomposes navigation into a 'planner' – that selects actions or a direction – and a 'controller' – that selects a velocity and executes the primitive actions a variable number of times – before returning control to the planner. We initialize the agent via imitation learning and fine-tune it using reinforcement learning for the goal of answering questions. We develop evaluation protocols for EmbodiedQA, and evaluate our agent in the House3D virtual environment. Additionally, we collect human demonstrations by connecting workers on Amazon Mechanical Turk to this environment to remotely control an embodied agents.

Chapter 5

Neural Modular Control for Embodied Question Answering

5.1 Introduction

Abstraction is an essential tool for navigating our daily lives. When seeking a late night snack, we certainly do not spend time planning out the mechanics of walking and are thankfully also unburdened of the effort of recalling to beat our heart along the way. Instead, we conceptualize our actions as a series of higher-level semantic goals – `exit bedroom; go to kitchen; open fridge; find snack;` – each of which is executed through specialized coordination of our perceptual and sensorimotor skills. This ability to abstract long, complex sequences of actions into semantically meaningful subgoals is a key component of human cognition [114] and it is natural to believe that artificial agents can benefit from applying similar mechanisms when navigating our world.

We study such hierarchical control in the context of our previous work – Embodied Question Answering (EmbodiedQA) [115] – where an embodied agent is spawned at a random location in a novel environment (*e.g.* a house) and asked to answer a question (*'What color is the piano in the living room?'*). To do so, the agent must navigate from egocentric vision alone (without access to a map of the environment), locate the entity in question (*'piano in the living room'*), and respond with the correct answer (*e.g. 'red'*). From a reinforcement learning (RL) perspective, EmbodiedQA presents challenges that are known to make learning particularly difficult – partial observability, planning over long time horizons, and sparse rewards – the agent may have to navigate through multiple rooms in search for the answer, executing hundreds of primitive motion actions along the way (`forward; forward; turn-right; ...`) and receiving a reward based only on its final answer.

To address this challenging learning problem, we develop a hierarchical **Neural Modular Controller (NMC)** – consisting of a *master* policy that determines high-level *subgoals*,

This chapter is based on work done during an internship at Facebook AI Research.

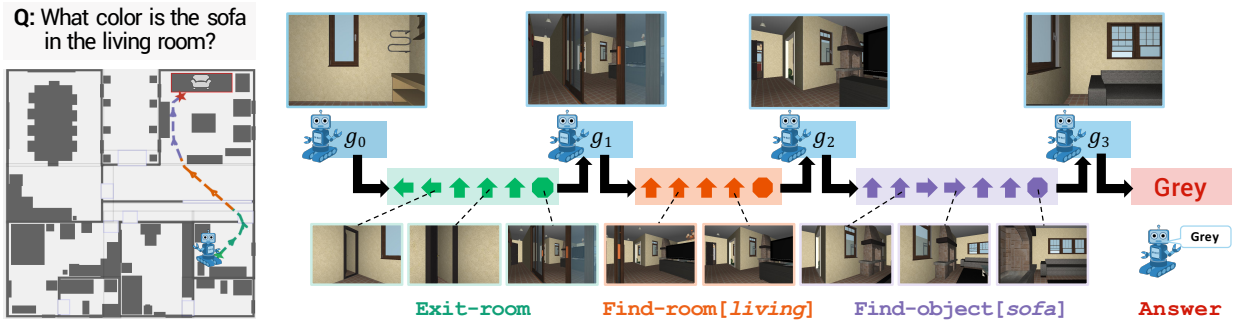


Figure 5.1: We introduce a hierarchical policy for Embodied Question Answering. Given a question ("What color is the sofa in the living room?") and observation, our master policy predicts a sequence of subgoals – `Exit-room`, `Find-room[living]`, `Find-object[sofa]`, `Answer` – that are then executed by specialized sub-policies to navigate to the target object and answer the question ("Grey").

and *sub-policies* that execute a series of low-level actions to achieve these subgoals. Our NMC model constructs a hierarchy that is arguably natural to this problem – navigation to rooms and objects *vs*. low-level motion actions. For example, NMC seeks to break down a question *'What color is the piano in the living room?'* to the series of subgoals `exit-room`; `find-room[living]`; `find-object[piano]`; `answer`; and execute this plan with specialized neural 'modules' corresponding to each subgoal. Each module is trained to issue a variable length series of primitive actions to achieve its titular subgoal – *e.g.* the `find-object[piano]` module is trained to navigate the agent to the input argument *piano* within the current room. Disentangling semantic subgoal selection from sub-policy execution results in easier to train models due to shorter time horizons. Specifically, this hierarchical structure introduces:

– **Compressed Time Horizons:** The master policy makes orders of magnitude fewer decisions over the course of a navigation than a *'flat model'* that directly predicts primitive actions – allowing the answering reward in EmbodiedQA to more easily influence high-level motor control decisions.
– **Modular Pretraining:** As each module corresponds to a specific task, they can be trained independently before being combined with the master policy. Likewise, the master policy can be trained assuming ideal modules. We do this through imitation learning [116, 117] sub-policies.
– **Interpretability:** The predictions made by the master policy correspond to semantic subgoals and exposes the reasoning of the agent to inspection (*'What is the agent trying to do right now?'*) in a significantly more interpretable fashion than just its primitive actions.

First, we learn and evaluate master and sub-policies for each of our subgoals, trained us-

ing behavior cloning on expert trajectories, reinforcement learning from scratch, and reinforcement learning after behavior cloning. We find that reinforcement learning after behavior cloning dramatically improves performance over each individual training regime. We then evaluate our combined hierarchical approach on the EQA [115] benchmark in House3D [107] environments. Our approach significantly outperforms prior work both in navigational and question answering performance – our agent is able to navigate closer to the target object and is able to answer questions correctly more often.

5.2 Related Work

Our work builds on and is related to prior work in hierarchical reinforcement and imitation learning, grounded language learning, and embodied question-answering agents in simulated environments.

Hierarchical Reinforcement and Imitation Learning. Our formulation is closely related to Le *et al*. [118], and can be seen as an instantiation of the options framework [119, 120], wherein a global master policy proposes subgoals – to be achieved by local sub-policies – towards a downstream task objective [109, 121, 122]. Relative to other work on automatic subgoal discovery in hierarchical reinforcement learning [123–125], we show that given knowledge of the problem structure, simple heuristics are quite effective in breaking down long-range planning into sequential subgoals. We make use of a combination of hierarchical behavior cloning [116] and actor-critic [126] to train our modular policy.

Neural Module Networks and Policy Sketches. At a conceptual-level, our work is analogous to recent work on neural module networks (NMNs) [59, 127, 128] for visual question answering. NMNs first predict a 'program' from the question, consisting of a sequence of primitive reasoning steps, which are then executed on the image to obtain the answer. Unlike NMNs, where each primitive reasoning module has access to the entire image (completely observable) our setting is partially observable – each sub-policy only has access to first-person RGB – making active re-evaluation of subgoals after executing each sub-policy essential. Our work is also closely related to policy sketches [59], which are symbolic descriptions of subgoals provided to the agent without any grounding or sub-policy for executing them. There are two key differences w.r.t. to our work. First, an important framework difference – Andreas *et al*. [59] assume access to a policy sketch *at test time*, *i.e*. for every task to be performed. In EmbodiedQA, this would correspond to the agent being provided with a high-level plan (`exit-room; find-room[living]; ...`) for every question it is ever asked, which is an unrealistic assumption in real-world scenarios with

a robot. In contrast, we assume that subgoal supervision (in the form of expert demonstrations and plans) are available on training environments but not on test, and the agent must *learn* to produce its own subgoals. Second, a subtle but important implementation difference – unlike [59], our sub-policy modules accept input arguments that are embeddings of target rooms and objects (*e.g.*`find-room[living]`, `find-object[piano]`). This results in our sub-policy modules being shared not just across tasks (questions) as in [59], but also across instantiations of *similar* navigation sub-policies – *i.e.*, `find-object[piano]` and `find-object[chair]` share parameters that enable data efficient learning without exhaustively learning separate policies for each.

Grounded Language Learning. Beginning with SHRDLU [56], there has been a rich progression of work in grounding language-based goal specifications into actions and pixels in physically-simulated environments. Recent deep reinforcement learning-based approaches to this explore it in 2D gridworlds [58, 59, 113], simple visual [48, 52, 54, 129–131] and textual [132, 133] environments, perceptually-realistic 3D home simulators [49, 50, 115, 134, 135], as well as real indoor scenes [51, 136, 137]. Our hierarchical policy learns to ground words from the question into two levels of hierarchical semantics. The master policy grounds words into subgoals (such as `find-room[kitchen]`), and sub-policies ground these semantic targets (such as `cutting board`, `bathroom`) into primitive actions and raw pixels, both parameterized as neural control policies and trained end-to-end.

Embodied Question-Answering Agents. Finally, hierarchical policies for embodied question answering have previously been proposed by us (described in Chapter 4 [115]) and by Gordon *et al*. [134] in the AI2-THOR environment [138]. Our hierarchical policy, in comparison, is human-interpretable, *i.e*. the subgoal being pursued at every step of navigation is semantic, and due to the modular structure, can navigate over longer paths than prior work, spanning multiple rooms.

5.3 Neural Modular Control

We now describe our approach in detail. Recall that given a question, the goal of our agent is to predict a sequence of navigation subgoals and execute them to ultimately find the target object and respond with the correct answer. We first present our modular hierarchical policy. We then describe how we extract optimal plans from shortest path navigation trajectories for behavior cloning. And finally, we describe how the various modules are combined and trained with a combination of imitation learning (behavior cloning) and reinforcement learning.

5.3.1 Hierarchical Policy

Notation. Recall that NMC has 2 levels in the hierarchy – a master policy that generates subgoals and sub-policies for each of these subgoals. We use i to index the sequence of subgoals and t to index actions generated by sub-policies. Let $\mathcal{S} = \{s\}$ denote the set of states, $\mathcal{G} = \{g\}$ the set of variable-time subgoals with elements $g = \langle g_{\text{task}}, g_{\text{argument}} \rangle$, *e.g.* $g = \langle$`exit-room`,`None`$\rangle$, or $g = \langle$`find-room`,`bedroom`$\rangle$. Let $\mathcal{A} = \{a\}$ be the set of primitive actions (`forward`, `turn-left`, `turn-right`). The learning problem can then be succinctly put as learning a master policy $\pi_\theta : \mathcal{S} \to \mathcal{G}$ parameterized by θ and sub-policies $\pi_{\phi_g} : \mathcal{S} \to \mathcal{A} \cup \{\texttt{stop}\}$ parameterized by ϕ_g, $\forall g \in \mathcal{G}$, where the `stop` action terminates a sub-policy and returns control to the master policy.

While navigating an environment, control alternates between the master policy selecting subgoals and sub-policies executing these goals through a series of primitive actions. More formally, given an initial state s_0 the master policy predicts a subgoal $g_0 \sim \pi_\theta(g|s_0)$, the corresponding sub-policy executes until some time T_0 when either (1) the sub-policy terminates itself by producing the `stop` token $a_{T_0} \sim \pi_{\phi_{g_0}}(a|s_{T_0}) =$ `stop` or (2) a maximum number of primitive actions has been reached. Either way, this returns the control back to the master policy which predicts another subgoal and repeats this process until termination. This results in a state-subgoal trajectory:

$$\Sigma = \Big(\underbrace{s_0, g_0}_{\text{subgoal 0}} \;,\; \underbrace{s_{T_0}, g_1}_{\text{subgoal 1}} \;, \ldots, \underbrace{s_{T_i}, g_{i+1}}_{\text{subgoal } i}, \ldots, \underbrace{s_{T_{\mathcal{T}-1}}, g_{\mathcal{T}}}_{\text{subgoal } \mathcal{T}} \Big) \tag{5.1}$$

for the master policy. Notice that the terminal state of the i^{th} sub-policy s_{T_i} forms the state for the master policy to predict the next subgoal g_{i+1}. For the $(i+1)^{\text{th}}$ subgoal g_{i+1}, the low-level trajectory of states and primitive actions is given by:

$$\sigma_{g_{i+1}} = \Big(\underbrace{s_{T_i}, a_{T_i}}_{\text{action 0}}, \underbrace{s_{T_i+1}, a_{T_i+1}}_{\text{action 1}}, \ldots, \underbrace{s_{T_i+t}, a_{T_i+t}}_{\text{action t}}, \ldots, s_{T_{i+1}} \Big). \tag{5.2}$$

Note that by concatenating all sub-policy trajectories in order $(\sigma_{g_0}, \sigma_{g_1}, \ldots, \sigma_{g_{\mathcal{T}}})$, the entire trajectory of states and primitive actions can be recovered.

Subgoals $\langle$Tasks, Arguments$\rangle$. As mentioned above, each subgoal is factorized into a task and an argument $g = \langle g_{\text{task}}, g_{\text{argument}} \rangle$. There are 4 possible tasks – `exit-room`, `find-room`, `find-object`, and `answer`. Tasks `find-object` and `find-room` accept as arguments one of the

Table 5.1: Subgoals and conditions used to automatically extract them from expert trajectories.

Subgoal	Argument(s)	Description	Success
`Exit-room`	None	When there is only 1 door in spawn room, or 1 door other than door entered through in an intermediate room; agent is forced to use the remaining door.	Stopping after exiting through the correct door.
`Find-room`	Room name (*gym, kitchen, ...*)	When there are multiple doors and the agent has to search and pick the door to the target room.	Stopping after entering target room.
`Find-object`	Object name (*oven, sofa, ...*)	When the agent has to search for a specific object in room.	Stopping within 0.75m of the target object.
`Answer`	None	When the agent has to provide an answer from the answer space.	Generating the correct answer to the question.

50 objects and 12 room types in EQA v1 dataset [115] respectively; `exit-room` and `answer` do not accept any arguments. This gives us a total of $50 + 12 + 1 + 1 = 64$ subgoals.

$$\left.\begin{array}{l}\langle \texttt{exit-room,none}\rangle, \quad \langle \texttt{answer,none}\rangle, \\ \langle \texttt{find-object,couch}\rangle, \langle \texttt{find-object,cup}\rangle, \ldots, \langle \texttt{find-object,xbox}\rangle, \\ \langle \texttt{find-room,living}\rangle, \langle \texttt{find-room,bedroom}\rangle, \ldots, \langle \texttt{find-room,patio}\rangle.\end{array}\right\} \begin{array}{l}\text{0 args} \\ \text{50 args} \\ \text{12 args}\end{array}$$

Descriptions of these tasks and their success criteria are provided in Table 5.1.

Master Policy. The master policy ß_θ parameterized by θ is implemented as a single layer Gated Recurrent Unit (GRU). At each high-level step $i+1$, the master policy $\text{ß}_\theta(g|s_{T_i})$ takes as input the concatenation of a encoding of the question $q \in \mathbb{R}^{128}$, the image feature $v_{T_i} \in \mathbb{R}^{128}$ of the current frame and an encoding $o_i \in \mathbb{R}^{32}$ computed from a 1-hot representation of the i^{th} subgoal, *i.e.* $\mathbb{1}(g_i)$. This information is used to update the hidden state $h_i \in \mathbb{R}^{1048}$ that encodes the entire trajectory up to time t and serves as the state representation. The policy then produces a probability distribution over all possible (64) subgoals $\mathcal{G}$. We train these policies with actor-critic methods and thus the network also produces a value estimate.

Sub-policies. To take advantage of the comparatively lower number of subgoal tasks,

we decompose sub-policy parameters ϕ_g into $\phi_{g_{\text{task}}}$ and $\phi_{g_{\text{argument}}}$, where $\phi_{g_{\text{task}}}$ are shared across the same task and $\phi_{g_{\text{argument}}}$ is an argument specific embedding. Parameter sharing enables us to learn the shared task in a sample-efficient manner, rather than exhaustively learning separate sub-policies for each combination.

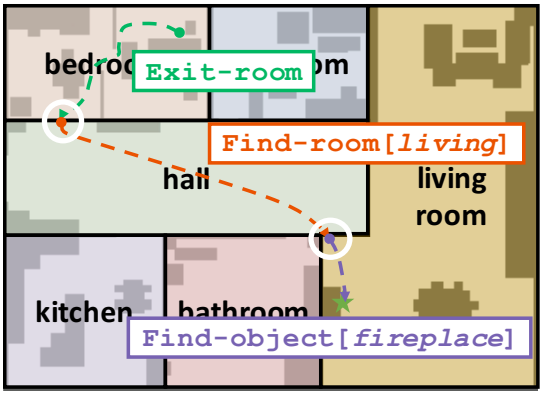
Like the master policy, each sub-policy π_{ϕ_g} is implemented as a single-layer GRU. At each low-level time step t, a sub-policy $\pi_{\phi_g}(a|s_t)$ takes as input the concatenation of the image feature $v_t \in \mathbb{R}^{128}$ of the current frame, an encoding $p_{t-1} \in \mathbb{R}^{32}$ computed from a 1-hot representation of the previous primitive action *i.e.* $\mathbb{1}(a_{t-1})$, and the argument embedding $\phi_{g_{\text{argument}}}$. These inputs are used to update the hidden state $h_t^g \in \mathbb{R}^{1048}$ which serves as the state representation. The policy then outputs a distribution over primitive actions (`forward`, `turn-left`, `turn-right`, `stop`). As with the master policy, each sub-policy also output a value estimate. shows this model structure.

Perception and Question Answering. To ensure fair comparisons to prior work, we use the same perception and question answering models as in Chapter 4 [115]. The perception model is a simple convolutional neural network trained to perform auto-encoding, semantic segmentation, and depth estimation from RGB frames taken from House3D [107]. Like [115], we use the bottleneck layer of this model as a fixed feature extractor. We also use the same post-navigational question-answering model as [115], which encodes the question with a 2-layer LSTM and performs dot-product based attention between the question encoding and the image features from the last five frames along the navigation path right before the `answer` module is called. This post-navigational answering module is trained using visual features along the shortest path trajectories and then frozen. By keeping these parts of the architecture identical to [115], our experimental comparisons can focus on the differences only due to our contributions, the Neural Modular Controller.

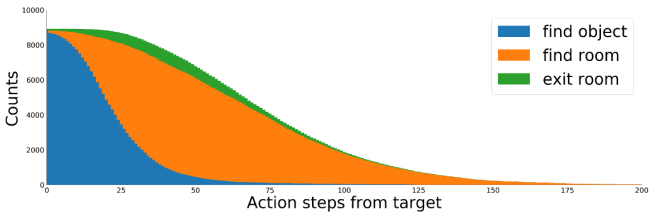
5.3.2 Hierarchical Behavior Cloning from Expert Trajectories

The questions in EQA v1 dataset [115] (*e.g.* *'What color is the fireplace?'*) are constructed to inquire about attributes (color, location, *etc.*) of specific target objects (*'fireplace'*). This notion of a target enables the construction of an automatically generated *expert trajectory* $(s_0^*, a_0^*, \ldots, s_T^*, a_T^*)$ – the states and actions along the shortest path from the agent spawn location to the object of interest specified in the question. Notice that these shortest paths may only be used as supervision on training environments but may not be utilized during evaluation on test environments (where the agent must operate from egocentric vision alone).

Specifically, we would like to use these expert demonstrations to pre-train our proposed NMC navigator using behavior cloning. However, these trajectories $(s_0^*, a_0^*, \ldots, s_T^*, a_T^*)$ correspond to a series of primitive actions. To provide supervision for both the master policy and sub-policies, these shortest-path trajectories must be annotated with a sequence of subgoals and segmented into their respective temporal extents, resulting in Σ^* and $(\sigma_{g_i}^*)$.



(a) **Q**: What color is the fireplace? **A**: Brown



(b) Distribution of subgoals with number of actions from the target object as per expert plans. Closer to the target object, the expert plan predominantly consists of `Find-object`, while as we move farther away, the proportion of `Find-room` and `Exit-room` goes up.

Figure 5.2: We extract expert subgoal trajectories from shortest paths by dividing paths on room transition boundaries (circled in (a))and following the rules in Tab. 5.1.

We automate this 'lifting' of annotation up the hierarchy by leveraging the object and room bounding boxes provided by the House3D [107]. Essentially, a floor plan may be viewed as an undirected graph with rooms as nodes and doorways as edges connecting a pair of adjacent rooms. An example trajectory is shown in Fig. 5.2a for the question *'What color is the fireplace?'*. The agent is spawned in a bedroom, the shortest path exits into the hall, enters the living room, and approaches the fireplace. We convert this trajectory to the subgoal sequence (`exit-room`, `find-room[living]`, `find-object[fireplace]`, `answer`) by recording the transitions on the shortest path from one room to another, which also naturally provides us with temporal extents of these subgoals.

We follow a couple of subtle but natural rules: (1) `find-object` is tagged only when the agent has reached the destination room containing the target object; and (2) `exit-room` is tagged only when the 'out-degree' of the current room in the floor-plan-graph is exactly 1 (*i.e.* either the current room has exactly one doorway or the current room has two doorways but the agent came in through one). Rule (2) ensures a semantic difference between `exit-room` and `find-room` – informally, `exit-room` means *'get me out of here'* and `find-room[name]` means *'look for room* `name`*'*.

Tab. 5.1 summarizes these subgoals and the heuristics used to automatically extract them

from navigational paths. Fig. 5.2b shows the proportions of these subgoals in expert trajectories as a function of the distance from target object. Notice that when the agent is close to the target, it is likely to be within the same room as the target and thus `find-object` dominates. On the other hand, when the agent is far away from the target, `find-room` and `exit-room` dominate.

We perform this lifting of shortest paths for all training set questions in EQA v1 dataset [115], resulting in N expert trajectories $\{\Sigma^*_n\}_{n=1}^N$ for the master policy and $K(>> N)$ trajectories $\{\sigma^*_{g_k}\}_{k=1}^K$ for sub-policies. We can then perform hierarchical behavior cloning by minimizing the sum of cross-entropy losses over all decisions in all expert trajectories. As is typical in maximum-likelihood training of directed probabilistic models (*e.g.* hierarchical Bayes Nets), full supervision results in decomposition into independent sub-problems. Specifically, with a slight abuse of notation, let $(s^*_i, g^*_{i+1}) \in \Sigma^*$ denote an iterator over all state-subgoal tuples in Σ^*, and $\sum_{(s^*_i, g^*_{i+1}) \in \Sigma^*}$ denote a sum over such tuples.

Now, the independent learning problems can be written as:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{n=1}^{N} \sum_{(s^*_i, g^*_{i+1}) \in \Sigma^*_n} -\log\left(\pi_\theta(g^*_{i+1} | s^*_i)\right) \qquad \text{(master policy cloning)} \quad (5.3a)$$

$$\phi^*_g = \underset{\phi}{\operatorname{argmin}} \underbrace{\sum_{k=1}^{K} [\![g_k = g]\!]}_{\text{demonstrations}} \underbrace{\sum_{(s^*_t, a^*_{t+1}) \in \sigma^*_{g_k}}}_{\text{transitions}} \underbrace{-\log\left(\pi_{\phi_g}(a^*_{t+1} | s^*_t)\right)}_{\text{negative-log-likelihood}} \qquad \text{(sub-policy cloning)} \quad (5.3b)$$

Intuitively, each sub-policy independently maximizes the conditional probability of actions observed in the expert demonstrations, and the master policy essentially trains assuming perfect sub-policies.

5.3.3 Asynchronous Advantage Actor-Critic (A3C) Training

After the independent behavior cloning stage, the policies have learned to mimic expert trajectories; however, they have not had to coordinate with each other or recover from their own navigational errors. As such, we fine-tune them with reinforcement learning – first independently and then jointly.

Reward Structure. The ultimate goal of our agent is to answer questions accurately; however, doing so requires navigating the environment sufficiently well in search of the answer. We mirror this structure in our reward R, decomposing it into a sum of a sparse

terminal reward R_{terminal} for the final outcome and a dense, shaped reward R_{shaped} [103] determined by the agent's progress towards its goals. For the master policy π_θ, we set R_{terminal} to be 1 if the model answers the question correctly and 0 otherwise. The shaped reward R_{shaped} at master-step i is based on the change of navigable distance to the target object before and after executing subgoal g_i. Each sub-policy π_{ϕ_g} also has a terminal 0/1 reward R_{terminal} for stopping in a successful state, *e.g.* `Exit-room` ending outside the room it was called in (see Tab. 5.1 for all success definitions). Like the master policy, R_{shaped} at time t is set according to the change in navigable distance to the sub-policy target (*e.g.* a point just inside a living room for `find-room[living]`) after executing the primitive action a_t. Further, sub-policies are also penalized a small constant (-0.02) for colliding with obstacles.

Policy Optimization. We update the master and sub-policies to maximize expected discounted future rewards $J(\pi_\theta)$ and $J(\pi_{\phi_g})$ respectively through the Asynchronous Advantage Actor Critic [126] policy-gradient algorithm. Specifically, for the master policy, the gradient of the expected reward is written as:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}\left[\nabla_\theta \log(\pi_\theta(g_i|s_{T_i}))\left(Q(s_{T_i}, g_i) - c_\theta(s_{T_i})\right)\right] \tag{5.4}$$

where $c_\theta(s_{T_i})$ is the estimated value of s_{T_i} produced by the critic for π_θ. To further reduce variance, we follow [139] and estimate $Q(s_{T_i}, g_i) \approx R_\theta(s_{T_i}) + \gamma c_\theta(s_{T_{i+1}})$ such that $Q(s_{T_i}, g_i) - c_\theta(s_{T_i})$ computes a generalized advantage estimator (GAE). Similarly, each sub-policy π_{ϕ_g} is updated according to the gradient

$$\nabla_{\phi_g} J(\pi_{\phi_g}) = \mathbb{E}\left[\nabla_\theta \log(\pi_{\phi_g}(a_i|s_i))\left(Q(s_i, a_i) - c_{\phi_g}(s_i)\right)\right]. \tag{5.5}$$

Recall from Section 5.3.1 that these critics share parameters with their corresponding policy networks such that subgoals with a common task also share a critic. We train each policy network independently using A3C [126] with GAE [139] with 8 threads across 4 GPUs. After independent reinforcement fine-tuning of the sub-policies, we train the master policy further using the trained sub-policies rather than expert subgoal trajectories.

Initial states and curriculum. Rather than spawn agents at fixed distances from target, from where accomplishing the subgoal may be arbitrarily difficult, we sample locations along expert trajectories for each question or subgoal. This ensures that even early in training, policies are likely to have a mix of positive and negative reward episodes. At the beginning of training, all points along the trajectory are equally likely; however, as training progresses and success rate improves, we reduce the likelihood of sampling points

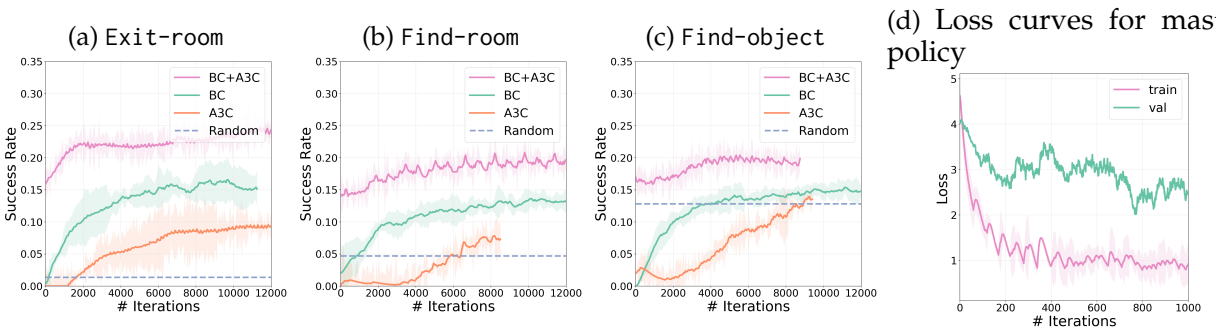


Figure 5.3: (a,b,c) Success rate over training iterations for each sub-policy task using behavior cloning (BC), reinforcement learning from scratch (A3C), and reinforcement finetuning after behavior cloning (BC+A3C) training regimes. We find BC+A3C significantly outperforms either BC or A3C alone. Each of these is averaged over 5 runs. (d) Losses for master policy during behavior cloning *i.e.* assuming access to perfect sub-policies.

nearer to the goal. This is implemented as a multiplier α on available states $[s_0, s_1, ..., s_{\alpha T}]$, initialized to 1.0 and scaled by 0.9 whenever success rate crosses a 40% threshold.

5.4 Experiments and Results

Dataset. We benchmark performance on the EQA v1 dataset [115], which contains $\sim$9,000 questions in 774 environments – split into 7129(648) / 853(68) / 905(58) questions (environments) for training/validation/testing respectively[1]. These splits have no overlapping environments between them, thus strictly checking for generalization to novel environments. We follow the same splits.

Evaluating sub-policies. We begin by evaluating the performance of each sub-policy with regard to its specialized task. For clarity, we break results down by subgoal task rather than for each task-argument combination. We compare sub-policies trained with behavior cloning (**BC**), reinforcement learning from scratch (**A3C**), and reinforcement fine-tuning after behavior cloning (**BC+A3C**). We also compare to a **random** agent that uniformly samples actions including `stop` to put our results in context. For each, we report the success rate (as defined in Tab. 5.1) on the EQA v1 validation set which consists of 68 novel environments unseen during training. We spawn sub-policies at randomly selected suitable rooms (*i.e.* `Find-object[sofa]` will only be executed in a room with a sofa) and allow them to execute for a maximum episode length of 50 steps or until they terminate.

Fig. 5.3 shows success rates for the different subgoal tasks over the course of training. We observe that:

[1]Note that the size of the publicly available dataset on embodiedqa.org/data is larger than the one reported in the original version of the paper due to changes in labels for color questions.

Table 5.2: Evaluation of EmbodiedQA agents on navigation and answering metrics for the EQA v1 test set.

	Navigation									QA		
	$\mathbf{d_0}$ (For reference)			$\mathbf{d_T}$ (Lower is better)			$\mathbf{d_\Delta}$ (Higher is better)			**accuracy** (Higher is better)		
	T_{-10}	T_{-30}	T_{-50}	T_{-10}	T_{-30}	T_{-50}	T_{-10}	T_{-30}	T_{-50}	T_{-10}	T_{-30}	T_{-50}
PACMAN (BC) [115]	1.15	4.87	9.64	1.19	4.25	8.12	-0.04	0.62	1.52	48.48%	40.59%	39.87%
PACMAN (BC+REINFORCE) [115]	1.15	4.87	9.64	**1.05**	4.22	8.13	**0.10**	0.65	1.51	50.21%	42.26%	40.76%
NMC (BC)	1.15	4.87	9.64	1.44	4.14	8.43	-0.29	0.73	1.21	43.14%	41.96%	38.74%
NMC (BC+A3C)	1.15	4.87	9.64	1.06	**3.72**	**7.94**	0.09	**1.15**	**1.70**	**53.58%**	**46.21%**	**44.32%**

- **Behavior cloning (BC) is more sample-efficient than A3C from scratch.** Sub-policies trained using BC improve significantly faster than A3C for all tasks, and achieve higher success rates for `Exit-room` and `Find-room`. Interestingly, this performance gap is larger for tasks where a random policy does *worse* – implying that BC helps more as task complexity increases.
- **Reinforcement Fine-Tuning with A3C greatly improves over BC training alone.** Initializing A3C with a policy trained via behavior cloning results in a model that significantly outperforms either approach on its own, nearly doubling the success rate of behavior cloning for some tasks. Intuitively, mimicking expert trajectories in behavior cloning provides dense feedback for agents about how to navigate the world; however, agents never have to face the consequences of erroneous actions *e.g.* recovering from collisions with objects – a weakness that A3C fine-tuning addresses.

Evalating master policy. Next, we evaluate how well the master policy performs during independent behavior cloning on expert trajectories *i.e.* assuming perfect sub-policies, as specified in Eq. 5.3a. Even though there is no overlap between training and validation environments, the master policy is able to generalize reasonably and gets $\sim$ 48% intersection-over-union (IoU) with ground truth subgoal sequences on the validation set. Note that a sequence of sub-goals that is different from the one corresponding to the shortest path may still be successful at navigating to the target object and answering the question correctly. In that sense, IoU against ground truth subgoal sequences is a strict metric. Fig. 5.3d shows the training and validation cross-entropy loss curves for the master policy.

Evalating NMC. Finally, we put together the master and sub-policies and evaluate navigation and question answering performance on EmbodiedQA. We compare against the PACMAN model proposed in [115]. For accurate comparison, both PACMAN and NMC use the same publicly available and frozen pretrained CNN[2], and the same visual ques-

[2] `github.com/facebookresearch/EmbodiedQA`

tion answering model – pretrained to predict answers from last 5 observations of expert trajectories, following [115]. Agents are evaluated by spawning 10, 30, or 50 primitive actions away from target, which corresponds to distances of 1.15, 4.87, and 9.64 meters from target respectively, denoted by $\mathbf{d_0}$ in Tab. 5.2. When allowed to run free from this spawn location, $\mathbf{d_T}$ measures final distance to target (how far is the agent from the goal at termination), and $\mathbf{d_\Delta} = \mathbf{d_T} - \mathbf{d_0}$ evaluates change in distance to target (how much progress does the agent make over the course of its navigation). Answering performance is measured by **accuracy** (*i.e.* did the predicted answer match ground-truth). Note that [115] report a number of additional metrics (percentage of times the agent stops, retrieval evaluation of answers, *etc.*). Accuracies for PACMAN are obtained by running the publicly available codebase accompanying [115], and numbers are different than those reported in the original version of [115] due to changes in the dataset[1].

As shown in Tab. 5.2, we evaluate two versions of our model – 1) NMC (BC) naively combines master and sub-policies without A3C finetuning at any level of hierarchy, and 2) NMC (BC+A3C) is our final model where each stage is trained with BC+A3C, as described in Sec. 5.3. As expected, NMC (BC) performs worse than NMC (BC + A3C), evident in worse navigation $\mathbf{d_T}$, $\mathbf{d_\Delta}$ and answering **accuracy**. PACMAN (BC) and NMC (BC) go through the same training regime, and there are no clear trends as to which is better – PACMAN (BC) has better $\mathbf{d_\Delta}$ and answering **accuracy** at T_{-10} and T_{-50}, but worse at T_{-30}. No A3C finetuning makes it hard for sub-policies to recover from erroneous primitive actions, and for master policy to adapt to sub-policies. A3C finetuning significantly boosts performance, *i.e.* NMC (BC + A3C) outperforms PACMAN with higher $\mathbf{d_\Delta}$ (makes more progress towards target), lower $\mathbf{d_T}$ (terminates closer to target), and higher answering **accuracy**. This gain primarily comes from the choice of subgoals and the master policy's ability to explore over this space of subgoals instead of primitive actions (as in PACMAN), enabling the master policy to operate over longer time horizons, critical for sparse reward settings as in EmbodiedQA.

5.5 Conclusion

We introduced Neural Modular Controller (NMC), a hierarchical policy for EmbodiedQA consisting of a master policy that proposes a sequence of semantic subgoals from question (*e.g. 'What color is the sofa in the living room?'* → `Find-room[living]`, `Find-object[sofa]`, `Answer`), and specialized sub-policies for executing each of these tasks. The master and sub-policies are trained using a combination of behavior cloning and reinforcement learning, which

is dramatically more sample-efficient than each individual training regime. In particular, behavior cloning provides dense feedback for how to navigate, and reinforcement learning enables policies to deal with consequences of their actions, and recover from errors. The efficacy of our proposed model is demonstrated on the EQA v1 dataset [115], where NMC outperforms prior work both in navigation and question answering.

Part III

Multi-Agent Populations That Can See, Talk, and Act

Chapter 6

TarMAC: Targeted Multi-Agent Communication

6.1 Introduction

Finally, in addition to being able to see, talk (to humans), and act, intelligent agents in real-world scenarios should also be able to *comunicate* with each other, to coordinate, strategize and utilize their combined sensory experiences and act in the physical world. The ability to communicate has wide-ranging applications for artificial agents – from multi-player gameplay in simulated games (*e.g.* DoTA, Quake, StarCraft) or physical worlds (*e.g.* robot soccer), to networks of self-driving cars communicating with each other to achieve safe and swift transport, to teams of robots on search-and-rescue missions deployed in hostile and fast-evolving environments.

A salient property of human communication is the ability to hold *targeted* interactions. Rather than the 'one-size-fits-all' approach of broadcasting messages to all participating agents, as has been previously explored [1, 96], it can be useful to send/receive certain messages to/from specific recipients/senders. This enables a more flexible collaboration strategy in complex environments. For example, within a team of search-and-rescue robots with a diverse set of roles and goals, a message for a fire-fighter ("smoke is coming from the kitchen") is largely meaningless for a bomb-defuser.

In this work, we develop a collaborative multi-agent deep reinforcement learning approach that supports targeted communication. Crucially, each individual agent *actively selects* which other agents to send messages to. This targeted communication behavior is operationalized via a simple signature-based soft attention mechanism: along with the message, the sender broadcasts a key which encodes properties of agents the message is intended for, and is used by receivers to gauge the relevance of the message. This communication mechanism is learned implicitly, without any attention supervision, as a result of

This chapter is based on work done during an internship at Facebook AI Research.

end-to-end training using a downstream task-specific team reward.

The inductive bias provided by soft attention in the communication architecture is sufficient to enable agents to 1) communicate agent-goal-specific messages (*e.g*. guide firefighter towards fire, bomb-defuser towards bomb, *etc*.), 2) be adaptive to variable team sizes (*e.g*. the size of the local neighborhood a self-driving car can communicate with changes as it moves), and 3) be interpretable through predicted attention probabilities that allow for inspection of *which* agent is communicating *what* message and to *whom*.

Our results however show that just using targeted communication is not enough. Complex real-world tasks might require *large populations of agents* to go through *multiple stages of collaborative communication and reasoning*, involving large amounts of information to be *persistent in memory* and exchanged via *high-bandwidth communication channels*. To this end, our actor-critic framework combines centralized training with decentralized execution [140], thus enabling scaling to a large number of agents. In this context, our inter-agent communication architecture supports multiple stages of targeted interactions at every time-step, and the agents' recurrent policies support persistent relevant information in internal states.

While natural language, *i.e*. a finite set of discrete tokens with pre-specified human-conventionalized meanings, may seem like an intuitive protocol for inter-agent communication – one that enables human-interpretability of interactions – forcing machines to communicate among themselves in discrete tokens presents additional training challenges. Since our work focuses on machine-only multi-agent teams, we allow agents to communicate via continuous vectors (rather than discrete symbols), and via the learning process, agents have the flexibility to discover and optimize their communication protocol as per task requirements.

We provide extensive empirical demonstration of the efficacy of our approach across a range of tasks, environments, and team sizes. We begin by benchmarking multi-agent communication with and without attention on a cooperative navigation task derived from the SHAPES environment [128]. We show that agents learn intuitive attention behavior across a spectrum of task difficulties. Next, we evaluate the same targeted multi-agent communication architecture on the traffic junction environment [1], and show that agents are able to adaptively focus on 'active' agents in the case of varying team sizes. Finally, we demonstrate effective multi-agent communication in 3D environments on a cooperative first-person point-goal navigation task in the rich House3D environment [107].

6.2 Related Work

Multi-agent systems fall at the intersection of game theory, distributed systems, and Artificial Intelligence in general [141], and thus have a rich and diverse literature. Our work builds on and is related to prior work in deep multi-agent reinforcement learning, the centralized training and decentralized execution paradigm, and emergent communication protocols.

Multi-Agent Reinforcement Learning (MARL). Within MARL (see [142] for a survey), our work is related to recent efforts on using recurrent neural networks to approximate agent policies [143], algorithms stabilizing multi-agent training [140, 144], and tasks in novel application domains such as coordination and navigation in 3D simulated environments [145–147].

Centralized Training & Decentralized Execution. Both [1] and [148] adopt a fully centralized framework at both training and test time – a central controller processes local observations from all agents and outputs a probability distribution over joint actions. In this setting, any controller (*e.g.* a fully-connected network) can be viewed as implicitly encoding communication. [1] present an efficient architecture to learn a centralized controller invariant to agent permutations – by sharing weights and averaging as in [149]. Meanwhile [148] proposes to replace averaging by an attentional mechanism to allow targeted interactions between agents. While closely related to our communication architecture, his work only considers fully supervised one-next-step prediction tasks, while we tackle the full reinforcement learning problem with tasks requiring planning over long time horizons.

Moreover, a centralized controller quickly becomes intractable in real-world tasks with many agents and high-dimensional observation spaces (*e.g.* navigation in House3D [107]). To address these weaknesses, we adopt the framework of centralized learning but decentralized execution (following [96, 140]) and further relax it by allowing agents to communicate. While agents can use extra information during training, at test time, they pick actions solely based on local observations and communication messages received from other agents.

Finally, we note that fully decentralized execution at test time *without communication* is very restrictive. It means 1) each agent must act myopically based solely on its local observation and 2) agents cannot coordinate their actions. In our setting, communication between agents offers a reasonable trade-off between allowing agents to globally coordinate while retaining tractability (since the communicated messages are much lower-

Table 6.1: Comparison with prior work on cooperative multi-agent continuous communication.

	Decentralized Execution	Targeted Communication	Multi-Stage Decisions	Reinforcement Learning
DIAL [96]	Yes	No	No	Yes (Q-Learning)
CommNets [1]	No	No	Yes	Yes (REINFORCE)
VAIN [148]	No	Yes	Yes	No (Supervised)
ATOC [153]	Yes	No	No	Yes (Actor-Critic)
TarMAC (this paper)	Yes	Yes	Yes	Yes (Actor-Critic)

dimensional than the observation space).

Emergent Communication Protocols. Our work is also related to recent work on learning communication protocols in a completely end-to-end manner with reinforcement learning – from perceptual input (*e.g.* pixels) to communication symbols (discrete or continuous) to actions (*e.g.* navigating in an environment). While [96, 97, 99, 150–152] constrain agents to communicate with discrete symbols with the explicit goal to study emergence of language, our work operates in the paradigm of learning a continuous communication protocol in order to solve a downstream task [1, 148, 153]. While [153] also operate in a decentralized execution setting and use an attentional communication mechanism, their setup is significantly different from ours as they use attention to decide *when* to communicate, not *who* to communicate with ('who' depends on a hand-tuned neighborhood parameter in their work). Table 6.1 summarizes the main axes of comparison between our work and previous efforts in this exciting space.

6.3 Technical Background

Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs). A Dec-POMDP is a cooperative multi-agent extension of a partially observable Markov decision process ([154]). For N agents, it is defined by a set of states S describing possible configurations of all agents, a global reward function R, a transition probability function T, and for each agent $i \in 1, ..., N$ a set of allowed actions A_i, a set of possible observations Ω_i and an observation function O_i. Operationally, at each time step every agent picks an action a_i based on its local observation ω_i following its own stochastic policy $\pi_{\theta_i}(a_i|\omega_i)$. The system randomly transitions to the next state s' given the current state and joint action $T(s'|s, a_1, ..., a_N)$. The agent team receives a global reward $r = R(s, a_1, ..., a_N)$ while each agent receives a local observation of the new state $O_i(\omega_i|s')$. Agents aim to maximize the total expected return $J = \sum_{t=0}^{T} \gamma^t r_t$ where γ is a discount factor and T is the episode time

horizon.

Actor-Critic Algorithms. Policy gradient methods directly adjust the parameters θ of the policy in order to maximize the objective $J(\theta) = \mathbb{E}_{s\sim p_\pi, a\sim\pi_\theta(s)}\left[R(s,a)\right]$ by taking steps in the direction of $\nabla J(\theta)$. We can write the gradient with respect to the policy parameters as

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim p_\pi, a\sim\pi_\theta(s)}\left[\nabla_\theta \log \pi_\theta(a|s) Q_\pi(s,a)\right],$$

where $Q_\pi(s,a)$ is called the action-value, it is the expected remaining discounted reward if we take action a in state s and follow policy π thereafter. Actor-Critic algorithms learn an approximation of the unknown true action-value function $\hat{Q}(s,a)$ by e.g. temporal-difference learning [155]. This $\hat{Q}(s,a)$ is called the Critic while the policy π_θ is called the Actor.

Multi-Agent Actor-Critic. [140] propose a multi-agent Actor-Critic algorithm adapted to centralized learning and decentralized execution. Each agent learns its own individual policy $\pi_{\theta_i}(a_i|\omega_i)$ conditioned on local observation ω_i, using a centralized Critic which estimates the joint action-value $\hat{Q}(s, a_1, ..., a_N)$.

6.4 TarMAC: Targeted Multi-Agent Communication

We now describe our multi-agent communication architecture in detail. Recall that we have N agents with policies $\{\pi_1, ..., \pi_N\}$, respectively parameterized by $\{\theta_1, ..., \theta_N\}$, jointly performing a cooperative task. At every timestep t, the ith agent for all $i \in \{1, ..., N\}$ sees a local observation ω_i^t, and must select a discrete environment action $a_i^t \sim \pi_{\theta_i}$ and a continuous communication message m_i^t, received by other agents at the next timestep, in order to maximize global reward $r_t \sim R$. Since no agent has access to the underlying state of the environment s_t, there is incentive in communicating with each other and being mutually helpful to do better as a team.

Policies and Decentralized Execution. Each agent is essentially modeled as a Dec-POMDP augmented with communication. Each agent's policy π_{θ_i} is implemented as a 1-layer Gated Recurrent Unit [156]. At every timestep, the local observation ω_i^t and a vector c_i^t aggregating messages sent by all agents at the previous timestep (described in more detail below) are used to update the hidden state h_i^t of the GRU, which encodes the entire message-action-observation history up to time t. From this internal state representation, the agent's policy $\pi_{\theta_i}\left(a_i^t \mid h_i^t\right)$ predicts a categorical distribution over the space of actions, and another output head produces an outgoing message vector m_i^t. Note that for all

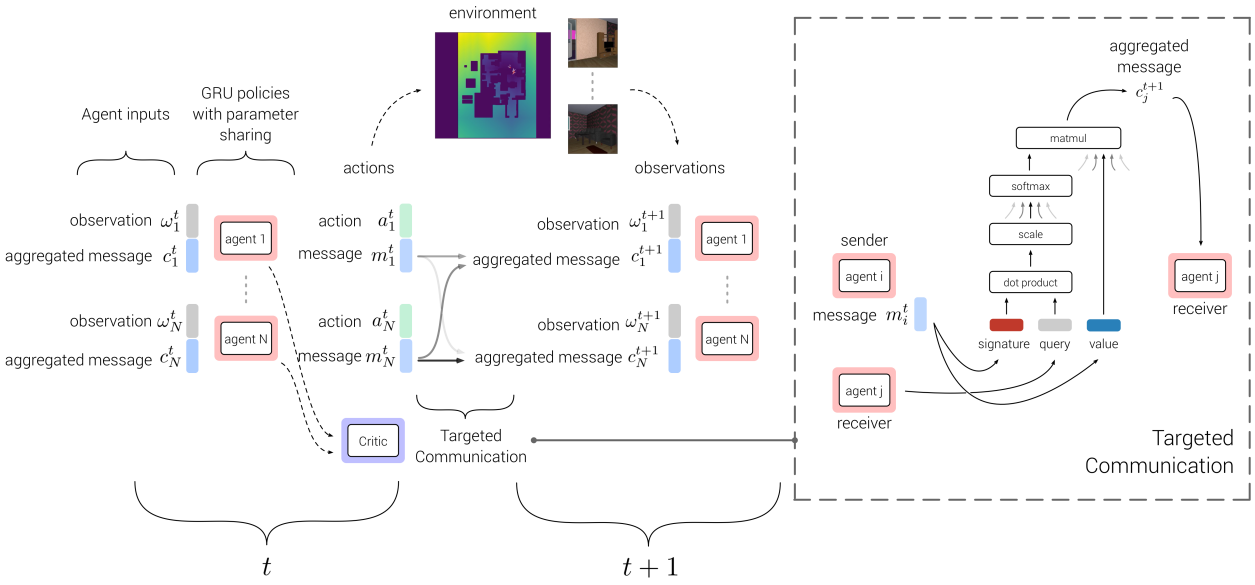


Figure 6.1: Overview of our multi-agent architecture with targeted communication. Left: At every timestep, each agent policy gets a local observation ω_i^t and aggregated message c_i^t as input, and predicts an environment action a_i^t and a targeted communication message m_i^t. Right: Targeted communication between agents is implemented as a signature-based soft attention mechanism. Each agent broadcasts a message m_i^t consisting of a signature k_i^t, which can be used to encode agent-specific information and a value v_i^t, which contains the actual message. At the next timestep, each receiving agent gets as input a convex combination of message values, where the attention weights are obtained by a dot product between sender's signature k_i^t and a query vector q_j^{t+1} predicted from the receiver's hidden state.

our experiments, agents are symmetric and policies are instantiated from the same set of shared parameters; *i.e.* $\theta_1 = ... = \theta_N$. This considerably speeds up learning.

Centralized Critic. Following prior work [140, 144], we operate under the centralized learning and decentralized execution paradigm wherein during training, a centralized critic guides the optimization of individual agent policies. The centralized Critic takes as input predicted actions $\{a_1^t, ..., a_N^t\}$ and internal state representations $\{h_1^t, ..., h_N^t\}$ from all agents to estimate the joint action-value $\hat{Q}_t$ at every timestep. The centralized Critic is learned by temporal difference [155] and the gradient of the expected return $J(\theta_i) = \mathbb{E}[R]$ with respect to policy parameters is approximated by:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}\left[\nabla_{\theta_i} \log \pi_{\theta_i}(a_i^t|h_i^t)\ \hat{Q}_t(h_1^t, ..., h_N^t, a_t^1, ..., a_t^N)\right].$$

Note that compared to an individual critic $\hat{Q}_i(h_i^t, a_i^t)$ for each agent, having a centralized critic leads to considerably lower variance in policy gradient estimates since it takes into

account actions from all agents. At test time, the critic is not needed anymore and policy execution is fully decentralized.

Targeted, Multi-Stage Communication. Establishing complex collaboration strategies requires targeted communication *i.e.* the ability to send specific messages to specific agents, as well as multi-stage communication *i.e.* multiple rounds of back-and-forth interactions between agents. We use a signature-based soft-attention mechanism in our communication structure to enable targeting. Each message m_i^t consists of 2 parts – a signature $k_i^t \in \mathbb{R}^{d_k}$ to target recipients, and a value $v_i^t \in \mathbb{R}^{d_v}$:

$$m_i^t = [\ \overbrace{k_i^t}^{\text{signature}} \quad \underbrace{v_i^t}_{\text{value}}\ ]. \tag{6.1}$$

At the receiving end, each agent (indexed by j) predicts a query vector $q_j^{t+1} \in \mathbb{R}^{d_k}$ from its hidden state h_j^{t+1} and uses it to compute a dot product with signatures of all N messages. This is scaled by $1/\sqrt{d_k}$ followed by a softmax to obtain attention weight α_{ji} for each message value vector:

$$\mathbf{ff}_j = \text{softmax}\left[\ \frac{{q_j^{t+1}}^T k_1^t}{\sqrt{d_k}}\ \ldots\ \underbrace{\frac{{q_j^{t+1}}^T k_i^t}{\sqrt{d_k}}}_{\alpha_{ji}}\ \ldots\ \frac{{q_j^{t+1}}^T k_N^t}{\sqrt{d_k}}\ \right] \tag{6.2}$$

$$c_j^{t+1} = \sum_{i=1}^{N} \alpha_{ji} v_i^t. \tag{6.3}$$

Note that (6.2) also includes α_{ii} corresponding to the ability to *self-attend* [157], which we empirically found to improve performance, especially in situations when an agent has found the goal in a coordinated navigation task and all it is required to do is stay at the goal, so others benefit from attending to this agent's message but return communication is not needed.

For multiple stages of communication, aggregated message vector c_j^{t+1} and internal state h_j^t are first used to predict the next internal state ${h'}_j^t$ taking into account a first round of communication:

$${h'}_j^t = \tanh\left(W_{h \to h'}[\ c_j^{t+1} \parallel h_j^t\ ]\right). \tag{6.4}$$

Next, ${h'}_j^t$ is used to predict signature, query, value followed by repeating Eqns 6.1-6.4 for multiple rounds until we get a final aggregated message vector c_j^{t+1} to be used as input

at the next timestep.

6.5 Experiments

We evaluate our targeted multi-agent communication architecture on a variety of tasks and environments. All our models were trained with a batched synchronous version of the multi-agent Actor-Critic described above, using RMSProp with a learning rate of 7×10^{-4} and $\alpha = 0.99$, batch size 16, discount factor $\gamma = 0.99$ and entropy regularization coefficient 0.01 for agent policies. All our agent policies are instantiated from the same set of shared parameters; *i.e.* $\theta_1 = ... = \theta_N$. Each agent's GRU hidden state is 128-d, message signature/query is 16-d, and message value is 32-d (unless specified otherwise). All results are averaged over 5 independent runs with different seeds.

6.5.1 SHAPES

The SHAPES dataset was introduced by [128][1], and originally created for testing compositional visual reasoning for the task of visual question answering. It consists of synthetic images of 2D colored shapes arranged in a grid (3×3 cells in the original dataset) along with corresponding question-answer pairs. There are 3 shapes (circle, square, triangle), 3 colors (red, green, blue), and 2 sizes (small, big) in total (see Fig. 6.2).

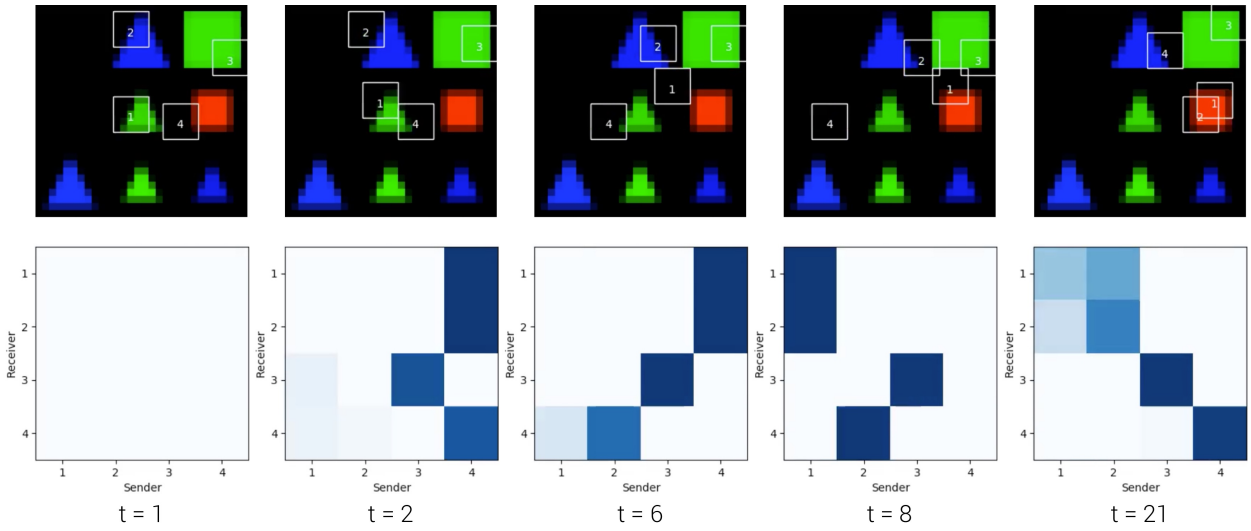
We convert each image from SHAPES into an active environment where agents can now be spawned at different regions of the image, observe a 5×5 local patch around them and their coordinates, and take actions to move around – {`up`, `down`, `left`, `right`, `stay`}. Each agent is tasked with navigating to a specified goal state in the environment – {'red', 'blue square', 'small green circle', *etc.* } – and the reward for each agent at every timestep is based on team performance *i.e.* $r_t = \frac{\text{\# agents on goal}}{\text{\# agents}}$.

Table 6.2: Success rates on 3 different settings of cooperative navigation in the SHAPES environment.

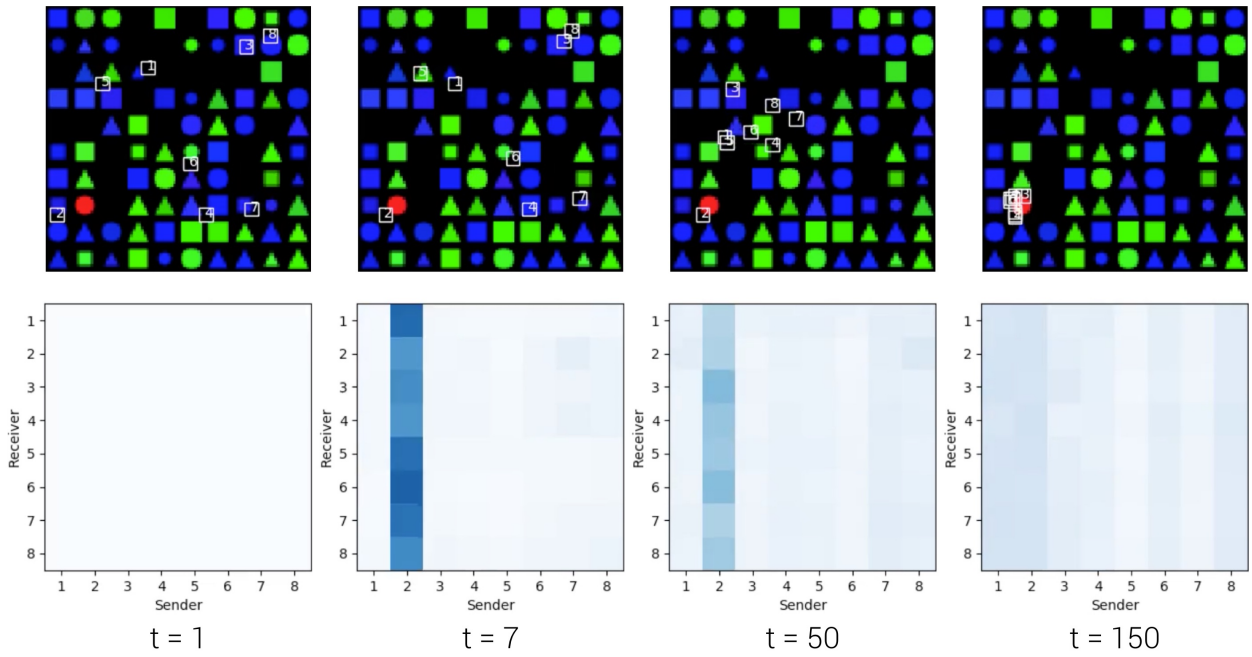
	30×30, 4 agents, find[`red`]	50×50, 4 agents, find[`red`]	50×50, 4 agents, find[`red,red,green,blue`]
No communication	95.3±2.8%	83.6±3.3%	69.1±4.6%
No attention	**99.7**±0.8%	**89.5**±1.4%	82.4±2.1%
TarMAC	**99.8**±0.9%	**89.5**±1.7%	**85.8**±2.5%

Having a symmetric, team-based reward incentivizes agents to cooperate with each other

[1] github.com/jacobandreas/nmn2/tree/shapes



(a) 4 agents have to find [red, red, green, blue] respectively. $t = 1$: inital spawn locations; $t = 2$: 4 was on red at $t = 1$ so 1 and 2 attend to messages from 4 since they have to find red. 3 has found its goal (green) and is self-attending; $t = 6$: 4 attends to messages from 2 as 2 is on 4's target – blue; $t = 8$: 1 finds red, so 1 and 2 shift attention to 1; $t = 21$: all agents are at their respective goal locations and primarily self-attending.



(b) 8 agents have to find red on a large 100×100 environment. $t = 7$: Agent 2 finds red and signals all other agents; $t = 7$ to $t = 150$: All agents make their way to 2's location and eventually converge around red.

Figure 6.2: Visualizations of learned targeted communication in SHAPES. Best viewed in color.

in finding each agent's goal. For example, as shown in Fig. 6.2a, if agent 2's goal is to find red and agent 4's goal is to find blue, it is in agent 4's interest to let agent 2 know if it passes by red ($t = 2$) during its exploration / quest for blue and vice versa ($t = 6$). SHAPES serves as a flexible testbed for carefully controlling and analyzing the effect of changing the size

of the environment, no. of agents, goal configurations, *etc*. Fig. 6.2 visualizes learned protocols from two different configurations, and Table 6.2 reports quantitative evaluation for three different configurations. Benefits of communication and attention increase with task complexity ($30 \times 30 \rightarrow 50 \times 50$ & find[red] $\rightarrow$ find[red,red,green,blue]).

How does targeting work in the communication learnt by TarMAC? Recall that each agent predicts a signature and value vector as part of the message it sends, and a query vector to attend to incoming messages. The communication is targeted because the attention probabilities are a function of both the sender's signature and receiver's query vectors. So it is not just the receiver deciding how much of each message to listen to. The sender also sends out signatures that affects how much of each message is sent to each receiver. The sender's signature could encode parts of its observation most relevant to other agents' goals (for example, it would be futile to convey coordinates in the signature), and the message value could contain the agent's own location. For example, in Fig. 6.2a, at $t = 6$, we see that when agent 2 passes by blue, agent 4 starts attending to agent 2. Here, agent 2's signature encodes the color it observes (which is blue), and agent 4's query encodes its goal (which is also blue) leading to high attention probability. Agent 2's message value encodes coordinates agent 4 has to navigate to, as can be seen at $t = 21$ when agent 4 reaches there.

6.5.2 Traffic Junction

Table 6.3: Success rates on traffic junction. Our targeted 2-stage communication architecture gets a success rate of 97.1% on the 'hard' variant of the task, significantly outperforming [1]. Note that 1- and 2-stage refer to the number of rounds of communication between actions ((6.4)).

	Easy	Hard
No communication	84.9±4.3%	74.1±3.9%
CommNets [1]	99.7±0.1%	78.9±3.4%
TarMAC 1-stage	**99.9**±0.1%	84.6±3.2%
TarMAC 2-stage	**99.9**±0.1%	**97.1**±1.6%

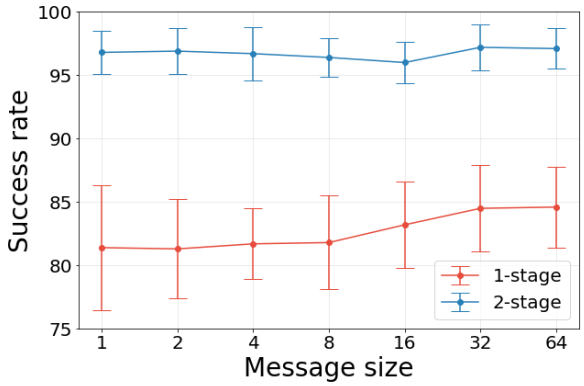
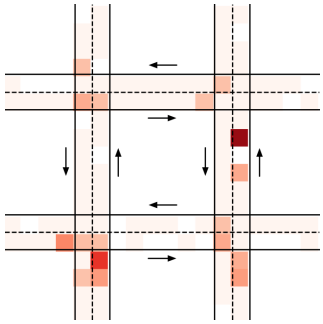
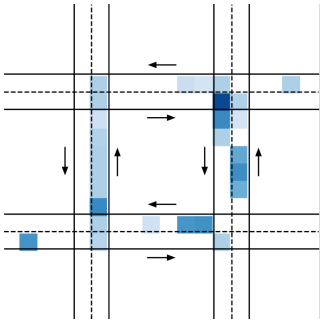


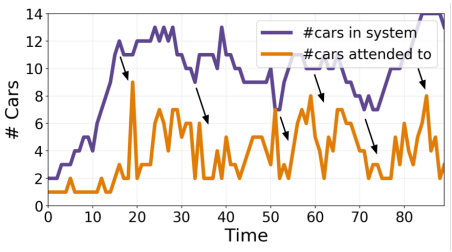
Figure 6.3: Success rates for 1 *vs*. 2-stage *vs*. message size on Hard. Performance does not decrease significantly even when the message vector is a single scalar, and 2 rounds of back-and-forth communication before taking an environment action leads to a significant improvement over 1-stage.



(a) Brake probabilities at different locations on the `hard` traffic junction environment. Cars tend to brake close to or right before entering junctions.



(b) Attention probabilities at different locations. Cars are most attended to in the 'internal grid' – right after the 1st junction and before the 2nd.



(c) No. of cars being attended to. 1) is positively correlated with total cars, indicating that TarMAC is adaptive to dynamic team sizes, and 2) is slightly right-shifted, since it takes few steps of communication to adapt.

Figure 6.4: Results on the traffic junction environment.

Environment and Task. The simulated traffic junction environments from [1] consist of cars moving along pre-assigned, potentially intersecting routes on one or more road junctions. The total number of cars is fixed at N_{max} and at every timestep, new cars get added to the environment with probability p_{arrive}. Once a car completes its route, it becomes available to be sampled and added back to the environment with a different route assignment. Each car has a limited visibility of a 3×3 region around it, but is free to communicate with all other cars. The action space for each car at every timestep is `gas` and `brake`, and the reward consists of a linear time penalty -0.01τ, where τ is the number of timesteps since car has been active, and a collision penalty $r_{\text{collision}} = -10$.

Quantitative Results. We compare our approach with CommNets [1] on the `easy` and `hard` difficulties of the traffic junction environment. The `easy` task has one junction of two one-way roads on a 7×7 grid with $N_{\text{max}} = 5$ and $p_{\text{arrive}} = 0.30$, while the `hard` task has four connected junctions of two-way roads on a 18×18 grid with $N_{\text{max}} = 20$ and $p_{\text{arrive}} = 0.05$. See Fig. 6.4a, 6.4b for an example of the four two-way junctions in the `hard` task. As shown in Table 6.3, a no communication baseline has success rates of 84.9% and 74.1% on `easy` and `hard` respectively. On `easy`, both CommNets and TarMAC get close to 100%. On `hard`, TarMAC with 1-stage communication significantly outperforms CommNets with a success rate of 84.6%, while 2-stage further improves on this at 97.1%, which is an $\sim$18% absolute improvement over CommNets.

Model Interpretation. Interpreting the learned policies, Fig. 6.4a shows braking probabilities at different locations: cars tend to brake close to or right before entering traffic junctions, which is reasonable since junctions have the highest chances for collisions.

Turning our attention to attention probabilities (Fig. 6.4b), we can see that cars are most-attended to when in the 'internal grid' – right after crossing the 1st junction and before hitting the 2nd junction. These attention probabilities are intuitive: cars learn to attentively attend to specific sensitive locations with the most relevant local observations to avoid collisions.

Finally, Fig. 6.4c compares total number of cars in the environment *vs*. number of cars being attended to with probability > 0.1 at any time. Interestingly, these are (loosely) positively correlated, with Spearman's $\sigma = 0.49$, which shows that TarMAC is able to adapt to variable number of agents. Crucially, agents learn this dynamic targeting behavior purely from task rewards with no hand-coding! Note that the right shift between the two curves is expected, as it takes a few timesteps of communication for team size changes to propagate. At a relative time shift of 3, the Spearman's rank correlation between the two curves goes up to 0.53.

Message size *vs*. multi-stage communication. We study performance of TarMAC with varying message value size and number of rounds of communication on the 'hard' variant of the traffic junction task. As can be seen in Fig. 6.3, multiple rounds of communication leads to significantly higher performance than simply increasing message size, demonstrating the advantage of multi-stage communication. In fact, decreasing message size to a single scalar performs almost as well as 64-d, perhaps because even a single real number can be sufficiently partitioned to cover the space of meanings/messages that need to be conveyed for this task.

6.5.3 House3D

Finally, we benchmark TarMAC on a cooperative point-goal navigation task in House3D [107]. House3D provides a rich and diverse set of publicly-available[2] 3D indoor environments, wherein agents do not have access to the top-down map and must navigate purely from first-person vision. Similar to SHAPES, the agents are tasked with finding a specified goal (such as 'fireplace'), spawned at random locations in the environment and allowed to communicate with each other and move around. Each agent gets a shaped reward based on progress towards the specified target. An episode is successful if all agents end within 0.5m of the target object in 50 navigation steps.

Table 6.4 shows success rates on a `find[fireplace]` task in House3D. A no-communication navigation policy trained with the same reward structure gets a success rate of 62.1%. Mean-pooled communication (no attention) performs slightly better with a success rate

[2] github.com/facebookresearch/house3d

of 64.3%, and TarMAC achieves the best success rate at 68.9%. Fig. 6.5 visualizes predicted navigation trajectories of 4 agents. Note that the communication vectors are significantly more compact (32-d) than the high-dimensional observation space, making our approach particularly attractive for scaling to large teams.

Table 6.4: Success rates on a 4-agent cooperative `find[fireplace]` navigation task in House3D.

	Success rate
No communication	62.1$\pm$5.3%
No attention	64.3$\pm$2.3%
TarMAC	**68.9**$\pm$1.1%

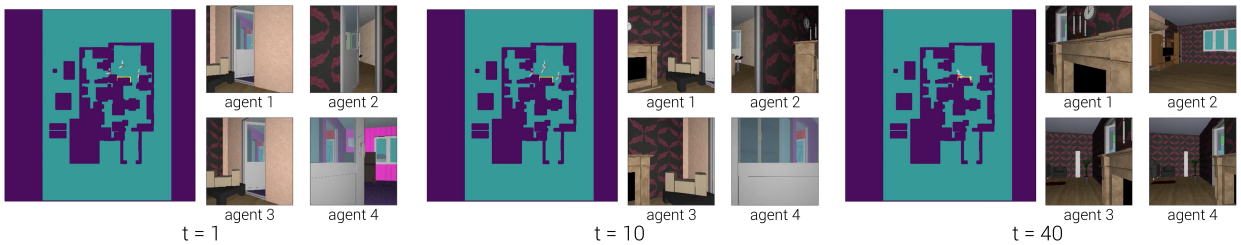


Figure 6.5: Agents navigating to the `fireplace` in House3D (marked in yellow). Note in particular that agent 4 is spawned facing away from it. It communicates with others, turns to face the `fireplace`, and moves towards it.

6.6 Conclusions and Future Work

We introduced TarMAC, an architecture for multi-agent reinforcement learning which allows targeted interactions between agents and multiple stages of collaborative reasoning at every timestep. Evaluation on three environments shows that our model is able to learn intuitive attention behavior and improves performance, with downstream task-specific team reward as sole supervision.

While multi-agent navigation experiments in House3D show promising performance, we aim to exhaustively benchmark TarMAC on more challenging 3D navigation tasks because we believe this is where decentralized targeted communication can have the most impact – as it allows scaling to a large number of agents with large observation spaces. Given that the 3D navigation problem is hard in and of itself, it would be particularly interesting to investigate combinations with recent advances orthogonal to our approach (*e.g.* spatial memory, planning networks) with TarMAC.

Part IV

Language as a Task-Agnostic Probe of World Knowledge

Chapter 7

Probing Emergent Semantics in Predictive Agents via Question Answering

7.1 Introduction

Since the time of Plato, philosophers have considered the apparent distinction between "knowing how" (procedural knowledge or skills) and "knowing what" (propositional knowledge or facts). It is uncontroversial that deep reinforcement learning (RL) agents can effectively acquire procedural knowledge as they learn to play games or solve tasks. Such knowledge might manifest in an ability to find all of the green apples in a room, or to climb all of the ladders while avoiding snakes. However, the capacity of such agents to acquire factual knowledge about their surroundings – of the sort that can be readily hard-coded in symbolic form in classical AI – is far from established. Thus, even if an agent successfully climbs ladders and avoids snakes, we have no certainty that it 'knows' that ladders are brown, that there are five snakes nearby, or that the agent is currently in the middle of a three-level tower with one ladder left to climb.

The acquisition of knowledge about objects, properties, relations and quantities by learning-based agents is desirable for several reasons. First, such knowledge should ultimately complement procedural knowledge when forming plans that enable execution of complex, multi-stage cognitive tasks. Second, there seems (to philosophers at least) to be something fundamentally human about having knowledge of facts or propositions [158]. If one of the goals of AI is to build machines that can engage with, and exhibit convincing intelligence to, human users (*e.g.* justifying their behaviour so humans understand/trust them), then a need for uncovering and measuring such knowledge in learning-based agents will inevitably arise.

Here, we propose the question-conditional probing of agent internal states as a means to study and quantify the knowledge about about objects, properties, relations and quanti-

This chapter is based on work done during an internship at DeepMind.

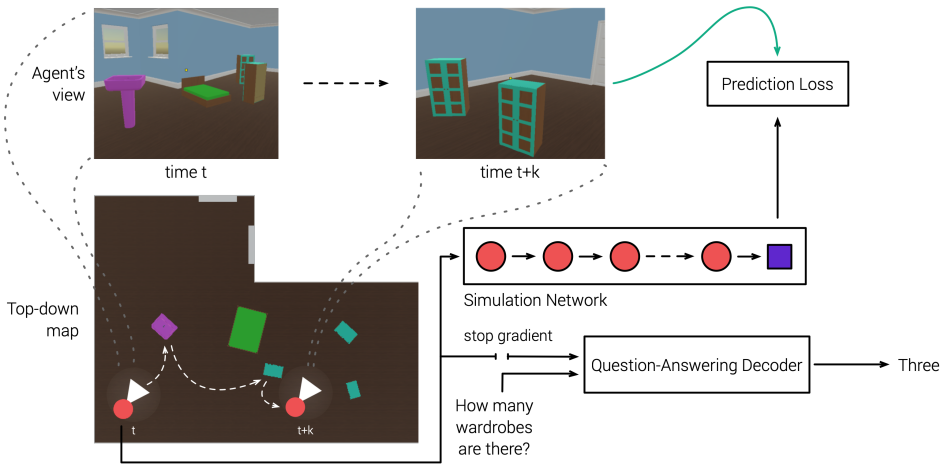


Figure 7.1: We train predictive agents to explore a visually-rich 3D environment with an assortment of objects of different shapes, colors and sizes. As the agent navigates (trajectory shown in white on the top-down map), an auxiliary network learns to simulate representations of future observations (labeled 'Simulation Network') k steps into the future, self-supervised by a loss against the ground-truth egocentric observation at $t + k$. Simultaneously, another decoder network is trained to extract answers to a variety of questions about the environment, conditioned on the agent's internal state but without affecting it (notice 'stop gradient' – gradients from the QA decoder are not backpropagated into the agent). We use this question-answering paradigm to decode and understand the internal representations that such agents develop. Note that the top-down map is only shown for illustration and not available to the agent.

ties encoded in the internal representations of neural-network-based agents. Couching an analysis of such knowledge in terms of question-answering has several pragmatic advantages. First, question-answering provides a general purpose method for agent-analysis and an intuitive investigative tool for humans – one can simply *ask* an agent what it knows about its environment and get an answer back, without having to inspect internal activations. Second, the space of questions is essentially open-ended – we can pose arbitrarily complex questions to an agent, enabling a comprehensive analysis of the current state of its propositional knowledge. Question-answering has previously been studied in textual [73, 159], visual [21, 22, 89] and embodied [115, 134] settings. Crucially, however, these systems are trained end-to-end for the goal of answering questions. Here, we utilize question-answering simply to probe an agent's internal representation, without backpropagating gradients from the question-answering decoder into the agent. That is, we view question-answering as a general purpose (conditional) decoder of environmental information designed to assist the development of agents by revealing the extent (and limits) of their knowledge.

Many techniques have been proposed for endowing agents with general (*i.e.* task-agnostic) knowledge, based on both hard-coding and learning. Here, we specifically focus on the

effect of self-supervised predictive modeling – a learning-based approach – on the acquisition of propositional knowledge. Inspired by learning in humans [160–163], predictive modeling, *i.e.* predicting future sensory observations, has emerged as a powerful method to learn general-purpose neural network representations [3, 4, 164–171]. These representations can be learned while exploring in and interacting with an environment in a task-agnostic manner, and later exploited for goal-directed behavior.

We evaluate predictive *vs.* non-predictive agents (both trained for exploration) on our question-answering testbed to investigate how much knowldge of object shapes, quantities, and spatial relations they acquire *solely by egocentric prediction*. The set of questions is intended to be holistic, *i.e.* answering tends to require knowledge of relevant aspects of the whole environment, rather than that which can be gleaned from a single observation or a few consecutive observations.

Concretely, we make the following contributions

- In a visually-rich 3D room environment developed in the Unity engine, we develop a set of questions designed to probe a diverse body of factive knowledge about the environment – from identifying shapes and colors ('What shape is the red object?') to counting ('How many blue objects are there?') to spatial relations ('What is the color of the chair near the table?'), exhaustive search ('Is there a cushion?'), and comparisons ('Are there the same number of tables as chairs?').
- We train RL agents augmented with predictive loss functions – 1) action-conditional CPC [3] and 2) SimCore [4] – for an exploration task and analyze the internal representations they develop by decoding answers to our suite of questions. Crucially, the QA decoder is trained independent of the predictive agent and we find that QA performance is indicative of the agent's ability to capture global environment structure and semantics *solely through egocentric prediction*. We compare these predictive agents to strong non-predictive LSTM baselines as well as to an agent that is explicitly optimized for the question-answering task.
- We establish generality of the encoded knowledge by testing zero-shot generalization of a trained QA decoder to compositionally novel questions (unseen combinations of seen attributes), suggesting a degree of compositionality in the internal representations captured by predictive agents.

Table 7.1: QA task templates. In every episode, objects and their configurations are randomly generated, and these templates get translated to QA pairs for all unambiguous <shape, color> combinations. There are 50 shapes and 10 colors in total. See B.4 for details.

Question type	Template	Level codename	# QA pairs
Attribute	What is the color of the <shape>?	color	500
	What shape is the <color> object?	shape	500
Count	How many <shape> are there?	count_shape	200
	How many <color> objects are there?	count_color	40
Exist	Is there a <shape>?	existence_shape	100
Compare + Count	Are there the same number of <color1> objects as <color2> objects?	compare_n_color	180
	Are there the same number of <shape1> as <shape2>?	compare_n_shape	4900
Relation + Attribute	What is the color of the <shape1> near the <shape2>?	near_color	24500
	What is the <color> object near the <shape>?	near_shape	25000

7.2 Background and related work

Our work builds on prior work on predictive modeling and auxiliary loss functions in reinforcement learning as well as grounded language learning and embodied question answering.

Propositional knowledge is knowledge that a statement, expressed in natural or formal language, is true [172]. Since at least Plato, epistemologist philosophers have contrasted propositional knowledge with *procedural knowledge* (knowledge of how to do something), and some (but not all) distinguish this from *perceptual knowledge* (knowledge obtained by the senses that cannot be translated into a proposition) [173]. An ability to exhibit this sort of knowledge in a convincing way is likely to be crucial for the long-term goal of having agents achieve satisfying interactions with humans, since an agent that cannot express its knowledge and beliefs in human-interpretable form may struggle to earn the trust of users.

Predictive modeling and auxiliary loss functions in RL. The power of predictive modeling for representation learning has been known since at least the seminal work of [160] on emergent language structures. More recent examples include Word2Vec [174], Skip-Thought vectors [175], and BERT [176] in language, while in vision similar principles have been applied to context prediction [177, 178], unsupervised tracking [179], inpainting [180] and colorization [181]. More related to us is the use of such techniques in designing auxiliary loss functions for training model-free RL agents, such as successor representations [50, 182], value and reward prediction [170, 183], contrastive predictive coding (CPC) [3, 184], and SimCore [4].

Grounded language learning. Inspired by the work of [56] on SHRDLU, several recent works have explored linguistic representation learning by grounding language into ac-

tions and pixels in physical environments – in 2D gridworlds [58, 59, 113], 3D [48–52, 54, 115, 130, 131, 134–136, 185] and textual [132, 133] environments. Closest to our work is the task of Embodied Question Answering [115, 134, 186–188] – where an embodied agent in an environment (*e.g.* a house) is asked to answer a question (*e.g.* "What color is the piano?"). Typical approaches to EmbodiedQA involve training agents to move for the goal of answering questions. In contrast, our focus is on learning a predictive model in a *goal-agnostic* exploration phase and using question-answering as a post-hoc testbed for evaluating the semantic knowledge that emerges in the agent's representations from predicting the future.

Neural population decoding. Probing an agent with a QA decoder can be viewed as a variant of neural population decoding, used as an analysis tool in neuroscience [189–191] and more recently in deep learning [3, 4, 192–195]. The idea is to test whether specific information is encoded in a learned representation, by feeding the representation as input to a probe network, generally a classifier trained to extract the desired information. In RL, this is done by training a probe to predict parts of the ground-truth state of the environment, such as an agent's position or orientation, without backpropagating through the agent's internal state.

Prior work has required a separate network to be trained for each probe, even for closely related properties such as position vs. orientation [3] or grammatical features of different words in the same sentence [194]. Moreover, each probe is designed with property-specific inductive biases, such as convnets for top-down views vs. MLPs for position [4]. In contrast, we train a single, general-purpose probe network that covers a variety of question types, with an inductive bias for language processing. This generality is possible because of the external conditioning, in the form of the question, supplied to the probe. External conditioning moreover enables agent analysis using novel perturbations of the probe's training questions.

Neuroscience. Predictive modeling is thought to be a fundamental component of human cognition [160, 163, 196]. In particular, it has been proposed that perception, learning and decision-making rely on the minimization of prediction error [161, 162]. A well-established strand of work has focused on decoding predictive representations in brain states [197, 198]. The question of how prediction of sensory experience relates to higher-order conceptual knowledge is complex and subject to debate [199, 200], though some have proposed that conceptual knowledge, planning, reasoning, and other higher-order functions emerge in deeper layers of a predictive network. We focus on the emergence of propositional knowledge in a predictive agent's internal representations.

7.3 Environment & Tasks

Environment. We use a Unity-based visually-rich 3D environment (see Figure 7.1). It is a single L-shaped room that can be programmatically populated with an assortment of objects of different colors at different spatial locations and orientations. In total, we use a library of 50 different objects, referred to as 'shapes' henceforth (*e.g.* chair, teddy, glass, *etc.*), in 10 different colors (*e.g.* red, blue, green, *etc.*). For a complete list of environment details, see Sec. B.4.

At every step, the agent gets a 96×72 first-person RGB image as its observation, and the action space consists of movements (`move-{forward,back,left,right}`), turns (`turn-{up,down,left,right}`), and object pick-up and manipulation (4 DoF: yaw, pitch, roll, and movement along the axis between the agent and object). See Table B.3 in the Appendix for the full set of actions.

Question-Answering Tasks. We develop a range of question-answering tasks of varying complexity that test the agent's local and global scene understanding, visual reasoning, and memory skills. Inspired by [111, 115, 134], we programmatically generate a dataset of questions (see Table 7.1). These questions ask about the presence or absence of objects (`existence_shape`), their attributes (`color, shape`), counts (`count_color, count_shape`), quantitative comparisons (`compare_count_color, compare_count_shape`), and elementary spatial relations (`near_color, near_shape`). Unlike the fully-observable setting in CLEVR [111], the agent does not get a global view of the environment, and must answer these questions from a sequence of partial egocentric observations. Moreover, unlike prior work on EmbodiedQA [115, 134], the agent is *not* being trained end-to-end to move to answer questions. It is being trained to explore, and answers are being decoded (without backpropagating gradients) from its internal representation. Thus, in order to answer these questions, the agent *must* learn to encode relevant aspects of the environment in a representation amenable to easy decoding into symbols (*e.g.* what does the word "chair" mean? or what representations does computing "how many" require?).

7.4 Approach

Learning an exploration policy. Predictive modeling has proven to be effective for an agent to develop general knowledge of its environment as it explores and behaves towards its goal, typically maximising environment returns [3, 4]. Since we wish to evaluate the effectiveness of predictive modeling independent of the agent's specific goal, we define a simple task that stimulates the agent to visit all of the 'important' places in the

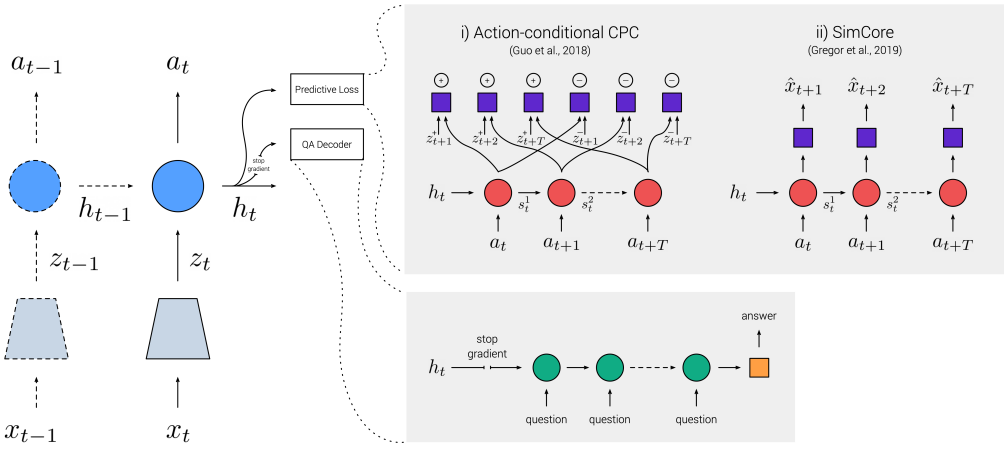


Figure 7.2: Approach: at every timestep t, the agent receives an RGB observation x_t as input, processes it using a convolutional neural network to produce z_t, which is then processed by an LSTM to select action a_t. The agent learns to explore – it receives a reward of 1.0 for navigating to each new object. As it explores the environment, it builds up an internal representation h_t, which receives pressure from an auxiliary predictive module to capture environment semantics so as to accurately predict consequences of its actions multiple steps into the future. We experiment with a vanilla LSTM agent and two recent predictive approaches – CPC|A [3] and SimCore [4]. The internal representations are then probed via a question-answering decoder whose gradients are not backpropagated into the agent. The QA decoder is an LSTM initialized with h_t and receiving the question at every timestep.

environment (*i.e.* to acquire an exploratory but otherwise task-neutral policy). This is achieved by giving the agent a reward of $+1.0$ every time it visits an object in the room for the first time. After visiting all objects, rewards are refreshed and available to be consumed by the agent again (*i.e.* re-visiting an object the agent has already been to will now again lead to a $+1.0$ reward), and this process continues for the duration of each episode (30 seconds or 900 steps).

During training on this exploration task, the agent receives a first-person RGB observation x_t at every timestep t, and processes it using a convolutional neural network to produce z_t. This is input to an LSTM policy whose hidden state is h_t and output a discrete action a_t. The agent optimizes the discounted sum of future rewards using an importance-weighted actor-critic algorithm [201].

Training the QA-decoder. The question-answering decoder is operationalized as an LSTM that is initialized with the agent's internal representation h_t and receives the question as input at every timestep (see Fig. 4.5). The question is a string that we tokenise into words and then map to learned embeddings. The question decoder LSTM is then unrolled for a fixed number of computation steps after which it predicts a softmax distribution over the vocabulary of one-word answers to questions in Table 7.1, and is trained via a cross-entropy loss. Crucially, this QA decoder is trained independent of the agent

policy; *i.e.* gradients from this decoder are not allowed to flow back into the agent. We evaluate question-answering performance by measuring top-1 accuracy at the end of the episode – we consider the agent's top predicted answer at the last time step of the episode and compare that with the ground-truth answer.

The QA decoder can be seen as a general purpose decoder trained to extract object-specific knowledge from the agent's internal state without affecting the agent itself. If this knowledge is not retained in the agent's internal state, then this decoder will not be able to extract it. This is an important difference with respect to prior work [115, 134] – wherein agents were trained to move to answer questions, *i.e.* all parameters had access to linguistic information. Recall that the agent's navigation policy has been trained for exploration, and so the visual information required to answer a question need not be present in the observation at the end of the episode. Thus, through question-answering, we are evaluating the degree to which agents encode relevant aspects of the environment (object colors, shapes, counts, spatial relations) in their internal representations *and* maintain this information in memory beyond the point at which it was initially received. See B.1.3 for more details about the QA decoder.

7.4.1 Auxiliary Predictive Losses

We augment the baseline architecture described above with an auxiliary predictive head consisting of a simulation network (operationalized as an LSTM) that is initialized with the agent's internal state h_t and deterministically simulates future latent states $s_t^1, \ldots, s_t^k, \ldots$ in an open-loop manner, receiving the agent's action sequence as input. We evaluate two predictive losses – action-conditional CPC [3] and SimCore [4]. See Fig. 4.5 for overview, B.1.2 for details.

Action-conditional CPC (CPC|A, [3]) makes use of a noise contrastive estimation model to discriminate between true observations processed by the convolutional neural network z_{t+k}^+ (k steps into the future) and negatives randomly sampled from the dataset z_{t+k}^-, in our case from other episodes in the minibatch. Specifically, at each timestep $t+k$ (up to a maximum), the output of the simulation core s_t^k and z_{t+k}^+ are fed to an MLP to predict 1, and s_t^k and z_{t+k}^- are used to predict 0.

SimCore [4] uses the simulated state s_t^k to condition a generative model based on ConvDRAW [202] and GECO [203] that predicts the distribution of true observations $p(x_{t+k}|h_t, a_{t,\ldots,(t+k)})$ in pixel space.

Baselines. We evaluate and compare the above approaches with 1) a vanilla RL agent without any auxiliary predictive losses (referred to as 'LSTM'), and 2) a question-only

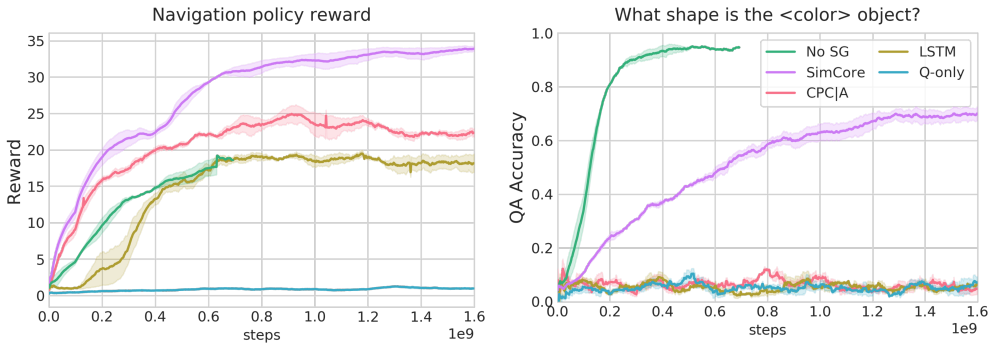


Figure 7.3: L – Reward in an episode. R – Top-1 QA accuracy. Averaged over 3 seeds. Shaded region is 1 SD.

Table 7.2: Top-1 accuracy on question-answering tasks.

	Overall	shape	color	exist	count_shape	count_color	compare_n_color	compare_n_shape	near_shape	near_color
Baseline: Question-only	29 ±3	04 ±2	10 ±2	63 ±4	24 ±3	24 ±3	49 ±3	70 ±3	04 ±2	09 ±3
LSTM	31 ±4	04 ±1	10 ±2	54 ±6	34 ±3	38 ±3	53 ±3	70 ±3	04 ±2	09 ±3
CPC|A	32 ±3	06 ±2	08 ±2	64 ±3	**39** ±3	39 ±3	50 ±4	70 ±3	06 ±2	10 ±3
SimCore	**60** ±3	**72** ±3	**81** ±3	**72** ±3	**39** ±3	**57** ±3	**56** ±3	**73** ±3	**30** ±3	**59** ±3
Oracle: No SG	63 ±3	96 ±2	81 ±2	60 ±3	45 ±3	57 ±3	51 ±3	76 ±3	41 ±3	72 ±3

agent that receives zero-masked observations as input and is useful to measure biases in our question-answering testbed. Such a baseline is critical, particularly when working with simulated environments, as it can uncover biases in the environment's generation of tasks that can result in strong but uninteresting performance from agents capable of powerful function approximation [204].

No stop gradient. We also compare against an agent without blocking the QA decoder gradients (labeled 'No SG'). This model differs from the above in that it is trained end-to-end – with supervision – to answer the set of questions in addition to the exploration task. Hence, it represents an agent receiving privileged information about how to answer and its performance provides an upper bound for how challenging these question-answering tasks are in this context.

7.5 Experiments & Results

7.5.1 Question-Answering Performance

We begin by analyzing performance on a single question – shape – which are of the form "what shape is the <color> object?". Figure 7.3 shows the average reward accumulated by the agent in one episode (left) and the QA accuracy at the last timestep of the episode (right) for all approaches over the course of training. We make the following observations:

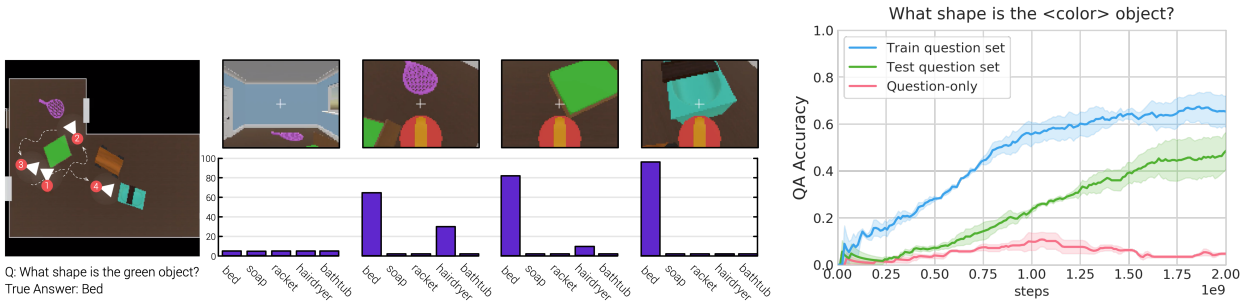


Figure 7.4: (Left): Sample trajectory (1 → 4) and QA decoding predictions (for top 5 most probable answers) for the 'What shape is the green object?' from SimCore. Note that top-down map is not available to the agent. (Right): QA accuracy on disjoint train and test splits.

- **All agents learn to explore**. With the exception 'question-only', all agents achieve high reward on the exploration task. This means that they visited all objects in the room more than once each and therefore, in principle, have been exposed to sufficient information to answer all questions.
- **Predictive models aid navigation**. Agents equipped with auxiliary predictive losses – CPC|A and SimCore – collect the most rewards, suggesting that predictive modeling helps navigate the environment efficiently. This is consistent with findings in [4].
- **QA decoding from LSTM and CPC|A representations is no better than chance**.
- **SimCore's representations lead to best QA accuracy**. SimCore gets to a QA accuracy of ∼72% indicating that its representations best capture propositional knowledge and are best suited for decoding answers to questions. Figure 7.4 (Left) shows example predictions.
- **Wide gap between SimCore and No SG**. There is a ∼24% gap between SimCore and the No SG oracle, suggesting scope for better auxiliary predictive losses.

It is worth emphasizing that answering this shape question from observations is not a challenging task in and of itself. The No SG agent, which is trained end-to-end to optimize

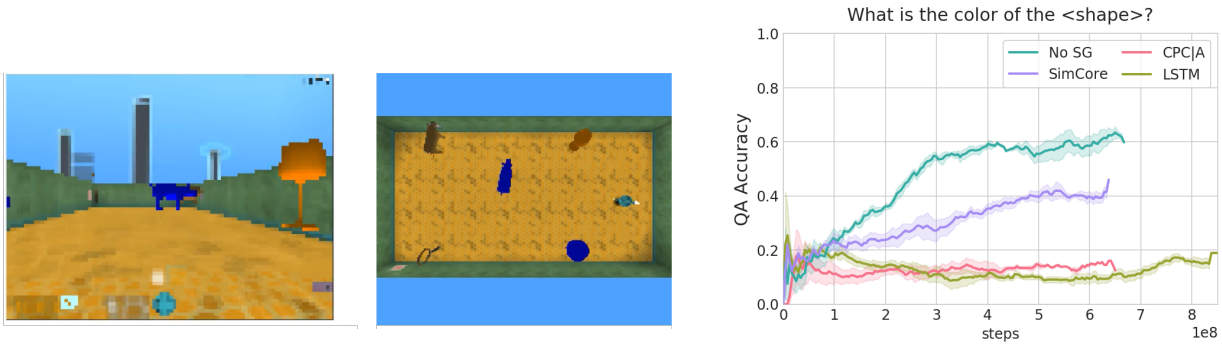


Figure 7.5: (Left) DeepMind Lab environment [5]: Rectangular-shaped room with 6 randomly selected objects out of a pool of 20 different objects of different colors. (Right) QA accuracy for `color` questions (What is the color of the <`shape`>?) in DeepMind Lab. Consistent with results in the main paper, internal representations of the SimCore agent lead to the highest accuracy while CPC|A and LSTM perform worse and similar to each other.

both for exploration and QA, achieves almost-perfect accuracy (~96%). The challenge arises from the fact that we are not training the agent end-to-end – from pixels to navigation to QA – but decoding the answer from the agent's internal state, which is learned agnostic to the question. The answer can only be decoded if the agent's internal state contains relevant information represented in an easily-decodable way.

Decoder complexity. To explore the possibility that answer-relevant information is present in the agent's internal state but requires a more powerful decoder, we experiment with QA decoders of a range of depths. As detailed in Figure B.2 in the appendix, we find that using a deeper QA decoder with SimCore does lead to higher QA accuracy (from $1 \rightarrow 12$ layers), although greater decoder depths become detrimental after 12 layers. Crucially, however, in the non-predictive LSTM agent, the correct answer cannot be decoded irrespective of QA decoder capacity. This highlights an important aspect of our question-answering evaluation paradigm – that while the absolute accuracy at answering questions may also depend on decoder capacity, relative differences provide an informative comparison between internal representations developed by different agents.

Table 7.2 shows QA accuracy for all QA tasks (see Figure B.3 in appendix for training curves). The results reveal large variability in difficulty across question types. Questions about attributes (`color` and `shape`), which can be answered from a single well-chosen frame of visual experience, are the easiest, followed by spatial relationship questions (`near_color` and `near_shape`), and the hardest are counting questions (`count_color` and `count_shape`). We further note that:

- **All agents perform better than the question-only baseline**, which captures any bi-

ases in the environment or question distributions (enabling strategies such as constant prediction of the most-common answer).

- **CPC|A representations are not better than LSTM on most question types**.
- **SimCore representations achieve higher QA accuracy than other approaches**, substantially above the question-only baseline on `count_color` (57% *vs*. 24%), `near_shape` (30% *vs*. 4%) and `near_color` (59% *vs*. 9%), demonstrating a strong tendency for encoding and retaining information about object identities, properties, and both spatial and temporal relations.

Finally, as before, the No SG agent trained to answer questions without stopped gradients achieves highest accuracy for most questions, although not all – perhaps due to trade-offs between simultaneously optimizing performance for different QA losses and the exploration task.

7.5.2 Compositional Generalization

While there is a high degree of procedural randomization in our environment and QA tasks, overparameterized neural-network-based models in limited environments are always prone to overfitting or rote memorization. We therefore constructed a test of the generality of the information encoded in the internal state of an agent. The test involves a variant of the `shape` question type (*i.e*. questions like "what shape is the <`color`> object?"), but in which the possible question-answer pairs are partitioned into mutually exclusive training and test splits. Specifically, the test questions are constrained such that they are compositionally novel – the <`color`, `shape`> combination involved in the question-answer pair is never observed during training, but both attributes are observed in other contexts. For instance, a test question-answer pair "Q: what shape is the **blue** object?, A: **table**" is excluded from the training set of the QA decoder, but "Q: what shape is the **blue** object?, A: **car**" and "Q: What shape is the **green** object?, A: **table**" are part of the training set (but not the test set).

We evaluate the SimCore agent on this test of generalization (since other agents perform poorly on the original task). Figure 7.4 (right) shows that the QA decoder applied to SimCore's internal states performs at substantially above-chance (and all baselines) on the held-out test questions (although somewhat lower than training performance). This indicates that the QA decoder extracts and applies information in a comparatively factorized (or compositional) manner, and suggests (circumstantially) that the knowledge acquired by the SimCore agent may also be represented in this way.

7.5.3 Robustness of the results

To check if our results are robust to the choice of environment, we developed a similar setup using the DeepMind Lab environment [5] and run the same experiments *without* any change in hyperparameters.

The environment consists of a rectangular room that is populated with a random selection of objects of different shapes and colors in each episode. There are 6 distinct objects in each room, selected from a pool of 20 objects and 9 different colors. We use a similar exploration reward structure as in our earlier environment to train the agents to navigate and observe all objects. Finally, in each episode, we introduce a question of the form 'What is the color of the <shape>?' where <shape> is replaced by the name of an object present in the room.

Figure 7.5 shows question-answering accuracies in the DeepMind Lab environment. Consistent with the results presented above, internal representations of the SimCore agent lead to the highest answering accuracy while CPC|A and the vanilla LSTM agent perform worse and similar to each other. Crucially, for running experiments in DeepMind Lab, we *did not* change any hyperparameters from the experimental setup described before. This demonstrates that our approach is not specific to a single environment and that it can be readily applied in a variety of settings.

7.6 Discussion

Developing agents with world models of their environments is an important problem in AI. To do so, we need tools to evaluate and diagnose the internal representations forming these world models in addition to studying task performance. Here, we marry together population or glass-box decoding techniques with a question-answering paradigm to discover how much propositional (or declarative) knowledge agents acquire as they explore their environment.

We started by developing a range of question-answering tasks in a visually-rich 3D environment, serving as a diagnostic test of an agent's scene understanding, visual reasoning, and memory skills. Next, we trained agents to optimize an exploration objective with and without auxiliary self-supervised predictive losses, and evaluated the representations they form as they explore an environment, via this question-answering testbed.

We compared model-free RL agents alongside agents that make egocentric visual predictions and found that the latter (in particular SimCore [4]) are able to reliably capture de-

tailed propositional knowledge in their internal states, which can be decoded as answers to questions, while non-predictive agents do not, even if they optimize the exploration objective well.

Interestingly, not all predictive agents are equally good at acquiring knowledge of objects, relations and quantities. We compared a model learning the probability distribution of future frames in pixel space via a generative model (SimCore [4]) with a model based on discriminating frames through contrastive estimation (CPC|A [3]). We found that while both learned to navigate well, only the former developed representations that could be used for answering questions about the environment. [4] previously showed that the choice of predictive model has a significant impact on the ability to decode an agent's position and top-down map reconstructions of the environment from its internal representations. Our experiments extend this result to decoding factual knowledge, and demonstrate that the question-answering approach has utility for comparing agents.

Finally, the fact that we can even decode answers to questions from an agent's internal representations learned solely from egocentric future predictions, without exposing the agent itself directly to knowledge in propositional form, is encouraging. It indicates that the agent is learning to form and maintain invariant object identities and properties (modulo limitations in decoder capacity) in its internal state *without explicit supervision*. It is $\sim$30 years since [160] showed how syntactic structures and semantic organization can emerge in the units of a neural network as a consequence of the simple objective of predicting the next word in a sequence. This work corroborates Elman's belief in the power of prediction by demonstrating the diversity of knowledge that can emerge when a situated neural-network agent is endowed with powerful predictive objectives applied to raw pixel observations. We think we have just scratched the surface of what might be discovered using such techniques, and hope our work inspires future research in evaluating predictive agents using natural linguistic interactions.

Chapter 8

Conclusion

In this dissertation, we have studied several tasks and techniques that situate agents in multimodal environments, enabling them to actively interact with other agents via language and the environment via actions. In Visual Dialog (Part I), we had an agent interacting in natural language with a human/machine partner about images. In Embodied Question Answering (Part II), we augmented this setup so agents could take physical actions in an embodied environment, in addition to answering questions asked by a human partner. In TarMAC (Part III), we further augmented this setup so agents interact and coordinate with not just one but multiple partners, while still taking actions in an embodied environment. And finally, in Part IV, we explored a complementary question – how do we train these embodied agents for *any* task, so that the knowledge they develop supports easy decoding to language – minimizing the requirement for preemptively-collected language annotations and demonstrating the efficacy of language as a tool for agent design.

Not only is embodiment important for agents to develop a richer understanding of language – grounded in physical concepts and actions (instead of text corpuses) – but it is also necessary so that agents have access to all the information humans have access to and/or have aggregated through years and generations of embodied interactions with the world – an intuitive notion of physics (*e.g.* how spring-body systems work), how people think and interact with each other (psychology and theory of mind), common sense knowledge (*e.g.* how switches work or how mirrors work), *etc*. It is practically infeasible to compress all of that causal knowledge and common sense into a static corpus. I posit that such knowledge learned from embodiment in realistic environments will not only be useful for embodied tasks, but also for tasks on static corpuses, where current state-of-the-art models have high accuracy as per established metrics but trivial for humans to break and expose inconsistencies in *e.g.* visual dialog, visual question answering, *etc*.

Now the real world is perhaps the best environment for robots to be embodied in and learn via interaction from, but training an embodied robot from scratch in the real-world is too dangerous and sample-inefficient to be viable. So the typical pipeline involves

pretraining in simulation, and then transfering to and finetuning on a real robot. This pipeline necessitates realistic and fast simulators (*e.g*. House3D [107], THOR [138], Habitat [205], Gibson [206]). But beyond their current capabilities, which primarily focus on visual navigation in indoor environments (+ object interaction in THOR), we need bigger environments (at the scale of complete cities), humans in the loop (via say web-based APIs), more modalities – smell, sound, fine-grained touch – and realistic sensing, actuation and physics. The move from focused tasks on static datasets (*e.g*. image classification [207], Visual Dialog [89]) to focused tasks in environments (*e.g*. navigation [48–54,136], EmbodiedQA [115,134,186–188]) has enabled recent advances in embodied AI, but the next stream of breakthroughs will be unlocked by the move to open-ended, realistic, multi-task environments (*e.g*. an agent that can get groceries from the market).

List of Publications

Contributions included in this thesis:

1. **Visual Dialog**
   Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M. F. Moura, Devi Parikh, Dhruv Batra
   CVPR, 2017, ***Spotlight***
   PAMI, 2018

2. **Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning**
   Abhishek Das*, Satwik Kottur*, José M. F. Moura, Stefan Lee, Dhruv Batra
   ICCV, 2017, ***Oral***
   (* denotes equal contribution)

3. **Embodied Question Answering**
   Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, Dhruv Batra
   CVPR, 2018, ***Oral***

4. **Neural Modular Control for Embodied Question Answering**
   Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, Dhruv Batra
   CoRL, 2018, ***Spotlight***

5. **TarMAC: Targeted Multi-Agent Communication**
   Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, Joelle Pineau
   ICML, 2019, ***Oral***

6. **Probing Emergent Semantics in Predictive Agents via Question Answering**
   Abhishek Das*, Federico Carnevale*, Hamza Merzic, Laura Rimell, Rosalia Schneider, Alden Hung, Josh Abramson, Arun Ahuja, Stephen Clark, Greg Wayne, Felix Hill
   ICML, 2020, ***Oral***
   (* denotes equal contribution)

Contributions not included in this thesis:

7. **Human Attention in Visual Question Answering:**
   Do Humans and Deep Networks Look at the Same Regions?
   Abhishek Das*, Harsh Agrawal*, C. Lawrence Zitnick, Devi Parikh, Dhruv Batra
   EMNLP, 2016
   CVIU, 2017

8. **Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization**
   Ramprasaath Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra
   ICCV, 2017
   IJCV, 2019

9. **Evaluating Visual Conversational Agents via Cooperative Human-AI Games**
   Prithvijit Chattopadhyay*, Deshraj Yadav*, Viraj Prabhu, Arjun Chandrasekaran, Abhishek Das, Stefan Lee, Dhruv Batra, Devi Parikh
   HCOMP, 2017

10. **IR-VIC: Unsupervised Discovery of Sub-goals for Transfer in Reinforcement Learning**
    Nirbhay Modhe, Prithvijit Chattopadhyay, Mohit Sharma, Abhishek Das, Devi Parikh, Dhruv Batra, Ramakrishna Vedantam
    IJCAI-PRICAI 2020

11. **Embodied Question Answering in Photorealistic Environments**
    with Point Cloud Perception
    Erik Wijmans*, Samyak Datta*, Oleksandr Maksymets, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, Dhruv Batra
    CVPR, 2019, ***Oral***

12. **Audio-Visual Scene-Aware Dialog**
    Huda Alamri, Vincent Cartillier, Abhishek Das, Jue Wang, Stefan Lee, Peter Anderson, Irfan Essa, Devi Parikh, Dhruv Batra, Anoop Cherian, Tim K. Marks, Chiori Hori
    CVPR, 2019

13. **End-to-end Audio Visual Scene-Aware Dialog Using Multimodal Attention-based Video Features**
    Chiori Hori, Huda Alamri, Jue Wang, Gordon Wichern, Takaaki Hori, Anoop Cherian, Tim K. Marks, Vincent Cartillier, Raphael Lopes, Abhishek Das, Irfan Essa, Dhruv Batra, Devi Parikh
    ICASSP, 2019

14. **Improving Generative Visual Dialog by Answering Diverse Questions**
    Vishvak Murahari, Prithvijit Chattopadhyay, Dhruv Batra, Devi Parikh, Abhishek Das
    EMNLP, 2019

15. **Large-scale Pretraining for Visual Dialog: A Simple State-of-the-Art Baseline**
    Vishvak Murahari, Dhruv Batra, Devi Parikh, Abhishek Das
    ECCV, 2020

16. **Auxiliary Tasks Speed Up Learning PointGoal Navigation**
    Joel Ye, Dhruv Batra, Erik Wijmans*, Abhishek Das*
    CoRL, 2020

Appendices

Appendix A

Embodied Question Answering

This supplementary document is organized as follows:

- Sec. A.1 presents the question-answer generation engine in detail, including functional programs associated with questions, and checks and balances in place to avoid ambiguities, biases, and redundancies.
- Sec. A.2 describes the CNN models that serve as the vision module for our EmbodiedQA model. We describe the model architecture, along with the training details, quantitative as well as qualitative results.
- Sec. A.3 describes the answering module in our agent.
- Sec. A.4 reports machine question answering performance conditioned on human navigation paths (collected via human studies on Amazon Mechanical Turk).
- Finally, youtube.com/watch?v=gVj-TeIJfrk shows example navigation and answer predictions by our agent.

Table A.1: Functional forms of all question types in the EQA dataset

Template	Functional Form
location	$select(objects) \rightarrow unique(objects) \rightarrow blacklist(location) \rightarrow query(location)$
color	$select(objects) \rightarrow unique(objects) \rightarrow blacklist(color) \rightarrow query(color)$
color_room	$select(room) \rightarrow unique(room) \rightarrow select(objects) \rightarrow unique(objects) \rightarrow blacklist(color) \rightarrow query(color\_room)$
preposition	$select(room) \rightarrow unique(room) \rightarrow select(objects) \rightarrow unique(objects) \rightarrow blacklist(preposition) \rightarrow relate() \rightarrow query(preposition)$
existence	$select(room) \rightarrow unique(room) \rightarrow select(objects) \rightarrow blacklist(existence) \rightarrow query(exist)$
logical	$select(room) \rightarrow unique(room) \rightarrow select(objects) \rightarrow blacklist(existence) \rightarrow query(logical)$
count	$select(room) \rightarrow unique(room) \rightarrow select(objects) \rightarrow blacklist(count) \rightarrow query(count)$
room_count	$select(room) \rightarrow query(room\_count)$
distance	$select(room) \rightarrow unique(room) \rightarrow select(objects) \rightarrow unique(objects) \rightarrow blacklist(distance) \rightarrow distance() \rightarrow query(distance)$

A.1 Question-Answer Generation Engine

Recall that each question in EQA is represented as a functional program that can be executed on the environment yielding an answer[1]. In this section, we describe this process in detail. In the descriptions that follow, an 'entity' can refer to either a queryable room or a queryable object from the House3D [107] environment.

Functional Forms of Questions. The functional programs are composed of elementary operations described below:

1. *select*(*entity*): Fetches a list of entities from the environment. This operation is similar to the 'select' query in relational databases.

2. *unique*(*entity*): Performs filtering to retain entities that are unique (*i.e.* occur exactly once). For example, calling *unique*(*rooms*) on the set of rooms [*'living_room'*, *'bedroom'*, *'bedroom'*] for a given house will return [*'living_room'*].

3. *blacklist*(*template*): This function operates on a list of object entities and filters out a pre-defined list of objects that are blacklisted for the given template. We do not ask questions of a given template type corresponding to any of the blacklisted objects. For example, if the blacklist contains the objects {*'column'*, *'range_hood'*, *'toy'*} and the objects list that the function receives is {*'piano'*, *'bed'*, *'column'*}, then the output of the *blacklist*() function is: {*'piano'*, *'bed'*}

4. *query*(*template*): This is used to generate the questions strings for the given template on the basis of the entities that it receives. For example, if *query*(*location*) receives the following set of object entities as input: [*'piano'*, *'bed'*, *'television'*], it produces 3 question strings of the form: *what room is the* <*OBJ*> *located in?* where <*OBJ*> = {*'piano'*, *'bed'*, *'television'*}.

5. *relate*(): This elementary operation is used in the functional form for preposition questions. Given a list of object entities, it returns a subset of the pairs of objects that have a {*'on'*, *'above'*, *'under'*, *'below'*, *'next to'*} spatial relationship between them.

6. *distance*(): This elementary operation is used in the functional form for distance comparison questions. Given a list of object entities, it returns triplets of objects such that the first object is closer/farther to the anchor object than the second object.

[1] or a response that the question is inapplicable (*e.g.* referring to objects not in the environment) or ambiguous (having multiple valid answers).

Having described the elementary operations that make up our functional forms, the explanations of the functional forms for each question template is given below. We categorize the question types into 3 categories based on the objects and the rooms that they refer to.

1. **Unambiguous Object**: There are certain question types that inquire about an object that must be unique and unambiguous throughout the environment. Examples of such question types are `location` and `color`. For example, we should ask *'what room is the piano located in?'* if there is only a single instance of a *'piano'* in the environment. Hence, the functional forms for location and color have the following structure:

$$select(objects) \rightarrow unique(objects) \rightarrow query(location/color).$$

   $select(objects)$ gets the list of all objects in the house and the $unique(objects)$ only retains objects that are unique, thereby ensuring unambiguity. The $query(template)$ function prepares the question string by filling in the slots in the template string.

2. **Unambiguous Room + Unambiguous Object**: In continuation of the above, there is another set of question types that talk about objects in rooms where in addition to the objects being unique, the rooms should also be unambiguous. Examples of such question types include `color_room`, `preposition`, and `distance`. The additional unambiguity constraint on the room is because the question *'what is next to the bathtub in the bathroom?'* would become ambiguous if there are two or more bathrooms in the house. The functional forms for such types are given by the following structure:

$$select(rooms) \rightarrow unique(rooms) \rightarrow select(objects) \rightarrow unique(objects) \rightarrow query(template).$$

   The first two operations in the sequence result in a list of unambiguous rooms whereas the next two result in a list of unambiguous objects in those rooms. Note that when $select(\cdot)$ appears as the first operation in the sequence (*i.e.*, select operates on an empty list), it is used to fetch the set of rooms or objects across the entire house. However, in this case, $select(object)$ operates on a set of rooms (the output of $select(rooms) \rightarrow unique(rooms) \rightarrow$), so it returns the set of objects found in those specific rooms (as opposed to fetching objects across all rooms in the house).

3. **Unambiguous Room**: The final set of question types are the ones where the rooms need to be unambiguous, but the objects in those rooms that are being referred to do

Table A.2: (L-R) Quantitative results for the segmentation, depth estimation, and autoencoder heads of our multi-task perception network. All metrics are reported on a held out validation set.

	Pixel Accuracy	Mean Pixel Accuracy	Mean IOU
single	0.780	0.246	0.163
hybrid	0.755	0.254	0.166

	Smooth-ℓ_1
single	0.003
hybrid	0.005

	Smooth-ℓ_1
single	0.003
hybrid	0.003

not. Examples of such question types are: `existence`, `logical`, and `count`. It is evident that for asking about the existence of objects or while counting them, we do not require the object to have only a single instance. *'Is there a television in the living room?'* is a perfectly valid question, even if there are multiple televisions in the living room (provided that there is a single living room in the house). The template structure for this is a simplified version of (2):

$$select(rooms) \rightarrow unique(rooms) \rightarrow select(objects) \rightarrow query(template).$$

Note that we have dropped $unique(objects)$ from the sequence as we no longer require that condition to hold true.

See Table A.1 for a complete list of question functional forms.

Checks and balances. Since one of our goals is to benchmark performance of our agents with human performance, we want to avoid questions that are cumbersome for a human to navigate for or to answer. Additionally, we would also like to have a balanced distribution of answers so that the agent is not able to simply exploit dataset biases and answer questions effectively without exploration. This section describes in detail the various checks that we have in place to ensure these properties.

1. **Entropy+Frequency-based Filtering**: It is important to ensure that the distribution of answers for a question is not too 'peaky', otherwise the mode guess from this distribution will do unreasonably well as a baseline. Thus, we compute the normalized entropy of the distribution of answers for a question. We drop questions where the normalized entropy is below 0.5. Further, we also drop questions that occur in less than 4 environments because the entropy counts for low-frequency questions are not reliable.

2. **Non-existence questions**: We add `existence` questions with *'no'* as the answer for objects that are absent in a given room in the current environment, but which are present in the same room in other environments. For example, if the living room in

the current environment does not contain a piano, but pianos are present in living rooms of other environments across the dataset, we add the question *'is there a piano in the living room?'* for the current environment with a ground truth answer *'no'*. The same is also done for `logical` questions.

3. **Object Instance Count Threshold**: We do not ask counting questions (`room_count` and `count`) when the answer is greater than 5, as they are tedious for humans.

4. **Object Distance Threshold**: We consider triplets of objects within a room consisting of an anchor object, such that the difference of distances between two object-anchor pairs is at least 2 metres. This is to avoid ambiguity in *'closer'*/*'farther'* object distance comparison questions.

5. **Collapsing Object Labels**: Object types that are visually very similar (*e.g.* *'teapot'* and *'coffee_kettle'*) or semantically hierarchical in relation (*e.g.* *'bread'* and *'food'*) introduce unwanted ambiguity. In these cases we collapse the object labels to manually selected labels (*e.g.* (*'teapot'*, *'coffee_kettle'*) → *'kettle'* and (*'bread'*, *'food'*) → *'food'*).

6. **Blacklists**:

   - **Rooms**: Some question types in the EQA dataset have room names in the question string (e.g. `color_room`, `exist`, `logical`). We do not generate such questions for rooms that have obscure or esoteric names such as *'loggia'*, *'freight elevator'*, *'aeration' etc*. or names from which the room being referred might not be immediately obvious *e.g.* *'boiler room'*, *'entryway' etc*.
   - **Objects**: For each question template, we maintain a list of objects that are not to be included in questions. These are either tiny objects or whose name descriptions are too vague e.g. *'switch'* (too small), *'decoration'* (not descriptive enough), *'glass'* (transparent), *'household appliance'* (too vague). These blacklists are manually defined based on our experiences performing these tasks.

A.2 CNN Training Details

The CNN comprising the visual system for our EmbodiedQA agents is trained under a multi-task pixel-to-pixel prediction framework. We have an encoder network that transforms the egocentric RGB image from the House3D renderer [107] to a fixed-size representation. We have 3 decoding heads that predict 1) original RGB values (*i.e.* an autoencoder), 2) semantic class, and 3) depth for each pixel. The information regarding semantic

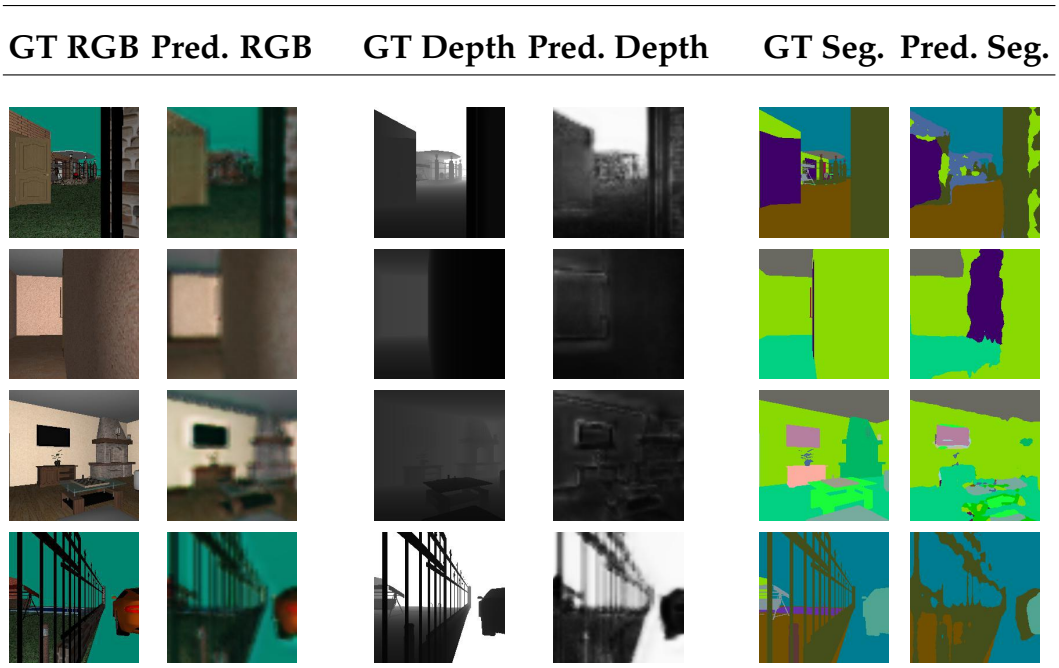


Figure A.1: Some qualitative results from the hybrid CNN. Each row represents an input image. For every input RGB image, we show the reconstruction from the autoencoder head, ground truth depth, predicted depth as well as ground truth segmentation and predicted segmentation maps.

class and depth of every pixel is available from the renderer. The range of depth values for every pixel lies in the range $[0, 1]$ and the segmentation is done over 191 classes.

Architecture. The encoder network has 4 Conv blocks, comprising of 5×5 Conv filters, ReLU, BatchNorm and 2×2 MaxPool. Each of the 3 decoder branches of our network upsample the encoder output to the spatial size of the original input image. The number of channels in the output of the decoder depends on the task head – 191, 1 and 3 for the semantic segmentation, depth and autoencoder branches respectively. The upsampling is done using bilinear interpolation. The architecture also has skip connections from the 2nd and the 3rd Conv layers.

Training Details. We use cross-entropy loss to train the segmentation branch of our hybrid network. The depth and autoencoder branches are trained using the Smooth-ℓ_1 loss. The total loss is a linear combination of the 3 losses, given by $\mathit{overall\_loss} = \mathit{seg\_loss} + 10 \times \mathit{depth\_loss} + 10 \times \mathit{reconstruction\_loss}$. We use the Adam optimizer with a learning rate of 10^{-3} and a batch size of 20. The hybrid network is trained for a total of 5 epochs on a dataset of $100k$ RGB training images from the renderer.

Quantitative Results. Table A.2 shows some quantitative results. For each of the 3 different decoding heads of our multi-task CNN, we report results on the validation set for two settings - when the network is trained for all tasks at once (hybrid) or each task in-

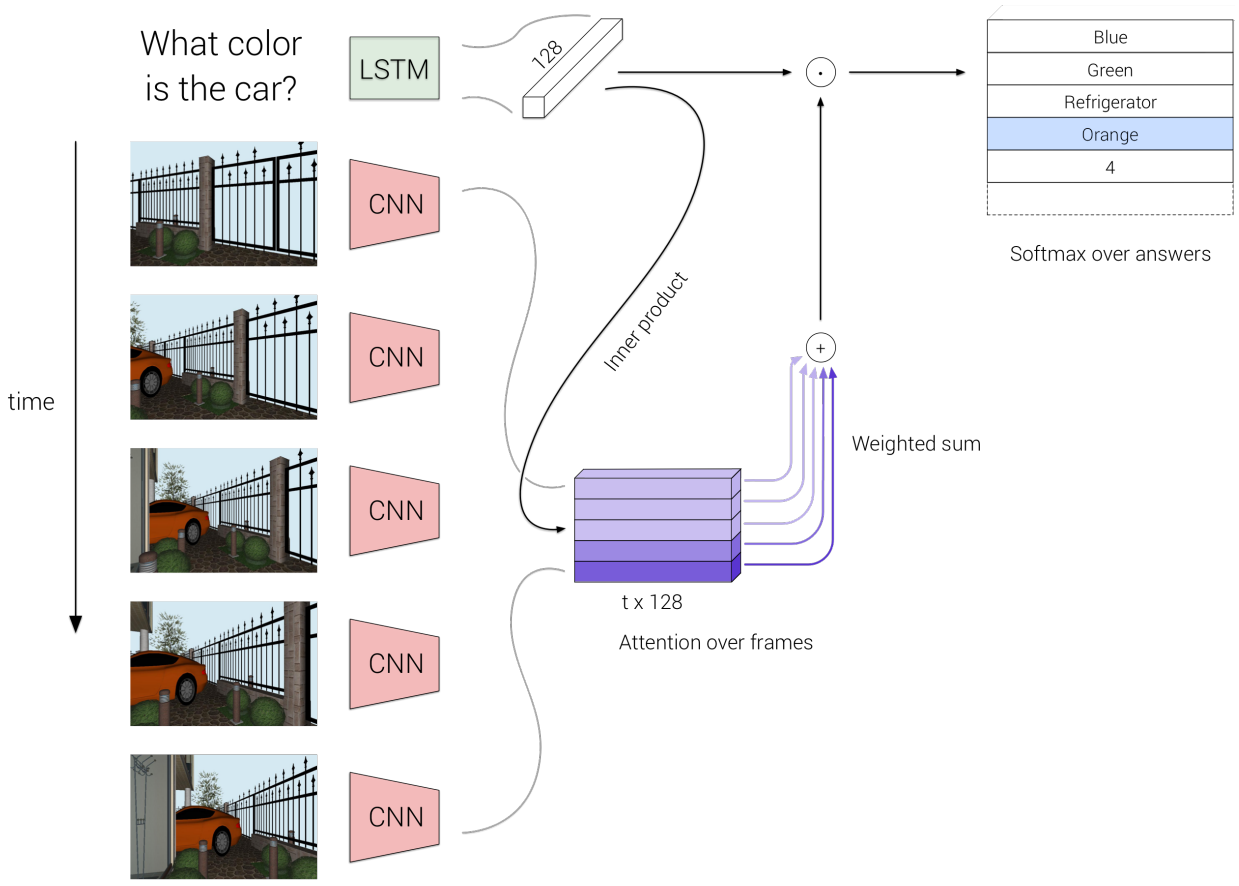


Figure A.2: Conditioned on the navigation frames and question, the question answering module computes dot product attention over the last five frames, and combines attention-weighted combination of image features with question encoding to predict the answer.

dependently (single). For segmentation, we report the overall pixel accuracy, mean pixel accuracy (averaged over all 191 classes) and the mean IOU (intersection over union). For depth and autoencoder, we report the Smooth-ℓ_1 on the validation set.

Qualitative Results. Some qualitative results on images from the validation set for segmentation, depth prediction and autoencoder reconstruction are shown in Figure A.1.

A.3 Question Answering Module

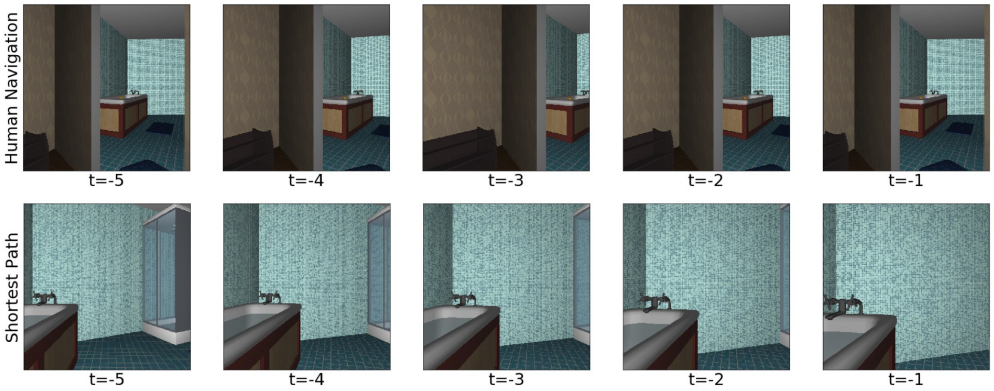
The question answering module predicts the agents' beliefs over the answer given the agent's navigation. It first encodes the question with an LSTM, last five frames of the navigation each with a CNN, and then computes dot product attention over the five frames to pick the most relevant ones. Next, it combines attention-weighted sum of image features with the question encoding to predict a softmax distribution over answers. See Fig. A.2.

A.4 Human Navigation + Machine QA

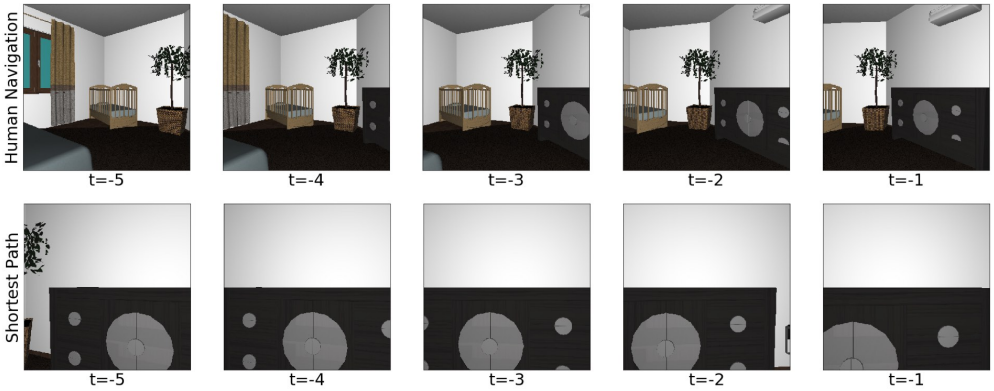
In order to contrast human and shortest-path navigations with respect to question answering, we evaluate our QA model on the last 5 frames of human navigations collected through our Amazon Mechanical Turk interface. We find the mean rank of the ground truth answer to be 3.51 for this setting (compared to 3.26 when computed from shortest-paths). We attribute this difference primarily to a mismatch between the QA system training on shortest paths and testing on human navigations. While the shortest paths typically end with the object of interest filling the majority of the view, humans tend to stop earlier as soon as the correct answer can be discerned. As such, human views tend to be more cluttered and pose a more difficult task for the QA module. Fig. A.5 highlights this difference by contrasting the last 5 frames from human and shortest-path navigations across three questions and environments.

Figure A.3: Visualizations of queryable objects from the House3D renderer. Notice that instances within the same class differ significantly in shape, size, and color.

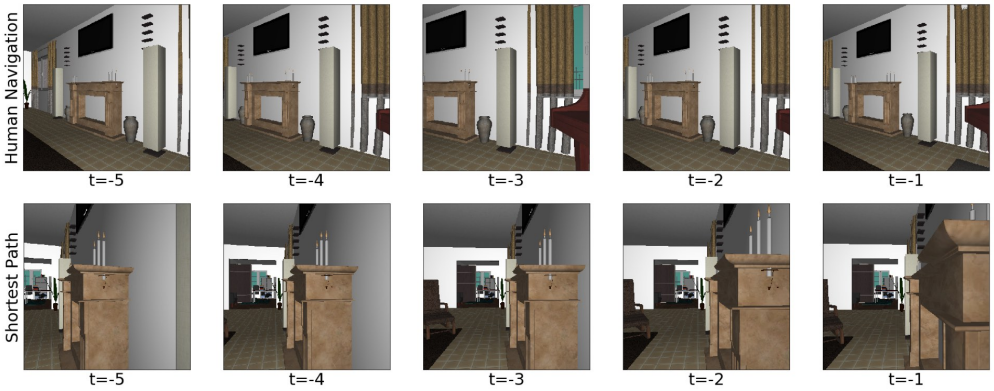
Figure A.4: Visualizations of queryable rooms from the House3D renderer.



Q: What color is the bathtub?

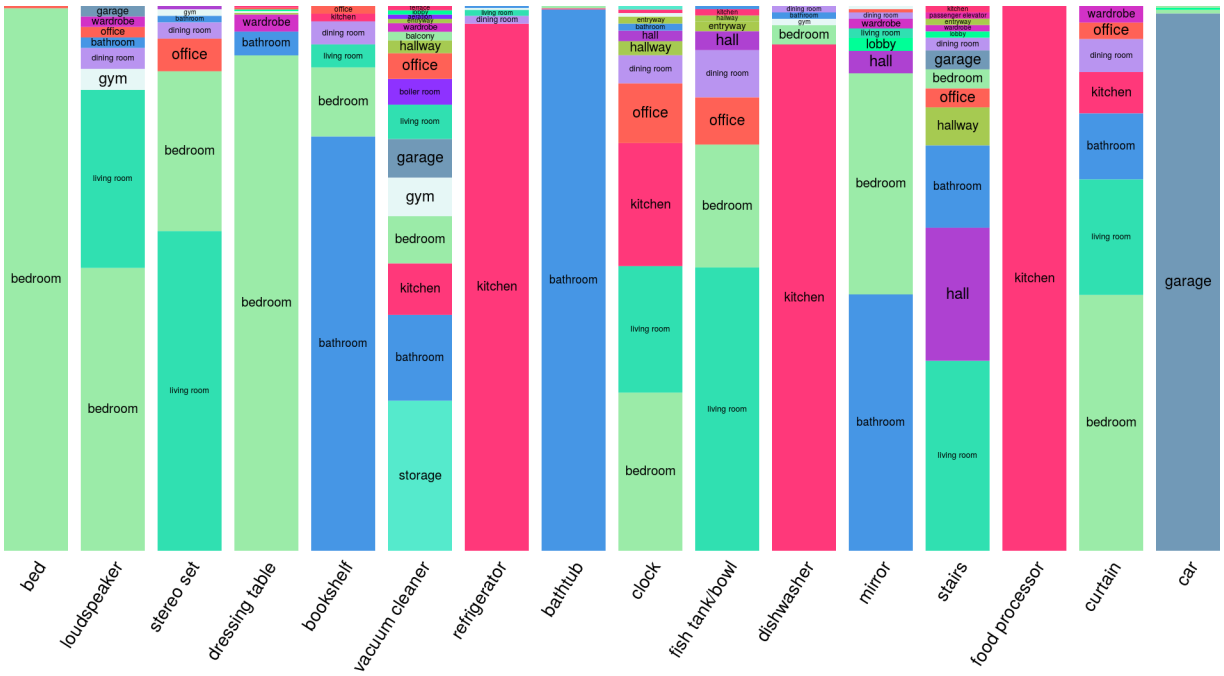


Q: What color is the dresser?

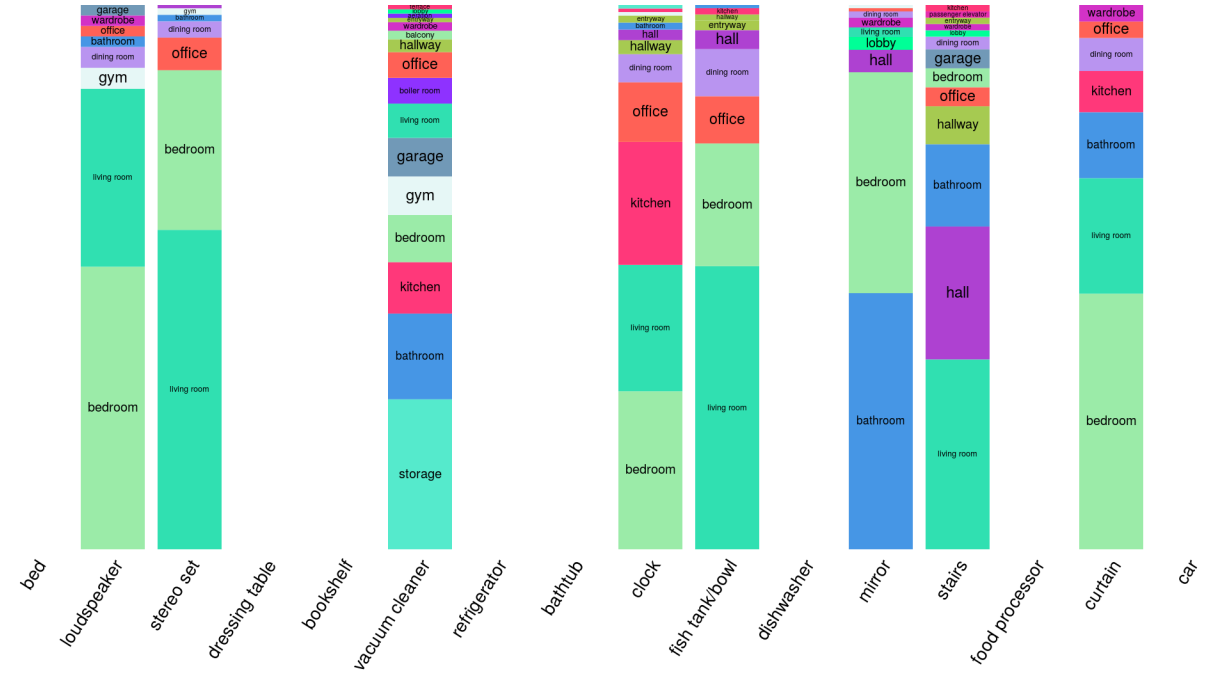


Q: What color is the fireplace in the living room?

Figure A.5: Examples of last five frames from human navigation vs. shortest path.

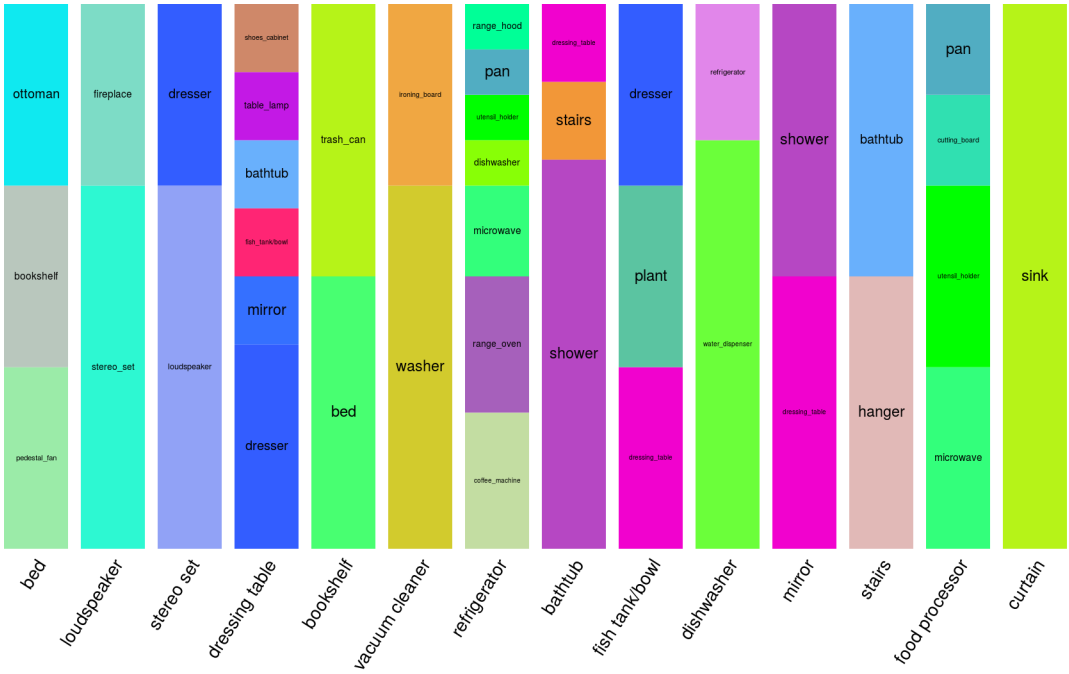


(a) `location` questions before entropy+count based filtering

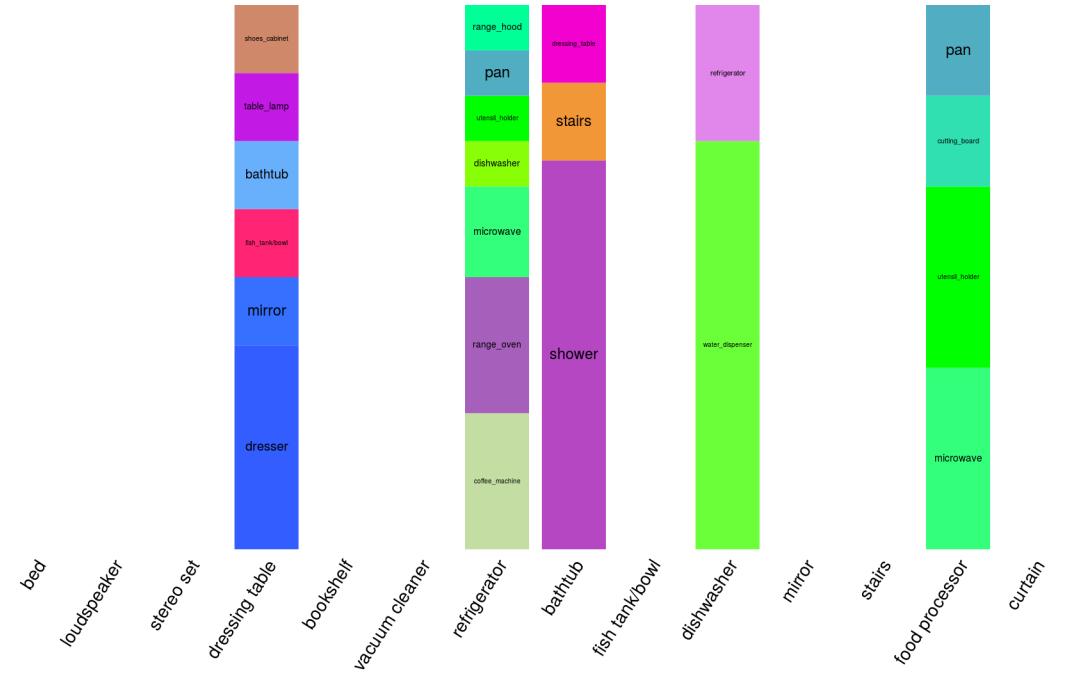


(b) `location` questions after entropy+count based filtering

Figure A.6: The answer distribution for `location` template questions. Each bar represents a question of the form *'what room is the <OBJ> located in?'* and shows a distribution over the answers across different environments. The blank spaces in A.6b represent the questions that get pruned out as a result of the entropy+count based filtering.

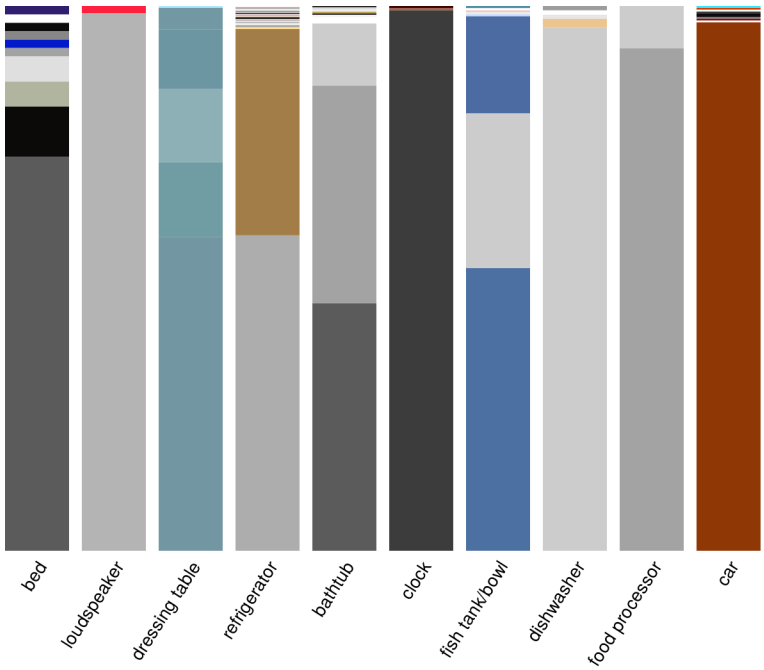


(a) `preposition` questions before entropy+count based filtering

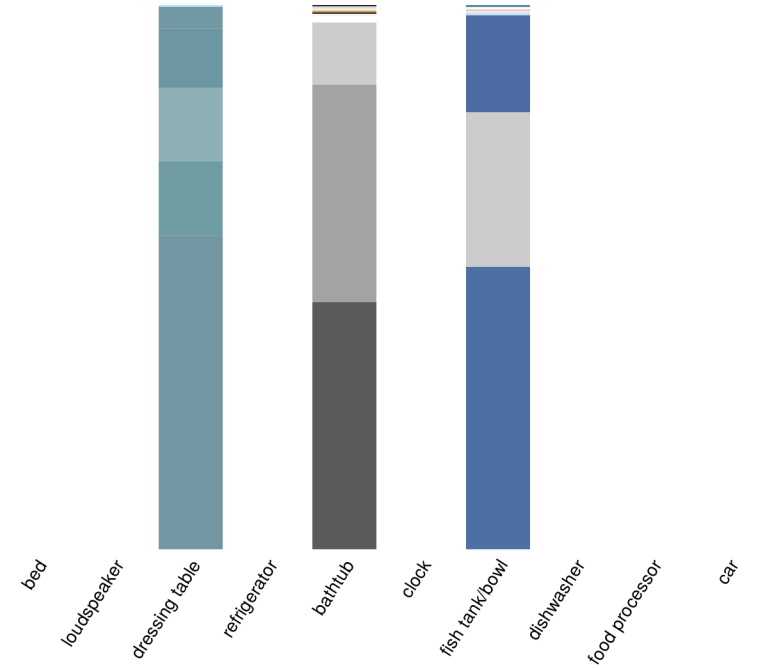


(b) `preposition` questions after entropy+count based filtering

Figure A.7: The answer distribution for `preposition` template questions. Each bar represents a question of the form *'what is next to the <OBJ>?'* and shows a distribution over the answers across different environments. The blank spaces in A.7b represent the questions that get pruned out as a result of the entropy+count based filtering.



(a) `color` questions before entropy+count based filtering



(b) `color` questions after entropy+count based filtering

Figure A.8: The answer distribution for `color` template questions. Each bar represents a question of the form *'what color is the <OBJ>?'* and shows a distribution over the answers across different environments (the color of each section on a bar denotes the possible answers). The blank spaces in A.8b represent the questions that get pruned out as a result of the entropy+count based filtering.

Appendix B

Probing Emergent Semantics in Predictive Agents via Question Answering

B.1 Network architectures and Training setup

B.1.1 Importance Weighted Actor-Learner Architecture

Agents were trained using the IMPALA framework [201]. Briefly, there are N parallel 'actors' collecting experience from the environment in a replay buffer and one learner taking batches of trajectories and performing the learning updates. During one learning update the agent network is unrolled, all the losses (RL and auxiliary ones) are evaluated and the gradients computed.

B.1.2 Agents

Input encoder To process the frame input, all models in this work use a residual network [208] of 6 64-channel ResNet blocks with rectified linear activation functions and bottleneck channel of size 32. We use strides of (2, 1, 2, 1, 2, 1) and don't use batch-norm. Following the convnet we flatten the ouput and use a linear layer to reduce the size to 500 dimensions. Finally, We concatenate this encoding of the frame together with a one hot encoding of the previous action and the previous reward.

Core architecture The recurrent core of all agents is a 2-layer LSTM with 256 hidden units per layer. At each time step this core consumes the input embedding described above and update its state. We then use a 200 units single layer MLP to compute a value baseline and an equivalent network to compute action logits, from where one discrete action is sampled.

Simulation Network Both predictive agents have a simulation network with the same architecture as the agent's core. This network is initialized with the agent state at some random time t from the trajectory and unrolled forward for a random number of steps up to 16, receiving only the actions of the agent as inputs. We then use the resulting LSTM

hidden state as conditional input for the prediction loss (SimCore or CPC | A).

SimCore We use the same architecture and hyperparameters described in [4]. The output of the simulation network is used to condition a Convolutional DRAW [202]. This is a conditional deep variational auto-encoder with recurrent encoder and decoder using convolutional operations and a canvas that accumulates the results at each step to compute the distribution over inputs. It features a recurrent prior network that receives the conditioning vector and computes a prior over the latent variables. See more details in [4].

Action-conditional CPC We replicate the architecture used in [3]. CPC | A uses the output of the simulation network as input to an MLP that is trained to discriminate true versus false future frame embedding. Specifically, the simulation network outputs a conditioning vector after k simulation steps which is concatenated with the frame embedding z_{t+k} produced by the image encoder on the frame x_{t+k} and sent through the MLP discriminator. The discriminator has one hidden layer of 512 units, ReLU activations and a linear output of size 1 which is trained to binary classify true embeddings into one class and false embeddings into another. We take the negative examples from random time points in the same batch of trajectories.

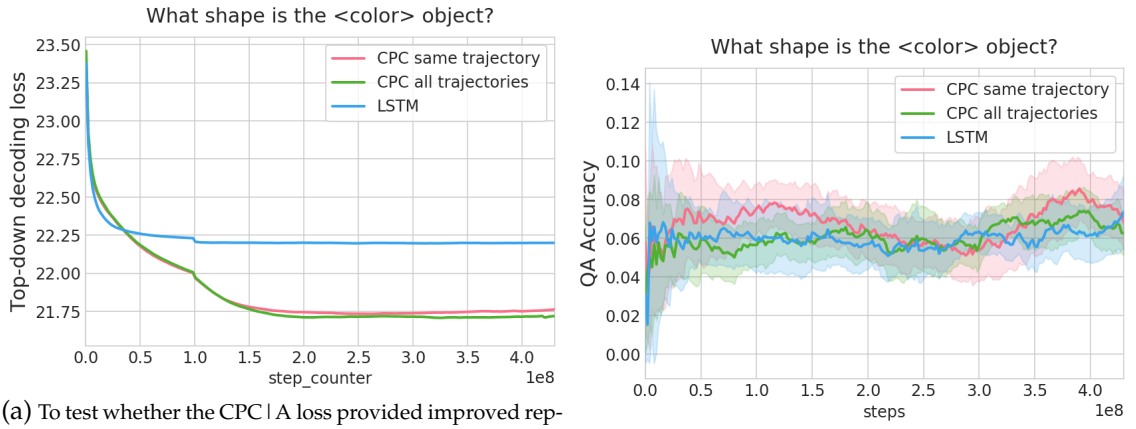
B.1.3 QA network architecture

Question encoding The question string is first tokenized to words and then mapped to integers corresponding to vocabulary indices. These are then used to lookup 32-dimensional embeddings for each word. We then unroll a 64-units single-layer LSTM for a fixed number of 15 steps. The language representation is then computed by summing the hidden states for all time steps.

QA decoder. To decode answers from the internal state of the agents we use a second LSTM initialized with the internal state of the agent's LSTM and unroll it for a fix number of steps, consuming the question embedding at each step. The results reported in the main section were computed using 12 decoding steps. The terminal state is sent through a two-layer MLP (sizes 256, 256) to compute a vector of answer logits with the size of the vocabulary and output the top-1 answer.

B.1.4 Hyper-parameters

The hyper-parameter values used in all the experiments are in Table B.1.



(a) To test whether the CPC | A loss provided improved representations we reconstructed the environment top-down view, similar to [4]. Indeed the reconstruction loss is lower for CPC | A than for the LSTM agent.

(b) QA accuracy for the CPC | A agent is not better than the LSTM agent, for both sampling strategies of negatives.

Figure B.1

B.1.5 Negative sampling strategies for CPC | A

We experimented with multiple sampling strategies for the CPC | A agent (whether or not negative examples are sampled from the same trajectory, the number of contrastive prediction steps, the number of negative examples). We report the best results in the main text. The CPC | A agent did provide better representations of the environment than the LSTM-based agent, as shown by the top-down view reconstruction loss (Figure B.1a). However, none of the CPC | A agent variations that we tried led to better-than-chance question-answering accuracy. As an example, in Figure B.1b we compare sampling negatives from the same trajectory or from any trajectory in the training batch.

B.2 Effect of QA network depth

To study the effect of the QA network capacity on the answer accuracy, we tested decoders of different depths applied to both the SimCore and the LSTM agent's internal representations (B.2). The QA network is an LSTM initialized with the agent's internal state that we unroll for a fix number of steps feeding the question as input at each step. We found that, indeed, the answering accuracy increased with the number of unroll steps from 1 to 12, while greater number of steps became detrimental. We performed the same analysis on the LSTM agent and found that regardless of the capacity of the QA network, we could not decode the correct answer from its internal state, suggesting that the limiting factor is not the capacity of the decoder but the lack of useful representations in the LSTM agent state.

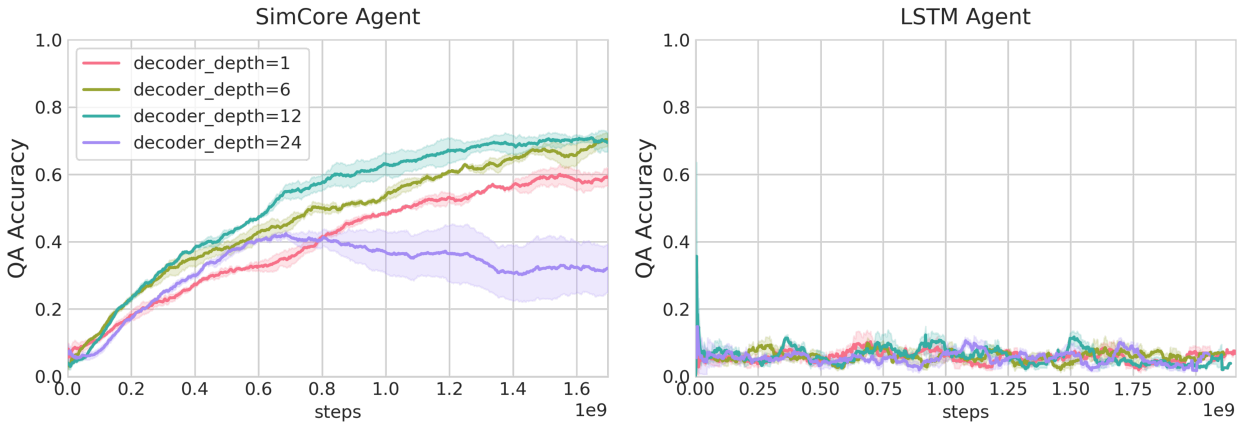


Figure B.2: Answer accuracy over training for increasing QA decoder's depths. Left subplot shows the results for the SimCore agent and right subplot for the LSTM baseline. For SimCore, the QA accuracy increases with the decoder depth, up to 12 layers. For the LSTM agent, QA accuracy is not better than chance regardless of the capacity of the QA network.

B.3 Answering accuracy during training for all questions

The QA accuracy over training for all questions is shown in Figure B.3.

B.4 Environment

Our environment is a single L-shaped 3D room, procedurally populated with an assortment of objects.

Actions and Observations. The environment is episodic, and runs at 30 frames per second. Each episode takes 30 seconds (or 900 steps). At each step, the environment provides the agent with two observations: a 96x72 RGB image with the first-person view of the agent and the text containing the question.

The agent can interact with the environment by providing multiple simultaneous actions to control movement (forward/back, left/right), looking (up/down, left/right), picking up and manipulating objects (4 degrees of freedom: yaw, pitch, roll + movement along the axis between agent and object).

Rewards. To allow training using cross-entropy, as described in Section 7.4, the environment provides the ground-truth answer instead of the reward to the agent.

Object creation and placement. We generate between 2 and 20 objects, depending on the task, with the type of the object, its color and size being uniformly sampled from the set described in Table B.2.

Objects will be placed in a random location and random orientation. For some tasks, we required some additional constraints - for example, if the question is "What is the color of the cushion near the bed?", we need to ensure only one cushion is close to the bed. This was done by checking the constraints and regenerating the placement in case they were not satisfied.

Table B.1: Hyperparameters.

Agent	
Learning rate	1e-4
Unroll length	50
Adam β_1	0.90
Adam β_2	0.95
Policy entropy regularization	0.0003
Discount factor	0.99
No. of ResNet blocks	6
No. of channel in ResNet block	64
Frame embedding size	500
No. of LSTM layers	2
No. of units per LSTM layer	256
No. of units in value MLP	200
No. of units in policy MLP	200
Simulation Network	
Overshoot length	16
No. of LSTM layers	2
No. of units per LSTM layer	256
No. of simulations per trajectory	6
No. of evaluations per overshoot	2
SimCore	
No. of ConvDRAW Steps	8
GECO kappa	0.0015
CPC|A	
MLP discriminator size	64
QA network	
Vocabulary size	1000
Maximum question length	15
No. of units in Text LSTM encoder	64
Question embedding size	32
No. of LSTM layers in question decoder	2
No. of units per LSTM layer	256
No. of units in question decoder MLP	200
No. of decoding steps	12

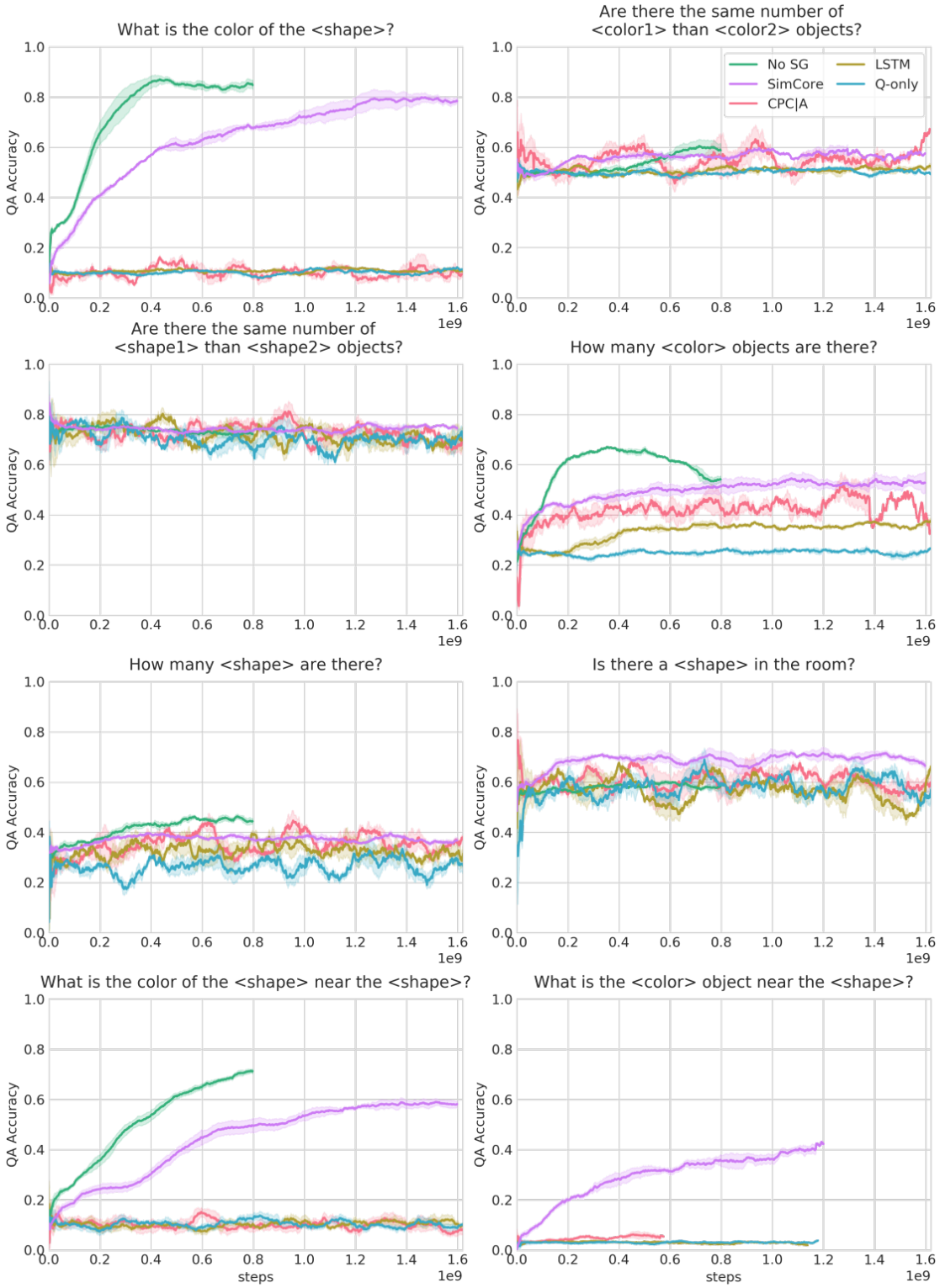


Figure B.3: QA accuracy over training for all questions and all models.

Table B.2: Randomization of objects in the Unity room. 50 different types, 10 different colors and 3 different scales.

Attribute	Options
Object	basketball, cushion, carriage, train, grinder, candle, teddy, chair, scissors, stool, book, football, rubber duck, glass, toothpaste, arm chair, robot, hairdryer, cube block, bathtub, TV, plane, cuboid block, car, tv cabinet, plate, soap, rocket, dining table, pillar block, potted plant, boat, tennisball, tape dispenser, pencil, wash basin, vase, picture frame, bottle, bed, helicopter, napkin, table lamp, wardrobe, racket, keyboard, chest, bus, roof block, toilet
Color	aquamarine, blue, green, magenta, orange, purple, pink, red, white, yellow
Size	small, medium, large

Table B.3: Environment action set.

Body movement actions	**Movement and grip actions**	**Object manipulation**
NOOP	GRAB	GRAB + SPIN_OBJECT_RIGHT
MOVE_FORWARD	GRAB + MOVE_FORWARD	GRAB + SPIN_OBJECT_LEFT
MOVE_BACKWARD	GRAB + MOVE_BACKWARD	GRAB + SPIN_OBJECT_UP
MOVE_RIGHT	GRAB + MOVE_RIGHT	GRAB + SPIN_OBJECT_DOWN
MOVE_LEFT	GRAB + MOVE_BACKWARD	GRAB + SPIN_OBJECT_FORWARD
LOOK_RIGHT	GRAB + LOOK_RIGHT	GRAB + SPIN_OBJECT_BACKWARD
LOOK_LEFT	GRAB + LOOK_LEFT	GRAB + PUSH_OBJECT_AWAY
LOOK_UP	GRAB + LOOK_UP	GRAB + PULL_OBJECT_CLOSE
LOOK_DOWN	GRAB + LOOK_DOWN	

Bibliography

[1] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. x, 81, 82, 83, 84, 90, 91

[2] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. xiii, 53, 55

[3] Z. D. Guo, M. G. Azar, B. Piot, B. A. Pires, T. Pohlen, and R. Munos, "Neural predictive belief representations," *arXiv preprint arXiv:1811.06407*, 2018. xv, 4, 97, 98, 99, 100, 101, 102, 108, 130

[4] K. Gregor, D. J. Rezende, F. Besse, Y. Wu, H. Merzic, and A. v. d. Oord, "Shaping Belief States with Generative Environment Models for RL," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. xv, 4, 97, 98, 99, 100, 101, 102, 104, 107, 108, 130, 131

[5] C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, J. Schrittwieser, K. Anderson, S. York, M. Cant, A. Cain, A. Bolton, S. Gaffney, H. King, D. Hassabis, S. Legg, and S. Petersen, "Deepmind lab," *CoRR*, vol. abs/1612.03801, 2016. xvi, 54, 105, 107

[6] H. Fang, S. Gupta, F. N. Iandola, R. K. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt, C. L. Zitnick, and G. Zweig, "From Captions to Visual Concepts and Back," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 4, 12

[7] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 4, 12

[8] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 4, 12, 45

[9] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term Recurrent Convolutional Networks for Visual Recognition and Description," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 4, 12

[10] S. Venugopalan, M. Rohrbach, J. Donahue, R. J. Mooney, T. Darrell, and K. Saenko,

"Sequence to sequence - video to text," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015. 4, 12

[11] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. J. Mooney, and K. Saenko, "Translating Videos to Natural Language Using Deep Recurrent Neural Networks," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies (NAACL HLT)*, 2015. 4, 12

[12] A. Rohrbach, M. Rohrbach, N. Tandon, and B. Schiele, "A dataset for movie description," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 4, 12

[13] C. Kong, D. Lin, M. Bansal, R. Urtasun, and S. Fidler, "What are you talking about? text-to-image coreference," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 4, 12

[14] V. Ramanathan, A. Joulin, P. Liang, and L. Fei-Fei, "Linking people with "their" names using coreference resolution," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2014. 4, 12

[15] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik, "Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015. 4, 12

[16] R. Hu, M. Rohrbach, and T. Darrell, "Segmentation from natural language expressions," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 4, 12

[17] A. Rohrbach, M. Rohrbach, R. Hu, T. Darrell, and B. Schiele, "Grounding of textual phrases in images by reconstruction," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 4, 12

[18] G. Christie, A. Laddha, A. Agrawal, S. Antol, Y. Goyal, K. Kochersberger, and D. Batra, "Resolving language and vision ambiguities together: Joint segmentation and prepositional attachment resolution in captioned scenes," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016. 4, 12

[19] T.-H. Huang, F. Ferraro, N. Mostafazadeh, I. Misra, A. Agrawal, J. Devlin, R. Girshick, X. He, P. Kohli, D. Batra, L. Zitnick, D. Parikh, L. Vanderwende, M. Galley, and M. Mitchell., "Visual storytelling," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies (NAACL HLT)*, 2016. 4, 12

[20] H. Agrawal, A. Chandrasekaran, D. Batra, D. Parikh, and M. Bansal, "Sort story: Sorting jumbled images and captions into stories," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016. 4, 12

[21] M. Malinowski and M. Fritz, "A Multi-World Approach to Question Answering about Real-World Scenes based on Uncertain Input," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. 4, 12, 18, 96

[22] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "VQA: Visual Question Answering," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015. 4, 12, 14, 15, 18, 19, 96

[23] M. Ren, R. Kiros, and R. Zemel, "Exploring Models and Data for Image Question Answering," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 4, 12, 18

[24] H. Gao, J. Mao, J. Zhou, Z. Huang, L. Wang, and W. Xu, "Are You Talking to a Machine? Dataset and Methods for Multilingual Image Question Answering," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 4, 12, 15, 18

[25] M. Malinowski, M. Rohrbach, and M. Fritz, "Ask your neurons: A neural-based approach to answering questions about images," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015. 4, 12

[26] J. Lu, J. Yang, D. Batra, and D. Parikh, "Hierarchical question-image co-attention for visual question answering," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 4, 12, 27

[27] P. Zhang, Y. Goyal, D. Summers-Stay, D. Batra, and D. Parikh, "Yin and yang: Balancing and answering binary visual questions," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4, 12, 15, 19, 53

[28] A. Agrawal, D. Batra, and D. Parikh, "Analyzing the behavior of visual question answering models," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016. 4, 15

[29] A. Das, H. Agrawal, C. L. Zitnick, D. Parikh, and D. Batra, "Human attention in visual question answering: Do humans and deep networks look at the same regions?," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016. 4, 12, 51

[30] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, "Making the v in vqa matter: Elevating the role of image understanding in visual question answering," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 4, 12, 15, 51, 53

[31] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach, "Multimodal compact bilinear pooling for visual question answering and visual grounding," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016. 4

[32] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 4

[33] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler, "MovieQA: Understanding Stories in Movies through Question-Answering," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4

[34] Y. Jang, Y. Song, Y. Yu, Y. Kim, and G. Kim, "TGIF-QA: toward spatio-temporal reasoning in visual question answering," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 4

[35] H. de Vries, F. Strub, S. Chandar, O. Pietquin, H. Larochelle, and A. C. Courville, "GuessWhat?! Visual object discovery through multi-modal dialogue," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 4, 12, 32, 35

[36] N. Mostafazadeh, C. Brockett, B. Dolan, M. Galley, J. Gao, G. P. Spithourakis, and L. Vanderwende, "Image-Grounded Conversations: Multimodal Context for Natural Question and Response Generation," *arXiv preprint arXiv:1701.08251*, 2017. 4, 12

[37] J. Weizenbaum, "ELIZA." http://psych.fullerton.edu/mbirnbaum/psych101/Eliza.htm. 5, 10, 12

[38] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau, "Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models," in *AAAI Conference on Artificial Intelligence*, 2016. 5, 12, 39

[39] A. Kannan, K. Kurach, S. Ravi, T. Kaufmann, A. Tomkins, B. Miklos, G. Corrado, L. Lukács, M. Ganea, P. Young, *et al.*, "Smart Reply: Automated Response Suggestion for Email," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016. 5, 12

[40] O. Vinyals and Q. Le, "A Neural Conversational Model," *arXiv preprint arXiv:1506.05869*, 2015. 5, 12

[41] J. Dodge, A. Gane, X. Zhang, A. Bordes, S. Chopra, A. Miller, A. Szlam, and J. Weston, "Evaluating Prerequisite Qualities for Learning End-to-End Dialog Systems," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. 5, 10, 12

[42] A. Bordes and J. Weston, "Learning End-to-End Goal-Oriented Dialog," *arXiv preprint arXiv:1605.07683*, 2016. 5, 12, 24, 26

[43] I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, and Y. Bengio, "A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues," *arXiv preprint arXiv:1605.06069*, 2016. 5, 12, 26, 39

[44] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, "Deep Reinforcement Learning for Dialogue Generation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016. 5, 12, 36, 47

[45] R. Lowe, N. Pow, I. Serban, and J. Pineau, "The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems," in *Proceedings of the Annual SIGdial Meeting on Discourse and Dialogue*, 2015. 5, 12

[46] C.-W. Liu, R. Lowe, I. V. Serban, M. Noseworthy, L. Charlin, and J. Pineau, "How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016. 5, 12, 23

[47] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005. 6

[48] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov, "Gated-attention architectures for task-oriented language grounding," in *AAAI Conference on Artificial Intelligence*, 2018. 6, 53, 54, 69, 99, 110

[49] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2017. 6, 63, 69, 99, 110

[50] Y. Zhu, D. Gordon, E. Kolve, D. Fox, L. Fei-Fei, A. Gupta, R. Mottaghi, and A. Farhadi, "Visual Semantic Planning using Deep Successor Representations," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017. 6, 54, 69, 98, 99, 110

[51] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6, 69, 99, 110

[52] K. M. Hermann, F. Hill, S. Green, F. Wang, R. Faulkner, H. Soyer, D. Szepesvari, W. Czarnecki, M. Jaderberg, D. Teplyashin, *et al.*, "Grounded language learning in a simulated 3D world," *arXiv preprint arXiv:1706.06551*, 2017. 6, 53, 69, 99, 110

[53] S. Brahmbhatt and J. Hays, "DeepNav: Learning to Navigate Large Cities," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6, 110

[54] J. Oh, S. Singh, H. Lee, and P. Kohli, "Zero-shot task generalization with multi-task deep reinforcement learning," in *Proceedings of the International Conference on*

Machine Learning (ICML), 2017. 6, 53, 54, 60, 69, 99, 110

[55] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2017. 6

[56] T. Winograd, "Understanding natural language," *Cognitive Psychology*, 1972. 6, 69, 98

[57] M. Denil, S. G. Colmenarejo, S. Cabi, D. Saxton, and N. de Freitas, "Programmable agents," *arXiv preprint arXiv:1706.06383*, 2017. 6, 59

[58] H. Yu, H. Zhang, and W. Xu, "Interactive Grounded Language Acquisition and Generalization in a 2D World," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. 6, 54, 69, 99

[59] J. Andreas, D. Klein, and S. Levine, "Modular multitask reinforcement learning with policy sketches," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2017. 6, 54, 59, 60, 68, 69, 99

[60] D. K. Misra, J. Langford, and Y. Artzi, "Mapping instructions and visual observations to actions with reinforcement learning," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017. 6

[61] S. I. Wang, P. Liang, and C. D. Manning, "Learning language games through interaction," in *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*, 2016. 6, 59

[62] J. P. Bigham, C. Jayant, H. Ji, G. Little, A. Miller, R. C. Miller, R. Miller, A. Tatarowicz, B. White, S. White, and T. Yeh, "VizWiz: Nearly Real-time Answers to Visual Questions," in *Proceedings of the Annual ACM symposium on User interface software and technology*, 2010. 8

[63] S. Wu, H. Pique, and J. Wieland, "Using Artificial Intelligence to Help Blind People 'See' Facebook." http://newsroom.fb.com/news/2016/04/using-artificial-intelligence-to-help-blind-people-see-facebook/, 2016. 8

[64] H. Mei, M. Bansal, and M. R. Walter, "Listen, attend, and walk: Neural mapping of navigational instructions to action sequences," in *AAAI Conference on Artificial Intelligence*, 2016. 8

[65] T. Paek, "Empirical methods for evaluating dialog systems," in *Proceedings of the workshop on Evaluation for Language and Dialogue Systems-Volume 9*, 2001. 10

[66] O. Lemon, K. Georgila, J. Henderson, and M. Stuttle, "An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the TALK

in-car system," in *Proceedings of the European Chapter of the Association for Computational Linguistics*, 2006. 10

[67] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. DollÃ¡r, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2014. 11, 12, 13

[68] D. Geman, S. Geman, N. Hallonquist, and L. Younes, "A Visual Turing Test for Computer Vision Systems," in *Proceedings of the National Academy of Science (PNAS)*, 2014. 12

[69] I. V. Serban, A. García-Durán, Ç. Gülçehre, S. Ahn, S. Chandar, A. C. Courville, and Y. Bengio, "Generating Factoid Questions With Recurrent Neural Networks: The 30M Factoid Question-Answer Corpus," in *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*, 2016. 12

[70] A. Bordes, N. Usunier, S. Chopra, and J. Weston, "Large-scale Simple Question Answering with Memory Networks," *arXiv preprint arXiv:1506.02075*, 2015. 12

[71] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 12

[72] J. Weston, A. Bordes, S. Chopra, and T. Mikolov, "Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. 12

[73] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016. 12, 96

[74] Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei, "Visual7W: Grounded Question Answering in Images," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 15, 18

[75] L. Yu, E. Park, A. C. Berg, and T. L. Berg, "Visual Madlibs: Fill in the blank Image Generation and Question Answering," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015. 18

[76] A. Ray, G. Christie, M. Bansal, D. Batra, and D. Parikh, "Question relevance in VQA: identifying non-visual and false-premise questions," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016. 19

[77] A. Mahendru, V. Prabhu, A. Mohapatra, D. Batra, and S. Lee, "The promise of premise: Harnessing question premises in visual question answering," *Conference on Empirical Methods in Natural Language Processing*, 2017. 19

[78] C. Danescu-Niculescu-Mizil and L. Lee, "Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs.," in *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*, 2011. 21

[79] Q. V. L. Ilya Sutskever, Oriol Vinyals, "Sequence to Sequence Learning with Neural Networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. 22

[80] Amazon, "Alexa." http://alexa.amazon.com/. 24

[81] A. Jabri, A. Joulin, and L. van der Maaten, "Revisiting visual question answering baselines," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 25

[82] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 25, 26, 39, 43

[83] Z. Yang, X. He, J. Gao, L. Deng, and A. J. Smola, "Stacked Attention Networks for Image Question Answering," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 26, 27

[84] J. Lu, X. Lin, D. Batra, and D. Parikh, "Deeper LSTM and Normalized CNN Visual Question Answering model." https://github.com/VT-vision-lab/VQA_LSTM_CNN, 2015. 27

[85] Microsoft, "Bing Spell Check API." https://www.microsoft.com/cognitive-services/en-us/bing-spell-check-api/documentation. 27

[86] "NLTK." http://www.nltk.org/. 27

[87] "Torch." http://torch.ch/. 27

[88] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 28, 45

[89] A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. M. F. Moura, D. Parikh, and D. Batra, "Visual Dialog," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 32, 34, 35, 36, 39, 43, 44, 45, 47, 48, 96, 110

[90] S. Kazemzadeh, V. Ordonez, M. Matten, and T. L. Berg, "ReferItGame: Referring to Objects in Photographs of Natural Scenes," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014. 35

[91] 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, pp. 484–489, 2016. 35

[92] D. Lewis, *Convention: A philosophical study*. John Wiley & Sons, 2008. 35

[93] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. 36

[94] J. Li, W. Monroe, T. Shi, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," *arXiv preprint arXiv:1701.06547*, 2017. 36

[95] S. Nolfi and M. Mirolli, *Evolution of Communication and Language in Embodied Agents*. Springer Publishing Company, Incorporated, 1st ed., 2009. 36

[96] J. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 36, 81, 83, 84

[97] A. Lazaridou, A. Peysakhovich, and M. Baroni, "Multi-agent cooperation and the emergence of (natural) language," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 36, 84

[98] S. Havrylov and I. Titov, "Emergence of language with multi-agent games: Learning to communicate with sequences of symbols," in *The International Conference on Learning Representations (ICLR) Workshop*, 2017. 36

[99] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," *arXiv preprint arXiv:1703.04908*, 2017. 36, 84

[100] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992. 40, 62

[101] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1st ed., 1998. 42

[102] L. Smith and M. Gasser, "The development of embodied cognition: six lessons from babies.," *Artificial life*, vol. 11, no. 1-2, 2005. 50

[103] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the International Conference on Machine Learning (ICML)*, 1999. 52, 62, 75

[104] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research (JMLR)*, vol. 17, pp. 1334–1373, Jan. 2016. 52

[105] D. Misra, J. Langford, and Y. Artzi, "Mapping instructions and visual observations to actions with reinforcement learning," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017. 52, 65

[106] A. Graves, "Adaptive computation time for recurrent neural networks," *arXiv preprint arxiv:1603.08983*, 2016. 52, 54, 59

[107] Y. Wu, Y. Wu, G. Gkioxari, and Y. Tian, "Building Generalizable Agents With a Realistic And Rich 3D Environment," *arXiv preprint arXiv:1801.02209*, 2018. 53, 54, 55, 68, 72, 73, 82, 83, 92, 110, 116, 119

[108] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese, "Joint 2D-3D-Semantic Data for Indoor Scene Understanding," *ArXiv e-prints*, 2017. 54

[109] C. Tessler, S. Givony, T. Zahavy, D. J. Mankowitz, and S. Mannor, "A deep hierarchical approach to lifelong learning in minecraft.," in *AAAI Conference on Artificial Intelligence*, 2017. 54, 60, 68

[110] "Planner5d." https://planner5d.com/. 55

[111] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick, "CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 56, 100

[112] H. Yu, H. Zhang, and W. Xu, "A deep compositional framework for human-like language acquisition in virtual environment," *arXiv preprint arXiv:1703.09831*, 2017. 59

[113] D. Misra, J. Langford, and Y. Artzi, "Mapping instructions and visual observations to actions with reinforcement learning," in *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*, 2017. 59, 69, 99

[114] B. Hayes-Roth and F. Hayes-Roth, "A cognitive model of planning," *Cognitive Science*, 1979. 66

[115] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied Question Answering," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 66, 68, 69, 71, 72, 74, 76, 77, 78, 79, 96, 99, 100, 102, 110

[116] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2004. 67, 68

[117] S. Ross and J. A. Bagnell, "Reinforcement and imitation learning via interactive no-regret learning," *arXiv preprint arXiv:1406.5979*, 2014. 67

[118] H. M. Le, N. Jiang, A. Agarwal, M. Dudík, Y. Yue, and H. Daumé III, "Hierarchical imitation and reinforcement learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018. 68

[119] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning," *Artificial Intelligence*, 1999. 68

[120] R. S. Sutton, D. Precup, and S. P. Singh, "Intra-option learning about temporally abstract actions," in *Proceedings of the International Conference on Machine Learning (ICML)*, 1998. 68

[121] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *AAAI Conference on Artificial Intelligence*, 2017. 68

[122] T. D. Kulkarni, K. R. Narasimhan, A. Saeedi, and J. B. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 68

[123] B. Bakker and J. Schmidhuber, "Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization," in *Proceedings of the Conference on Intelligent Autonomous Systems (IAS)*, 2004. 68

[124] S. Goel and M. Huber, "Subgoal discovery for hierarchical reinforcement learning using learned policies," in *AAAI Conference on Artificial Intelligence*, 2003. 68

[125] A. McGovern and A. G. Barto, "Automatic discovery of subgoals in reinforcement learning using diverse density," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2001. 68

[126] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2016. 68, 75

[127] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, "Learning to Compose Neural Networks for Question Answering," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies (NAACL HLT)*, 2016. 68

[128] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, "Neural Module Networks," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 68, 82, 88

[129] F. Hill, K. M. Hermann, P. Blunsom, and S. Clark, "Understanding grounded language learning agents," *arXiv preprint arXiv:1710.09867*, 2017. 69

[130] T. Shu, C. Xiong, and R. Socher, "Hierarchical and interpretable skill acquisition in multi-task reinforcement learning," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. 69, 99

[131] A. Vogel and D. Jurafsky, "Learning to follow navigational directions," in *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*, 2010. 69, 99

[132] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox, "Learning to parse natural language commands to a robot control system," in *Proceedings of International Sym-*

posium on Experimental Robotics (ISER), 2013. 69, 99

[133] K. Narasimhan, T. Kulkarni, and R. Barzilay, "Language understanding for text-based games using deep reinforcement learning," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015. 69, 99

[134] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, "IQA: Visual Question Answering in Interactive Environments," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 69, 96, 99, 100, 102, 110

[135] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "Virtualhome: Simulating household activities via programs," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 69, 99

[136] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 69, 99, 110

[137] S. Gupta, D. Fouhey, S. Levine, and J. Malik, "Unifying map and landmark based representations for visual navigation," *arXiv preprint arXiv:1712.08125*, 2017. 69

[138] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "Ai2-thor: An interactive 3d environment for visual ai," *arXiv preprint arXiv:1712.05474*, 2017. 69, 110

[139] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. 75

[140] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 82, 83, 85, 86

[141] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2008. 83

[142] L. Busoniu, R. Babuska, and B. De Schutter, "A Comprehensive Survey of Multiagent Reinforcement Learning," *Trans. Sys. Man Cyber Part C*, 2008. 83

[143] M. Hausknecht and P. Stone, "Deep Recurrent Q-Learning for Partially Observable MDPs," in *AAAI Conference on Artificial Intelligence*, 2015. 83

[144] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *AAAI Conference on Artificial Intelligence*, 2018. 83, 86

[145] P. Peng, Q. Yuan, Y. Wen, Y. Yang, Z. Tang, H. Long, and J. Wang, "Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games," *arXiv preprint arXiv:1703.10069*, 2017. 83

[146] OpenAI, "OpenAI Five." https://blog.openai.com/openai-five/, 2018. 83

[147] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, N. Sonnerat, T. Green, L. Deason, J. Z. Leibo, D. Silver, D. Hassabis, K. Kavukcuoglu, and T. Graepel, "Human-level performance in first-person multiplayer games with population-based deep reinforcement learning," *arXiv preprint arXiv:1807.01281*, 2018. 83

[148] Y. Hoshen, "VAIN: Attentional multi-agent predictive modeling," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 83, 84

[149] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 83

[150] E. Jorge, M. Kågebäck, and E. Gustavsson, "Learning to play guess who? and inventing a grounded language as a consequence," in *NIPS workshop on Deep Reinforcement Learning*, 2016. 84

[151] A. Das, S. Kottur, J. M. Moura, S. Lee, and D. Batra, "Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017. 84

[152] S. Kottur, J. M. Moura, S. Lee, and D. Batra, "Natural language does not emerge 'naturally' in multi-agent dialog," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017. 84

[153] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," *arXiv preprint arXiv:1805.07733*, 2018. 84

[154] F. A. Oliehoek, "Decentralized POMDPs," in *Reinforcement Learning: State of the Art*, Springer Berlin Heidelberg, 2012. 84

[155] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. MIT Press, 1998. 85, 86

[156] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014. 85

[157] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 87

[158] S. P. Stich, "Do animals have beliefs?," *Australasian Journal of Philosophy*, vol. 57, no. 1, pp. 15–28, 1979. 95

[159] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for squad," in *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*, 2018. 96

[160] J. L. Elman, "Finding structure in time," *Cognitive science*, 1990. 97, 98, 99, 108

[161] R. P. Rao and D. H. Ballard, "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects," *Nature neuroscience*, 1999. 97, 99

[162] A. Clark, *Surfing uncertainty*. Oxford: Oxford University Press, 2016. 97, 99

[163] J. Hohwy, *The predictive mind*. Oxford: Oxford University Press, 2013. 97, 99

[164] P. Elias, "Predictive coding – I," *IRE Transactions on Information Theory*, 1955. 97

[165] B. S. Atal and M. R. Schroeder, "Adaptive predictive coding of speech signals," *Bell System Technical Journal*, 1970. 97

[166] J. Schmidhuber, "Curious model-building control systems," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 1991. 97

[167] T. Schaul and M. Ring, "Better generalization with forecasts," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2013. 97

[168] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2015. 97

[169] D. Silver, H. van Hasselt, M. Hessel, T. Schaul, A. Guez, T. Harley, G. Dulac-Arnold, D. Reichert, N. Rabinowitz, A. Barreto, *et al.*, "The predictron: End-to-end learning and planning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2017. 97

[170] G. Wayne, C.-C. Hung, D. Amos, M. Mirza, A. Ahuja, A. Grabska-Barwinska, J. Rae, P. Mirowski, J. Z. Leibo, A. Santoro, *et al.*, "Unsupervised predictive memory in a goal-directed agent," *arXiv preprint arXiv:1803.10760*, 2018. 97, 98

[171] S. Recanatesi, M. Farrell, G. Lajoie, S. Deneve, M. Rigotti, and E. Shea-Brown, "Predictive learning extracts latent space representations from sensory observations," *bioRxiv*, 2019. 97

[172] D. Truncellito, "Epistemology. internet encyclopedia of philosophy," 2007. 98

[173] F. Dretske, "Meaningful perception," *An Invitation to Cognitive Science: Visual Cognition,*, pp. 331–352, 1995. 98

[174] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013. 98

[175] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 98

[176] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019. 98

[177] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015. 98

[178] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 98

[179] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015. 98

[180] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 98

[181] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 98

[182] P. Dayan, "Improving generalization for temporal difference learning: The successor representation," *Neural Computation*, 1993. 98

[183] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," *arXiv preprint arXiv:1611.05397*, 2016. 98

[184] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018. 98

[185] C. Cangea, E. Belilovsky, P. Liò, and A. Courville, "VideoNavQA: Bridging the gap between visual and embodied question answering," *arXiv preprint arXiv:1908.04950*, 2019. 99

[186] A. Das, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Neural Modular Control for Embodied Question Answering," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2018. 99, 110

[187] L. Yu, X. Chen, G. Gkioxari, M. Bansal, T. L. Berg, and D. Batra, "Multi-target embodied question answering," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 99, 110

[188] E. Wijmans, S. Datta, O. Maksymets, A. Das, G. Gkioxari, S. Lee, I. Essa, D. Parikh, and D. Batra, "Embodied Question Answering in Photorealistic Environments with Point Cloud Perception," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 99, 110

[189] A. P. Georgopoulos, A. B. Schwartz, and R. E. Kettner, "Neuronal population coding of movement direction," *Science*, vol. 233, no. 4771, pp. 1416–1419, 1986. 99

[190] W. Bialek, F. Rieke, R. D. R. Van Steveninck, and D. Warland, "Reading a neural code," *Science*, vol. 252, no. 5014, pp. 1854–1857, 1991. 99

[191] E. Salinas and L. Abbott, "Vector reconstruction from firing rates," *Journal of computational neuroscience*, vol. 1, no. 1-2, pp. 89–107, 1994. 99

[192] M. G. Azar, B. Piot, B. A. Pires, J.-B. Grill, F. Altché, and R. Munos, "World discovery models," *arXiv preprint arXiv:1902.07685*, 2019. 99

[193] G. Alain and Y. Bengio, "Understanding intermediate layers using linear classifier probes," *arXiv preprint arXiv:1610.01644*, 2016. 99

[194] A. Conneau, G. Kruszewski, G. Lample, L. Barrault, and M. Baroni, "What you can cram into a single $&# vector: Probing sentence embeddings for linguistic properties," in *Proceedings of ACL*, 2018. 99

[195] I. Tenney, P. Xia, B. Chen, A. Wang, A. Poliak, R. T. McCoy, N. Kim, B. V. Durme, S. Bowman, D. Das, and E. Pavlick, "What do you learn from context? probing for sentence structure in contextualized word representations," in *ICLR*, 2019. 99

[196] A. K. Seth, "The cybernetic bayesian brain: From interoceptive inference to sensorimotor contingencies," in *Open MIND: 35(T)* (T. M. . J. M. Windt, ed.), Frankfurt am Main: MIND Group, 2015. 99

[197] N. Nortmann, S. Rekauzke, S. Onat, P. König, and D. Jancke, "Primary visual cortex represents the difference between past and present," *Cerebral Cortex*, 2013. 99

[198] A. G. Huth, T. Lee, S. Nishimoto, N. Y. Bilenko, A. T. Vu, and J. L. Gallant, "Decoding the semantic content of natural movies from human brain activity," *Frontiers in systems neuroscience*, 2016. 99

[199] D. Williams, "Predictive coding and thought," *Synthese*, pp. 1–27, 2018. 99

[200] A. Roskies and C. Wood, "Catching the prediction wave in brain science," *Analysis*, vol. 77, pp. 848–857, 2017. 99

[201] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, *et al.*, "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018. 101, 129

[202] K. Gregor, F. Besse, D. J. Rezende, I. Danihelka, and D. Wierstra, "Towards conceptual compression," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 102, 130

[203] D. J. Rezende and F. Viola, "Taming VAEs," *arXiv preprint arXiv:1810.00597*, 2018. 102

[204] J. Thomason, D. Gordan, and Y. Bisk, "Shifting the baseline: Single modality performance on visual navigation & QA," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019. 103

[205] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2019. 110

[206] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: real-world perception for embodied agents," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 110

[207] O. Russakovsky, J. Deng, J. Krause, A. Berg, and L. Fei-Fei, "The ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)." http://www.image-net.org/challenges/LSVRC/2012/. 110

[208] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 129