UC Berkeley
UC Berkeley Electronic Theses and Dissertations

Title
Building Generally Intelligent Robots

Permalink
https://escholarship.org/uc/item/0ph1t5c4

ISBN
9798173789396

Author
Walke, Homer

Publication Date
2026-05-10

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

Building Generally Intelligent Robots

by

Homer Walke

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Sergey Levine, Chair

Professor Pieter Abeel

Professor Dorsa Sadigh

Professor Ken Goldberg

Summer 2026

Building Generally Intelligent Robots

Copyright © 2026

by

Homer Walke

Abstract

Building Generally Intelligent Robots

by

Homer Walke

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Sergey Levine, Chair

Large-scale imitation learning is a promising approach to building a generally intelligent robot. In principle, by training on a large and diverse dataset of human demonstrations we should obtain a model that generalizes broadly across tasks and environments, similar to foundation models in language and vision. However, training a foundation model for robotics presents unique challenges. Collecting robot data is expensive, so robot pre-training datasets are small and often do not provide enough coverage for meaningful generalization across the vast amount of complexity in the real world. Additionally, the model architecture must balance competing demands. The architecture needs to be flexible, expressive, and high-capacity to fit heterogenous data collected on robots with widely varying sensor and actuators. At the same time, the model needs to be efficient enough to support real-time control, even when conditioned on a history of high-dimensional image observations. To address these challenges, we first present techniques for collecting robot data with sufficient coverage of tasks and environments as well as a dataset for generalization research. Then, we propose a robot foundation model architecture and demonstrate its broad generalization when trained on a dataset of 800K demonstrations on single-arm manipulators. Next, we extend the architecture to enable joint training on data from multiples types of robots, including bimanual arms, wheeled robots, and quadrupeds, and we find promising signs of positive transfer across embodiments. Finally, we tackle the challenge of adding memory to a robot foundation model. We design a memory system that enables tasks requiring anywhere from a few seconds to many minutes of observation history.

Acknowledgements

Thank you to my advisor, Sergey Levine, for providing incredibly valuable guidance throughout my time at Berkeley. Sergey steered me towards impactful problems and provided deep technical advice. But perhaps more importantly, Sergey's genuine excitement for research is infectious and made my PhD a uniquely fulfilling experience. Thank you to the other members of my committee, Pieter Abbeel, Dorsa Sadigh, and Ken Goldberg, for providing helpful feedback on my qualifying exam and this thesis. Thank you to my undergraduate advisors, Michael Littman and Daniel Ritchie, for their extraordinary mentorship and inspiring me to pursue a PhD.

I'm also grateful for everyone I had the privilege of collaborating with and learning from: Dibya Ghosh, Kevin Black, Karl Pertsch, Oier Mees, Ria Doshi, Kuan Fang, Aviral Kumar, Kyle Stachowicz, Sudeep Dasari, Vivek Myers, Paul Zhou, Pranav Atreya, Mitsuhiko Nakamoto, Patrick Yin, Matt Yan, Jonathan Yang, Abraham Lee, and many others. Thank you to everyone in RAIL for countless research discussions and making the lab such a fun place to do research. I also had the opportunity to work with the amazing team at Physical Intelligence. Thank you to Danny Driess for providing superb mentorship during my internship and the rest of the Pi team for giving me the opportunity to learn how to do research in an industry lab.

Outside of research, I made my best friends during the PhD. Thank you to Kevin Black, Katie Kang, Dibya Ghosh, Paul Zhou, and Colin Li for always being willing to throw a frisbee and making the last five years so special. I was also incredibly lucky to have met the love of my life, Sophie Tomatz, during my PhD. Thank you, Sophie, for always being there for me. Lastly, thank you to my mom, dad, and sisters for their unwavering support.

Contents

1 Introduction **1**
1.1 Contributions . . . 2

2 Collecting Robot Data for Generalization Research **4**
2.1 Motivation . . . 4
2.2 Related Work . . . 5
2.3 BridgeData V2 . . . 7
2.3.1 System setup . . . 7
2.3.2 Data collection . . . 7
2.3.3 Dataset composition . . . 8
2.4 Offline Learning Methods . . . 9
2.4.1 Goal-conditioned methods . . . 9
2.4.2 Language-conditioned methods . . . 10
2.5 Experiments . . . 10
2.5.1 Can we learn a broad range of skills that generalize to new objects and environments? . . . 10
2.5.2 Can another institution use the dataset? . . . 11
2.5.3 How does varying the size of the model and the size and diversity of the dataset affect performance? . . . 12
2.6 Discussion . . . 13

3 Designing a Generalist Robot Policy **15**
3.1 Motivation . . . 15
3.2 Related Work . . . 16
3.3 The Octo Model . . . 17
3.3.1 Architecture . . . 19
3.3.2 Training data . . . 20
3.3.3 Training objective . . . 21
3.3.4 Training Details . . . 21
3.3.5 Model Checkpoints & Code . . . 22
3.4 Experiments . . . 22
3.4.1 Octo Controls Multiple Robots Out-of-the-Box . . . 23
3.4.2 Octo Enables Data-Efficient Learning in New Domains . . . 24
3.4.3 Design Decisions for Generalist Robot Policy Training . . . 25

3.5 Discussion . . . 26

4 Controlling Multiple Robot Embodiments with a Single Policy 28
4.1 Motivation . . . 28
4.2 Related Work . . . 29
4.3 Designing a Cross-Embodied Policy . . . 30
4.3.1 Training data . . . 31
4.3.2 Tokenizing variable observation types and task specifications . . . 31
4.3.3 Predicting variable length actions . . . 32
4.3.4 Training details . . . 33
4.4 Evaluation . . . 33
4.4.1 Can CrossFormer match the performance of training on only a single-robot dataset, as well as the best prior method in each setting? . . . 35
4.4.2 Is aligning the observation and action spaces really needed? . . . 35
4.5 Discussion . . . 36

5 Adding Memory to a Generalist Robot Policy 38
5.1 Motivation . . . 38
5.2 Related Work . . . 39
5.3 Multi-Scale Embodied Memory for VLAs . . . 40
5.3.1 Multi-Scale Embodied Memory (MEM) . . . 40
5.3.2 Language Memory for Long-Term Memory . . . 41
5.3.3 Video Encoder for Dense Short-Term Visual Memory . . . 42
5.3.4 Integrating MEM into the $\pi_{0.6}$ VLA . . . 44
5.4 Experimental Evaluation . . . 45
5.4.1 MEM Solves Tasks Requiring Long-Horizon Memory . . . 45
5.4.2 In-Context Adaptation of Manipulation Strategies . . . 47
5.4.3 Analysis Experiments . . . 49
5.5 Discussion . . . 51

6 Conclusion 53

7 References 55

8 Appendix 75
8.1 Collecting Robot Data for Generalization Research . . . 75
8.1.1 Data Statistics . . . 75
8.1.2 Learning Method Implementation Details . . . 75
8.1.3 Hardware Setup . . . 78
8.1.4 Comparison to the Original Bridge Data . . . 78
8.2 Designing a Generalist Robot Policy . . . 79
8.2.1 Contributions . . . 79
8.2.2 Octo Code Example . . . 79
8.2.3 Data Mixture . . . 80
8.2.4 Training Hyperparameters . . . 80

8.2.5 Things that Worked and Did Not Work (Yet) . . . 81
8.2.6 Experimental Setups . . . 83
8.3 Controlling Multiple Robot Embodiments with a Single Policy . . . 86
8.3.1 Training Data . . . 86
8.3.2 Training Hyperparameters . . . 86
8.3.3 Evaluation Setups . . . 86
8.3.4 Evaluation Results . . . 89
8.4 Adding Memory to a Generalist Robot Policy . . . 89
8.4.1 Contributions . . . 89
8.4.2 Task Details . . . 90
8.4.3 Video encoder with Space-Time Separable Attention . . . 93

Chapter 1

Introduction

Modern robots are capable of automating a vast amount of physical labor, from folding laundry in a home, to picking strawberries on a farm, to sorting packages on an assembly line. However, robots are still not widely deployed in every setting where they would be useful. The key challenge is not building more capable robot hardware, but rather developing general-purpose robot software or a "robot brain." Given the scale and diversity of tasks we want robots to perform, manually programming the robot for each new application would be impossible. Ideally, the robot should learn in a manner similar to humans by watching demonstrations of a task, internalizing the strategy, and generalizing the behaviors to similar tasks. This approach, known as imitation learning, offers a promising path towards a general-purpose robot. More concretely, if we fit a high-capacity neural network to a large and diverse dataset of human demonstrations (e.g. collected via teleoperation), we can obtain a model that not only performs the tasks in the dataset, but also generalizes to tasks in the distribution of the dataset. Imitation learning is appealing because, in principle, we can simply scale the size of the training dataset such that all the tasks we want to perform are in-distribution, thereby obtaining a generally intelligent robot. Recent trends in natural language processing and computer vision support this scaling hypothesis. The now standard approach to solving a language or vision problem is to apply a foundation model, or a very high-capacity model trained on an internet-scale dataset. Foundation models outperform hand-engineered approaches and specialist models by generalizing knowledge gained from the abundance of tasks seen in their training data. Critically, these generalization abilities scale with model size and the amount of training tokens. In this thesis, we examine the challenges involved in training such a foundation model for robotics, with the ultimate goal of building a generally intelligent robot.

Unfortunately, scaling the size of the training data and model is not as straightforward in robotics. Unlike in language and vision, robot data cannot be easily scraped from the internet, so we must generate it. However, collecting robot data via teleoperation is time-consuming and expensive, so existing robot pretraining datasets do not have nearly enough coverage of all the settings in which we would like to deploy robots. This issue is exacerbated by the fact that robot data is extremely heterogeneous since different robots have widely varying sensors, actuators, and control frequencies. The result is that robot datasets often have "data islands" or pockets of very similar data with no coverage in between, making meaningful generalization difficult. We also cannot directly adopt model architectures from language and vision research since robotics has a variety of unique constraints. We need a very flexible architecture that can support different sets of inputs and outputs to accommodate heterogenous data. The model also needs to

be expressive enough to model complex action distributions. But perhaps most problematic, the model needs to be high-capacity to fit a large amount of data but still efficient enough to operate in real-time, even when conditioning on high-resolution images from multiple cameras and potentially even a short history of past image observations.

In this thesis we argue that despite these challenges, scaling up robot datasets and models is the most promising path towards generally intelligent robots, and we take steps towards solving both the data and modeling problems. In Chapter 2 we present techniques for collecting robot data with enough coverage for meaningful generalization, and we contribute a dataset to accelerate generalization research. Then, in Chapter 3, we combine this dataset with robot data from labs around the world to form a massive pre-training dataset of over 800K demonstrations on robot arms. We propose a generalist robot policy architecture, train it on this dataset, and find that the model has impressive generalization abilities. In Chapter 4 we push the size and diversity of the pre-training dataset even further by jointly training on multiple types of robot embodiments, including bimanual robot arms, wheeled robots, and quadrupeds. We demonstrate promising signs of transfer across these embodiments. Finally, in Chapter 5, we significantly enhance our ability to fit this large and diverse data by adding memory to the model. We present a system capable of performing tasks requiring anywhere from a few second to many minutes of observation history. We conclude in Chapter 6 by summarizing the key results, identifying important open problems, and proposing directions for further research.

1.1 Contributions

The research in this thesis was conducted in collaboration with many others. Below are the publications supporting each chapter as well as my contributions.

Chapter 2: Homer Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. BridgeData V2: A dataset for robot learning at scale. In *Conference on Robot Learning*, 2023.

Contributions: I led the collection of the dataset (designing tasks and protocols, managing student data collectors), trained many of the models, designed the evaluations, conducted the evaluations, and wrote the paper.

Chapter 3: Dibya Ghosh*, Homer Walke*, Karl Pertsch*, Kevin Black*, Oier Mees*, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*, 2024.

Contributions: I led model evaluations, designed the main WidowX evaluation benchmark, contributed to the initial model design and implementation, and ran many of the evaluations.

Chapter 4: Ria Doshi*, Homer Walke*, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. In *Conference*

on Robot Learning, 2024.

Contributions: I contributed to the design, implementation, and training of the model and wrote the paper.

Chapter 5: Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z. Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, Karan Dhabalia, Michael Equi, Quan Vuong, Jost Tobias Springenberg, Sergey Levine, Chelsea Finn, and Danny Driess. MEM: Multi-Scale Embodied Memory for Vision Language Action Models. In *arXiv preprint arXiv:2603.03596*, 2026.

Contributions: I contributed to the design and implementation of the video encoder and short-horizon memory system. I iterated on early versions of the long-horizon memory system.

*Lead authors

Chapter 2

Collecting Robot Data for Generalization Research

2.1 Motivation

A useful robotic system needs skills that generalize across the wide variety of conditions found in the real world. Recent results in computer vision and natural language processing have demonstrated that we can obtain broad generalization by training high-capacity models on large and diverse datasets [23, 166]. How can we apply this same recipe in robotics? The general strategy seems straightforward. First, collect a large dataset of demonstrations of robot behaviors. Then, train an expressive policy to extract the desirable behaviors with behavioral cloning or offline reinforcement learning (RL). Given enough data in enough settings, we should learn a policy that can generalize across many tasks and environments.

However, in practice, assembling a dataset with the right features to accelerate research in large-scale robot learning presents a significant challenge. Since collecting a large dataset of robot behaviors is time-consuming, the dataset should be reusable outside of the institution where it was collected. Ideally, any researcher could train on the dataset and obtain a policy with reasonable performance on their tasks. To enable this, the dataset needs extensive coverage of tasks and environments so that policies learned on the data will generalize to new lab settings. Many existing robot datasets contain only one or a few environments and tasks [183, 131, 84], meaning a researcher would need to exactly replicate a scene from the data to use them for robot learning. Additionally, the dataset should support flexible task conditioning, through goal images or natural language instructions, so that researchers can easily command policies trained on the data to perform new tasks. Importantly, the dataset should contain data for many feasible tasks in a given environment so that a multi-task policy must learn to pay attention to the task specification rather than inferring the task from the initial observation.

We propose a new dataset, which we call BridgeData V2 because it greatly expands on the previously released Bridge Dataset [54]. BridgeData V2 contains 50,365 demonstrations of 13 skills across 24 environments, more than 7 times as many demonstrations as the original Bridge Dataset. BridgeData V2 is not just a quantitative improvement: the new dataset is specifically designed as a useful resource for a wide range of robot learning research. The greater quantity and diversity of data enables better generalization to new lab settings and, unlike the first Bridge Dataset, the demonstrations were collected in scenes

with many possible skills to support open-vocabulary task specification through either goal images or language instructions. We also augment the demonstrations with 9,731 trajectories collected from a heavily randomized pick-and-place policy to boost the robustness of the foundational objection repositioning skill.

These features make BridgeData V2 well suited for use with a variety of different robot learning methods, many of which make different assumptions. While a number of datasets have been proposed in the past, these have typically only been used with one type of method (e.g., text-conditional behavioral cloning) [84, 20]. To illustrate the versatility and broad applicability of BridgeData V2, we include a comprehensive evaluation of 6 different state-of-the-art robot learning methods, which include algorithms for text-conditioned imitation learning [20], goal-conditioned imitation learning, and goal-conditioned RL [88, 175]. These methods cover a range of key design decisions involving the policy architecture, the use of observation histories, action discretization, and action prediction horizon. We also analyze how the performance of some of these methods changes with dataset size and diversity, showing that large and diverse datasets like BridgeData V2 are an important ingredient in enabling good performance.

Our contributions are a new dataset of robotic manipulation behaviors as well as the empirical study of state-of-the-art offline learning methods using the introduced dataset. We find that BridgeData V2 is useful as an offline dataset for a wide variety of learning algorithms and across multiple labs. Additionally, in our scaling analysis, we find that performance improves with both model size and dataset size and diversity. We believe BridgeData V2 is both a useful resource for robot learning research and a demonstration of the promise of data-driven robot learning.

2.2 Related Work

Prior work on robot learning has often leveraged small datasets that cover a single task in a single domain to develop better learning methods [112, 72, 53, 228, 186, 133, 109, 230, 36]. Several prior projects have assembled large robot datasets for a single behavior like grasping [158, 113, 90, 65, 42], poking [1], pushing [160, 61], or rope manipulation [142]. Others include multiple tasks [183, 131, 223, 25, 91, 28, 84, 11, 19]. However, even the multi-task datasets are difficult for other researchers to use, because they are collected in a single environment, and therefore do not facilitate cross-environment generalization. In order for another researcher to use such a dataset, they would need to make their own environment match perfectly. We instead aim to provide a dataset that covers many tasks and many domains, so that other researchers can use the data to experiment with robot learning on large datasets in their own laboratories. Some prior work has proposed large, diverse, and multi-task datasets, but uses proprietary robots [20, 84]. We aim to facilitate academic research by using a low-cost publicly available robot that any lab can acquire. Other projects have tried using low-cost, handheld grasping devices to collect large datasets [190, 221], but this technique requires inferring actions from video and then transferring behaviors to a robot. There are also several large datasets for robotic navigation [95, 206, 71], some covering multiple widely available mobile robot platforms [179]. While these datasets approach a similar research topic, our aim is specifically to provide a dataset for large-scale robotic manipulation. Assembling a large real-world dataset is time-consuming and expensive, so there has also been significant work on developing simulated environments and datasets for robotic manipulation [133, 224, 141, 83] and navigation [172, 102]. However, it is difficult to replicate the complexity of the real world (e.g., objects, environments, lighting, and physics) in a simulator well enough to thoroughly evaluate the robustness and generalization abilities of robot

Dataset	# Traj.	# Skills [1]	# Env.	Lang.	Public Data	Public Robot	Collection
MIME [183]	8.30k	12	1	✗	✓	✓	human
RoboTurk [131]	2.10k	2	1	✗	✓	✓	human
RoboNet [42]	162k	n/a	10	✗	✓	✓	scripted
MT-Opt [91]	800k	1	1	✗	✗	✓	scripted & learned
BridgeData [54]	7.20k	4	12	✓	✓	✓	human
BC-Z [84]	26.0k	3	1	✓	✓	✗	human
RT-1 [20]	130k	2	3	✓	✗	✗	human
RH20T [57]	13.0k	41	50	✓	✓	✓	human
RoboSet [11]	98.5k	6	11	✓	✓	✓	29% human, 71% scripted
BridgeData V2	60.1k	13	24	✓	✓	✓	84% human, 16% scripted

Table 2.1: BridgeData V2 is a large and diverse publicly available robotic manipulation dataset suitable for a wide variety of learning methods. See Section 2.3.3 for definitions of "skill" and "environment."

learning methods. A real-world dataset provides a research testbed that is truer to downstream robotics applications.

Most closely related to this project is the original Bridge Dataset [54, 105]. While our dataset is quantitatively larger, our work also provides a number of qualitative differences (see Table 2.1). We include both demonstration data and autonomously collected data, as well as provide language descriptions. The aim is to make the dataset useful not only for multi-task imitation learning, but for a variety of learning methods that might have different assumptions: for example, reinforcement learning methods that require data with broader coverage and may benefit from noisy suboptimal trajectories [103], methods that require language descriptions to run language-conditioned imitation learning [84], or methods that use high-capacity models and require very large datasets before they can perform at their full potential [20]. To validate this, we specifically select a wide variety of methods in our evaluation to show that our work provides a single dataset that is compatible with many approaches [20, 68, 56, 230]. This is highly nontrivial, as virtually all prior robotic manipulation datasets have been evaluated with only one or a few methods [42, 84, 20]. Additionally, unlike many prior datasets [42, 84], our experiments isolate the effect of data diversity and show that greater diversity improves generalization, corroborating the result from Brohan et al. [20]. Our experiments extend Brohan et al. [20] by showing that skill diversity improves generalization, not just object diversity. Released earlier this year and collected concurrently with BridgeData V2, RH20T [57] proposes a dataset with several robots and 13k demonstrations. While RH20T is large and diverse, we provide a comprehensive analysis of policies trained on BridgeData V2, and show that these policies can generalize across environments and institutions. Training on a combination of the largest datasets released so far is an exciting and promising direction for future work.

[1] We count pick-and-place as one skill since it is typically performed in one motion. We also count motions that are the reverse of each other (e.g. opening and closing a drawer) as a single skill. In general, the number of skills reported in this table may not match the number reported in the papers associated with each dataset.

2.3 BridgeData V2

Our goal is to design a dataset that facilitates research in large-scale robot learning. The dataset should support generalization to novel tasks, environments, and even institutions. The dataset should also support flexible task conditioning through either goal images or natural language instructions. In this section, we discuss how we collected the data as well as its composition.

2.3.1 System setup

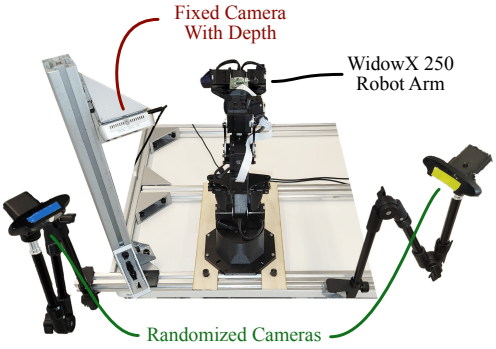


Figure 2.1: **(System setup)** A picture of our robot setup showing the WidowX 250 robot arm and various cameras. For sensing, we use an RGBD camera that is fixed in an over-the-shoulder view, two RGB cameras with poses that are randomized during data collection, and an RGB camera attached to the robot's wrist. The images are saved at a 640x480 resolution and the control frequency is 5 Hz. We collect demonstrations by teleoperating the robot with a VR controller. See Appendix 8.1.3 for additional details.

The robot setup (Figure 2.1) costs approximately $4,000 in total and consists of parts that are all publicly available with a turnaround time of less than two weeks.

2.3.2 Data collection

We designed our data collection protocol to support learning multi-task, generalizable skills. To support broad generalization, we collected data for a wide range of tasks in many environments with suitable variations in objects, camera pose, and workspace positioning. To support the evaluation of multi-task learning methods, we collected data for many possible tasks simultaneously in each environment. This ensures that a policy must pay attention to the task specification, rather than inferring the task from its observations. Notably, this philosophy differs from the original Bridge Dataset, where data was collected for a specific (smaller) set of predefined tasks.

For example, we might set up a kitchen scene with several food items and utensils as well as a drawer. Then, the data collector collects demonstrations by performing any feasible task, such as opening the drawer or putting a utensil in the sink. To speed up data collection, we do not require the positions of the objects in the environment or the robot to be reset between trajectories. We also do not require the data collector to label trajectories with task names. Every 50 trajectories, the collector randomizes the poses of the cameras, switches out the objects in the scene, and randomizes the position of the workspace relative to the robot.

Repositioning objects is a widely applicable skill and thus we wanted to learn policies that can pick-and-place a large number of objects in many environments. During the data collection process, we realized we could partially automate the collection of pick-and-place data with a heavily randomized scripted policy. While this policy fails frequently, we can run it autonomously to collect a large amount of pick-and-place data for a wide range of objects more quickly than teleoperating the robot. Methods that benefit from

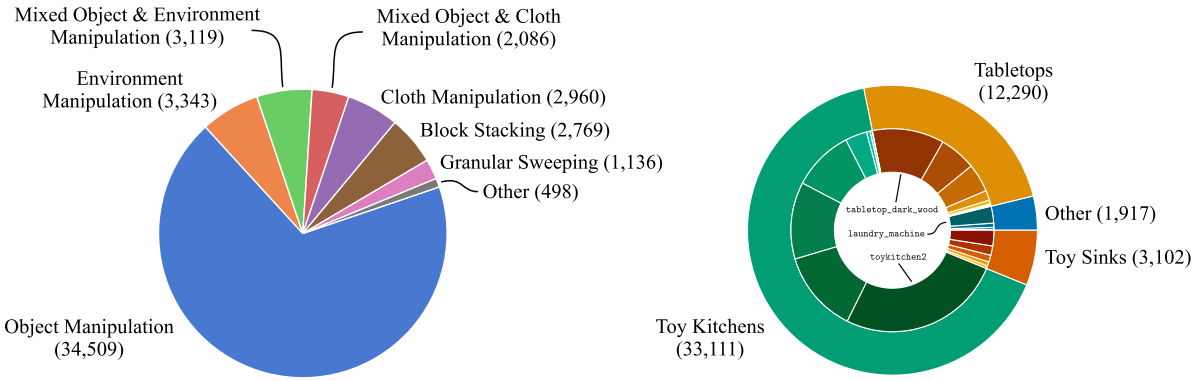


Figure 2.2: **(Dataset composition) (L)** The various types of tasks in BridgeData V2. The majority of the data comes from foundational object manipulation tasks, such as pick-and-place, pushing, and sweeping. Additional data comes from environment manipulation, which includes opening and closing doors and drawers. The remaining data comes from more complex tasks, such as stacking blocks, folding cloths, and sweeping granular media. Some segments of the data contain mixtures of these categories. **(R)** The 24 environments in BridgeData V2, grouped into 4 categories. The majority of the data comes from 7 distinct toy kitchens, which include some combination of sinks, stoves, and microwaves. The remaining environments come from diverse sources, including various tabletops, standalone toy sinks, a toy laundry machine, and more.

suboptimal data, such as offline RL, can leverage this autonomous data to learn more robust behaviors. Note that users are free to exclude the autonomous data from training, but our dataset offers the flexibility to explore both options.

Since we do not annotate trajectories with task names during data collection, we used a crowdsourcing platform to label the data post-hoc. Annotators were asked to describe the task being performed by the robot in each trajectory, with particular emphasis on the final location of any moved objects.

2.3.3 Dataset composition

To effectively describe the composition of our dataset, we will first define "skill" and "task," two terms that have had many different meanings in prior work. We use "skill" to denote groups of trajectories that exhibit similar motions — such as pick-and-place, sweeping, folding, or door opening — but potentially with different objects or arrangements of objects. We use "task" to denote groups of trajectories that correspond to a similar language instruction, which in practice typically means that similar motions (e.g., pick-and-place) with different objects (e.g., a fork or a bowl) correspond to the same skill, but to different tasks. This usage of the terms resembles some prior work on language-conditioned robot learning [20, 84].

BridgeData V2 includes 13 skills that range in complexity. A large portion of the data consists of the foundational skills of pick-and-place, pushing, and reorienting objects, since these skills are applicable in a wide variety of settings and mastery of these skills may transfer to more complicated behaviors. The more complex skills include opening and closing doors and drawers, wiping surfaces, folding cloths, stacking blocks, twisting knobs, flipping switches, turning faucets, zipping zippers, and using tools to sweep

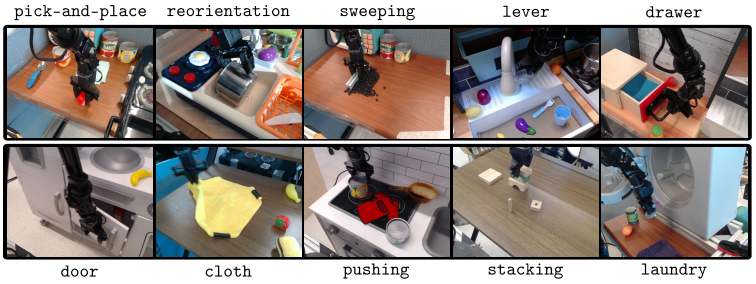


Figure 2.3: **(Skills + Environments)** Examples of skills and environments in the dataset. BridgeData V2 includes a wide variety of skills, environments, and objects to support broad generalization.

granular media. To ensure that we can learn generalizable skills, the data includes examples of each these skills applied to a wide variety of objects and in a wide variety of environments. BridgeData V2 features 24 environments, including kitchens, sinks, and tabletops, as well as more than 100 objects. In total, BridgeData V2 contains 50,365 expert demonstrations and 9,731 trajectories from a scripted policy. Figure 2.2 shows a breakdown of the environments and skills in the demonstration data and Figure 2.3 shows some examples. We provide additional statistics in Appendix 8.1.1.

2.4 Offline Learning Methods

To demonstrate that BridgeData V2 is compatible with a variety of learning methods with different assumptions, we evaluated several state-of-the-art offline learning methods using the data. We selected both goal-conditioned methods (in the form of goal images) and language-conditioned methods because our dataset is designed for open-vocabulary task specification. The observation space for all of the methods is 128x128 RGB images, except for RT-1 which uses 320x256 RGB images. We use only the over-the-shoulder camera view. The 7D action space of the robot consists of continuous 6D Cartesian end-effector motion, corresponding to relative changes in pose, as well as a discrete dimension to control the opening and closing of the gripper. We provide the full implementation details and training data statistics in Appendix 8.1.2.

2.4.1 Goal-conditioned methods

Behavioral cloning with goals (GCBC). As a baseline, we evaluate a standard goal-conditioned behavior cloning method. We use a ResNet-34 to encode the observation and goal. The image encoding is then passed through several fully connected layers to produce a robot action.

Diffusion behavioral cloning (D-GCBC). Prior work has shown that highly expressive policies can better model multi-modal action distributions, leading to improved performance on a variety of tasks [177, 36, 68]. We modify our GCBC baseline to represent the policy using a diffusion process and train with the DDPM objective [73].

Action chunking with transformers (ACT). Previous work has also noted that predicting action sequences instead of single actions can benefit imitation learning [230, 36]. ACT instantiates this with transformers and a conditional VAE [189] training objective, allowing the model to generate multi-modal action sequences.

Contrastive RL (CRL). Along with imitation learning methods, we also evaluate a goal-conditioned

reinforcement learning method. CRL casts goal-conditioned RL as a representation learning problem, parameterizing goal-conditioned value functions as log-linear representations analogous to contrastive learning [56, 55, 232].

2.4.2 Language-conditioned methods

Behavioral cloning with language (LCBC). As a baseline, we evaluate a standard language-conditioned behavior cloning method [193]. The natural language instruction is first encoded using the MUSE sentence embedding [218]. Then the image observation is encoded using a ResNet-34 with FiLM conditioning on the language encoding [153]. The final encoding is then passed into a fully connected policy network to produce a robot action.

RT-1 [20] is a large transformer model trained via behavior cloning to predict low-level robot actions. The inputs are first tokenized using a pre-trained EfficientNet [197] (for images) and a pre-trained language model [27] (for the language instruction). These tokens are then passed to a decoder-only transformer to predict discretized robot actions. RT-1 is the only method to take a history of observations as input and uses a higher image resolution than the other methods.

2.5 Experiments

The goal of our experiments is to evaluate the utility of our dataset for testing a variety of multi-task offline learning methods. Rather than attempting to rigorously analyze different design decisions in the learning methods, we aim to determine the extent to which our dataset facilitates large-scale robot learning research. Our experiments are designed to answer the following questions:

1. Can we learn a broad range of tasks with both goal and language-conditioned methods?
2. Can we generalize the skills in the dataset to new objects and environments?
3. Can another institution use the dataset to obtain skills that work in their lab?
4. How does model size, dataset size, and dataset diversity affect the performance of our learned policies?

2.5.1 Can we learn a broad range of skills that generalize to new objects and environments?

We first evaluated the methods in Section 2.4 on tasks that are seen in the training data (see Table 2.2). Even though these tasks are seen in training, the methods must still generalize to novel object positions, distractor objects, and lighting. These tasks test a range of abilities: opening the drawer requires precisely inserting the gripper into the handle, sweeping the beans involves using a tool and predicting the motion of granular media, stacking a block requires delicate balancing, grasping the corn cob requires correctly orienting the wrist joint, and flipping the pot requires coordinating all six degrees of freedom. To obtain success rates for each method, we collected 10 trials for each task, varying the positions of objects and distractors between trials. Most apparently, RT-1 is significantly better than our LCBC baseline,

Task	GCBC	D-GCBC	ACT	CRL	LCBC	RT-1
Open drawer	0.4	0.6	0.5	0.4	0.5	1.0
Sweep beans into pile with bar	0.9	0.9	0.9	0.7	0.4	0.6
Fold thin blue cloth over object	0.4	0.7	0.7	0.5	0.5	0.9
Stack green block on yellow block	0.4	0.2	0.3	0.6	0.0	0.0
Put corn in pot	0.9	0.8	0.8	0.8	0.0	0.0
Put carrot on plate	0.7	0.4	0.1	0.0	0.0	0.8
Flip pot upright	0.1	0.1	0.0	0.4	0.4	0.4
Put eggplant in pot	0.1	0.2	0.0	0.0	0.0	0.2
Average	**0.49**	**0.49**	**0.41**	**0.42**	**0.23**	**0.49**

Table 2.2: **(Evaluation on seen tasks)** Overall, the goal-conditioned methods were comparable. However, RT-1 outperformed LCBC. Success rates are averaged over 10 trials.

likely due to a combination of design decisions such as larger images, action discretization, and observation histories. The goal-conditioned methods are comparable to each other in success rate. We also noticed qualitative differences in the robotic behaviors each method produced. For instance, D-GCBC, ACT, and RT-1 exhibit more pauses, because their expressive policy distributions can accurately model pauses in the demonstration data. However, since D-GCBC lacks observation histories like RT-1 or action chunking like ACT, it sometimes produces jerky behaviors indicating oscillation between modes in the action distribution. These behaviors are best observed in the video on our website.

Next, we evaluated the methods on tasks that require generalizing skills in the data to novel objects and environments (see Table 2.3). For instance, in the rice sweeping task, both the rice and brush are unseen. In the cloth folding task, the cloth is a different color and thicker than the cloths in the training data. In the task of putting a marker into a bowl, the entire environment is unseen, including the objects. The large and diverse training set enabled a considerable degree of generalization, with most methods attaining non-zero success on most tasks. However, the language-conditioned methods particularly struggled on tasks involving unseen objects since these object names are not grounded in the dataset. Once again, RT-1 greatly outperformed the LCBC baseline.

2.5.2 Can another institution use the dataset?

To demonstrate that the dataset is useful for other researchers, we had another institution set up the robot and evaluate the same methods in their lab. The tasks and environments in this evaluation matched tasks seen in the training data; however, there were differences in robot setup, camera placement, lighting conditions, and object instances that led to a significant domain shift. We evaluated 3 tasks in our lab (Lab 1) and at the other institution (Lab 2) with all the methods. As seen in Table 2.4, performance was somewhat worse in Lab 2, but all methods attained non-zero success. RT-1 especially showed only a small degradation in performance. The performance of goal-conditioned methods degraded more, though we note that the "flip the pot upright" task is particularly difficult for a goal-conditioned method since the orientation of the pot is difficult to distinguish. Additionally, the "put eggplant in pot" is a very challenging task in both labs since the eggplant easily slips out of the gripper. Note that these evaluations were performed zero-shot, without any new data collected in Lab 2, and we expect fine-tuning on a small

Task	GCBC	D-GCBC	ACT	CRL	LCBC	RT-1
Sweep rice into pile with brush*	0.6	0.0	0.3	0.3	0.0	0.1
Fold thick gray cloth over object*	0.3	0.6	0.7	0.0	0.0	0.4
Put marker in bowl†	0.6	0.6	0.2	0.7	0.0	0.0
Wipe the table with the cloth‡	0.6	0.5	0.4	0.6	0.4	0.9
Put the mushroom in the pot‡	0.7	0.9	0.1	0.7	0.1	0.6
Put the spoon on the cloth‡	0.8	0.7	0.0	0.8	0.0	1.0
Average	**0.60**	**0.55**	**0.28**	**0.52**	**0.08**	**0.50**

* Unseen objects, seen environment † Unseen objects, unseen environment ‡ Seen objects, unseen environment

Table 2.3: **(Evaluation on unseen tasks)** Either the objects, environment, or both objects and environment were unseen. We found that both language and goal conditioned methods exhibit non-zero success, demonstrating that BridgeData V2 supports broad generalization. Success rates are averaged over 10 trials.

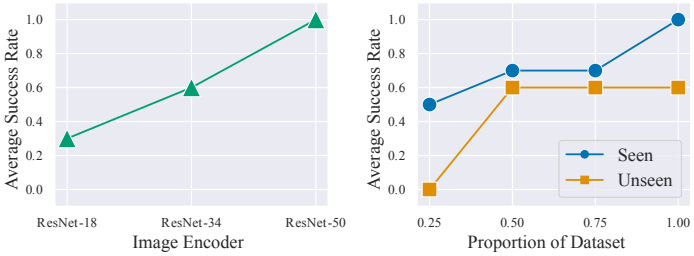
	Lab 1 → Lab 2					
Task	GCBC	D-GCBC	ACT	CRL	LCBC	RT-1
Put carrot on plate	0.7 → 0.3	0.4 → 0.0	0.1 → 0.0	0.0 → 0.3	0.0 → 0.0	0.8 → 0.4
Flip pot upright	0.1 → 0.0	0.1 → 0.2	0.0 → 0.3	0.4 → 0.2	0.4 → 0.1	0.4 → 0.6
Put eggplant in pot	0.1 → 0.1	0.2 → 0.2	0.0 → 0.0	0.0 → 0.1	0.0 → 0.0	0.2 → 0.2
Average	**0.30 → 0.13**	**0.23 → 0.13**	**0.03 → 0.10**	**0.13 → 0.20**	**0.13 → 0.03**	**0.47 → 0.40**

Table 2.4: **(Cross-institution evaluation)** We find that both goal-conditioned and language-conditioned methods achieve non-zero success at a new institution (Lab 2), demonstrating that our dataset is useful for other researchers. Success rates are averaged over 10 trials.

amount of data in a new lab to significantly improve performance, as discussed in prior work [54].

2.5.3 How does varying the size of the model and the size and diversity of the dataset affect performance?

Lastly, we analyzed how the performance of our goal-conditioned BC method varies with model size and dataset size and diversity (see Figure 2.4). First, we tested GCBC with different sizes of image encoders on the task of moving a spoon. We found that higher capacity models perform strictly better at this task. Qualitatively, the smaller models often failed to correctly rotate the gripper to pick up the spoon when it was positioned horizontally. Next, we evaluated GCBC trained on datasets of different size. To create smaller datasets, we randomly subsampled the data stratified by task, preserving diversity while decreasing the number of trajectories. We found that as dataset size increases, performance improves for both a seen and unseen task. To test how skill diversity affects performance, we trained GCBC on two datasets: one dataset with only 3 skills (pick-and-place, pushing, and wiping) and another dataset with all 13 skills. We kept the size of these datasets approximately the same (28k vs 27k trajectories). We found that performance on an unseen pick-and place task was significantly improved by training on data with



Train on 3 skills	Train on 13 skills
0.30	**0.65**

Figure 2.4: **(Scaling analysis) (L)** Performance of goal-conditioned behavior cloning trained on BridgeData V2 improves with higher capacity models and larger training sets, suggesting that our dataset provides an effective foundation for studying scalable policy learning. **(R)** We compared the performance of GCBC trained on a dataset with only 3 skills and a dataset with all 13 skills, keeping the size of the datasets approximately the same. On an unseen task, the policy trained with greater skill diversity performed significantly better, indicating positive transfer between skills. We used 20 trials per policy in this experiment.

greater skill diversity. This result indicates that data from other skills can improve the robustness of the pick-and-place skill. Brohan et al. [20] also isolate data diversity and find that greater diversity improves generalization, however they study diversity in objects. We analyze skill diversity and arrive at a similar conclusion. Taken together, the results in this section indicate that scaling the dataset and model capacity further is a promising path toward greater capabilities.

2.6 Discussion

We presented BridgeData V2, a dataset with 60,096 trajectories of robotic manipulation behaviors designed to enable research on scalable robot learning methods. Our dataset covers a wide range of tasks and environments and supports methods with various assumptions, including goal and language conditioned imitation learning and reinforcement learning. BridgeData V2 was collected with a low-cost and widely available robot arm, making it suitable for accessible research on robot learning. Our evaluation also demonstrates that a wide range of learning algorithms can use BridgeData V2 to produces policies that generalize across tasks, environments, and institutions. We believe this is a strong validation of the utility of our dataset, as many previously proposed datasets have only been evaluated with one or a small number of methods, and typically at a single institution.

Our dataset does have a number of limitations. While we provide a variety of behaviors, including manipulation of deformable objects, tools, and furniture, the tasks are generally low-precision and do not require complex manipulation of forces. This is reasonable for studying generalization but does not cover the challenges that might occur with more forceful manipulation or more dynamic tasks, such as throwing, moving heavy objects, or low-tolerance industrial insertion. Pushing the frontier on more precise, dexterous, and dynamic tasks is an exciting avenue for future data collection efforts. Additionally, our dataset was collected entirely at a single institution, and although we demonstrate that by utilizing a variety of environments our data enables robots to perform tasks in other labs, an even broader coverage of environments would be a desirable property in future datasets. Finally, although we deliberately chose an

accessible and low-cost robot arm, it may be difficult for other researchers to standardize around the same robot. Therefore, a particularly exciting direction for future data collection work would be to assemble multi-robot datasets that enable some degree of generalization across robot morphology. Nonetheless, we hope that our dataset will serve as an important stepping stone toward making scalable robot learning research practical, accessible, and effective.

Chapter 3

Designing a Generalist Robot Policy

3.1 Motivation

The common approach for robotic learning is to train policies on datasets collected for the specific robot and task at hand. Learning from scratch in this way requires significant data collection effort for each task, and the resulting policies usually exhibit only narrow generalization. In principle, collected experience from other robots and tasks offers a possible solution, exposing models to a diverse set of robotic control problems that may improve generalization and performance on downstream tasks. However, even as general-purpose models become ubiquitous in natural language [149, 205] and computer vision [168, 101], it has proven challenging to build the analogous "general-purpose robot model" that can control many robots for many tasks. Training a unified control policy in robotics presents unique challenges, requiring handling different robot embodiments, sensor setups, action spaces, task specifications, environments, and compute budgets.

Towards this direction, several works have proposed robotic foundation models that directly map robot observations to actions and provide zero-shot or few-shot generalization to new domains and robots. We broadly refer to these models as "generalist robot policies" (GRPs), emphasizing their ability to perform low-level visuomotor control across tasks, environments, and robotic systems [167, 19, 50, 239, 20, 181, 2, 209, 74, 217, 106]. For example, the GNM model [180] generalizes across different robotic navigation scenarios, the RoboCat model [19] handles different robot embodiments for goal-conditioned tasks, and the RT-X model [148] performs language-conditioned manipulation across five robot embodiments. Although these models represent significant steps toward a true "general-purpose robot model," they have been limited in multiple important aspects: they typically constrain downstream users to a pre-defined and often restrictive set of input observations, e.g., a single camera stream; they lack support for effective finetuning to new domains; and importantly, the largest of these models are not available to the general public.

We design a system for pretraining generalist robot policies more suitable for the diversity of interfaces in downstream robotic applications. The core of our model is a transformer architecture that maps arbitrary input tokens (created from observations and tasks) to output tokens (then decoded into actions), which can be trained on a diverse dataset of robots and tasks. With no additional training, this policy can accept different camera configurations (e.g., workspace or wrist cameras), can control different robots, and can

be guided via either language commands or goal images — all by simply changing which tokens are fed into the model. Most importantly, the model can be adapted to new robot setups with new sensory inputs, action spaces, or morphologies by adding appropriate adapters and finetuning with a small target domain dataset and an accessible compute budget.

Our primary contribution is Octo, a transformer-based policy pretrained on the largest robot manipulation dataset to date: 800k robot demonstrations from the Open X-Embodiment dataset [148]. Octo is the first GRP that can be effectively finetuned to new observations and action spaces and the first generalist robot manipulation policy that is fully open-source, including the training pipeline, model checkpoints, and data. Finally, while the individual components that comprise Octo — a transformer backbone, support for both language and goal image specification, and a diffusion head to model expressive action distributions — have been discussed in prior work, the particular combination of these components into a powerful generalist robot policy is unique and novel.

We demonstrate through extensive experiments on 9 robots across 4 institutions that our combined system leads to state-of-the-art performance for out-of-the-box multi-robot control for single and dual-arm manipulation tasks and that Octo can be used as an effective initialization for finetuning to unseen setups with new observation and action spaces. In the process, we carefully study the effect of different design decisions when pretraining GRPs; we evaluate how the choice of data distribution, model architecture, and policy formulation affects the quality of the pretrained GRP. Our evaluation highlights the utility of scale and flexibility: our best models are those trained on the widest data mixtures, with the least restrictive inductive biases, and with policy objectives that can fit the diversity of behaviors in the pretraining data.

We release all resources required to train, use, reproduce, and finetune an Octo model. We provide pretrained Octo model checkpoints with 27M and 93M parameters that, out of the box, support multiple RGB camera inputs as well as both language and goal image task specification. We also provide scripts for finetuning these models on new domains, as well as our complete pretraining pipeline, including optimized data loaders, transformer implementations for multimodal inputs, and tools to monitor training progress.

3.2 Related Work

Many works train policies using a large dataset of trajectories collected from a robot, from early efforts using autonomous data collection for scaling policy training [159, 114, 89, 43, 60, 66] to more recent efforts that explore the combination of modern transformer-based policies with large demonstration datasets [20, 87, 230, 63, 185, 194]. These works primarily focus on a single embodiment, while Octo trains policies on robot datasets assembled across *multiple* embodiments, increasing the effective size of the training dataset and allowing finetuning to a range of robot setups.

More recently, papers have focused on broadening the generalization abilities of robot policies. Multiple works leverage diverse non-robot data or pretrained vision-language foundation models to boost policy generalization to new scenes and tasks [194, 239, 225, 35, 79, 22, 187, 76, 6, 77, 13, 5, 107, 33, 50]. More closely related to Octo are recent works that train robot policies across data from multiple robot embodiments: the GNM model [181, 180] generalizes across robot navigation setups while RoboCat [19] and RT-X [148] control multiple single-arm manipulation robots. While these models deliver impressive policy learning

results, a key issue is their lack of flexibility: they typically require users to stick to the sensory inputs and action space used during pretraining and do not support adaptation to new observation and action spaces. Furthermore, the largest models are not publicly accessible. Octo differs from these works in multiple aspects: it is trained on a larger and more diverse robot data mix, it supports a wider range of downstream applications via efficient finetuning to new robot setups, and it is fully open source and reproducible.

Octo's design is inspired by several recent advances in robot imitation learning and scalable transformer training, including the use of denoising diffusion objectives [73] for action decoding [37, 67, 191], the prediction of "action chunks", i.e., sequences of future actions [230, 37, 63], and model layouts and learning rate schedules inspired by the literature on scalable vision transformer training [49, 227]. Our work is the first to leverage these approaches in the context of learning cross-embodied generalist policies and we find that they can lead to substantial performance improvements. In our evaluation, we present ablations to assess the importance of these components, alongside a more comprehensive list of what we found to be (un)important in Appendix 8.2.5; we hope our findings are useful for future research on generalist policy learning.

A key ingredient for training generalist robot policies is robot training data. In contrast to vision and language data that can be scraped from the web, obtaining robot data at scale is challenging and often involves significant investments in hardware and human labor. There are multiple large robot navigation and autonomous driving datasets [64, 222, 26, 195, 180, 96, 206]. In recent years, there have also been multiple efforts for building robot *manipulation* datasets of increasing scale and diversity, either collected via scripted and autonomous policies [43, 89, 92, 24, 159, 66] or human teleoperation [131, 134, 54, 207, 85, 20, 57, 10, 169, 139, 178]. Octo is trained on the Open X-Embodiment dataset [148], a recent effort that pooled many of these aforementioned robot datasets. The Open-X dataset contains approximately 1.5M robot episodes, of which we curate 800k for Octo training. We note that the RT-X model [148] used a more restricted subset of 350K episodes, so to the best of our knowledge, Octo is trained on the largest robotics manipulation demonstration dataset to date.

3.3 The Octo Model

In this section, we describe the Octo model, our open-source generalist robot policy that can be adapted to new robots and tasks — including new sensory inputs and action spaces — via finetuning. We discuss the key design decisions, training objectives, training dataset, and infrastructure. The design of the Octo model emphasizes flexibility and scale: it supports a variety of commonly used robots, sensor configurations, and actions while providing a generic and scalable recipe that can be trained on large amounts of data. It also supports natural language instructions, goal images, observation histories, and multi-modal, chunked action prediction via diffusion decoding [37]. Furthermore, we designed Octo specifically to enable efficient finetuning to new robot setups, including robots with different action spaces and different combinations of cameras and proprioceptive information. This design was selected to make Octo a flexible and broadly applicable generalist robot policy that can be utilized for a variety of downstream robotics applications and research projects.

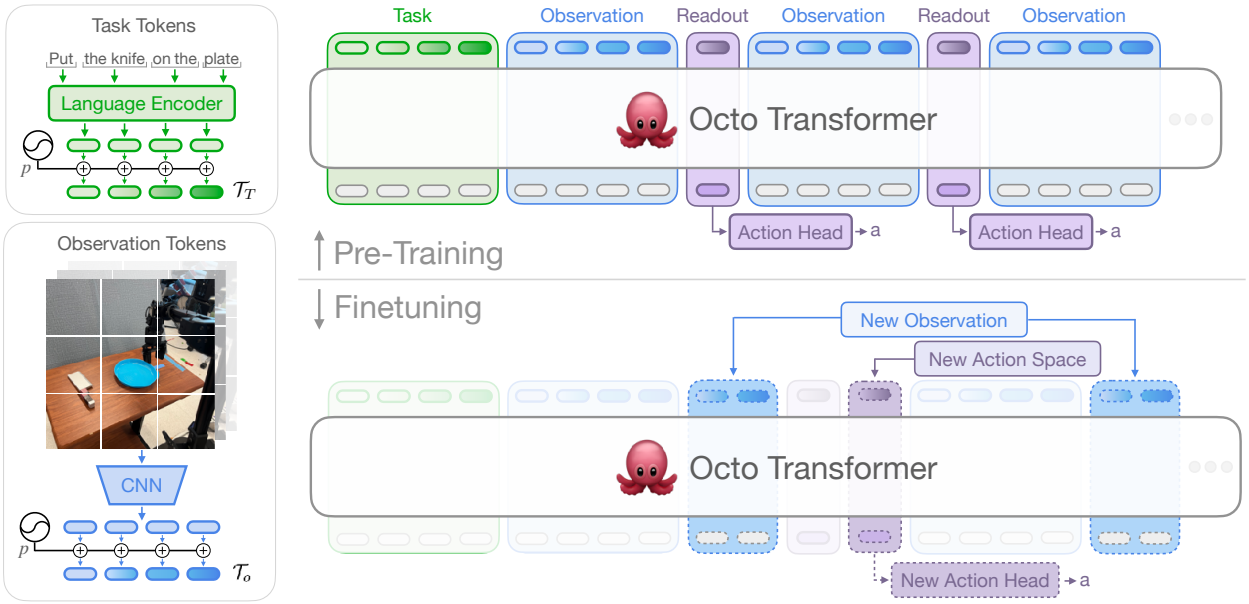


Figure 3.1: **Model architecture. Left**: Octo tokenizes task descriptions (green) and input observations (blue) using a pretrained language model and a lightweight CNN, respectively. **Top**: The transformer backbone processes the sequence of task and observation tokens and produces readout tokens (purple) that get passed to output heads to produce actions. **Bottom**: The block-wise attention structure of the transformer allows us to add or remove inputs and outputs during finetuning: for example, we can add new observations (blue, dashed) or action spaces (purple, dashed) without modifying any pretrained parameters.

3.3.1 Architecture

At its core, Octo is a transformer-based policy π. It consists of three key parts: **input tokenizers** that transform language instructions ℓ, goals g, and observation sequences $o_1, \ldots, o_H$ into tokens $\left[\mathcal{T}_l, \mathcal{T}_g, \mathcal{T}_o\right]$ (fig. 3.1, left); a **transformer backbone** that processes the tokens and produces embeddings $e_l, e_g, e_o = T(\mathcal{T}_l, \mathcal{T}_g, \mathcal{T}_o)$ (fig. 3.1, top); and **readout heads** $R(e)$ that produce the desired outputs, i.e., actions a.

Task and observation tokenizers We convert task definitions (e.g., language instructions ℓ and goal images g) and observations o (e.g., wrist and third-person camera streams) into a common "tokenized" format using modality-specific tokenizers (see fig. 3.1, left):

- **Language inputs** are tokenized, then passed through a pretrained transformer that produces a sequence of language embedding tokens. We use the `t5-base` (111M) model [163].
- **Image observations and goals** are passed through a shallow convolution stack, then split into a sequence of flattened patches [49].

We assemble the input sequence of the transformer by adding learnable position embeddings p to task and observation tokens and then arranging them sequentially $\left[\mathcal{T}_T, \mathcal{T}_{o,1}, \mathcal{T}_{o,2}, \ldots\right]$.

Transformer backbone and readout heads Once the inputs have been cast to a unified token sequence, they are processed by a transformer (see fig. 3.1, top). This is similar to prior works that train transformer-based policies on sequences of observations and actions [213, 162]. The attention pattern of the Octo transformer is block-wise masked: observation tokens can only attend causally to tokens from the same or earlier time steps $\mathcal{T}_{o,0:t}$ as well as task tokens $\mathcal{T}_T$ (green). Tokens corresponding to non-existing observations are fully masked out (e.g., a dataset without language instructions). This modular design enables us to add and remove observations or tasks during finetuning (see below). In addition to these input token blocks, we insert learned *readout tokens* $\mathcal{T}_{R,t}$ (purple). A readout token at $\mathcal{T}_{R,t}$ attends to observation and task tokens before it in the sequence, but is not attended to by *any* observation or task token — hence, they can only passively read and process internal embeddings without influencing them. Readout tokens act similarly to the `[CLS]` token in BERT, serving as a compact vector embedding of the observation sequence thus far. A lightweight "action head" that implements the diffusion process is applied to the embeddings for the readout tokens. This action head predicts a "chunk" of several consecutive actions, similar to prior work [230, 37].

Our design allows us to flexibly add new task and observation inputs or action output heads to the model during downstream finetuning. When adding new tasks, observations, or loss functions downstream, we can wholly retain the pretrained weights for the transformer, only adding new positional embeddings, a new lightweight encoder, or the parameters of the new head as necessitated by the change in specification (see fig. 3.1, bottom). This is in contrast to prior architectures [20, 181], where adding or removing an image input or changing the task specification would require re-initializing or re-training large components of the pre-trained model.

This flexibility is crucial to make Octo a truly "generalist" model: since we cannot cover all possible robot sensor and action configurations during pretraining, being able to adapt Octo's inputs and outputs during finetuning makes it a versatile tool for the robotics community. Prior model designs that use standard

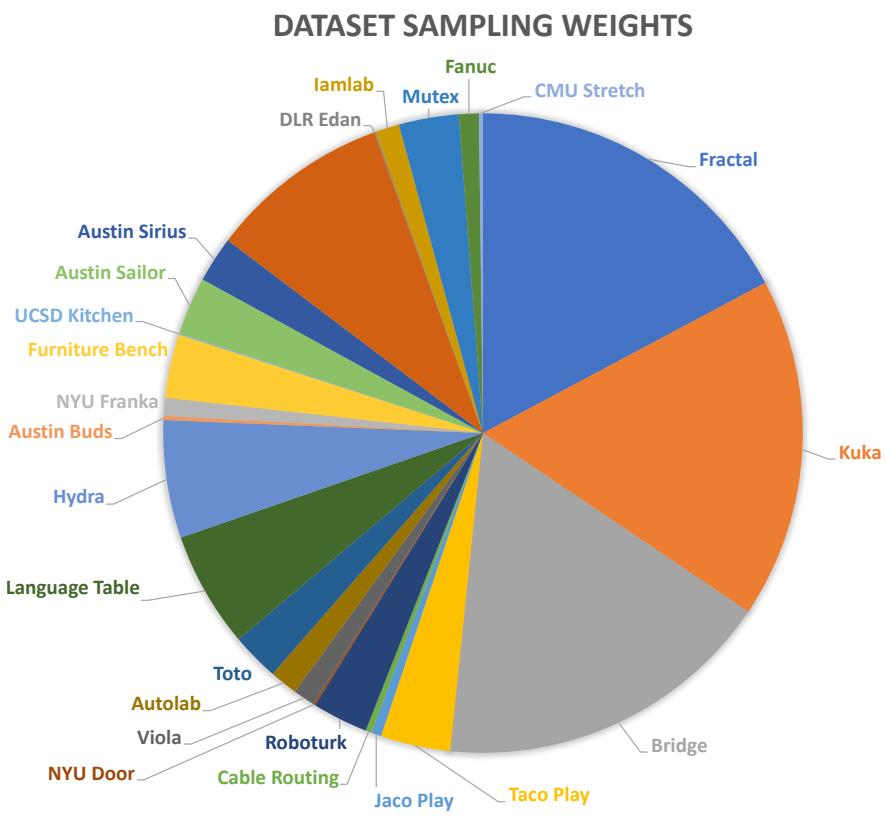


Figure 3.2: **Training dataset composition.** We curate a subset of 25 datasets from the Open X-Embodiment dataset that have image observations, end-effector actions, and show diverse behaviors. The pie chart visualizes the fractions that each dataset contributes to every training batch on average. The dataset weights are determined by the number of samples in each dataset with small modifications to balance dataset size and diversity (see section 3.3.2 for details).

transformer backbones or fuse visual encoders with MLP output heads lock in the type and order of inputs expected by the model. In contrast, switching the observation or task for Octo *does not* require re-initializing most of the model.

3.3.2 Training data

We train Octo on a mixture of 25 datasets from the Open X-Embodiment Dataset [148], a diverse collection of robot learning datasets. Our training mixture includes demonstration data of a variety of tasks from several robot embodiments and scenes. These datasets are heterogeneous not just in terms of the robot type, but also in the sensors (e.g., including or not including wrist cameras) and labels (e.g., including or not including language instructions). See fig. 3.2 and Appendix 8.2.3 for the detailed mixture. To create our training mixture D, we curate the data by first removing all Open-X datasets that contain no image streams, as well as those that do not use delta end-effector control. We also remove datasets that are too repetitive, have a low image resolution, or consist of excessively niche tasks. For the remaining datasets, we roughly categorize them into "more diverse" and "less diverse" datasets based on the tasks and environments, and then double the weight of the more diverse datasets during training. We also down-weight a few datasets with many repetitive episodes to avoid dominating the mixture. Finally, we

zero-pad any missing camera channels and align the gripper action spaces between the datasets such that a gripper command of +1 means "the gripper is open" and 0 means "the gripper is closed." While we found the resulting training mixture to work well, future work should perform a more thorough analysis of data mixture quality for pretraining general robot policies.

3.3.3 Training objective

We use a conditional diffusion decoding head to predict continuous, multi-modal action distributions [73, 37]. Importantly, only one forward pass of the transformer backbone is performed per action prediction, after which the multi-step denoising process is carried out entirely within the small diffusion head. We found this policy parameterization to outperform policies trained with MSE action heads or discretized action distributions [20] in both zero-shot and finetuning evaluations. To generate an action, we sample a Gaussian noise vector $x^K \sim \mathcal{N}\big(0, I\big)$ and apply K steps of denoising with a learned denoising network $\epsilon_\theta(x^k, e, k)$ that is conditioned on the output x^k of the previous denoising step, the step index k, and the output embedding e of the transformer action readout:

$$x^{k-1} = \alpha(x^k - \gamma\epsilon_\theta(x^k, e, k) + \mathcal{N}\big(0, \sigma^2 I\big)). \tag{3.1}$$

The hyperparameters α, γ, and σ correspond to the noise schedule: we use the standard cosine schedule from [145]. We train the diffusion head using the standard DDPM objective first proposed in [73], where we add Gaussian noise to the dataset actions and train the denoising network $\epsilon_\theta(x^k, e, k)$ to reconstruct the original action. For a detailed explanation of diffusion policy training, see Chi et al. [37]. We list all hyperparameters in Appendix 8.2.4.

We use the same diffusion training objective during finetuning and update the full model, a recipe which outperformed those that freeze subsets of the pretrained parameters. In all finetuning experiments, we employ the same recipe: given a small target domain dataset with around 100 trajectories, we finetune for 50k steps using a cosine decay learning rate decay with linear warmup.

3.3.4 Training Details

We trained two variants of our model: Octo-Small with a transformer backbone that mirrors the size of a ViT-S, and Octo-Base with a transformer backbone that mirrors the size of a ViT-B [49].

We use the AdamW optimizer [123] with an inverse square root decay learning rate schedule [227] with weight decay of 0.1 and gradient clipping of 1.0. The ViT-B was trained for 300k steps with a batch size of 2048 using a TPU v4-128 pod, which took 14 hours. A finetuning run of the same model on a single NVIDIA A5000 GPU with 24GB of VRAM takes approximately 5 hours and can be sped up with multi-GPU training.

We train using 2 frames of observation history; in our preliminary experiments, we found significantly diminishing gains beyond the first additional frame. We use hindsight goal relabeling [3], which selects a state uniformly from the future in the trajectory to assign as the goal image, similar to prior work [126, 207, 181, 169, 139]. We apply common image data augmentations during training, and randomly zero out the language instruction or goal image per training example to enable Octo to be conditioned on *either* language instructions *or* goal images. For datasets without language annotations, we always

use goal image conditioning. This enables our model to learn control mostly from self-supervised visual observations and reduces the burden on language annotation, similar to prior work on multi-context imitation learning [126, 138, 137, 139]. For more details on the choice of hyperparameters, see Appendix 8.2.4.

3.3.5 Model Checkpoints & Code

We open-source all resources required to train, finetune and run our model (see `https://octo-models.github.io`):

- **Pretrained Octo checkpoints** for Octo-Small (27M params) and Octo-Base (93M params).
- **Finetuning scripts** for Octo models, in JAX.
- **Model pretraining pipeline** for Octo pretraining on the Open X-Embodiment dataset, in JAX.
- **Standalone data loaders** for Open X-Embodiment data, compatible with JAX and PyTorch.

We provide a simple example for loading and running a pretrained Octo model in Appendix 8.2.2.

3.4 Experiments

Our experiments provide an empirical analysis of Octo, evaluating its ability to serve as a general robotic foundation model across several axes:

1. Can Octo control multiple robot embodiments and solve language and goal tasks out of the box?
2. Do Octo weights serve as a good initialization for data-efficient finetuning to new tasks and robots, and does it improve over training from scratch and commonly used pretrained representations?
3. Which design decisions in Octo matter most for building generalist robot policies?

Evaluation setups We evaluate Octo's capabilities across a representative spectrum of 9 robot learning setups at 4 institutions (see fig. 3.3). We test Octo's ability to control different robots out-of-the-box ("zero-shot") for language and goal image tasks using robot setups that match the pretraining data, where all robots are controlled with delta end-effector control actions and the observation spaces are RGB images. We also evaluate Octo for data-efficient finetuning to new environments and tasks, including with new observations (force-torque inputs in "Berkeley Insertion"), new action spaces (joint position control in "Berkeley Pick-Up") and new robot embodiments ("Berkeley Coke" and "Berkeley Bimanual"). Each of the finetuning setups uses $\sim$ 100 in-domain demonstrations and finetunes in $<$ 5 hours on a NVIDIA A5000 GPU, using the same hyperparameters across all setups (see Appendix 8.2.4). Our evaluation tasks test Octo's ability to interact with diverse objects (e.g., "WidowX BridgeV2"), solve long-horizon tasks (e.g., "Stanford Coffee") and perform precise manipulation (e.g., "Berkeley Insertion"). For more details on each evaluation setup, see Appendix 8.2.6.

Comparisons We compare Octo's ability to control multiple robots out-of-the-box to the best openly available generalist robot policy, **RT-1-X** [148], using the released checkpoint. Similar to Octo, RT-1-X

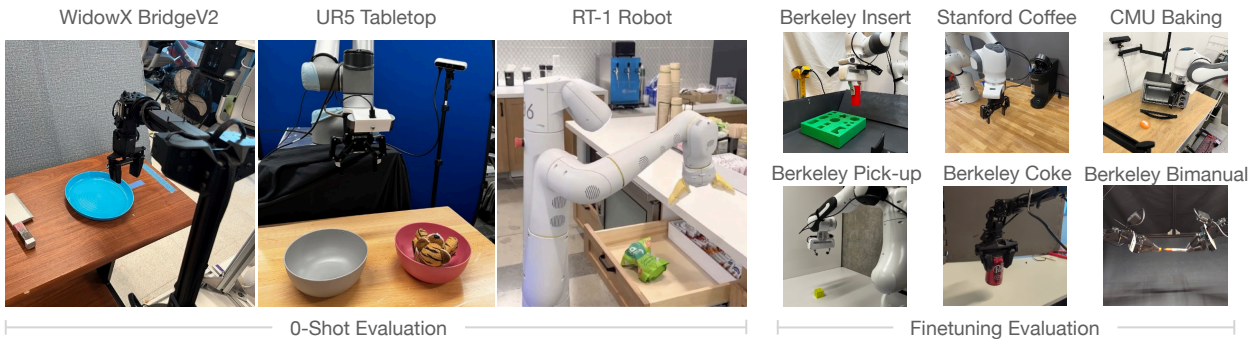


Figure 3.3: **Evaluation Tasks.** We evaluate Octo on 9 real robot setups across 4 institutions. Our evaluations capture diverse object interactions (e.g., "WidowX BridgeV2"), long task horizons (e.g., "Stanford Coffee") and precise manipulation (e.g., "Berkeley Peg Insertion"). We evaluate Octo's capabilities to control robots in environments from the pretraining data out-of-the-box and to efficiently finetune to new tasks and environments with small target domain datasets. We also test finetuning with new observations (force-torque inputs for "Berkeley Peg Insertion"), action spaces (joint position control in "Berkeley Pick-Up" and "Berkeley Bimanual") and new robot embodiments (e.g., "Berkeley Bimanual" and "Berkeley Coke").

is pretrained on the Open X-Embodiment robot dataset and aims to control multiple robots zero-shot, thus providing a natural point of comparison. We also compare the zero-shot capabilities of Octo to **RT-2-X**, a 55 billion parameter vision-language model finetuned on the Open X-Embodiment dataset to produce robot actions. The RT-1-X and RT-2-X models [148] are trained on a more restricted subset of 350K episodes (compared to 800k episodes for Octo). We further compare Octo's performance as a policy initialization for data efficient finetuning to two common approaches: (1) training on the target domain demonstrations *from scratch* and (2) using pretrained visual representations. While a number of prior works have proposed other pretraining schemes for imitation finetuning [54, 52, 57], to our knowledge no prior method provides a pretrained *policy* that has been demonstrated to finetune successfully to *new* observation and action spaces. However, pretrained visual representations such as VC-1 [128] have been used in this way, and therefore we use these methods as another point of comparison.

For finetuning, we found that training our large transformer architecture from scratch overfit quickly on the small datasets. Instead, we obtained better from-scratch results using a canonical policy architecture employed by many prior works: a ResNet visual encoder with FiLM [154] language conditioning, combined with a small transformer action decoder trained with a diffusion objective, similar to [20, 230, 37, 127]. Our instantiation of this architecture has 28M parameters (similar to RT-1 [20]). We adopt this as our from-scratch baseline (**"ResNet+Transformer Scratch"**). We also compare to a pretrained visual representation following the procedure of Majumdar et al. [129]. A ViT-B visual encoder is initialized to the VC-1 weights [129], a state-of-the-art visual representation pretrained on 4,000 hours of ego-centric videos and ImageNet, and combined with an MLP action decoder. The full model is trained to predict expert actions using an MSE loss (**"VC-1"**).

3.4.1 Octo Controls Multiple Robots Out-of-the-Box

[2]For the WidowX, since RT-2-X is not openly available, we report the RT-2-X numbers from [13] (dashed bar) and use the same

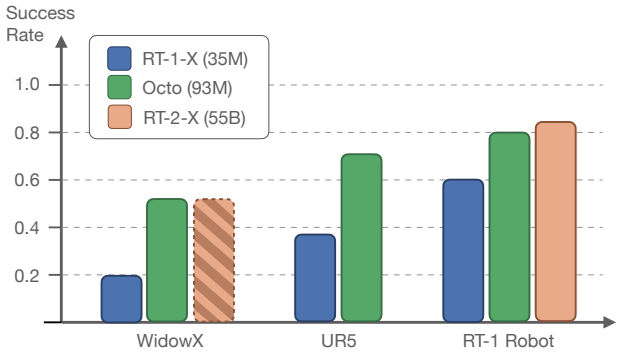


Figure 3.4: **Zero-Shot Evaluation.** Out-of-the-box, Octo can control multiple robots in environments from the pretraining data. When using natural language to specify tasks, Octo outperforms RT-1-X [148], the current best openly available generalist robot policy across three different robot embodiments and setups. Octo also performs similarly to RT-2-X [239] on the tested WidowX and RT-1 Robot tasks.[2]

We compare the zero-shot manipulation capabilities of Octo, RT-1-X, and RT-2-X in fig. 3.4. We evaluated on several tasks selected from the pre-training dataset including picking and placing, wiping a table with a cloth, and opening and closing drawers. For each robot, we selected two language tasks from the corresponding OXE dataset and performed 10 trials per task with varying initial conditions (details in Appendix 8.2.6). The chosen tasks are "in-distribution" from the pre-training data, but the evaluation requires methods to generalize to new object positions, lighting conditions, backgrounds, and distractor objects. While all methods acted reasonably across tasks in the pretraining environments, we found that on average Octo had a 29% higher success rate than RT-1-X (35M parameters). For the WidowX and RT-1 Robot evaluations, we also compared to RT-2-X (55 billion parameters) [239] and found that Octo performed similarly.

Additionally, while RT-1-X and RT-2-X only support conditioning on language instructions, Octo also supports conditioning on goal images. We evaluated our model on the WidowX tasks using goal image conditioning and found that it achieved a 25% higher success rate than when evaluated with language conditioning. This is likely because goal images provide more information about how to achieve the task. In the BridgeV2 domain, we performed a fine-grained analysis of the zero-shot capabilities in table 8.7; measuring performance on setups seen in the dataset, and for novel environments, scenes, and skills. While the Octo model achieves high success on novel objects, zero-shot performance slightly degrades in a new scene, and high degradation for novel behaviors like flipping or precise insertion.

3.4.2 Octo Enables Data-Efficient Learning in New Domains

We report data-efficient finetuning results to new domains in table 3.1. We find that finetuning Octo leads to better policies than starting from scratch or with the pretrained VC-1 weights. On average across the six evaluation setups (detailed in Appendix 8.2.6), Octo outperforms the next best baseline by 52%. Importantly, we use the same recipe and hyperparameters for fine-tuning Octo on all evaluation tasks (see section 3.3.3), making this a good default configuration.

tasks for the Octo and RT-1-X evaluations. For the RT-1 Robot, the authors of RT-2-X kindly performed the evaluations for us.

	Berkeley Insertion*	Stanford Coffee	CMU Baking	Berkeley Pick-Up†	Berkeley Coke	Berkeley Bimanual†	Average
ResNet+Transformer Scratch	10%	45%	25%	0%	20%	20%	20%
VC-1 [129]	5%	0%	30%	0%	10%	50%	15%
Octo (Ours)	**70%**	**75%**	**50%**	**60%**	**100%**	**80%**	**72%**

Table 3.1: **Finetuning Evaluation.** Octo enables data-efficient finetuning to new domains and outperforms training from scratch as well as state-of-the-art pretrained visual representations. Each domain uses $\sim$ 100 target demonstrations and the same finetuning hyperparameters. In each domain, success rates are averaged over 20 trials. $*$: New observation input (force-torque proprioception). †: New action space (joint position control).

		Aggregate Performance
	Octo-Small (Ours)	**83%**
DATA	RT-X dataset mix [148]	60%
DATA	Single robot dataset (Bridge Data)	43%
POLICY	Discretized Action Prediction [148]	18%
POLICY	Continuous Action Prediction (MSE)	35%
ARCH	Resnet-50 + Transformer[148]	70%

Table 3.2: **Model Ablations.** We achieve best performance when using the ViT architecture, diffusion action head, and wide training data mixture. All evaluations are performed on the WidowX setup. Success rates are averaged over 40 trials across two language-conditioned tasks and two goal-conditioned tasks.

The results also underline Octo's ability to accommodate new observations (force-torque inputs for "Berkeley Insertion"), action spaces (joint position control for "Berkeley Pick-Up") and new robot embodiments ("Berkeley Coke" and "Berkeley Bimanual"). This makes Octo applicable to a wide range of single and dual arm robotic manipulation problems that go beyond a single camera input and end-effector position control.

3.4.3 Design Decisions for Generalist Robot Policy Training

We have demonstrated the effectiveness of Octo as a zero-shot multi-robot controller and as an initialization for policy finetuning. We next analyze the effects of different design decisions on the performance of the Octo policy. Concretely, we focus on the following aspects: (1) model architecture, (2) training data, (3) training objective, and (4) model scale. Unless noted otherwise, we perform all ablations on the Octo-Small model due to our compute budget.

Model architecture Prior transformer-based policy designs typically encode input images with large ResNet-style [69] encoders and fuse the resulting image features with a comparatively small transformer [20, 148, 181, 37, 230, 137, 185]. Instead, we opt for a "transformer-first" architecture that uses very shallow CNN patch encoders and concentrates most of the parameters and FLOPS in the transformer backbone, similar to canonical vision transformer architectures [49]. In Table 3.2 we show that this scalable architecture leads to substantially improved performance when training on the full Open X-Embodiment data

mix. Importantly, we found ResNet-based architectures to perform better than ViTs when training on small datasets, e.g., in our "from scratch" comparisons, underlining that large transformer policies are uniquely suited for scalable training on diverse datasets.

Training data Octo is trained on the most diverse cross-embodied robot dataset to date, a mix of 25 datasets that we manually curated from the Open X-Embodiment dataset [148] (see section 3.3.2). We ablate the impact of this training mix by comparing to Octo models trained on a smaller mix of 11 datasets used in training the RT-X models [148] and a baseline trained only on data from the target robot domain. In Table 3.2 we show that the performance of Octo increases as we increase the number of training datasets. This suggests that expanding the data mix to even more datasets may further improve policy performance. We will leave this for future work, along with a more thorough investigation of best practices for data curation.

Training objective We compare Octo's diffusion decoding training objective (see section 3.3.3) to common alternatives from prior work: simple MSE loss [17, 111] and cross-entropy loss on discretized actions [20, 239]. In Table 3.2 we find that Octo's diffusion training objective leads to substantially improved performance. This improvement is likely because the diffusion head can model multi-modal action distributions (unlike the MSE head) while retaining the precision of continuous actions (unlike the discrete head). Qualitatively, the policy acts more decisively than MSE-trained policies, and more precisely than those trained with discretized actions.

Model scale We compare Octo models of three different sizes following the ladder of common vision transformer models [227]: Octo-Tiny (10M), Octo-Small (27M), and Octo-Base (93M). In Figure 3.5 we show that the zero-shot performance of the policy scales with increasing model size. We find that the Base model is more robust to initial scene configuration than the Small model, and is less prone to early grasp attempts, indicating the larger model has better visual scene perception.

3.5 Discussion

We introduced Octo, a large transformer-based policy pretrained on the largest robot manipulation dataset to date, 800k robot trajectories. We demonstrated that Octo can solve a variety of tasks out-of-the-box and showed how Octo's compositional design enables finetuning to new inputs and action spaces, making Octo a versatile initialization for a wide range of robotic control problems. Apart from the model itself, we have released our full training and finetuning code, alongside tools that make it easier to train on large robot datasets.

While Octo achieves strong performance in both zero-shot and finetuning evaluations, we find that the current model still has several short-comings, which we attribute in large parts to characteristics of the training data. First, we found that the current Octo model struggles with adequately processing wrist camera information. Often finetuning results were stronger when using only a third person camera instead of combining third person and wrist camera. Additionally, we notice a large difference between language-conditioned policy performance and goal-conditioned policy performance. In both cases, a lack

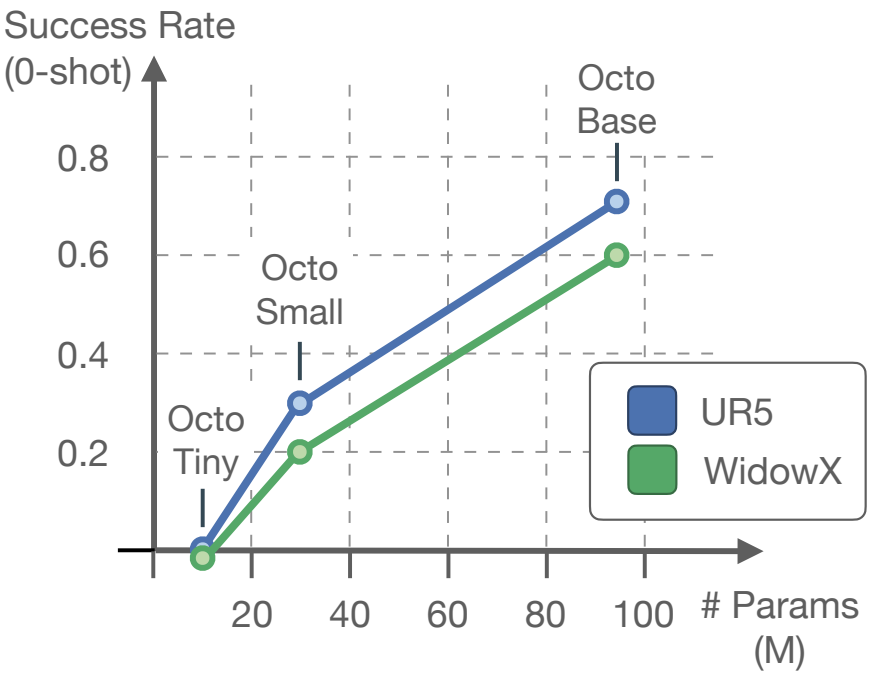


Figure 3.5: **Model Scaling.** The performance of Octo improves with larger model sizes on both UR5 and WidowX tasks. Success rates are averaged over 10 trials on one language-conditioned task per robot.

of the respective modalities in the training data is the likely reason: only 27% of the data contains wrist camera information and only 56% of the pretraining data contains language annotations.

Expanding the data used to train Octo is a natural avenue of improvement. Since the Open X-Embodiment dataset is comprised of optimal robot demonstrations, the current model trains via imitation; future work may consider learning from sub-optimal or online interaction data that require alternative objectives. Further, while we trained and evaluated Octo exclusively on single and dual-arm manipulators; expanding to a wider set of robots that perform navigation or mobile manipulation would be an direction of high opportunity.

While Octo represents a step towards building generalist robot policies that work out-of-the-box on diverse robot setups, there remains work to improve the model, including better language conditioning, improved support for wrist cameras, and incorporating data beyond optimal demonstrations. We hope that Octo offers a simple launchpad for researchers and practitioners to access larger robotic datasets and leverage pretrained robotics models for efficient learning of new tasks and broad generalization.

Chapter 4

Controlling Multiple Robot Embodiments with a Single Policy

4.1 Motivation

Much of the recent success in machine learning has been driven by training general-purpose models on increasingly diverse and multi-task data. For example, visual and language tasks, once handled by task-specific methods, are now performed more effectively by general vision-language models that can transfer knowledge across tasks [18, 208, 119, 150]. Similarly, in robotics, recent data aggregation efforts [148] have made it possible to train general-purpose policies on robot data collected across multiple embodiments, tasks, and environments. These generalist policies can out-perform narrow policies trained with data from only the target robot and task by transferring visual representations and skills [147, 148]. In addition to the benefits of positive transfer, training generalist cross-embodied policies minimizes the amount of engineering that must be done to design and tune policy architectures for each robot.

However, training a general-purpose robot policy is uniquely challenging, since robot systems can vary widely in their camera views, proprioceptive inputs, joint configurations, action outputs, and control frequencies. Initial efforts at training large-scale cross-embodied policies have often been limited to single robot arms or ground navigation robots which can be controlled with a single camera view and relative waypoint action for the base or end-effector [148, 147, 217, 216]. Further increasing the diversity of embodiments these policies can control requires a model architecture that supports conditioning on any number of camera views or proprioceptive observations, as well as predicting actions of any dimension. Following prior work, we take a sequential modeling approach to cross-embodied imitation learning [167, 19]. We propose a transformer-based policy that supports variable observations and actions by casting inputs and outputs to a sequence. We scale this approach to control the most diverse set of embodiments with a single policy to date, including single and bimanual robot arms, ground navigation robots, quadcopters and quadrupeds.

With our transformer policy, we can train on robot data with any number of camera views or proprioceptive sensors by simply tokenizing and arranging the observations into a sequence. At the same time, we can predict actions of any dimension, crucially without the need to manually align the action spaces of different embodiments [216]. For each action type, we insert a set of *action readout tokens* into the

input token sequence. Then, we pass the corresponding output embeddings into action-space specific heads to produce a vector of the correct dimensions. Our policy can accept tasks in the form of language instructions or goal images, allowing a user to choose the task modality that is most natural for a given embodiment.

Our primary contribution is a cross-embodied robot policy trained on the largest and most diverse robot dataset to date with 900K trajectories and 20 distinct embodiments. Our policy can control robots with varying observation and action types, from a quadruped with proprioceptive sensors and 12 joints to a bimanual robot with 3 cameras and 14 joints. We find in extensive real world experiments that our policy matches the performance of the same architecture trained on just the target robot data, as well as the best prior method in each setting, demonstrating that our architecture can absorb heterogeneous robot data without negative transfer, while performing comparably to state-of-the-art specialist methods tailored for each robot. Additionally, we find that our method outperforms the state of the art in cross-embodiment learning while alleviating the need for manually aligning observation and action spaces.

4.2 Related Work

Early work on cross-embodied robot policy learning has explored a number of techniques, including conditioning on explicit or learned representations of the embodiment, [32, 174], domain randomization and adaptation [152, 104, 59, 165, 203, 226], modular policies [47, 78, 34, 220], or model-based RL [75, 62, 171]. Generally, these prior projects have operated on a smaller scale, only evaluating in simulation or training policies on small amounts of robot data to control a small number of robots for a few tasks.

A number of prior works have sought to scale up robot learning by using large amounts of data from a single robot embodiment, collected either autonomously [159, 114, 89, 60, 66, 108] or with human teleoperation [20, 239, 87, 97, 207, 54, 131, 10]. Other prior works train on data from multiple robots, but require each robot to have the same observation and action space [217, 94, 148, 180, 181, 122, 176]. For example, Shah et al. [181] train one policy across many navigation robots using an egocentric camera view and 2D waypoint actions, and the RT-X models [148] are trained across single robotic arms using a 3rd person camera view and 7-DoF end-effector position actions. Our method does not require the data to have a common observation and action space, enabling simultaneous control of robots with disjoint sets of sensors and actuators, e.g., robot arms and quadrupeds.

There have been a few large-scale efforts to train a single policy on robot data with varying observations and action spaces [19, 167, 147, 216]. Octo [147] can be fine-tuned on robots with observations and actions that are distinct from those seen during pre-training. However, Octo only pre-trains on data from single robot arms and does not explore co-training on more heterogeneous data. Reed et al. [167] and Bousmalis et al. [19] propose a flexible transformer-based policy that handles varying observation and action spaces during pre-training. They demonstrate their policy can control robot arms with different action spaces, including 4-DoF, 6-DoF, 7-DoF, and 14-DoF (using a 3-finger hand). We explore the challenges in increasing both the diversity of embodiments and environments a single policy can operate in. Along with increasing the number of arm embodiments we can control, e.g., high-frequency bimanual arms, we demonstrate navigation and quadrupedal locomotion, and we evaluate across a wide range of real-world settings, while they limit their evaluation to a standardized cage. We also explicitly evaluate for transfer across embodiments by comparing policies with and without cross-embodiment co-training, while their

focus is on autonomous self-improvement. Perhaps most related to our work, Yang et al. [216] study transfer across manipulation and navigation data. However, their focus is on leveraging the fact that egocentric motion in navigation looks similar to egocentric motion from wrist cameras in manipulation, and they perform manual alignment of the actions across these two embodiments. Instead, our focus is on training a policy that can control embodiments beyond arms and ground navigation robots, including those with observations and action spaces that cannot be cast to a common format. Our method, CrossFormer, is the first to co-train on four distinct action spaces (single-arm manipulation, ground navigation, bimanual manipulation, and quadrupedal locomotion) without any observation space constraints or action-space alignment while maintaining state-of-the-art performance on each robot.

Training generalist, cross-embodied policies requires multi-robot datasets, and there have been several efforts to collect such large-scale cross-embodied datasets [43, 180, 181, 148, 57]. In particular, the Open Cross-Embodiment dataset (OXE) [148] aggregates 1.5 million episodes of robot data, and we train on a subset of 900Ktrajectories. We note that the Octo [147] and RT-X [148] models are also trained on the OXE dataset, however they only use a subset with single robot arms. We additionally use the GNM [180] navigation and DROID [97] (large-scale Franka manipulation) datasets, along with Go1 quadruped and ALOHA [230] bimanual data collected by concurrent work [44].

4.3 Designing a Cross-Embodied Policy

The primary challenge in robotic learning with many embodiments lies in handling widely varying observation and actions spaces, as well as differences in control frequency and other aspects of the robotic system. Robotic systems can have varying numbers of camera views or proprioceptive sensors, and they may be controlled by a variety of different action representations, including joint angles, Cartesian positions, and motor torques. In order to standardize the data to a common format, some prior work on training cross-embodied policies has ignored certain observation types (such as either wrist or third person views in manipulation) [148, 217] or aligned action spaces across robots [216]. Instead, following other prior work [167, 19, 147], we cast cross-embodied imitation learning as a sequence-to-sequence problem and choose a transformer-based policy architecture that can handle varying length sequential inputs and outputs.

Due to their sequential nature, a transformer policy enables encoding all available observation types from each embodiment by serializing them into a flat sequence. Similarly, we can decode variable length actions which allows us to use the optimal action type for each embodiment. Using this flexible output, we can also predict variably sized chunks of actions. Action chunking [136, 230, 37] improves the temporal consistency of actions and reduces compounding error which is particularly important for high-frequency fine manipulation. Together, a transformer backbone and action chunking enables our policy to control robots ranging from a bimanual ALOHA system with joint position control at 20Hz, to ground and aerial navigation robots with 2D waypoint control at 5Hz.

At a high level, our transformer policy follows prior work that trains transformers on multi-modal data [167, 19, 147]. Observations and a task specification are tokenized by modality-specific tokenizers, assembled into a token sequence, and fed into a causal, decoder-only transformer backbone that is shared across all embodiments. The output embeddings are then fed into separate action heads for each class of embodiment to produce actions of the corresponding dimension, as shown in Figure 4.1.

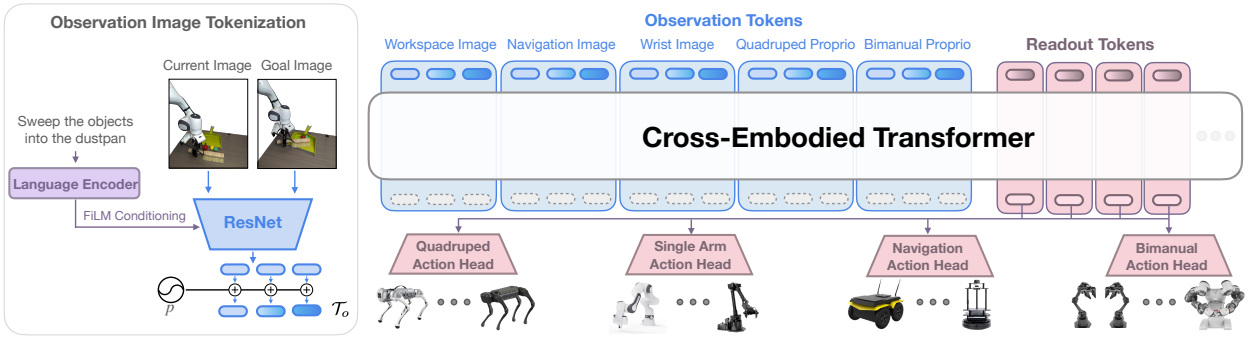


Figure 4.1: **Policy architecture.** Our architecture enables cross-embodied policy learning through a transformer backbone. Our policy accepts variable observation inputs by tokenizing images and proprioceptive information, predicts variable action outputs using action readout tokens, and conditions on language instructions or goal images.

4.3.1 Training data

Our training data mixture covers 20 different robot embodiments, varying widely in observation space, action space, and control frequency. We start from the single-arm manipulation subset of the Open Cross-Embodiment dataset used by Octo [147]. Then, we add the DROID Franka manipulation dataset [97], 7K trajectories of ALOHA data collected across two institutions (referred to as ALOHA-multi-task), 60 hours of navigation from the GNM dataset [180], 25 minutes of walking data from a Go1 quadruped (referred to as Go1-walk), and 200 trajectories of additional Franka data collected in our own lab (referred to as Franka-tabletop). We classify the datasets that are most relevant to each of our evaluation settings (see Section 4.4) as the *target datasets* and up-weight them relative to the other datasets during training. Our target datasets are BridgeData [207] for WidowX evaluation, ALOHA-multi-task for ALOHA evaluation, GNM [180] for navigation evaluation, Go1-walk for quadruped evaluation, and Franka-tabletop for Franka evaluation. We collected the Go1 data by rolling out an expert policy trained with RL in simulation [188]. See Figure 4.2 for our training data mix and Appendix 8.3.1 for further details on the composition of datasets collected ourselves along with the sampling weights.

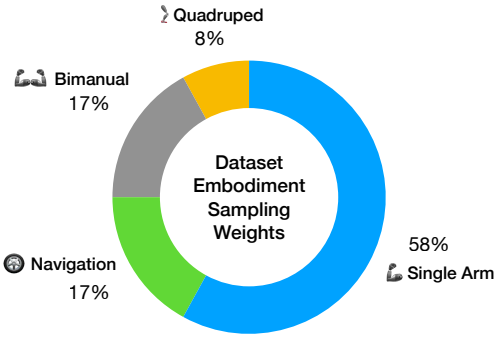


Figure 4.2: **Training Data Mix.** We group the 20 embodiments in our training data into categories and visualize their contribution to our data mix. The pie chart shows the average composition of each training batch based on the sampling weights.

4.3.2 Tokenizing variable observation types and task specifications

The first step in training our cross-embodied policy is creating the input sequence. The trajectories in our robot training data are sequences of time-steps where each time-step contains image observations I, proprioceptive observations P, and an action. Data from each embodiment may have a different number of camera views per time-step and may or may not include proprioceptive observations. To create the

input sequence, we first define an observation history length k and chunk each trajectory into k-length segments, $[I_t, P_t ..., I_{t+k}, P_{t+k}]$. We then tokenize each observation according to its modality. Images are processed with a ResNet-26 encoder [69] to produce a feature map that is flattened along the spatial dimensions and projected to the token embedding size. Proprioceptive observations are simply projected to the token embedding size. Along with the observation sequence, the policy also accepts a task specification. Importantly for cross-embodied control, our policy accepts task specifications either in the form of a language instruction l or a goal image g. In some settings, like navigation, tasks are more naturally specified as an image goal whereas in other settings, like manipulation, tasks are more easily specified with language. Language instructions are jointly processed with image observations using FiLM [154]. Goal images are stacked on the current image along the channel dimension before being fed into the image encoder.

Because our training data consists of data from single-arm manipulators, dual-arm manipulators, quadrupeds, and ground navigation robots, our policy supports conditioning on any subset of the following observation types: **(1) Workspace Image:** A 3rd person camera view in manipulation settings. **(2) Navigation Image:** An egocentric camera view in navigation settings. **(3) Wrist Image:** A view from a wrist-mounted camera in manipulation settings. **(4) Quadruped Proprioception:** Joint positions and velocity estimates for quadrupeds. **(5) Bimanual Proprioception:** Joint positions for bimanual manipulation settings. To maximize transfer across embodiments, we share image encoder weights for camera views of the same type. So, for example, the workspace images in single arm and bimanual manipulation settings are processed by the same ResNet image encoder. In total we use four image encoders: one for the workspace view in manipulation settings, one for the egocentric view from ground navigation robots, and two for wrist cameras in manipulation settings. After input tokenization, we have a sequence of observation tokens $[I_t^{1:L}, P_t^{1:M} ..., I_{t+k}^{1:L}, P_{t+k}^{1:M}]$, where L and M denote the number of tokens for image and proprioceptive observations.

4.3.3 Predicting variable length actions

After creating the input sequence, the next step is to process the input sequence with a transformer to predict an action of an appropriate dimension for each embodiment. We use a transformer with a blockwise causal attention mask such that observation tokens can only attend to observation tokens at the same or prior time-steps t. Following prior work [147], we insert special readout tokens R after the observation tokens at each time-step in the input token sequence. These readout tokens can only attend to prior observation tokens, and thus serve as a convenient representation from which to predict actions. The final input sequence is $[I_t^{1:L}, P_t^{1:M}, R_t^{1:N} ..., I_{t+k}^{1:L}, P_{t+k}^{1:M}, R_{t+k}^{1:N}]$ where N denotes the number of readout tokens. We pass the input token sequence through the transformer to obtain an embedding sequence. We then apply an action head to the embeddings corresponding to the readout tokens to produce actions. There are several possibilities for the action head. Past work has explored regression with an L1 or L2 loss, classification with a cross-entropy loss, or diffusion. We choose to predict continuous actions and employ an L1 loss because of its success in prior work on high-frequency bimanual manipulation [230]. Accordingly, our action heads simply project the readout token embeddings to the action dimension. For some embodiments, we predict a chunk of sequential actions. Action chunking has been shown to improve policy performance in prior work [230, 37] and is essential for embodiments with high control frequencies where compounding error would build up too quickly. Since our action heads project the readout tokens to the action dimension size, we match the number of readout tokens to the action chunk

size for each embodiment. Our policy has 4 action heads which produce chunked actions of the following types: **(1) Single arm Cartesian positions:** A 7-dimensional action denoting the relative change in Cartesian position of the end-effector and the gripper actuation. We predict a chunk of 4 actions and execute on single robot arms at 5-15Hz [147]. **(2) Navigation waypoints:** A 2-dimensional action denoting a waypoint relative to the robot's current position. We predict a chunk of 4 actions and execute on navigation robots at 4Hz [181] . **(3) Bimanual joint positions:** A 14-dimensional action denoting the joint positions of both arms. We predict a chunk of 100 actions and execute on bimanual robot arms at 20Hz [230]. **(4) Quadruped joint positions:** A 12-dimensional action, predicted with no chunking, denoting the joint positions of the legs. We predict only 1 action and execute on a quadruped at 20Hz [188]. Action chunk sizes are taken from prior work in each robot setting.

4.3.4 Training details

In practice, we mask observations that are missing for an embodiment, so that each batch element contains all observation types and all readout token groups, and these token groups occupy fixed locations within the context window. Alternatively, for memory efficiency, observation and readout tokens could be densely packed to remove padding and fit more time-steps of context for embodiments with fewer observation types. This is a strategy used by prior work [167]. However, by not fixing observation and readout token types to a set location in the context window, the model would need to infer the embodiment purely from the observations in order to predict actions of the correct type (rather than relying on the positional embeddings of the readout tokens). The observations for some embodiments can look similar (such as navigation and manipulation with only a wrist camera), so this design may require appending a prefix to the token sequence indicating the embodiment.

Our transformer backbone has 12 layers, 8 attention heads, an MLP dimension of 2048, and a token embedding size of 512. In total, our model has 130M parameters. We initialize the ResNet-26 encoders with ImageNet pre-trained weights. We use a context widow size of 2135 tokens, which fits 5 time-steps of context with all observation and readout token groups present. We found that good performance on navigation required 5 time-steps of observation history and using this context length did not hurt performance for other embodiments. We trained for 300K gradient steps with a batch size of 512 which took 47 hours on a TPU V5e-256 pod. We use the AdamW optimizer [123], an inverse square root decay learning rate schedule [227], weight decay of 0.1, and gradient clipping of 1.0. We apply standard image augmentations. During training, we use hindsight goal relabeling and sample future observations uniformly at random to use as goals [3]. If a language instruction is available for a trajectory, we randomly mask either the language or goal so that at test time we can condition our policy using either task specification [126, 137].

4.4 Evaluation

Our experiments are designed to test whether our single cross-embodied policy can control multiple robot embodiments without sacrificing performance when compared to robot-specific imitation learning approaches. Specifically, we aim to answer the following questions: **(1)** Can our policy trained on many robots match the performance of a policy trained on data from only the target robot? **(2)** Does our policy match the performance of the best prior imitation learning method in each robot setting? **(3)** How

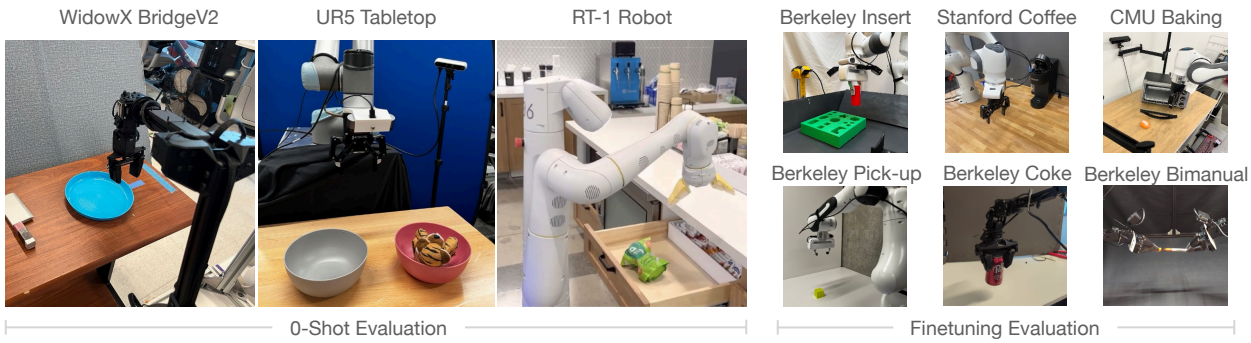


Figure 4.3: **Evaluation Settings:** Our tasks include single-arm manipulation settings, dexterous and bimanual task settings, navigation, and aviation. Please refer to Sec 4.4 for a full breakdown.

does our policy compare to prior cross-embodied policies that align observation and action spaces?

Evaluation setups. We evaluate over a wide range of tasks and embodiments (see Fig. 4.3): **(1) WidowX Manipulation**: We use the Bridge setup from Walke et al. [207]. We use an over-the-shoulder camera view and and sample actions from the single arm head of our policy. We perform 48 trials per policy over two language-conditioned tasks and two goal-conditioned tasks. **(2) Franka Manipulation**: We use the DROID setup from Khazatsky et al. [97]. We use an over-the-shoulder camera view and sample actions from the single arm head of our policy. We perform 39 trials per policy over two language-conditioned tasks. **(3) ALOHA Bimanual Manipulation**: We use the ALOHA setup from Zhao et al. [230]. We use 3 camera views, one overhead and two wrist, and sample actions from the bimanual head of our policy. We perform 20 trials per policy over two language-conditioned tasks. **(4) LoCoBot Navigation**: We use the LoCoBot setup from Shah et al. [181] which has one camera view. We evaluate on suite of three skills: path-following, obstacle avoidance, and sharp corner-turning. We collect a topological map of goals similar to Shah et al. [181] and evaluate success based on distance to the closest node within the topological map of goal images. We sample actions from the navigation head of our policy. We evaluate in 6 locations, using one trial per location and policy as done in prior work [180, 181]. **(5) Go1 Quadruped**: We evaluate on a Unitree Go1 which uses proprioceptive observations $o_t \in \mathcal{R}^{59}$. We sample actions from the quadruped head of our policy. As our evaluation metric we report the average reward achieved over 25 minutes normalized by the the reward achieved by the RL-trained expert policy that generated the data (see Section 4.3.1). **(6) Tello Quadcopter**: Finally, we experiment with a Tello quadcopter using the navigation head of our policy. Since the navigation head outputs 2-D relative waypoints, we maintain a static height throughout the trajectory [181, 180]. Notably, we do not train on quadcopter data so this setting requires *zero-shot* generalization to a new embodiment (though not to a new set of observation inputs and action outputs). We evaluate in 3 locations, using one trial per location and policy as done in prior work [180, 181].

Baselines and comparison methods. To evaluate how co-training on data from other embodiments affects performance for a given embodiment, we train our policy architecture on just the data from the target embodiment using the target dataset (see Section 4.3.1) for each evaluation setting. Because these datasets are smaller than our full data mix, we create smaller versions of our architecture, ranging from 5M to 95M parameters, to avoid overfitting. We refer to this method as the **single-robot dataset** baseline. We also compare our method to the **best prior method** in each setting to evaluate whether we can match the performance of an approach that was specifically tuned for each robot. For the WidowX we use Octo

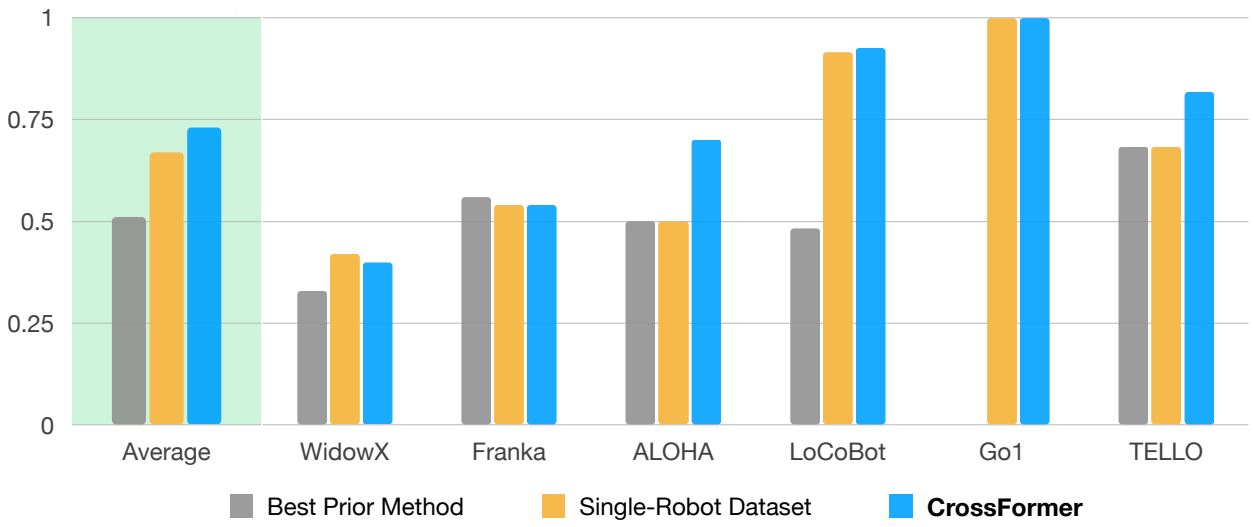


Figure 4.4: **Real Evaluation.** We compare CrossFormer to the same architecture trained on just the data from the target robot, as well as the best prior method on the data from the target robot.

[147], for the Franka we use OpenVLA [99], for ALOHA we use ACT [230], for the LoCoBot and Tello quadcopter we use ViNT [181]. We do not compare to a prior state-of-the-art method for the Go1 setting due to the lack of a corresponding imitation learning method for quadrupedal locomotion. Finally, in the navigation and manipulation settings, we compare to the policy from **Yang et al. [216]**, by using the original 186M parameter policy. Yang et al. [216] leverage the fact that the egocentric view in navigation looks similar to wrist camera views in manipulation, and they align the action spaces of these two settings accordingly. This comparison evaluates whether our approach can perform similarly with no manual alignment of action spaces.

4.4.1 Can CrossFormer match the performance of training on only a single-robot dataset, as well as the best prior method in each setting?

Figure 4.4 shows the performance of our method compared to our architecture trained on just the single-embodiment target datasets for each setting (see Section 4.3.1) and the best prior method for each setting. Overall, we find that our method performs comparably to training on just the target dataset in all evaluations, demonstrating that our architecture can absorb data from widely varying embodiments with no negative transfer. On average over all embodiments, CrossFormer has a 73% success rate, while the single-robot dataset baseline has a 67% success rate. Additionally, CrossFormer performs similarly to the best prior imitation learning method in each evaluation setting, demonstrating that our policy architecture can match state-of-the-art methods that have been specifically developed and tuned for each robot. On average over all embodiments, CrossFormer has a 73% success rate, while the best prior method has a 51% success rate. Because there is no suitable prior imitation learning method for the Go1, it is not included in the reported best prior method average.

4.4.2 Is aligning the observation and action spaces really needed?

Lastly, we compare our policy to Yang et al. [216] in the LoCoBot navigation and WidowX manipulation settings. In the navigation setting, our policy significantly outperforms Yang et al. [216] in all tested

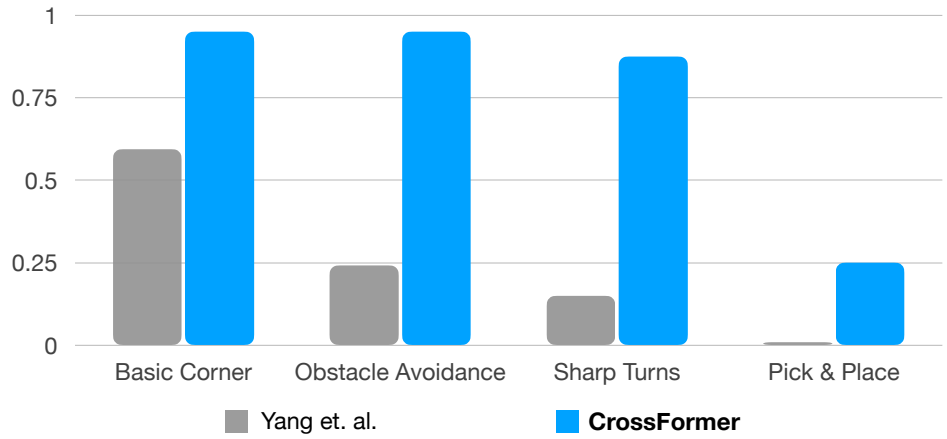


Figure 4.5: **Comparison to Yang et al. [216].** We compare CrossFormer to Yang et al. [216], which aligns actions for navigation and manipulation and only uses a single camera view at a time. CrossFormer outperforms Yang et al. [216] by 3x overall, on both tabletop manipulation from a third person camera view as well as common navigation tasks.

scenarios including turning corners, avoiding obstacles, and making sharp turns. Qualitatively, we also found our policy to be much smoother, navigating in straighter lines with fewer stops and starts. Both policies are trained on the same amount of LoCoBot data (from the GNM dataset), so this result indicates that our policy architecture can better fit its cross-embodiment training data. Surprisingly, in the WidowX manipulation setting, we found that Yang et al. [216] obtained a zero success rate, despite training on the same relevant WidowX data as CrossFormer (i.e the Bridge dataset) and additional WidowX data collected in their own lab. We hypothesize that this is because we evaluate from the 3rd-person camera view. While Yang et al. [216] train with both 3rd-person and wrist camera views, only one view is input to the policy at a time, potentially resulting in underfitting to 3rd-person control. CrossFormer only uses the 3rd-person view for the WidowX setting due to a lack of publicly available WidowX wrist-camera data. However, we note that our policy can accept a varied number of camera views (using up to three simultaneously for bimanual manipulation) and does not require forcing two camera views into the same input slot. In summary, our policy outperforms Yang et al. [216] in both settings, demonstrating that manual alignment of observation and action spaces is not necessary for good performance, and that our flexible architecture allows our policy to better fit heterogeneous cross-embodied data.

4.5 Discussion

We introduced CrossFormer, a scalable and flexible transformer policy trained on the largest and most diverse dataset to date, 900K trajectories across 20 different robot embodiments. We demonstrated a principled way to learn a single policy that can control vastly different embodiments including single and dual arm manipulation systems, wheeled robots, quadcopters, and quadrupeds. Our results show that CrossFormer matches the performance of specialist policies tailored for individual embodiments, while also significantly outperforming the state of the art in cross-embodiment learning. However, our work does have limitations. Our results do not yet show significant positive transfer across embodiments. We anticipate that as we train on larger robot datasets with more embodiments, we will see greater positive transfer. Another limitation is that our data mix uses hand-picked sampling weights to avoid over-training

on datasets with many repetitive episodes and under-training on the data most relevant to our evaluation settings. In principle, as we scale model size, the policy should have the capacity to fit all the data equally well without any data weighting. Finally, given that we need large models to fit large multi-robot datasets, the model's inference speed can become a limiting factor. In this work we successfully applied our policy to a high-frequency, fine-grained bimanual manipulation task, but as we scale the model's size we may not be able to control these higher frequency embodiments. Future hardware improvements will help to alleviate the issue, but further research is needed on techniques for using large models to control high-frequency robots. Future work could also include exploring techniques to enable greater positive transfer across embodiments while maintaining the flexibility of our architecture, techniques for data curation, and incorporating even more diverse data sources like sub-optimal robot data or action-free human video. We hope that this work will open the door for more general-purpose and flexible robot policies that can efficiently learn and transfer knowledge from the experience collected across diverse robot embodiments.

Chapter 5

Adding Memory to a Generalist Robot Policy

5.1 Motivation

Efficiently and effectively endowing robotic policies with memory requires multiple levels of abstraction. While in principle we could simply encode the entire sequence of past observations into the context of the policy, this becomes intractable for long tasks, necessitating either very short sequences or significant subsampling. However, in practical settings, the *representation* required for long- and short-term memory is likely to be very different. For example, the robot might need to remember a recent observation to handle occlusions, and it might need to remember that it has already added one of the ingredients when cooking a meal. But these memories are fundamentally different: the former might require storing a few images over a short time period, the latter might require long-term memory but only a few bits of information.

An effective memory architecture for robot policies should use multiple modalities to represent memories at these different levels of abstraction. For short-horizon memory, dense image-based memory is well-suited to resolve occlusions and allows the robot to quickly adapt its manipulation strategy, e.g., by changing the grasp after failing to pick up an object. For long-horizon memory, we often only need to keep track of events at a *semantic* level, such as which ingredient has already been added to a dish. In this case, a language-based representation provides much better compression than raw observations, and allows us to store high-level memories over long time periods.

Based on these observations, we introduce *Multi-Scale Embodied Memory* (MEM), a system for equipping policies with multi-modal, long-horizon memory. MEM combines two key ingredients to make long-horizon memory tractable. First, we use a video encoder architecture to effectively encode multiple seconds of dense image-based memory into a compact representation. Second, we introduce a language-based memory mechanism in which the policy keeps track of semantic events in a compressed language format. This memory system can not only accommodate very long horizon tasks, but also enables a variety of new capabilities by leveraging the short-term memory, such as in-context adaptation to correct mistakes, and resilience to partial observability and self-occlusion.

To evaluate MEM, we integrate it into the $\pi_{0.6}$ model [156], a generalist VLA trained on a diverse mixture of robot, vision-language, and video data. We show that the resulting policy achieves state-of-the-art performance across a wide range of complex manipulation tasks. We also show that MEM enables our policy to solve long-horizon tasks like cleaning up a whole kitchen or preparing a grilled cheese sandwich,

which require keeping track of memories for up to fifteen minutes.

In summary, we introduce a system for multi-scale, long-horizon memory for robot policies. By effectively representing short- and long-horizon memory via video and language representations respectively, MEM allows robot policies to keep track of memories across tens of minutes, without sacrificing runtime constraints. To implement MEM, we use an efficient video encoder architecture and a language-based memory system, and demonstrate their effectiveness through state-of-the-art performance across diverse robot tasks. With MEM, we enable robot policies to perform complex tasks like cleaning up whole kitchens, which can span up to fifteen minutes.

5.2 Related Work

Vision language action models with memory. Recent works have demonstrated that learned robot control policies, trained on large amounts of diverse robot experience, can lead to generalizable manipulation, e.g., in unseen environments [81, 80, 199, 202, 146, 219]. Vision language action models (VLAs) [50, 21, 148, 147, 48, 98, 14, 211, 231, 7, 196, 155, 200, 212, 12, 214, 201] are a popular approach for training such generalist policies, in which a pre-trained vision-language model is finetuned with robot experience. While many of today's state-of-the-art VLAs are trained without memory and act purely based on the current observation of the environment [98, 80, 81, 199, 202, 146], a growing number of works have explored adding memory to policy training since it is a core requirement for solving a wide range of real-world tasks. While early works explore architectures with recurrent memory modules [164, 132], more recent works that use transformer-based architectures simply pass a dense history of prior observations into the policy [177, 110, 147]; computational and latency constraints make it challenging to scale such approaches to support very long-horizon memory. Some works have explored latent memory architectures [115, 184, 58, 40], but only evaluated on short-horizon memory tasks. Others have proposed various heuristics to compress memory information, e.g., by relying on purely proprioceptive memory [229], 2D point tracks [233, 29], by only retaining keyframes from prior timesteps [192, 210, 135], or by representing memory in natural language [117]. A challenge for all these approaches is that it is hard to find a one-modality-fits-all solution for robot memory, and each individual representation will result in a compromise on capabilities: proprioception, point traces, or natural language alone, for example, lose precise spatial information about grasp angle and height that is necessary to correct a slipped grasp. Keyframes, on the other hand, need to aggressively sparsify the observation history to make inference computationally feasible in long-horizon tasks, but may lose the ability to estimate environment and robot dynamics, which often requires more densely sampled observations. In contrast to these prior works, we introduce a *multi-modal* memory system that combines short-horizon, dense vision-based memory with long-horizon, language-based memory. This allows us to resolve partial observability, perform in-context adaptation, and solve long-horizon tasks, all without sacrificing efficiency. Finally, an orthogonal line of work proposes approaches for mitigating causal confusion [39, 233] in which a policy with memory erroneously learns to copy over prior actions, e.g., by introducing auxiliary objectives [204]. While similar approaches *could* be combined with our memory system, our experiments suggest that we can achieve high policy performance without such objectives, which may be attributed to our large-scale and diverse training data.

Long-Context Models. Outside of robotics, a large body of work has explored the training of long-context language and vision-language models [121, 198, 144]. In particular, in the context of video pro-

cessing, there is a rich literature on designing efficient encoders for long video inputs [4, 9, 116]. Our work leverages similar ideas to [9, 4] for using sparse attention operations to efficiently process video inputs. Yet, in the context of robotics, latency constraints imposed by the real world require additional efficiency considerations: processing more than ten minutes of high-frequency, multi-frame video within a latency budget of a few hundred milliseconds is challenging, even on modern hardware. Thus, our work combines a short-horizon, video-based memory architecture with a significantly more compressed, language-based memory architecture for effective long-horizon context.

5.3 Multi-Scale Embodied Memory for VLAs

In this section, we describe our proposed system for equipping VLAs with memory on multiple time scales. For robot policies deciding on the next action to take, relevant context often spans multiple time scales. The most recent timesteps help understand the dynamics of the robot and scene, resolve self-occlusions, and quickly adapt fine-grained manipulation strategies like re-grasps. On the other hand, long-horizon memory is required to understand, for example, which steps of a recipe have already been completed, or which subtask has repeatedly failed. In theory, both types of context could be provided by conditioning the policy on a dense sequence of all its previous observations. However, this becomes quickly infeasible for tasks spanning tens of minutes. Our goal in designing *Multi-Scale Embodied Memory* (MEM) is to enable both efficient short *and* long-term memory. For short-horizon memory, we use an efficient **video encoder** architecture that allows us to pass a multi-*second* horizon of observations to the policy, enabling rapid in-context adaptation and robustness to self-occlusions. For long-horizon memory, we design a **language-based memory** representation, and train the model to keep track of past events *at a semantic level*, in natural language. This way, we can equip our policies with memory spanning up to 15 minutes, without sacrificing runtime latency constraints.

We first give an overview of the MEM memory system (section 5.3.1), then provide a detailed description of both the language-based memory (section 5.3.2) and video encoder architecture (section 5.3.3), and finally detail the instantiation of our memory system in the $\pi_{0.6}$-MEM VLA (section 5.3.4).

5.3.1 Multi-Scale Embodied Memory (MEM)

fig. 5.1 shows an overview of our MEM system. Our goal is to train a policy $\pi(a_{t:t+H}|o_{t-T:t}, g)$ that predicts a chunk of continuous robot actions $a_{t:t+H}$ conditioned on the task goal g in natural language, as well as a *sequence* of dense observations (e.g. images, proprioceptive state) provided as input. In comparison, most previous VLAs only condition on a single observation o_t, while we want the policy to process many (T) observations.

As mentioned above, scaling the number of past observations such that they span multiple minutes, and can thus serve as a form of long-range memory is infeasible. Therefore, we factorize the problem of action prediction as follows:

$$\begin{aligned}\pi(a_{t:t+H}, l_{t+1}, m_{t+1}|o_{t-T:t}, m_t, g)\\ \approx \pi_{\text{LL}}(a_{t:t+H}|o_{t-K:t}, l_{t+1}, g)\\ \pi_{\text{HL}}(l_{t+1}, m_{t+1}|o_t, m_t, g),\end{aligned}$$

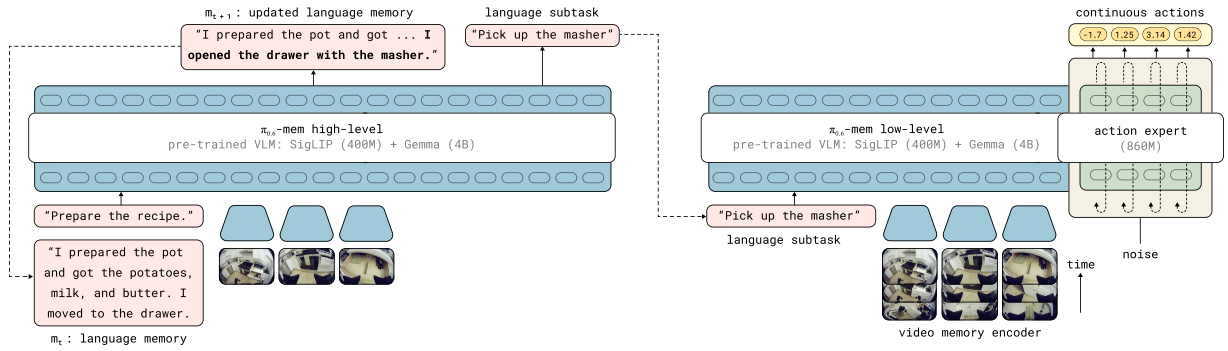


Figure 5.1: The MEM memory system equips VLAs like $\pi_{0.6}$ with long-horizon memory via two key components: (1) a high-level policy is trained to keep track of **long-horizon** semantic events by updating a **language memory** m_t (left, section 5.3.2), (2) the low-level policy uses a **short-horizon** observation-based memory that is efficiently encoded via a **video encoder** (right, section 5.3.3).

where we separated the probability of actions into a low-level policy π_{LL} and high-level policy π_{HL}. The low-level policy models sequences of actions conditioned on the task goal g, a shorter sequence of observations ($K \ll T$) and a subtask instruction l_{t+1}. The subtask instruction, in turn, is generated by the high-level policy, which is not only conditioned on the task goal, but also a summarization m_t of previous semantic events in natural language. We call this summarization the *language memory* in the following. It allows us to significantly reduce the number $K \ll T$ of dense observations that are input to the model, without sacrificing the ability to capture memory on the order of many minutes. While previous work has employed a similar split into high- and low-level policies with subtask instructions as the interface, the key novelty here is that π_{HL} additionally predicts the updated language memory m_{t+1} based on its own previous prediction m_t.

5.3.2 Language Memory for Long-Term Memory

The language memory m_t is a summary of previous *semantic* events that happened while the policy is solving a task. The main idea is to train the high-level policy $\pi_{\text{HL}}(l_{t+1}, m_{t+1} | o_{t-K:t}, m_t, g)$ such that it predicts the relevant summary m_{t+1} from its previous summary m_t and current observations and task. This way, the model explicitly decides when and how to update its memory representation. For example, for a kitchen cleaning robot that already placed a plate in the cabinet and just finished picking up the next bowl, a memory update may look like the following:

$$m_t\text{: } \texttt{I placed a plate in the cabinet and moved to the counter.}$$

$$\Downarrow$$

$$m_{t+1}\text{: } \texttt{I placed a plate in the cabinet, moved to the counter, and picked up a bowl.}$$

In order to train π_{HL}, we need to obtain appropriate training data, including summarization actions we want the high-level policy to take. For this, we develop a pipeline to generate training data for the transition between m_t and m_{t+1} as follows. Given a robot episode with subtask language annotations $l_{0:T}$, we pass the subtask instructions together with an indicator whether the subtask execution failed or

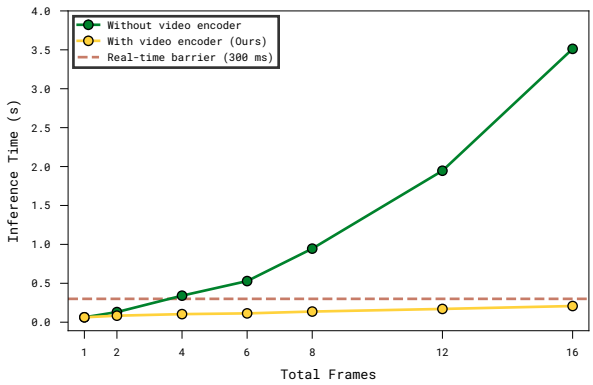


Figure 5.2: Naively passing a sequence of observations into the backbone of a VLA rapidly increases inference latency. Our efficient video encoder architecture allows us to use many observation frames while remaining under critical real-time inference thresholds [15, 16]. Timings measured for the $\pi_{0.6}$ VLA, with four input camera streams on one NVIDIA H100 GPU.

succeeded to an off-the-shelf pre-trained language model (LLM) and ask it to summarize all information from the previous subtasks that is still relevant for future task execution. We collect the corresponding output and use it to label the sequence for training the high-level policy.

Importantly, we instruct the LLM to *remove* or *compress* information in the language memory whenever appropriate. For example, instead of remembering the precise attributes of all objects that were manipulated ("`I put a light green bowl, a dark blue bowl and a bright yellow bowl into the top right cabinet`"), it is often sufficient to just remember where the bowls were placed ("`I placed three bowls in the top right cabinet`"). Compressing the language memory helps keep it succinct (and thus inference fast) and reduces the potential for train-inference distribution shift, since fewer bits of information are carried over between timesteps. We find that current generation LLMs are effective at this task when prompted to keep only the minimal set of relevant information.

5.3.3 Video Encoder for Dense Short-Term Visual Memory

The language memory mechanism, as described in the last section, can capture long-term *semantic* concepts. In order to enable our policies to reason about fine-grained details, dynamics, and resolve self-occlusion, access to a dense sequence of previous observations $o_{t-K:t}$ is required as input to the policy. In most VLAs, encoding the observations, especially for images, accounts for the largest fraction of compute spent during training and significantly impacts inference speed. Therefore, as we show in fig. 5.2, simply encoding a sequence of observations one-by-one and passing them into the VLA backbone quickly becomes infeasible as the context grows beyond a few timesteps. The VLA will rapidly incur inference latencies beyond what prior work has deemed acceptable to prevent significant degradation in on-robot performance for dexterous manipulation tasks [15, 16].

To address this, we propose to use a video encoder that can compress frames in the time dimension before passing the encoding into the VLM backbone. Our video encoder extends Vision Transformers (ViTs) [49], typically used as the vision encoder in most VLAs, to encode video inputs. Importantly, our encoder leaves the single-image performance of the VLM we start with invariant, and thus can be used to endow any pre-trained VLM with highly effective (yet computationally cheap) visual memory

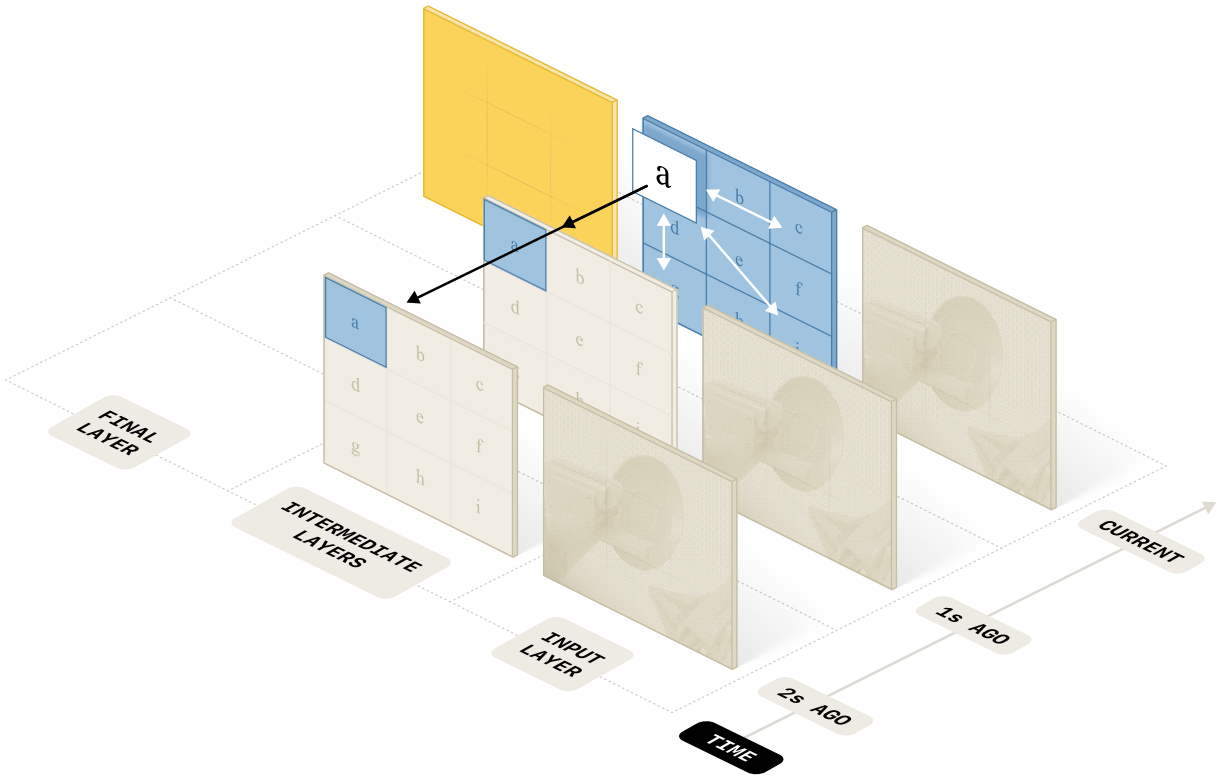


Figure 5.3: We propose to use an efficient video encoder architecture for compressing short-horizon, image-based memory. Our architecture expands standard ViTs for encoding video inputs by interleaving layers that apply bidirectional spatial attention *within* each observation (white arrows) with layers that additionally apply causal-temporal attention operations *across* observations (black arrows). We drop observation tokens for past timesteps in upper layers of the ViT to compress the inputs and reduce the number of tokens passed to the VLA backbone.

capabilities.

ViTs process images in patches and repeatedly perform bidirectional attention across the patch embeddings (across multiple layers) to derive a final set of output embeddings. We keep the same general structure and our video encoder first patchifies all images in the input video separately. Inspired by existing work on space-time separable attention for video understanding (see e.g. Bertasius et al. [9] for an overview) we then modify the attention mechanism every 4-th layer of the ViT to incorporate both spatial (as is standard in ViTs) but also temporal context. To avoid prohibitively expensive joint attention operations over the very large number of total patches in time and space, our architecture factorizes attention into separate *spatial* and *temporal* attention operations. Every 4-th layer additively adds attention over the time dimension by performing attention over the timesteps' representations for *the same* image patch using a causal attention mask ("temporal") – see fig. 5.3 for a visual depiction and Appendix 8.4.3 for a mathematical description. This reduces the computational complexity of the corresponding attention in each layer from $\mathcal{O}(n^2K^2)$ (for naive attention over both time and space) to $\mathcal{O}(Kn^2 + nK^2)$. Finally, to reduce the number of patches processed by the subsequent VLA transformer backbone, we only pass the representation computed for the current timestep onwards (dropping representations for all patches from past timesteps). Thus, our video encoder *matches* the number of tokens typically passed into the VLA backbone in single-timestep VLAs without memory; and we effectively force the video-encoder to incorporate temporal information into the representation produced for the current observation (via the modified attention mechanism).

A key property of our video encoder is that it *does not* introduce new learnable parameters compared to standard, single-image ViTs. Video encoding capabilities are added by modifying the *attention pattern* of the ViT and adding a fixed sinusoidal temporal position encoding. Thus, we can initialize the weights of our video encoder from the pre-trained ViT weights of any standard vision-language-models, just like in no-memory VLAs. To maximize feature transfer, we ensure that for $K = 1$, i.e., single-image input, our encoder's initialization *exactly* matches that of the VLM, which we achieve with a sinusoidal temporal position embedding that has value 0 for $t = 0$.

In summary, our video encoder architecture allows us to effectively scale observation-based memory to tens of seconds without incurring prohibitive computational overhead during training or inference (fig. 5.2), while at the same time permitting initialization from pre-trained vision-language model weights.

5.3.4 Integrating MEM into the $\pi_{0.6}$ VLA

For our experimental evaluation, we equip the $\pi_{0.6}$ VLA [157] with MEM memory by adapting its architecture to support the video encoder from section 5.3.3. Like $\pi_{0.6}$, the $\pi_{0.6}$ VLA with MEM is initialized from a pre-trained Gemma3-4B VLM [93]. Following Driess et al. [51], the model is trained with both discrete FAST action token prediction [155] and a flow-matching action expert [80] with 860M parameters. Gradients don't flow from the action expert into the VLM backbone [51]. The model is trained with an input resolution of 448x448 px per camera stream and up to four camera streams (depending on the robot embodiment).

In addition to past camera frames, the observation memory of the $\pi_{0.6}$ model with MEM memory also contains past proprioceptive states, e.g., joint angles. The $\pi_{0.6}$ VLA represents the robot's state in text form. However, in the case of a *sequence* of past states, this would quickly lead to a very large number

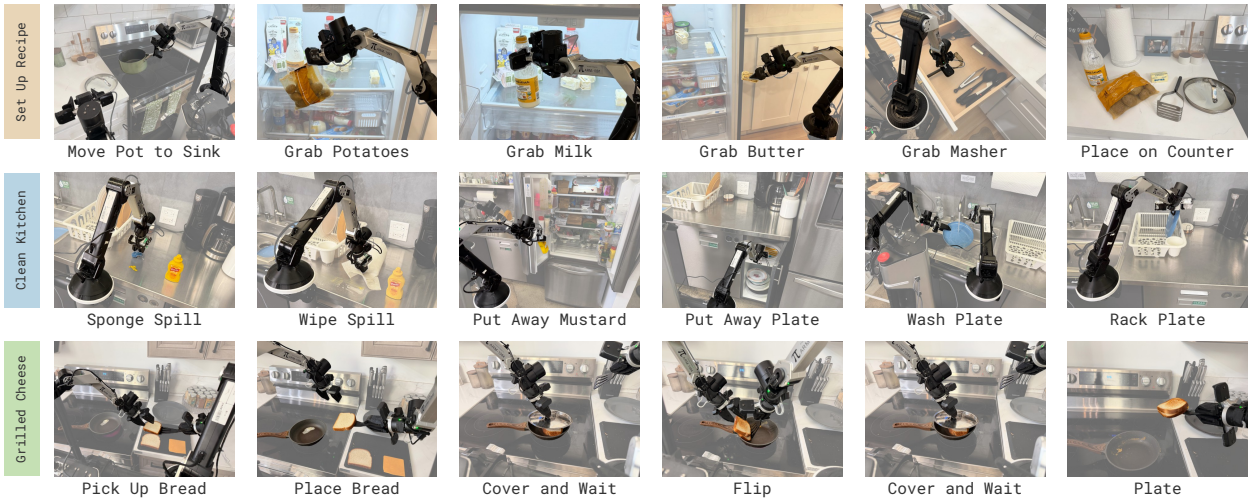


Figure 5.4: We test MEM policies across multiple challenging, long-horizon dexterous manipulation tasks that require retaining memory for up to fifteen minutes, including setting up a recipe, cleaning up a kitchen (section 5.4.1), and making a grilled cheese sandwich (section 5.4.3).

of state text tokens. To prevent this, we instead use a continuous state embedding by projecting each proprioceptive state into the backbone embedding space with a linear projection. This way, we only produce K proprioceptive state tokens for an observation memory of length K.

We pre-train $\pi_{0.6}$-MEM on a diverse mixture of data that includes teleoperated robot demonstrations, policy rollout data, and human corrections similarly to Physical Intelligence et al. [157], vision-language tasks, and *video*-language tasks, such as video captioning. During pre-training, we train the model with a sequence of six observations (five past and the current observation), spaced with a stride of one second. During post-training, we find that we can flexibly expand this horizon and accommodate much longer observation-based memory (up to 18 frames and 54 seconds of observation-based memory in our experiments), similarly to what has been observed in language model and vision-language model training [31]. We run all on-robot experiments using either inference-time real-time chunking (RTC, [15]) or training-time RTC [16] for asynchronous real-time inference.

5.4 Experimental Evaluation

Our experimental evaluation aims to answer the following questions: (1) Does MEM enable VLAs to perform tasks that require long-term memory spanning up to 15 minutes? (2) Does MEM unlock new capabilities in VLA models, like in-context adaptation of manipulation strategies, based on short-term memory of recent failures? (3) How does the performance of MEM compare to prior approaches for adding memory to VLA models?

5.4.1 MEM Solves Tasks Requiring Long-Horizon Memory

We evaluate our method's potential for enabling long-horizon tasks using two challenging scenarios that require retaining memory for up to fifteen minutes (see fig. 5.4, rows 1 & 2): (1) **Recipe setup**: The robot

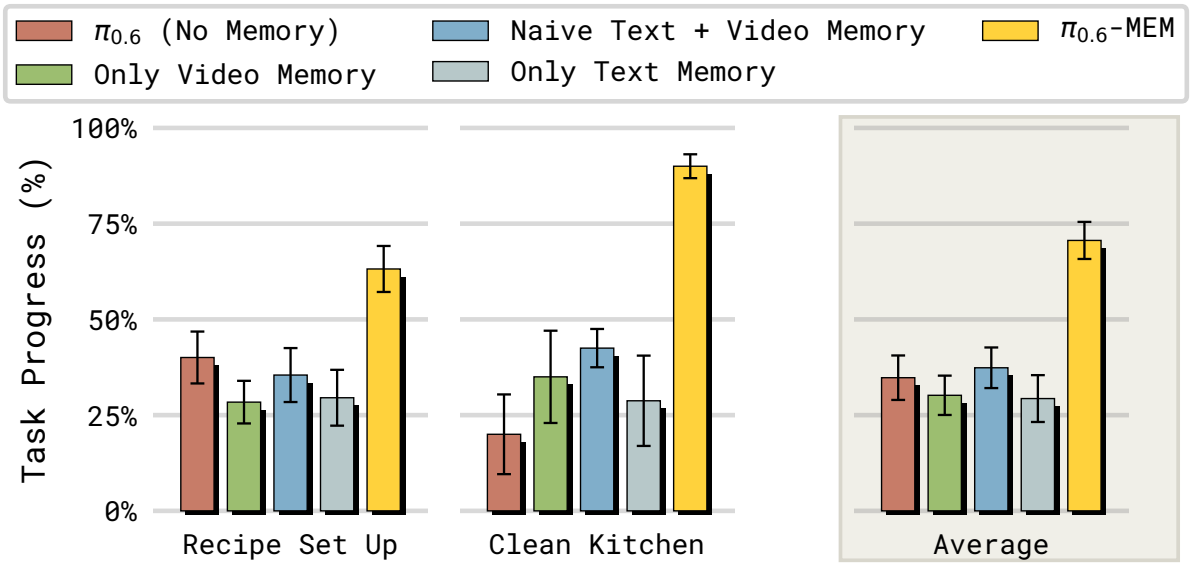


Figure 5.5: Performance of policies on challenging, long-horizon manipulation tasks. Without memory, even state-of-the-art generalist policies like $\pi_{0.6}$ struggle to perform such tasks. We ablate the memory components and show these tasks are solvable by combining short-horizon, observation-based memory, with long-horizon, language-based memory. Naive memory of past language instructions, without compression, struggles with training-inference distribution shifts.

is given a detailed prompt specifying the ingredients and cookware required to cook a recipe and their location, and is asked to fetch all of them from various cabinets, drawers, or, e.g., the fridge, and place them at a specific location, e.g., on the stove or a specific countertop. Memory is required to keep track of all items that have already been assembled and remember to close drawers, cabinets, or the fridge once the task is complete. We train on 42 recipes across diverse kitchen scenes, and evaluate on five of the recipes in unseen kitchens and with unseen objects. (2) **Clean up kitchen**: The robot needs to clean up a messy kitchen environment, including stowing objects in the fridge, wiping countertops, washing dishes with soap and running water, and placing dishes in the drying rack. Memory is required for remembering steps in the cleanup process (e.g., whether soap has been added to the dishes before washing, whether both front and back of the dish have been washed), remembering which surfaces have already been cleaned, and remembering to close cabinets after objects have been stowed. This task is not only challenging from a memory perspective, but also requires policies to perform multiple dexterous manipulations, from wiping surfaces to carefully handling water and soap. For more details on training data, evaluation conditions, and scoring rubrics for progress scores, see section 8.4.2. All evaluations use 10 rollouts per policy and task or recipe, and we report mean $\pm$ standard error in all graphs.

fig. 5.5 shows the results: without memory, even state-of-the-art models like $\pi_{0.6}$ struggle to perform these challenging long-horizon tasks. While prior works have shown policies that perform manipulation over extended periods of time [157], solving real-world tasks with flexible subtask sequences and frequent partial observability fundamentally requires memory. The results show that MEM is effective at providing the required context to the policy across both short and long-horizon time intervals, and increases policy success rate significantly.

To better understand what contributes to $\pi_{0.6}$-MEM's strong performance, we perform a detailed ablation study of the different components of our approach. We report the results in fig. 5.5. We compare to versions of our policy that *remove* short-horizon video memory or long-horizon language memory, and show that both are essential to solving the challenge tasks. Without video memory, the robot may struggle to understand how long it has been washing a plate or wiping a surface, and get "stuck" indefinitely. Video memory also helps robustify manipulation tasks via in-context adaptation (see section 5.4.2). Long-horizon language memory, on the other hand, is essential for remembering semantic events in the more distant past. It allows the policy to keep track of steps in the recipe, or remember to close doors that it has opened. We also evaluate a version of our policy that removes the *compression* applied to the language-based memory (see section 5.3.2): instead of training the model to compress and discard information that is no longer needed, we simply concatenate all previous subtask instructions up to a maximum length in the input of the high-level policy. We find that this type of "naive" language memory works significantly worse than our model-predicted summaries. The core challenge with "naive" language memory is a large train-inference distribution shift: during training, most episodes utter any given subtask instruction only once (e.g., "`pick up bowl`" $\rightarrow$ "`place bowl in cabinet`") since they are typically near-optimal human demonstrations. Yet, during inference time policies may repeatedly fail on a given subtask, causing the high-level policy to repeatedly produce the same subtask before finally succeeding and moving on ("`pick up bowl`" $\rightarrow$ "`pick up bowl`" $\rightarrow$ "`pick up bowl`" $\rightarrow$ "`place bowl in cabinet`"), leading to a distribution shift that can degrade overall policy performance. In contrast, MEM's language memory mechanism would simply not update the memory representation until the bowl was successfully picked up. This *compression* of context (e.g., discarding failed attempts) thus reduces distribution shift and improves overall performance.

5.4.2 In-Context Adaptation of Manipulation Strategies

In the previous section, we showed that MEM can enable VLAs to solve very long-horizon tasks. In this section, we investigate whether equipping VLAs with memory can unlock improved performance, even on shorter-horizon tasks that at first glance may not require memory. Intuitively, while memory across tens of minutes is useful for keeping track of overall task progress, short-horizon memory can enable policies to *adapt* their behavior *in-context* and intelligently react to mistakes: instead of failing in the same way over and over, policies can use context of previous failed attempts to, for example, modify the way in which they are trying to pick up an object, or how they open a door.

To test whether MEM unlocks such in-context adaptation, we set up two tasks on which current state-of-the-art VLAs like $\pi_{0.6}$ struggle (fig. 5.6): picking up flat objects like chopsticks with an out-of-distribution table height, which leads to frequent mis-grasps, and opening fridges where the direction the door opens is unclear, resulting in repeated failed opening attempts.

To teach policies the in-context adaptation strategy, we follow [157] and collect targeted human feedback: after the policy fails, a human intervenes and provides a demonstration of the corrected manipulation strategy, like adjusting grasp height for picking up the chopstick. For the fridge-opening task, we collect exploration rollouts in which the demonstrator initially does not know the opening mechanism, so the data naturally contains both the failed attempt and the subsequent corrective demonstration. We then simply finetune the $\pi_{0.6}$-MEM policy with this correction data, keeping the failed attempt that preceded the correction in the short-term memory of the model during training. As a result, the model learns to adapt

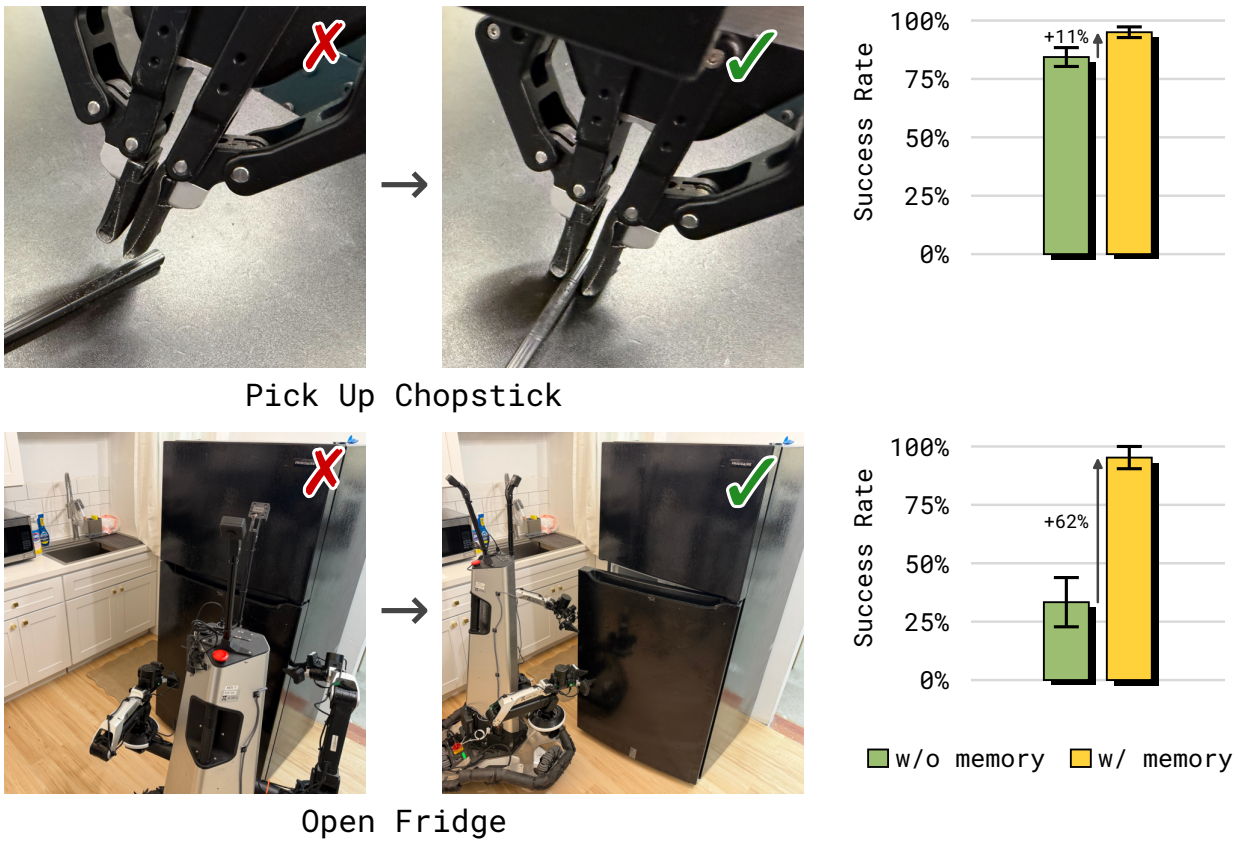


Figure 5.6: VLAs with memory can perform in-context adaptation of manipulation strategies, like adjusting grasp height or door opening direction. Without memory, policies get stuck with a suboptimal strategy.

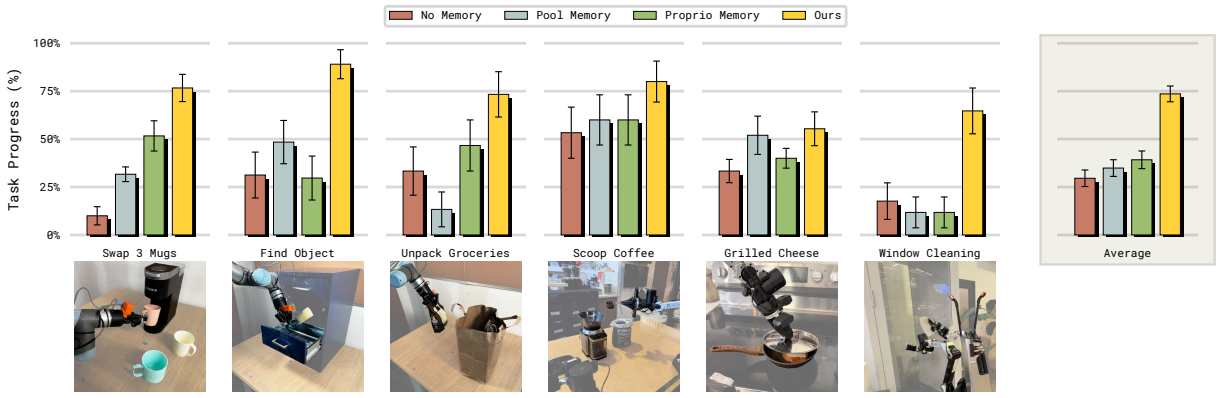


Figure 5.7: Comparison of approaches for equipping VLAs with memory across core memory capabilities (handling partial observability, counting, visual memory). VLAs without memory struggle to perform these tasks. Only MEM's memory approach performs well across all core capabilities.

its manipulation strategy when it sees a mistake in its short-term memory. We compare to finetuning the memoryless $\pi_{0.6}$ policy on the same intervention data.

The results in fig. 5.6 show that the MEM-VLA with memory is much more effective at leveraging the corrections, and learns to adapt its manipulation strategy on the fly. The policy without memory has no way to remember which strategy was attempted before, and therefore cannot intelligently change the strategy after a mistake. In contrast, policies with memory can use the context to understand which strategy has already been tried and failed, and adjust accordingly.

5.4.3 Analysis Experiments

Tasks. To compare our method to existing approaches that equip VLAs with memory across a wide range of capabilities, we develop a suite of challenging manipulation tasks that measure the ability of policies to use memory efficiently, and perform dexterous manipulations (see figs. 5.7 and 5.9 for task visualizations). The tasks span a diverse set of scenarios involving single-arm, dual-arm, and mobile robots. They test for core memory capabilities, like handling **partial observability** (e.g., remembering which drawer an object was placed in by a human; unpacking a grocery bag and remembering whether objects are left to unpack; placing and removing multiple mugs under a coffee machine), **counting** (e.g., counting scoops of coffee to add to a coffee grinder), timing (e.g., remembering how long a grilled cheese sandwich has been cooking, see fig. 5.4), and **spatial memory** (e.g., remembering which parts of a window have already been wiped). They also require policies to perform **precise manipulations** (e.g., folding a stack of laundry or assembling a cardboard box). They are thus well-suited to test the limits of existing memory approaches and also compare to the performance of state-of-the-art VLAs without memory. Following prior work [80], we collect small post-training datasets for the most challenging manipulation tasks and for the memory tasks, and evaluate policies out-of-the-box for all remaining tasks (fig. 5.9). We provide detailed descriptions of all task setups in section 8.4.2.

Comparisons. A number of prior works leverage observation-based (shorter-horizon) memory, and address its computational challenges through aggressive compression of observations from past timesteps.

We compare our approach to two representative prior works to understand the tradeoffs of different memory representations: (1) **Pool Memory** akin to Jang et al. [86], we compress all past observations into a single "memory token" using average pooling. We implement it by encoding each of the past observations separately with the pre-trained single-frame ViT, and then applying an average pooling operation on the frame encodings before passing them into the VLA backbone. The current timestep's observation is separately encoded and passed into the VLA without pooling; (2) **Proprio Memory** conditions on a history of low-dimensional robot states only, akin to [229], to avoid the computational cost of high-dimensional image memory. All proprioceptive states from past timesteps are separately passed through a linear projection, and then fed into the VLA backbone. To ensure fair comparison, we re-implement all approaches on top of the $\pi_{0.6}$ VLA backbone and use a version of our model *without* language-based long-horizon memory. To test the performance of our policy on the most dexterous tasks, we also compare to $\pi_{0.6}$ *without* memory, a state-of-the-art VLA that has shown strong results on complex manipulation tasks [156]. Finally, to better understand the effect of *pre-training* on memory efficacy, we compare to **MEM-Posttrain-Only**, an ablation of our method that introduces the video encoder only during *post-training* on the target task, starting from the pre-trained $\pi_{0.6}$ checkpoint, akin to Li et al. [115].

We report results in fig. 5.7. We first compare the different memory approaches on tasks that test core memory capabilities (fig. 5.7). As expected, the $\pi_{0.6}$ VLA *without* memory struggles on these tasks and often has to resort to random chance, e.g., when picking which of four drawers to open to find the object hidden inside (25% success) or whether or not to add another scoop of coffee (50% success). In contrast, existing memory solutions improve performance, particularly on simpler memory tasks that only require a few bits of memory, e.g., remembering how many scoops of coffee have already been added to the hopper. We find that Pool-Memory's aggressive observation compression via average-pooling can lead it to struggle particularly on tasks that require longer-term memory, like remembering the past positions of multiple coffee mugs or recalling how many objects are left to unpack in a grocery bag. Proprio-Memory on the other hand is only effective in tasks where the robot needs to remember *its own* state, but struggles in scenarios where remembering the *environment* state is necessary: e.g., recalling which drawer contains a particular object.

In comparison, the MEM VLA is the only model that achieves strong performance across *all* tested memory capabilities. It can reliably handle partial observability challenges like remembering the location of objects, and also leverage these capabilities in dexterous tasks like unpacking a bag of groceries. Notably, pre-training the observation-based memory on a diverse data mix of robot and non-robot data significantly improves the ability of the model to leverage its memory – even when the memory horizon is significantly expanded during post-training (e.g., from 5 seconds in pre-training to up to 1 minute in post-training). The version of our model that only introduces memory during post-training is noticeably worse at leveraging the information from past timesteps (see fig. 5.8). We emphasize that this model is still initialized from a base $\pi_{0.6}$ checkpoint pre-trained on the same diverse robot data, but did not develop memory capabilities during this pre-training. Finally, the head-to-head comparison of Ours (post-train only) vs. Pool-Memory demonstrates that even without pre-training, our video encoder design is a more effective approach for extracting information from prior timesteps than separately encoding each timestep and average pooling. By interleaving temporal attention operations throughout the layers of the ViT, our video encoder can more effectively extract and compress the relevant information.

In addition to testing core memory capabilities, we also test the performance of MEM across a range of challenging dexterous manipulation tasks (fig. 5.9). We find that the MEM VLA not only makes effective

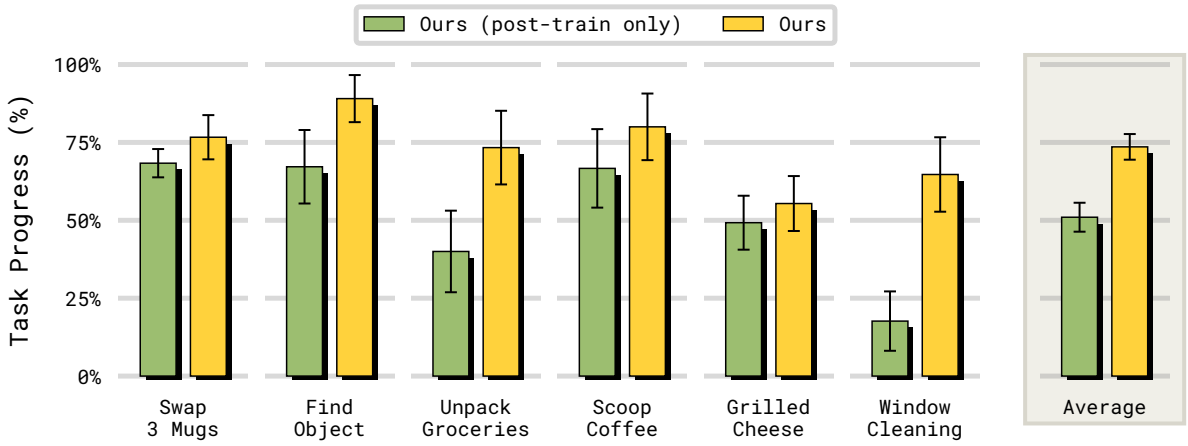


Figure 5.8: Pre-training MEM on a diverse dataset of robot and non-robot video data makes it more effective at using its memory to solve diverse manipulation tasks. Introducing memory during post-training only results in substantially lower performance.

use of memory in tasks that require it, but it can *also* match the state-of-the-art performance of the $\pi_{0.6}$ VLA across challenging dexterous manipulation tasks. This is notable, since numerous previous works have reported degradations in performance from adding memory to policies, e.g., due to causal confusion [39, 233]. We attribute MEM's strong performance in large parts to our diverse pre-training data mixture, which contains episodes of varying optimality, speed, and control frequency, in addition to diverse internet videos. Together, this diversity can prevent spurious correlations in the video encoder that may arise when training on smaller, more uniform robot datasets, and lead to the brittleness and performance degradation of policies with memory reported in prior work.

5.5 Discussion

We presented Multi-Scale Embodied Memory (MEM), an approach for equipping VLAs with long-horizon memory. MEM enables VLAs to perform long-horizon tasks that require tens of minutes of memory, while obeying real-world latency constraints. MEM also enables in-context adaptation of manipulation strategies and, when combined with the $\pi_{0.6}$ VLA, achieves state-of-the-art performance across a wide range of manipulation tasks. To achieve this, MEM introduces a mixed-modal memory architecture that combines short-horizon, video-based memory, with a long-horizon, language-based memory mechanism. We believe that MEM is only the first step towards building robot policies that can effectively manage very long-horizon memory. Future work can explore how we can scale memory to last beyond the horizon of a single episode, to span weeks, months, or years of deployment, and allow us to build robots that learn continually at deployment time.

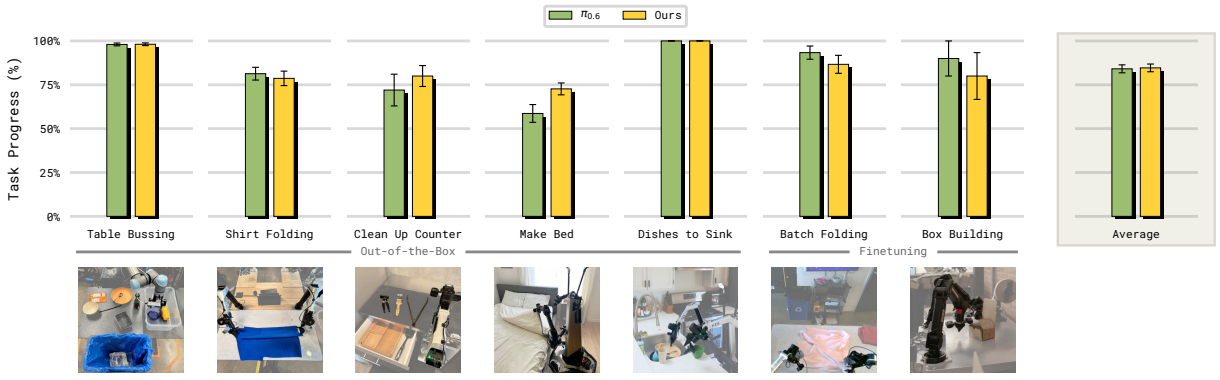


Figure 5.9: In addition to effectively using its memory, MEM also matches the performance of state-of-the-art non-memory VLAs across challenging manipulation tasks that do not require memory.

Chapter 6

Conclusion

In this thesis, we addressed the challenges involved in training a generalist robot policy through large-scale imitation learning. First, we tackled collecting the pre-training dataset. We found it critical to ensure that every skill was collected in many environments and each environment afforded many skills to enable generalizable behaviors and a steerable policy. We employed these techniques to collect BridgeData V2, a dataset that has been invaluable in further research on generalist policies. Next, we used BridgeData V2, along with many other datasets released in the Open X-Embodiment project, to study the design of a generalist policy architecture. We found that we could achieve a high-capacity, expressive, and finetune-able model while maintaining fast inference through a modular design, consisting of observation tokenizers, a large transformer backbone, and a diffusion action head. Our model, Octo, outperformed prior generalist policies and from-scratch baselines when evaluated zero-shot and finetuned. However, Octo was only pre-trained on single robot arms. With CrossFormer we explored whether we could train a single policy to control multiple robot embodiments, including bimanual robot arms, wheeled robots, and quadrupeds. We designed a model that supported variable observation spaces and action spaces during pre-training and maximized parameter sharing across embodiments. CrossFormer performed as well as or better than policies trained on data from a single robot, indicating no negative transfer and some promising signs of positive transfer across embodiments. Finally, we tackled a key component missing from both Octo and CrossFormer: memory. We designed a two-part memory system consisting of a textual representation to capture longer horizon, high-level history and an efficient video encoding to capture shorter horizon, low-level history. Our system, MEM, enabled tasks requiring anywhere from a few seconds to many minutes of observation history.

We have outlined a promising path towards building a generally intelligent robot by continuing to scale the breadth and depth of robot teleoperation data and fitting ever larger models to this data via imitation learning. However, there are a number of remaining challenges that are not likely be solved entirely by scaling this approach. First, while the work in this thesis demonstrates impressive generalization of skills across objects and environments, current robot models struggle to generalize to tasks requiring entirely new behaviors. For example, while our robot models may be able to fold a new type of shirt when trained on ample shirt folding data, they cannot open the lid of a washing machine if lid opening is not present in their training data. Current robot models lack the "physical common sense" necessary to perform tasks requiring entirely new motions. Secondly, the performance of our robot models is still too low for them to be useful in practical settings. For example, robot models trained entirely with imitation learning may

achieve 60% success on folding shirts, but we need 99% success to feasibly deploy the robot in real homes.

In principle, both of these challenges could be addressed by collecting a greater breadth and depth of teleoperation data. With enough demonstrations of enough tasks, the model should start generalizing to entirely new behaviors, and with enough demonstrations of a single task, the model should almost never fail. However, given that teleoperation is time-consuming and expensive, it is unreasonable to expect we could collect enough data to fully address both of these issues. The space of possible robot behaviors is vast, and achieving near perfect success on a task with imitation learning is notoriously difficult [170]. Solving these challenges will potentially require entirely new data collection techniques and learning approaches.

One promising research direction is alternative data sources beyond teleoperation. Data collection with handheld grippers or even human hands is significantly cheaper and faster. If we can reliably track the positions of the grippers or hands to recover actions, this data could significantly augment teleoperation data [38, 234] and enable enough coverage to achieve broader generalization over behaviors. Another promising direction is reinforcement learning. Reinforcement learning allows the model to learn from its mistakes by training on its own sub-optimal rollouts, minimizing the distribution shift between training and deployment, and potentially pushing the performance of the model to near 100% [82]. While significant work remains in the pursuit of generally intelligent robots, we anticipate that the ideas in this thesis will form a strong foundation for further research.

Chapter 7

References

[1] Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to Poke by Poking: Experiential Learning of Intuitive Physics. In *Advances in Neural Information Processing Systems*, 2016. URL `http://arxiv.org/abs/1606.07419`.

[2] Scale AI. Introducing scale's automotive foundation model, 2023. URL `https://scale.com/blog/afm1`.

[3] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *NeurIPS*, 2017.

[4] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. ViViT: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6836–6846, 2021.

[5] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Human-to-robot imitation in the wild. *arXiv preprint arXiv:2207.09450*, 2022.

[6] Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from human videos as a versatile representation for robotics. In *CVPR*, 2023.

[7] Suneel Belkhale and Dorsa Sadigh. Minivla: A better vla with a smaller footprint, 2024. URL `https://github.com/Stanford-ILIAD/openvla-mini`.

[8] Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Hydra: Hybrid robot actions for imitation learning. *arxiv*, 2023.

[9] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *International Conference on Machine Learning (ICML)*, volume 2, page 4, 2021.

[10] Homanga Bharadhwaj, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. *arXiv preprint arXiv:2309.01918*, 2023.

[11] Homanga Bharadhwaj, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. Roboagent: Towards sample efficient robot manipulation with semantic augmentations and action chunking. *arxiv*, 2023.

[12] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Groot n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[13] Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. *arXiv preprint arXiv:2310.10639*, 2023.

[14] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[15] Kevin Black, Manuel Y Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *arXiv preprint arXiv:2506.07339*, 2025.

[16] Kevin Black, Allen Z Ren, Michael Equi, and Sergey Levine. Training-time action conditioning for efficient real-time chunking. *arXiv preprint arXiv:2512.05964*, 2025.

[17] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* , 2016.

[18] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2022.

[19] Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauza,

Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

[20] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[21] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alex Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *arXiv preprint arXiv:2307.15818*, 2023.

[22] Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, pages 287–318. PMLR, 2023.

[23] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[24] Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, Oleg Sushkov, David Barker, Jonathan Scholz, Misha Denil, Nando de Freitas, and Ziyu Wang. Scaling data-driven robotics with reward sketching and batch reinforcement learning. *arXiv preprint arXiv:1909.12200*, 2019.

[25] Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, et al. Scaling Data-Driven Robotics With Reward Sketching and Batch Reinforcement Learning. *arXiv preprint arXiv:1909.12200*, 2019.

[26] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[27] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder for english. In *Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations*, pages 169–174, 2018.

[28] Yevgen Chebotar, Karol Hausman, Yao Lu, Ted Xiao, Dmitry Kalashnikov, Jake Varley, Alex Irpan, Benjamin Eysenbach, Ryan Julian, Chelsea Finn, and Sergey Levine. Actionable Models: Unsupervised Offline Reinforcement Learning of Robotic Skills. 4 2021.

[29] Jingjing Chen, Hongjie Fang, Chenxi Wang, Shiquan Wang, and Cewu Lu. History-aware visuomotor policy learning via point tracking. *arXiv preprint arXiv:2509.17141*, 2025.

[30] Lawrence Yunliang Chen, Simeon Adebola, and Ken Goldberg. Berkeley UR5 demonstration dataset. `https://sites.google.com/view/berkeley-ur5/home`.

[31] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

[32] Tao Chen, Adithyavairavan Murali, and Abhinav Gupta. Hardware conditioned policies for multi-robot transfer learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[33] William Chen, Oier Mees, Aviral Kumar, and Sergey Levine. Vision-language models provide promptable representations for reinforcement learning. *arXiv preprint arXiv:2402.02651*, 2024.

[34] Yu Chen, Yingfeng Chen, Zhipeng Hu, Tianpei Yang, Changjie Fan, Yang Yu, and Jianye Hao. Learning action-transferable policy with action embedding. *arXiv preprint arXiv:1909.02291*, 2019.

[35] Zoey Chen, Sho Kiami, Abhishek Gupta, and Vikash Kumar. Genaug: Retargeting behaviors to unseen situations via generative augmentation. *arXiv preprint arXiv:2302.06671*, 2023.

[36] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137* , 2023.

[37] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[38] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[39] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2025.

[40] Nhat Chung, Taisei Hanyu, Toan Nguyen, Huy Le, Frederick Bumgarner, Duy Minh Ho Nguyen, Khoa Vo, Kashu Yamazaki, Chase Rainwater, Tung Kieu, et al. Rethinking progression of memory state in robotic manipulation: An object-centric perspective. *arXiv preprint arXiv:2511.11478*, 2025.

[41] Zichen Jeff Cui, Yibin Wang, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. From play to policy: Conditional behavior generation from uncurated robot data, 2022.

[42] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

[43] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. In *Conference on Robot Learning*, pages 885–897. PMLR, 2020.

[44] Sudeep Dasari, Oier Mees, Sebastian Zhao, Mohan Kumar Srirama, and Sergey Levine. The ingredients for robotic diffusion transformers. *arxiv TBD*, 2024.

[45] Shivin Dass, Jullian Yapeter, Jesse Zhang, Jiahui Zhang, Karl Pertsch, Stefanos Nikolaidis, and Joseph J. Lim. CLVR jaco play dataset, 2023. URL `https://github.com/clvrai/clvr_jaco_play_dataset`.

[46] Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *NeurIPS*, 2019.

[47] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2169–2176. IEEE, 2017.

[48] Ria Doshi, Homer Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. In *Conference on Robot Learning*, 2024.

[49] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[50] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[51] Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *arXiv preprint arXiv:2505.23705*, 2025.

[52] Maximilian Du, Suraj Nair, Dorsa Sadigh, and Chelsea Finn. Behavior retrieval: Few-shot imitation learning by querying unlabeled datasets. *ArXiv*, abs/2304.08742, 2023. URL `https://api.semanticscholar.org/CorpusID:258186973`.

[53] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[54] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

[55] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=tc5qisoB-C`.

[56] Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

[57] Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Junbo Wang, Haoyi Zhu, and Cewu Lu. Rh20t: A robotic dataset for learning diverse skills in one-shot. In *RSS 2023 Workshop on Learning for Task and Motion Planning*, 2023.

[58] Haoquan Fang, Markus Grotz, Wilbert Pumacay, Yi Ru Wang, Dieter Fox, Ranjay Krishna, and Jiafei Duan. Sam2act: Integrating visual foundation model with a memory architecture for robotic manipulation. *arXiv preprint arXiv:2501.18564*, 2025.

[59] Gilbert Feng, Hongbo Zhang, Zhongyu Li, Xue Bin Peng, Bhuvan Basireddy, Linzhu Yue, Zhitao Song, Lizhi Yang, Yunhui Liu, Koushil Sreenath, et al. Genloco: Generalized locomotion controllers for quadrupedal robots. In *Conference on Robot Learning*, pages 1893–1903. PMLR, 2023.

[60] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[61] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *Advances in neural information processing systems*, 29, 2016.

[62] Justin Fu, Sergey Levine, and Pieter Abbeel. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4019–4026. IEEE, 2016.

[63] Zipeng Fu, Tony Z Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117* , 2024.

[64] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[65] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Gandhi, and Lerrel Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. 7 2018. URL `http://arxiv.org/abs/1807.07049`.

[66] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Prakashchand Gandhi, and Lerrel Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. *Advances in neural information processing systems*, 31, 2018.

[67] Huy Ha, Pete Florence, and Shuran Song. Scaling up and distilling down: Language-guided robot skill acquisition. In *Conference on Robot Learning*, pages 3766–3777. PMLR, 2023.

[68] Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

[69] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[70] Minho Heo, Youngwoon Lee, Doohyun Lee, and Joseph J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. In *Robotics: Science and Systems*, 2023.

[71] Noriaki Hirose, Amir Sadeghian, Marynel Vázquez, Patrick Goebel, and Silvio Savarese. Gonet: A semi-supervised deep learning approach for traversability estimation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3044–3051. IEEE, 2018.

[72] Jonathan Ho and Stefano Ermon. Generative Adversarial Imitation Learning. In *Advances in Neural Information Processing Systems*, 2016. URL `https://arxiv.org/pdf/1606.03476.pdf`.

[73] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[74] Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving, 2023.

[75] Edward S Hu, Kun Huang, Oleh Rybkin, and Dinesh Jayaraman. Know thyself: Transferable visual control policies through robot-awareness. *arXiv preprint arXiv:2107.09047*, 2021.

[76] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10608–10615. IEEE, 2023.

[77] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Audio visual language maps for robot navigation. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*, Chiang Mai, Thailand, 2023.

[78] Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pages 4455–4464. PMLR, 2020.

[79] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

[80] Physical Intelligence. π_0: A vision-language-action flow model for general robot control, 2024. URL `https://arxiv.org/abs/2410.24164`.

[81] Physical Intelligence. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL `https://arxiv.org/abs/2504.16054`.

[82] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

[83] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2): 3019–3026, 2020.

[84] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-z: Zero-shot task generalization with robotic imitation learning. In *5th Annual Conference on Robot Learning*, 2021. URL `https://openreview.net/forum?id=8kbp23tSGYv`.

[85] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

[86] Huiwon Jang, Sihyun Yu, Heeseung Kwon, Hojin Jeon, Younggyo Seo, and Jinwoo Shin. Contextvla: Vision-language-action model with amortized multi-frame context. *arXiv preprint arXiv:2510.04246*, 2025.

[87] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. VIMA: Robot manipulation with multimodal prompts. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 14975–15022. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/jiang23b.html`.

[88] Leslie Pack Kaelbling. Learning to Achieve Goals. In *IJCAI*, 1993.

[89] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[90] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation. In *Conference on Robot Learning*, pages 651–673, 2018.

[91] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale. 2021.

[92] Dmitry Kalashnikov, Jake Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Scaling up multi-task robotic reinforcement learning. In *5th Annual Conference on Robot Learning*, 2021.

[93] Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 4, 2025.

[94] Katie Kang, Gregory Kahn, and Sergey Levine. Hierarchically integrated models: Learning to navigate from heterogeneous robots. *arXiv preprint arXiv:2106.13280*, 2021.

[95] Haresh Karnan, Anirudh Nair, Xuesu Xiao, Garrett Warnell, Sören Pirk, Alexander Toshev, Justin Hart, Joydeep Biswas, and Peter Stone. Socially compliant navigation dataset (scand): A large-scale dataset of demonstrations for social navigation. *IEEE Robotics and Automation Letters*, 2022.

[96] Haresh Karnan, Anirudh Nair, Xuesu Xiao, Garrett Warnell, Sören Pirk, Alexander Toshev, Justin Hart, Joydeep Biswas, and Peter Stone. Socially compliant navigation dataset (scand): A large-scale dataset of demonstrations for social navigation. *IEEE Robotics and Automation Letters*, 7(4): 11807–11814, 2022.

[97] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[98] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[99] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *arXiv preprint*, 2024.

[100] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015.

[101] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, et al. Segment Anything, April 2023.

[102] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[103] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline Reinforcement Learning With Implicit Q-Learning. *arXiv preprint arXiv:2110.06169*, 2021.

[104] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[105] Aviral Kumar, Anikait Singh, Frederik Ebert, Yanlai Yang, Chelsea Finn, and Sergey Levine. Pre-training for robots: Offline rl enables learning new tasks from a handful of trials. *arXiv preprint arXiv:2210.05178*, 2022.

[106] Vikash Kumar, Rutav Shah, Gaoyue Zhou, Vincent Moens, Vittorio Caggiano, Abhishek Gupta, and Aravind Rajeswaran. Robohive: A unified framework for robot learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL `https://openreview.net/forum?id=0H5fRQcpQ7`.

[107] Teyun Kwon, Norman Di Palo, and Edward Johns. Language models as zero-shot trajectory generators. *arXiv preprint arXiv:2310.11604*, 2023.

[108] Thomas Lampe, Abbas Abdolmaleki, Sarah Bechtle, Sandy H Huang, Jost Tobias Springenberg, Michael Bloesch, Oliver Groth, Roland Hafner, Tim Hertweck, Michael Neunert, et al. Mastering stacking of diverse shapes with large-scale iterative reinforcement learning on real robots. *arXiv preprint arXiv:2312.11374*, 2023.

[109] Alex X. Lee, Coline Devin, Jost Tobias Springenberg, Yuxiang Zhou, Thomas Lampe, Abbas Abdolmaleki, and Konstantinos Bousmalis. How to spend your robot time: Bridging kickstarting and offline reinforcement learning for vision-based robotic manipulation. 2022.

[110] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

[111] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[112] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-End Training of Deep Visuomotor Policies. *Journal of Machine Learning Research (JMLR)*, 17(1):1334–1373, 2016. ISSN 15337928. doi: 10.1007/s13398-014-0173-7.2. URL `https://arxiv.org/pdf/1504.00702.pdf`.

[113] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. *International Journal of Robotics Research*, 2017.

[114] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018.

[115] Hao Li, Shuai Yang, Yilun Chen, Xinyi Chen, Xiaoda Yang, Yang Tian, Hanqing Wang, Tai Wang, Dahua Lin, Feng Zhao, et al. Cronusvla: Towards efficient and robust manipulation via multi-frame vision-language-action modeling. *AAAI*, 2025.

[116] Kunchang Li, Xinhao Li, Yi Wang, Yinan He, Yali Wang, Limin Wang, and Yu Qiao. Videomamba: State space model for efficient video understanding. In *European conference on computer vision*, pages 237–255. Springer, 2024.

[117] Fanqi Lin, Ruiqian Nai, Yingdong Hu, Jiacheng You, Junming Zhao, and Yang Gao. Onetwovla: A unified vision-language-action model with adaptive reasoning. *arXiv preprint arXiv:2505.11917* , 2025.

[118] Yixin Lin, Austin S. Wang, Giovanni Sutanto, Akshara Rai, and Franziska Meier. Polymetis. `https://facebookresearch.github.io/fairo/polymetis/`, 2021.

[119] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.

[120] Huihan Liu, Soroush Nasiriany, Lance Zhang, Zhiyao Bao, and Yuke Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. In *Robotics: Science and Systems (RSS)*, 2023.

[121] Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, et al. A comprehensive survey on long context language modeling. *arXiv preprint arXiv:2503.17407* , 2025.

[122] Antonio Loquercio, Ana I Maqueda, Carlos R Del-Blanco, and Davide Scaramuzza. Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 3(2):1088–1095, 2018.

[123] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

[124] Jianlan Luo, Charles Xu, Xinyang Geng, Gilbert Feng, Kuan Fang, Liam Tan, Stefan Schaal, and

Sergey Levine. Multi-stage cable routing through hierarchical imitation learning. *arXiv preprint arXiv:2307.08927* , 2023.

[125] Jianlan Luo, Charles Xu, Fangchen Liu, Liam Tan, Zipeng Lin, Jeffrey Wu, Pieter Abbeel, and Sergey Levine. Fmb: a functional manipulation benchmark for generalizable robotic learning. *arXiv preprint arXiv:2401.08553*, 2024.

[126] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. In *RSS*, 2021.

[127] Corey Lynch, Ayzaan Wahid, Jonathan Tompson, Tianli Ding, James Betker, Robert Baruch, Travis Armstrong, and Pete Florence. Interactive language: Talking to robots in real time. *IEEE Robotics and Automation Letters*, 2023.

[128] Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Yecheng Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Pieter Abbeel, Jitendra Malik, Dhruv Batra, Yixin Lin, Oleksandr Maksymets, Aravind Rajeswaran, and Franziska Meier. Where are we in the search for an artificial visual cortex for embodied intelligence? 2023.

[129] Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Yecheng Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Pieter Abbeel, Jitendra Malik, et al. Where are we in the search for an artificial visual cortex for embodied intelligence? *arXiv preprint arXiv:2303.18240*, 2023.

[130] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, Silvio Savarese, and Li Fei-Fei. RoboTurk: A crowdsourcing platform for robotic skill learning through imitation. *CoRR*, abs/1811.02790, 2018. URL `http://arxiv.org/abs/1811.02790`.

[131] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018.

[132] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *CoRL*, 2021.

[133] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

[134] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *7th Annual Conference on Robot Learning*, 2023.

[135] Max Sobol Mark, Jacky Liang, Maria Attarian, Chuyuan Fu, Debidatta Dwibedi, Dhruv Shah, and Aviral Kumar. Bpp: Long-context robot imitation learning by focusing on key history frames. *arXiv preprint arXiv:2602.15010*, 2026.

[136] David Q Mayne and Hannah Michalska. Receding horizon control of nonlinear systems. In *Proceedings of the 27th IEEE Conference on Decision and Control*, pages 464–465. IEEE, 1988.

[137] Oier Mees, Lukas Hermann, and Wolfram Burgard. What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters*, 7(4): 11205–11212, 2022.

[138] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022.

[139] Oier Mees, Jessica Borja-Diaz, and Wolfram Burgard. Grounding language with visual affordances over unstructured data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, London, UK, 2023.

[140] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Structured world models from human videos. *CoRL*, 2023.

[141] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021.

[142] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2146–2153. IEEE, 2017.

[143] Soroush Nasiriany, Tian Gao, Ajay Mandlekar, and Yuke Zhu. Learning and retrieval from prior data for skill-based imitation learning. In *Conference on Robot Learning (CoRL)*, 2022.

[144] Bolin Ni, Houwen Peng, Minghao Chen, Songyang Zhang, Gaofeng Meng, Jianlong Fu, Shiming Xiang, and Haibin Ling. Expanding language-image pretrained models for general video recognition. In *European conference on computer vision*, pages 1–18. Springer, 2022.

[145] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[146] NVIDIA. Groot n1: An open foundation model for generalist humanoid robots, 2025. URL `https://arxiv.org/abs/2503.14734`.

[147] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[148] Open X-Embodiment Collaboration, Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, Antonin Raffin, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Brian Ichter, Cewu Lu, Charles Xu, Chelsea Finn, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Chuer Pan, Chuyuan Fu, Coline Devin, Danny Driess, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Federico Ceola, Fei Xia, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Giulio Schiavi, Hao Su, Hao-Shu Fang, Haochen Shi, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer

Walke, Hongjie Fang, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jaehyung Kim, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jiajun Wu, Jialin Wu, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jitendra Malik, Jonathan Tompson, Jonathan Yang, Joseph J. Lim, João Silvério, Junhyek Han, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Zhang, Keyvan Majd, Krishan Rana, Krishnan Srinivasan, Lawrence Yunliang Chen, Lerrel Pinto, Liam Tan, Lionel Ott, Lisa Lee, Masayoshi Tomizuka, Maximilian Du, Michael Ahn, Mingtong Zhang, Mingyu Ding, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Pannag R Sanketi, Paul Wohlhart, Peng Xu, Pierre Sermanet, Priya Sundaresan, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Sherry Moore, Shikhar Bahl, Shivin Dass, Shuran Song, Sichun Xu, Siddhant Haldar, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Sudeep Dasari, Suneel Belkhale, Takayuki Osa, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xuanlin Li, Yao Lu, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yueh hua Wu, Yujin Tang, Yuke Zhu, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zhuo Xu, and Zichen Jeff Cui. Open X-Embodiment: Robotic learning datasets and RT-X models. `https://arxiv.org/abs/2310.08864`, 2023.

[149] OpenAI. GPT-4 Technical Report, March 2023.

[150] OpenAI. Gpt-4 technical report, 2024.

[151] Jyothish Pari, Nur Muhammad Shafiullah, Sridhar Pandian Arunachalam, and Lerrel Pinto. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021.

[152] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[153] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual Reasoning with a General Conditioning Layer, December 2017.

[154] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[155] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *Robotics: Science and Systems*, 2025.

[156] Physical Intelligence. $\pi_{0.6}$ model card. 2025. URL `https://website.pi-asset.com/pi06star/PI06_model_card.pdf`.

[157] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[158] Lerrel Pinto and Abhinav Gupta. Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours. *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[159] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016.

[160] Lerrel Pinto and Abhinav Gupta. Learning to push by grasping: Using multiple tasks for effective learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2161–2168. IEEE, 2017.

[161] Gabriel Quere, Annette Hagengruber, Maged Iskandar, Samuel Bustamante, Daniel Leidner, Freek Stulp, and Joern Vogel. Shared Control Templates for Assistive Robotics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, page 7, Paris, France, 2020.

[162] Ilija Radosavovic, Baifeng Shi, Letian Fu, Ken Goldberg, Trevor Darrell, and Jitendra Malik. Robot learning with sensorimotor pre-training. *Conference on Robot Learning*, 2023.

[163] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL `http://jmlr.org/papers/v21/20-074.html`.

[164] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3758–3765. IEEE, 2018.

[165] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

[166] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[167] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *Transactions on Machine Learning Research*, 2022.

[168] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models, April 2022.

[169] Erick Rosete-Beas, Oier Mees, Gabriel Kalweit, Joschka Boedecker, and Wolfram Burgard. Latent plans for task agnostic offline reinforcement learning. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.

[170] Stephane Ross and Drew Bagnell. Efficient reductions for imitation learning. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 661–668, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/ross10a.html.

[171] Gautam Salhotra, I-Chun Arthur Liu, and Gaurav S Sukhatme. Learning robot manipulation from cross-morphology demonstration. In *7th Annual Conference on Robot Learning*, 2023.

[172] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347, 2019.

[173] Saumya Saxena, Mohit Sharma, and Oliver Kroemer. Multi-resolution sensing for real-time control with vision-language models. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=WuBv9-IGDUA.

[174] Charles Schaff, David Yunis, Ayan Chakrabarti, and Matthew R Walter. Jointly learning to construct and control agents using deep reinforcement learning. In *2019 international conference on robotics and automation (ICRA)*, pages 9798–9805. IEEE, 2019.

[175] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal Value Function Approximators. In *International Conference on Machine Learning (ICML)*, pages 1312–1320, 2015. ISBN 9781510810587. URL http://proceedings.mlr.press/v37/schaul15.pdfhttp://jmlr.org/proceedings/papers/v37/schaul15.html.

[176] Milad Shafiee, Guillaume Bellegarda, and Auke Ijspeert. Manyquadrupeds: Learning a single locomotion policy for diverse quadruped robots. *arXiv preprint arXiv:2310.10486*, 2023.

[177] Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=agTr-vRQsa.

[178] Nur Muhammad Mahi Shafiullah, Anant Rai, Haritheja Etukuru, Yiqian Liu, Ishan Misra, Soumith Chintala, and Lerrel Pinto. On bringing robots home, 2023.

[179] Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. Gnm: A general navigation model to drive any robot. *arXiv preprint arXiv:2210.03370*, 2022.

[180] Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. Gnm: A general navigation model to drive any robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7226–7233. IEEE, 2023.

[181] Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. ViNT: A foundation model for visual navigation. In *7th Annual Conference on Robot Learning*, 2023. URL https://arxiv.org/abs/2306.14846.

[182] Rutav Shah, Roberto Martín-Martín, and Yuke Zhu. MUTEX: Learning unified policies from multimodal task specifications. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=PwqiqaaEzJ.

[183] Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In *Conference on robot learning*, pages 906–915. PMLR, 2018.

[184] Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025.

[185] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[186] Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. COG: Connecting New Skills to Past Experience with Offline Reinforcement Learning. In *Conference on Robot Learning (CoRL)*, 2020.

[187] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023.

[188] Laura Smith, Ilya Kostrikov, and Sergey Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*, 2022.

[189] Kihyuk Sohn, Xinchen Yan, and Honglak Lee. Learning Structured Output Representation using Deep Conditional Generative Models. In *Advances in Neural Information Processing Systems*, 2015.

[190] Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5(3):4978–4985, 2020.

[191] Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. *arXiv preprint arXiv:2310.07896*, 2023.

[192] Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Memer: Scaling up memory for robot control via experience retrieval. *arXiv preprint arXiv:2510.20328*, 2025.

[193] Simon Stepputtis, Joseph Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor. Language-conditioned imitation learning for robot manipulation tasks. *ArXiv*, abs/2010.12083, 2020.

[194] Austin Stone, Ted Xiao, Yao Lu, Keerthana Gopalakrishnan, Kuang-Huei Lee, Quan Vuong, Paul Wohlhart, Sean Kirmani, Brianna Zitkovich, Fei Xia, et al. Open-world object manipulation using pre-trained vision-language models. In *7th Annual Conference on Robot Learning*, 2023.

[195] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[196] Andrew Szot, Bogdan Mazoure, Omar Attia, Aleksei Timofeev, Harsh Agrawal, Devon Hjelm,

Zhe Gan, Zsolt Kira, and Alexander Toshev. From multimodal llms to generalist embodied agents: Methods and lessons. *arXiv preprint arXiv:2412.08442*, 2024.

[197] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[198] Yunlong Tang, Jing Bi, Siting Xu, Luchuan Song, Susan Liang, Teng Wang, Daoan Zhang, Jie An, Jingyang Lin, Rongyi Zhu, et al. Video understanding with large language models: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 2025.

[199] Gemini Robotics Team. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer, 2025. URL `https://arxiv.org/abs/2510.03342`.

[200] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

[201] GigaBrain Team, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Jie Li, Jindi Lv, Jingyu Liu, Lv Feng, et al. Gigabrain-0.5 m*: a vla that learns from world model-based reinforcement learning. *arXiv preprint arXiv:2602.12099*, 2026.

[202] TRI LBM Team. A careful examination of large behavior models for multitask dexterous manipulation. *arXiv preprint*, 2025. URL `https://arxiv.org/abs/2507.05331`.

[203] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[204] Marcel Torne, Andy Tang, Yuejiang Liu, and Chelsea Finn. Learning long-context diffusion policies via past-token prediction. *arXiv preprint arXiv:2505.09561*, 2025.

[205] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and Efficient Foundation Language Models, February 2023.

[206] Samuel Triest, Matthew Sivaprakasam, Sean J Wang, Wenshan Wang, Aaron M Johnson, and Sebastian Scherer. Tartandrive: A large-scale dataset for learning off-road dynamics models. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2546–2552. IEEE, 2022.

[207] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[208] Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, et al. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*, 2023.

[209] Wayve. Lingo: Natural language for autonomous driving, 2023. URL https://wayve.ai/thinking/lingo-natural-language-autonomous-driving/.

[210] Yi-Lin Wei, Haoran Liao, Yuhao Lin, Pengyue Wang, Zhizhao Liang, Guiliang Liu, and Wei-Shi Zheng. Cyclemanip: Enabling cyclic task manipulation via effective historical perception and understanding. *arXiv preprint arXiv:2512.01022*, 2025.

[211] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, and Jian Tang. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2409.12514*, 2024.

[212] Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.

[213] Philipp Wu, Arjun Majumdar, Kevin Stone, Yixin Lin, Igor Mordatch, Pieter Abbeel, and Aravind Rajeswaran. Masked trajectory models for prediction, representation, and control. *International Conference on Machine Learning*, 2023.

[214] Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, et al. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*, 2026.

[215] Ge Yan, Kris Wu, and Xiaolong Wang. ucsd kitchens Dataset. August 2023.

[216] Jonathan Yang, Catherine Glossop, Arjun Bhorkar, Dhruv Shah, Quan Vuong, Chelsea Finn, Dorsa Sadigh, and Sergey Levine. Pushing the limits of cross-embodiment learning for manipulation and navigation. *arXiv preprint arXiv:2402.19432*, 2024.

[217] Jonathan Heewon Yang, Dorsa Sadigh, and Chelsea Finn. Polybot: Training one policy across robots while embracing variability. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=HEIRj51lcS.

[218] Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-Hsuan Sung, et al. Multilingual Universal Sentence Encoder for Semantic Retrieval, July 2019.

[219] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

[220] Heng You, Tianpei Yang, Yan Zheng, Jianye Hao, and Matthew E Taylor. Cross-domain adaptive transfer reinforcement learning based on state-action correspondence. In *Uncertainty in Artificial Intelligence*, pages 2299–2309. PMLR, 2022.

[221] Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. Visual imitation made easy. In *Conference on Robot Learning*, pages 1992–2005. PMLR, 2021.

[222] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2636–2645, 2020.

[223] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557* , 2018.

[224] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[225] Tianhe Yu, Ted Xiao, Austin Stone, Jonathan Tompson, Anthony Brohan, Su Wang, Jaspiar Singh, Clayton Tan, Jodilyn Peralta, Brian Ichter, et al. Scaling robot learning with semantically imagined experience. *arXiv preprint arXiv:2302.11550*, 2023.

[226] Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017.

[227] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022.

[228] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5628–5635. IEEE, 2018.

[229] Zongzheng Zhang, Haobo Xu, Zhuo Yang, Chenghao Yue, Zehao Lin, Huan-ang Gao, Ziwei Wang, and Hao Zhao. Ta-vla: Elucidating the design space of torque-aware vision-language-action models. *arXiv preprint arXiv:2509.07962*, 2025.

[230] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[231] Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3d-vla: 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*, 2024.

[232] Chongyi Zheng, Benjamin Eysenbach, Homer Walke, Patrick Yin, Kuan Fang, Ruslan Salakhutdinov, and Sergey Levine. Stabilizing contrastive rl: Techniques for offline goal reaching, 2023.

[233] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024.

[234] Ruijie Zheng, Dantong Niu, Yuqi Xie, Jing Wang, Mengda Xu, Yunfan Jiang, Fernando Castañeda, Fengyuan Hu, You Liang Tan, Letian Fu, et al. Egoscale: Scaling dexterous manipulation with diverse egocentric human data. *arXiv preprint arXiv:2602.16710*, 2026.

[235] Gaoyue Zhou, Victoria Dean, Mohan Kumar Srirama, Aravind Rajeswaran, Jyothish Pari, Kyle Hatch, Aryan Jain, Tianhe Yu, Pieter Abbeel, Lerrel Pinto, Chelsea Finn, and Abhinav Gupta. Train offline, test online: A real robot learning benchmark, 2023.

[236] Xinghao Zhu, Ran Tian, Chenfeng Xu, Mingyu Ding, Wei Zhan, and Masayoshi Tomizuka. Fanuc manipulation: A dataset for learning-based manipulation with fanuc mate 200id robot. 2023.

[237] Yifeng Zhu, Peter Stone, and Yuke Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *IEEE Robotics and Automation Letters*, 7(2): 4126–4133, 2022.

[238] Yifeng Zhu, Abhishek Joshi, Peter Stone, and Yuke Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors, 2023.

[239] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *7th Annual Conference on Robot Learning*, 2023.

Chapter 8

Appendix

8.1 Collecting Robot Data for Generalization Research

8.1.1 Data Statistics

We provide a full list of the skills in our dataset (following our definition of skill in Section 2.3.3) in Table 8.1. We provide a breakdown of which portions of the dataset include which sensors in Figure 8.1.

8.1.2 Learning Method Implementation Details

Below we list relevant implementation details for each method. The complete set of evaluation tasks is shown in Figure 8.2.

All the goal-conditioned methods take in both an observation and goal. During training, the goal associated with an observation is selected by uniformly sampling an observation from the future timesteps in the trajectory.

Goal-conditioned behavior cloning

The observation and goal are stacked channel-wise before being passed into a ResNet-34 image encoder. This image encoding is passed through 3 256-unit fully connected layers to predict the robot action. We augment the observation and goal with random crops, random resizing, and color jitter. We use the Adam optimizer [100] with a learning rate of 3e-4. We use a linear warmup schedule with 2000 steps.

Diffusion goal-conditioned behavior cloning

In our implementation, we adopt the behavior cloning with diffusion strategy from the IDQL [68]. However, we do not learn a value function. The observation and goal are stacked channel-wise before being passed into a ResNet-34 image encoder. This image encoding is used to condition a diffusion process that models the action distribution. We use the DDPM (Denoising Diffusion Probabilistic Models) style objective as introduced by Ho et al. [73]. We augment the observation and goal with random crops,

Skill
Pick-and-place (includes reorienting objects in place)
Pushing objects
Wiping (e.g., wiping the table with a cloth)
Sweeping (e.g., sweeping beans into a pile)
Stacking
Folding cloths
Opening and closing drawers
Opening and closing doors
Opening and closing cardboard box flaps
Twisting knobs
Flipping switches
Zipping and unzipping
Turning levers

Table 8.1: The full set of skills in BridgeData V2. We define a skill to be a group of trajectories that require similar motions from the robot but may include different objects and start/end positions.

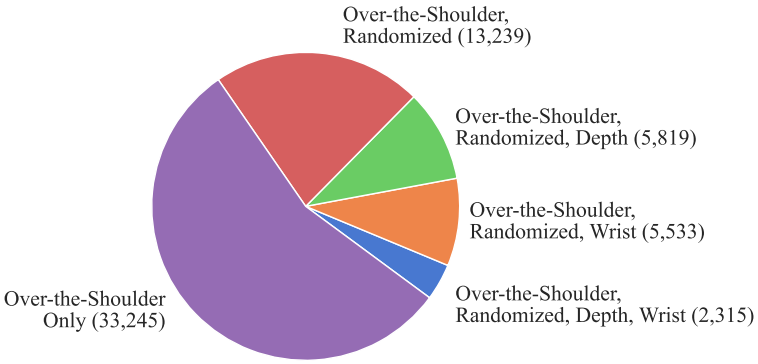


Figure 8.1: Breakdown of the entire dataset, including the autonomously collected data, by what camera views are included. "Over-the-shoulder" refers to the primary fixed camera, and "randomized" refers to the two alternative camera views that are randomized by the data collectors every 50 trajectories. "Depth", when present, is from the same perspective as the primary fixed camera. "Wrist" refers to the wide-angle wrist-mounted camera. More cameras were added to the hardware setup throughout data collection, so the majority of the data only includes the primary fixed camera view, and very little data currently includes all 4 views. However, now that the hardware is in place, more and more data will include all 4 views as the dataset continues to grow.



Figure 8.2: The complete set of evaluation tasks. The seen tasks are (clockwise): sweeping beans into a pile, putting a corn cob in a pot, opening a drawer, putting an eggplant in a pot, stacking a block, flipping a pot upright, folding a cloth, and putting a carrot on a plate. The unseen tasks are (left to right): putting a marker in a bowl, sweeping rice into a pile, folding a thick cloth, putting a mushroom in a pot, putting a spoon on a cloth, and wiping the table with a cloth.

random resizing, and color jitter. We use the Adam optimizer [100] with a learning rate of 3e-4. We use a linear warmup schedule with 2000 steps.

Action Chunking with Transformers

We use the same ACT hyperparameters as the original paper [230] except for the chunk size. The original ACT has chunk size 100 to accommodate for high-frequency control (50Hz) and long trajectories (1000 steps), while in our case the control frequency is 5Hz and the trajectory is much shorter at around 50-100 steps. We therefore reduced the chunk size to 5 and noted better performance. In addition, ACT was originally proposed as a single task method; we modify ACT to make it goal-conditioned. The observation and goal are stacked channel-wise before being passed into a ResNet-18 image encoder. The ACT policy was trained on a consumer grade workstation with 1x Nvidia 2080Ti GPU for 3 days.

Contrastive RL

Our implementation of contrastive RL mostly follows the design decisions described in [232] except for the following changes. First, given the observation and goal images, we feed them separately through a ResNet-34 encoder instead of a 3-layer CNN image encoder to get output encodings. Those image encodings then pass through two MLPs to get representations of the observation and the goal. Second, we share the ResNet-34 encoder between the value function and the policy. Intuitively, this encourages sharing of the causal information learned by the representations and speeds up learning. This ResNet backbone helps the method to consume large amounts of training data. Third, we increase the GCBC regularization coefficient to 0.2 to avoid sampling out-of-distribution actions and use the same batch size as other methods to make fair comparisons. Our contrastive RL objective retains the temporal-difference (TD) style used in [55]. We use the Adam optimizer [100] with a learning rate of 3e-4. We use a linear warmup schedule with 2000 steps.

Task	BridgeData V1 + PTR	BridgeData V2
Put marker in bowl†	0.05	0.65
Put mushroom in pot‡	0.10	0.70
Average	**0.08**	**0.70**

† Unseen objects, unseen environment ‡ Seen objects, unseen environment

Table 8.2: **(Dataset size and diversity ablation)** Comparison of the performance of GCBC trained on only BridgeData V1 + PTR (13k trajectories, 15 environments, 11 skills) and GCBC trained on BridgeData V2 (60k trajectories, 24 environments, 13 skills). The greater size and diversity of BridgeData V2 enables significantly better generalization to these unseen tasks.

Language-conditioned behavior cloning

Our implementation of LCBC uses a ResNet-34 with FiLM conditioning. The language instruction is first encoded with a frozen MUSE encoder and passed through 2 fully connected layers. The image observation is then passed into the ResNet, which is conditioned on the language embedding using FiLM layers. FiLM layers are applied at the end of every ResNet block to condition on language throughout the network. Finally, the image encoding is passed into a fully connected network to predict the action. We use the Adam optimizer [100] with a learning rate of 3e-4. We use a linear warmup schedule with 2000 steps.

RT-1

We use the same hyper-parameters as the original RT-1 paper [20], except for increasing the sequence length of the transformer from 6 to 15 to accommodate for the longer episode length. We scale each action dimension to range -1 and 1 and use a vocabulary size of 256 to tokenize the actions.

8.1.3 Hardware Setup

We use an Intel RealSense D435 RGBD camera as a fixed over-the-shoulder camera view and two Logitech C920 webcams to capture alternative camera views that are randomized during data collection. For the wrist camera, we designed a custom 3D printed mount to attach a Raspberry Pi camera module to the gripper. We used a Meta Quest 2 VR headset to teleoperate the robot.

8.1.4 Comparison to the Original Bridge Data

In Table 8.2, we compared the performance of GCBC trained on the combination of the original Bridge Data and data collected for PTR [105], with the performance of GCBC trained on all of BridgeData V2. We find that the additional data released as part of this paper significantly improves performance on unseen pick-and-place tasks.

8.2 Designing a Generalist Robot Policy

8.2.1 Contributions

Dibya Ghosh: led model development, proposed and implemented large parts of the final model design, babysitted the training runs and touched all parts of the codebase, helped with model evaluations and tech report writing.

Homer Walke: led model evaluations, designed the main Bridge evaluation benchmark, contributed to the initial model implementation and ran many of the evals for this tech report.

Karl Pertsch: managed the overall project, led Open-X data integration and curation, led writing of this tech report, contributed to model development and implementation and ran model evaluations for the tech report.

Kevin Black: led data loading and training infrastructure, managed TPU pod training, created the project website, contributed to model development and implementation, and helped with robot evaluations and tech report writing.

Oier Mees: ran countless ablations for model development, contributed to model implementation, helped with evals and writing of the tech report.

Sudeep Dasari: contributed the model evaluations at CMU, experimented with pretrained encoders.

Joey Hejna: contributed the model evaluations at Stanford.

Tobias Kreiman: contributed model evaluations in simulated environments.

Ria Doshi: contributed to explorations into cross-morphology training.

Charles Xu: contributed the model evaluations for Berkeley Peg Insert.

Jianlan Luo: contributed the model evaluations for Berkeley Peg Insert.

You Liang Tan: helped diagnose and resolve bottlenecks in data loading.

Lawrence Yunliang Chen: helped set up and run UR5 finetuning experiments at Berkeley AutoLab.

Pannag Sanketi, Quan Vuong, Ted Xiao: contributed model evaluations on the Google Robot.

Dorsa Sadigh, Chelsea Finn, Sergey Levine: provided guidance throughout the project and feedback on the writing of this tech report.

8.2.2 Octo Code Example

Loading a pretrained Octo model and performing inference requires little code:

```
import jax
from octo.model.octo_model import OctoModel

model = OctoModel.load_pretrained("hf://rail-berkeley/octo-base")
print(model.get_pretty_spec()) # Print out the input-output spec
observation = {"image_primary": img}
task = model.create_tasks(texts=["pick up the fork"])
action = model.sample_actions(
```

Octo Pretraining Dataset Mixture	
Fractal [20]	17.0%
Kuka [89]	17.0%
Bridge[54, 207]	17.0%
BC-Z [85]	9.1%
Stanford Hydra Dataset [8]	6.0%
Language Table [127]	5.9%
Taco Play [169, 139]	3.6%
Furniture Bench Dataset [70]	3.3%
UTAustin Mutex [182]	3.0%
Austin Sailor Dataset [143]	2.9%
Roboturk [130]	2.8%
Toto [235]	2.4%
Austin Sirius Dataset [120]	2.3%
Berkeley Autolab UR5 [30]	1.5%
IAMLab CMU Pickup Insert [173]	1.2%
Viola [238]	1.2%
Berkeley Fanuc Manipulation [236]	1.0%
NYU Franka Play Dataset [41]	0.9%
UCSD Kitchen Dataset [215]	<0.1%
Jaco Play [45]	0.6%
Berkeley Cable Routing [124]	0.3%
Austin Buds Dataset [237]	0.3%
CMU Stretch [140]	0.2%
NYU Door Opening [151]	0.1%
DLR EDAN Shared Control [161]	0.1%

Table 8.3: Octo pretraining data mixture using datasets from the Open X-Embodiment dataset [148].

```
    observation, task, rng=jax.random.PRNGKey(0))
```

Listing 8.1: Example Python code to perform inference with a pretrained Octo model.

8.2.3 Data Mixture

We list the detailed training mixture used for training the Octo models in Table 8.3. The sampling weights are mostly determined by the relative size of the datasets with a few manual adjustments (see Section 3.3.2).

8.2.4 Training Hyperparameters

We mostly follow documented practices for training vision transformers [227]. We use the AdamW optimizer [123] with an inverse square root decay learning rate schedule [227] and linear learning rate warmup. We list hyperparamaters used during training in Table 8.4 and the model parameters for the different

Hyperparameter	Value
Learning Rate	3e-4
Warmup Steps	2000
LR Scheduler	reciprocal square-root
Weight Decay	0.1
Gradient Clip Threshold	1
Batch Size	2048

Table 8.4: Hyperparameters used during training.

Model	Layers	Hidden size D	MLP size	Heads	Params
Octo-Small	12	384	1536	6	27M
Octo-Base	12	768	3072	12	93M

Table 8.5: Architecture details of Octo model variants.

sizes in Table 8.5. We apply standard image augmentations during training. Concretely, for the 3rd person camera we apply stochastic crops followed be a resize to 256×256, followed by color jitter. Finally, we normalize the input image to have pixels with float values between -1.0 and 1.0. For the wrist camera, we apply the same procedure except without the random crop and resizing to 128×128 instead.

The images are passed through a shallow convolution stack, then split into a sequence of flattened patches [49] of size 16×16. This results in 256 tokens for the 3rd person camera images and 64 tokens for the wrist camera images. For datasets containing language annotations, we use a pretrained `t5-base` (111M) transformer model [163] that produces a sequence of 16 language embedding tokens.

The diffusion action head consists of a 3-layer MLP with a hidden dimension of 256, residual connections, and layer normalization. We use the standard DDPM objective as introduced by [73] with a cosine noise schedule [145] and 20 diffusion steps.

8.2.5 Things that Worked and Did Not Work (Yet)

Things we found improved performance

- **Adding history during pretraining**: Models with one frame of history as context performed better in zero-shot evals than models pretrained without history. We did not observe benefits of increasing the history length further on the few tasks we evaluated on, though other tasks may benefit.
- **Using action chunking**: We found it helpful to use "action chunking" [230], i.e., to predict multiple actions into the future, for getting more coherent policy movements. In our evaluations, we did not find temporal ensembling of future actions to provide additional benefits beyond receding horizon control.
- **Decreasing patch size** Tokenizing images into patches of size 16×16 led to improved performance over patches of size 32×32, particularly for grasping and other fine-grained tasks. This

does add compute complexity (the number of tokens is $4\times$), so understanding how to balance compute costs and resolution remains a problem of interest.

- **Increasing shuffle buffer size**: Loading data from 25 datasets in parallel is a challenge. Specifically, we found that achieving good shuffling of frames during training was crucial — zero-shot performance with a small shuffle buffer (20k) and trajectory-level interleaving suffered significantly. We solved this issue by shuffling and interleaving frames from different trajectories *before* decoding the images, allowing us to fit a much larger shuffle buffer (up to 500k). We also subsample at most 100 randomly chosen steps from each training trajectory during data loading to avoid "overcrowding" the shuffle buffer with single, very long episodes.

Things that did not work (yet)

- **MSE Action Heads**: Replacing our diffusion decoding head with a simple L2 loss led to "hedging" policies that move very slowly and e.g., fail to rotate the gripper in WidowX evaluations.
- **Discrete Action Heads**: Discretizing actions into 256 bins per dimension and training with cross-entropy loss like in Brohan et al. [20] led to more "decisive" policies, yet they lacked precision and often missed the grasp.
- **ResNet Encoders**: did not scale as well to larger datasets in our evaluations (see Table 3.2), though they did outperform our ViT architecture when training from scratch on a small dataset (around 100 demonstrations).
- **Pretrained Encoders**: ImageNet pretrained ResNet encoders did not provide benefit on zero-shot evals, though may be confounded with ResNet architectures underperforming as mentioned above.
- **Relative Gripper Action Representation**: When aligning the gripper action representations of the different datasets, we tried (A) absolute gripper actions, i.e., actions are +1 when the gripper is open and 0 if it is closed, and (B) relative gripper actions, i.e., gripper action is +1/0 only in the timestep when the gripper opens/closes and 0.5 otherwise. We found that the latter tends to open/close the gripper less often since most of the training data represents "do not change gripper" actions, leading to a slightly higher grasp success rate. At the same time, the relative representation led to less retrying behavior after a grasp failed, which was ultimately worse. Thus, we chose the absolute gripper action representation.
- **Adding Proprioceptive Inputs**: Policies trained with propioceptive observations seemed generally worse, potentially due to a strong correlation between states and future actions. We hypothesize this might be due to a causal confusion between the proprioceptive information and the target actions [46].
- **Finetuning Language Model**: In order to improve the visuo-lingual grounding of Octo we experimented with: i) varying sizes of the T5 encoder [163]: `small` (30M), `base` (111M), and `large` (386M) as well as ii) finetuning the last two layers of the encoder. Using the frozen `base` model resulted in the best language-conditioned policies. We did not find improvements when using larger encoders or finetuning the encoder. We hypothesize this might be due to the lack of rich, diverse, free-form language annotations in most of the datasets.

8.2.6 Experimental Setups

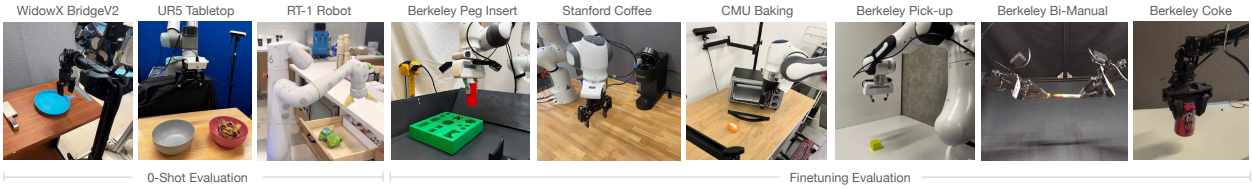


Figure 8.3: **Evaluation Tasks.** Replicated from the main text for convenience. We evaluate Octo on 9 real robot setups across 4 institutions in zero-shot and finetuning scenarios.

Zero-Shot Evaluations

WidowX BridgeV2 Uses the setup of Walke et al. [207], in which a Trossen WidowX 250 6-DOF robot performs diverse manipulation tasks. The observation consists of a single third person camera stream and the action space is end-effector position deltas. We evaluated two language-conditioned tasks in which a the robot needs to "place the carrot on plate", and "put the eggplant in the pot." While these tasks are in-distribution, the policy must still generalize to novel object positions. We performed 10 trials per task and varied objects positions between trials.

UR5 Uses the setup of Chen et al. [30], in which a UR5 robot arm performs multiple table top manipulation tasks. The observation consists of a single third person camera stream and the action space is end-effector position deltas. We evaluated twp language-conditioned tasks: picking a toy tiger from a bowl and placing it into a different bowl as well as wiping a table with a cloth. While these tasks are in-distribution, the policy must still generalize to novel object positions, distractor objects, and lighting. Since the training data was collected months ago and the robot setup was taken down and re-assembled, the policy must also generalize to other miscellaneous changes in the environment like a slightly different camera view and background. We performed 10 trials per task and varied objects positions between trials.

RT-1 Robot Uses the setup of Brohan et al. [20], in which a proprietary robot performs multiple table top and furniture manipulation tasks. The observation consists of a single third person camera stream and the action space is end-effector position deltas. We evaluated on the task of picking up a 7up can, apple, blue chip bag, or brown chip bag, as well as the task of opening or closing drawers on a cabinet. While these tasks are in-distribution, the policy must still generalize to novel object positions. Additionally, since the robot setup was moved between buildings, the policy must also generalize to miscellaneous changes in the environment like a slightly different camera view and background. We performed 10 trials per task and varied objects positions between trials.

Model Ablations

All of our model ablations were evaluated on the WidowX setup. We present a more detailed breakdown of the success rates per task in Table 8.6. We evaluated on two language-conditioned tasks (put carrot on plate and put eggplant in pot) and two goal-conditioned tasks (put bread on plate and put spoon

		Language-conditioned		Goal-conditioned		
		Put carrot on plate	Put eggplant in pot	Put bread on plate	Put spoon on glove	Average
	Octo-small (Ours)	80%	90%	70%	90%	**83%**
DATA	RT-X dataset mix [148]	80%	80%	40%	40%	60%
DATA	Single robot dataset (Bridge Data)	20%	70%	60%	20%	43%
POLICY	Discretized Action Prediction [148]	0%	20%	10%	40%	18%
POLICY	Continuous Action Prediction (MSE)	70%	30%	0%	40%	35%
ARCH	Resnet-50 + Transformer [148]	80%	60%	100%	40%	70%

Table 8.6: **Model Ablations.** We achieve best performance when using the ViT architecture, diffusion action head, and wide training data mixture. All evaluations are performed on the WidowX setup. Success rates are averaged over 40 trials across two language-conditioned tasks and two goal-conditioned tasks.

Generalization Type	Task	Success Rate	Average
In-distribution	Put carrot on plate	80%	85%
	Put eggplant in pot	90%	
Novel objects	Put bread on plate	70%	80%
	Put spoon on glove	90%	
Novel environment	Put mushroom in pot	20%	40%
	Put spoon on cloth	60%	
Novel skill	Flip cup on its side	10%	5%
	Put block in slot	0%	

Table 8.7: **Zero-shot Generalization Analysis.** Using the WidowX setup, we analyze the generalization capabilities of Octo-small along several different axes. Octo performs the best on in-distribution tasks and generalizes well to novel objects and environments. However, Octo struggles to generalize to skills not seen in the WidowX embodiment. Success rates are averaged over 20 trials across two tasks.

on glove). The goal-conditioned tasks contain objects not seen in the Bridge dataset. Additionally, we analyze the generalization capabilities of Octo across several different axes, such as novel objects, novel environments and novel skills in Table 8.7.

Finetuning Evaluations

CMU Baking The robot must pick up the toy bread object, place it in the toaster, and shut the toaster. This task requires generalization across initial positions (of both the toaster and object) and the shape of the target toy bread object. We use an end-effector delta action space (Cartesian position + rotation delta). Observations come from the 3rd-person front-facing Zed camera. Actions are predicted at 15 Hz, and executed on the robot using the R2D2 Franka controller. The finetuning dataset consists of 120 demos collected via expert VR tele-operation, and every policy was evaluated using 20 trials (4 novel test objects with 5 positions each).

Stanford Coffee The robot is tasked with picking up one of four different Keurig Coffee Pods and placing it inside of a Keurig machine. This task requires both generalization across initial positions and colors of the coffee pod, as well as precise placement in the Keurig machine. We use an end effector delta action space with an open source controller running at 10 Hz based on Polymetis [118] (found here). We use only a single 3rd-person wrist observation. Our training dataset contained 118 expert demonstrations from varied coffee pods and positions collected via VR tele-operation. We evaluated policies for 20 episodes, five episodes for each of four different color coffee pods.

Berkeley Peg Insertion The task is to insert a pre-grasped 3D-printed peg into a matching slot on a 3D-printed board inside the bin. The matching tolerance between the peg and the hole is 1.5mm; which makes it a contact-rich precise part-mating task. The robot must learn an appropriate policy to "search" for the matching opening through contact, which necessitates the use of force/torque measurements. The observation space of the policy consists of a single side-view camera image, the end-effector twist, and the end-effector force/torque reading. The policy sends action commands as the robot's end-effector twists at 5 HZ, tracked at 1000 HZ by a low-level impedance controller. Our finetuning dataset is composed of 100 human demonstrations from the FMB dataset [125]. We evaluated trained policies for 20 trials with randomized board positions.

Berkeley Pick Up We use the setup of Radosavovic et al. [162]. The robot needs to pick up a block from a table top surface after being trained on a dataset of 100 pickups of various objects. The robot uses joint position control with an underlying Polymetis control stack [118] (here). It is conditioned on a wrist camera input image, as well as the proprioceptive readings of the robot. We evaluted on the task of picking up the yellow cube and used 20 trials.

Berkeley Coke A Trossen Robotics ViperX robot is tasked with picking up a coke can from a table. The ViperX is completely unseen in the training mixture, thus this task tests the model's capabilities to generalize to a new robot embodiment. The task also requires generalization across different positions of the coke can with the ViperX robot using end-effector delta control at 5Hz. The policy uses image observations from a third person and a wrist camera. Our training dataset contained 115 expert demonstrations collected via VR tele-operation from varied positions.

Berkeley Bimanual The task requires an ALOHA bimanual robot [230], consisting of two ViperX robot arms, to pick up a sharpie marker with the right hand from the workbench and remove its cap with the left hand. Since Octo was only pre-trained with single-arm robot data, we re-initialize a new action head that maps to the 14-dimensional action space of ALOHA (2x 6 joint positions + 2x gripper position). A successful policy needs to use image observations from the left and right wrist cameras to predict precise manipulation behaviors. Best performance for both our methods and baselines was achieved by predicting an action chunk of 64 during training, and then during test-time executing 12 actions with receding horizon control before re-planning. We evaluated the uncapping task using 10 trials, where the pen was placed in a different location for each trial.

8.3 Controlling Multiple Robot Embodiments with a Single Policy

More information and videos can be found on our website: crossformer-model.github.io.

8.3.1 Training Data

We list the sampling weights for each dataset in our data mixture in Table 8.8. We up-weight our target datasets: Bridge, ALOHA-multi-task, GNM, Go1-walk, and Franka-tabletop.

8.3.2 Training Hyperparameters

In Table 8.9 we list the hyperparameters for the optimizer and policy architecture. In total, along with the ResNet-26 image encoders and action heads, our model has 130M parameters. We initialize the ResNet-26 encoders with ImageNet pre-trained weights. Training took 80 hours on a TPU V5e-256 pod. We apply color jitter and random resizing/cropping image augmentations. During training, we use hindsight goal relabeling and sample future observations uniformly at random to use as goals [3]. If a language instruction is available for a trajectory, we randomly mask either the language or goal so that at test time we can condition our policy using either task specification [126]. For our method and baselines we trained, we selected checkpoints based on the validation mean squared error since we found this very roughly correlates with robot performance.

8.3.3 Evaluation Setups

Below we provide further details on our evaluation settings (see Fig. 4.3 for images). For goal-condtioned tasks, we arrange the scene and take the goal image before rolling out the policy. For the language-conditioned tasks, we use language instructions that are semantically in the distribution of the training data (though may not appear exactly in the training data).

For the manipulation settings, we ran the models on a machine with an NVIDIA 4090 (or similar GPU) that was connected to the robot. For the ground navigation and drone experiments, we ran the models on a separate server and sent actions over a wireless network.

WidowX Manipulation We use the Bridge setup from Walke et al. [207]. We use an over-the-shoulder camera view and and sample actions from the single arm head of our policy. We evaluated two language-conditioned tasks, putting a spoon on a cloth and putting a carrot on a plate, and two goal-conditioned tasks, putting a mushroom in a pot and putting a cloth on a saucer. We used 12 trials per policy and task. Positions of the objects were varied between trials.

Franka Manipulation We use the DROID setup from Khazatsky et al. [97]. We use an over-the-shoulder camera view and sample actions from the single arm head of our policy. We evaluated for 27 trials per policy on the language-conditioned task of using a sponge to sweep pinecones into a dustpan.

CrossFormer Dataset Mixture	
Fractal [20]	17%
Kuka [89]	2.2%
BC-Z [85]	2.2%
Stanford Hydra Dataset [8]	0.015%
Language Table [127]	1.5%
Taco Play [169, 139]	1.2%
Furniture Bench Dataset [70]	0.83%
UTAustin Mutex [182]	0.76%
Austin Sailor Dataset [143]	0.74%
Roboturk [130]	0.79%
Toto [235]	0.68%
Austin Sirius Dataset [120]	0.59%
Berkeley Autolab UR5 [30]	0.41%
IAMLab CMU Pickup Insert [173]	0.31%
Viola [238]	0.32%
Berkeley Fanuc Manipulation [236]	0.26%
NYU Franka Play Dataset [41]	0.28%
Jaco Play [45]	1.6%
Berkeley Cable Routing [124]	0.089%
Austin Buds Dataset [237]	0.072%
CMU Stretch [140]	0.053%
DLR EDAN Shared Control [161]	0.019%
DROID [97]	0.022%
Bridge [54, 207]	17%
GNM [180]	17%
ALOHA-multi-task	17%
Go1-walk	8.5%
Franka-tabletop	8.5%

Table 8.8: The CrossFormer training data mixture uses datasets from the Open X-Embodiment dataset [148] and additional data collected for this project.

Hyperparameter	Value
Optimizer	AdamW [123]
Learning Rate	3e-4
Warmup Steps	2000
LR Scheduler	reciprocal square-root
Weight Decay	0.1
Gradient Clip Threshold	1
Batch Size	512
Layers	12
Attention heads	8
Token embedding size	512
MLP dimension	2048
Context length	2135
Total training steps	300K

Table 8.9: Training hyperparameters for CrossFormer.

We also evaluated for 12 trials per policy on the task of flipping a pot upright. The positions of the objects were varied between trials.

ALOHA Bimanual Manipulation We use the ALOHA setup from Zhao et al. [230]. We use three camera views, one overhead and two wrist, and sample actions from the bimanual head of our policy. We evaluated two language-conditioned tasks, taking the cap off of a pen and picking up a knife and using it to cut a piece of sushi. We used 10 trials per policy and task. The position of the pen, knife, and sushi was varied between trials.

LoCoBot Navigation We use the LoCoBot setup from Shah et al. [181] which has one camera view. We evaluate on suite of three skills: path-following, obstacle avoidance, and sharp corner-turning. We sample actions from the navigation head of our policy. We combine our policy with the graph-based planner and distance function from Shah et al. [181]. We first obtain a topological map $\mathcal{M}$ of the environment by teleoperating the robot. Then, at every time-step we find the closest node in $\mathcal{M}$, search the graph for the shortest path from this node to the goal, and command the policy with the most immediate subgoal in the path. We evaluate the success of a trajectory based on the number of subgoals between the closest node at the end of the trajectory and the goal. We evaluate in 6 locations, using one trial per location and policy as done in prior work [180, 181].

Go1 Quadruped We evaluate on a Unitree Go1 which uses proprioceptive observations $o_t \in \mathcal{R}^{59}$. We sample actions from the quadruped head of our policy, and the task is to walk forward. Importantly, unlike prior work [216], we directly control the quadruped's joints rather than doing higher level control with navigation waypoints. As our evaluation metric we report reward achieved over 25 minutes normalized by the reward achieved by the RL-trained expert policy that generated the data (see Section 4.3.1).

	Task	Single-Robot Dataset	Best Prior Method	CrossFormer
WidowX	Put the spoon on the cloth	0.25 (12)	0.25 (12)	0.25 (12)
	Put the mushroom in the pot	0.00 (12)	0.17 (12)	0.25 (12)
	Put the cloth on the saucer	0.75 (12)	0.67 (12)	0.75 (12)
	Put the carrot on the plate	0.67 (12)	0.25 (12)	0.33 (12)
Franka	Sweep the pinecones into the dustpan	0.41 (27)	0.52 (27)	0.41 (27)
	Flip the pot upright	0.83 (12)	0.67 (12)	0.83 (12)
ALOHA	Uncap the pen	0.60 (10)	0.70 (10)	0.80 (10)
	Use the knife to cut the sushi	0.40 (10)	0.30 (10)	0.60 (10)
LoCoBot	Obstacle avoidance	0.95 (2)	0.30 (2)	0.95 (2)
	Cornering	0.95 (2)	0.85 (2)	0.95 (2)
	Sharp cornering	0.85 (2)	0.30 (2)	0.88 (2)
Tello Quadcopter	Cornering	0.68 (3)	0.68 (3)	0.82 (3)
G01	Walking	1	N/A	1
Average		0.68 (116)	0.51 (116)	0.73 (116)

Table 8.10: **Real Evaluation (detailed).** We compare CrossFormer to the same architecture trained on just the data from the target robot, as well as the best prior method on the data from the target robot. The number of trials is in parentheses. For ground and aerial navigation (LoCoBot and Tello Quadcopter) the success metric is the proportion of subgoals reached on the path towards the goal. For the G01, we report reward achieved over 25 minutes normalized by the reward achieved by the RL-trained expert policy that generated the data. Because there is no suitable best prior imitation learning method for the G01, we do not include it in the best prior method average.

Tello Quadcopter Finally, we perform evaluation on a Tello quadcopter using the navigation head of our policy. Since the navigation head outputs 2-D relative waypoints, we maintain a static height throughout the trajectory [181, 180]. Notably, we do not train on quadcopter data so this setting requires *zero-shot* generalization to a new embodiment. We evaluated on the skill of turning a corner. We evaluated in 3 locations, using one trial per policy and location as done in prior work [180, 181].

8.3.4 Evaluation Results

In Table 8.10 we provide a more detailed breakdown of our evaluation results by task.

8.4 Adding Memory to a Generalist Robot Policy

8.4.1 Contributions

The project was started by HW and DD. DD, HW, and JTS designed the video encoder and short-horizon memory system. KP designed the long-horizon memory system. MT, KP, HW, KV, SN, and ME designed the evaluation suite and performed experimental evaluations, including model post-training. DD, KP, AR, and QV developed the pre-training recipe and data mixture, and ran large-scale model training. HW, KD, KS, and MT developed the infrastructure for long-context VLA training. JT optimized inference

of the MEM VLA. SL and CF provided advice throughout the project. MT, KP, DD, BI, KV, SN, SL, and CF worked on writing, illustrations, and the video.

8.4.2 Task Details

Long-horizon Tasks (section 5.4.1)

Preparing the items for a recipe The goal of the task is for the robot to take out all of the items for a recipe. The items are placed in various, randomized locations throughout the kitchen (fridge, cabinet, drawers, stove ...). The policy's prompt specifies which items should be assembled, where they should be placed (countertop, stove, sink), and their respective initial location in the kitchen. We list a few prompt examples below. The robot will receive 1 point per item retrieved and placed in the requested position. Most recipes consist of 6 to 7 points. We collected data on over 40 recipes with a large variety of objects, scenes, appliances, prompts. The evaluation is done on a variety of unseen scenes, with seen recipes.

Mashed potatoes

Take out supplies for the recipe. Remove the lid from the pot on the stove and place the lid on the countertop to the left of the sink, and pick up the pot and place it in the sink. Get the potatoes, butter, milk from the fridge and place them on the countertop to the left of the sink. Get the masher from the top drawer to the left of the sink and place it on the countertop to the left of the stove.

Fried rice

Take out the supplies for the recipe. Get the bag of rice and the spam from the cabinet on the bottom right of the dishwasher and place them on the countertop to the left of the sink. Get the soy sauce from the fridge and place it on the same section of the countertop to the left of the sink. Get the pan from the bottom cabinet to the right of the dishwasher and put it on the stovetop. Get the spatula from the top drawer to the right of the dishwasher and place it on the pan on the stovetop.

Pizza

Take out the supplies for the recipe. Get the baking tray from the bottom cabinet on the left of the dishwasher and place it on the countertop to the left of the sink. Get the pizza dough, package of pepperoni, and bag of cheese from the fridge and place them on the countertop to the left of the sink. Get the dough roller from the top drawer to the right of the stove and place it on the countertop to the left of the sink.

Cleaning the kitchen This task requires cleaning a kitchen by wiping the countertop with the sponge, drying the countertop with a paper towel, storing any food items left in the countertop inside the fridge, putting all the dishes from the dishrack into the cabinets, and washing all of the dishes from the sink. The scoring consists of +1 point per subtask done and on average episodes consist of 8 subtasks. Below we provide an example prompt.

Clean kitchen

Clean up the kitchen. Wipe the countertop with the sponge and dry it with the paper towel and throw the towels in the trash can under the sink. Put the mustard in the fridge. Put the dishes in the bottom cabinet on the left of the fridge. Wash the blue plate and black plate and place them in the dishrack.

In-context adaptation (section 5.4.2)

Chopstick Pick Up The policy is tasked with picking up a chopstick on a variable height table. During data collection, the chopstick is placed at random locations on the table with random table heights in the upper half of the table height range. We collect human interventions whenever the policy mis-grasps, and then train both memory and non-memory policies on this data and evaluate them with the chopstick randomly placed on the table at the lowest table height setting. The final policies are scored +1 for picking up the chopstick and +1 for placing it in the bin, and success is a score of 2.

Open Refrigerator The goal of the task is to open a fridge in front of the robot. The fridge does not have obvious visual cues indicating which side the door hinge is on, leading policies to often attempt to open it in the wrong direction. We collect human interventions whenever the robot opens the fridge in the wrong direction. For evaluation, an episode is defined as successful if it takes ≤ 4 grasps to open the door, to test for intentional strategy switching rather than repeated random sampling.

Analysis Experiments (section 5.4.3)

Three-way swap mugs The goal is to place three coffee mugs under a coffee machine sequentially. Two mugs start at random positions on the table and a third mug starts under the coffee machine. The scoring consists of obtaining +1 point for each mug that goes on the coffee machine without tipping and without repeating mugs. Only one mug can go on the coffee machine at a time, so the robot needs to place it back on a random location on the table after it was placed under the coffee machine to make space for the next mug.

Find object The goal is to retrieve the hidden object in a cabinet with four drawers. In the beginning of the episode, a person places the object in a random drawer. The robot needs to remember which drawer the person placed the object into, then open the correct drawer and retrieve the item and place it on the table. One point will be received for successfully accomplishing the task without opening any incorrect drawer.

Unpack groceries The goal is to retrieve all items from a grocery bag without missing any items inside. The inside of the bag is not observable from the third person camera of the robot, and the number of items in the bag is randomized across episodes. Thus, to complete the task, the robot will need to remember how many objects are left in the bag from its past wrist camera observations. Episodes are marked as a

success if the policy retrieved all items from the bag and indicated completion by moving home once all items are retrieved.

Scoop coffee The goal is to put exactly two scoops of coffee beans in a grinder. The robot needs to remove the lid from the hopper of the grinder, pick up a coffee scoop, put two scoops of coffee from an opened coffee bean package into the grinder, then put the lid back on. Episodes are marked as a success if exactly two scoops are added to the grinder and the lid is put back on.

Grilled cheese The goal of the task is to prepare a grilled cheese sandwich. The robot has to assemble the grilled cheese sandwich in the pan, then wait for the grilled cheese to cook, flip it, wait for the other side to cook, and finally plate the grilled cheese. The policy receives +1 point for assembling the sandwich, cooking for the correct amount of time (between 30 seconds and 3 minutes per side), flipping, and plating.

Window cleaning The goal is to wipe the window door of a phone booth. The task consists of spraying windex on the window, ripping off a paper towel from a roll, then wiping the whole window, and finally throwing the paper towel in the trash. The robot needs to remember which steps have been completed and which part of the window have already been wiped during the cleaning phase. Episodes are marked as a success if all steps of the task are completed. It is considered a failure if the policy fails to wipe parts of the window.

Table bussing The goal is to sort 12 objects from a tabletop into (A) a bussing bin (for utensils and tableware), and (B) a trash can (for plastic containers, bottles, and wrapping paper). The policy receives a score of +1 for every correctly placed item.

Shirt folding The robot has to fold a shirt on a tabletop. At the beginning of the episode the shirt is flattened on the table. The episode is scored as a success if the shirt is successfully folded into a rectangle without seams sticking out or excessive wrinkles.

Clean up counter The goal is to clean items from a counter into a drawer. The robot has to open the drawer, place all items on the counter into the drawer, and close it. An episode is counted as a success if all subtasks are completed successfully.

Make bed The goal is to tidy a bed, by straightening the blanket, and placing any pillows at the head of the bed. The episode is scored as a success if all pillows are placed at the head of the bed, and the blanket is tidied.

Kitchen cleanup The goal is to clean the counter of a kitchen, by placing multiple plates, bowls, and a cutting board from the counter into a nearby sink. The episode is scored as a success if all plates are placed into the sink.

Batch folding In contrast to the shirt folding task, here the clothes start in a hamper on one side of the table. The robot needs to take the clothing item out of the hamper, flatten it, fold it, and then stack it onto a stack of existing clothes on the table. The hamper may contain various t-shirts and shorts. Episodes are scored as successes if the clothing items are folded into rectangles without seams sticking out or excessive wrinkling.

Box building Given a flattened cardboard cutout, the robot is tasked to fold the cardboard into a box. This task requires multiple precise, well-coordinated bimanual manipulations to assemble the box. Episodes are scored as successful if the box is fully assembled without damaging the box.

8.4.3 Video encoder with Space-Time Separable Attention

We describe how we can adjust a layer in a given ViT image encoder to instantiate our video-encoding scheme.

Let $\mathbf{z}_{p,t}^{l-1}$ denote the input embeddings to layer l for spatial patch p and timestep t (where $t \in [-K, 0]$). We first modify the embedding by adding a sinusoidal position embedding based on t, denote the output of this step with

$$\hat{\mathbf{z}}_{p,t}^{l-1} = \mathbf{z}_{p,t}^{l-1} + e(t),$$

where $e(t)$ denotes the sinusoidal position embedding and we set the boundary condition $e(0) = 0$.

We then re-use the ViT's standard query key and value projections which in our case are given as

$$\begin{aligned} \mathbf{q}_{p,t}^{l,a} &= W_Q^{l,a}\mathrm{LN}\big(\hat{\mathbf{z}}_{p,t}^{l-1}\big), \\ \mathbf{k}_{p,t}^{l,a} &= W_K^{l,a}\mathrm{LN}\big(\hat{\mathbf{z}}_{p,t}^{l-1}\big), \\ \mathbf{v}_{p,t}^{l,a} &= W_V^{l,a}\mathrm{LN}\big(\hat{\mathbf{z}}_{p,t}^{l-1}\big), \end{aligned} \tag{8.1}$$

where a indexes the attention head and LN denotes a layer-norm implementation (we use RMSNorm but this choice is dependent on the ViT we start from).

Next, we can define the general attention mechanism (ignoring normalization for brevity of notation) as the softmax over the query and key outer product:

$$\alpha_{p,t}^{l,a}(\mathcal{S} = \{1, \dots N\}, \mathcal{T} = \{1, \dots, T\})[\hat{\mathbf{z}}] = \mathrm{SM}\Big((\mathbf{q}_{p,t}^{l,a})^T \cdot \big[\mathbf{k}_{0,0}^{l,a}\{\mathbf{k}_{p',t'}^{l,a}\}_{p' \in \mathcal{S}, t' \in \mathcal{T}}\big]\Big), \tag{8.2}$$

where SM denotes the softmax operation and $\mathcal{S} = \{1, \dots N\}$ and $\mathcal{T} = \{1, \dots, T\}$ denote the space and time indices we want to perform attention over respectively. With this general definition we can define the space only attention mechanism as instantiating $\alpha_{p,t}^{l,a}(\mathcal{S} = \{1, \dots N\}, \mathcal{T} = \{\})$ and the time only attention mechanism as instantiating $\alpha_{p,t}^{l,a}(\mathcal{S} = \{\}, \mathcal{T} = \{1, \dots, T\})$. Thus the attention mechanism used in our video encoder is precisely given as

$$\alpha_{p,t}^{l,a}(\mathcal{S} = \{1, \dots N\}, \mathcal{T} = \{\})[\alpha_{p,t}^{l,a}(\mathcal{S} = \{1, \dots N\}, \mathcal{T} = \{1, \dots, T\})[\hat{\mathbf{z}}]]. \tag{8.3}$$

And thereafter we follow the standard computation of a transformer layer, i.e. we compute the outputs of the transformer by first combining attention weights α with values $\mathbf{v}$ and passing the corresponding outputs to a MLP.