

ABSTRACT

Title of Dissertation: **TOWARDS GENERALIST EMBODIED AGENTS
VIA REPRESENTATION LEARNING**

Ruijie Zheng, Doctor of Philosophy, 2025

Dissertation directed by: **Professor Furong Huang
Department of Computer Science**

In our dynamic and ever-evolving world, embodied agents for sequential decision-making (SDM) lie at the heart of intelligent behavior in machine learning systems. Just as foundation models in vision and language have revolutionized natural language processing and computer vision through large-scale pretraining, foundation models for SDM hold similar potential by capturing the structure and semantics of decision trajectories. This thesis addresses this challenge from the perspective of representation learning, specifically, how to learn compact yet expressive state and action abstractions that are well suited for downstream policy learning in embodied agents. To this end, it explores both state and action representations and further introduces a surprisingly simple yet effective approach that leverages explicit visual prompting to harness the grounding capabilities of modern vision-language-action (VLA) foundation models, bridging the gap between perception and action.

In the first part of the thesis, a temporal contrastive learning objective, TACO, is proposed for visual representation learning. This method enables the learned embeddings to encode control-relevant dynamics in a compact latent space, significantly improving data efficiency during policy learning. When used for pretraining, these representations allow embodied agents to generalize to novel tasks with minimal expert demonstrations. Building on this idea of future latent prediction, the approach is further scaled to recent large VLA models through FLARE, which augments the standard action prediction objective with a future latent alignment loss. This extension achieves

state-of-the-art policy learning performance on multitask benchmarks and enables learning from action-free human video data.

In addition to state representation, the second part of the thesis investigates how temporal action representations can be leveraged for more efficient policy learning through horizon reduction. Inspired by the recent success of large language models (LLMs), I develop PRISE, a simple yet effective framework for learning temporally abstracted action representations. By capturing higher-level temporal structure, this approach shortens the effective planning horizon, substantially improving the performance of multitask imitation learning algorithms and enhancing generalization to unseen tasks with limited demonstrations.

Finally, beyond learning-based state and action representations, the last part of the thesis explores how symbolic representations can further enhance the efficiency of policy learning in embodied agents. Pretrained on diverse internet-scale vision and text data, recent vision-language models (VLMs) possess strong visual and semantic understanding capabilities but struggle to ground their perception in executable 3D robot actions. I demonstrate that symbolic representations, such as visual traces, can help these large VLMs bridge perception and action generation. To this end, I introduce TraceVLA, an explicit visual prompting technique that encodes a robot’s execution history as a symbolic visual trace. This representation provides large VLA models with richer spatio-temporal context, leading to more robust generalization across embodiments and outperforming existing VLA baselines on various real-world robotic manipulation tasks.

Together, these works present a unified framework for world-model-based representation learning for embodied agents. By jointly advancing state, action, and symbolic abstractions, this thesis takes a step toward scalable foundation models for sequential decision-making, capable of reasoning, acting, and learning across diverse tasks and embodiments.

TOWARDS GENERALIST EMBODIED AGENTS VIA REPRESENTATION LEARNING

by

Ruijie Zheng

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2025

Advisory Committee:

Professor Furong Huang, Chair/Advisor
Professor Hal Daumé, Co-Advisor
Professor Min Wu, Dean's Representative
Professor Jia-Bin Huang
Professor Ruohan Gao

© Copyright by
Ruijie Zheng
2025

Acknowledgments

I am deeply grateful to my advisor, Professor Furong Huang, who has mentored me since my sophomore year, when she first welcomed me as an undergraduate researcher. From the beginning, her insightful questions and rigorous discussions taught me what it truly means to do research — how to identify meaningful research problems and, more importantly, how to approach them from the right angle. Throughout this journey, her incisive questions, high standards, and thoughtful feedback have continuously guided my research, and I will carry the lessons I have learned from her throughout my career. Her mentorship and research vision have profoundly shaped my research taste and mindset as a scientist.

I also want to sincerely thank my co-advisor, Professor Hal Daumé III, for his generous mentorship and the many inspiring discussions we have had. I have thoroughly enjoyed every meeting with him — his sharp feedback, deep insights, and thoughtful advice on both research and career development have been invaluable. Working with both of them has been one of the most rewarding parts of my Ph.D. journey, and I will carry their lessons with me throughout my career.

I would like to express my deepest gratitude to the members of my Ph.D. committee, Professor Jia-Bin Huang, Min Wu, and Ruohan Gao, for their invaluable time, feedback, and guidance on my thesis. I am also grateful to my collaborators and internship mentors at Microsoft and NVIDIA, Dr. Huazhe Xu, Dr. Jieyu Zhao, Dr. Shuang Ma, Dr. Ching-An Cheng, Dr. Andrey Kolobov, Dr. Adith Swaminathan, Dr. Jianwei Yang, Dr. Jim Fan, and Dr. Yuke Zhu, for their mentorship and inspiration.

I feel truly fortunate to have been surrounded by inspiring peers, supportive labmates, and brilliant collaborators. Their camaraderie, insightful discussions, and encouragement have made my graduate experience both enriching and memorable. Special thanks to Jiahao Su, Yanchao Sun, Yuancheng Xu, Xiaoyu Liu, Bang An, Sicheng Zhu, Xiyao Wang, Michael-Andrei Panaitescu-Liess, Chenghao Deng, Marco Bornstein, Mucong Ding, Wichayaporn Ongkamjan, Souradip Chakraborty, Tahseen Rabbani, Dehao Yuan, Yu Hou, Navita Goyal, Connor Baumler, Lingjun Zhao, Huy Nghiem, Tin Nguyen, and Sandra Sandoval for their invaluable support, collaboration, and the many moments of

learning and growth we have shared.

I am endlessly grateful for the support of my dear friends, Yuancheng Xu, Yuan Gao, Trista Cao, Shuhan Kou, Bo He, Haozhe An, Yu Hou, Jiaxin Yuan, Xijun Wang, Xiyao Wang, Yiran Xu, Peihong Yu, and Chen Chen, who have accompanied me throughout my Ph.D. journey.

I would also like to sincerely thank my girlfriend for always being there for me. More than a partner, she has been a teammate and an inspiration, standing by my side through the most challenging times. Her love, patience, and unwavering belief in me have brought comfort, strength, and joy to my life. Finally, I owe my deepest thanks to my parents, my aunt, and my cousin, whose unconditional love and faith in me made this dream possible. Without them, none of this would have been achievable.

List of Tables

2.1	Episode reward of TACO and SOTA visual RL algorithms on the image-based DMControl 1M benchmark. Results are averaged over 6 random seeds. Within the table, entries shaded represent the best performance of model-free algorithms, while text in bold signifies the highest performance across all baseline algorithms, including model-based algorithms.	9
2.2	Comparison with other objectives including SPR [122], ATC [131], and DRIML [94]	11
2.3	Offline Performance (Normalized Reward) for different offline RL methods. Results are averaged over 6 random seeds. $\pm$ captures the standard deviation over seeds. . .	12
2.4	Few-shot BC performance on DMC. Premier-TACO outperforms baselines on 10 unseen tasks. Bold = best.	17
2.5	Five-shot Behavior Cloning (BC) performance on 8 unseen MetaWorld tasks. Results averaged over 4 seeds. Bold = best.	18
2.6	Frames per second (FPS) for visual RL algorithms. B stands for batch size used in their original implementations.	23
2.7	Frames per second (FPS) for TACO with different batch size	23
2.8	Language instructions for 8 LIBERO downstream tasks.	28
3.1	Task Success Rate Breakdown for Multitask Policy on RoboCasa and GR1 Tabletop Manipulation	34
3.2	Ablation of target embedding models.	37
3.3	Action-Aware Vision Language Embedding Pre-training Dataset Statistics	41
5.1	Performance results on three SimplerEnv robot tasks under two evaluation metrics: visual matching and variant aggregation. Overall performance is the mean across all six task-metric pairs. For each TraceVLA method, we report both the raw success rate and its difference ($\uparrow$ or $\downarrow$) versus the corresponding base model.	65
5.2	Impact of line thickness on TraceVLA performance.	71
5.3	Impact of transparency on TraceVLA performance.	71
5.4	Impact of color scheme on TraceVLA performance.	72

List of Figures

2.1	Comparison of average episode reward across nine challenging tasks in Deepmind Control Suite after one million environment steps.	4
2.2	A demonstration of TACO temporal contrastive loss: Given a batch of state-action transition triples $\{(s_t^{(i)}, [a_t^{(i)}, \dots, a_{t+K-1}^{(i)}], s_{t+K}^{(i)})\}_{i=1}^N$, I first apply the state encoder and action encoder to get latent state-action encodings: $\{(z_t^{(i)}, [u_t^{(i)}, \dots, u_{t+K-1}^{(i)}], z_{t+K}^{(i)})\}_{i=1}^N$. Then I apply two different projection layers to map $(z_t^{(i)}, [u_t^{(i)}, \dots, u_{t+K-1}^{(i)}])$ and $z_{t+K}^{(i)}$ into the shared contrastive embedding space. Finally, I learn to predict the correct pairings between $(z_t, [u_t, \dots, u_{t+K-1}])$ and z_{t+K} using an InfoNCE loss.	7
2.3	(Deepmind Control Suite) Performance of TACO against two strongest model-free visual RL baselines. Results of DrQ-v2 and A-LIX are reproduced from their open-source implementations, and all results are averaged over 6 random seeds.	9
2.4	(Meta-world) Performance of TACO against DrQ-v2 and A-LIX. All results are averaged over 6 random seeds.	10
2.5	1M Performance of TACO with and without action representation	10
2.6	TACO with different batch sizes and with different component of its learning objectives removed	11
2.7	An illustration of Premier-TACO contrastive loss design. The two ‘State Encoder’s are identical, as are the two ‘Proj. Layer H ’s. One negative example is sampled from the neighbors of framework s_{t+K}	14
2.8	Difference between Premier-TACO and TACO for sampling negative examples.	15
2.9	Pretrain and Test Tasks split for Deepmind Control Suite, MetaWorld and Libero. The left figures are Deepmind Control Suite tasks and the right figures MetaWorld tasks.	16
2.10	[(W1) Versatility (W2) Efficiency] Mean success rate of 5-shot imitation learning for 8 unseen tasks in LIBERO. Results are aggregated over 4 random seeds. Bold numbers indicate the best results.	17
2.11	[(W1) Versatility] (Left) DMC: Generalization of Premier-TACO pre-trained visual representation to unseen embodiments. (Right) MetaWorld: Few-shot adaptation to unseen tasks from an unseen camera view	19
2.12	[(W3) Robustness] Premier-TACO pretrained with exploratory dataset vs. Premier-TACO pretrained with randomly collected dataset	20
2.13	[(W4) Compatibility] Finetune R3M [96], a generalized Pretrained Visual Encoder with Premier-TACO learning objective vs. R3M with in-domain finetuning in Deepmind Control Suite and MetaWorld.	20

2.14	[(W1) Versatility] (Left) Premier-TACO vs. TACO on 10 Deepmind Control Suite Tasks across different batch sizes. (Right) Averaged performance of Premier-TACO on 10 Deepmind Control Suite Tasks across different window sizes	21
2.15	1M Performance of TACO with step size $K = 1$ and $K = 3$	22
2.16	1M Performance of TACO with different prediction horizon K under different action repeat sizes. Shaded columns correspond to the best prediction horizon K under a fixed action repeat size.	22
2.17	1M Performance of TACO with different dimensionality of latent action representations on Quadruped Run ($ \mathcal{A} = 12$). Error bar represents standard deviation across 8 random seeds.	25
2.18	Robustness analysis of TACO vs. DrQ-v2 across multiple (6) random seeds	26
2.19	[(W4) Compatibility] Finetune R3M [96], a generalized Pretrained Visual Encoder with Premier-TACO learning objective vs. R3M with in-domain finetuning in Deepmind Control Suite and MetaWorld.	26
2.20	[(W4) Compatibility] Finetune R3M [96], a generalized Pretrained Visual Encoder with Premier-TACO learning objective vs. R3M with in-domain finetuning in Deepmind Control Suite and MetaWorld.	27
3.1	Comparison of FLARE to a conventional flow-matching (or diffusion) policy. FLARE can train using both action flow-matching and future latent alignment objectives, leading to improved performance as well as enabling learning from video-only data such as human ego-view demonstrations.	30
3.2	FLARE architecture. State and action token embeddings are concatenated into a sequence with learnable future token embeddings. The flow matching DiT blocks perform self-attention on this sequence, and cross-attention to the current vision and text observation embeddings. At a middle layer, the activations corresponding to the future token embeddings are used to compute a future latent alignment loss, which is the cosine similarity with vision-language embeddings from a future observation.	31
3.3	Data mixture of pretrained action-aware vision language embedding model	32
3.4	Multitask Simulation Benchmarks: I use 24 RoboCasa [97] and 24 GR-1 tabletop manipulation tasks as a multitask simulation benchmark suite in this chapter.	33
3.5	Real GR1 Tasks Setup: I evaluate four tabletop manipulation tasks on a real GR1 humanoid robot.	35
3.6	(Left): Post-training results on 24 RoboCasa tasks. (Right): Post-training results on 4 Real GR1 humanoid tasks.	35
3.7	Generalizing to unseen objects with human egocentric videos and few-shot real robot demos	36
3.8	(Left): Ablation of the DiT Layer used in FLARE loss (Left): Ablation of FLARE loss coefficient.	37
3.9	Effect of EMA Coefficient ρ: I report the policy success rate using 24×300 training trajectories across 24 RoboCasa tasks. Baseline is trained without FLARE future alignment loss, <i>i.e.</i> , a policy-only objective.	38
3.10	Q-former based Vision Language Embedding Module for FLARE	40

4.1	(a) Pretraining Stage I of PRISE : The goal is to learn a action quantization module such that conditioned on the current state and action (o_t, a_t) , it could assign a discrete action code. (b) Pretraining Stage II of PRISE : First it converts a trajectory of continuous state and actions into discrete codes. Then based on the corpus of quantized trajectories from the multitask offline dataset, PRISE applies BPE (illustrated in Figure 4.2) to learn vocabulary of skill tokens, where each token represents a sequence of discrete action code.	45
4.2	An illustration of how Byte Pair Encoding is done in PRISE	47
4.3	During evaluation time, PRISE rollout its policy by querying the skill-token policy π for the skill token and then using pretrained decoder ψ to decode actions.	47
4.4	PRISE tokenizes downstream demonstration trajectories by greedily searching for the longest token for each time step.	48
4.5	Few-shot IL on unseen tasks: pretrain and test tasks split for MetaWorld and LIBERO.	50
4.6	Multitask policy learning performance on LIBERO-90 . Error bar represents the standard deviation across 4 random seeds. Results of the first three methods are taken directly from [85]	51
4.7	(Metaworld): 5-shot IL performance averaged across 5 unseen tasks. Error bar represents the standard deviation across 4 random seeds.	52
4.8	(LIBERO): 5-shot IL performance averaged across 8 unseen tasks. Error bar stands for the standard deviation across 4 random seeds.	53
4.9	(Left) Mean success rate of PRISE across 8 unseen LIBERO tasks with different vocabulary size. (Right) The mean success rate of PRISE across 8 unseen LIBERO tasks with a different numbers of discrete action codes pre-trained.	54
4.10	(Left) Histogram of token length when $C = 10$. (Right) Histogram of token length when $C = 40$	54
4.11	Performance of PRISE with and without the forward dynamics objective on Meta-world and LIBERO.	55
4.12	Downstream performance of PRISE under different pretraining dataset sizes	55
4.13	Pseudocode for the pretraining stage I of PRISE	58
4.14	Architecture of the Shallow CNN encoder used in MetaWorld.	59
5.1	An illustration of our method. The first image shows the original robot’s observation, while the second contains the same image with overlaid visual traces. A separator token is then inserted between the visual tokens of these two images, then concatenating with text tokens and feeding into the underlying vision language model backbone to output action tokens.	61
5.2	An illustration of visual trace generation. Given a sequence of historical image observations, I first use Co-tracker to extract dense point trajectories and keep active point trajectories with significant movement. I then overlay active point trajectories on the robot’s initial observation frame as visual trace prompting. I feed both the image overlaid with visual traces and the original image into VLA as model input.	63
5.3	Performance comparison of TraceVLA against OpenVLA (Left): 7B TraceVLA vs. 7B OpenVLA. (Right): 4B TraceVLA-Phi3 vs. 4B OpenVLA-Phi3. Numbers are averaged across the visual matching and variant aggregation metrics.	66

5.4	Comparison of OpenVLA and TraceVLA performance across various environmental variations: camera orientations, lighting, background, distractors, and table texture.	66
5.5	Real robot setup of TraceVLA. I design 4 real-world robot tasks with different manipulation skills and objects.	67
5.6	Four unseen tasks for testing TraceVLA generalization in real robot settings. . . .	67
5.7	Performance comparison of TraceVLA and OpenVLA on 8 real-world WidowX-250 robot manipulation tasks.	68
5.8	(Left): Comparison of average success rates between the base OpenVLA and OpenVLA-Phi3 models and their finetuned versions, with and without visual trace prompting. (Right): Comparison of average success rates between the base OpenVLA, TraceVLA , and OpenVLA finetuned with a sequence of 6 images.	69
5.9	(Left): Comparing visual trace prompting and text trace prompting. (Right) Text trace prompts example.	69
5.10	TraceVLA under different length of visual traces.	70
5.11	(Left): Comparison of GPU memory cost of 7B TraceVLA , OpenVLA and 4B TraceVLA-Phi3 , OpenVLA-Phi3. (Right): Comparison of inference time across different models.	71
5.12	Comparison of TraceVLA against OpenVLA with different steps of observation history.	72
5.13	Four additional tasks for testing generalization in real robot settings.	73
5.14	Number of successful trials for TraceVLA against OpenVLA on four additional real robot experiments.	73

List of Figures

Acknowledgements	ii
List of Tables	iii
List of Figures	iv
1 Introduction	1
2 Temporal Contrastive Observation Representation Learning	4
2.0.1 Introduction	4
2.1 Preliminaries	6
2.1.1 Visual reinforcement learning	6
2.1.2 Contrastive learning and the InfoNCE objective	6
2.2 Temporal contrastive learning objectives and practical implementation	6
2.3 Experiments and results	8
2.3.1 Comparison between TACO and strong baselines in online RL tasks	8
2.3.2 Combining TACO with offline RL algorithms	12
2.4 Related work	13
2.4.1 Contrastive learning in visual reinforcement learning	13
2.4.2 Action representation in reinforcement learning	14
2.5 From TACO to Premier-TACO: Better Performance with Smaller Batch Size	14
2.6 Experiment	15
2.7 Supplemental Materials: Additional Details and Experiments	22
2.7.1 Effects of the choice of prediction horizon K	22
2.7.2 Time-efficiency of TACO	23
2.7.3 Experiment Details	23
2.7.4 Sensitivity of TACO to action representation dimensionality: an additional experiment	25
2.7.5 Sensitivity of TACO vs. DrQ-v2 to random seeds	25
2.8 Additional Experiments Results of Premier-TACO	26
2.8.1 Implementation Details of Premier-TACO	26
2.8.2 Task instructions of downstream LIBERO tasks	28
3 FLARE: Future Latent Representation Alignment	29
3.1 Introduction	29
3.2 Background	30

3.3	Method	31
3.3.1	Latent World Modeling through Future Latent Representation Alignment	31
3.3.2	Action-aware Future Embedding Model	32
3.4	Experiments	33
3.4.1	Multitask Benchmark Performance	33
3.4.2	Data-efficient Post-training with Cross-embodiment Pretrained Embedding Model	34
3.4.3	Leveraging Human Egocentric Trajectories without Action Labels	36
3.4.4	Ablation Study	37
3.5	Related Work	38
3.6	Conclusion	39
3.7	Supplementary Material: Additional Details and Experiments	40
3.7.1	Details of Q-former based Vision Language Embedding Module	40
3.7.2	Pretraining Data Mixture	40
3.8	Training Details	40
3.8.1	Pseudocode of FLARE	42
4	PRISE: From Latent State Representations to Temporal Action Abstractions	43
4.1	Introduction	43
4.2	Preliminaries	44
4.3	Algorithm	45
4.3.1	Pretraining	45
4.3.2	Multitask generalist policy learning.	47
4.3.3	Downstream few-shot adaptation to unseen tasks	48
4.3.4	Discussion	49
4.4	Experiments	49
4.4.1	Pretraining datasets and architecture	50
4.4.2	Multitask generalist policy learning	50
4.4.3	Few-shot adaptation to unseen tasks	51
4.4.4	Ablation Analyses	53
4.5	Related Work	56
4.6	Supplementary Material: Additional Details and Experiments	58
4.6.1	Implementation details of PRISE	58
5	Visual Trace Prompting: From Neural Latent Representations To Explicit Visual Cues	60
5.1	Introduction	60
5.2	Preliminaries	62
5.3	TraceVLA	62
5.3.1	Visual Trace Prompting	62
5.3.2	Model Architecture Design	64
5.3.3	Implementation Details	64
5.4	Experiment	65
5.4.1	Simulation Evaluation	65
5.4.2	Real Robot Experiments	67
5.4.3	Ablation Studies	68

5.5	Limitation Analysis: Training Memory Cost and Inference Speed	70
5.6	Supplementary Material: Additional Details and Experiments	71
5.6.1	Additional Ablation Studies of TraceVLA : Thickness, Transparency, and Color of Visual Prompting	71
5.6.2	Baseline with Different Steps of Historical Observations	72
5.6.3	Additional Real Robot Experiments	72
6	Conclusion and Future Direction	75
6.1	Conclusion	75
6.2	Future Work	75
	Bibliography	77

Chapter 1: Introduction

In our dynamic and ever-evolving world, embodied agents for sequential decision-making (SDM) stand at the core of intelligent behavior in machine learning systems. Unlike static perception tasks, SDM captures the fluidity of real-world processes, ranging from robotic manipulation to adaptive healthcare, where each action influences future outcomes.

Just as foundation models in vision and language (e.g., BERT [31], DINO [102], SAM [117], GPT-4o [100]) have transformed natural language processing and computer vision by learning from large-scale data, pretrained foundation models hold similar promise for SDM by capturing the structure and semantics of decision trajectories.

In vision and language, these models serve as powerful priors, encoding rich contextual information that transfers effectively to a wide range of downstream tasks. Analogously, in SDM, decisions unfold over time and depend on complex dependencies between past actions, current observations, and future goals. A foundation model can provide a generalized understanding of behavior. This knowledge, distilled from diverse task distributions and decision sequences, can be fine-tuned to new domains in a data-efficient manner, mirroring the adaptability of language models to specialized tasks.

However, several key challenges differentiate SDM from conventional pretraining domains:

- **(C1) Data Distribution Shift:** Training data often stems from specific behavior policies, resulting in mismatched distributions during pretraining, fine-tuning, and deployment. This mismatch can significantly degrade performance [79].
- **(C2) Task Heterogeneity:** Unlike vision or language data that share common semantics, SDM tasks vary drastically in transition dynamics, state and action spaces, and objectives, making unified modeling difficult.
- **(C3) Data Quality and Supervision:** High-quality, expert-annotated decision data is scarce and expensive [13], constraining the scalability of representation learning in real-world SDM applications.

In response to these challenges, this thesis articulates three desiderata for SDM foundation models:

- **(W1) Versatility** across task configurations, embodiments, and observation modalities;
- **(W2) Efficiency** in adapting to new tasks with minimal labeled data (e.g., few-shot learning);
- **(W3) Robustness** to noisy or imperfect data sources, enabling generalization under inconsistent supervision.

To this end, my research investigates representation learning for generalist embodied agents, aiming to build models that scale with diverse embodied data while remaining adaptable to downstream tasks. Specifically, this thesis focuses on learning compact yet expressive state and action abstractions that capture control-relevant information across varied environments, unified under the principle of world-model-based latent representation learning.

A world-model-based approach is particularly well suited for this goal because, despite the diversity of embodiments, sensors, and actuation spaces, all embodied agents share the same underlying physical laws and interaction dynamics. By learning to model these dynamics rather than task-specific action policies, a world model can develop a transferable understanding of how actions influence future states. For example, a suboptimal or failed human demonstration may not directly teach an effective control strategy, yet it still provides valuable cues about the causal structure of the environment—why a failure occurred and what sequence of actions led to it. Such understanding enables the agent to reason about physical cause and effect, forming the foundation for generalization across tasks, embodiments, and modalities.

Furthermore, this thesis adopts a latent representation perspective on world modeling. Compared to explicit world models that predict in the raw observation space, such as pixel-level image or video generation, learning world models in a compact latent space allows the system to focus on high-level, semantically meaningful dynamics while filtering out irrelevant details such as texture, lighting, or background noise. This abstraction not only improves sample efficiency and stability but also leads to representations that are more aligned with the control objectives of embodied agents. This overarching principle begins with learning latent state representations that capture the predictive structure of the environment. In Chapter 2, I introduce TACO [179] and its extension, Premier-TACO [177], which employ temporal contrastive learning objectives to encode control-relevant dynamics into compact visual embeddings. These representations substantially improve data efficiency in downstream policy learning and, when used for pretraining, enable embodied agents to generalize to novel tasks with minimal expert demonstrations.

Building on this idea of future latent prediction, I demonstrate in Chapter 3 that the same principle can be scaled to large vision-language-action (VLA) foundation models through FLARE [178]. FLARE augments the standard action flowmatching prediction objective with a future latent alignment loss, effectively grounding large multimodal models in predictive state dynamics. This approach achieves state-of-the-art multitask policy learning performance and, notably, enables learning directly from action-free human videos, bridging the gap between human observation and robot execution.

The notion of latent prediction further extends naturally from states to actions. In Chapter 4, I show that similar world-model-based objectives can be leveraged to learn temporally abstracted action representations, which help reduce the effective planning horizon in complex decision problems. PRISE [174] discretizes and tokenizes raw robot trajectories to uncover higher-level temporal structure, inspired by the success of large language models (LLMs) in modeling long-range dependencies. By shortening the planning horizon, this approach significantly improves the efficiency and generalization of multitask imitation learning with limited demonstrations.

Beyond these learned latent representations, this thesis also explores how symbolic representations can complement neural ones in bridging perception and action. In Chapter 5, I present TraceVLA [175], a simple yet powerful visual prompting technique that encodes a robot’s execution history as a symbolic visual trace. When integrated into modern VLA backbones, this explicit representation provides a richer spatio-temporal grounding, enhancing the model’s ability to connect

visual perception with executable robot actions. Empirically, TraceVLA achieves more than a 3.5× improvement over existing VLA baselines on real-robot manipulation tasks and exhibits robust generalization across embodiments and scenarios.

Chapter 2: Temporal Contrastive Observation Representation Learning

In this chapter, I focus on state representation learning for embodied agent tasks using high-dimensional inputs, such as pixel-based observations. Learning in these environments is particularly difficult when rewards are sparse or expert demonstrations are limited. To overcome these challenges, rather than depending solely on reward modeling or imitation learning, I leverage the rich dynamics inherent in the sequential decision making data. I propose a simple yet effective method called the Temporal Action-Driven Contrastive (**TACO**) objective, which learns meaningful state representations through contrastive learning. Although **TACO** achieves strong performance, it requires large batch sizes due to an inefficient negative sampling strategy. This becomes a major challenge when scaling up the approach for pretraining general visual representations. To overcome this challenge, I introduce Premier-TACO, which builds on **TACO** to enable a more efficient representation pretraining without relying on large batches of negative samples.

2.0.1 Introduction

Developing reinforcement learning (RL) agents that can perform complex continuous control tasks from high-dimensional observations, such as pixels, has been a longstanding challenge. A central aspect of this challenge lies in addressing the sample inefficiency problem in visual RL. Despite significant progress in recent years [76, 77, 158, 157, 43, 42, 45, 46, 49, 146], there remains a considerable gap in sample efficiency between RL with physical state-based features and those with pixel inputs. This disparity is particularly pronounced in complex tasks; thus tackling it is crucial for advancing visual RL algorithms and enabling their practical application in real-world scenarios. State representation learning has become an essential aspect of RL research, aiming to improve sample efficiency and enhance agent performance. Initial advancements like CURL [76] utilized a self-supervised contrastive InfoNCE objective [141] for state representation, yet it overlooks the temporal dynamics of the environment. Subsequent works, including CPC [53], ST-DIM [6], and ATC [131], made progress to rectify this by integrating temporal elements into the contrastive loss, linking pairs of observations with short temporal intervals. The objective here was to develop a state representation capable of effectively predicting future observations. A more comprehensive approach was taken by DRIML [94], which incorporated the first action of the action sequence into the temporal contrastive learning framework. However, these methods, while innovative, have their shortcomings. The positive relations in

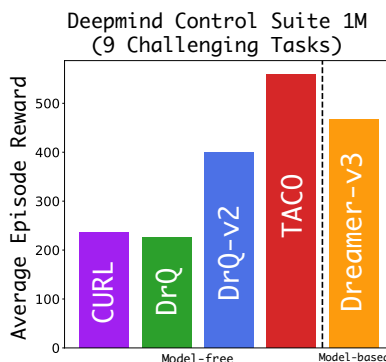


Figure 2.1: Comparison of average episode reward across nine challenging tasks in Deepmind Control Suite after one million environment steps.

their contrastive loss designs are often policy-dependent, potentially leading to instability during policy updates throughout the training process. Consequently, they lack the theoretical foundation needed to capture all information representing the optimal policy. Furthermore, these methods except for CURL and ATC typically focus on environments such as Atari games with well-represented, abstract discrete action spaces, thereby overlooking the importance of action representation in continuous control tasks [5, 57]. However, by learning an action representation that groups semantically similar actions together in the latent action space, the agent can better generalize its knowledge across various state-action pairs, enhancing the sample efficiency of RL algorithms. Therefore, learning both state and action representations is crucial for enabling the agent to more effectively reason about its actions’ long-term outcomes for continuous control tasks.

I introduce **Temporal Action-driven Contrastive Learning (TACO)** as a promising approach to visual continuous control tasks. **TACO** simultaneously learns a state and action representation by optimizing the mutual information between representations of current states paired with action sequences and representations of the corresponding future states. By optimizing the mutual information between state and action representations, **TACO** can be theoretically shown to capture the essential information to represent the optimal value function. In contrast to approaches such as DeepMDP [39] and SPR [122], which directly model the latent environment dynamics, our method transforms the representation learning objective into a self-supervised InfoNCE objective. This leads to more stable optimization and requires minimal hyperparameter tuning efforts. Consequently, **TACO** yields expressive and concise state-action representations that are better suited for high-dimensional continuous control tasks.

I demonstrate the effectiveness of representation learning by **TACO** through extensive experiments on the DeepMind Control Suite (DMC) in both online and offline RL settings. **TACO** is a flexible plug-and-play module that could be combined with any existing RL algorithm. In the online RL setting, combined with the strong baseline DrQ-v2[157], **TACO** significantly outperforms the SOTA model-free visual RL algorithms, and it even surpasses the strongest model-based visual RL baselines such as Dreamer-v3 [46]. As shown in Figure 2.1, across nine challenging visual continuous control tasks from DMC, **TACO** achieves a 40% performance boost after one million environment interaction steps on average. For offline RL, **TACO** can be combined with existing strong offline RL methods to further improve performance. When combined with TD3+BC [36] and CQL [75], **TACO** outperforms the strongest baselines across offline datasets with varying quality. I list our contributions as follows:

1. I present **TACO**, a simple yet effective temporal contrastive learning framework that simultaneously learns state and action representations.
2. The framework of **TACO** is flexible and could be integrated into both online and offline visual RL algorithms with minimal changes to the architecture and hyperparameter tuning efforts.
3. I theoretically show that the objectives of **TACO** is sufficient to capture the essential information in state and action representation for control.
4. Empirically, I show that **TACO** outperforms prior state-of-the-art model-free RL by 1.4x on nine challenging tasks in Deepmind Control Suite. Applying TACO to offline RL with SOTA algorithms also achieves significant performance gain in 4 selected challenging tasks with pre-collected offline datasets of various quality.

2.1 Preliminaries

2.1.1 Visual reinforcement learning

Let $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ be a Markov Decision Process (MDP). Here, $\mathcal{S}$ is the state space, and $\mathcal{A}$ is the action space. The state transition kernel is denoted by $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$, where $\Delta(\mathcal{S})$ is a distribution over the state space. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function. The objective of the Reinforcement Learning (RL) algorithm is the identification of an optimal policy $\pi^* : \mathcal{S} \rightarrow \Delta(\mathcal{A})$ that maximizes the expected value $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$. Additionally, I can define the optimal Q function as follows: $Q^*(s, a) = E_{\pi^*}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a]$, such that the relationship between the optimal Q function and optimal policy is $\pi(s) = \arg \max_a Q^*(s, a)$. In the domain of visual RL, high-dimensional image data are given as state observations, so the simultaneous learning of both representation and control policy becomes the main challenge. This challenge is exacerbated when the environment interactions are limited and the reward signal is sparse.

2.1.2 Contrastive learning and the InfoNCE objective

Contrastive Learning, a representation learning approach, imposes similarity constraints on representations, grouping similar/positive pairs and distancing dissimilar/negative ones within the representation space. Contrastive learning objective is often formulated through InfoNCE loss [141] to maximize the mutual information between representations of positive pairs by training a classifier. In particular, let X, Y be two random variables. Given an instance $x \sim p(x)$, and a corresponding positive sample $y^+ \sim p(y|x)$ as well as a collection of $Y = \{y_1, \dots, y_{N-1}\}$ of $N - 1$ random samples from the marginal distribution $p(y)$, the InfoNCE loss is defined as

$$\mathcal{L}_N = \mathbb{E}_x \left[\log \frac{f(x, y^+)}{\sum_{y \in Y \cup \{y^+\}} f(x, y)} \right] \quad (2.1)$$

Optimizing this loss will result in $f(x, y) \propto \frac{p(y|x)}{p(y)}$ and one can show that InfoNCE loss upper bounds the mutual information, $\mathcal{L}_N \geq \log(N) - \mathcal{I}(X, Y)$.

2.2 Temporal contrastive learning objectives and practical implementation

In the following, I present the learning objectives of **TACO**. The guiding principle of our method is to learn state and action representations that capture the essential information about the environment's dynamics sufficient for learning the optimal policy. This allows the agent to develop a concise and expressive understanding of both its current state and the potential effects of its actions, thereby enhancing sample efficiency and generalization capabilities.

Let S_t, A_t be the state and action variables at timestep t , $Z_t = \phi(S_t)$, $U_t = \psi(A_t)$ be their corresponding representations. Then, our method aims to maximize the mutual information between representations of current states paired with action sequences and representations of the corresponding future states: $\mathbb{I}_{\text{TACO}} = \mathcal{I}(Z_t, U_t : Z_{t+H})$. In practice, I estimate the lower bound of the mutual information by the InfoNCE loss. (2.2)

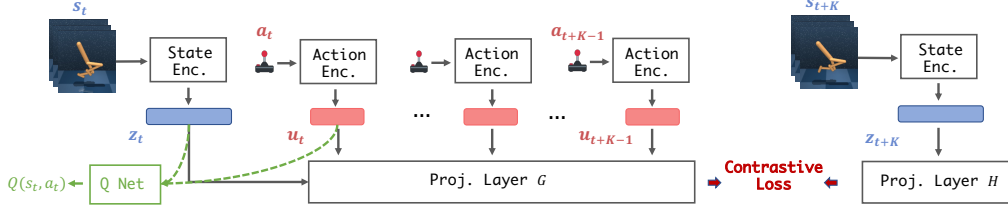


Figure 2.2: A demonstration of **TACO** temporal contrastive loss: Given a batch of state-action transition triples $\{(s_t^{(i)}, [a_t^{(i)}, \dots, a_{t+K-1}^{(i)}], s_{t+K}^{(i)})\}_{i=1}^N$, I first apply the state encoder and action encoder to get latent state-action encodings: $\{(z_t^{(i)}, [u_t^{(i)}, \dots, u_{t+K-1}^{(i)}], z_{t+K}^{(i)})\}_{i=1}^N$. Then I apply two different projection layers to map $(z_t^{(i)}, [u_t^{(i)}, \dots, u_{t+K-1}^{(i)}])$ and $z_{t+K}^{(i)}$ into the shared contrastive embedding space. Finally, I learn to predict the correct pairings between $(z_t, [u_t, \dots, u_{t+K-1}])$ and z_{t+K} using an InfoNCE loss.

Next, provide a detailed description of the practical implementation of **taco**. In Figure 2.2, I illustrate the architecture design of **TACO**. Our approach minimally adapts a base RL algorithm by incorporating the temporal contrastive loss as an auxiliary loss during the batch update process. Specifically, given a batch of state and action sequence transitions $\{(s_t^{(i)}, [a_t^{(i)}, \dots, a_{t'-1}^{(i)}], s_{t'}^{(i)})\}_{i=1}^N$ ($t' = t + K$), I optimize:

$$\mathcal{J}_{\text{TACO}}(\phi, \psi, W, G_\theta, H_\theta) = -\frac{1}{N} \sum_{i=1}^N \log \frac{g_t^{(i)\top} W h_{t'}^{(i)}}{\sum_{j=1}^N g_t^{(i)\top} W h_{t'}^{(j)}} \quad (2.3)$$

Here let $z_t^{(i)} = \phi(s_t^{(i)})$, $u_t^{(i)} = \psi(a_t^{(i)})$ be state and action embeddings respectively. $g_t^{(i)} = G_\theta(z_t^{(i)}, u_t^{(i)}, \dots, u_{t'-1}^{(i)})$, and $h_{t'}^{(i)} = H_\theta(z_{t'}^{(i)})$, where G_θ and H_θ denote two learnable projection layers that map the latent state $z_t^{(i)}$ as well as latent state and action sequence $(z_t^{(i)}, u_t^{(i)}, \dots, u_{t'-1}^{(i)})$ to a common contrastive embedding space. W is a learnable parameter providing a similarity measure between g_i and h_j in the shared contrastive embedding space. Subsequently, both state and action representations are fed into the agent's Q network, allowing the agent to effectively reason about the long-term effects of their actions and better leverage their past experience through state-action abstractions.

In addition to the main **TACO** objective, in our practical implementation, I find that the inclusion of two auxiliary objectives yields further enhancement in the algorithm's overall performance. The first is the CURL [76] loss:

$$\mathcal{J}_{\text{CURL}}(\phi, \psi, W, H_\theta) = -\frac{1}{N} \sum_{i=1}^N \log \frac{h_t^{(i)\top} W h_t^{(i)+}}{h_t^{(i)\top} W h_t^{(i)+} + \sum_{j \neq i} h_t^{(i)\top} W h_t^{(j)}} \quad (2.4)$$

Here, $h_t^{(i)+} = H_\theta(\phi(s_i^{(t)+}))$, where $s_i^{(t)+}$ is the augmented view of $s_i^{(t)}$ by applying the same random shift augmentation as DrQ-v2 [157]. W and H_θ share the same weight as the ones in **TACO** objective. The third objective is reward prediction:

$$\mathcal{J}_{\text{Reward}}(\phi, \psi, \hat{R}_\theta) = \sum_{i=1}^N (\hat{R}_\theta(z_t^{(i)}, u_t^{(i)}, \dots, u_{t'-1}^{(i)}) - r^{(i)})^2 \quad (2.5)$$

Here $r^{(i)} = \sum_{j=t}^{t'-1} r_j^{(i)}$ is the sum of reward from timestep t to $t' - 1$, and $\hat{R}_\theta$ is a reward prediction layer. For our final objective, I combine the three losses together with equal weight. As verified in Section 2.3.1, although **TACO** serves as the central objective that drives notable performance improvements, the inclusion of both CURL and reward prediction loss can further improve the algorithm’s performance.

I have opted to use DrQ-v2 [157] for the backbone algorithm of **TACO**, although in principle, **TACO** could be incorporated into any visual RL algorithms. **TACO** extends DrQ-v2 with minimal additional hyperparameter tuning. The only additional hyperparameter is the selection of the prediction horizon K . Throughout our experiments, I have limited our choice of K to either 1 or 3, depending on the nature of the environment.

2.3 Experiments and results

This section provides an overview of our empirical evaluation, conducted in both online and offline RL settings. To evaluate our approach under online RL, I apply **TACO** to a set of nine challenging visual continuous control tasks from Deepmind Control Suite (DMC) [135]. Meanwhile, for offline RL, I combine **TACO** with existing offline RL methods and test the performance on four DMC tasks, using three pre-collected datasets that differ in the quality of their data collection policies.

2.3.1 Comparison between **TACO** and strong baselines in online RL tasks

Environment Settings: In our online RL experiment, I first evaluate the performance of **TACO** on nine challenging visual continuous control tasks from Deepmind Control Suite [135]: Quadruped Run, Quadruped Walk, Hopper Hop, Reacher Hard, Walker Run, Acrobot Swingup, Cheetah Run, Finger Turn Hard, and Reach Duplo. These tasks demand the agent to acquire and exhibit complex motor skills and present challenges such as delayed and sparse reward. As a result, these tasks have not been fully mastered by previous visual RL algorithms, and require the agent to learn an effective policy that balances exploration and exploitation while coping with the challenges presented by the tasks.

In addition to Deepmind Control Suite, I also present the results of **TACO** on additional six challenging robotic manipulation tasks from Meta-world [164]: Hammer, Assembly, Disassemble, Stick Pull, Pick Place Wall, Hand Insert. Unlike the DeepMind Control Suite, which primarily concentrates on locomotion tasks, Meta-world domain provides tasks that involve complex manipulation and interaction tasks. This sets it apart by representing a different set of challenges, emphasizing precision and control in fine motor tasks rather than broader locomotion skills.

Baselines: I compare **TACO** with four model-free visual RL algorithms **CURL** [76], **DrQ** [158], **DrQ-v2** [157], and **A-LIX** [20]. **A-LIX** builds on **DrQ-v2** by adding adaptive regularization to the encoder’s gradients. While **TACO** could also extend **A-LIX**, our reproduction of results from its open-source implementation does not consistently surpass **DrQ-v2**. As such, I do not choose **A-LIX** as the backbone algorithm for **TACO**. Additionally, I compare with two state-of-the-art model-based RL algorithms for visual continuous control, **Dreamer-v3** [46] and **TDMPC** [49], which learn world models in latent space and select actions using either model-predictive control or a learned policy. **TACO** achieves a significantly better sample efficiency and performance compared with SOTA visual RL algorithm. The efficacy of **TACO** is evident from the findings presented in Figure 2.3

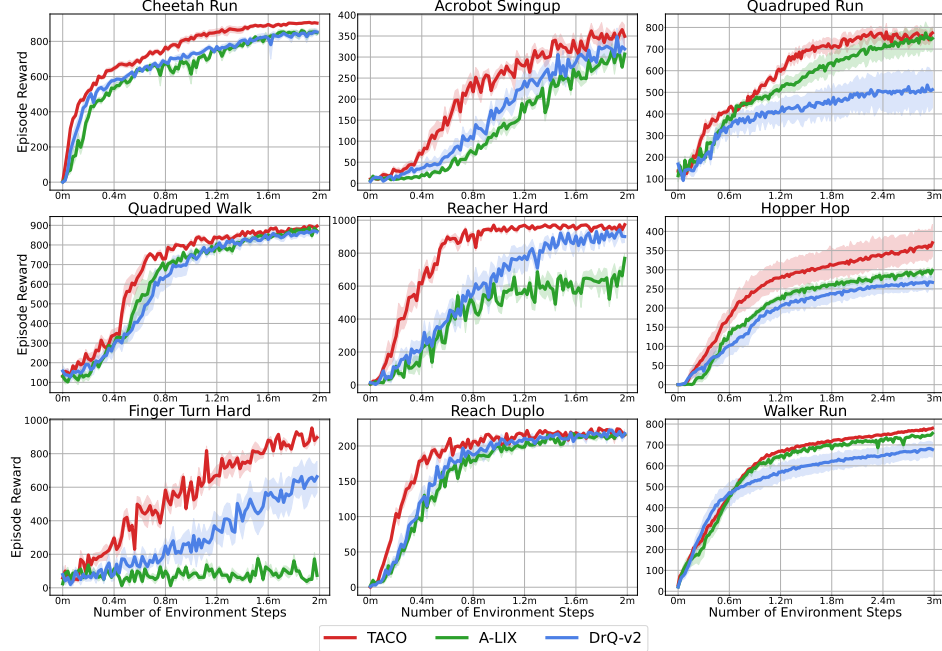


Figure 2.3: (Deepmind Control Suite) Performance of **TACO** against two strongest model-free visual RL baselines. Results of DrQ-v2 and A-LIX are reproduced from their open-source implementations, and all results are averaged over 6 random seeds.

Table 2.1: Episode reward of **TACO** and SOTA visual RL algorithms on the image-based DMControl 1M benchmark. Results are averaged over 6 random seeds. Within the table, **entries shaded** represent the best performance of model-free algorithms, while **text in bold** signifies the highest performance across all baseline algorithms, including model-based algorithms.

Environment (1M Steps)	<i>Model-free</i>					<i>Model-based</i>	
	TACO	DrQv2	A-LIX	DrQ	CURL	Dreamer-v3	TDMPC
Quadruped Run	541 ± 38	407 ± 21	454 ± 42	179 ± 18	181 ± 14	331 ± 42	397 ± 37
Hopper Hop	261 ± 52	189 ± 35	225 ± 13	192 ± 41	152 ± 34	369 ± 21	195 ± 18
Walker Run	637 ± 11	517 ± 43	617 ± 12	451 ± 73	387 ± 24	765 ± 32	600 ± 28
Quadruped Walk	793 ± 8	680 ± 52	560 ± 175	120 ± 17	123 ± 11	353 ± 27	435 ± 16
Cheetah Run	821 ± 48	691 ± 42	676 ± 41	474 ± 32	657 ± 35	728 ± 32	565 ± 61
Finger Turn Hard	632 ± 75	220 ± 21	62 ± 54	91 ± 9	215 ± 17	810 ± 58	400 ± 113
Acrobot Swingup	241 ± 21	128 ± 8	112 ± 23	24 ± 8	5 ± 1	210 ± 12	224 ± 20
Reacher Hard	883 ± 63	572 ± 51	510 ± 16	471 ± 45	400 ± 29	499 ± 51	485 ± 31
Reach Duplo	234 ± 21	206 ± 32	199 ± 14	36 ± 7	8 ± 1	119 ± 30	117 ± 12

(DMC) , Table 2.1 (DMC), and Figure 2.4 (Meta-world). In contrast to preceding model-free visual RL algorithms, **TACO** exhibits considerably improved sample efficiency. For example, on the challenging Reacher Hard task, **TACO** achieves optimal performance in just 0.75 million environment steps, whereas DrQ-v2 requires approximately 1.5 million steps. When trained with only 1 million environment steps, **TACO** on average achieves 40% better performance, and it is even better than the model-based visual RL algorithms Dreamer-v3 and TDMPC on 6 out of 9 tasks. In addition, on more demanding tasks such as Quadruped Run, Hopper Hop, Walker Run, and Cheetah Run, **TACO** continues to outshine competitors, exhibiting superior overall performance

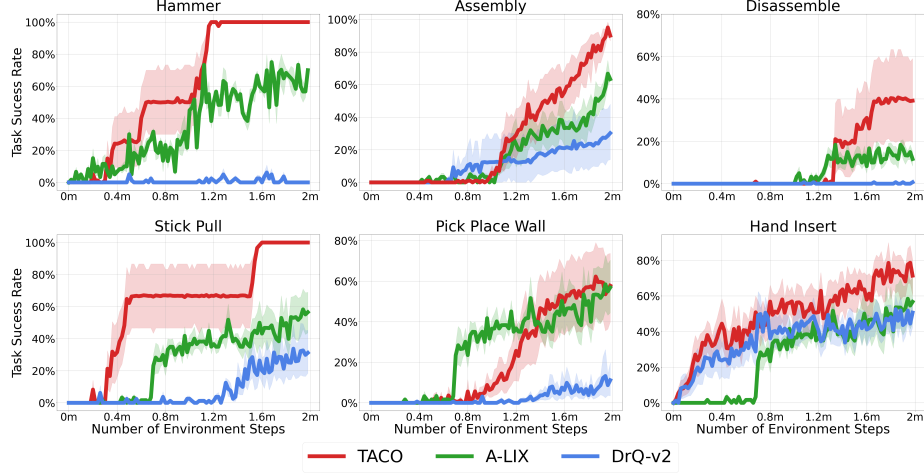


Figure 2.4: (Meta-world) Performance of **TACO** against DrQ-v2 and A-LIX. All results are averaged over 6 random seeds.

after two or three million steps, as illustrated in Figure 2.3. For robotic manipulation tasks, as shown in Figure 2.4, **TACO** also significantly outperform the baseline model-free visual RL algorithms, highlighting the broad applicability of **TACO**.

Concurrently learning state and action representation is crucial for the success of TACO. To demonstrate the effectiveness of action representation learning in **TACO**, I evaluate its performance on a subset of 4 difficult benchmark tasks and compare it with a baseline method without action representation, as shown in Figure 2.5. The empirical results underscore the efficacy of the temporal contrastive learning objectives, even in the absence of action representation. For instance, **TACO** records an enhancement of 18% on Quadruped Run and a substantial 51% on Reacher Hard, while the remaining tasks showcase a performance comparable to DrQ-v2. Furthermore, when comparing against **TACO** without action representation, **TACO** achieves a consistent performance gain, ranging from 12.2% on Quadruped Run to a significant 70.4% on Hopper Hop. These results not only emphasize the inherent value of the temporal contrastive learning objective in **TACO**, but also underscore the instrumental role of high-quality action representation in bolstering the performance of the underlying RL algorithms.

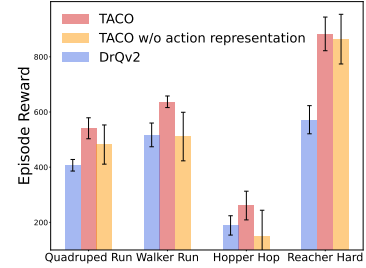
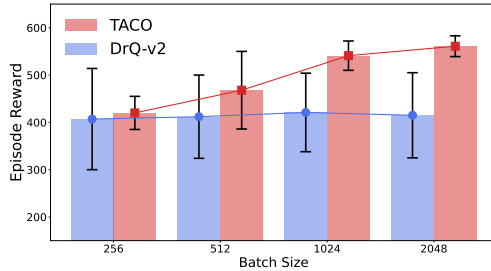
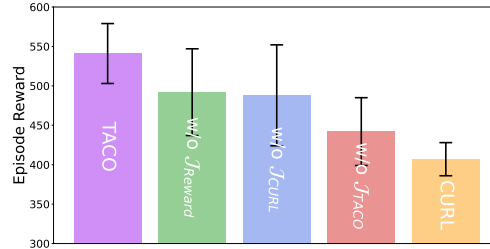


Figure 2.5: 1M Performance of **TACO** with and without action representation

The effectiveness of our temporal contrastive loss is enhanced with a larger batch size. As is widely acknowledged in contrastive learning research [25, 51, 114, 40], our contrastive loss sees significant benefits from utilizing a larger batch size. In Figure 2.6a, I illustrate the performance of our algorithms alongside DrQv2 after one million environment steps on the Quadruped Run task. As evident from the plot, batch size greatly influences the performance of our algorithm, while DrQ-v2’s baseline performance remains fairly consistent throughout training. In order to strike a balance between time efficiency and performance, I opt for a batch size of 1024, which is 4 times larger than the 256 batch size employed in DrQ-v2, but 4 times smaller than the 4096 which is commonly used in the contrastive learning literature [25, 51, 40].



(a) TACO and DrQ-v2 across different batch sizes.



(b) TACO with different learning objective removed

Figure 2.6: TACO with different batch sizes and with different component of its learning objectives removed

Reward prediction and CURL loss serves an auxiliary role to further improve the performance of TACO, while the temporal contrastive loss of TACO is the most crucial component. In the practical deployment of **TACO**, two additional objectives, namely reward prediction and CURL loss, are incorporated to enhance the algorithm’s performance. In Figure 2.6b, I remove one objective at a time on the Quadraped Run task to assess its individual impact on the performance after one million environment steps. As illustrated in the figure, the omission of **TACO**’ temporal contrastive objective results in the most significant performance drop, emphasizing its critical role in the algorithm’s operation. Meanwhile, the auxiliary reward prediction and CURL objectives, although secondary, contribute to performance improvement to some degree.

InfoNCE-based temporal action-driven contrastive objective in TACO outperforms other representation learning objectives including SPR [124], ATC [131], and DRIML [94]. In Table 2.2, I have showcased a comparison between our approach and other visual RL representation learning objectives such as **SPR**, **ATC**, and **DRIML**. Given that **SPR** and **DRIML** were not initially designed for continuous control tasks, I have re-implemented their learning objectives using the identical backbone algorithm, DrQ-v2. A similar approach was taken for ATC, with their learning objectives also being reimplemented on DrQ-v2 to ensure a fair comparison. (Without the DrQ-v2 backbone algorithm, the performance reproduced by their original implementation is significantly worse.) Furthermore, recognizing the significance of learning action encoding, as discussed earlier, I have integrated action representation learning into all these baselines. Therefore, the model architecture remains consistent across different representation learning objectives, with the sole difference being the design of the temporal contrastive loss. For **DRIML**, given that only the first action of the action sequence is considered in the temporal contrastive loss, **TACO** and **DRIML** differ when the number of steps K is greater than one. Thus, I indicate N/A for tasks where I choose $K = 1$ for **TACO**.

Table 2.2: Comparison with other objectives including SPR [122], ATC [131], and DRIML [94]

Environment	TACO	SPR	ATC	DRIML	DrQ-v2
Quadraped Run	541 ± 38	448 ± 79	432 ± 54	N/A	407 ± 21
Walker Run	637 ± 21	560 ± 71	502 ± 171	N/A	517 ± 43
Hopper Hop	261 ± 52	154 ± 10	112 ± 98	216 ± 13	192 ± 41
Reacher Hard	883 ± 63	711 ± 92	863 ± 12	835 ± 72	572 ± 51
Acrobot Swingup	241 ± 21	198 ± 21	206 ± 61	222 ± 39	210 ± 12

Table 2.2 showcases that while previous representation learning objectives have proven benefit in assisting the agent to surpass the DrQ-v2 baseline by learning a superior representation, our

approach exhibits consistent superiority over other representation learning objectives in all five evaluated environments. These results reinforce our claim that **TACO** is a more effective method for learning state-action representations, allowing agents to reason more efficiently about the long-term outcomes of their actions in the environment.

2.3.2 Combining **TACO** with offline RL algorithms

In this part, I discuss the experimental results of **TACO** within the context of offline reinforcement learning, emphasizing the benefits our temporal contrastive state/action representation learning objective brings to visual offline RL. Offline visual reinforcement learning poses unique challenges, as algorithms must learn an optimal policy solely from a fixed dataset without further interaction with the environment. This necessitates that the agent effectively generalizes from limited data while handling high-dimensional visual inputs. The state/action representation learning objective of **TACO** plays a vital role in addressing these challenges by capturing essential information about the environment’s dynamics, thereby enabling more efficient generalization and improved performance. **TACO** can be easily integrated as a plug-and-play module on top of existing strong offline RL methods, such as **TD3+BC** [36] and **CQL** [75].

For evaluation, I select four challenging visual control tasks from DMC: Hopper Hop, Cheetah Run, Walker Run, and Quadruped Run. For each task, I generate three types of datasets. The **medium** dataset consists of trajectories collected by a single policy of medium performance. The precise definition of “medium performance” is task-dependent but generally represents an intermediate level of mastery, which is neither too poor nor too proficient. The **medium-replay** dataset contains trajectories randomly sampled from the online learning agent’s replay buffer before it reaches a medium performance level. The **full-replay** dataset includes trajectories randomly sampled throughout the online learning phase, from the beginning until convergence. The dataset size for Walker, Hopper, and Cheetah is 100K, while for the more challenging Quadruped Run task, a larger dataset size of 500K is used to account for the increased difficulty. I compute the normalized reward by dividing the offline RL reward by the best reward I get during online **TACO** training.

Table 2.3: Offline Performance (Normalized Reward) for different offline RL methods. Results are averaged over 6 random seeds. $\pm$ captures the standard deviation over seeds.

Task/ Dataset		TD3+BC w. TACO	TD3+BC	CQL w. TACO	CQL	DT	IQL	BC
Hopper Hop	Medium	52.4 $\pm$ 0.4	51.2 $\pm$ 0.8	47.9 $\pm$ 0.6	46.7 $\pm$ 0.2	40.5 $\pm$ 3.6	2.0 $\pm$ 1.7	48.2 $\pm$ 0.8
	Medium-replay	67.2 $\pm$ 0.1	62.9 $\pm$ 0.1	74.6 $\pm$ 0.4	68.7 $\pm$ 0.1	65.3 $\pm$ 1.8	57.6 $\pm$ 1.4	25.9 $\pm$ 3.2
	Full-replay	97.6 $\pm$ 1.4	83.8 $\pm$ 2.3	101.2 $\pm$ 1.9	94.2 $\pm$ 2.0	92.4 $\pm$ 0.3	47.7 $\pm$ 2.2	65.7 $\pm$ 2.7
Cheetah Run	Medium	66.6 $\pm$ 0.5	66.1 $\pm$ 0.3	70.1 $\pm$ 0.4	66.7 $\pm$ 1.7	64.3 $\pm$ 0.7	1.7 $\pm$ 1.1	62.9 $\pm$ 0.1
	Medium-replay	62.6 $\pm$ 0.2	61.1 $\pm$ 0.1	72.3 $\pm$ 1.2	67.3 $\pm$ 1.1	67.0 $\pm$ 0.6	26.5 $\pm$ 3.2	48.0 $\pm$ 3.1
	Full-replay	92.5 $\pm$ 2.4	91.2 $\pm$ 0.8	86.9 $\pm$ 2.4	65.0 $\pm$ 3.9	89.6 $\pm$ 1.4	14.6 $\pm$ 3.7	69.0 $\pm$ 0.3
Walker Run	Medium	49.2 $\pm$ 0.5	48.0 $\pm$ 0.2	49.6 $\pm$ 1.0	49.4 $\pm$ 0.9	47.3 $\pm$ 0.3	4.4 $\pm$ 0.4	46.2 $\pm$ 0.6
	Medium-replay	63.1 $\pm$ 0.6	62.3 $\pm$ 0.2	62.3 $\pm$ 2.6	59.9 $\pm$ 0.9	61.7 $\pm$ 1.1	41.4 $\pm$ 2.8	18.5 $\pm$ 0.8
	Full-replay	86.8 $\pm$ 0.6	84.0 $\pm$ 1.6	88.1 $\pm$ 0.1	79.8 $\pm$ 0.6	81.6 $\pm$ 0.8	18.1 $\pm$ 3.7	30.8 $\pm$ 1.8
Quadruped Run	Medium	60.6 $\pm$ 0.1	60.0 $\pm$ 0.2	58.1 $\pm$ 3.7	55.9 $\pm$ 9.1	14.6 $\pm$ 3.8	0.8 $\pm$ 0.8	56.2 $\pm$ 1.1
	Medium-replay	61.3 $\pm$ 0.3	58.1 $\pm$ 0.5	61.9 $\pm$ 0.2	61.2 $\pm$ 0.9	19.5 $\pm$ 2.2	58.4 $\pm$ 4.4	51.6 $\pm$ 3.3
	Full-replay	92.6 $\pm$ 0.7	89.3 $\pm$ 0.4	92.1 $\pm$ 0.1	85.2 $\pm$ 2.5	14.5 $\pm$ 1.1	36.3 $\pm$ 5.9	57.6 $\pm$ 0.7
Average Normalized Score		71.0	68.2	72.1	66.7	48.4	25.8	54.9

I compare the performance of **TD3+BC** and **CQL** with and without **TACO** on our benchmark.

Additionally, I also compare with the decision transformer (**DT**) [24], a strong model-free offline RL baseline that casts the problem of RL as conditional sequence modeling, **IQL** [71], another commonly used offline RL algorithm, and the behavior cloning (**BC**) baseline. For **TD3+BC**, **CQL** and **IQL**, which were originally proposed to solve offline RL with vector inputs, I add the their learning objective on top of DrQ-v2 to handle image inputs.

Table 2.3 provides the normalized reward for each dataset. The results underscore that when combined with the strongest baselines **TD3+BC** and **CQL**, **TACO** achieves consistent performance improvements across all tasks and datasets, setting new state-of-the-art results for offline visual reinforcement learning. This is true for both the medium dataset collected with a single policy and narrow data distribution, as well as the medium-replay and replay datasets with a diverse distribution.

2.4 Related work

2.4.1 Contrastive learning in visual reinforcement learning

Contrastive learning has emerged as a powerful technique for learning effective representations across various domains, particularly in computer vision [141, 25, 51, 53, 151]. This success is attributed to its ability to learn meaningful embeddings by contrasting similar and dissimilar data samples. In visual reinforcement learning, it’s used as a self-supervised auxiliary task to improve state representation learning, with InfoNCE [141] being a popular learning objective. In CURL [76], it treats augmented states as positive pairs, but it neglects the temporal dependency of MDP. CPC [141], ST-DIM [6], and ATC [131] integrate temporal relationships into the contrastive loss by maximizing mutual information between current state representations (or state histories encoded by LSTM in CPC) and future state representations. However, they do not consider actions, making positive relationships in the learning objective policy-dependent. DRIML [94] addresses this by maximizing mutual information between state-action pairs at the current time step and the resulting future state, but its objective remains policy-dependent as it only provides the first action of the action sequence. Besides, ADAT [65] and ACO [169] incorporate actions into contrastive loss by labeling observations with similar policy action outputs as positive samples, but these methods do not naturally extend to tasks with non-trivial continuous action spaces. A common downside of these approaches is the potential for unstable encoder updates due to policy-dependent positive relations. In contrast, **TACO** is theoretically sufficient, and it tackles the additional challenge of continuous control tasks by simultaneously learning state and action representations.

In addition to the InfoNCE objective, other self-supervised learning objective is also proposed. Approaches such as DeepMDP [39], SPR [122], SGI [124], and EfficientZero [161] directly learn a latent-space transition model. Notably, these methods predominantly target Atari games characterized by their small, well-represented, and abstract discrete action spaces. When dealing with continuous control tasks, which often involve a continuous and potentially high-dimensional action space, the relationships between actions and states become increasingly intricate. This complexity poses a significant challenge in effectively capturing the underlying dynamics. In contrast, by framing the latent dynamics model predictions as a self-supervised InfoNCE objective, the mutual information guided approach used by **TACO** is better suited for continuous control task, resulting in more stable optimization and thus better state and action representations.

2.4.2 Action representation in reinforcement learning

Although state or observation representations are the main focus of prior research, there also exists discussion on the benefits and effects of learning action representations. Chandak et al. [21] propose to learn a policy over latent action space and transform the latent actions into actual actions, which enables generalization over large action sets. Allshire et al. [5] introduce a variational encoder-decoder model to learn disentangled action representation, improving the sample efficiency of policy learning. In model-based RL, strategies to achieve more precise and stable model-based planning or roll-out are essential. To this end, Park et al. [104] propose an approach to train an environment model in the learned latent action space. In addition, action representation also has the potential to improve multi-task learning [54], where latent actions can be shared and enhance generalization.

2.5 From TACO to Premier-TACO: Better Performance with Smaller Batch Size

Building upon the success of **TACO** in acquiring latent state representations that encapsulate individual task dynamics, our goal in the following sections is to learn a generic embodied agent’s observation representation that effectively captures the intrinsic dynamics spanning a diverse set of tasks found in offline datasets. Our overarching objective is to ensure that these learned representations exhibit the versatility to generalize across unseen tasks that share the underlying dynamic structures.

Nevertheless, when adapted for multitask offline pre-training, the original learning objective of **TACO** poses a notable challenge. Specifically, **TACO**’s mechanism, which utilizes the InfoNCE [141] loss, categorizes all subsequent states s_{t+k} in the batch as negative examples. While this methodology has proven effective in single-task reinforcement learning scenarios, it encounters difficulties when extended to a multitask context. During multitask offline pretraining, image observations within a batch can come from different tasks with vastly different visual appearances, rendering the InfoNCE loss significantly less effective.

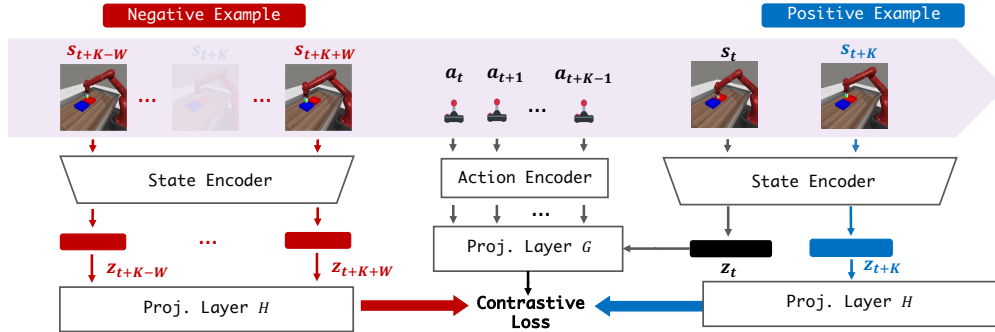


Figure 2.7: An illustration of **Premier-TACO** contrastive loss design. The two ‘State Encoder’*s* are identical, as are the two ‘Proj. Layer *H*’*s*. One negative example is sampled from the neighbors of framework s_{t+K} .

To deal with this challenge, I propose a straightforward yet highly effective mechanism for selecting challenging negative examples. Instead of treating all the remaining examples in the batch as negatives, **Premier-TACO** selects the negative example from a window centered at state s_{t+k} within the same episode as shown in Figure 2.8.

This approach is both computationally efficient and more statistically powerful due to negative examples which are challenging to distinguish from similar positive examples forcing the model capture temporal dynamics differentiating between positive and negative examples. In practice, this allows us to use much smaller batch sizes for **Premier-TACO**. On MetaWorld, with only $\frac{1}{8}$ of the batch size (512 vs. 4096), **Premier-TACO** achieves a 25% performance gain compared to TACO, saving around 87.5% of computational time.

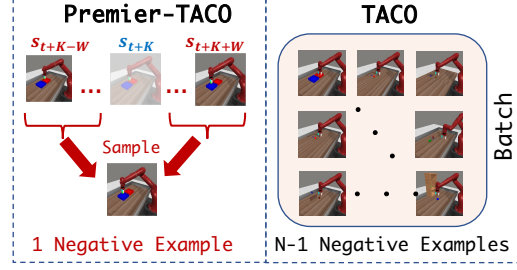


Figure 2.8: Difference between **Premier-TACO** and TACO for sampling negative examples.

In Figure 2.7, I illustrate the design of **Premier-TACO** objective. Specifically, given a batch of state and action sequence transitions $\{(s_t^{(i)}, [a_t^{(i)}, \dots, a_{t+K-1}^{(i)}], s_{t+K}^{(i)})\}_{i=1}^N$, let $z_t^{(i)} = \phi(s_t^{(i)})$, $u_t^{(i)} = \psi(a_t^{(i)})$ be latent state and latent action embeddings respectively. Furthermore, let $\widetilde{s_{t+K}^{(i)}}$ be a negative example uniformly sampled from the window of size W centered at s_{t+K} : $(s_{t+K-W}, \dots, s_{t+K-1}, s_{t+K+1}, \dots, s_{t+K+W})$ with $\widetilde{z_t^{(i)}} = \phi(\widetilde{s_{t+K}^{(i)}})$ a negative latent state.

Given these, define $g_t^{(i)} = G_\theta(z_t^{(i)}, u_t^{(i)}, \dots, u_{t+K-1}^{(i)})$, $\widetilde{h_t^{(i)}} = H_\theta(\widetilde{z_{t+K}^{(i)}})$, and $h_t^{(i)} = H_\theta(z_{t+K}^{(i)})$ as embeddings of future predicted and actual latent states. I optimize:

$$\mathcal{J}_{\text{Premier-TACO}}(\phi, \psi, G_\theta, H_\theta) = -\frac{1}{N} \sum_{i=1}^N \log \frac{g_t^{(i)\top} h_{t+K}^{(i)}}{g_t^{(i)\top} h_{t+K}^{(i)} + \widetilde{g_t^{(i)}}^\top h_{t+K}^{(i)}} \quad (2.6)$$

Few-shot Generalization. After pretraining the representation encoder, I leverage our pretrained model Φ to learn policies for downstream tasks. To learn the policy π with the state representation $\Phi(s_t)$ as inputs, I use behavior cloning (BC) with a few expert demonstrations. For different control domains, I employ significantly fewer demonstrations for unseen tasks than what is typically used in other baselines. This underscores the substantial advantages of **Premier-TACO** in few-shot generalization. More details about the experiments on downstream tasks will be provided in Section 5.4.

2.6 Experiment

In our empirical evaluations, I consider three benchmarks, Deepmind Control Suite [136] for locomotion control, MetaWorld [163] and LIBERO [85] for robotic manipulation tasks. It is important to note the varied sources of data employed for pretraining in these benchmarks. For the Deepmind Control Suite, our pretraining dataset comes from the replay buffers of online reinforcement learning (RL) agents. In MetaWorld, the dataset is generated through a pre-defined scripted policy. In LIBERO, I utilize its provided demonstration dataset, which was collected through human teleoperation. By evaluating on a wide range of pretraining data types that have been explored in previous works, I aim to provide a comprehensive evaluation for the pretraining effects of **Premier-TACO**.

Deepmind Control Suite (DMC) [136]: I consider a selection of 16 challenging tasks from Deepmind Control Suite. Note that compared with prior works such as [92, 132], I consider much harder tasks, including ones from the humanoid and dog domains, which feature intricate kinematics,

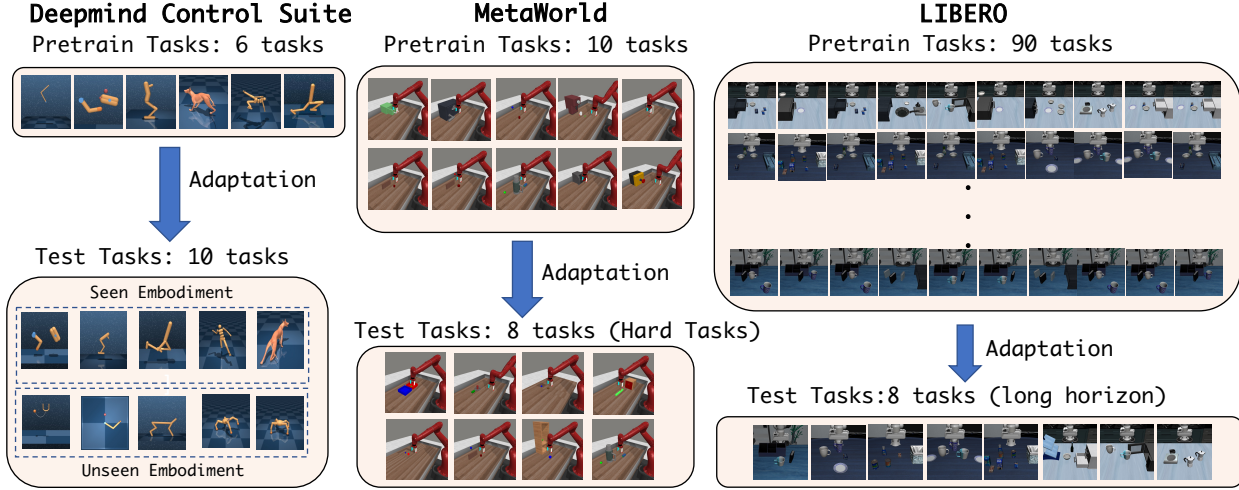


Figure 2.9: Pretrain and Test Tasks split for Deepmind Control Suite, MetaWorld and Libero. The left figures are Deepmind Control Suite tasks and the right figures MetaWorld tasks.

skinning weights and collision geometry. For pretraining, I select six tasks (**DMC-6**), including Acrobot Swingup, Finger Turn Hard, Hopper Stand, Walker Run, Humanoid Walk, and Dog Stand. I generate an exploratory dataset for each task by sampling trajectories generated in exploratory stages of a DrQ-v2 [157] learning agent. In particular, I sample 1000 trajectories from the online replay buffer of DrQ-v2 once it reaches the convergence performance. This ensures the diversity of the pretraining data, but in practice, such a high-quality dataset could be hard to obtain. So, later in the experiments, I will also relax this assumption and consider pretrained trajectories that are sampled from uniformly random actions. In terms of the encoder architecture, I pretrain **Premier-TACO** with the same shallow ConvNet encoder as in DrQv2 [157].

MetaWorld [163]: I select a set of 10 tasks for pretraining, which encompasses a variety of motion patterns of the Sawyer robotic arm and interaction with different objects. To collect an exploratory dataset for pretraining, I execute the scripted policy with Gaussian noise of a standard deviation of 0.3 added to the action. After adding such a noise, the success rate of collected policies on average is only around 20% across ten pretrained tasks. I use the same encoder network architecture as DMC.

LIBERO [85]: I pretrain on 90 short-horizon manipulation tasks (**LIBERO-90**) with human demonstration dataset provided by the original paper. For each task, it contains 50 trajectories of human teleoperated trajectories. I use ResNet18 encoder [52] to encode the image observations of resolution 128×128 . For the downstream task, I assess the few-shot imitation learning performance on the first 8 long-horizon tasks of LIBERO-LONG.

Baselines. I compare **Premier-TACO** with the following representation pretraining baselines:

- ▷ Learn from Scratch: Behavior Cloning with randomly initialized shallow ConvNet encoder. I carefully implement the behavior cloning from scratch baseline. For DMC and MetaWorld, following [48], I include the random shift data augmentation into behavior cloning. For LIBERO, I take the ResNet-T model architecture in [85], which uses a transformer decoder module on top of the ResNet encoding to extract temporal information from a sequence of observations, addressing the non-Markovian characteristics inherent in human demonstration.

	DMControl	LfS	PVRs	TD3+BC	Inv.	CURL	ATC	SPR	TACO	Premier-TACO
Seen	Finger Spin	34.8 ± 3.4	38.4 ± 9.3	68.8 ± 7.1	33.4 ± 8.4	35.1 ± 9.6	51.1 ± 9.4	55.9 ± 6.2	28.4 ± 9.7	75.2 ± 0.6
	Hopper Hop	8.0 ± 1.3	23.2 ± 4.9	49.1 ± 4.3	48.3 ± 5.2	28.7 ± 5.2	34.9 ± 3.9	52.3 ± 7.8	21.4 ± 3.4	75.3 ± 4.6
	Walker Walk	30.4 ± 2.9	32.6 ± 8.7	65.8 ± 2.0	64.4 ± 5.6	37.3 ± 7.9	44.6 ± 5.0	72.9 ± 1.5	30.6 ± 6.1	88.0 ± 0.8
	Humanoid Walk	15.1 ± 1.3	30.1 ± 7.5	34.9 ± 8.5	41.9 ± 8.4	19.4 ± 2.8	35.1 ± 3.1	30.1 ± 6.2	29.1 ± 8.1	51.4 ± 4.9
	Dog Trot	52.7 ± 3.5	73.5 ± 6.4	82.3 ± 4.4	85.3 ± 2.1	71.9 ± 2.2	84.3 ± 0.5	79.9 ± 3.8	80.1 ± 4.1	93.9 ± 5.4
Unseen	Cup Catch	56.8 ± 5.6	93.7 ± 1.8	97.1 ± 1.7	96.7 ± 2.6	96.7 ± 2.6	96.2 ± 1.4	96.9 ± 3.1	88.7 ± 3.2	98.9 ± 0.1
	Reacher Hard	34.6 ± 4.1	64.9 ± 5.8	59.6 ± 9.9	61.7 ± 4.6	50.4 ± 4.6	56.9 ± 9.8	62.5 ± 7.8	58.3 ± 6.4	81.3 ± 1.8
	Cheetah Run	25.1 ± 2.9	39.5 ± 9.7	50.9 ± 2.6	51.5 ± 5.5	36.8 ± 5.4	30.1 ± 1.0	40.2 ± 9.6	23.2 ± 3.3	65.7 ± 1.1
	Quadruped Walk	61.1 ± 5.7	63.2 ± 4.0	76.6 ± 7.4	82.4 ± 6.7	72.8 ± 8.9	81.9 ± 5.6	65.6 ± 4.0	63.9 ± 9.3	83.2 ± 5.7
	Quadruped Run	45.0 ± 2.9	64.0 ± 2.4	48.2 ± 5.2	52.1 ± 1.8	55.1 ± 5.4	2.6 ± 3.6	68.2 ± 3.2	50.8 ± 5.7	76.8 ± 7.5
	Mean	38.2	52.3	63.3	61.7	50.4	52.7	62.4	47.5	79.0

Table 2.4: Few-shot BC performance on DMC. Premier-TACO outperforms baselines on 10 unseen tasks. Bold = best.

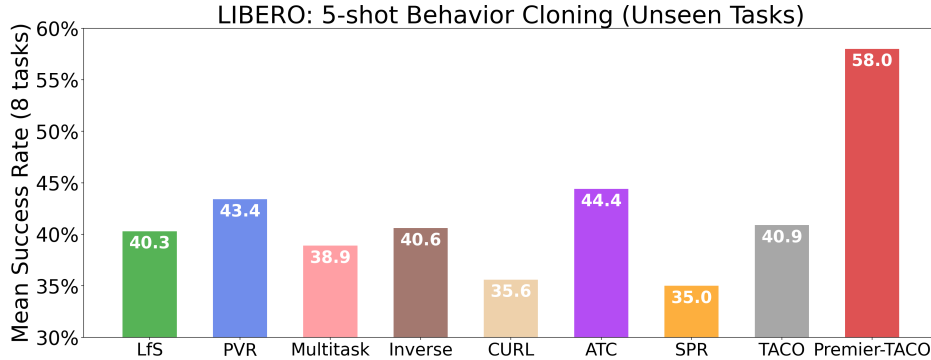


Figure 2.10: [(W1) Versatility (W2) Efficiency] Mean success rate of 5-shot imitation learning for 8 unseen tasks in LIBERO. Results are aggregated over 4 random seeds. **Bold** numbers indicate the best results.

- ▷ **Policy Pretraining:** I first train a multitask policy by TD3+BC [36] on the pretraining dataset. While numerous alternative offline RL algorithms exist, I choose TD3+BC as a representative due to its simplicity and great empirical performance. For LIBERO, I use Multitask BC since offline RL in generally does not perform well on the imitation learning benchmark with human demonstrated dataset. After pretraining, I take the pretrained ConvNet encoder and drop the policy MLP layers.
- ▷ **Pretrained Visual Representations (PVRs):** I evaluate the state-of-the-art frozen pretrained visual representations including PVR [103], MVP [152], R3M [96] and VC-1 [92], and report the best performance of these PVRs models for each task.

MetaWorld	LFS	PVRs	TD3+BC	Inv.	CURL	ATC	SPR	TACO	Premier-TACO
Bin Picking	62.5 ± 12.5	60.2 ± 4.3	50.6 ± 3.7	55.0 ± 7.9	45.6 ± 5.6	55.6 ± 7.8	67.9 ± 6.4	67.3 ± 7.5	78.5 ± 7.2
Disassemble	56.3 ± 6.5	70.4 ± 8.9	56.9 ± 11.5	53.8 ± 8.1	66.2 ± 8.3	45.6 ± 9.8	48.8 ± 5.4	51.3 ± 10.8	86.7 ± 8.9
Hand Insert	34.7 ± 7.5	35.5 ± 2.3	46.2 ± 5.2	50.0 ± 3.5	49.4 ± 7.6	51.2 ± 1.3	52.4 ± 5.2	56.8 ± 4.2	75.0 ± 7.1
Peg Insert Side	28.7 ± 2.0	48.2 ± 3.6	30.0 ± 6.1	33.1 ± 6.2	28.1 ± 3.7	31.8 ± 4.8	39.2 ± 7.4	36.3 ± 4.5	62.7 ± 4.7
Pick Out Of Hole	53.7 ± 6.7	66.3 ± 7.2	46.9 ± 7.4	50.6 ± 5.1	43.1 ± 6.2	54.4 ± 8.5	55.3 ± 6.8	52.9 ± 7.3	72.7 ± 7.3
Pick Place Wall	40.5 ± 4.5	63.2 ± 9.8	63.8 ± 12.4	71.3 ± 11.3	73.8 ± 11.9	68.7 ± 5.5	72.3 ± 7.5	37.8 ± 8.5	80.2 ± 8.2
Shelf Place	26.3 ± 4.1	32.4 ± 6.5	45.0 ± 7.7	36.9 ± 6.7	35.0 ± 10.8	35.6 ± 10.7	38.0 ± 6.5	25.8 ± 5.0	70.4 ± 8.1
Stick Pull	46.3 ± 7.2	52.4 ± 5.6	72.3 ± 11.9	57.5 ± 9.5	43.1 ± 15.2	72.5 ± 8.9	68.5 ± 9.4	52.0 ± 10.5	80.0 ± 8.1
Mean	43.6	53.6	51.5	51.0	48.3	51.9	55.3	47.5	75.8

Table 2.5: Five-shot Behavior Cloning (BC) performance on 8 unseen MetaWorld tasks. Results averaged over 4 seeds. **Bold** = best.

- ▷ Inverse Dynamics Model: I pretrain an inverse dynamics model to predict actions and use the pretrained representation for downstream task.
- ▷ Contrastive/Self-supervised Learning Objectives: CURL [76], ATC [131], SPR [122, 124]. CURL and ATC are two approaches that apply contrastive learning into sequential decision making problems.

While CURL treats augmented states as positive pairs, it neglects the temporal dependency of MDP. In comparison, ATC takes the temporal structure into consideration. The positive example of ATC is an augmented view of a temporally nearby state. SPR applies BYOL objective [40] into sequential decision making problems by pretraining state representations that are self-predictive of future states.

Pretrained feature representation by Premier-TACO facilitates effective few-shot adaptation to unseen tasks. I measure the performance of pretrained visual representation for few-shot imitation learning of unseen downstream tasks in both DMC and MetaWorld. In particular, for DMC, I use **20 expert trajectories** for imitation learning except for the two hardest tasks, Humanoid Walk and Dog Trot, for which I use 100 trajectories instead. Note that I only use $\frac{1}{5}$ of the number of expert trajectories used in [92] and $\frac{1}{10}$ of those used in [132].

I record the performance of the agent by calculating the ratio of $\frac{\text{Agent Reward}}{\text{Expert Reward}}$, where Expert Reward is the episode reward of the expert policy used to collect demonstration trajectories. For MetaWorld and LIBERO, I use **5 expert trajectories** for all downstream tasks, and I use task success rate as the performance metric. In Table 2.4 Table 2.5, and Figure 4.8 I present the results for Deepmind Control Suite, MetaWorld, and LIBERO, respectively. As shown here, pretrained representation of **Premier-TACO** significantly improves the few-shot imitation learning performance compared with learning from Scratch, with a **101%** improvement on Deepmind Control Suite and **74%** improvement on MetaWorld, respectively. Moreover, it also outperforms all the baselines across all tasks by a large margin. In LIBERO, consistent with what is observed in [85], existing pretraining methods on large-scale multitask offline dataset fail to enhance downstream policy learning performance. In particular, methods like multitask pretraining actually degrade

downstream policy learning performance. In contrast, using ResNet-18 encoders pretrained by **Premier-TACO** significantly boosts few-shot imitation learning performance by a substantial margin.

Premier-TACO pre-trained representation enables knowledge sharing across different embodiments. Ideally, a resilient and generalizable state feature representation ought not only to encapsulate universally applicable features for a given embodiment across a variety of tasks, but also to exhibit the capability to generalize across distinct embodiments. Here, I evaluate the few-shot behavior cloning performance of **Premier-TACO** pre-trained encoder from **DMC-6** on four tasks featuring unseen embodiments: Cup Catch, Cheetah Run, and Quadruped Walk. In comparison to learn from scratch, as shown in Figure 2.11 (left), **Premier-TACO** pre-trained representation realizes an **82%** performance gain, demonstrating the robust generalizability of our pre-trained feature representations.

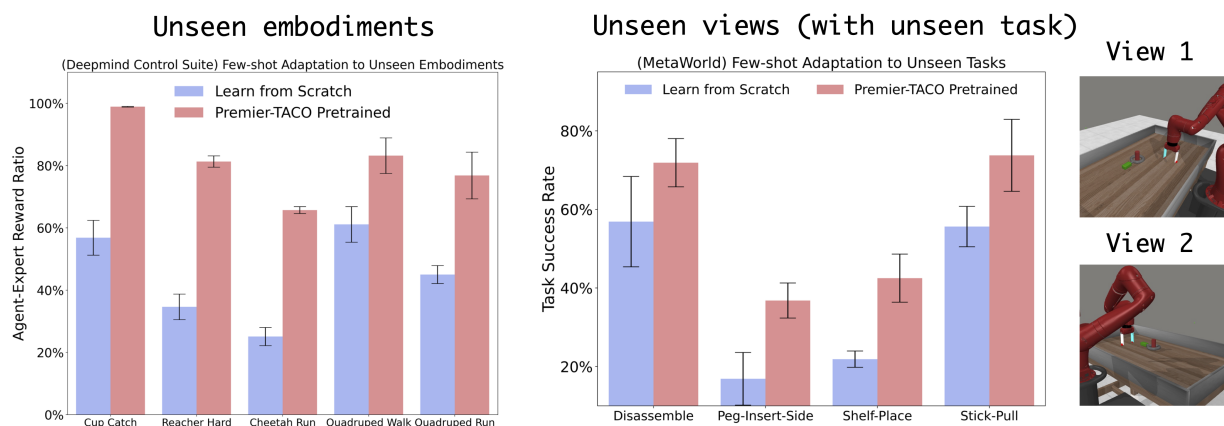


Figure 2.11: [(W1) Versatility] (Left) DMC: Generalization of **Premier-TACO** pre-trained visual representation to unseen embodiments. (Right) MetaWorld: Few-shot adaptation to unseen tasks from an unseen camera view

Premier-TACO Pretrained Representation is also generalizable to unseen tasks with camera views. Beyond generalizing to unseen embodiments, an ideal robust visual representation should possess the capacity to adapt to unfamiliar tasks under novel camera views. In Figure 2.11 (right), I evaluate the five-shot learning performance of our model on four previously unseen tasks in MetaWorld with a new view. In particular, during pretraining, the data from MetaWorld are generated using the same view as employed in [49, 127]. Then for downstream policy learning, the agent is given five expert trajectories under a different corner camera view, as depicted in the figure. Notably, **Premier-TACO** also achieves a substantial performance enhancement, thereby underscoring the robust generalizability of our pretrained visual representation.

Premier-TACO Pre-trained Representation is resilient to low-quality data. I evaluate the resilience of **Premier-TACO** by employing randomly collected trajectory data from Deepmind Control Suite for pretraining and compare it with **Premier-TACO** representations pretrained using an exploratory dataset and the learn-from-scratch approach. As illustrated in Figure 2.12, across all downstream tasks, even when using randomly pretrained data, the **Premier-TACO** pretrained model still maintains a significant advantage over learning-from-scratch. When compared with representations pretrained using exploratory data, there are only small disparities in a few individual

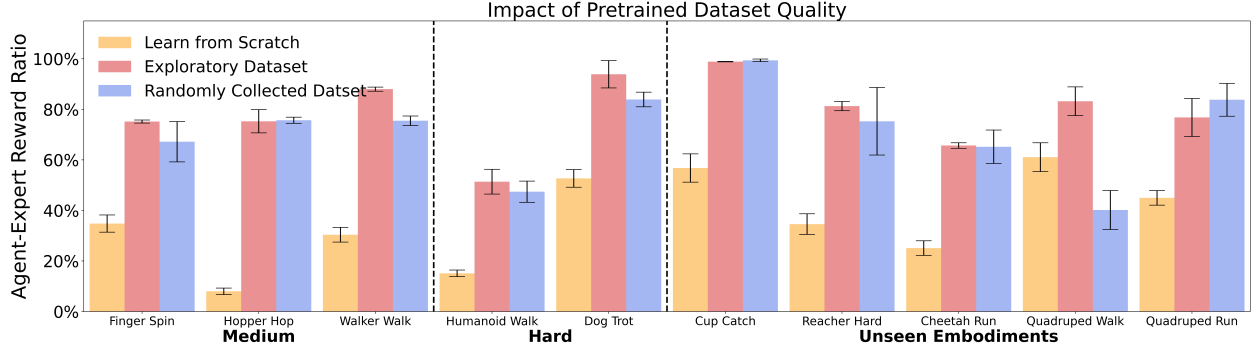


Figure 2.12: [(W3) Robustness] Premier-TACO pretrained with exploratory dataset vs. Premier-TACO pretrained with randomly collected dataset

tasks, while they remain comparable in most other tasks. This strongly indicates the robustness of **Premier-TACO** to low-quality data. Even without the use of expert control data, our method is capable of extracting valuable information.

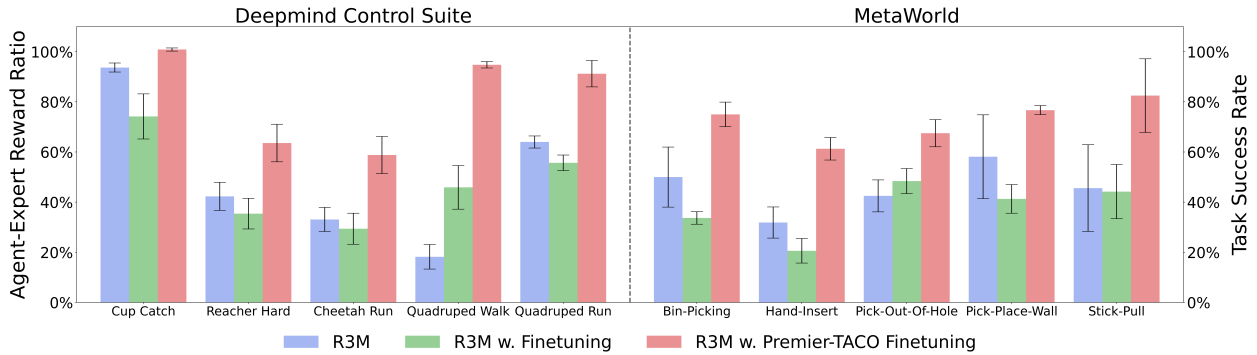


Figure 2.13: [(W4) Compatibility] Finetune R3M [96], a generalized Pretrained Visual Encoder with Premier-TACO learning objective vs. R3M with in-domain finetuning in Deepmind Control Suite and MetaWorld.

Pretrained visual encoder finetuning with Premier-TACO. In addition to evaluating our pre-trained representations across various downstream scenarios, I also conducted fine-tuning on pretrained visual representations using in-domain control trajectories following **Premier-TACO** framework. Importantly, our findings deviate from the observations made in prior works like [48] and [92], where fine-tuning of R3M [96] on in-domain demonstration data using the task-centric behavior cloning objective, resulted in performance degradation. I speculate that two main factors contribute to this phenomenon. First, a domain gap exists between out-of-domain pretraining data and in-domain fine-tuning data. Second, fine-tuning with few-shot learning can lead to overfitting for large pretrained models.

To further validate the effectiveness of our **Premier-TACO** approach, I compared the results of R3M with no fine-tuning, in-domain fine-tuning [48], and fine-tuning using our method on selected Deepmind Control Suite and MetaWorld pretraining tasks. Figure 2.13 unequivocally demonstrate that direct fine-tuning on in-domain tasks leads to a performance decline across multiple tasks. However, leveraging the **Premier-TACO** learning objective for fine-tuning substantially enhances

the performance of R3M. This not only underscores the role of our method in bridging the domain gap and capturing essential control features but also highlights its robust generalization capabilities. Furthermore, these findings strongly suggest that our **Premier-TACO** approach is highly adaptable to a wide range of multi-task pretraining scenarios, irrespective of the model’s size or the size of the pretrained data.

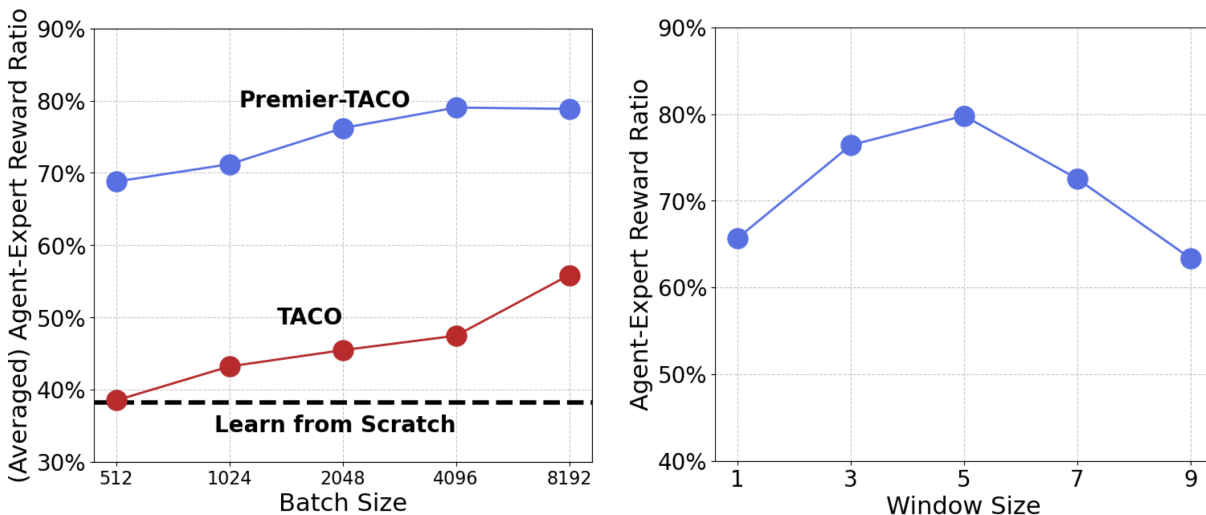


Figure 2.14: [(W1) Versatility] (Left) Premier-TACO vs. TACO on 10 Deepmind Control Suite Tasks across different batch sizes. (Right) Averaged performance of Premier-TACO on 10 Deepmind Control Suite Tasks across different window sizes

Ablation Study - Batch Size: Compared with TACO, the negative example sampling strategy employed in **Premier-TACO** allows us to sample harder negative examples within the same episode as the positive example. This implies the promising potential to significantly improve the performance of existing pretrained models across diverse domains. I expect **Premier-TACO** to work much better with small batch sizes, compared with TACO where the negative examples from a given batch could be coming from various tasks and thus the batch size required would scale up linearly with the number of pretraining tasks. In our previous experimental results, **Premier-TACO** is pretrained with a batch size of 4096, a standard batch size used in contrastive learning literature. Here, to empirically verify the effects of different choices of the pretraining batch size, I train **Premier-TACO** and TACO with different batch sizes and compare their few-shot imitation learning performance. Figure 2.14 (left) displays the average performance of few-shot imitation learning across all ten tasks in the DeepMind Control Suite. As depicted in the figure, our model significantly outperform TACO across all batch sizes tested in the experiments, and exhibits performance saturation beyond a batch size of 4096. This observation substantiate that the negative example sampling strategy employed by **Premier-TACO** is indeed the key for the success of multitask offline pretraining.

Ablation Study - Window Size: In **Premier-TACO**, the window size W determines the hardness of the negative example. A smaller window size results in negative examples that are more challenging to distinguish from positive examples, though they may become excessively difficult to differentiate in the latent space. Conversely, a larger window size makes distinguishing relatively straightforward, thereby mitigating the impacts of negative sampling. In preceding experiments, a consistent window size of 5 was applied across all trials on both the DeepMind Control Suite and MetaWorld. Here

in Figure 2.14 (right) I empirically evaluate the effects of varying window sizes on the average performance of our model across ten DeepMind Control Tasks. Notably, our observations reveal that performance is comparable when the window size is set to 3, 5, or 7, whereas excessively small ($W = 1$) or large ($W = 9$) window sizes lead to worse performance.

2.7 Supplemental Materials: Additional Details and Experiments

2.7.1 Effects of the choice of prediction horizon K

TACO functions as a versatile add-on to current online and offline RL algorithms, requiring only the prediction horizon K as an additional hyperparameter. In our investigations, I select K as either 1 or 3 across all tasks. The performance of **TACO** with K values of 1 and 3 is compared against the DrQ-v2 baselines in Figure 2.15. As illustrated in the figure, **TACO** outperforms the baseline DrQ-v2 for both $K = 1$ and 3, indicating the effectiveness of the temporal contrastive loss in concurrent state and action representation learning. In comparing $K = 1$ to $K = 3$, I observe that a longer prediction horizon ($K = 3$) yields superior results for four out of nine tasks, specifically Hopper Hop, Quadruped Walk, Acrobot Swingup, and Reacher Hard. Conversely, for the Quadruped Run and Walker Run tasks, a shorter temporal contrastive loss interval ($K = 1$) proves more beneficial. For the remaining tasks, the choice between $K = 1$ and $K = 3$ appears to have no discernable impact.

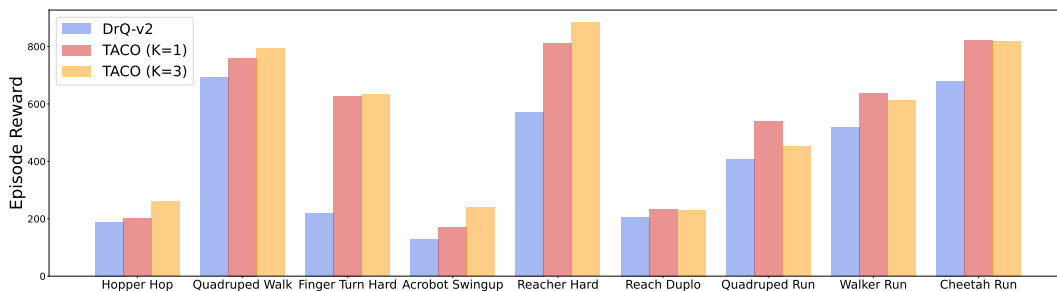


Figure 2.15: 1M Performance of **TACO** with step size $K = 1$ and $K = 3$.

Our speculation hinges on the rate of changes in the agent’s observations. While **TACO** applies regardless of the prediction horizon K , a shorter horizon (such as $K = 1$) can offer a somewhat shortsighted perspective. This becomes less beneficial in continuous control tasks where state transitions within brief time intervals are negligible. Therefore, in environments exhibiting abrupt state transitions, a larger K would be advantageous, whereas a smaller K would suffice for environments with gradual state transitions.

To substantiate this conjecture, I conduct a simple experiment on the task Hopper Hop where I aim to simulate the different rates of changes in the agent’s observations. I verify this by varying the size of action repeats, which corresponds to how many times a chosen action is repeated

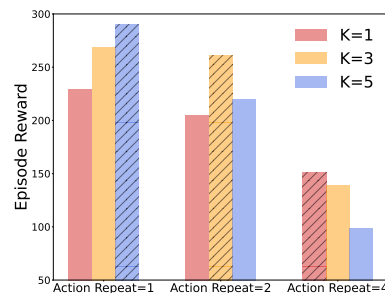


Figure 2.16: 1M Performance of **TACO** with different prediction horizon K under different action repeat sizes. Shaded columns correspond to the best prediction horizon K under a fixed action repeat size.

per environmental step. Consequently, a larger action repeat size induces more pronounced observational changes. In our prior experiments, following the settings of DrQ-v2, I fixed the size of the action repeat to be 2. In this experiment, I alter the size of the action repeat for Hopper Hop to be 1, 2, and 4.

For each action repeat size, I then compare the 1M performance of **TACO** with the prediction horizon K selected from the set $\{1, 3, 5\}$. Interestingly, as demonstrated in Figure 2.16, the optimal K values for different action repeats are reversed: 5 for an action repeat of 1, 3 for an action repeat of 2, and 1 for an action repeat of 4. This observation further substantiates our assertion that the optimal choice of prediction horizon correlates with the rate of change in environmental dynamics.

2.7.2 Time-efficiency of **TACO**

In this section, I compare the time efficiency of different visual RL algorithms. In Table 2.6, I present a comprehensive comparison of the speed of these algorithms, measured in frames per second (**FPS**). Additionally, I also provide **FPS** for **TACO** with different batch sizes in Table 2.7 as a reference. To ensure a fair comparison, all the algorithms are tested on Nvidia A100 GPUs.

As mentioned in subsection 2.3.1, the InfoNCE objective in **TACO** requires a large batch size of 1024, which is 4 times larger than the batch size used in DrQ-v2. Consequently, this increases the processing time, making our method to run about 3.6 times slower than DrQ-v2, with similar time efficiency as DrQ. Therefore, the primary limitation of our method is this time inefficiency caused by the usage of large batch sizes. Potential solutions to improve the time efficiency could involve the implementation of a distributed version of our method to speed up training. I intend to explore these improvements in future work related to our paper.

Table 2.6: Frames per second (**FPS**) for visual RL algorithms. **B** stands for batch size used in their original implementations.

TACO (B:1024)	DrQv2 (B:256)	A-LIX (B:256)	DrQ (B:512)	CURL (B:512)	Dreamer-v3 (B:256)	TDMPC (B:256)
35	130	98	33	20	31	22

Table 2.7: Frames per second (**FPS**) for **TACO** with different batch size

TACO (B:1024)	TACO (B:512)	TACO (B:256)
35	65	94

2.7.3 Experiment Details

In this section, I describe the implementation details of **TACO** for the online RL experiments. I implement **TACO** on top of the released open-source implementation of DrQ-v2, where I interleave the update of **TACO** objective with the original actor and critic update of DrQ-v2. Below I demonstrate the pseudo-code of the new update function.

```

1 ### Extract feature representation for state and actions.
2 def update(batch):

```

```

3  obs, action_sequence, reward, next_obs = batch
4  ### Update Agent's critic function
5  update_critic(obs, action_sequence, reward, next_obs)
6  ### Update the agent's value function
7  update_actor(obs)
8  ### Update TACO loss
9  update_taco(obs, action_sequence, reward, next_obs)

```

Listing 2.1: Pytorch-like pseudo-code how **TACO** is incorporated into the update function of existing visual RL algorithms.

Next, I demonstrate the pseudo-code of how **TACO** objective is computed with pytorch-like pseudocode.

```

1  # state_encoder:      State/Observation Encoder (CNN)
2  # action_encoder:     Action Encoder (MLP with 1-hidden layer)
3  # sequence_encoder:   Action Sequence encoder (Linear Layer)
4  # reward_predictor:   Reward Prediction Layer (MLP with 1-hidden layer)
5  # G:                  Projection Layer I (MLP with 1-hidden layer)
6  # H:                  Projection Layer II (MLP with 1-hidden layer)
7  # aug:                Data Augmentation Function (Radnom Shift)
8  # W:                  Matrix for computing similarity score
9
10 def compute_taco_objective(obs, action_sequence, reward, next_obs):
11     ### Compute feature representation for both state and actions.
12     z = state_encoder(aug(obs))
13     z_anchor = state_encoder(aug(obs), stop_grad=True)
14     next_z = state_encoder(aug(next_obs), stop_grad=True)
15     u_seq = sequence_encoder(action_sequence)
16     ### Project to joint contrastive embedding space
17     x = G(torch.cat([z, u_seq], dim=-1))
18     y = H(next_z)
19     ### Compute bilinear product  $x^TWy$ 
20     ### Diagonal entries of  $x^TWy$  correspond to positive pairs
21     logits = torch.matmul(x, torch.matmul(W, y.T))
22     logits = logits - torch.max(logits, 1)
23     labels = torch.arange(n)
24     taco_loss = cross_entropy_loss(logits, labels)
25
26     ### Compute CURL loss
27     x = H(z)
28     y = H(z_anchor).detach()
29     logits = torch.matmul(x, torch.matmul(W, y.T))
30     logits = logits - torch.max(logits, 1)
31     labels = torch.arange(n)
32     curl_loss = cross_entropy_loss(logits, labels)
33
34     ### Reward Prediction Loss
35     reward_pred = reward_predictor(z, u_seq)
36     reward_loss = torch.mse_loss(reward_pred, reward)
37
38     return taco_loss + curl_loss + reward_loss

```

Listing 2.2: Pytorch-like pseudo-code for how **TACO** objective is computed

Then when I compute the Q values for both actor and critic updates, I use the trained state and action encoder. Same as what is used in DrQ-v2, I use 1024 for the hidden dimension of all the encoder layers and 50 for the feature dimension of state representation. For action representation, I choose the dimensionality of the action encoding, which corresponds to the output size of the action encoding layer, as $\lceil 1.25 \times |\mathcal{A}| \rceil$. In practice, I find this works well as it effectively extracts relevant control information from the raw action space while minimizing the inclusion of irrelevant control information in the representation. See [subsection 2.7.4](#) for an additional experiment on testing the robustness of **TACO** against the dimensionality of action representation.

2.7.4 Sensitivity of **TACO** to action representation dimensionality: an additional experiment

In all of our previous experiments, I use $\lceil 1.25 \times |\mathcal{A}| \rceil$ as the latent dimensions of the action space for **TACO**. In practice, I find this works well so that it retains the rich information of the raw actions while being able to group semantically similar actions together. To test the algorithm’s sensitivity to this hyperparameter, I conduct an experiment on the Quadrupted Run task, which has a 12-dimensional action space. In [Figure 2.17](#), I show the 1M performance of **TACO** with different choices of latent action dimensions. Notably, I observe that as long as the dimensionality of the action space is neither too small (6-dimensional), which could limit the ability of latent actions to capture sufficient information from the raw actions, nor too large (24-dimensional), which might introduce excessive control-irrelevant information, the performance of **TACO** remains robust to the choice of action representation dimensionality.

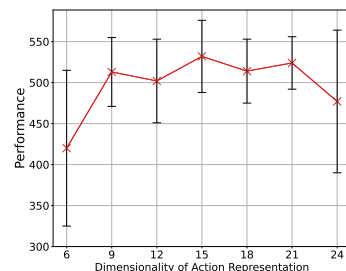


Figure 2.17: 1M Performance of **TACO** with different dimensionality of latent action representations on Quadrupted Run ($|\mathcal{A}| = 12$). Error bar represents standard deviation across 8 random seeds.

2.7.5 Sensitivity of **TACO** vs. DrQ-v2 to random seeds

In this section, I conduct a comprehensive analysis of the algorithm’s robustness under random seed, comparing the performance of our proposed method, **TACO**, with the baseline algorithm, DrQ-v2, across multiple seeds on nine distinct tasks shown in [Figure 2.3](#). Our aim is to gain insights into the stability and robustness of both algorithms under varying random seeds, providing valuable information to assess their reliability.

Remarkably, [Figure 2.18](#) demonstrates the impressive robustness of **TACO** compared to DrQ-v2. **TACO** consistently demonstrates smaller performance variations across different seeds, outperforming the baseline on almost every task. In contrast, runs of DrQ-v2 frequently encounter broken seeds. These results provide strong evidence that **TACO** not only improves average performance but also significantly enhances the robustness of the training process, making it more resilient against failure cases. This enhanced robustness is crucial for real-world applications, where stability and consistency are essential for successful deployment.

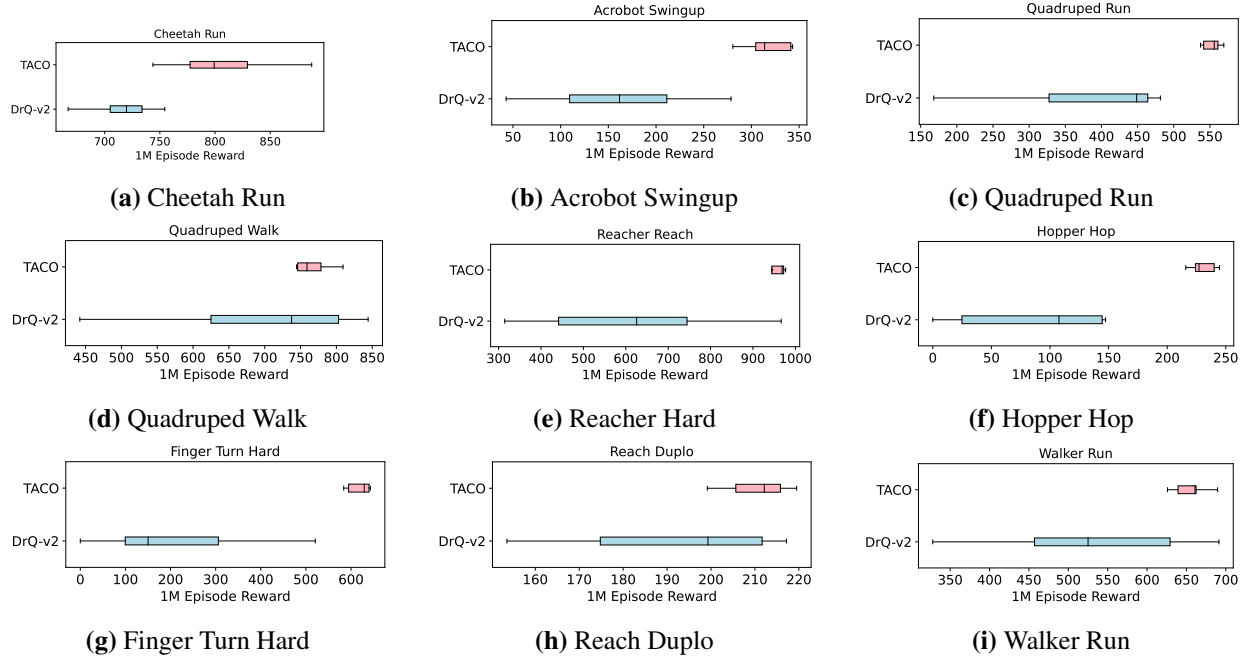


Figure 2.18: Robustness analysis of TACO vs. DrQ-v2 across multiple (6) random seeds

2.8 Additional Experiments Results of Premier-TACO

Comparisons among R3M [96], R3M with in-domain finetuning [48] and R3M finetuned with **Premier-TACO** in Deepmind Control Suite and MetaWorld are presented in Figure 2.20 and 2.19.

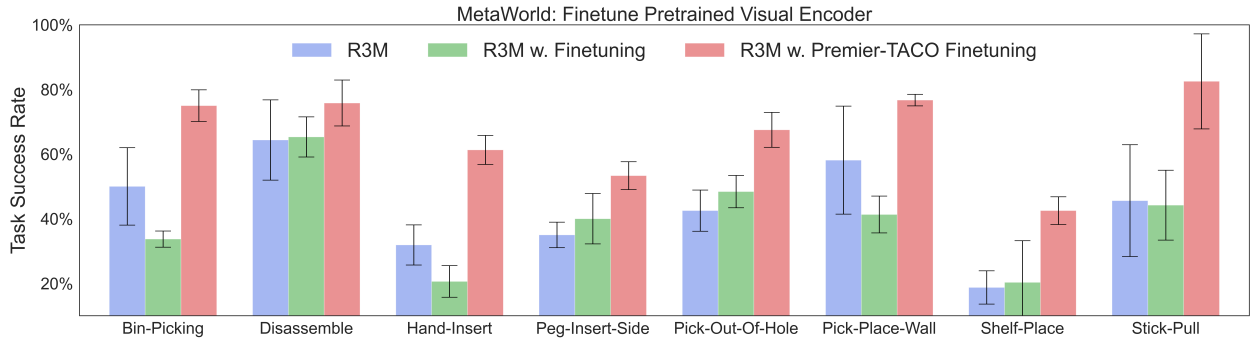


Figure 2.19: [(W4) Compatibility] Finetune R3M [96], a generalized Pretrained Visual Encoder with **Premier-TACO** learning objective vs. R3M with in-domain finetuning in Deepmind Control Suite and MetaWorld.

2.8.1 Implementation Details of Premier-TACO

Dataset For six pretraining tasks of Deepmind Control Suite, I train visual RL agents for individual tasks with DrQ-v2 [157] until convergence, and I store all the historical interaction steps in a separate buffer. Then, I sample 200 trajectories from the buffer for all tasks except for Humanoid Stand and Dog Walk. Since these two tasks are significantly harder, I use 1000 pretraining trajectories instead.

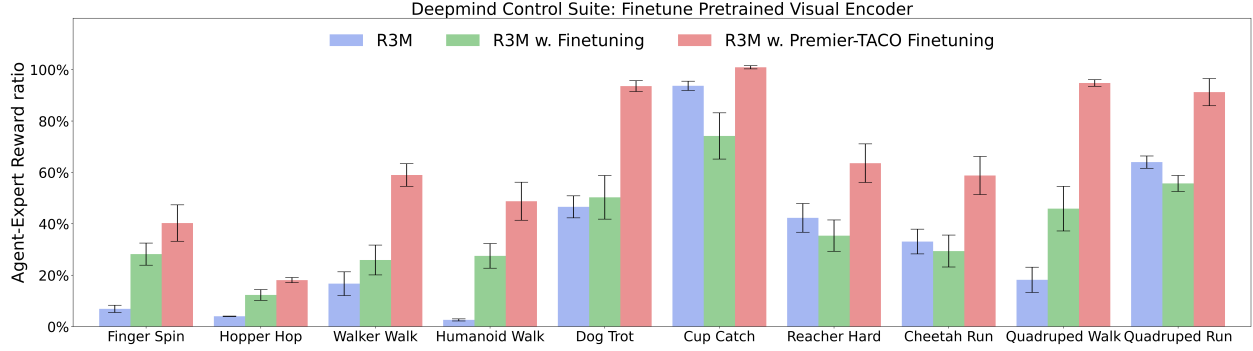


Figure 2.20: [(W4) Compatibility] Finetune R3M [96], a generalized Pretrained Visual Encoder with **Premier-TACO** learning objective vs. R3M with in-domain finetuning in Deepmind Control Suite and MetaWorld.

Each episode in Deepmind Control Suite consists of 500 time steps. In terms of the randomly collected dataset, I sample trajectories by taking actions with each dimension independently sampled from a uniform distribution $\mathcal{U}(-1, 1)$. For MetaWorld, I collect 1000 trajectories for each task, where each episode consists of 200 time steps. I add a Gaussian noise of standard deviation 0.3 to the provided scripted policy. For LIBERO, I take the human demonstration dataset from [85], which contains 50 demonstration trajectories per task.

Pretraining For the shallow convolutional network used in Deepmind Control Suite and MetaWorld, I follow the same architecture as in [157] and add a layer normalization on top of the output of the ConvNet encoder. I set the feature dimension of the ConNet encoder to be 100. In total, this encoder has around 3.95 million parameters.

```

1 class Encoder(nn.Module):
2     def __init__(self):
3         super().__init__()
4         self.repr_dim = 32 * 35 * 35
5
6         self.convnet = nn.Sequential(nn.Conv2d(84, 32, 3, stride=2),
7                                     nn.ReLU(), nn.Conv2d(32, 32, 3, stride=1),
8                                     nn.ReLU(), nn.Conv2d(32, 32, 3, stride=1),
9                                     nn.ReLU(), nn.Conv2d(32, 32, 3, stride=1),
10                                    nn.ReLU())
11        self.trunk = nn.Sequential(nn.Linear(self.repr_dim, feature_dim),
12                                  nn.LayerNorm(feature_dim), nn.Tanh())
13
14    def forward(self, obs):
15        obs = obs / 255.0 - 0.5
16        h = self.convnet(obs).view(h.shape[0], -1)
17        return self.trunk(h)

```

Listing 2.3: Shallow Convolutional Network Architecture Used in **Premier-TACO**

For LIBERO, I use two randomly initialized (or pretrained) ResNet-18 encoders to encode the third-person view and first-person view images with FiLM [108] encoding method to incorporate the BERT embedding [32] of the task language instruction. During downstream behavior cloning, I apply a transformer decoder module with context length 10 on top of the ResNet encodings to extract the temporal information, and then attach a two-layer MLP with hidden size 1024 as the policy head. The architecture follows ResNet-T in [85].

For **Premier-TACO** loss, the number of timesteps K is set to be 3 throughout the experiments, and the window size W is fixed to be 5. Action Encoder is a two-layer MLP with input size being the action space dimensionality, hidden size being 64, and output size being the same as the dimensionality of action space. The projection layer G is a two-layer MLP with input size being feature dimension plus the number of timesteps times the dimensionality of the action space. Its hidden size is 1024. In terms of the projection layer H , it is also a two-layer MLP with input and output size both being the feature dimension and hidden size being 1024. Throughout the experiments, I set the batch size to be 4096 and the learning rate to be 1e-4. For the contrastive/self-supervised based baselines, CURL, ATC, and SPR, I use the same batch size of 4096 as **Premier-TACO**. For Multitask TD3+BC and Inverse dynamics modeling baselines, I use a batch size of 1024.

Imitation Learning A batch size of 128 and a learning rate of 1e-4 are used for Deepmind Control Suite and Metaworld, and a batch size of 64 is used for LIBERO. During behavior cloning, I finetune the Shallow ConvNet Encoder. However, when I applied **Premier-TACO** for the large pre-trained ResNet/ViT encoder models, I keep the model weights frozen.

In total, I take 100,000 gradient steps and conduct evaluations for every 1000 steps. For evaluations within the DeepMind Control Suite, I utilize the trained policy to execute 20 episodes, subsequently recording the mean episode reward. In the case of MetaWorld and LIBERO, I execute 40 episodes and document the success rate of the trained policy. I report the average of the highest three episode rewards/success rates from the 100 evaluated checkpoints.

Computational Resources For our experiments, I use 8 NVIDIA RTX A6000 with PyTorch Distributed DataParallel for pretraining visual representations, and I use NVIDIA RTX2080Ti for downstream imitation learning on Deepmind Control Suite and Metaworld, and RTX A5000 on LIBERO.

2.8.2 Task instructions of downstream LIBERO tasks

Here in table 2.8, I provide the language instruction for each of the LIBERO downstream task. I refer readers to [85] for more details of the tasks.

TASK ID	TASK SCENE	TASK INSTRUCTION
0	LIVING ROOM SCENE2	PUT BOTH THE ALPHABET SOUP AND THE TOMATO SAUCE IN THE BASKET
1	LIVING ROOM SCENE2	PUT BOTH THE CREAM CHEESE BOX AND THE BUTTER IN THE BASKET
2	KITCHEN SCENE3	TURN ON THE STOVE AND PUT THE MOKA POT ON IT
3	KITCHEN SCENE4	PUT THE BLACK BOWL IN THE BOTTOM DRAWER OF THE CABINET AND CLOSE IT
4	LIVING ROOM SCENE5	PUT THE WHITE MUG ON THE LEFT PLATE AND PUT THE YELLOW AND WHITE MUG ON THE RIGHT PLATE
5	STUDY SCENE1	PICK UP THE BOOK AND PLACE IT IN THE BACK COMPARTMENT OF THE CADDY
6	LIVING ROOM SCENE6	PUT THE WHITE MUG ON THE PLATE AND PUT THE CHOCOLATE PUDDING TO THE RIGHT OF THE PLATE
7	LIVING ROOM SCENE1	PUT BOTH THE ALPHABET SOUP AND THE CREAM CHEESE BOX IN THE BASKET

Table 2.8: Language instructions for 8 LIBERO downstream tasks.

Chapter 3: FLARE: Future Latent Representation Alignment

In Chapter 2, I demonstrated that temporal contrastive learning with TACO and Premier-TACO objectives enable significant sample-efficient policy learning by capturing control-relevant dynamics in a compact latent space. Building on this principle, the current chapter presents FLARE, which extends the same principle to large-scale vision-language-action (VLA) models. In particular, I show that the concept of future latent prediction, which proved effective for compact state representations, can also be applied to these large VLA models to improve policy learning. This approach maintains the efficiency benefits of latent prediction while leveraging the rich multimodal understanding of pretrained VLA architectures.

3.1 Introduction

Human cognitive processes involve sophisticated predictive capabilities that operate largely implicitly. Consider a common action such as reaching for a coffee mug on a cluttered desk: without thinking about it, human brains could predict how the hand will move, what obstacles it might encounter, and how the mug will feel when grasped. This capacity to construct internal representations of future states, a form of world modeling, is fundamental to efficient human motor control and decision-making.

Several recent works [150, 22, 81, 182, 170, 34] have explored jointly learning world models and policies by generating future visual frames in parallel with actions. While intuitive, this approach faces notable practical and conceptual challenges. High-fidelity visual prediction typically requires large-scale generative models, introducing significant computational overhead and latency. Moreover, optimizing simultaneously for pixel-level reconstruction and action prediction places competing demands on model capacity: visual generation emphasizes detailed spatial fidelity and texture synthesis, whereas action modeling benefits from compact, abstract, task-relevant representations, often leading to diluted learning efficiency. In this work, I show that a surprisingly simple and flexible recipe, fully compatible with existing VLA architectures, can surpass prior VLA policy learning methods by a substantial margin.

I introduce **Future Latent REpresentation Alignment (FLARE)**, a lightweight yet highly effective extension to diffusion or flow-matching policies that introduces latent-space world modeling via a future alignment objective, eliminating the need for full-frame reconstruction. At its core, FLARE predicts a compact representation of the robot’s future observation from the hidden states of the action denoising network. FLARE operates in two key stages. First, I pretrain a compact, action-aware observation embedding model. While general-purpose embedding models could be used for the target future embeddings, I find that an action-aware embedding explicitly optimized for downstream control tasks offers superior performance and efficiency due to its compactness and task alignment. Next, I co-train the diffusion transformer by introducing a minimal set of additional

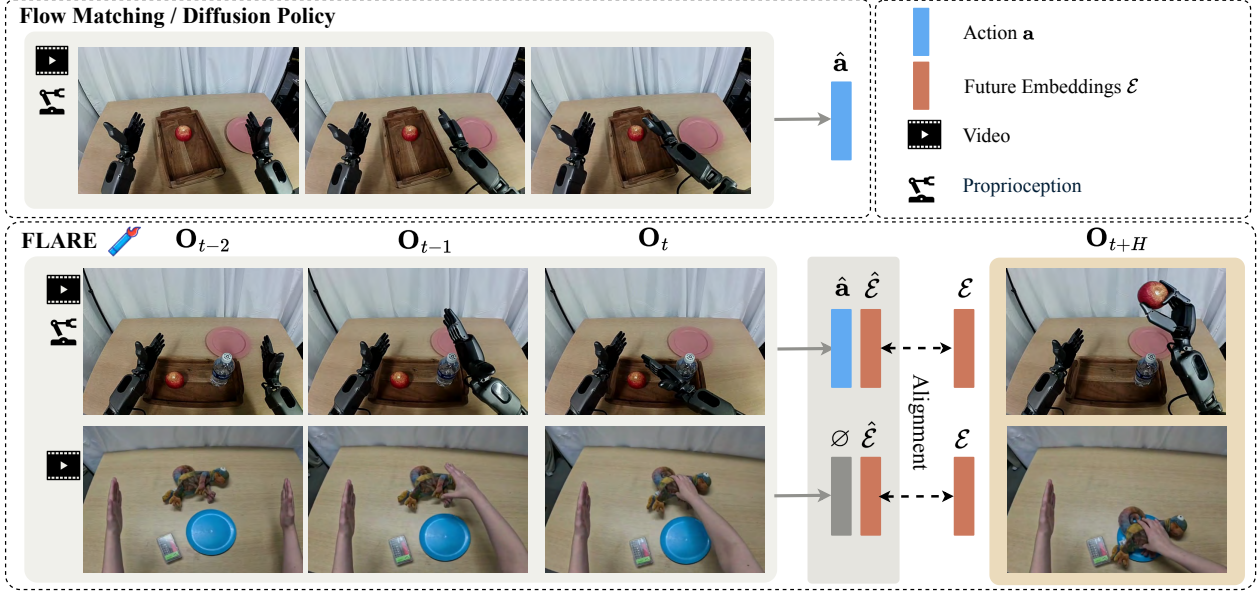


Figure 3.1: Comparison of FLARE to a conventional flow-matching (or diffusion) policy. FLARE can train using both action flow-matching and future latent alignment objectives, leading to improved performance as well as enabling learning from video-only data such as human ego-view demonstrations.

tokens, which are optimized to predict the future observation embeddings. This approach requires minimal modifications to existing VLA architectures [12, 98], making it broadly applicable and easy to deploy. Despite its simplicity, FLARE achieves state-of-the-art performance across two multitask imitation-learning benchmarks spanning single-arm and humanoid tabletop manipulation. Notably, when trained on diverse cross-embodiment robot data, our action-aware embedding model generalizes effectively to unseen embodiment and tasks. With just 100 trajectories per task collected on a real GR1 humanoid postrained from our pretrained action-aware observation embedding model, the FLARE policy achieves a 95% success rate in real-world evaluations. Finally, FLARE enables learning from action-free data sources, such as human videos. By leveraging GoPro-collected human egocentric video demonstrations and only a single real robot demonstration per object, FLARE successfully learns novel grasping strategies, highlighting its potential for scalable robot learning from less structured data sources.

3.2 Background

In this work, following π_0 and GR00T N1 [12, 98], I adopt **flow-matching** [84] as the learning objective for fitting actions from human demonstrations. Let o_t denote the robot’s observation, which includes image inputs (potentially from multiple views) and a language instruction; let q_t be the robot’s proprioceptive state; and let $A_t = (a_t, \dots, a_{t+H})$ be an action chunk drawn from expert demonstrations. I define $\phi_t = VL(o_t)$ as the vision-language embedding of the observation. Given the VL embedding ϕ_t , an action chunk A_t , a flow-matching timestep $\tau \in [0, 1]$, and sampled noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, I construct the noised action chunk as:

$$A_t^\tau = \tau A_t + (1 - \tau)\epsilon.$$

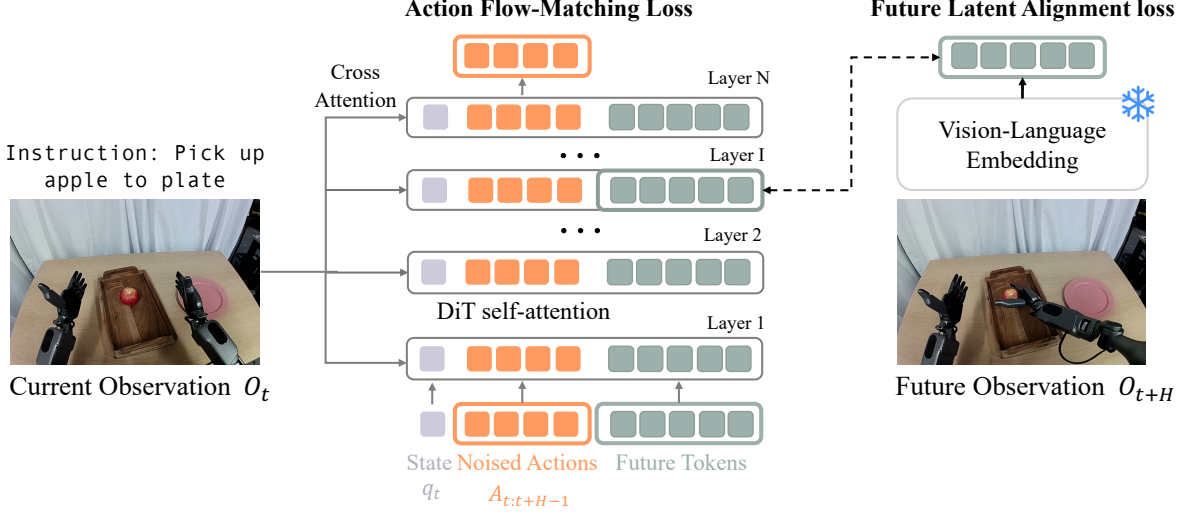


Figure 3.2: FLARE architecture. State and action token embeddings are concatenated into a sequence with learnable future token embeddings. The flow matching DiT blocks perform self-attention on this sequence, and cross-attention to the current vision and text observation embeddings. At a middle layer, the activations corresponding to the future token embeddings are used to compute a future latent alignment loss, which is the cosine similarity with vision-language embeddings from a future observation.

Then the model prediction $V_\theta(\phi_t, A_t^\tau, q_t)$ is trained to approximate the denoising direction $\varepsilon - A_t$, by minimizing the following flow-matching loss:

$$\mathcal{L}_{fm}(\theta) = \mathbb{E}_\tau [\|V_\theta(\phi_t, A_t^\tau, q_t) - (\varepsilon - A_t)\|^2]. \quad (3.1)$$

I sample the timestep τ from the distribution $p(\tau) = \text{Beta}(\frac{s-\tau}{s}; 1.5, 1)$ with $s = 0.999$ as in [12]. At inference time, I generate action chunks via K -step denoising. I first sample an initial chunk $A_t^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and then apply forward Euler integration to iteratively refine it:

$$A_t^{\tau+1/K} = A_t^\tau + \frac{1}{K} V_\theta(\phi_t, A_t^\tau, q_t).$$

Following GR00T N1 [98], I set $K = 4$ throughout all of our experiments, and I use the same Diffusion Transformer (DiT) architecture [107] for V_θ with alternating cross-attention and self-attention layers to condition on the robot’s vision language embedding ϕ_t .

3.3 Method

3.3.1 Latent World Modeling through Future Latent Representation Alignment

To enable the latent representation within the DiT blocks to predict future latent states, I add M learnable future token embeddings to the input sequence, such that the sequence contains three components: (1) the current proprioceptive state q_t encoded via a state encoder, (2) noised action chunk $A_t^\tau = \{\tau a_t + (1 - \tau)\varepsilon\}_t^{t+H}$ encoded by an action encoder, and (3) a set of M learnable future tokens. Next, I slice out the intermediate DiT representations corresponding to the M future tokens at an internal layer L , project those features using an MLP, and finally align these with the frozen vision-language embeddings of the future observation ϕ_{t+H} (see Figure 3.2).

Our approach is similar to how Representation Alignment (REPA) [162] is applied to improve text-to-image diffusion models, but with several important differences arising from the setting of latent world modeling. First, I align a DiT policy with *future* embeddings, rather than embeddings of the current observation. Second, our architecture adds learnable future tokens, so that the flow matching and alignment proceed along separate streams within the DiT, which interact via self-attention.

In this way, I encourage the DiT modules to internally reason about the future latent state while maintaining their action prediction capability through action flow-matching. Letting B indicate batch dimension and D indicate embedding dimension, I can write the latent alignment objective as

$$\mathcal{L}_{align}(\theta) = -\mathbb{E}_{\tau} [\cos(f_{\theta}(\phi_t, A_t^{\tau}, q_t), g(\phi_{t+H}))] \quad (3.2)$$

where $f_{\theta} \rightarrow \mathbb{R}^{B \times M \times D}$ outputs the DiT activations for the M future tokens at layer L , and $g \rightarrow \mathbb{R}^{B \times M \times D}$ is the encoder of the future observation ϕ_{t+H} . The overall loss function is

$$\mathcal{L} = \mathcal{L}_{fm} + \lambda \mathcal{L}_{align} \quad (3.3)$$

Empirically, I found $\lambda = 0.2$ worked the best in our experiments. I refer the readers to Section 3.4.4 for a detailed analysis of this choice.

3.3.2 Action-aware Future Embedding Model

While our future latent alignment framework is broadly compatible with various embedding models, I find that incorporating an *action-aware* future embedding yields further improvements in both performance and efficiency. To this end, I propose a compact vision-language embedding of the robot’s current observation, explicitly optimized for policy learning. The design objective is twofold: achieving **compactness** while ensuring **action-awareness**.

Specifically, I leverage both the vision and text encoders from SigLIP-2 [140] to encode the robot’s image observations and text instructions. The encoded tokens are then fused using four layers of self-attention transformer blocks to capture cross-modal dependencies. Subsequently, I apply a Q-former [80] module to compress the fused sequence into $M = 32$ learnable query tokens, producing a compact, fixed-size representation that naturally generalizes to multi-camera inputs. To ensure action-awareness, I train the vision language embedding end-to-end with the regular action flow-matching objective to predict the robot’s actions by attaching 8 DiT blocks. In this way, all task-relevant information is guaranteed to be captured within the latent token embeddings.

To pretrain the embedding model, I leverage a diverse mixture of cross-embodiment robot datasets, comprising both simulated and real-world humanoid tabletop manipulation data from GR00T N1 [98] and seven additional datasets from Open X-Embodiment [29], totaling approximately 2,000 hours of robotic data. Following pretraining, I posttrain the downstream policy jointly with the latent world model and the action prediction objective across downstream domains and tasks. Specifically, for posttraining, I initialize the downstream policy’s encoder with the pretrained embedding model, while also using the pretrained

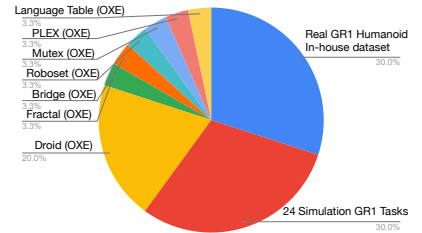


Figure 3.3: Data mixture of pre-trained action-aware vision language embedding model

embedding model to define the prediction targets for future latent representations. To mitigate distribution shifts between pretraining and downstream visual observations, rather than keeping the embedding model entirely frozen, I adopt an exponential moving average (EMA) update with respect to the policy’s encoder. This strategy allows the embedding model to gradually adapt in tandem with the evolving vision and language encoders during policy fine-tuning. Empirically, I find that an EMA update rate of 0.995 performs the best. I refer the readers to Section 3.4.4 for a detailed analysis of this choice.

3.4 Experiments

3.4.1 Multitask Benchmark Performance

In this section, I evaluate our latent world model on two multitask benchmarks that cover both single-arm manipulation and bimanual humanoid tabletop manipulation tasks. For the single-arm manipulation benchmark, I adopt RoboCasa [97], consisting of 24 atomic tasks in a simulated kitchen environment, including pick-and-place, door manipulation, faucet operation, and more. Robot’s observations include three RGB images captured from cameras mounted on the left, right, and wrist of the robot. Next, I incorporate 24 GR-1 tabletop simulation tasks from GR00T N1 [98],

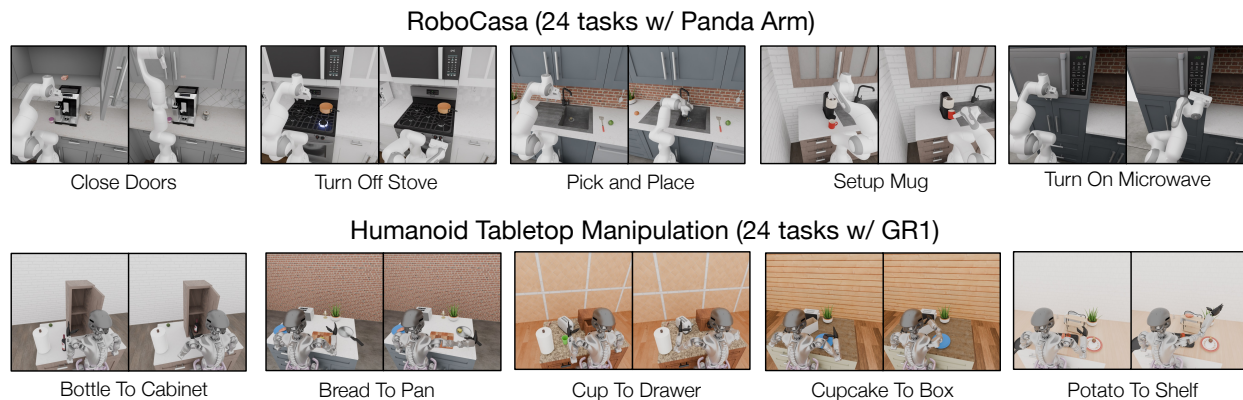


Figure 3.4: Multitask Simulation Benchmarks: I use 24 RoboCasa [97] and 24 GR-1 tabletop manipulation tasks as a multitask simulation benchmark suite in this chapter.

which emphasize dexterous hand control with the GR-1 humanoid robot. This suite includes 18 object rearrangement tasks—picking up and placing objects between source and target containers—and 6 tasks involving interaction with articulated objects such as cabinets, drawers, and microwaves. Observation consists of a single RGB image from an egocentric camera positioned on the robot’s head. To ensure a fair comparison between our method and the baseline, for experiments in this section, I do not use the pretrained embedding model mentioned in Section 3.3.2. Instead, I pretrain the embedding model exclusively on the same in-domain multitask dataset for 80,000 gradient steps, ensuring that any performance gains cannot be attributed to pretraining data with the embedding model. In particular, I include the following baselines for the experimental results:

1. **Diffusion Policy** [28]: Diffusion Policy models action distributions via a diffusion-based generative process, rather than using flow matching. It uses a U-Net architecture that progressively denoises random noise to generate the final action.

Methods	FLARE	Policy Only	UWM	GR00T N1 (Scratch)	Diffusion Policy
Pick and Place	53.2%	43.8%	35.6%	44.1%	29.2%
Open & Close Doors / Drawers	88.8%	78.7%	82.0%	80.0%	78.7%
Others	80.0%	75.2%	74.2%	69.6%	61.3%
24 RoboCasa Tasks Average	70.1%	61.9%	60.8%	60.6%	51.7%
Pick and Place Tasks	56.2%	42.9%	26.5%	45.9%	37.8%
Articulated Tasks	51.3%	47.4%	38.4%	42.8%	50.1%
24 GR1 Tasks Average	55.0%	44.0%	29.5%	45.1%	40.9%

Table 3.1: Task Success Rate Breakdown for Multitask Policy on RoboCasa and GR1 Tabletop Manipulation

2. **UWM** [182]: I select UWM as the main baseline for methods that jointly learn video and action prediction objectives. UWM predicts image VAE latents and actions jointly with a diffusion objective.
3. **GR00T N1 (Scratch)** [98]: Since GR00T N1 is pretrained on a much broader data mixture, I ensure a fair comparison by using the same architecture but initializing the DiT layers from scratch, while only loading the pretrained Eagle VLM [83] model weights.
4. **FLARE with Policy Only Objective**: I use the exact same model architecture as FLARE, as mentioned in Section 3.3.2, but train it solely with the policy learning objective.

All methods are trained for 80,000 gradient steps on the multitask robot dataset, except for UWM. I noticed that UWM performance is still improving at the end of 80k gradient steps, and thus I extend its training to 400k steps—five times the training budget allocated to the other methods. Following GR00T N1 [98], I evaluate each model checkpoint for 50 episodes per task every 1000 gradient steps, and report the maximum success rate over the final five checkpoints for each method.

As shown in Table 3.1, I draw two key observations. First, FLARE consistently outperforms all baseline methods, including both the policy-only baselines and UWM. This highlights the strength of our compact, action-aware latent world modeling objective in enabling more effective policy learning. Additionally, in our experiments, I also observe that FLARE with the policy-only objective, trained for 160k gradient steps, achieves only 44.1% success rate, resulting in no performance difference compared with 80k gradient steps. Thus, the improved results cannot simply be attributed to more training steps with FLARE. Second, even when trained with only the policy objective, FLARE still achieves performance on par with GR00T N1 initialized from scratch, despite GR00T N1 using a larger VLM backbone. This result underscores the quality of our Q-former-based vision-language embedding model in capturing action-relevant information.

3.4.2 Data-efficient Post-training with Cross-embodiment Pretrained Embedding Model

While the latent world model demonstrates substantial performance gains, as shown in the previous section, it requires training a separate embedding model for each domain. In this section, I

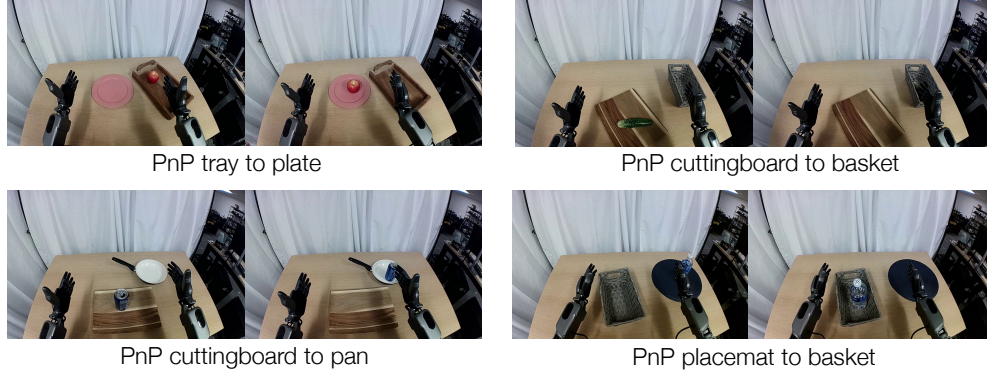


Figure 3.5: Real GR1 Tasks Setup: I evaluate four tabletop manipulation tasks on a real GR1 humanoid robot.

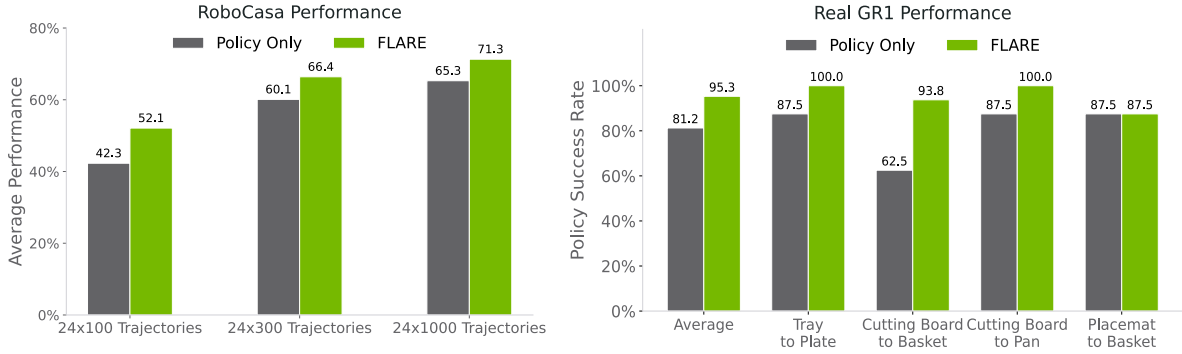


Figure 3.6: (Left): Post-training results on 24 RoboCasa tasks. **(Right):** Post-training results on 4 Real GR1 humanoid tasks.

evaluate FLARE with the pretrained embedding model mentioned in Section 3.3.2 as the future prediction target, focusing on unseen embodiments and tasks with data-limited posttraining settings. Specifically, I select 24 RoboCasa arm tasks and 4 real-world GR1 humanoid tabletop manipulation tasks as the evaluation benchmarks, and post-train the policy jointly with the latent world model and policy objectives, comparing it against a baseline that is post-trained using only the policy objective. In particular, for the policy-only baseline, I initialize both the Q-former-based vision language embedding and the policy’s DiT model weights from the cross-embodiment pretrained model. For FLARE, I only warm start the vision language embedding model.

For the evaluation protocol, I follow the same procedure described in Section 3.4.1 for the 24 RoboCasa tasks. For the 4 real-world GR-1 tasks shown in Figure 3.5, I define 8 reference initial frames per task, each involving 4 distinct objects (apple, can, bottled water, cucumber) to manipulate, and report the success rate of the final policy checkpoint for each method.

As shown in Figure 3.6, across both the 24 RoboCasa simulation tasks and the real-world GR-1 humanoid tasks, FLARE consistently outperforms the policy-only baseline. The improvement is especially pronounced under limited data conditions, achieving a 10% gain on RoboCasa with 100 trajectories per task for posttraining. Notably, although the pretrained embedding model has never seen RoboCasa tasks during pretraining, using it as the future embedding achieves comparable performance with 1000 trajectories to an embedding model trained exclusively on the 24 RoboCasa

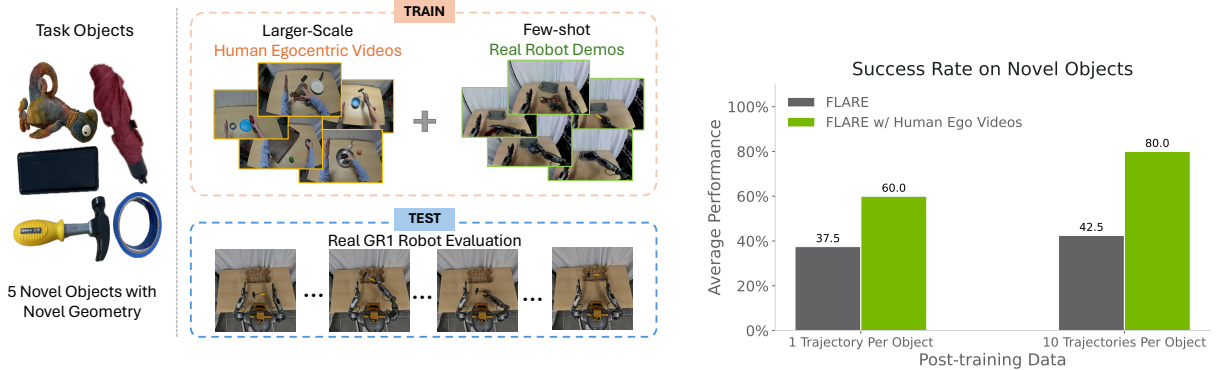


Figure 3.7: Generalizing to unseen objects with human egocentric videos and few-shot real robot demos

arm tasks (71.3% vs. 70.2% as reported in Section 3.4.1).

On the real GR-1 humanoid robot, I achieve a success rate of up to **95.1%**, averaging 14% higher than the baseline method. Qualitatively, I observe that in scenarios where a can or water bottle is placed close to the robot’s hand, the baseline method trained with only the policy objective often knocks over the object. In contrast, FLARE policy learns to maneuver around or over the object and successfully grasp, highlighting the benefits of future latent reasoning enabled by FLARE.

3.4.3 Leveraging Human Egocentric Trajectories without Action Labels

While our previous experiments demonstrate that the proposed future latent alignment objective significantly enhances policy performance when trained on action-labeled data, I further show that it can be naturally extended to trajectories without action annotations, such as human egocentric demonstrations. This setting is particularly attractive, as collecting human demonstrations is substantially more cost-effective and efficient than teleoperating a robot to execute the same tasks. To evaluate this, I select five novel objects with distinctive geometries that are absent from the training dataset, each requiring novel grasping strategies. For instance, the blue tape object is large and thus requires a top-down grasp by the robot hand. For each object, I collect 150 human egocentric demonstrations per object by mounting a GoPro on the demonstrator’s head while they perform similar tasks as the humanoid robot. On the robot side, I collect only 10 teleoperated demonstrations per object and train the policy using a mixture of these limited demonstrations, our GR-1 pretraining dataset, and the egocentric human videos.

For real-robot demonstrations with actions, I apply both the action flow-matching loss and the future alignment objective. In contrast, for the human egocentric videos without action labels, I rely solely on the future alignment loss to learn the latent dynamics. At evaluation time, I select five initial poses as reference images for each object and measure the robot’s success rate. Partial credit (0.5) is given when the robot successfully grasps the object but fails to place it into the basket.

As shown in Figure 3.7, with only 1 teleoperated trajectory per object, FLARE already achieves up to a 60% success rate on novel objects. When provided with 10 trajectories per object, and jointly trained with human videos, FLARE further improves to an 80% success rate—roughly doubling the performance of a baseline trained solely on action-labeled data. These results highlight that FLARE not only enhances learning from action-labeled demonstrations, but also effectively leverages unlabeled human demonstrations to improve generalization by capturing latent task

dynamics.

3.4.4 Ablation Study

Using the Pretrained Siglip2 as Future Embedding model:

While leveraging a policy-oriented future embedding model results in strong policy performance and enhanced training efficiency, I also explore an alternative setting that employs pretrained SigLIP2-Large vision tokens at timestep $t + 16$ as prediction targets. Specifically, I experiment using both raw SigLIP2 vision tokens (256 tokens per image) and 2×2 average-pooled tokens (64 tokens per image). As illustrated in Table 3.2, our FLARE framework maintains compatibility with diverse teacher encoder models beyond the policy-oriented embedding model. Although I get the optimal performance with the embedding model pretrained specifically on the target domain, using a more general-purpose vision encoder such as SigLIP2 still yields a significant 7% improvement over baseline methods.

Method	Success Rate (%)
No FLARE loss	43.9
SigLIP2	49.6
SigLIP2 (Average Pooled)	50.9
Action-aware Embedding	55.0

Table 3.2: Ablation of target embedding models.

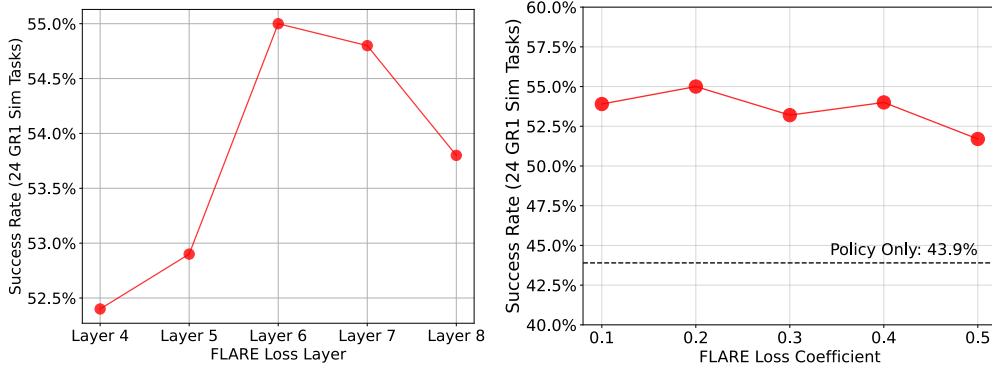


Figure 3.8: (Left): Ablation of the DiT Layer used in FLARE loss (Left): Ablation of FLARE loss coefficient.

Index of FLARE Loss Layer and Coefficient of FLARE Loss: A key design decision in FLARE is selecting the DiT layer at which to apply the future latent alignment loss, and the coefficient λ of FLARE loss. In our main experiments, I apply this objective at layer 6 out of 8 total layers in the DiT architecture. Applying it at deeper layers allows a larger portion of the model weights to benefit from the supervision of future latent prediction, but may also lead to conflicts between the action prediction and future alignment objectives. To evaluate the effect of these two hyperparameters, I evaluate FLARE on the GR1 simulation benchmark with different layer indexes and coefficients used for alignment. As shown in Figure 3.8, the model maintains strong performance across a range of hyperparameter setups. However, I do notice that applying the alignment objective too early—*e.g.*, at layer 4—leads to a notable drop in performance, highlighting the importance of aligning the future prediction objective with the action denoising process.

Exponential Moving Average (EMA) of Pretrained Action-aware Embedding Model: As discussed in Section 3.3.2, to address the distribution shift between pretraining and downstream tasks for our action-aware vision-language target embedding model, I incorporate an exponential moving average (EMA) update. Specifically, at each gradient step, the target embedding model

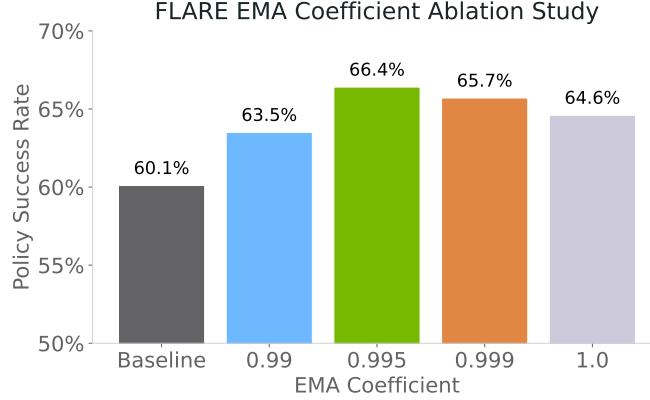


Figure 3.9: Effect of EMA Coefficient ρ : I report the policy success rate using 24×300 training trajectories across 24 RoboCasa tasks. Baseline is trained without FLARE future alignment loss, *i.e.*, a policy-only objective.

parameters are updated as follows:

$$\theta_{\text{target_vl_embedding}} \leftarrow \rho \theta_{\text{target_embedding}} + (1 - \rho) \theta_{\text{policy_vl_embedding}}$$

The EMA update enables the prediction target to adapt slowly in tandem with the evolving policy encoder, providing stability across training. Here, I evaluate several choices of the EMA coefficient $\rho \in \{0.99, 0.995, 0.999, 1.0\}$, each using 24×300 trajectories to train the FLARE policy. The final average success rates are reported in Figure 3.9. I find that while all EMA variants outperform the baseline method without FLARE future latent alignment objective, $\rho = 0.995$ yields the best performance and is used in all experiments. Notably, even with $\rho = 1.0$ (*i.e.*, no EMA), FLARE still surpasses the baseline, whereas $\rho = 0.99$ performs the worst, likely due to the instability caused by frequent target updates.

3.5 Related Work

Generative World Models for Robotics: There has been a rich body of research on world models for robotics, ranging from model-based control to model-based reinforcement learning [59, 44, 47, 27, 146, 180]. More recently, with advances in image and video generation, several works have explored the integration of generative modeling into policy learning [150, 22, 34, 182, 81, 35]. One line of work [34, 56] uses image diffusion models with inverse dynamics models to close the perception-to-action loop. The GR1 and GR2 families introduce end-to-end models that jointly predict discrete image tokens and actions using a unified next-token prediction objective. Other approaches [182, 81, 181, 125, 123, 179, 177] instead aim to jointly predict continuous image latents and actions. For instance, UWM [182] and UVA [81] jointly denoise VAE latents of future frames along with robot actions. DINO-WM [181] utilizes DINO features [181] to train a latent dynamics model for model-based planning.

Our work builds upon recent advances in representation learning, particularly Representation Alignment [162], which has shown remarkable success in accelerating the convergence of diffusion transformers for image generation and is key to state-of-the-art flow-matching models like Seedream-

3.0 [38]. However, our approach differs in two crucial ways: I train a flow-matching *policy* rather than an image model, and I align the DiT representation with features from *future* observations rather than current ones. In contrast to existing works, FLARE introduces an implicit latent world model objective that bypasses explicit reconstruction of future frames or latents. This simple design enables reasoning over a compact, action-aware latent space and avoids the computational burden of high-fidelity generation, while maintaining compatibility with standard VLA architectures, without requiring major architectural redesign. While DINO-WM focuses on zero-shot planning, FLARE is designed for policy and world model co-training, though planning could be a valuable future extension.

Vision Language Action Models. A growing body of recent work [16, 14, 12, 68, 175, 149, 23, 82, 173, 55, 160, 155] has focused on developing general-purpose foundational vision-language-action (VLA) models by fine-tuning vision-language models for downstream robotics tasks. Among these works, models such as [68, 155, 110, 66] autoregressively predict sequences of discrete action tokens using the next-token prediction objective. In contrast, methods like [99, 12, 98] leverage diffusion-based or flow-matching policy heads to bridge pretrained VLMs with continuous action generation. In this work, inspired by the architecture of GR00T-N1 [98], I adopt a flow-matching policy head built with diffusion transformer blocks, using interleaved self-attention and cross-attention layers to condition on the fused vision-language embeddings.

Learning from Egocentric Videos. Several approaches have sought to enhance robot learning by leveraging human egocentric videos. These efforts extract diverse forms of information, such as human-object interactions [166], object affordances [7, 61, 130, 129], and visual trace trajectories [148, 10]. Other lines of work aim to translate human motions into robotic behaviors using hand pose estimators [145, 183, 129, 9, 159, 112] or motion capture systems [156]. In this work, I show that future latent alignment provides a lightweight and effective alternative that does not require explicit pose estimators or point tracking tools, maximally reducing the engineering efforts. A complementary direction focuses on learning latent actions from visual deltas between current and future frames to guide downstream policy learning [160, 18, 26, 121, 118, 19]. Unlike latent actions as intermediate representations, whose correlation with ground-truth actions is unclear, our action-aware vision-language embedding directly aligns with future observations, resulting in a simple yet effective framework that naturally captures all the temporal dynamics information essential for effective policy learning.

3.6 Conclusion

I present Future Latent Representation Alignment (FLARE), a simple yet effective framework for jointly learning robot policy and latent world dynamics. By aligning the future representations of the robot’s observations with the hidden states of the action denoising network, FLARE enables the policy to implicitly reason about future states while predicting actions. This approach leads to state-of-the-art performance on challenging robotic manipulation benchmarks. Furthermore, FLARE unlocks co-training with human egocentric video demonstrations that lack action labels, significantly improving generalization to novel objects with minimal real-robot teleoperation data.

3.7 Supplementary Material: Additional Details and Experiments

3.7.1 Details of Q-former based Vision Language Embedding Module

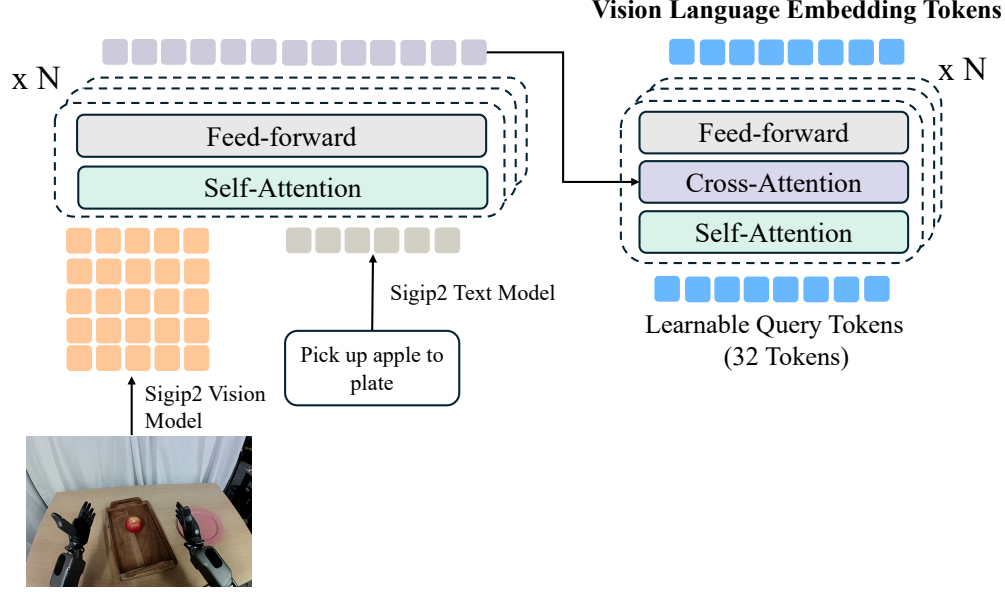


Figure 3.10: Q-former based Vision Language Embedding Module for FLARE

I present the architectural details of our compact Q-former-based vision-language embedding module. Specifically, I adopt siglip2-large-patch16-256 as the backbone for both vision and language encoders. The SigLIP2 vision encoder processes 256×256 resolution robot images into 256 patch tokens, while the language encoder encodes padded robot instructions into 32 language tokens. These 256 vision tokens and 32 language tokens are concatenated and passed through four layers of self-attention transformers to yield 288 fused vision-language tokens. To obtain a compact representation, I apply a Q-former architecture [80], where 32 learnable query tokens—randomly initialized—interact with the 288 fused tokens through interleaved self-attention and cross-attention layers, producing 32 compressed vision-language tokens.

3.7.2 Pretraining Data Mixture

Details of pretraining data mixture is presented in Table 3.3.

3.8 Training Details

For the pretraining of action-aware vision language embedding module, I use 256 NVIDIA H100 GPUs with batch size 8192 for 150,000 gradient steps. I use AdamW [87] optimizer with $\beta_1 = 0.95$, $\beta_2 = 0.999$, and $\epsilon = 1e-8$. A weight decay of $1e-5$ is applied, and the learning rate follows a cosine scheduling strategy with a warmup ratio of 0.05. Following [12, 98], I sample the flowmatching denoising timestep from $p(\tau) = \text{Beta}(\frac{s-\tau}{s}; 1.5, 1)$, $s = 0.999$.

Table 3.3: Action-Aware Vision Language Embedding Pre-training Dataset Statistics

Dataset	Length (Frames)	Duration (hr)	FPS	Camera View	Category
GR-1 In-house Dataset	6.4M	88.4	20	Egocentric	Real robot
DROID (OXE) [64]	23.1M	428.3	15	Left, Right, Wrist	Real robot
RT-1 (OXE) [16]	3.7M	338.4	3	Egocentric	Real robot
Language Table (OXE) [91]	7.0M	195.7	10	Front-facing	Real robot
Bridge-v2 (OXE) [?]	2.0M	111.1	5	Shoulder, left, right, wrist	Real robot
MUTEX (OXE) [128]	362K	5.0	20	Wrist	Real robot
Plex (OXE) [138]	77K	1.1	20	Wrist	Real robot
RoboSet (OXE) [11]	1.4M	78.9	5	Left, Right, Wrist	Real robot
GR-1 Simulation	125.5M	1,742.6	20	Egocentric	Simulation
Total	169.5M	2,989.5	–	–	–

For the multitask experiments of FLARE conducted in Section 3.4.1 and 3.4.2, I use 32 NVIDIA H100 GPUS with batch size 1024 for 80,000 gradient steps, while keeping the rest of the hyperparameter setups exactly the same.

3.8.1 Pseudocode of FLARE

Here I present a Python-style pseudocode of FLARE loss calculation as well as the entire training loop.

Algorithm 1 Python-style pseudocode for FLARE training

```
1 # target_vl_embedding: pretrained action-aware vision language embedding
2 # vl_embedding: vision language embedding of the current policy
3 # dit: diffusion transformer of the current policy
4 # action_embedding: 2-layer MLP to embed noisy actions
5 # state_embedding: 2-layer MLP to embed proprioceptive state
6 # action_decode: 2-layer MLP to decode robot's actions
7 # embedding_decode: 2-layer MLP to decode predicted embeddings
8 # N: Number of gradient steps
9 # M: Number of tokens in VL
10 # lambda: coefficient of FLARE loss (default is 0.2)
11
12 ### Initialization
13 future_tokens = nn.Embedding(M, hidden_dim)
14 vl_embedding.load_state_dict(vl_embedding.state_dict())
15 target_vl_embedding.requires_grad = False
16
17 for n in range(N):
18     obs, proprio, actions, future_obs = dataset.next()
19
20     ### Prepare noisy action inputs
21     noise = gaussian.sample()
22     timestep = beta.sample() # sample flowmatching timestep
23     noisy_action = timestep * actions + (1-timestep) * noise
24     velocity = actions - noise
25
26     ### Get state, action, and observation embedding tokens
27     action_tokens = action_embed(noisy_action, timestep)
28     state_token = state_embed(state)
29     vl_tokens = vl_embedding(obs)
30
31     ### Pass through DiT layers
32     sa_tokens = torch.concat([state_token, action_tokens, future_tokens], dim=1)
33     policy_outputs = dit(sa_tokens, vl_tokens)
34
35     ### Calculate action flowmatching loss
36     action_outputs = action_decoder(policy_outputs[:, 1:1 + action_tokens.shape[1]])
37     action_loss = MSE(action_outputs, velocity)
38
39     ### Calculate FLARE loss
40     with torch.no_grad():
41         embedding_to_align = target_vl_embedding(future_obs)
42         predict_embedding = decode_embedding(policy_outputs[:, -M:])
43         flare_loss = 1-COSINE_SIMILARITY(predict_embedding, embedding_to_align)
44
45     ### Optimize the combined loss
46     loss = action_loss + lambda * flare_loss
47     optimizer.zero_grad()
48     loss.backward()
49     optimizer.step()
```

Chapter 4: PRISE: From Latent State Representations to Temporal Action Abstractions

In Chapter 2 and Chapter 3, I demonstrate how the future latent prediction objective, at the core of world-model-based representation learning, can be used to learn compact and predictive state abstractions from high-dimensional observations. This objective encourages the model to capture the underlying dynamics of the environment in latent space, enabling it to reason about how current states evolve into future outcomes.

Interestingly, the same predictive principle that drives effective state representation learning can also guide the discovery of meaningful temporal action abstractions. If predicting future latent states allows an agent to understand how the world changes, then identifying consistent latent transitions over time provides a principled way to understand what actions cause those changes. In this sense, the future latent prediction objective offers not only a mechanism for learning world dynamics but also a foundation for uncovering hierarchical, temporally extended action structure. In this chapter, I introduce PRISE, a simple yet effective framework that leverages this world-model-based predictive principle to learn temporal action abstractions directly from offline robotic datasets, drawing inspiration from the compositional and hierarchical structures captured by large language models (LLMs).

4.1 Introduction

In **PRISE**, I posit that, similar to how LLMs use tokens to break down each sentence in natural language, adapting discrete coding and sequence compression techniques from LLMs could offer untapped potential for action representation learning for continuous control. Specifically, given a pretraining dataset of demonstrations from multiple decision tasks over a continuous action space and a high-dimensional pixel observation space, I consider the problem of learning temporally extended action primitives, i.e., *skills*, to improve the downstream learning efficiency of reward-free imitation learning algorithms, in particular, behavior cloning (BC). I show that embedding continuous actions into *discrete* codes and then applying a popular NLP sequence compression method called *Byte Pair Encoding (BPE)* [37] to the resulting discrete-code sequences identifies variable-timespan action primitives with the property I desire. Namely, in downstream tasks, policies learned by BC using these action primitives *consistently perform better*, often substantially, than policies learned by BC directly over the original action space.

PRISE's main contribution is *Primitive Sequence Encoding (PRISE)*, a novel method for learning multi-task temporal action abstractions that capitalizes on a novel connection to NLP methodology. **PRISE** quantizes the agent's original continuous action space into discrete codes, converts the pretraining trajectories into action code sequences, and uses BPE to induce variable-timestep skills. During BC for downstream tasks, learning policies over these skills and then decoding skills into

primitive action sequences gives **PRISE** a significant boost in learning efficiency over strong baselines such as ACT [168]. I conduct extensive ablation studies to show the effect of various parameters on **PRISE**’s performance, which demonstrates that BPE is critical to **PRISE**’s success.

4.2 Preliminaries

Problem Setting. I consider a set of tasks T , denoted as $\{(\mathcal{O}, \mathcal{A}, \mathcal{P}, g_{\mathcal{T}})\}_{\mathcal{T} \in \mathcal{T}}$ that share the action space $\mathcal{A}$, the transition dynamics $\mathcal{P}$, and the observation space $\mathcal{O}$. The tasks differ only in their goals $g_{\mathcal{T}}$. Since I focus on learning visuomotor continuous control policies, I assume $\mathcal{A}$ is a finite-dimensional vector space and $\mathcal{O}$ is the image space.

Let $\mathcal{D} = \bigcup_{\mathcal{T} \in \mathcal{T}} \mathcal{D}_{\mathcal{T}}$ represent a multitask dataset, with each $\mathcal{D}_{\mathcal{T}}$ containing expert trajectories specific to task $\mathcal{T}$. The focus of our work is to leverage the multi-task dataset $\mathcal{D}$ to learn a vocabulary $\mathcal{V}$ of skill *tokens* representing temporally extended low-level policies, which I call *skills*. Note that our concept of a skill is different from some others in prior work. Here skill policies can be thought of as control primitives for behaviors that are common across many tasks, e.g., lowering a robotic arm’s end-effector to the table-top level or flipping an object the end-effector is holding. Due to their universality, these primitives can enable efficient few-shot imitation learning (IL) for unseen tasks. Formally, I define the skill token vocabulary as $\mathcal{V} = \{\xi_1, \dots, \xi_K\}$, where each skill token ξ is a tuple $\langle f_{\xi}, L_{\xi} \rangle$ s.t. if the agent chooses token ξ at time step t , then for the next L_{ξ} steps it will take actions $a_{t+k} = f_{\xi}(o_{t+k}, k)$, $0 \leq k < L$ recommended by the token’s skill policy f_{ξ} .

Vector Quantization. In order to construct the temporal action abstraction skill tokens, our algorithm **PRISE** first converts continuous actions into discrete codes. To do so, it uses the vector quantization module proposed in [142]. This vector quantization module $\mathcal{F}$ maintains a collection $\mathcal{E} = \{e_1, \dots, e_C\}$ of C codes. For any query vector q (in our case, q is a concatenation of latent state and action, as described in Section 4.3.1), the module maps q to $\mathcal{F}(q) = e_k$, where $e_k = \arg \min_{e_i \in \mathcal{E}} \|e_i - q\|$. During training, $\mathcal{F}$ is optimized with respect to the loss $\mathcal{L}[\theta_{\mathcal{F}}] = \|\text{stopgrad}(q) - e_k\|_2 + \|q - \text{stopgrad}(e_k)\|_2$, where $\text{stopgrad}(\cdot)$ stands for stop-gradient. The first term here brings the code closer to the query vector q , and the second term prevents the query vector from switching between different codes.

Byte Pair Encoding (BPE). After quantizing the action space into codes, **PRISE** identifies common code (action) sequences via a technique from the realm of NLP, the method of Byte Pair Encoding (BPE) [37, 126]. In NLP, BPE has emerged as a pivotal technique for managing the vast and varied vocabulary encountered in textual data. Language models are expected to predict a text completion based on a text prefix. Doing so naively, by generating the text character-by-character is problematic, because the notion of a character differs significantly across languages. Instead, NLP models operate at the language-agnostic level of *bytes*. However, the high granularity of byte prediction has its own computational challenges. Therefore, language models first find high-frequency byte *sequences* in the training data, assign *tokens* to these sequences, learn to predict the resulting tokens, which then get decoded into text. BPE is the algorithm that handles the common byte sequence construction problem.

BPE operates by merging the most frequent pairs of bytes, assigning tokens to these pairs, and iteratively merging new pairs of tokens or bytes. Thereby, it builds up a vocabulary of tokens

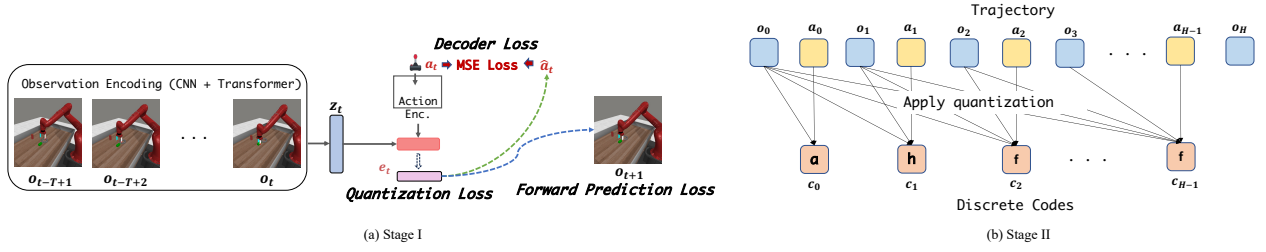


Figure 4.1: (a) Pretraining Stage I of **PRISE** : The goal is to learn a action quantization module such that conditioned on the current state and action (o_t, a_t) , it could assign a discrete action code. (b) Pretraining Stage II of **PRISE** : First it converts a trajectory of continuous state and actions into discrete codes. Then based on the corpus of quantized trajectories from the multitask offline dataset, **PRISE** applies BPE (illustrated in Figure 4.2) to learn vocabulary of skill tokens, where each token represents a sequence of discrete action code.

that plays a crucial role in modern large language models [101, 17, 115]. See Figure 4.2 for a demonstration. I apply BPE in the same way, but to action codes instead of bytes.

4.3 Algorithm

Figure 4.1 illustrates the key steps of **PRISE**. At a high level, **PRISE** first learns a state-dependent action quantization module (Pretraining I stage). This module processes the pretraining multitask dataset $\mathcal{D}$ by transforming each of its trajectories – a sequence of $\langle \text{observation}, \text{continuous action} \rangle$ pairs – into a sequence of discrete codes, one code per time step, as shown in Figure 4.1(a). Then, in Pretraining II stage, **PRISE** employs the BPE tokenization algorithm to identify commonly occurring code sequences in the processed dataset $\mathcal{D}$, assigns a token to each of them, and thereby constructs primitive skills, shown in Figure 4.1(b). Once these primitive skill tokens are discovered, they can be used to relabel a demonstration dataset and learn via behavior cloning (BC) a policy that chooses these skill tokens instead of raw actions, as I later show in Section 4.3.2 and Section 4.3.3. In the rest of the section, I describe **PRISE** in more detail.

4.3.1 Pretraining

Pretraining I: action quantization. Let $\mathcal{G} : \mathcal{O}^T \rightarrow \mathcal{Z}$ be the observation embedding which embeds a sequence of pixel observations of length T into the latent embedding space $\mathcal{Z}$. In our implementation, I use the transformer architecture [144] and set $z_t = \mathcal{G}(o_t, o_{t-1}, \dots, o_{t-T+1}) = \text{Transformer_block}(\text{CNN}(o_t), \dots, \text{CNN}(o_{t-T+1}))$. Using the multitask dataset $\mathcal{D}$, I jointly pretrain the observation embedding $\mathcal{G}$ and a state-dependent action quantization module $\mathcal{F} : \mathcal{Z} \times \mathcal{A} \rightarrow \mathcal{E}$, where $\mathcal{E} = \{e_1, e_2, \dots, e_C\}$ denotes the set of codes. The set of codes $\mathcal{E}$ produced by learning $\mathcal{F}$ depends on the inductive bias of $\mathcal{F}$ ’s training procedure. The inductive bias I chose for **PRISE** follows the intuition that, given the current latent state, a desirable action code should be predictive of both the next latent state and of the raw action mapped by $\mathcal{F}$ into that action code. A similar intuition underlies action representation learning in, e.g., [21], although that work doesn’t attempt to simultaneously learn a latent state space.

First, to ensure the action code can predict future states, I train a latent forward transition model

$\mathcal{T} : \mathcal{Z} \times \mathcal{A} \rightarrow \mathcal{Z}$. To optimize the forward latent dynamics model while preventing state and action representation collapse, I adopt a BYOL-like objective inspired from [124], where I minimize

$$\mathcal{L}_{\text{dyn}}[\theta_{\mathcal{F}, \mathcal{G}, Q, P}](o_{t:T}, a_t) = -\cos(Q(P(\mathcal{T}(z_t, e_t))), \text{sg}(P(z_{t+1}))) \quad (4.1)$$

Here, $o_{t:T} = o_t, o_{t-1}, \dots, o_{t-T+1}$ is the observation history, $z_t = \mathcal{G}(o_t, o_{t-1}, \dots, o_{t-T+1})$ is the latent state, $e_t = \mathcal{F}(\text{stopgrad}(z_t), a_t)$ is the quantized action code, P is a projection MLP, and Q is a prediction MLP. In our implementation, P has 2 layers and Q has 1.

Next, to guarantee the raw action can be decoded from the action code and latent state effectively, I train a latent state-dependent decoder, ψ , aiming to reconstruct the raw action given latent state z_t and action code e_t . In general, I let ψ be a stochastic mapping $\psi : \mathcal{Z} \times \mathcal{E} \rightarrow \Delta(\mathcal{A})$, where $\Delta(\mathcal{A})$ stands for the space of probability distributions over $\mathcal{A}$. I use a stochastic Gaussian Mixture Model (GMM) to parameterize the decoded action distribution of ψ following [93]. This choice has been found effective in dealing with the inherent multimodality and noise in such human teleoperation demonstrations.

Then the learning objective becomes to minimize the negative log likelihood of the GMM distribution:

$$\mathcal{L}_{\text{act_decode}}[\theta_{\mathcal{F}, \mathcal{G}, \psi}](o_{t:T}, a_t) = \mathcal{L}_{\text{act}}(\psi(z_t, e_t), a_t) = -\log \psi(a_t | z_t, e_t). \quad (4.2)$$

When I know the the pretraining data is collected by a fixed deterministic policy, I let $\psi : \mathcal{Z} \times \mathcal{E} \rightarrow \mathcal{A}$ be a deterministic function that minimizes the $L1$ -loss:

$$\mathcal{L}_{\text{act_decode}}[\theta_{\mathcal{F}, \mathcal{G}, \psi}](o_{t:T}, a_t) = \mathcal{L}_{\text{act}}(\psi(z_t, e_t), a_t) = \|\psi(z_t, e_t) - a_t\|_1. \quad (4.3)$$

Combining the two objectives, I pretrain the state encoder and action encoder/quantization module by

$$\mathcal{L}[\theta_{\mathcal{F}, \mathcal{G}, Q, P, \psi}] = \mathcal{L}_{\text{dyn}}[\theta_{\mathcal{F}, \mathcal{G}, Q, P}] + \beta \mathcal{L}_{\text{act_decode}}[\theta_{\mathcal{F}, \mathcal{G}, \psi}] \quad (4.4)$$

Throughout the experiment, I set $\beta = 1$ when ψ is a deterministic decoder, and $\beta = 0.01$ when ψ is parametrized by GMM to deal with numerical scale of the likelihood loss.

Pretraining II: temporal action abstractions via BPE. Figure 4.1(b) illustrates the mechanics of this stage. After training the action quantizer, I first use the pretrained observation embedding $\mathcal{G}$, action quantizer $\mathcal{F}$ to convert the trajectories from pretraining dataset $\mathcal{D}$ into sequences of discrete quantized action codes, $\{(e_0^{(i)}, e_1^{(i)}, \dots, e_{H_i}^{(i)})\}_{i=1}^{|\mathcal{D}|}$, where $e_t^{(i)} \in \mathcal{E}$ for all t, i and H_i stands for the length of episode i . Then, drawing analogy to NLP, I view the sequence of action codes in each trajectory as a ‘‘sentence’’ and employ the BPE algorithm to create a vocabulary of skill tokens $\mathcal{V} = \{\xi_1, \dots, \xi_K\}$, where tokens represent ‘‘sentence’’/trajectory segments. To do so, I initialize BPE using codes from $\mathcal{E}$ and let BPE construct the vocabulary $\mathcal{V}$ by iteratively merging the most frequent pairs of action codes or action code sequences into a single token. Different resulting tokens $\xi \in \mathcal{V}$ may correspond to skills with different horizons L_ξ .

Summary. Overall, the pretraining of **PRISE** produces observation embedding $\mathcal{G}$, action quantizer $\mathcal{F}$, and a vocabulary $\mathcal{V}$ of skill tokens. In the following, I discuss how to use them to learn generalist multitask policies and achieve efficient downstream adaptation.

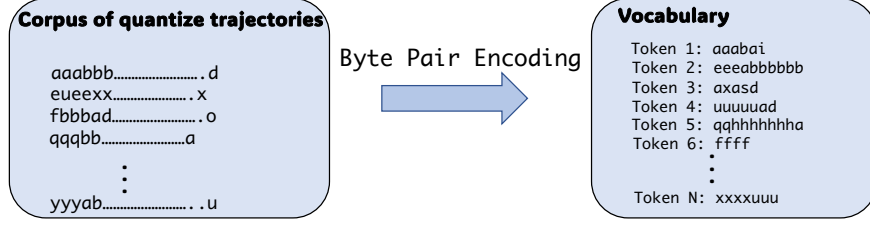


Figure 4.2: An illustration of how Byte Pair Encoding is done in **PRISE**.

4.3.2 Multitask generalist policy learning.

Once I have learned the skill tokens that capture the common motion patterns shared across various tasks, I can leverage these skills to learn a multitask generalist policy. I train a high-level skill-token policy $\pi : \mathcal{Z} \rightarrow \Delta(\mathcal{V})$ that predicts a skill token based on a latent state. At rollout time, π works as follows. At timestep t , suppose π predicts a token ξ_t after seeing observation $o_{t,T}$ and the token ξ_t corresponds to L_{ξ_t} action codes, without loss of generality, e_{t+k} , $0 \leq k < L_{\xi_t}$. Then for the subsequent L_{ξ_t} steps, the agent applies the codes e_{t+k} , $0 \leq k < L_{\xi_t}$ in their order, and uses the decoder $\psi : Z \times \mathcal{E}$ learned in Pretraining I stage to convert each e_{t+k} to an action a_{t+k} from the original action space $\mathcal{A}$. In other words, $a_{t+k} \sim \psi(z_{t+k}, e_{t+k})$ for each k s.t. $0 \leq k < L_{\xi_t}$. After L_{ξ_t} steps, the agent queries π again to choose the next sequence of actions. To learn this skill-token

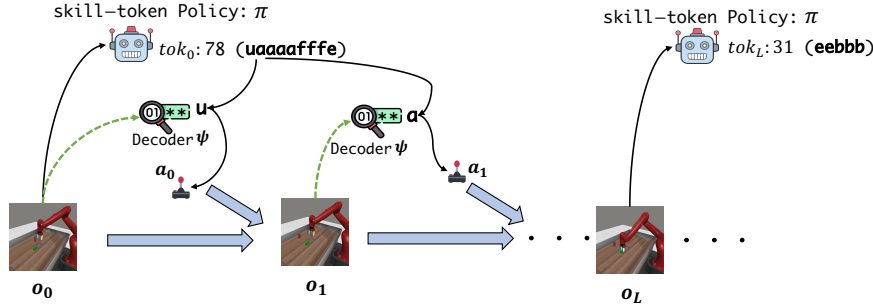


Figure 4.3: During evaluation time, **PRISE** rollout its policy by querying the skill-token policy π for the skill token and then using pretrained decoder ψ to decode actions.

policy π from a set of expert demonstration trajectories $\mathcal{D} = \{(o_0^{(i)}, a_0^{(i)}, \dots, o_{H_i}^{(i)})\}_{i=1}^{|\mathcal{D}|}$, I first run the action quantizer $\mathcal{F}$ and observation encoder $\mathcal{G}$ to get the quantized action codes of the trajectories $\{(c_0^{(i)}, c_1^{(i)}, \dots, c_{H_i}^{(i)})\}_i$, as well as latent observation embeddings $\{(z_0^{(i)}, z_1^{(i)}, \dots, z_{H_i}^{(i)})\}_i$. Then for each timestep t in episode i , I compute the target expert token $\xi_t^{(i)}$ by greedily searching for the longest token in the BPE tokenizer’s vocabulary that matches the sequence $(c_t^{(i)}, c_{t+1}^{(i)}, \dots, c_{H_i}^{(i)})$. I then train π by minimizing the cross-entropy loss:

$$\mathcal{L}_{\text{CE}}(\pi(\text{stopgrad}(z_t)), \xi_t)[\theta_\pi] \quad (4.5)$$

Note, that as stopgrad implies, I freeze the encoder $\mathcal{G}$.

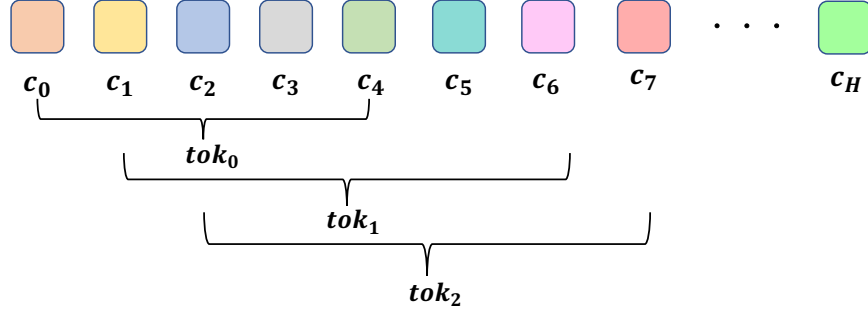


Figure 4.4: PRISE tokenizes downstream demonstration trajectories by greedily searching for the longest token for each time step.

4.3.3 Downstream few-shot adaptation to unseen tasks

In addition to learning a generalist multitask policy π , I can also use the pretrained skill tokens to adapt π to unseen downstream tasks with a few expert demonstrations. Unlike in the multitask pretraining setting, during finetuning I need to finetune the decoder ψ for downstream unseen tasks, in addition to adapting π itself.

I optimize ψ with the following objective, where the summation over training trajectories and time steps t in each trajectory has been omitted for clarity:

$$\mathcal{L}_{\text{FT_DECODER}}[\theta_{\pi, \psi}] = \mathbb{E}_{\xi_t \sim \pi(\text{stopgrad}(z_t))} [\mathcal{L}[\theta_{\psi}](\xi)] \quad (4.6)$$

where

$$\mathcal{L}[\theta_{\psi}](\xi_t) = \sum_{i=0}^{\hat{K}} \mathcal{L}_{\text{action}}(\psi(\text{stopgrad}(z_{t+i}), \xi_t), a_{t+i}) \quad (4.7)$$

In this equation, $\hat{K} = \min(K, L_{\xi})$, where K is a hyperparameter, the motivation behind which is explained at the end of Section 4.3.4.

I optimize skill-token policy π w.r.t. the cross-entropy loss $\mathcal{L}_{CE}$ in (4.5). Thus, the overall objective is

$$\mathcal{L}[\theta_{\pi, \psi}] = \mathcal{L}_{CE}(\pi(\text{sg}(z_t)), \xi_t) + \mathcal{L}_{\text{FT_DECODER}}[\theta_{\pi, \psi}] \quad (4.8)$$

Note that the gradients of the decoder finetuning loss $\mathcal{L}_{\text{FT_DECODER}}$ not only backpropagate into the decoder ψ but also into the skill-token policy π so that π should be aware of the decoder error. The underlying rationale for making the objective pay attention to matching both the skill tokens and the actions that these tokens generate is as follows. The number of skill tokens in the vocabulary is often large in order to encapsulate all motion patterns observed in the pretraining dataset, and simply minimizing the cross-entropy loss over the tokens with sparse data from few expert trajectories does not suffice to learn an accurate skill-token policy. Instead, the objective in (4.8) says that the skill-token policy π should not only match the target token but also predict the tokens that have small decoding errors.

4.3.4 Discussion

PRISE combines state-action abstraction (by optimizing the quantization loss in Equation (4.4)) and a temporal flavor of action abstraction (by applying BPE tokenization) to reduce the complexity of downstream IL problems. It is known that the complexity of IL depends on mainly two factors: the size of the state-action space and the problem horizon [116]. The former determines the minimum complexity of the learner’s policy. The latter determines how much the behavior cloning error will compound. **PRISE** addresses these two factors by its state-action abstraction and temporal abstraction, respectively.

The effectiveness of **PRISE**’s abstraction scheme is determined by several hyperparameters. First, **the codebook size**, which determines the granularity of quantization, trades off the approximation error of state-action abstraction and the complexity of the state-action representation that the learner’s policy sees. I want the codebook to be large enough to represent the demonstrator’s policy well, but not too large in order to avoid several codes collapsing to the same set of actions in the original space. Many duplicate codes would reduce the amount of temporal abstraction **PRISE** can achieve.

Second, **the size of the token vocabulary**, which is at least as large as the codebook size by the design of BPE, not only affects the complexity of the state-action representation, but also controls how generalizable the learned tokens are. When the token vocabulary is large, BPE picks up less frequent patterns, which typically are longer. Although these tokens may help compress the length of the training data, they might overfit to the training trajectories and end up never getting used in downstream tasks. Moreover, the overfit tokens might be too long and lead to approximation errors when decoded back to the original action space. **PRISE**’s hierarchical learning scheme implicitly assumes the expert policy applies the action codes in an *open-loop* manner up the horizon L_ξ of the skill token ξ that is decoded into these codes. This assumption stems from how the policy is used in roll out (see Fig. 4.3), and is related to also the Pretraining II stage, where BPE assembles skill tokens from action code sequences in an open-loop fashion ignoring latent states. In practice, the demonstrator’s policy may be closed-loop, in which case our approximation of it with open-loop skill policies introduces an error. This error grows with the length of a skill’s open-loop action sequence. Therefore, I introduce a hyperparameter K (see Section 4.3.3) that caps skills’ possible horizon in downstream adaptation.

On the contrary, when the token vocabulary size is set to be too small, the effects of temporal abstraction are minimal and would not reduce learning complexity. In the extreme, when the vocabulary size is the same as the codebook size, **PRISE** would only perform state-action abstraction by learning a discrete action space with state-action dependent encoder and decoder, without reducing the compounding error.

4.4 Experiments

In this section, I empirically evaluate the effectiveness of the skill tokens of **PRISE**. I run **PRISE** to pretrain these tokens using large-scale, multitask offline datasets. Then I evaluate them on two offline IL scenarios: learning a multitask generalist policy and few-shot adaptation to unseen tasks. I show that **PRISE** is able to improve the performance of the learner policy, both compared to an **PRISE** version approaches that doesn’t use the skill tokens and compared to a very strong existing approach, ACT [171].

4.4.1 Pretraining datasets and architecture

I evaluate **PRISE** on two multitask robotic manipulation benchmarks: Metaworld [163] and LIBERO [85]. Below I describe the setups of these datasets and the architectures used in the experiments.

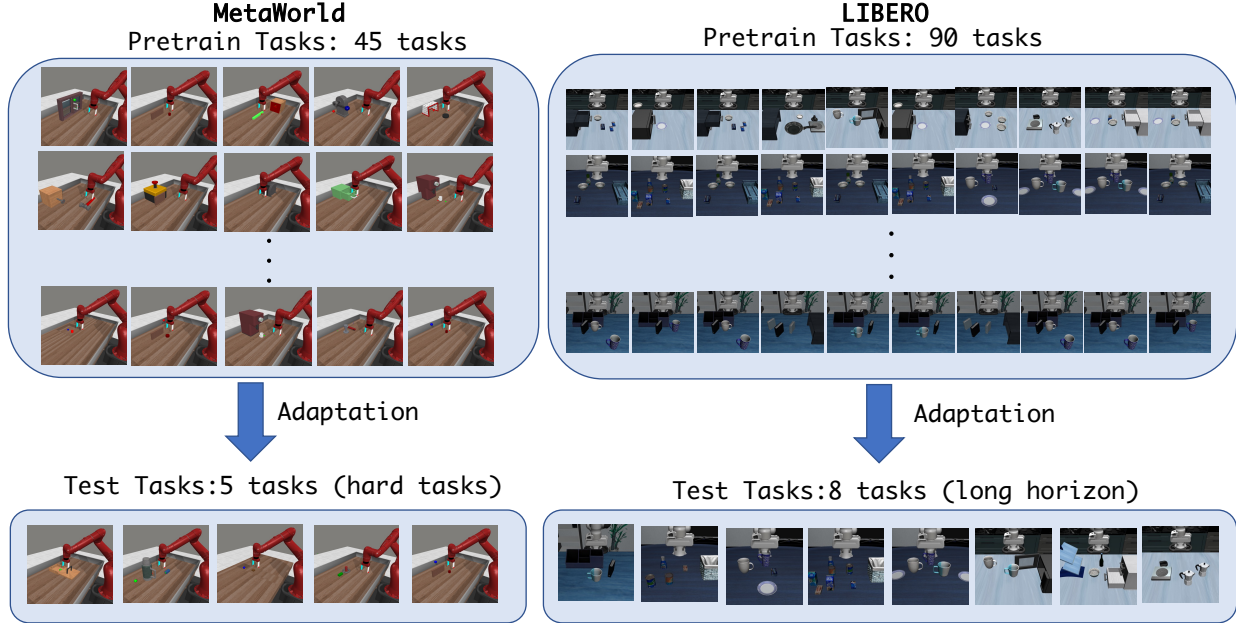


Figure 4.5: Few-shot IL on unseen tasks: pretrain and test tasks split for MetaWorld and LIBERO.

For LIBERO, I pretrain on the 90 short-horizon tasks (LIBERO-90) with offline dataset provided by the original paper. I test the learned skill tokens both in terms of the multitask learning performance on LIBERO-90 as well as 5-shot imitation learning (IL) performance on the first 8 unseen tasks from LIBERO-LONG. The pretraining dataset contains 50 demonstration trajectories for each task, collected by human teleoperation. I use the exact architecture of ResNet-T in [85] for the observation embedding.

For MetaWorld, I focus on 5-shot IL since multitask IL on pretraining tasks is straightforward, with baseline algorithms, including **PRISE**, all achieving an average success rate of around 80%. For the pretrain-test split, I hold out five hard tasks (hand-insert, box-close, disassemble, stick-pull, pick-place-wall) for few-shot evaluation and perform pretraining on the rest 45 tasks. I generate 100 expert trajectories for each pretraining task using the scripted policy provided in MetaWorld. Same as in as in [157, 154], I use a shallow CNN encoder to encode the observation, and a transformer decoder module as the temporal backbone to encode temporal information into the latent embedding z_t .

For BPE on both domains, I set the codebook size C to be 10 and vocabulary size to be 200, and I provide detailed ablation study to analyze their impacts later in this section.

4.4.2 Multitask generalist policy learning

First I evaluate whether the pretrained skill tokens of **PRISE** could enable efficient knowledge sharing across tasks and improve multitask IL performance. Here I focus on **LIBERO-90**, a chal-

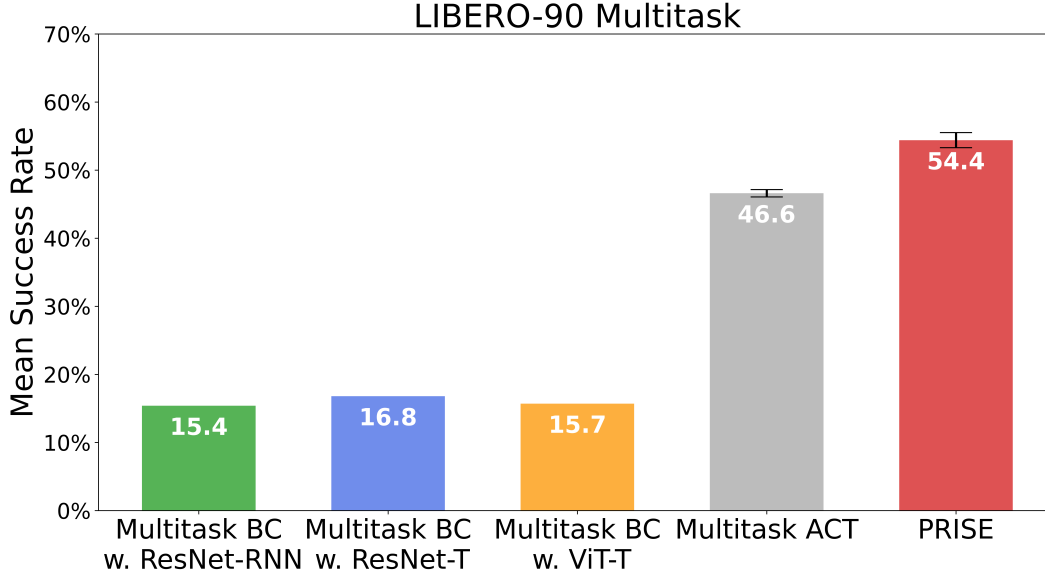


Figure 4.6: Multitask policy learning performance on **LIBERO-90**. Error bar represents the standard deviation across 4 random seeds. Results of the first three methods are taken directly from [85]

lenging multi-task benchmark where existing algorithms and architectures have not demonstrated satisfactory performance. For each algorithm in the following comparison, I first perform pretraining on the LIBERO-90, if applicable, and then train a multi-task generalist policy on the same dataset based on the pretrained outcomes (e.g., encoders, skill tokens).

Baselines. For multitask learning, our comparison includes **PRISE** and three network architectures for multitask behavior cloning (BC): ResNet-RNN, ResNet-T, and ViT-T, as implemented in [85]. These architectures utilize either ResNet-18 [52] or ViT [33] to encode the pixel observations and then apply either a transformer or a LSTM module as temporal backbone to process a sequence of visual tokens. Recall that the main architecture of **PRISE** is the same as ResNet-T, except I add the vector quantization as well as other modules unique to **PRISE**. Additionally, I compare with ACT [172], an IL algorithm that learns a generative model for sequences of actions, termed “action chunks,” rather than single-step actions. These action chunks can be seen as a form of temporal action abstraction. During policy rollout, ACT combines overlapping action chunks predicted by its policy using a weighted average. Thus, the critical hyperparameter of ACT is the length of the action chunk, which I set at 8 after evaluating the best option from five choices: {3, 5, 8, 10, 15}.

Experimental Results. Figure 4.6 presents the average success rate across 90 tasks in the LIBERO-90 dataset. It clearly demonstrates the significance of temporal action abstraction for knowledge sharing across diverse tasks, as Multitask ACT significantly outperforms the baseline BC approaches. More importantly, the use of pretrained skill tokens in **PRISE** further leads to a substantial performance improvement over all other existing algorithms, underscoring the efficacy of **PRISE**’s pretrained skill tokens.

4.4.3 Few-shot adaptation to unseen tasks

In addition to learning a generalist multitask policy, I show that the learned skill token of **PRISE** can also make learning a new task more efficient. Here, I evaluate the 5-shot IL performance for

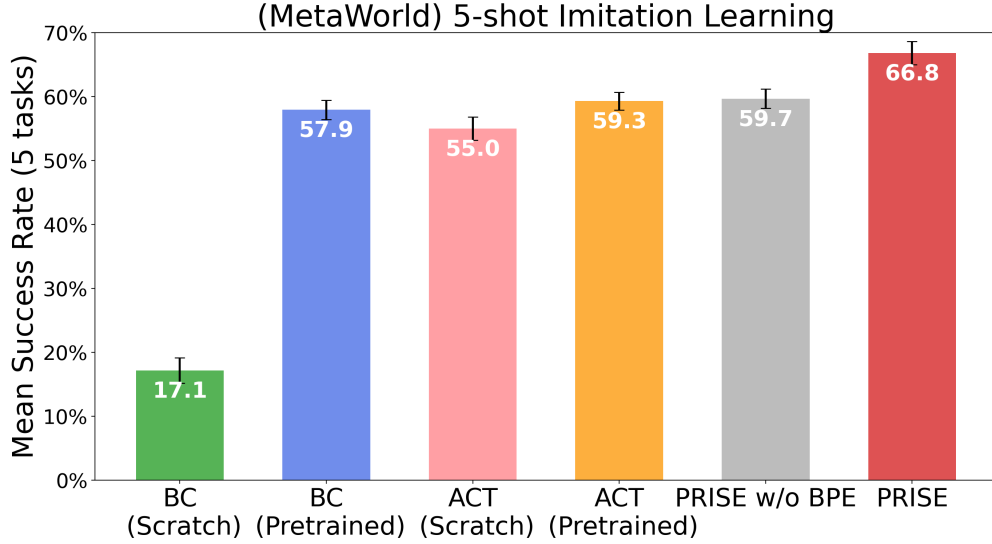


Figure 4.7: (Metaworld): 5-shot IL performance averaged across 5 unseen tasks. Error bar represents the standard deviation across 4 random seeds.

unseen tasks in both MetaWorld and LIBERO.

Baselines. I compare **PRISE** with the baselines below. (1) BC from scratch. (2) BC with pretrained CNN encoder from **PRISE**: This is a more computationally efficient version of the first. I find that whether freezing the CNN encoder or not does not have statistically significant effects on the downstream performance. (3) **PRISE** without BPE: This is the same as **PRISE** with a skill-token vocabulary $V = \{1, \dots, C\}$ of just the quantized action codes. That is, there is no temporal abstraction. This baseline can be viewed not only as an ablated version of **PRISE** but also as an analog to existing hierarchical skill learning approaches such as OPAL [4] and TACO-RL [119]. (4) ACT [172] from scratch. (5) ACT [172] with pretrained CNN encoder from **PRISE**. I freeze the CNN encoder and finetune other parts of the model.

In few-shot learning, I train each baseline algorithm for 30 epochs and evaluate each algorithm every 3 epochs. I then report the best success rate for the 10 evaluated checkpoints.

Experimental Results (MetaWorld). In Figure 4.7, I plot the averaged task success rate of 5-shot IL across 5 unseen tasks in MetaWorld. As shown in the figure, **PRISE** surpasses all other baselines (including **PRISE** w/o BPE) with a large margin, highlighting the effectiveness of the learned temporally extended skill tokens in adapting to unseen downstream tasks. Furthermore, I observe that with the pretrained visual encoder from **PRISE**, BC and ACT consistently yields improved results, indicating that the **PRISE** learning objective is also beneficial for pretraining visual representations from multitask offline data.

Experimental Results (LIBERO). I present the average 5-shot IL performance across 8 tasks of LIBERO in Figure 4.8. Different from MetaWorld, as shown in the figure, **PRISE** pretrained encoder does not improve the performance of the base IL algorithms for BC and ACT. This is consistent with what is reported in [85], where they observe that pretraining on LIBERO-90 with multitask BC even leads to negative transfer to downstream tasks. However, with pretrained skill tokens from **PRISE**, I could improve the average success rate by 9.2% compared with the best baseline algorithms. This further demonstrates the effectiveness of the proposed skill tokenization

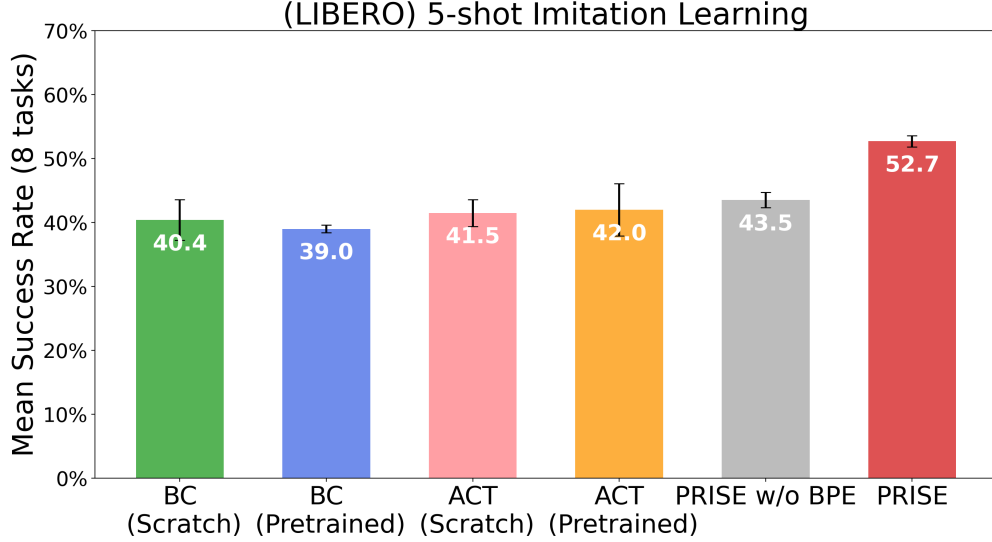


Figure 4.8: (LIBERO): 5-shot IL performance averaged across 8 unseen tasks. Error bar stands for the standard deviation across 4 random seeds.

mechanisms by **PRISE**.

4.4.4 Ablation Analyses

Vocabulary Size. The vocabulary size of the BPE tokenizer, $|\mathcal{V}|$, plays an important role in **PRISE**. A larger vocabulary size captures more motion patterns but increases the complexity of learning the skill-token policy due to more prediction classes. In addition, it learns less frequent tokens, which may not generalize or be useful to downstream tasks. On the other hand, a smaller vocabulary size limits the range of motion patterns that can be captured, but may reduce overfitting (see Section 4.3.4). In the previous experiments, I set $|\mathcal{V}|$ to be 200. Here, I evaluate five hyperparameter settings $\{100, 150, 200, 250, 300\}$ for $|\mathcal{V}|$ to assess their impact on downstream policy adaptation, in terms of the averaged 5-shot IL performance across 8 unseen LIBERO tasks. As Figure 4.9 (left) illustrates, **PRISE**’s performance remains relatively stable across different hyperparameter settings, except when $|\mathcal{V}|$ is extremely large or small. This demonstrates the robustness of **PRISE** to the choice of $|\mathcal{V}|$, indicating that as long as the vocabulary size is not too large or small, the performance should not vary significantly.

Number of Action Codes. Additionally, I verify the effects of varying the number of action codes, C . As C increases, action patterns become less distinct, making it challenging for the BPE tokenization algorithm to identify long skill tokens common across tasks. This diminishes the advantages of temporal action abstraction. In previous experiments, C was set to 10. I now present an analysis of the averaged 5-shot IL performance in LIBERO (consistent with the previous ablation) across five code size options 10, 20, 30, 40, 50, while maintaining a constant vocabulary size of $|\mathcal{V}| = 200$. As shown in Figure 4.9 (right), it is evident that with larger code sizes, such as $C = 40$ or $C = 50$, the performance discrepancy between **PRISE** and **PRISE** without BPE narrows. Furthermore, I also compare the histogram of skill token lengths for $C = 10$ versus $C = 40$ in Figure 4.10. The comparison reveals a clear reduction in token length with an increase in the number of codes.

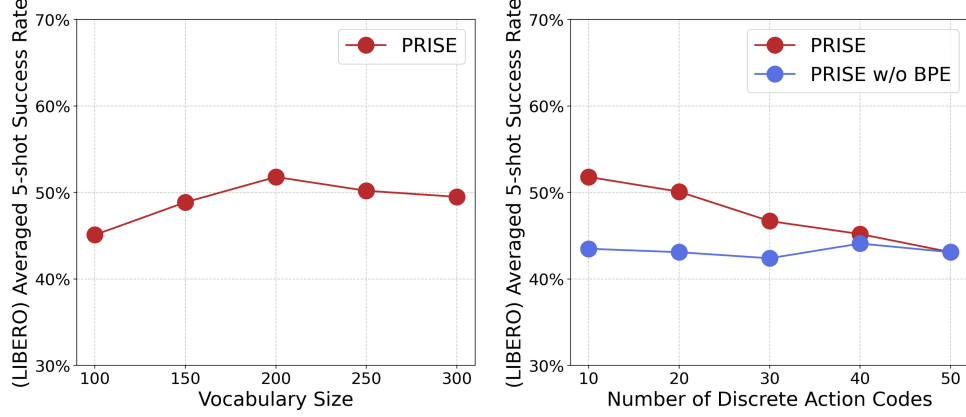


Figure 4.9: (Left) Mean success rate of **PRISE** across 8 unseen LIBERO tasks with different vocabulary size. (Right) The mean success rate of **PRISE** across 8 unseen LIBERO tasks with a different numbers of discrete action codes pre-trained.

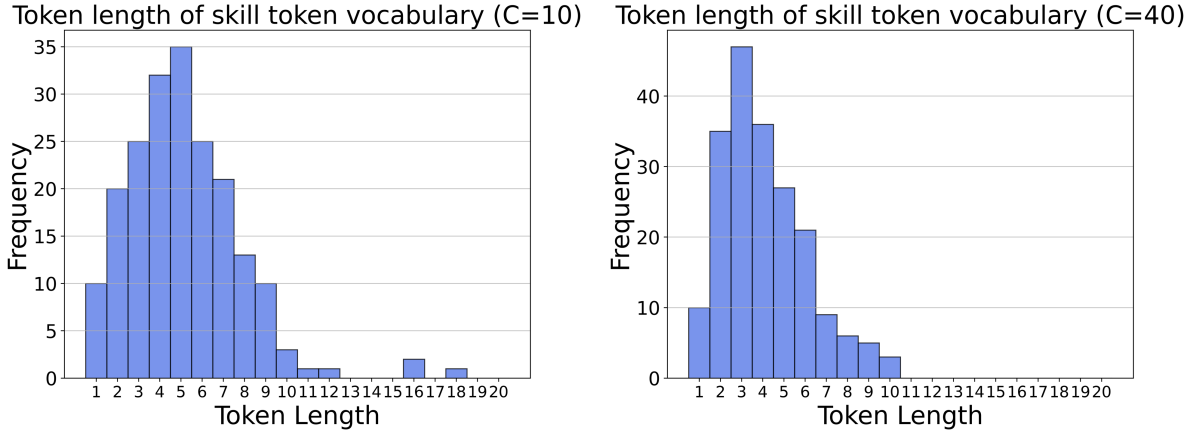


Figure 4.10: (Left) Histogram of token length when $C = 10$. (Right) Histogram of token length when $C = 40$.

Latent-forward-dynamics objective is crucial for learning good action codes. The forward-dynamics objective introduced in **PRISE** plays a crucial role in learning action code. Since I aim to learn a decoder $\psi(z_t, e_t)$ depending on the latent variable z_t , without the forward prediction term, the decoder $\psi(z_t, e_t)$ may learn to depend solely on z_t to reconstruct the action a_t , ignoring the action code e_t and leading to the collapse of action codes (i.e., different codes being decoded into identical actions). In Figure 4.11, I empirically compare the few-shot IL performance of **PRISE** in LIBERO with and without the forward dynamics objective. Indeed, I see a performance degradation in both domains when the forward dynamics objective is removed.

Additionally, I conducted an extra experiment on LIBERO to illustrate the action collapsing problem further. Here, I sample a batch of 5000 data points $\{(z_i, a_i)\}_{i=1}^{5000}$. For each state z_i , I calculate a matrix $\mathcal{M}^{(i)}$ of size $C \times C$ that quantifies the difference between the decoded action distributions of each pair of codes. Specifically, for entry (j, k) of $\mathcal{M}$, I compute the Monte Carlo estimate of the KL divergence $\mathcal{D}_{\text{KL}}(\psi(\cdot|z_i, e_j), \psi(\cdot|z_i, e_k))$ with 1000 samples. Then, I compute $\mathcal{M}$ as the average of $\mathcal{M}^{(i)}$. If the action distributions of different codes significantly differ, the non-diagonal entries of $\mathcal{M}$ should be large. Conversely, if some codes are decoded into similar action distributions, I expect

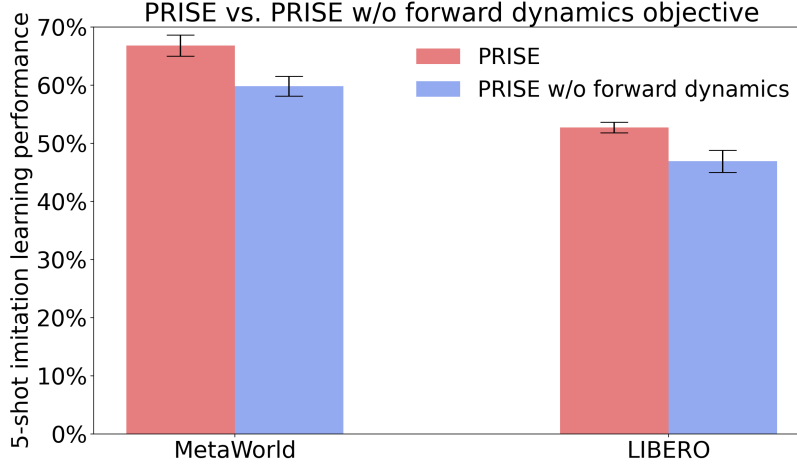


Figure 4.11: Performance of **PRISE** with and without the forward dynamics objective on Metaworld and LIBERO.

to see small non-diagonal entries in $\mathcal{M}$. I compare the value $\zeta = \frac{\text{sum}(\mathcal{M})}{\binom{C}{2}}$ for **PRISE** with and without the forward dynamics objective. As illustrated in the table, **PRISE** without the forward

	PRISE	PRISE W/O FORWARD DYNAMICS
ζ	72.42	14.53

dynamics objective exhibits a significantly smaller code-wise distance, demonstrating the necessity of the latent-forward-dynamics objective.

Effects of Dataset Size. To further study how much data is needed to learn a good temporal action abstraction in **PRISE**, I conduct a set of additional experiments on LIBERO. Instead of using the entire LIBERO-90 dataset, where each task includes 50 human demonstration trajectories, I now sample N trajectories per task, with $N \in \{10, 20, 30, 40, 50\}$. I then evaluate the downstream performance of the **PRISE** skill token. The result is shown in Figure 4.12.

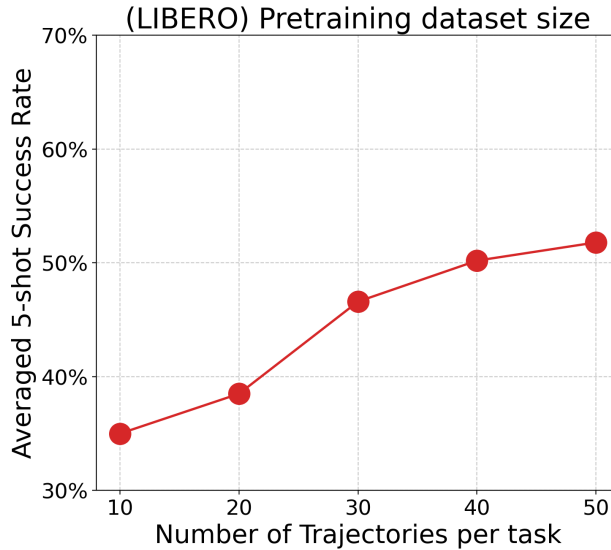


Figure 4.12: Downstream performance of **PRISE** under different pretraining dataset sizes

From the Figure, it is evident that the performance of **PRISE** monotonically gets better as the number of pretraining trajectories per task increases. Notably, the performance of **PRISE** is good with up to 40 trajectories per task but drops significantly with fewer than 30 trajectories per task. It would be interesting for future work to further scale up the existing LIBERO-90 dataset so that **PRISE** may further benefit from an even larger pretraining dataset. But this should be beyond the scope of the current work.

4.5 Related Work

Why learn temporal action abstractions? Learning temporally extended action primitives has a long history in sequential decision-making literature. In fully observable and partially observable MDPs [111, 133], the standard mathematical model for decision-making, every action takes one time step to execute. In most RL, IL, and planning algorithms, the decision space consists of individual actions, so standard methods produce policies that make a separate decision at every time step. It has long been informally observed and shown formally under various assumptions that the difficulty of policy learning scales with the problem horizon [120, 58, 116, 153]. This motivates hierarchical approaches [105, 8, 78, 95, 70, 74], which view the decision-making process as consisting of a high-level policy that identifies task substeps and invokes lower-level temporally extended routines sometimes called *options* [134] or *primitives* [4] to complete them. Conceptually, our **PRISE** method can be viewed as hierarchical imitation learning (HIL).

Temporal abstraction discovery for decision-making. Recent methods that learn temporally extended action primitives and subsequently use them for shortening the effective decision-making horizon during high-level policy induction include CompILE [70], RPL [41], OPAL [4], TAP [60], and ACT [172]. They operate in two stages, learning the primitives during the first and applying them to solve a downstream task during the second, possibly adapting the primitives in the process. It is during this latter stage that the learned primitives provide their temporal abstraction benefits. This is subtly but crucially different from methods like DDCO [72], which learn the primitives and a higher-level policy simultaneously for a given task and benefit from the primitives via non-temporal mechanisms.

PRISE’s use of BPE is dissimilar from any other method of inducing temporal action abstractions that I am aware of, and sidesteps some of the challenges of these other methods. E.g., CompILE, which, like **PRISE**, learns *variable*-duration primitives, does so by segmenting pretraining demonstrations into semantically meaningful subtasks, for which it needs to know the number of subtasks *in each pretraining trajectory* and is sensitive to errors in these values [70]. RPL and OPAL avoid this complication but learn primitives of a fixed duration, determined by a hyperparameter. This hyperparameter is nontrivial to tune, because short primitive durations yield little gains from temporal abstraction during learning, and long ones cause many primitives to idle after achieving a subtask. This is distinct from **PRISE**’s hyperparameter K , which merely puts an *upper bound* on skills’ duration.

I view ACT [171] as the most comparable method to **PRISE** and use it as a baseline in our experiments. Like **PRISE**, ACT handles high-dimensional pixel observations out of the box and, crucially, uses BC during downstream learning. In contrast CompILE, RPL, and OPAL assume access to ground-truth states and rely on online RL for finetuning. In many physical continuous control scenarios such as robotics, BC is arguably a more practical algorithm, and benefits that

temporal abstractions provide for BC are expected to be different from those in RL, where they enable more efficient exploration.

Policy quantization methods. **PRISE**'s state-conditioned policy quantization method is related to SAQ [88], but, crucially, also works for pixel observations characteristic of many continuous control scenarios in robotics. The policy encoder at the heart of **PRISE** and SAQ is VQ-VAE [143]. Conditional VAE [69] is another common choice for this role (see, e.g., [89] and [70]).

Temporal abstractions versus skills in robot learning. Many works on robot learning focus on the acquisition of *skills*. While seemingly similar to an option or a primitive, this term has a somewhat different meaning, and most of these works don't use skills for temporal abstraction, with TACO-RL [119] being a notable exception. Namely, approaches such as DMP [106], [50], Play-LMP [89], and MCIL [90] aim to learn a multi-task goal-conditioned policy, and understand a skill as a latent plan for achieving a goal or several goal variations starting from a variety of initial states. In this paradigm, learning a multitask policy constitutes embedding plans into a continuous latent space, which happens without using those plans to shorten a task's horizon. Indeed, the learned skills usually don't have a termination condition (although TACO-RL assumes having a mechanism for detecting subgoal attainment), and policies produced by the aforementioned methods resample a latent skill at every time step. This is in contrast to **PRISE** and, e.g., TAP [60] that learn temporally extended action primitives and apply them for several time steps during deployment.

Pretraining data assumptions. In terms of the assumptions **PRISE** imposes on the trajectories it uses for pretraining, it is more similar to Play-LMP [89], RPL [41], MCIL [90], and TACO-RL [119] rather than to IL or offline RL. Namely, **PRISE** needs this data to contain meaningful behavioral patterns, which it extracts using an imitation-like procedure (as does RPL). While **PRISE** could be applied to arbitrary-quality data commonly assumed by offline RL or some state representation pretraining algorithms such as [124, 147, 179, 176], many of the patterns there are unlikely to be useful, and may not easily aggregate into common temporally extended sequences that BPE aims to extract. On the other hand, the pretraining data doesn't need to be imitation-quality either.

Tokenization in language models Subword tokenization methods such as BPE [37], Unigram [73], and WordPiece [31] play an important role in training modern large language models. These algorithms learn tokens from a large corpus texts to mine reusable patterns. A trained tokenizer is used to compress training text data into tokens, and a large language model is trained to predict over this compressed tokens. Compared to directly predicting at the alphabet level, predicting tokens (with a proper vocabulary size) allows better generalization, because the effective sequence length that the model needs to predict is shorter.

Next token prediction in NLP is a special case of behavior cloning in the context of decision making, where the state is the history of tokens (context). Therefore, our algorithm can be viewed as extension of the next-token-prediction paradigm in NLP to the continuous control domain. However, our setup introduces additional complexities not faced in NLP. Our raw action space is continuous rather than discrete finite alphabets in NLP. In addition, our decision agents need to consider high-dimensional image observations as part of the state representation, whereas in NLP the history is simply past tokens. Namely, if I view alphabets in NLP as action vectors here, the BC problem in NLP has a stateless dynamical system, but in, e.g., robotics it has a non-trivial internal state. These complications motivate the need for observation and action encodes as well as an action quantizer.

4.6 Supplementary Material: Additional Details and Experiments

4.6.1 Implementation details of **PRISE**

```
1 # next_observation: A sequence of K=3 future pixel observations o_{t
    +1},..., o_{t+K}
2 # past_observation: A sequence of H historical pixel observations o_{t-H
    +1},..., o_{t}
3 # action:           A sequence of $K$ action a_t,..., a_{t+K-1}
4 # obs_encoder:      Observation Encoder (CNN+Transformer)
5 # quantizer:        Discrete action code quantizer
6 # dynamics:         Latent Dynamics Model
7 # project:          Projection layer
8 # predictor:         Latent state predictor
9 # action_decoder:    Quantized action code decoder
10 # beta:             Weight of decoder loss (1.0 in MetaWorld and 0.01 in
    LIBERO for          # numerical stability)
11
12 dynamic_losses, quantization_losses, decoder_losses = 0, 0, 0
13 z = state_encoder(past_observation)
14 z_hat = z
15 for k in range(K):
16     z = state_encoder(past_observation)
17     past_observation.push(next_observation[k])
18     u_quantized, quantization_loss = quantizer(action[k], z.detach())
19     quantization_losses += quantization_loss
20     decode_action = decoder(z, u_quantized)
21     decoder_losses += action_loss(decode_action, action[k])
22     z_hat = dynamics(z_hat, u_quantized)
23     z_next = state_encoder(observation[k+1])
24     y_hat = predictor(project(z_hat))
25     y_next = project(z_next).detach()
26     dynamic_losses += -cosine_similarity(y_hat, y_next)
27 (dynamic_losses + quantization_losses + beta*decoder_losses).backward()
```

Figure 4.13: Pseudocode for the pretraining stage I of **PRISE**

Metaworld: I generate 100 expert trajectories for each pretraining task using the scripted policy provided in MetaWorld. The observation for each step includes an 84×84 third-person view image as well as an 8-dimensional proprioceptive state vector of the robot’s end-effector. I use the same shallow CNN as in [157] to encode the observations into a 64-dimensional latent vector and apply a linear layer to embed the 8-dimensional state vector also into a 64-dimensional latent vector. The architecture of the shallow CNN is presented in Figure 4.14. Next, I apply a transformer decoder module with 4 layers and number of heads equal to 8 to extract the observation embedding. I set the context length to be 10. The action decoder ψ is a three-layer MLP with hidden size being 1024.

LIBERO: For LIBERO, I pretrain **PRISE** on the provided LIBERO-90 dataset. The pretraining dataset contains 50 demonstration trajectories for each task, collected by human teleoperation. For each timestep, the agent observes a third-person view image, a first-person view image from its wrist camera (both with resolution 128×128), a 9-dimensional vector of the robot’s proprioceptive


```

1 class Encoder(nn.Module):
2     def __init__(self):
3         super().__init__()
4         self.repr_dim = 32 * 35 * 35
5         self.convnet = nn.Sequential(nn.Conv2d(84, 32, 3, stride=2),
6                                     nn.ReLU(), nn.Conv2d(32, 32, 3, stride=1),
7                                     nn.ReLU(), nn.Conv2d(32, 32, 3, stride=1),
8                                     nn.ReLU(), nn.Conv2d(32, 32, 3, stride=1),
9                                     nn.ReLU(),
10                                    nn.Linear(self.repr_dim, feature_dim),
11                                    nn.LayerNorm(feature_dim), nn.Tanh())
12         self.trunk = nn.Sequential(nn.Linear(self.repr_dim, feature_dim),
13                                    nn.LayerNorm(feature_dim), nn.Tanh())
14
15     def forward(self, obs):
16         obs = obs / 255.0 - 0.5
17         h = self.convnet(obs).view(h.shape[0], -1)
18         return self.trunk(h)

```

Figure 4.14: Architecture of the Shallow CNN encoder used in MetaWorld.

state, and a task instruction in natural language. I use the exact same architecture as ResNet-T in [85]. In particular, I use two ResNet18 encoders [52] to process the image observations, utilizing FiLM [109] encoding to incorporate the 768-dimensional BERT embedding of the task language instruction. The action decoder ψ is also a three-layer MLP with hidden size being 1024.

Evaluation of Few-shot Imitation Learning A batch size of 128 and a learning rate of $1e-4$ are used for Metaworld, and a batch size of 64 is used for LIBERO. In total, I take 30,000 gradient steps and conduct evaluations for every 3000 steps. For both MetaWorld and LIBERO, I execute 40 episodes and calculate the success rate of the trained policy. I report the highest success rates across 10 evaluated checkpoints.

Computational Resources For our experiments, I use 8 NVIDIA RTX A6000 with PyTorch Distributed DataParallel for pretraining **PRISE**, and I use NVIDIA RTX2080Ti for downstream imitation learning on Metaworld, and RTX A5000 on LIBERO.

Chapter 5: Visual Trace Prompting: From Neural Latent Representations To Explicit Visual Cues

In the previous chapters, I discussed my research on achieving effective state and action abstraction through temporal contrastive representation learning and a tokenization-based approach to temporal abstraction. In this chapter, in the context of the recent success of large multimodal Vision-Language-Action (VLA) models, I discuss how such representations can also help these large multimodal foundation models, pretrained on large-scale internet data. Specifically, I introduce a surprisingly simple yet effective strategy: encoding the robot’s historical execution trajectory as a visual trace and using it to prompt the VLA model. This visual prompting significantly enhances the model’s ability to understand and reason about spatio-temporal dynamics.

5.1 Introduction

Robotic manipulation policies, typically trained on specific task demonstrations, often struggle to generalize beyond their training data, particularly when faced with novel objects, environments, instructions, and embodiments. In contrast, foundation models for vision and language—such as CLIP [113], LLaVA [86], Phi-3-Vision [2], and GPT-4V [3]—have demonstrated impressive generalization across diverse vision-language tasks. However, these models are not equipped to handle the challenges unique to robot manipulation, such as understanding kinematics, adapting to different embodiment configurations, and executing reliable physical actions. Vision-Language-Action models (VLAs) [15, 67] seek to address this gap by fine-tuning vision-language models to generate robot control actions using large-scale robotic datasets [e.g., 30], combining the generalization power of foundation models with task-specific robotic expertise. The approach has yielded promising results in developing generalist robot policies capable of adapting to a wide range of manipulation tasks. However, VLA-powered robots often struggle to maintain awareness of their past movements, leading to decisions that are more reactive to current inputs rather than informed by spatial history. I posit that this limitation arises because simply mapping image inputs as current states to control actions is insufficient. To address this, I propose explicitly computing multi-point temporal trajectories and overlaying them directly onto the image inputs for VLA models. I hypothesize that this will effectively provide spatial and temporal relations necessary for improving manipulation tasks [148, 165].

Our approach introduces an additional multi-point visual input during VLA training that tracks the robot’s past movement trajectory [62]. I refer to these multi-point trajectories as *visual traces*, and show that even with only 2D images as inputs (which allows for better scalability and integration with existing VLM models), VLA models enhanced with visual traces exhibit improved spatial-temporal awareness. This visual trace prompting technique enables better adaptation to variations in manipulation tasks and improves overall generalization.

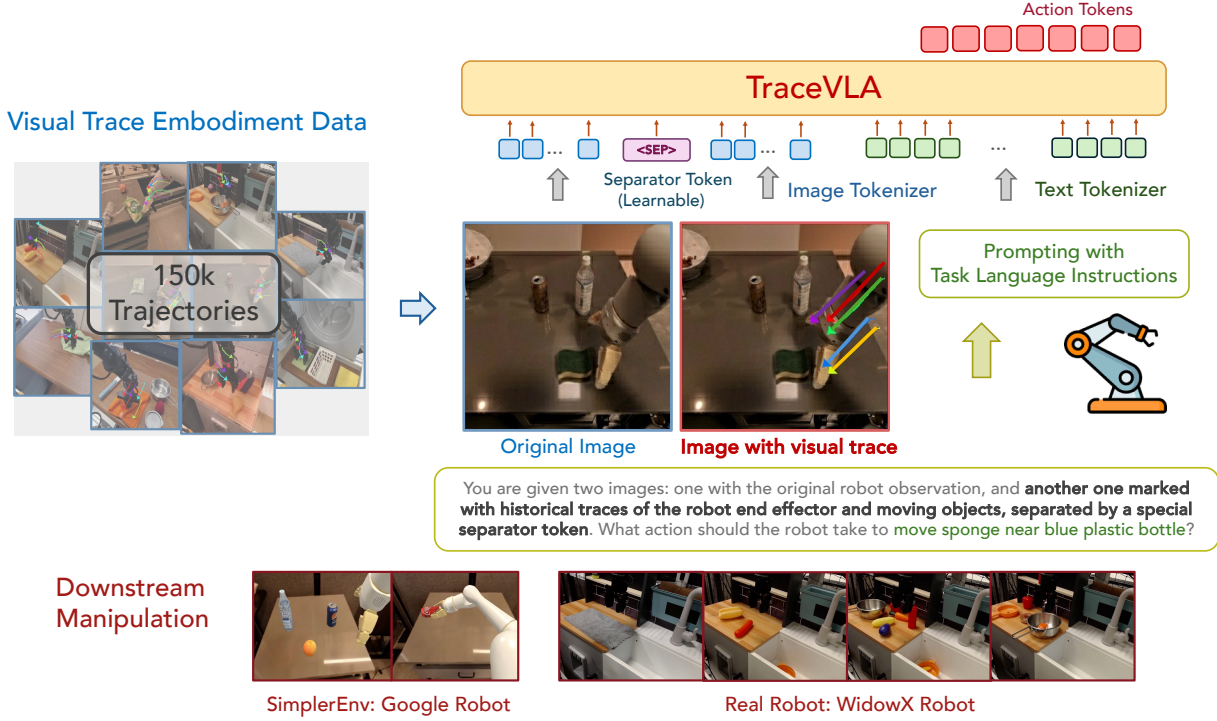


Figure 5.1: An illustration of our method. The first image shows the original robot’s observation, while the second contains the same image with overlaid visual traces. A separator token is then inserted between the visual tokens of these two images, then concatenating with text tokens and feeding into the underlying vision language model backbone to output action tokens.

I introduce **TraceVLA**, a 7B-parameter VLA model fine-tuned from OpenVLA using our novel visual trace prompting dataset, which includes 150K robot manipulation trajectories as shown in Figure 5.1. In addition, I finetuned a more compact VLA model, **TraceVLA-Phi3**, using the 4B-parameter Phi-3-Vision as a backbone on the Open X-Embodiments dataset, which comprises 970K trajectories across diverse robot embodiments, tasks, and scenes. **TraceVLA-Phi3** offers improved inference efficiency and reduced computational requirements while maintaining robust performance.

To assess the generalization capabilities of our models, I conducted evaluations across 131 diverse environment settings on Google Robots in the SimplerEnv simulator, which closely mimics real-robot scenarios. Additionally, I design diverse manipulation experiments on a physical WidowX robot to evaluate performance in real-world settings. Notably, our models consistently outperform existing VLA models across all embodiments and environments, demonstrating exceptional generalization under environmental variations.

Our key contributions are summarized as follows.

- **Method.** I introduce visual trace prompting, a novel technique that significantly enhances VLA models’ spatial-temporal reasoning in manipulation tasks.
- **Dataset & models.** I provide a visual trace prompting dataset and present state-of-the-art 7B and 4B VLA models fine-tuned using our proposed visual trace prompting, offering an efficient method to boost VLA model performance.

- **Validation.** Our approach is rigorously validated through extensive evaluations in both simulated and real-world robot tasks across diverse embodiments, demonstrating superior generalization capabilities by leveraging spatial-temporal information.

5.2 Preliminaries

Visual-Language-Action Models. Vision-language-action models (VLAs) extend vision-language models (VLMs) to predict discretized robot actions. Their architecture comprises: (1) a visual encoder [167, 102] that converts input images into patch embeddings, (2) a projector that maps these embeddings to the language model’s input space, and (3) a large language model backbone [139]. Action discretization, a crucial feature of VLAs, involves mapping continuous robot actions to discrete tokens. This process typically divides each action dimension into 256 bins based on data quantiles. These discrete actions are then incorporated into the language model’s vocabulary, often replacing the least frequently used tokens. VLA training builds upon the foundation established during VLM training. During VLM training, the model is trained end-to-end with a next text token prediction objective on paired or interleaved vision and language data curated from various Internet sources. VLA training then extends this approach, encompassing fine-tuning of the pretrained VLM. The cross-entropy loss focuses specifically on the predicted action tokens. This approach allows VLAs to leverage the capabilities of VLMs for complex robot control tasks while operating within the constraints of tokenized language models.

Generalist Control Policies. Robotic policy learning typically relies on task-specific demonstrations $\mathcal{D} = \{\tau_1, \tau_2, \dots, \tau_n\}$, where each $\tau_i = \{(o_t, s_t, a_t)\}_{t=1}^T$ represents an expert-level trajectory. The learning architecture comprises a visual encoder $\mathcal{F}_\phi$, mapping image observations o_i to features $z_i = \mathcal{F}_\phi(o_i)$, and a policy network π_θ outputting action distributions $\hat{a} \sim \pi_\theta(\cdot | z, s)$. Training minimizes the error between predicted $\hat{a}$ and optimal a actions. To overcome task-specificity limitations, generalist policies are being developed, aiming to handle diverse sensors, action spaces, and robotic platforms in various scenarios. Vision-Language-Action (VLA) models, leveraging VLM’s visual understanding and multimodal reasoning capabilities, show promise in creating more adaptable generalist policies. These models offer improved generalization across tasks, enhanced semantic understanding of environments, and the ability to follow natural language instructions, paving the way for more flexible and intuitive robotic control in broader applications.

5.3 TraceVLA

Our ultimate goal is to equip the Vision-Language-Action (VLA) model with the necessary context to better understand both temporal and spatial dynamics. In this section, I describe the details of our method, **TraceVLA**, to achieve this objective. First, I introduce visual trace prompting in Section 5.3.1. Next, I explain the model architecture of **TraceVLA** in Section 5.3.2. Finally, I provide the implementation details in Section 5.3.3.

5.3.1 Visual Trace Prompting

A straightforward way to have VLMs understand temporal history information is to concatenate historical frames and feed them into the VLA model. However, this approach can often distract the

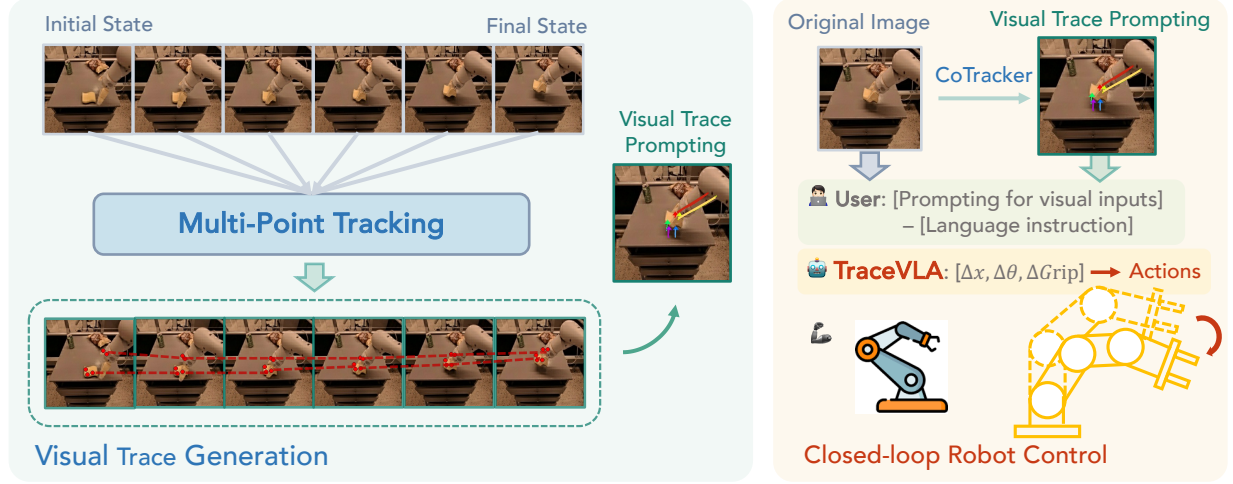


Figure 5.2: An illustration of visual trace generation. Given a sequence of historical image observations, I first use Co-tracker to extract dense point trajectories and keep active point trajectories with significant movement. I then overlay active point trajectories on the robot’s initial observation frame as visual trace prompting. I feed both the image overlaid with visual traces and the original image into VLA as model input.

model, as the frames are typically highly visually similar and redundant, making it difficult for the model to focus on the control-relevant information.

In this chapter, I introduce visual trace prompting to address this challenge. Instead of naively concatenating history frames, I employ an off-the-shelf point tracking algorithm to generate traces of key points. These visual traces are then visually overlaid on the robot’s original observations, serving as visual prompts that provide the model with a spatial memory of its historical actions.

At any given timestep t and a time window budget N , I first extract a set of dense point trajectories, $\mathcal{P}$, from a sequence of historical image observations, $\mathbf{h}_t = (o_{t-N}, \dots, o_t)$, using a dense point tracking model. Specifically, a point trajectory represents the trace of a moving point over time. Thus, a set of dense point trajectories captures the traces of multiple critical moving points. The specific dense point tracking model I employ is Co-Tracker [62], chosen for its efficiency and simplicity. Co-Tracker partitions the starting frame o_t into a $K \times K$ grid and tracks each grid cell across N frames to construct point trajectories. Consequently, I generate a total of $K \times K$ dense point trajectories, with each trajectory representing the location of a single point from timestep $t - N$ to timestep t .

While Co-Tracker provides a $K \times K$ grid of dense point trajectories, it does not inherently identify “active points.” To address this, I identify active point trajectories in $\mathcal{P}$ by analyzing changes in pixel locations, focusing on those with significant movement and distinguishing them from static background points. For each point trajectory $\mathbf{p} \in \mathcal{P}$, I first compute the absolute movement $\Delta \mathbf{p}_{t'}$ between two adjacent frames at timestep t' and $t' + 1$, as $\Delta \mathbf{p}_{t'} = |\mathbf{p}_{t'+1} - \mathbf{p}_{t'}|_1$. I then identify active point trajectories $\tilde{\mathcal{P}}$ by computing the total movement over N timesteps, keeping those whose movement exceeds a threshold κ . In other words, $\tilde{\mathcal{P}} = \{\mathbf{p} \in \mathcal{P} \mid \sum_{t'=t-N}^{t-1} \Delta \mathbf{p}_{t'} > \kappa\}$. From $\tilde{\mathcal{P}}$, I randomly sample M active point trajectories, denoted as $\hat{\mathcal{P}}$, for use in visual prompting.

Finally, I generate the visual trace by overlaying the sampled active point trajectories $\hat{\mathcal{P}}$ onto the robot’s original observation frame o_t , as shown in Figure 5.2. This overlaid frame serves as a visual prompt, providing the model with spatial information about its historical states and actions.

5.3.2 Model Architecture Design

In Figure 5.1, I outline the design of TraceVLA using visual trace prompting. While the overlaid visual trace provides valuable spatial-temporal information about the robot’s historical movements, it may obstruct the robot’s end-effector or key objects, potentially hindering the model’s ability to generate the correct action. To address this, I also include the original image observation in the model input, inserting a special separator token between the two images. As shown in Figure 5.1, I adjust the text prompt to inform the VLA model of this additional visual input before requesting the appropriate action output.

Additionally, since visual traces may not be available in all test-time scenarios—such as when the Co-Tracker model fails under pool lightning conditions—I implement a dropout mechanism during training. For each training example, with probability α , I replace the visual trace prompt image with the original image and remove the corresponding hint from the text prompt. This dropout strategy introduces variability into the inputs, encouraging the model to effectively utilize both the original image and the visual trace. As a result, at test time, even if the Co-Tracker model is unable to extract the visual trace, our model can still function correctly.

5.3.3 Implementation Details

In this section, I describe the implementation details of the dataset and models used in this chapter. For the visual trace generation pipeline in training, I use a grid size of $K=40$, sample $M=5$ active point trajectories, and employ a time window $N=6$. To reduce computational overhead, I run dense point tracking every N step for the future $2N$ frames, rather than at every timestep. I divide demo trajectories into overlapping $2N$ -sized segments (e.g., $[0, 2N)$, $[N, 3N)$, etc.) and run Co-Tracker once per segment. This approach ensures that for each timestep $t > N$, at least N steps of historical context are available, while significantly reducing computational costs.

To create our dataset with visual trace annotations, I applied our visual trace generation pipeline to the following training datasets: BridgeData-v2 [?], Google RT1 Robot datasets [15], as well as to 120 demonstrations collected from 4 manipulation tasks on our WidowX-250 Robot setup for downstream evaluation. In total, I gathered approximately **150,000** robot trajectories annotated with visual trace prompting, forming our fine-tuning dataset for TraceVLA.

To obtain real-time visual traces during inference, I reduce the computational overhead of densely querying the Co-tracker at every timestep by tracking M active points and sparsely querying Co-tracker. Specifically, at timestep $t=0$, I perform dense $K \times K$ point tracking to identify these active points (see Sec. 5.3.1 for details). For every timestep $t>0$, I query Co-tracker only for these active points to generate M corresponding traces, and update the tracked active points from the traces.

For the VLA models, I started with OpenVLA [67], a 7B VLA model based on the Prismatic vision-language model [63], trained on the Open X-Embodiment dataset [29]. Additionally, I pretrained a 4B VLA model with Phi3-Vision as its backbone VLM [1], on the Open X-Embodiment dataset using a batch size of 4096 for 30 epochs with 32 H100 GPUs, following the same recipe as OpenVLA. This lightweight 4B model allows us to test the flexibility of our visual trace prompting across different VLM model architectures. Additionally, the 4B Phi3V-based VLA model will also provide the community with a more compact VLA model for finetuning compared to the larger 7B Prismatic model, while the reduced memory cost allows fine-tuning to be performed on smaller GPUs such as RTX4090 or RTX A5000’s. For both **TraceVLA** and **TraceVLA-Phi3**, I finetune the

Models	Visual Matching			Variant Aggregation			Performance
	Move Near	Pick Coke Can	Open/Close Drawer	Move Near	Pick Coke Can	Open/Close Drawer	
OpenVLA-Phi3	53.3%	56.7%	38.4%	55.4%	57.7%	19.3%	46.8%
TraceVLA-Phi3	70.8% (↑17.5%)	69.7% (↑13.0%)	35.4% (↓3.0%)	75.4% (↑20.0%)	67.8% (↑10.1%)	37.5% (↑18.2%)	59.4% (↑12.6%)
OpenVLA	65.0%	23.7%	57.4%	55.8%	61.3%	24.9%	48.0%
TraceVLA	63.8% (↓1.2%)	45.0% (↑21.3%)	63.1% (↑5.7%)	60.6% (↑4.8%)	64.3% (↑3.0%)	61.6% (↑36.7%)	59.7% (↑11.7%)
Octo-Base	3.0%	1.3%	1.0%	4.2%	17.0%	22.0%	8.2%
RT1-X	55.0%	52.8%	22.5%	34.2%	54.0%	56.0%	45.8%

Table 5.1: Performance results on three SimplerEnv robot tasks under two evaluation metrics: visual matching and variant aggregation. Overall performance is the mean across all six task-metric pairs. For each TraceVLA method, we report both the raw success rate and its difference (↑ or ↓) versus the corresponding base model.

base VLA model for an additional five epochs.

5.4 Experiment

To comprehensively evaluate our model’s performance, I conducted experiments across a wide range of environmental setups, including 3 tasks with 137 different configurations in simulation and 4 tasks on real robots.

Baseline. I benchmark our approach against the following generalist policies, including state-of-the-art open-sourced models:

OpenVLA [67]: A 7B parameter VLA trained on the Open-X-Embodiment [29] Dataset, representing large-scale generalist policies.

OpenVLA-Phi3: A 4.5B parameter VLA pretrained model using Phi-3-Vision as backbone.

Octo-Base [137]: A 93M parameter transformer-based policy trained on 800k trajectories from the Open-X-Embodiment Dataset.

RT1-X [29]: A compact 35M parameter model trained on the same dataset as Octo-Base, exemplifying efficient architectures.

TraceVLA and **TraceVLA-Phi3:** Finetuned from OpenVLA and OpenVLA-Phi3 with visual trace prompting.

5.4.1 Simulation Evaluation

SimplerEnv. Our simulation evaluation utilizes SimplerEnv, which incorporates two distinct settings: **visual matching** and **variant aggregation**. The visual matching setting aims to minimize the visual appearance gap between real environments and raw simulation, significantly enhancing the correlation between policy performance in simulation and real-world scenarios. Complementing this, the variant aggregation setting covers a wide range of environmental variations as shown in Figure 5.4, including backgrounds from different rooms, lighter and darker lighting conditions, varying numbers of distractors, solid color and complex table textures, and different robot camera

poses. This comprehensive set of variations allows us to assess the robustness and adaptability of our approach in handling diverse manipulation scenarios, particularly evaluating the spatial and temporal awareness brought by visual trace prompting.

Overall Performance. As shown in Table 5.1, **TraceVLA** consistently outperforms OpenVLA across various tasks and evaluation metrics in the SimplerEnv Google robot tasks. The improvements are evident in both the full-scale 7B models (**TraceVLA** vs OpenVLA) and their 4B versions (**TraceVLA-Phi3** vs OpenVLA-Phi3). **TraceVLA** shows significant performance gains, with improvements ranging from 2.4% to 12.7% in all three tasks across different metrics. When compared to other baselines like Octo-Base and RT1-X, both **TraceVLA** and **TraceVLA-Phi3** generally perform better, with a few exceptions where RT1-X shows competitive performance in specific tasks. These results suggest that the visual trace prompting technique employed in **TraceVLA** enhances the model’s ability to generalize across different robotic manipulation tasks and environmental conditions, leading to improved performance in both visual matching and variant aggregation scenarios.

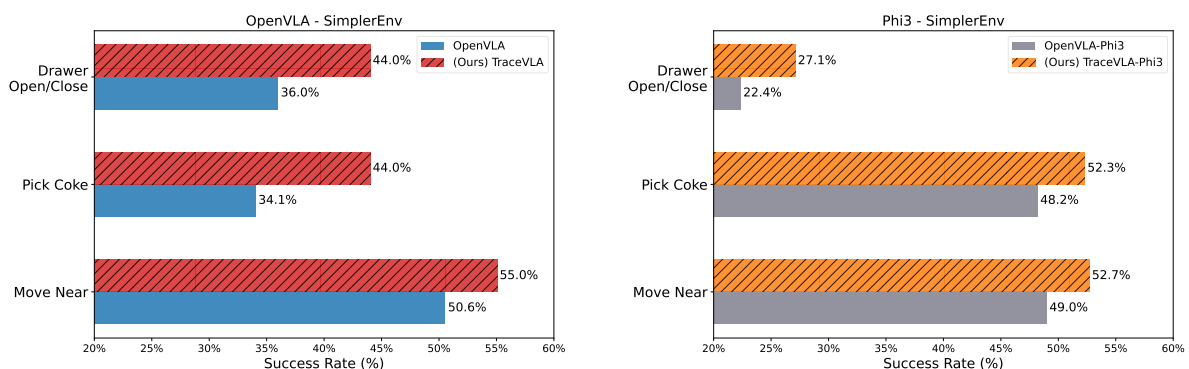


Figure 5.3: Performance comparison of **TraceVLA** against OpenVLA (**Left**): 7B **TraceVLA** vs. 7B OpenVLA. (**Right**): 4B **TraceVLA-Phi3** vs. 4B OpenVLA-Phi3. Numbers are averaged across the visual matching and variant aggregation metrics.

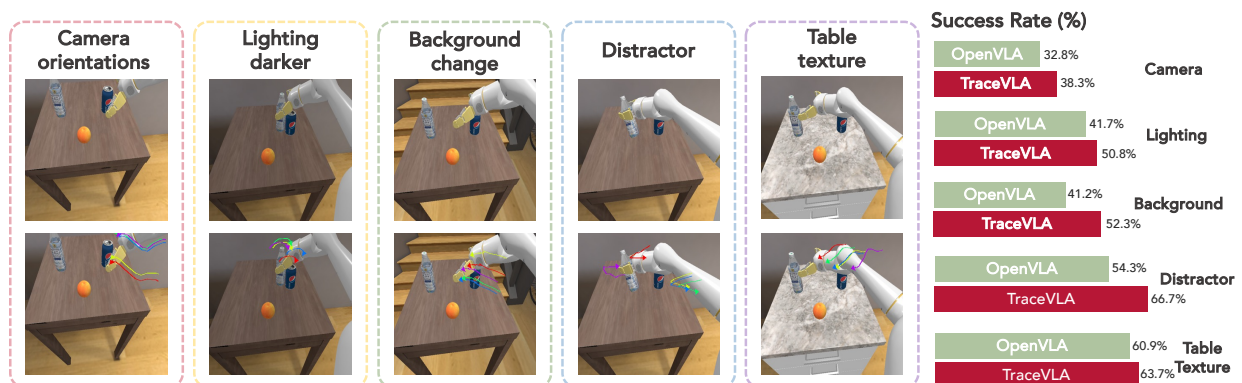


Figure 5.4: Comparison of OpenVLA and **TraceVLA** performance across various environmental variations: camera orientations, lighting, background, distractors, and table texture.

Environmental Variant Aggregation. Figure 5.4 demonstrates significant performance improvements of **TraceVLA** over OpenVLA across various environmental conditions. Notably, **TraceVLA**

shows substantial enhancements under camera orientation changes, distractor presence, and background alterations, with an average improvement exceeding 20% in these categories. The use of visual trace prompting proves particularly effective when camera angles shift, as the trace provides valuable spatial trajectory information. This helps the model maintain performance despite changes in perspective. Similarly, when faced with background changes, varying table textures, or lighting alterations, the visual trace allows the model to remain more stable and less influenced by environmental backgrounds. In scenarios with added distractors, the visual trace effectively reminds the policy of past trajectory interferences, evidenced by noticeable arm diversions. These varied environmental conditions collectively validate the spatial and temporal understanding provided by visual trace prompting.

5.4.2 Real Robot Experiments

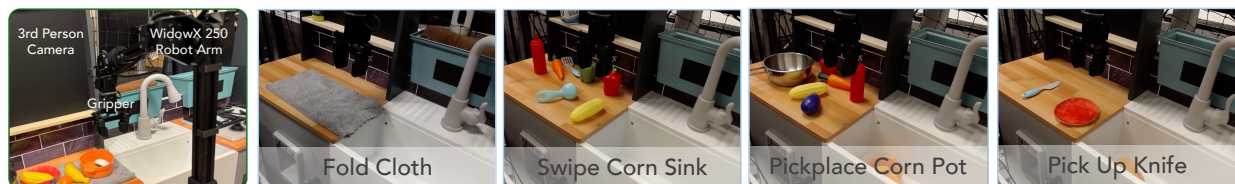


Figure 5.5: Real robot setup of TraceVLA. I design 4 real-world robot tasks with different manipulation skills and objects.

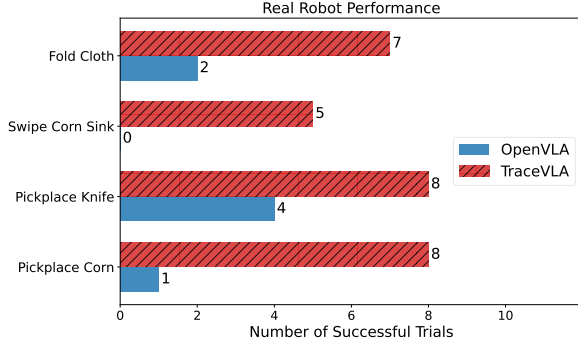


Figure 5.6: Four unseen tasks for testing **TraceVLA** generalization in real robot settings.

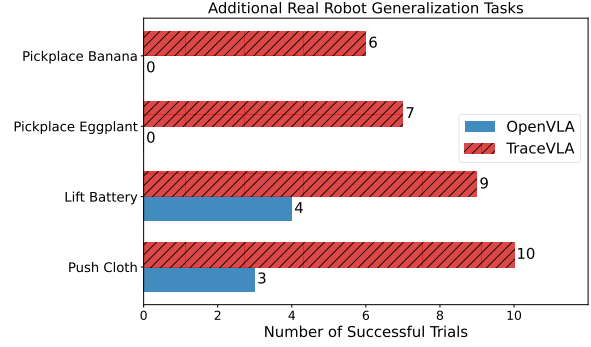
I evaluate TraceVLA on physical WidowX-250 robot manipulation tasks using a fixed-mounted third-person view camera capturing 256×256 RGB images. Despite sharing the same robot embodiment as BridgeData-v2, differences in setup, lighting, and camera angles necessitated collecting 30 demonstration trajectories per task for finetuning.

As shown in Figure 5.7a, TraceVLA consistently outperforms the baseline across diverse tasks including soft object manipulation, pick-and-place operations, and object movement. Notably, in the pick-place corn task, which was not included in training data, TraceVLA achieved 8/10 successful trials compared to OpenVLA’s 1/10, demonstrating strong generalization from similar training tasks, e.g., picking and placing eggplant in a pot.

To further evaluate generalization capabilities, I conducted additional experiments with four unseen tasks involving novel objects, goals, and language instructions. These trials included 2-3 random distracting objects in the scene except for pushing cloth. The results in Figure 5.7b demonstrate TraceVLA’s superior generalization over OpenVLA. In the pick-place banana task, TraceVLA’s



(a) **TraceVLA** outperforms OpenVLA on diverse real-robot manipulation tasks.



(b) **TraceVLA** showcases superior generalization on unseen real robot experiments.

Figure 5.7: Performance comparison of **TraceVLA** and OpenVLA on 8 real-world WidowX-250 robot manipulation tasks.

only failures occurred due to grasping issues, while OpenVLA, even when successfully grasping the banana, failed to follow the language instruction by placing it on the plate rather than to its right. This highlights **TraceVLA**’s enhanced language grounding capability and resistance to spurious correlations.

5.4.3 Ablation Studies

To analyze the performance gain from visual trace prompting, I further study the following questions. **Is performance gain of TraceVLA coming from further finetuning on a smaller subset of Open X-embodiment dataset?** To answer this, I also tested the performance of the 7B OpenVLA and 4B OpenVLA-Phi3 models finetuned on the exact same dataset as ours, but without using visual trace prompting. The results, as shown in Figure 5.8 (Left), indicate that the performance gain observed in **TraceVLA** is unlikely due to finetuning the pretrained VLA model on a smaller subset. While finetuning the pretrained OpenVLA model brings a 1.1% gain, finetuning OpenVLA-Phi3 model even degrades the performance by 0.3%. However, when visual trace prompting is incorporated, the success rate of OpenVLA model increases to 47.7%, highlighting the significant impact of visual traces on model performance.

Will appending historical image observations also give similar performance gain as TraceVLA?

Instead of using visual trace prompting, the most straightforward way to integrate the model’s historical movement knowledge is to provide it with multiple past image observations. To test this method, I input N frames into the VLA model, separated by a learnable separator token, similar to our approach in **TraceVLA**. I also include a sentence in the text prompt to inform the model of this change in observation. Here, $N = 6$, which matches the length of the visual trace used in **TraceVLA**. In theory, the model should receive more information from these 6 complete frames compared to visual trace prompting. However, as shown in 5.8 (Right), finetuning OpenVLA with historical information not only fails to improve overall performance but also reduces it by 6%. This performance drop is likely due to redundant information between visual tokens at different timesteps, which may distract the model from focusing on the most relevant information for deciding next actions. In contrast, visual trace prompting offers useful hints that enhance the context for the vision-language model.

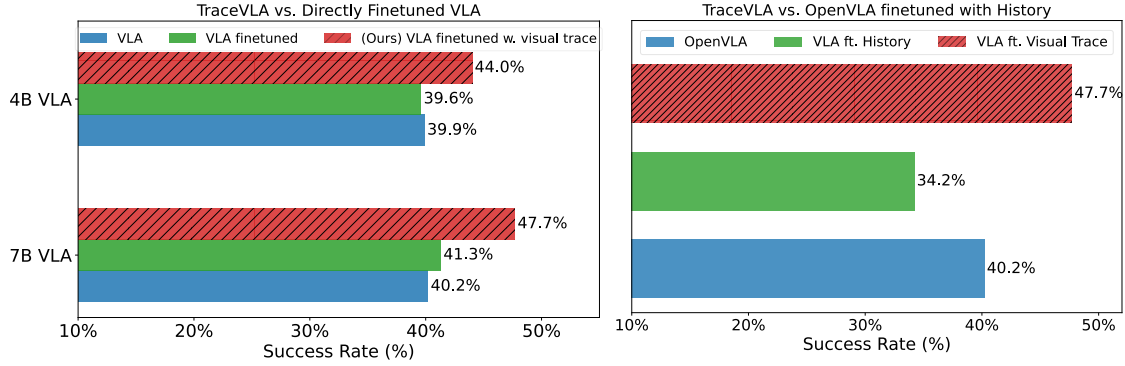


Figure 5.8: (Left): Comparison of average success rates between the base OpenVLA and OpenVLA-Phi3 models and their finetuned versions, with and without visual trace prompting. **(Right):** Comparison of average success rates between the base OpenVLA, **TraceVLA**, and OpenVLA finetuned with a sequence of 6 images.

Is visual trace prompting better than text trace prompting for grounding VLA with temporal-spatial understanding? In addition to guiding the model with point tracking by overlaying the visual trace on the original image, I can also describe the movement of the points using 2D coordinates in text. To assess whether visual prompting is the most effective method for improving the VLA model’s spatial-temporal understanding, I implemented an alternative approach that describes the points’ movement verbally, treating the trace as text tokens, as shown in Figure 5.9 **(Right)**.

Interestingly, using this text-based trace yields a 2.4% average performance gain over the baseline VLA model, suggesting that point tracking information is indeed useful for the robot model. However, compared to our approach (**TraceVLA**), the additional 6.4% gain indicates that leveraging the vision encoder’s ability to process visual prompts is much more effective than describing the points’ location and movement in text. While text descriptions can precisely convey the location and movement of each point, they also increase token count (by ~ 150 tokens) compared with visual trace prompting, leading to substantially higher GPU costs. Moreover, relying solely on text fails to fully leverage the multimodal grounding capabilities of current vision-language models. Our visual trace prompting approach strikes an optimal balance between efficiency and efficacy, demonstrating superior performance gains (an additional 6.4% improvement) over text-based alternatives while maintaining computational efficiency.

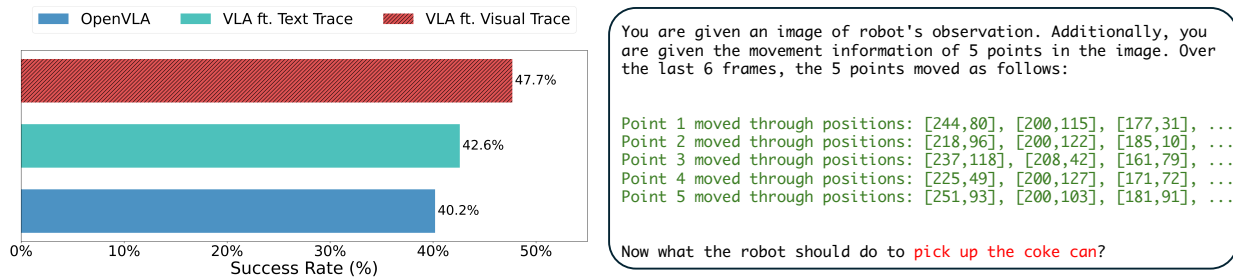


Figure 5.9: (Left): Comparing visual trace prompting and text trace prompting. **(Right)** Text trace prompts example.

How does TraceVLA perform under different lengths of visual trace?

An important hyperparameter of **TraceVLA** is the length of visual trace prompting N , which determines the length of the historical information provided in the visual prompt. While a longer N includes more past observations, it can clutter the visual context and potentially obscure key objects or the robot end-effector, while a shorter N has less risk of covering the key information but contains less historical information. As shown in Figure 5.10, using a smaller number of steps ($N = 3$) results in a 3.2% performance improvement. However, since a visual trace of only three steps is often too brief, the performance gain is not as substantial compared to larger values of N . Conversely, setting N too large, such as $N = 12$, may lead to a less informative trace that obscures key scene elements and overlaps partially with previous motion traces of robots in the overlaid images, potentially distracting the VLM model, thereby slightly reducing overall performance. In practice, tuning N should be straightforward, and might be dataset-specific. It involves sampling a few episodes and visually inspecting the generated trace to ensure that the selected N provides an appropriate balance between historical context and the clarity of the scene.

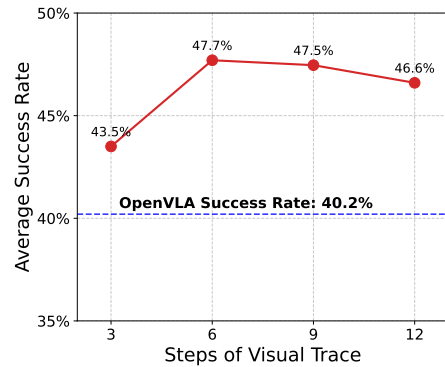


Figure 5.10: TraceVLA under different length of visual traces.

5.5 Limitation Analysis: Training Memory Cost and Inference Speed

Since **TraceVLA** introduces an additional image input into the model and uses CoTracker to obtain the visual trace during testing, I examine both the training memory cost and inference speed of **TraceVLA**. For evaluating memory cost, I launch a single-node multi-gpu training job with 8 H100 graphics cards under varying batch sizes, and I measure the maximum GPU memory usage across 8 H100s. All tests are conducted with flash attention and torch.bfloat16 as the datatype of model weights and inputs. As shown in Figure 5.11 (left), when the batch size is 32, the memory difference between **TraceVLA** and models without visual trace prompting (for both Phi3 and OpenVLA) is less than 10GB. Notably, this difference becomes even smaller with a reduced batch size, indicating that while **TraceVLA** incurs some extra GPU memory cost, this additional GPU memory cost remains manageable and not significantly impactful.

For testing inference speed, I evaluate the time cost using a single H100 GPU. In addition, I also analyze the time cost of each additional component introduced in **TraceVLA** compared to the original OpenVLA model. During inference time, the extra computation introduced by **TraceVLA** consists of approximately 300 additional image and text tokens for the transformer model at each timestep, 5-point CoTracker point tracking for every timestep, and $K \times K$ (where $K = 40$) dense point tracking every 20 steps. For the 40×40 dense point tracking, as it requires recalculation only every 20 steps, the average time cost per timestep is computed by evenly distributing the total cost over each timestep. As shown in Figure 5.11 (right), I observe that the additional text and image tokens result in a negligible inference cost (around 0.002 seconds), likely due to GPU optimization for attention. Using M points ($M = 5$) for CoTracker per timestep adds an extra 0.03 seconds per step, while the dense 40×40 point tracking has an amortized cost of 0.004 seconds per step.

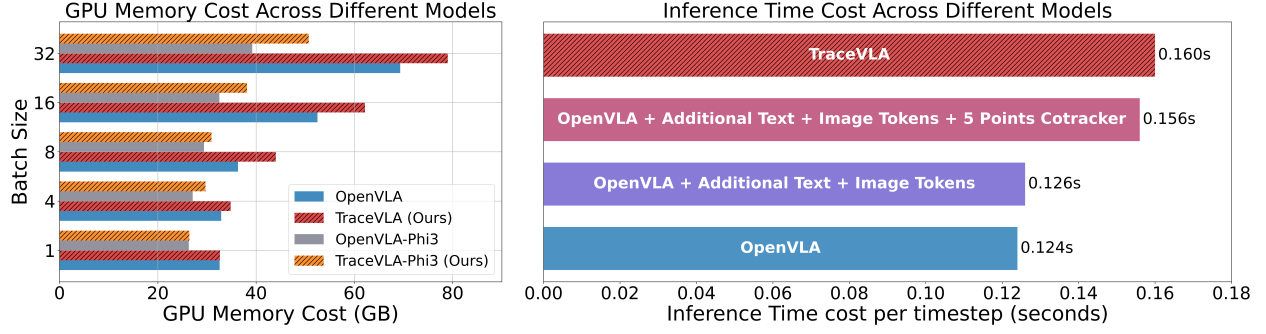


Figure 5.11: (Left): Comparison of GPU memory cost of 7B **TraceVLA**, OpenVLA and 4B **TraceVLA-Phi3**, OpenVLA-Phi3. **(Right):** Comparison of inference time across different models.

5.6 Supplementary Material: Additional Details and Experiments

5.6.1 Additional Ablation Studies of **TraceVLA**: Thickness, Transparency, and Color of Visual Prompting

To further investigate the impact of visualization parameters, I conducted additional ablation studies. Specifically, I fine-tuned the **TraceVLA** model on datasets with variations in trace visualization settings, including different line thicknesses, transparency levels, and color schemes. Our findings indicate that performance variations across these parameters are minimal within a reasonable range. Below, I provide detailed experimental results for each parameter setting.

Thickness: The effect of varying the line thickness of visual traces on the SimplerEnv Average Success Rate is shown in Table 5.2. I observe only minor differences in performance when adjusting this parameter.

Thickness	SimplerEnv Average Success Rate
linewidth=1	47.2%
linewidth=2 (TraceVLA)	47.7%
linewidth=3	47.8%

Table 5.2: Impact of line thickness on **TraceVLA** performance.

Transparency: I varied the transparency of the visual traces by adjusting the α parameter. Lower α values result in more transparent traces. Table 5.3 summarizes the findings, demonstrating the robustness of **TraceVLA**'s performance to these adjustments.

Transparency (α)	SimplerEnv Average Success Rate
$\alpha = 1$ (TraceVLA)	47.7%
$\alpha = 0.8$	47.3%

Table 5.3: Impact of transparency on **TraceVLA** performance.

Color: The choice of color scheme was also tested. The default **TraceVLA** color scheme uses RYPBG (Red, Yellow, Purple, Blue, Green), while an alternative scheme POBGG (Pink, Orange, Blue, Grey, Green) was evaluated. Results are presented in Table 5.4, showing negligible differences in success rates.

Color Scheme	SimplerEnv Average Success Rate
RYPBG (TraceVLA)	47.7%
POBGG	47.3%

Table 5.4: Impact of color scheme on **TraceVLA** performance.

Takeaway: Our experiments reveal that the choice of visualization parameters, including thickness, transparency, and color, has a negligible impact on TraceVLA’s performance when chosen within reasonable ranges. These results suggest that such parameters do not require extensive hyperparameter tuning, simplifying their selection process. This robustness underscores TraceVLA’s reliability across different visualization settings.

5.6.2 Baseline with Different Steps of Historical Observations

In Figure 5.8, I compared TraceVLA with OpenVLA using 6 steps of observation history to ensure both models had access to the same amount of historical information. Here, in Figure 5.12, I further compare TraceVLA with OpenVLA fine-tuned using 2 and 3 steps of observation history on SimplerEnv. While a slight performance improvement is observed with 2-step history for OpenVLA, TraceVLA consistently and significantly outperforms the baseline in success rates, highlighting the effectiveness of visual trace prompting.

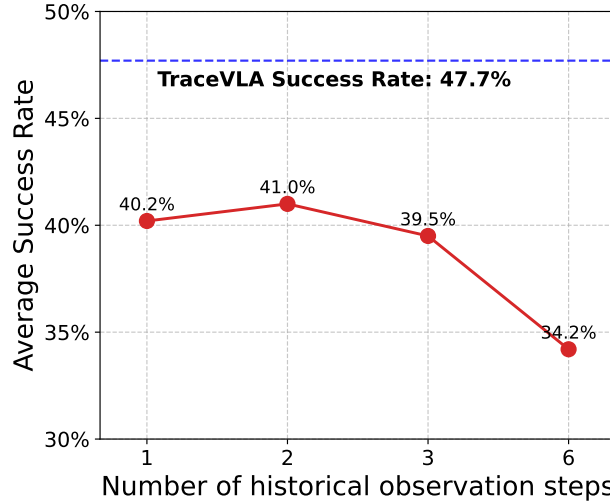


Figure 5.12: Comparison of TraceVLA against OpenVLA with different steps of observation history.

5.6.3 Additional Real Robot Experiments

In this section, I present additional real robot experiments to further evaluate the generalization capability of TraceVLA. I designed four unseen tasks to test the model’s ability to handle novel objects, goals, language instructions, and motion scenarios. Additionally, for each trial except for pushing cloth, I also add 2–3 random distracting objects into the scene.

- **Task 1: pickplace banana:** *Pick up the banana and place it to the right of the plate.* The trial is counted as a success only when the robot correctly picks up the banana and places

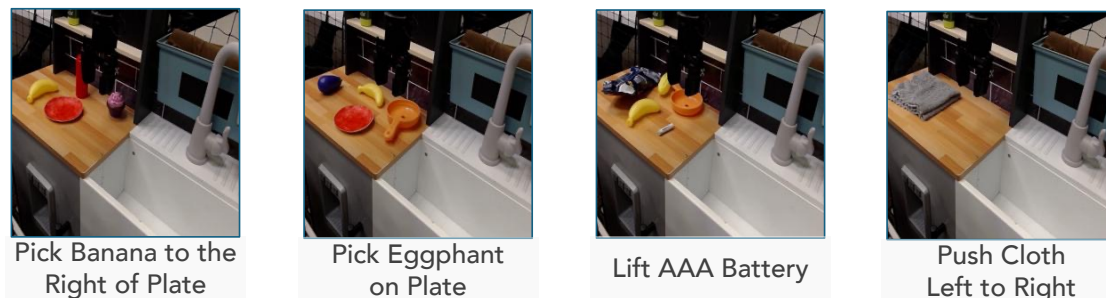


Figure 5.13: Four additional tasks for testing generalization in real robot settings.

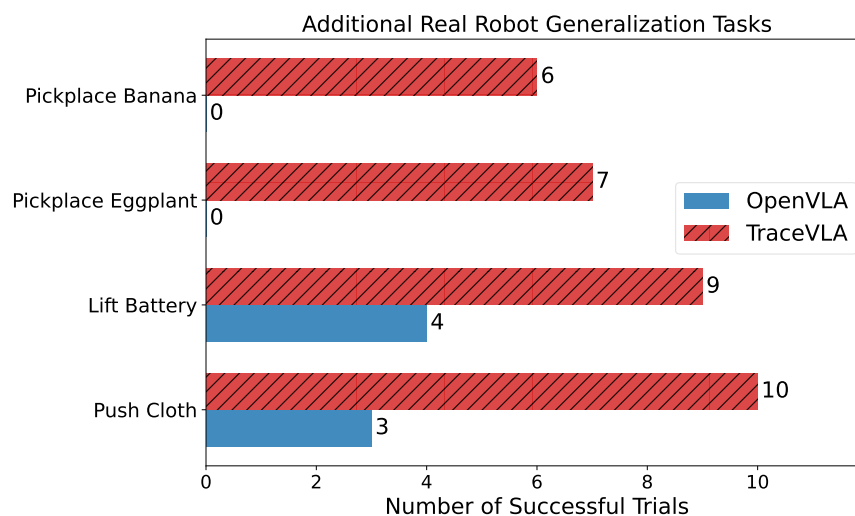


Figure 5.14: Number of successful trials for **TraceVLA** against OpenVLA on four additional real robot experiments.

it to the right of the plate. This task is particularly challenging because the banana object is unseen in our real-robot finetuning dataset. Additionally, solving this task requires the model to leverage its language understanding capability to ground spatial knowledge, rather than relying on spurious correlations, as the instructions in our finetuning dataset only involve placing objects on the plate.

- **Task 2: pickplace eggplant:** *Pick up the eggplant and place it on the plate.* The trial is counted as a success only when the robot correctly places the eggplant on the plate. This task tests the model’s capability for handling unseen goals, as the finetuning dataset only includes placing the eggplant into a pot. Additionally, the eggplant is difficult to grasp, as incorrect placement of the end-effector could easily cause the eggplant to rotate and miss the target.
- **Task 3: lift battery:** *Lift the AAA battery.* The trial is counted as a success only when the robot correctly picks up the battery and lifts it without dropping or damaging it. This task tests the model’s capability to handle unseen objects, as the battery is not included in our finetuning dataset.
- **Task 4: push cloth:** *Push the cloth from the left to the right of the table.* The trial is counted as a success only when the robot successfully pushes the cloth to within 1 inch of the right edge of the table. This task evaluates the model’s motion generalization capability, as the finetuning dataset only includes tasks involving folding cloth.

Analysis and Discussion. Interestingly, I find that **TraceVLA** demonstrates stronger language grounding capability compared to OpenVLA, as evident in the *pickplace banana* task, where the model successfully interprets the spatial relation “to the right of the plate” despite never encountering such an instruction during finetuning. This suggests that **TraceVLA** is not merely overfitting to the finetuning data distribution but instead learns to condition its actions based on natural language instructions. Moreover, across all four unseen tasks, **TraceVLA** consistently achieves higher success rates, reflecting its improved robustness to novel objects and goals. I attribute this enhanced generalization to **TraceVLA**’s temporally grounded visual-language alignment, which enables more coherent reasoning over sequential observations and instructions.

Chapter 6: Conclusion and Future Direction

6.1 Conclusion

This thesis studies representation learning for embodied agents by developing a unified framework grounded in the principle of world-model-based latent representation learning. Across diverse embodiments, modalities, and tasks, the central insight is that learning to model the dynamics of the world, rather than merely imitating actions, provides a foundation for scalable and generalizable intelligence in sequential decision-making systems.

Through this perspective, the thesis contributes along three major dimensions. First, it establishes that learning compact latent state representations can substantially improve the efficiency and generalization of downstream policy learning. TACO and Premier-TACO demonstrate how temporal contrastive learning objectives can encode predictive structure and control-relevant dynamics, yielding transferable visual representations that enable few-shot task adaptation. Building upon this principle, FLARE extends the world-model formulation to large vision-language-action (VLA) foundation models by incorporating a future latent alignment loss, enabling these models to learn from action-free human video data and bridge human observation with robotic execution.

Second, the thesis shows that the same world-model objective can guide the learning of temporal action abstractions. PRISE introduces a framework for discovering temporally extended actions directly from raw trajectories, shortening the effective planning horizon in complex control problems. By discretizing and tokenizing long-horizon behaviors, PRISE connects world modeling with the hierarchical structure of action generation, leading to more efficient and robust multitask policy learning.

Finally, the thesis explores the complementary role of symbolic representations in grounding large multimodal models. TraceVLA demonstrates that explicit visual traces, symbolic encodings of execution history, can serve as powerful visual prompts, enhancing a model’s ability to connect perception with executable actions. This approach provides large VLA models with richer spatio-temporal grounding and achieves significant improvements in real-robot manipulation tasks.

Together, these contributions outline a cohesive trajectory toward scalable generalist embodied agents. By combining predictive world modeling, latent abstraction, and symbolic grounding, this work moves beyond task-specific imitation to a deeper understanding of how agents can learn to reason, plan, and act in dynamic environments.

6.2 Future Work

Building upon this foundation, the next step toward truly general-purpose embodied intelligence is scaling world-model-based learning to large and diverse data. World models thrive on rich and varied interaction data that captures the underlying physics of the real world. Thus, expanding their

training distribution beyond narrowly collected robot demonstrations is both natural and necessary. To reach the capability frontier observed in language models, robotic foundation models will require orders of magnitude more diverse and physically grounded data covering the full spectrum of interactions and embodiments.

However, collecting such large-scale data purely through robot teleoperation is prohibitively expensive. Unlike text or image corpora that can be scaled automatically, teleoperation data collection demands human operators, robotic platforms, and specialized motion-tracking devices such as exoskeletons and mocap suit, all constrained by hardware availability and human effort. As a result, current datasets remain several orders of magnitude smaller than those needed to train truly generalist models.

A promising path forward lies in leveraging the most abundant source of interaction data available—humans themselves. Humans perform manipulation activities continuously in everyday life, across diverse environments and with a wide variety of objects and goals. If we can capture this rich stream of naturalistic human activity through lightweight, wearable devices such as cameras, inertial sensors, or gloves, and translate it into robot-trainable representations, we can unlock an unprecedented scale of data for embodied learning. Such data would enable robots to learn from both expert demonstrations and everyday human behaviors, improving their understanding of how the physical world works.

In this thesis, I have already taken preliminary steps in this direction through FLARE, which showed that even action-free human video data can contribute meaningfully to robot policy learning via latent alignment objectives. Nevertheless, these datasets still involve humans mimicking robot motions rather than capturing natural human behavior in unconstrained settings. Future research should therefore explore how to bridge this gap by combining advances in computer vision, such as pose estimation and motion retargeting, to map human activities into robot-executable representations. Once such mappings are established, world-model-based learning objectives can serve as a powerful framework to digest and learn from this heterogeneous data, just as demonstrated in this thesis for robot-collected teleoperation data. By doing so, we can train robots on truly large, diverse, and physically grounded data, enabling embodied intelligence to scale in the same way that language models scaled with text.

Bibliography

- [1] Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Weizhu Chen, Yen-Chun Chen, Yi-Ling Chen, Hao Cheng, Parul Chopra, Xiyang Dai, Matthew Dixon, Ronen Eldan, Victor Fragoso, Jianfeng Gao, Mei Gao, Min Gao, Amit Garg, Allie Del Giorno, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Wenxiang Hu, Jamie Huynh, Dan Iter, Sam Ade Jacobs, Mojan Javaheripi, Xin Jin, Nikos Karampatziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Xihui Lin, Zeqi Lin, Ce Liu, Liyuan Liu, Mengchen Liu, Weishung Liu, Xiaodong Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola, Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Liliang Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacrose, Shital Shah, Ning Shang, Hiteshi Sharma, Yelong Shen, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Praneetha Vaddamanu, Chunyu Wang, Guanhua Wang, Lijuan Wang, Shuohang Wang, Xin Wang, Yu Wang, Rachel Ward, Wen Wen, Philipp Witte, Haiping Wu, Xiaoxia Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Jilong Xue, Sonali Yadav, Fan Yang, Jianwei Yang, Yifan Yang, Ziyi Yang, Donghan Yu, Lu Yuan, Chenruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. Phi-3 technical report: A highly capable language model locally on your phone, 2024.
- [2] Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.
- [3] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [4] Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. OPAL: Offline primitive discovery for accelerating offline reinforcement learning. In *ICLR*, 2021.

- [5] Arthur Allshire, Roberto Martín-Martín, Charles Lin, Shawn Manuel, Silvio Savarese, and Animesh Garg. Laser: Learning a latent action space for efficient reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6650–6656. IEEE, 2021.
- [6] Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised state representation learning in atari. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [7] Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from human videos as a versatile representation for robotics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.
- [8] Andrew Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications*, 13:41–77, 2003.
- [9] Homanga Bharadhwaj, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. Zero-shot robot manipulation from passive human videos. *arXiv preprint arXiv:2302.02011*, 2023.
- [10] Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2act: Predicting point tracks from internet videos enables diverse zero-shot robot manipulation. *arXiv e-prints*, pages arXiv–2405, 2024.
- [11] 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. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024.
- [12] 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.
- [13] 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, Jaspier 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.
- [14] 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.
- [15] 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.
- [16] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2212.06817*, 2022.
- [17] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.
- [18] Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments, 2024.
- [19] Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xu Huang, Shu Jiang, et al. Agibot world colosseum: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.
- [20] Edoardo Cetin, Philip J Ball, Stephen Roberts, and Oya Celiktutan. Stabilizing off-policy deep reinforcement learning from pixels. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International*

Conference on Machine Learning, volume 162 of *Proceedings of Machine Learning Research*, pages 2784–2810. PMLR, 17–23 Jul 2022.

- [21] Yash Chandak, Georgios Theodorou, James Kostas, Scott Jordan, and Philip Thomas. Learning action representations for reinforcement learning. In *International conference on machine learning*, pages 941–950. PMLR, 2019.
- [22] Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, Hanbo Zhang, and Minzhao Zhu. Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.
- [23] Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, Hanbo Zhang, and Minzhao Zhu. Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.
- [24] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 15084–15097. Curran Associates, Inc., 2021.
- [25] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020.
- [26] Yi Chen, Yuying Ge, Weiliang Tang, Yizhuo Li, Yixiao Ge, Mingyu Ding, Ying Shan, and Xihui Liu. Moto: Latent motion token as the bridging language for learning robot manipulation from videos, 2025.
- [27] Jin Cheng, Dongho Kang, Gabriele Fadini, Guanya Shi, and Stelian Coros. Rambo: RL-augmented model-based optimal control for whole-body loco-manipulation, 2025.
- [28] 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*, 2024.
- [29] Open X-Embodiment Collaboration, Abby O’Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel

Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booyer, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minh Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaesan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Mart'in-Mart'in, Rohan Bajjal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open X-Embodiment: Robotic learning datasets and RT-X models. <https://arxiv.org/abs/2310.08864>, 2023.

- [30] Open-X Embodiment Collaboration, A Padalkar, A Pooley, A Jain, A Bewley, A Herzog, A Irpan, A Khazatsky, A Rai, A Singh, et al. Open x-embodiment: Robotic learning datasets

and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

- [31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [32] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [33] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.
- [34] Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- [35] Yilun Du, Sherry Yang, Pete Florence, Fei Xia, Ayzaan Wahid, brian ichter, Pierre Sermanet, Tianhe Yu, Pieter Abbeel, Joshua B. Tenenbaum, Leslie Pack Kaelbling, Andy Zeng, and Jonathan Tompson. Video language planning. In *The Twelfth International Conference on Learning Representations*, 2024.
- [36] Scott Fujimoto and Shixiang (Shane) Gu. A minimalist approach to offline reinforcement learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 20132–20145. Curran Associates, Inc., 2021.
- [37] Philip Gage. A new algorithm for data compression. *C Users Journal*, 12(2):23—38, 1994.
- [38] Yu Gao, Lixue Gong, Qiushan Guo, Xiaoxia Hou, Zhichao Lai, Fanshi Li, Liang Li, Xiaochen Lian, Chao Liao, Liyang Liu, et al. Seedream 3.0 technical report. *arXiv preprint arXiv:2504.11346*, 2025.
- [39] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pages 2170–2179. PMLR, 2019.
- [40] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284. Curran Associates, Inc., 2020.

- [41] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *CoRL*, 2019.
- [42] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.
- [43] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565. PMLR, 09–15 Jun 2019.
- [44] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.
- [45] Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.
- [46] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2023.
- [47] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. 2022.
- [48] Nicklas Hansen, Zhecheng Yuan, Yanjie Ze, Tongzhou Mu, Aravind Rajeswaran, Hao Su, Huazhe Xu, and Xiaolong Wang. On pre-training for visuo-motor control: Revisiting a learning-from-scratch baseline. In *CoRL 2022 Workshop on Pre-training Robot Learning*, 2022.
- [49] Nicklas A Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8387–8406. PMLR, 17–23 Jul 2022.
- [50] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *ICLR*, 2018.
- [51] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735, 2020.
- [52] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

- [53] Olivier Henaff. Data-efficient image recognition with contrastive predictive coding. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4182–4192. PMLR, 13–18 Jul 2020.
- [54] Pu Hua, Yubei Chen, and Huazhe Xu. Simple emergent action representations from multi-task policy training. *arXiv preprint arXiv:2210.09566*, 2022.
- [55] Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. An embodied generalist agent in 3d world. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.
- [56] Shuaiyi Huang, Mara Levy, Zhenyu Jiang, Anima Anandkumar, Yuke Zhu, Linxi Fan, De-An Huang, and Abhinav Shrivastava. Ardup: Active region video diffusion for universal policies, 2025.
- [57] Hong Jun Jeon, Dylan Losey, and Dorsa Sadigh. Shared Autonomy with Learned Latent Actions. In *Proceedings of Robotics: Science and Systems*, Corvallis, Oregon, USA, July 2020.
- [58] Nan Jiang, Alex Kulesza, Satinder Singh, and Richard Lewis. The dependence of effective planning horizon on model accuracy. In *AAMAS*, 2015.
- [59] Xiaoyu Jiang, Qiuxuan Chen, Shiyi Han, Mingxuan Li, Jingyan Dong, and Ruochen Zhang. When to trust your model: Model-based policy optimization, 2020. Submitted to NeurIPS 2019 Reproducibility Challenge.
- [60] Zhengyao Jiang, Tianjun Zhang, Michael Janner, Yueying Li, Tim Rocktäschel, Edward Grefenstette, and Yuandong Tian. Efficient planning in a compact latent action space. In *ICLR*, 2023.
- [61] Aditya Kannan, Kenneth Shaw, Shikhar Bahl, Pragna Mannam, and Deepak Pathak. Deft: Dexterous fine-tuning for real-world hand policies. *arXiv preprint arXiv:2310.19797*, 2023.
- [62] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. *arXiv:2307.07635*, 2023.
- [63] Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. In *International Conference on Machine Learning (ICML)*, 2024.
- [64] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong,

- Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, David Antonio Herrera, Minh Heo, Kyle Hsu, Jiaheng Hu, Donovan Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O’Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.
- [65] Minbeom Kim, Kyeongha Rho, Yong-duk Kim, and Kyomin Jung. Action-driven contrastive representation for reinforcement learning. *PLOS ONE*, 17(3):1–14, 03 2022.
- [66] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.
- [67] 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.
- [68] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, 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 arXiv:2406.09246*, 2024.
- [69] Durk Kingma and Max Welling. Auto-encoding variational Bayes. In *NIPS*, 2014.
- [70] Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Alvaro Sanchez-Gonzalez, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. Compile: Compositional imitation learning and execution. In *ICML*, 2019.
- [71] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.
- [72] Sanjay Krishnan, Roy Fox, Ion Stoica, and Ken Goldberg. DDCO: Discovery of deep continuous options for robot learning from demonstrations. In *CoRL*, 2017.
- [73] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*, 2018.
- [74] Kalle Kujanpää, Joni Pajarinen, and Alexander Ilin. Hierarchical imitation learning with vector quantized models. In *ICML*, 2023.
- [75] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan,

- and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191. Curran Associates, Inc., 2020.
- [76] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5639–5650. PMLR, 13–18 Jul 2020.
 - [77] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19884–19895. Curran Associates, Inc., 2020.
 - [78] Hoang M. Le, Nan Jiang, Alekh Agarwal, Miroslav Dudík, Yisong Yue, and Hal Daumé III au2. Hierarchical imitation and reinforcement learning. In *ICML*, 2018.
 - [79] Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *5th Annual Conference on Robot Learning*, 2021.
 - [80] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. 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 19730–19742. PMLR, 23–29 Jul 2023.
 - [81] Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. *arXiv preprint arXiv:2503.00200*, 2025.
 - [82] Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, et al. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*, 2023.
 - [83] Zhiqi Li, Guo Chen, Shilong Liu, Shihao Wang, Vibashan VS, Yishen Ji, Shiyi Lan, Hao Zhang, Yilin Zhao, Subhashree Radhakrishnan, et al. Eagle 2: Building post-training data strategies from scratch for frontier vision-language models. *arXiv preprint arXiv:2501.14818*, 2025.
 - [84] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*.
 - [85] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, qiang liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

- [86] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.
- [87] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.
- [88] Jianlan Luo, Perry Dong, Jeffrey Wu, Aviral Kumar, Xinyang Geng, and Sergey Levine. Action-quantized offline reinforcement learning for robotic skill learning. In *CoRL*, 2023.
- [89] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *CoRL*, 2019.
- [90] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. In *RSS*, 2021.
- [91] 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, 2022.
- [92] 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.
- [93] 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.
- [94] Bogdan Mazoure, Remi Tachet des Combes, Thang Long Doan, Philip Bachman, and R Devon Hjelm. Deep reinforcement and infomax learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3686–3698. Curran Associates, Inc., 2020.
- [95] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *NeurIPS*, 2018.
- [96] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. In *6th Annual Conference on Robot Learning*, 2022.
- [97] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024.
- [98] NVIDIA, :, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llonet, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott

Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025.

- [99] 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, Lawrence Yunliang Chen, 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.
- [100] OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Gertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O’Connell, Ian O’Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward,

Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavín Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljube, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huét, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunningham, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiye Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. Gpt-4o system card, 2024.

- [101] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.
- [102] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khilodov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.
- [103] Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta. The un-

- surprising effectiveness of pre-trained vision models for control. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17359–17371. PMLR, 17–23 Jul 2022.
- [104] Seohong Park and Sergey Levine. Predictable mdp abstraction for unsupervised model-based rl, 2023.
 - [105] Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. In *NIPS*, pages 1043–1049, 1998.
 - [106] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *ICRA*, 2009.
 - [107] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.
 - [108] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.
 - [109] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.
 - [110] 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, 2025.
 - [111] Martin L Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley and Sons, 1994.
 - [112] Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. In *European Conference on Computer Vision*, 2022.
 - [113] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
 - [114] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021.
 - [115] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

- [116] Nived Rajaraman, Lin Yang, Jiantao Jiao, and Kannan Ramchandran. Toward the fundamental limits of imitation learning. 2020.
- [117] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollar, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [118] Zhongwei Ren, Yunchao Wei, Xun Guo, Yao Zhao, Bingyi Kang, Jiashi Feng, and Xiaojie Jin. Videoworld: Exploring knowledge learning from unlabeled videos, 2025.
- [119] Erick Rosete-Beas, Oier Mees, Gabriel Kalweit, Joschka Boedecker, and Wolfram Burgard. Latent plans for task agnostic offline reinforcement learning. 2022.
- [120] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.
- [121] Dominik Schmidt and Minqi Jiang. Learning to act without actions. In *The Twelfth International Conference on Learning Representations*, 2024.
- [122] Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021.
- [123] Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021.
- [124] Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron Courville. Pretraining representations for data-efficient reinforcement learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.
- [125] Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. Pretraining representations for data-efficient reinforcement learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12686–12699. Curran Associates, Inc., 2021.
- [126] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *ACL*, 2016.
- [127] Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. In *CoRL*, volume 205 of *Proceedings of Machine Learning Research*, pages 1332–1344. PMLR, 2022.
- [128] Rutav Shah, Roberto Martín-Martín, and Yuke Zhu. Mutex: Learning unified policies from multimodal task specifications. In *7th Annual Conference on Robot Learning*, 2023.

- [129] Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. Videodex: Learning dexterity from internet videos. In *Conference on Robot Learning*, 2023.
- [130] Mohan Kumar Srirama, Sudeep Dasari, Shikhar Bahl, and Abhinav Gupta. Hrp: Human affordances for robotic pre-training. *arXiv preprint arXiv:2407.18911*, 2024.
- [131] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9870–9879. PMLR, 18–24 Jul 2021.
- [132] Yanchao Sun, Shuang Ma, Ratnesh Madaan, Rogerio Bonatti, Furong Huang, and Ashish Kapoor. SMART: Self-supervised multi-task pretraining with control transformers. In *The Eleventh International Conference on Learning Representations*, 2023.
- [133] Richard Sutton and Andrew Barto. *Reinforcement learning: An introduction*. The MIT Press, 2nd edition, 2018.
- [134] Richard Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.
- [135] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite, 2018.
- [136] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite, 2018.
- [137] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.
- [138] Garrett Thomas, Ching-An Cheng, Ricky Loynd, Felipe Vieira Frujeri, Vibhav Vineet, Mihai Jalobeanu, and Andrey Kolobov. Plex: Making the most of the available data for robotic manipulation pretraining. In *CoRL*, 2023.
- [139] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [140] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.
- [141] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.

- [142] Aaron Van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [143] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, 2017.
- [144] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.
- [145] Chen Wang, Linxi Fan, Jiankai Sun, Ruohan Zhang, Li Fei-Fei, Danfei Xu, Yuke Zhu, and Anima Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. *arXiv preprint arXiv:2302.12422*, 2023.
- [146] Xiyao Wang, Ruijie Zheng, Yanchao Sun, Ruonan Jia, Wichayaporn Wongkamjan, Huazhe Xu, and Furong Huang. COPlanner: Plan to roll out conservatively but to explore optimistically for model-based RL. In *The Twelfth International Conference on Learning Representations*, 2024.
- [147] Yao Wei, Yanchao Sun, Ruijie Zheng, Sai Vemprala, Rogerio Bonatti, Shuhang Chen, Ratnesh Madaan, Zhongjie Ba, Ashish Kapoor, and Shuang Ma. Is imitation all you need? generalized decision-making with dual-phase training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16221–16231, October 2023.
- [148] Chuan Wen, Xingyu Lin, John So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning. *arXiv preprint arXiv:2401.00025*, 2023.
- [149] Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Kun Wu, Zhiyuan Xu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2409.12514*, 2024.
- [150] Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *The Twelfth International Conference on Learning Representations*, 2024.
- [151] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018.
- [152] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control, 2022.
- [153] Tengyang Xie, Ching-An Cheng, Nan Jiang, Paul Mineiro, and Alekh Agarwal. Bellman-consistent pessimism for offline reinforcement learning. In *NeurIPS*, 2021.

- [154] Guowei Xu, Ruijie Zheng, Yongyuan Liang, Xiyao Wang, Zhecheng Yuan, Tianying Ji, Yu Luo, Xiaoyu Liu, Jiaxin Yuan, Pu Hua, Shuzhen Li, Yanjie Ze, Hal Daumé III, Furong Huang, and Huazhe Xu. Drm: Mastering visual reinforcement learning through dormant ratio minimization. In *The Twelfth International Conference on Learning Representations*, 2024.
- [155] Jianwei Yang, Reuben Tan, Qianhui Wu, Ruijie Zheng, Baolin Peng, Yongyuan Liang, Yu Gu, Mu Cai, Seonghyeon Ye, Joel Jang, Yuquan Deng, Lars Liden, and Jianfeng Gao. Magma: A foundation model for multimodal ai agents, 2025.
- [156] Jingyun Yang, Zi-ang Cao, Congyue Deng, Rika Antonova, Shuran Song, and Jeannette Bohg. Equibot: Sim (3)-equivariant diffusion policy for generalizable and data efficient learning. *arXiv preprint arXiv:2407.01479*, 2024.
- [157] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *International Conference on Learning Representations*, 2022.
- [158] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021.
- [159] Jianglong Ye, Jiashun Wang, Binghao Huang, Yuzhe Qin, and Xiaolong Wang. Learning continuous grasping function with a dexterous hand from human demonstrations. *IEEE Robotics and Automation Letters*, 8(5):2882–2889, 2023.
- [160] Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [161] Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.
- [162] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [163] 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 (CoRL)*, 2019.
- [164] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Avnish Narayan, Hayden Shively, Adithya Bellathur, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2021.

- [165] Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction in robotics. In *8th Annual Conference on Robot Learning*, 2024.
- [166] Jia Zeng, Qingwen Bu, Bangjun Wang, Wenke Xia, Li Chen, Hao Dong, Haoming Song, Dong Wang, Di Hu, Ping Luo, et al. Learning manipulation by predicting interaction. *arXiv preprint arXiv:2406.00439*, 2024.
- [167] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986, 2023.
- [168] Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021.
- [169] Qihang Zhang, Zhenghao Peng, and Bolei Zhou. Learning to drive by watching youtube videos: Action-conditioned contrastive policy pretraining. *European Conference on Computer Vision (ECCV)*, 2022.
- [170] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Ming-Yu Liu, Donglai Xiang, Gordon Wetzstein, and Tsung-Yi Lin. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models, 2025.
- [171] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *RSS*, 2023.
- [172] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023.
- [173] 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.
- [174] Ruijie Zheng, Ching-An Cheng, Hal Daumé III, Furong Huang, and Andrey Kolobov. PRISE: LLM-style sequence compression for learning temporal action abstractions in control. In *Forty-first International Conference on Machine Learning*, 2024.
- [175] 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. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [176] Ruijie Zheng, Yongyuan Liang, Xiyao Wang, Shuang Ma, Hal Daumé III au2, Huazhe Xu, John Langford, Praveen Palanisamy, Kalyan Shankar Basu, and Furong Huang. Premier-taco

is a few-shot policy learner: Pretraining multitask representation via temporal action-driven contrastive loss, 2024.

- [177] Ruijie Zheng, Yongyuan Liang, Xiyao Wang, Shuang Ma, Hal Daumé III, Huazhe Xu, John Langford, Praveen Palanisamy, Kalyan Shankar Basu, and Furong Huang. Premier-taco is a few-shot policy learner: pretraining multitask representation via temporal action-driven contrastive loss. In *Proceedings of the 41st International Conference on Machine Learning, ICML’24*. JMLR.org, 2024.
- [178] Ruijie Zheng, Jing Wang, Scott Reed, Johan Bjorck, Yu Fang, Fengyuan Hu, Joel Jang, Kaushil Kundalia, Zongyu Lin, Loïc Magne, Avnish Narayan, You Liang Tan, Guanzhi Wang, Qi Wang, Jiannan Xiang, Yinzhen Xu, Seonghyeon Ye, Jan Kautz, Furong Huang, Yuke Zhu, and Linxi Fan. FLARE: Robot learning with implicit world modeling. In *9th Annual Conference on Robot Learning*, 2025.
- [179] Ruijie Zheng, Xiyao Wang, Yanchao Sun, Shuang Ma, Jieyu Zhao, Huazhe Xu, Hal Daumé III, and Furong Huang. TACO: Temporal latent action-driven contrastive loss for visual reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- [180] Ruijie Zheng, Xiyao Wang, Huazhe Xu, and Furong Huang. Is model ensemble necessary? model-based RL via a single model with lipschitz regularized value function. In *The Eleventh International Conference on Learning Representations*, 2023.
- [181] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.
- [182] Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. 2025.
- [183] Yifeng Zhu, Arisrei Lim, Peter Stone, and Yuke Zhu. Vision-based manipulation from single human video with open-world object graphs. *arXiv preprint arXiv:2405.20321*, 2024.