PRISE: LLM-Style Sequence Compression for Learning Temporal Action Abstractions in Control

Ruijie Zheng [1 2] **Ching-An Cheng** [3] **Hal Daumé III** [1 3] **Furong Huang** [1] **Andrey Kolobov** [3]

The code is available at:
https://ruijiezheng.com/project/PRISE/index.html

Abstract

Temporal action abstractions, along with belief state representations, are a powerful knowledge sharing mechanism for sequential decision making. In this work, we propose a novel view that treats inducing temporal action abstractions as a sequence compression problem. To do so, we bring a subtle but critical component of LLM training pipelines – input tokenization via byte pair encoding (BPE) – to bear on the seemingly distant task of learning skills of variable time span in continuous control domains. We introduce an approach called Primitive Sequence Encoding (PRISE) that combines continuous action quantization with BPE to learn powerful action abstractions. We empirically show that high-level skills discovered by PRISE from a multitask set of robotic manipulation demonstrations significantly boost the performance of Behavior Cloning on downstream tasks.

1. Introduction

High-dimensional observations and decision-making over complex, continuous action spaces are hallmarks of typical scenarios in sequential decision-making, including robotics. A common way of dealing with these complexities is constructing *abstractions* – compact representations of belief states and actions that generalize across tasks and make learning to act in new scenarios robust and data-efficient. Inspired by techniques that have enabled ground-breaking progresses in computer vision (CV) and natural language processing (NLP) over the past decade, much of continuous control research has focused on *learning* abstractions from data rather than hand-crafted them. The lion's share of these efforts study learning multi-task representations, which has analogies to representation learning in CV and NLP and has been tackled by adapting these fields' models (see, e.g., DT (Chen et al., 2021) vs. GPT-2 (Radford et al., 2019)) and methods (see, e.g., R3M (Nair et al., 2022) vs. InfoNCE (den Oord et al., 2019; Zheng et al., 2023a)).

On the other hand, learning *temporal action abstractions* – representations of multi-step primitive behaviors – has not benefited nearly as much from such a methodology transfer. This omission is especially glaring in continuous control, where complex policies can clearly be decomposed into versatile lower-level routines such as picking up objects, walking, etc, and whose popular learning method, Behavior Cloning (BC) (Arora et al., 2022), has many commonalities with LLM training.

In this paper, we posit that adapting discrete coding and sequence compression techniques from NLP offers 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, we consider the problem of learning temporally extended action primitives, i.e., *skills*, to improve downstream learning efficiency of reward-free BC. We show that embedding continuous actions into *discrete* codes and then applying a popular NLP sequence compression method called *Byte Pair Encoding (BPE)* (Gage, 1994) to the resulting discrete-code sequences identifies variable-timespan action primitives with the property we 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.

Our work's main contribution is ***Pri**mitive **S**equence **E**ncoding (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

[1]Department of Computer Science, University of Maryland, College Park [2]Part of the work done while at Microsoft [3]Microsoft Research. Correspondence to: Ruijie Zheng <rzheng12@umd.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

original continuous action space into discrete codes, converts the pretraining training 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 (Zhang et al., 2021). We conduct extensive ablation studies to show the effect of various parameters on PRISE's performance, which demonstrate BPE to be critical to PRISE's success.

2. Preliminaries

Problem Setting. We consider a set of tasks T, denoted as $\{(\mathcal{O}, \mathcal{A}, \mathcal{P}, g_{\mathcal{T}})\}_{\mathcal{T} \in \mathscr{T}}$, 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 we focus on learning visuomotor continuous control policies, we assume $\mathcal{A}$ is a finite-dimensional vector space and $\mathcal{O}$ is the image space.

Let $\mathcal{D} = \bigcup_{\mathcal{T} \in \mathscr{T}} \mathcal{D}_{\mathcal{T}}$ 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 we call *skills*. Note that our concept of a skill is different from some others in prior work; we compare them in Section 4. 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, we define the skill token vocabulary as $\mathcal{V} = \{\xi_1, \ldots, \xi_K\}$, 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 tokens, our algorithm PRISE first converts continuous actions into discrete codes. To do so, it uses the vector quantization module proposed in Van den Oord et al. (2017). This vector quantization module $\mathcal{F}$ maintains a collection $\mathcal{E} = \{e_1, ..., e_C\}$ of C codes. For any query vector q (in our case, q is a concatenation of latent state and action, as described in Section 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 w.r.t. the loss $\mathcal{L}[\theta_{\mathcal{F}}] = \|\text{stopgrad}(q) - e_k\|_2 + \|q - \text{stopgrad}(e_k)\|_2$, where stopgrad(.) 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) (Gage, 1994; Sennrich et al., 2016). 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 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.

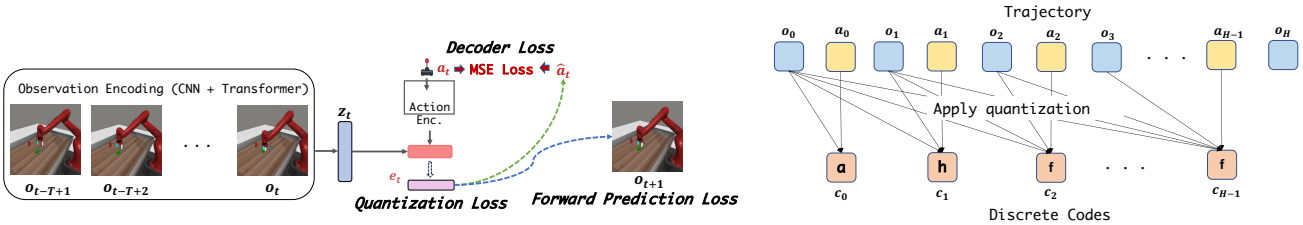
BPE operates by merging the most frequent pairs of bytes, assigning tokens to these pairs, and iteratively merging new pairs of tokens or bytes. Thereby, it builds up a vocabulary of tokens that plays a crucial role in modern large language models (OpenAI, 2023; Brown et al., 2020; Radford et al., 2019). See Figure 2 for a demonstration. We apply BPE in the same way, but to action codes instead of bytes.

3. Algorithm

Figure 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 ⟨observation, continuous action⟩ pairs – into a sequence of discrete codes, one code per time step, as shown in Figure 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 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 we later show in Section 3.2 and Section 3.3. In the rest of the section, we describe PRISE in more detail.

3.1. Pretraining

Pretraining I: action quantization. This stage is illustrated in Figure 1(a), and its pseudocode is available in Figure 13 in Appendix C. Let $\mathcal{G} : \mathcal{O}^T \to \mathcal{Z}$ be the observation embedding which embeds a sequence of pixel observations of length T into the latent embedding space $\mathcal{Z}$. In our implementation, we use the transformer architecture (Vaswani et al., 2017) and set $z_t = \mathcal{G}(o_t, o_{t-1}, ..., o_{t-T+1}) =$



(a) Stage I (b) Stage II

Figure 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 2) to learn vocabulary of skill tokens, where each token represents a sequence of discrete action code.

Transformer_block$(\text{CNN}(o_t), , ..., \text{CNN}(o_{t-T+1}))$. Using the multitask dataset $\mathcal{D}$, we jointly pretrain the observation embedding $\mathcal{G}$ and a state-dependent action quantization module $\mathcal{F} : \mathcal{Z} \times \mathcal{A} \to \mathcal{E}$, where $\mathcal{E} = \{e_1, e_2, ..., e_C\}$ 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 we 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., (Chandak et al., 2019), although that work doesn't attempt to simultaneously learn a latent state space.

First, to ensure the action code can predict future states, we train a latent forward transition model $\mathcal{T} : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$. To optimize the forward latent dynamics model while preventing state and action representation collapse, we adopt a BYOL-like objective inspired from (Schwarzer et al., 2021), where we minimize

$$\mathcal{L}_{\text{dyn}}[\theta_{\mathcal{F},\mathcal{G},Q,P}](o_{t,T}, a_t) = \\ -\,\textbf{cos_sim}\big(Q(P(\mathcal{T}(z_t, e_t))),\ \text{stopgrad}(P(z_{t+1}))\big) \tag{1}$$

Here, $o_{t,T} = o_t, o_{t-1}, ..., o_{t-T+1}$ is the observation history, $z_t = \mathcal{G}(o_t, o_{t-1}, ..., o_{t-T+1})$ 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 two layers and Q has one.

Next, to guarantee the raw action can be decoded from the action code and latent state effectively, we 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, we let ψ be a stochastic mapping $\psi : \mathcal{Z} \times \mathcal{E} \to \Delta(\mathcal{A})$, where $\Delta(\mathcal{A})$ stands for the space of probability distributions over $\mathcal{A}$. We use a stochastic Gaussian Mixture Model (GMM) to parameterize the decoded action distribution of ψ following Mandlekar et al. (2021). 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: $\mathcal{L}_{\text{act_decode}}[\theta_{\mathcal{F},\mathcal{G},\psi}](o_{t,T}, a_t) = \mathcal{L}_{\text{act}}(\psi(z_t, e_t), a_t) = -\log \psi(a_t|z_t, e_t)$. When we know the the pretraining data is collected by a fixed deterministic policy, we let $\psi : \mathcal{Z} \times \mathcal{E} \to \mathcal{A}$ be a deterministic function that minimizes the $L1$-loss: $\mathcal{L}_{\text{act_decode}}[\theta_{\mathcal{F},\mathcal{G},\psi}](o_{t,T}, a_t) = \mathcal{L}_{\text{act}}(\psi(z_t, e_t), a_t) = \|\psi(z_t, e_t) - a_t\|_1$.

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

$$\mathcal{L}[\theta_{\mathcal{F},\mathcal{G},Q,P,\psi}] = \mathcal{L}_{\text{dyn}}[\theta_{\mathcal{F},\mathcal{G},Q,P}] + \beta\mathcal{L}_{\text{act_decode}}[\theta_{\mathcal{F},\mathcal{G},\psi}] \tag{2}$$

Throughout the experiment, we 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 1(b) illustrates the mechanics of this stage. After training the action quantizer, we first use the pretrained observation embedding $\mathcal{G}$, action quantizer $\mathcal{F}$ to convert the trajectories from pretrainig dataset $\mathcal{D}$ into sequences of discrete quantized action codes, $\{(e_0^{(i)}, e_1^{(i)}, ..., e_{H_i}^{(i)})\}_{i=1}^{|\mathcal{D}|}$, 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, we view the sequence of action codes in each trajectory as a "sentence" and employ the BPE algorithm to create a vocabulary of skill tokens $\mathcal{V} = \{\xi_1, ..., \xi_K\}$, where tokens represent "sentence"/trajectory segments. To do so, we 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.

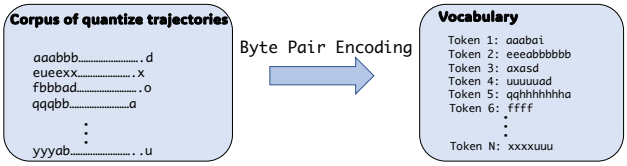


Figure 2. Byte Pair Encoding.

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, we discuss how to use them to learn generalist multitask policies and achieve efficient downstream adaptation.

3.2. Multitask generalist policy learning.

Once we have learned the skill tokens that capture the common motion patterns shared across various tasks, we can leverage these skills to learn a multitask generalist policy. We train a high-level skill-token policy $\pi : \mathcal{Z} \to \Delta(\mathcal{V})$ 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.

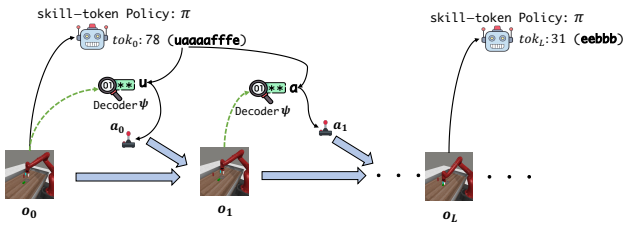


Figure 3. During evaluation time, PRISE rollout its policy by querying the skill-token policy π for the skill token and then using pretrained decoder ψ to decode actions.

To learn this skill-token policy π from a set of expert demonstration trajectories $\mathcal{D} = \{(o_0^{(i)}, a_0^{(i)}, ...o_{H_i}^{(i)})\}_{i=1}^{|\mathcal{D}|}$, we first run the action quantizer $\mathcal{F}$ and observation encoder $\mathcal{G}$ to get the quantized action codes of the trajectories $\{(c_0^{(i)}, c_1^{(i)}, ...o_{H_i}^{(i)})\}_i$, as well as latent observation embeddings $\{(z_0^{(i)}, z_1^{(i)}, ...z_{H_i}^{(i)})\}_i$. Then for each timestep t in episode i, we 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 $(c_t^{(i)}, c_{t+1}^{(i)}, ..., c_{H_i}^{(i)})$.

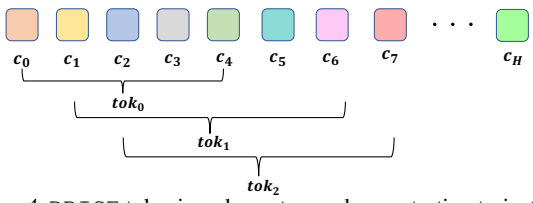


Figure 4. PRISE tokenizes downstream demonstration trajectories by greedily searching for the longest token for each time step.

We then train π by minimizing the cross-entropy loss:

$$\mathcal{L}_{\text{CE}}(\pi(\text{stopgrad}(z_t)), \xi_t)[\theta_\pi] \tag{3}$$

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

3.3. Downstream few-shot adaptation to unseen tasks

In addition to learning a generalist multitask policy π, we 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 we need to finetune the decoder ψ for downstream unseen tasks, in addition to adapting π itself.

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

$$\mathcal{L}_{\text{FT_DECODER}}[\theta_{\pi,\psi}] = \mathbb{E}_{\xi_t \sim \pi(\text{stopgrad}(z_t))}\Big[\mathcal{L}[\theta_\psi](\xi_t)\Big] \tag{4}$$

where

$$\mathcal{L}[\theta_\psi](\xi_t) = \sum_{i=0}^{\hat{K}} \mathcal{L}_{\text{action}}(\psi(\text{stopgrad}(z_{t+i}), \xi_t[i]), a_{t+i}) \tag{5}$$

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 3.4. $\xi_t[i]$ represents the i-th code of the skill token ξ_t.

We optimize skill-token policy π w.r.t. the cross-entropy loss $\mathcal{L}_{\text{CE}}$ in (3). Thus, the overall objective is

$$\mathcal{L}[\theta_{\pi,\psi}] = \mathcal{L}_{\text{CE}}(\pi(\text{stopgrad}(z_t)), \xi_t) + \mathcal{L}_{\text{FT_DECODER}}[\theta_{\pi,\psi}] \tag{6}$$

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 (6) says that the skill-token policy π should not only match the target token but also predict the tokens that have small decoding errors.

3.4. Discussion

PRISE combines state-action abstraction (by optimizing the quantization loss in Equation (2)) 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 (Rajaraman et al., 2020). 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. We 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. 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, we introduce a hyperparameter K (see Section 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. Related Work

Please refer to Appendix A for an extensive discussion of related work. Here we provide its summary

Temporal Action Abstractions in Sequential Decision-Making: Temporal action abstraction and options or skills by compression is not newly introduced in this paper. In fact, the concept of temporal action abstraction is rooted in the literature of both fully and partially observable MDPs (Puterman, 1994; Thrun & Schwartz, 1994; Sutton & Barto, 2018). The complexity of policy learning is proportional to the problem horizon (Ross et al., 2011; Jiang et al., 2015; Rajaraman et al., 2020; Xie et al., 2021), leading to hierarchical approaches (Parr & Russell, 1998; Barto & Mahadevan, 2003; Le et al., 2018; Nachum et al., 2018; Kipf et al., 2019; Kujanpää et al., 2023; Jiang et al., 2022) and the use of options or primitives (Sutton et al., 1999; Ajay et al., 2021; Cui et al., 2023).

Temporal Abstraction Discovery for Decision-Making: Various methods like CompILE (Kipf et al., 2019), RPL (Gupta et al., 2019), OPAL (Ajay et al., 2021), TAP (Jiang et al., 2023), and ACT (Zhao et al., 2023a) learn temporally extended action primitives in two stages. DDCO (Krishnan et al., 2017), in contrast, learns both primitives and high-level policy simultaneously.

PRISE differs in its use of BPE for inducing temporal action abstractions, avoiding challenges faced by CompILE (Kipf et al., 2019), RPL, and OPAL. ACT (Zhao et al., 2023b) is the closest to PRISE, especially in using BC, not RL, for downstream learning and handling pixel observations rather than low-level states.

Policy Quantization Methods: PRISE relates to SAQ (Luo et al., 2023) and employs VQ-VAE (van den Oord et al., 2017), with Conditional VAE (Kingma & Welling, 2014) being another common choice.

Temporal Abstractions vs. Skills in Robot Learning: Distinct from skill acquisition approaches such as LOVE (Jiang et al., 2022), BeT (Shafiullah et al., 2022), C-BeT (Cui et al., 2023), DMP (Pastor et al., 2009), Play-LMP (Lynch et al., 2019), and MCIL (Lynch & Sermanet, 2021), PRISE and methods like TAP (Jiang et al., 2023) focus on learning temporally extended action primitives.

Pretraining Data Assumptions: PRISE aligns more with C-BeT (Cui et al., 2023), Play-LMP (Lynch et al., 2019), RPL (Gupta et al., 2019), MCIL (Lynch & Sermanet, 2021), and TACO-RL (Rosete-Beas et al., 2022), requiring meaningful behavioral patterns in pretraining data.

Tokenization in Language Models: BPE (Gage, 1994), Unigram (Kudo, 2018), and WordPiece (Devlin et al., 2018) are crucial in training language models. PRISE extends the next-token-prediction paradigm to continuous control,

dealing with challenges like high-dimensional image observations and continuous action spaces.

5. Experiments

In this section, we empirically evaluate the effectiveness of the skill tokens of PRISE. We run PRISE to pretrain these tokens using large-scale, multitask offline datasets. Then we evaluate them on two offline IL scenarios: learning a multitask generalist policy and few-shot adaptation to unseen tasks. We 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 (Zhao et al., 2023b).

5.1. Pretraining datasets and architecture

We evaluate PRISE on two multitask robotic manipulation benchmarks: Metaworld (Yu et al., 2019) and LIBERO (Liu et al., 2023). Below we describe the setups of these datasets and the architectures used in the experiments.

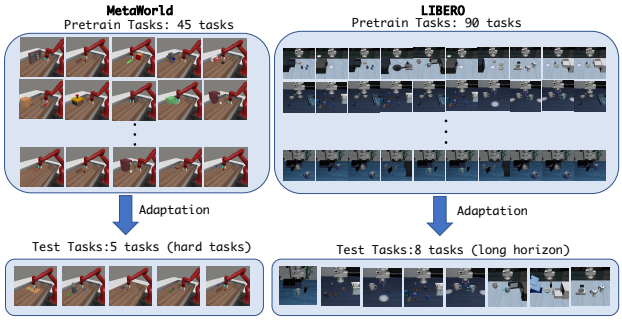


Figure 5. Few-shot IL on unseen tasks: pretrain and test tasks split for MetaWorld and LIBERO.

For LIBERO, we pretrain on the 90 short-horizon tasks (LIBERO-90) with offline dataset provided by the original paper. We 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. We use the exact architecture of ResNet-T in (Liu et al., 2023) for the observation embedding.

For MetaWorld, we 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%. We refer the readers to Figure 15 in Appendix D for the results. For the pretrain-test split, we 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. We generate 100 expert trajectories for each pretraining task using the scripted policy provided in MetaWorld. Same as in as in (Yarats et al., 2022; Xu et al., 2024), we 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.

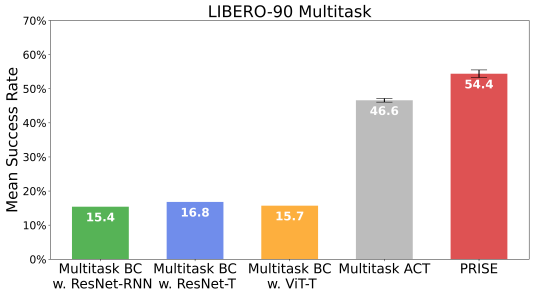


Figure 6. Multitask policy learning performance on **LIBERO-90.** Error bar represents the standard deviation across 4 random seeds. Results of the first three methods are taken directly from (Liu et al., 2023)

For BPE on both domains, we set the codebook size C to be 10 and vocabulary size to be 200, and we provide detailed ablation study to analyze their impacts later in this section. For more implementation details, we refer the readers to Appendix C.

5.2. Multitask generalist policy learning

First we evaluate whether the pretrained skill tokens of PRISE could enable efficient knowledge sharing across tasks and improve multitask IL performance. Here we focus on **LIBERO-90**, a challenging multi-task benchmark where existing algorithms and architectures have not demonstrated satisfactory performance. For each algorithm in the following comparison, we 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 (Liu et al., 2023). These architectures utilize either ResNet-18 (He et al., 2016) or ViT (Dosovitskiy et al., 2021) 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 we add the vector quantization as well as other modules unique to PRISE. Additionally, we compare with ACT (Zhao et al., 2023a), 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 we set at 8 after evaluating the best option from five choices: $\{3, 5, 8, 10, 15\}$.

Experimental Results. Figure 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. For the details (e.g., per-task success rateas well as the evaluation protocol), we refer readers to Appendix D.

5.3. Few-shot adaptation to unseen tasks

In addition to learning a generalist multitask policy, we show that the learned skill token of PRISE can also make learning a new task more efficient. Here, we evaluate the 5-shot IL performance for unseen tasks in both MetaWorld and LIBERO.

Baselines. We 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. We 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 $V = \{1, ..., C\}$ 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 (Ajay et al., 2021) and TACO-RL (Rosete-Beas et al., 2022). We refer readers to Section 4 and Section A for a detailed discussion. (4) ACT (Zhao et al., 2023a) from scratch. (5) ACT (Zhao et al., 2023a) with pretrained CNN encoder from PRISE. We freeze the CNN encoder and finetune other parts of the model.

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

Experimental Results (MetaWorld). In Figure 7, we 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, we observe that with the pretrained visual encoder from PRISE, BC and ACT consistently yields improved results, indicating that the

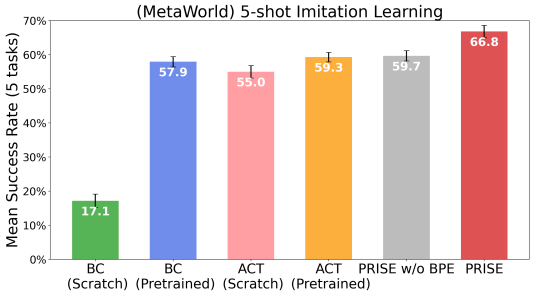


Figure 7. (Metaworld): 5-shot IL performance averaged across 5 unseen tasks. Error bar represents the standard deviation across 4 random seeds.

PRISE learning objective is also beneficial for pretraining visual representations from multitask offline data.

Experimental Results (LIBERO). We present the average 5-shot IL performance across 8 tasks of LIBERO in Figure 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 (Liu et al., 2023), 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, we 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 mechanisms by PRISE.

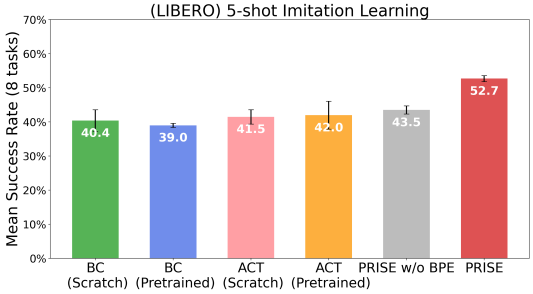


Figure 8. (LIBERO): 5-shot IL performance averaged across 8 unseen tasks. Error bar stands for the standard deviation across 4 random seeds.

5.4. 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 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 3.4). In the previous experiments, we set $|\mathcal{V}|$ to be 200. Here, we evaluate five hyperparame-

ter 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 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.

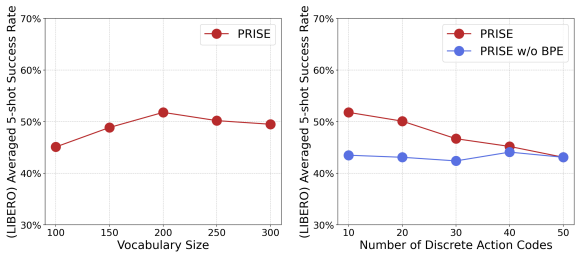


Figure 9. (**Left**) Mean success rate of PRISE across 8 unseen LIBERO tasks with different vocabulary size. (**Right**) The mean success rate of PRISE across 8 unseen LIBERO tasks with a different numbers of discrete action codes pre-trained.

Number of Action Codes. Additionally, we 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. We 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 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, we also compare the histogram of skill token lengths for $C = 10$ versus $C = 40$ in Figure 10 . The comparison reveals a clear reduction in token length with an increase in the number of codes.

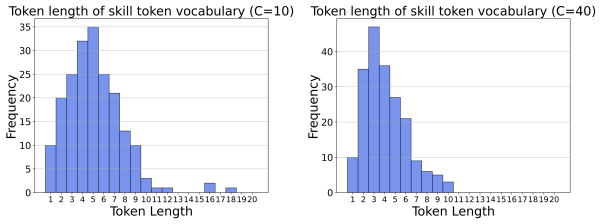


Figure 10. (**Left**) Histogram of token length when $C = 10$. (**Right**) Histogram of token length when $C = 40$.

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 we 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 11, we empirically compare the few-shot IL performance of PRISE in LIBERO with and without the forward dynamics objective. Indeed, we see a performance degradation in both domains when the forward dynamics objective is removed.

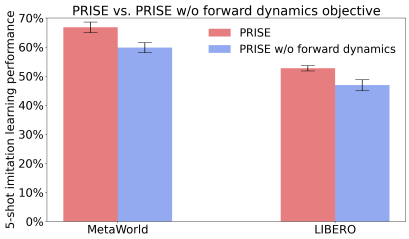


Figure 11. Performance of PRISE with and without the forward dynamics objective on Metaworld and LIBERO.

Additionally, we conducted an extra experiment on LIBERO to illustrate the action collapsing problem further. Here, we sample a batch of 5000 data points $\{(z_i, a_i)\}_{i=1}^{5000}$. For each state z_i, we 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 $\mathcal{M}$, we 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, we 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, we expect to see small non-diagonal entries in $\mathcal{M}$. We compare the value $\zeta = \dfrac{\text{sum}(\mathcal{M})}{\binom{C}{2}}$ for PRISE with and without the forward dynamics objective. As illustrated in the table,

	PRISE	PRISE W/O FORWARD DYNAMICS
ζ	72.42	14.53

PRISE without the forward 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, we conduct a set of additional experiments on LIBERO. Instead of using the entire LIBERO-90 dataset, where each task includes 50 human demonstration trajectories, we now sample N trajectories per task, with $N \in \{10, 20, 30, 40, 50\}$. We then evaluate the downstream performance of the PRISE skill token. The result is shown in Figure 12.

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

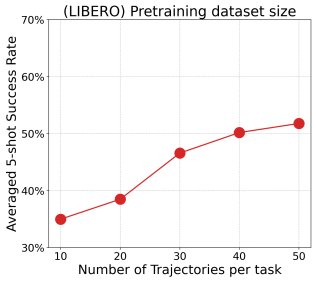


Figure 12. Downstream performance of PRISE under different pretraining dataset sizes

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.

6. Conclusion and Future Work

In this work, we propose a new temporal action method for continuous control scenarios, PRISE, that leverages powerful NLP methodology during pretraining for sequential decision making tasks. By first pretraining a vector quantization module to discretize the action space into codes and then applying Byte Pair Encoding to learn temporally extended skill tokens, PRISE captures diverse motion patterns across pretraining tasks, enabling efficient multitask policy learning and few-shot adaptation to unseen problems. One exciting future direction is to further scale up this approach to large real-robot datasets with diverse embodiments, such as Open X-Embodiment (Padalkar et al., 2024). Additionally, instead of finetuning the model to different downstream tasks tabula rasa, we could leverage the pretrained tokens to instruction-finetune an existing large language model, so that we could capitalize on its generalization power across different tasks and scenarios.

Impact Statement

This paper presents a pretraining algorithm to advance policy learning for sequential decision making problems. The proposed method is demonstrated in experiments of simulation robotic manipulation. As it is, this work does not have significant societal consequences. However, when the method here is scaled up to train large, foundation models, it would build AI/robotics systems that can more efficiently adapt and generalize. Such an improvement may pose potential societal risks (such as taking over human labour).

Acknowledgements

Ruijie Zheng and Furong Huang are supported by National Science Foundation NSF-IIS-2147276 FAI, DOD-ONR-Office of Naval Research under award number N00014-22-1-2335, DOD-AFOSR-Air Force Office of Scientific Research under award number FA9550-23-1-0048, DOD-DARPA-Defense Advanced Research Projects Agency Guaranteeing AI Robustness against Deception (GARD) HR00112020007, Adobe, Capital One and JP Morgan faculty fellowships.

References

Ajay, A., Kumar, A., Agrawal, P., Levine, S., and Nachum, O. OPAL: Offline primitive discovery for accelerating offline reinforcement learning. In *ICLR*, 2021. URL https://openreview.net/forum?id=V69LGwJ0lIN. 5, 7, 13

Arora, K., Asri, L. E., Bahuleyan, H., and Cheung, J. C. K. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. In *Findings of ACL*, 2022. 1

Barto, A. and Mahadevan, S. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications*, 13:41–77, 2003. 5, 13

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *NeurIPS*, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf. 2

Chandak, Y., Theocharous, G., Kostas, J., Jordan, S., and Thomas, P. Learning action representations for reinforcement learning. In *ICML*, 2019. 3

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*, 2021. 1

Cui, Z. J., Wang, Y., Shafiullah, N. M. M., and Pinto, L. From play to policy: Conditional behavior generation from uncurated robot data. In *ICLR*, 2023. URL https://openreview.net/forum?id=c7rM7F7jQjN. 5, 14

den Oord, A. V., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding, 2019. 1

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 5, 14

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy. 6

Gage, P. A new algorithm for data compression. *C Users Journal*, 12(2):23—38, 1994. 1, 2, 5, 14

Gupta, A., Kumar, V., Lynch, C., Levine, S., and Hausman, K. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *CoRL*, 2019. URL https://proceedings.mlr.press/v100/gupta20a.html. 5, 13, 14

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models, 2023. 16

Hansen, N., Wang, X., and Su, H. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022. 16

Hausman, K., Springenberg, J. T., Wang, Z., Heess, N., and Riedmiller, M. Learning an embedding space for transferable robot skills. In *ICLR*, 2018. 13

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016. 6, 16

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. In *NeurIPS*, 2019. 16

Jiang, N., Kulesza, A., Singh, S., and Lewis, R. The dependence of effective planning horizon on model accuracy. In *AAMAS*, 2015. 5, 13

Jiang, Y., Liu, E. Z., Eysenbach, B., Kolter, J. Z., and Finn, C. Learning options via compression. In *NeurIPS*, 2022. URL https://openreview.net/forum?id=D4fuQ1MveDM. 5, 13

Jiang, Z., Zhang, T., Janner, M., Li, Y., Rocktäschel, T., Grefenstette, E., and Tian, Y. Efficient planning in a compact latent action space. In *ICLR*, 2023. URL https://openreview.net/forum?id=cA77NrVEuqn. 5, 13

Kingma, D. and Welling, M. Auto-encoding variational Bayes. In *NIPS*, 2014. 5, 13

Kipf, T., Li, Y., Dai, H., Zambaldi, V., Sanchez-Gonzalez, A., Grefenstette, E., Kohli, P., and Battaglia, P. Compile: Compositional imitation learning and execution. In *ICML*, 2019. 5, 13

Krishnan, S., Fox, R., Stoica, I., and Goldberg, K. DDCO: Discovery of deep continuous options for robot learning from demonstrations. In *CoRL*, 2017. 5, 13

Kudo, T. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*, 2018. 5, 14

Kujanpää, K., Pajarinen, J., and Ilin, A. Hierarchical imitation learning with vector quantized models. In *ICML*, 2023. URL https://proceedings.mlr.press/v202/kujanpaa23a.html. 5, 13

Le, H. M., Jiang, N., Agarwal, A., Dudík, M., Yue, Y., and au2, H. D. I. Hierarchical imitation and reinforcement learning. In *ICML*, 2018. 5, 13

Liu, B., Zhu, Y., Gao, C., Feng, Y., qiang liu, Zhu, Y., and Stone, P. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=xzEtNSuDJk. 6, 7, 16, 18

Luo, J., Dong, P., Wu, J., Kumar, A., Geng, X., and Levine, S. Action-quantized offline reinforcement learning for robotic skill learning. In *CoRL*, 2023. URL https://openreview.net/forum?id=n9lew97SAn. 5, 13

Lynch, C. and Sermanet, P. Language conditioned imitation learning over unstructured data. In *RSS*, 2021. 5, 13, 14

Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., and Sermanet, P. Learning latent plans from play. In *CoRL*, 2019. 5, 13, 14

Mandlekar, A., Xu, D., Wong, J., Nasiriany, S., Wang, C., Kulkarni, R., Fei-Fei, L., Savarese, S., Zhu, Y., and Martín-Martín, R. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021. 3

Nachum, O., Gu, S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. In *NeurIPS*, 2018. 5, 13

Nair, S., Rajeswaran, A., Kumar, V., Finn, C., and Gupta, A. R3m: A universal visual representation for robot manipulation. In *CoRL*, 2022. URL https://openreview.net/forum?id=tGbpgz6yOrI. 1

OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023. 2

Padalkar, A., Pooley, A., Jain, A., Bewley, A., Herzog, A., Irpan, A., Khazatsky, A., Rai, A., Singh, A., Brohan, A., Raffin, A., Wahid, A., Burgess-Limerick, B., Kim, B., Schölkopf, B., Ichter, B., Lu, C., Xu, C., Finn, C., Xu, C., Chi, C., Huang, C., Chan, C., Pan, C., Fu, C., Devin, C., Driess, D., Pathak, D., Shah, D., Büchler, D., Kalashnikov, D., Sadigh, D., Johns, E., Ceola, F., Xia, F., Stulp, F., Zhou, G., Sukhatme, G. S., Salhotra, G., Yan, G., Schiavi, G., Su, H., Fang, H.-S., Shi, H., Amor, H. B., Christensen, H. I., Furuta, H., Walke, H., Fang, H., Mordatch, I., Radosavovic, I., Leal, I., Liang, J., Kim, J., Schneider, J., Hsu, J., Bohg, J., Bingham, J., Wu, J., Wu, J., Luo, J., Gu, J., Tan, J., Oh, J., Malik, J., Tompson, J., Yang, J., Lim, J. J., Silvério, J., Han, J., Rao, K., Pertsch, K., Hausman, K., Go, K., Gopalakrishnan, K., Goldberg, K., Byrne, K., Oslund, K., Kawaharazuka, K., Zhang, K., Majd, K., Rana, K., Srinivasan, K., Chen, L. Y., Pinto, L., Tan, L., Ott, L., Lee, L., Tomizuka, M., Du, M., Ahn, M., Zhang, M., Ding, M., Srirama, M. K., Sharma, M., Kim, M. J., Kanazawa, N., Hansen, N., Heess, N., Joshi, N. J., Suenderhauf, N., Palo, N. D., Shafiullah, N. M. M., Mees, O., Kroemer, O., Sanketi, P. R., Wohlhart, P., Xu, P., Sermanet, P., Sundaresan, P., Vuong, Q., Rafailov, R., Tian, R., Doshi, R., Martín-Martín, R., Mendonca, R., Shah, R., Hoque, R., Julian, R., Bustamante, S., Kirmani, S., Levine, S., Moore, S., Bahl, S., Dass, S., Song, S., Xu, S., Haldar, S., Adebola, S., Guist, S., Nasiriany, S., Schaal, S., Welker, S., Tian, S., Dasari, S., Belkhale, S., Osa, T., Harada, T., Matsushima, T., Xiao, T., Yu, T., Ding, T., Davchev, T., Zhao, T. Z., Armstrong, T., Darrell, T., Jain, V., Vanhoucke, V., Zhan, W., Zhou, W., Burgard, W., Chen, X., Wang, X., Zhu, X., Li, X., Lu, Y., Chebotar, Y., Zhou, Y., Zhu, Y., Xu, Y., Wang, Y., Bisk, Y., Cho, Y., Lee, Y., Cui, Y., hua Wu, Y., Tang, Y., Zhu, Y., Li, Y., Iwasawa, Y., Matsuo, Y., Xu, Z., and Cui, Z. J. Open X-Embodiment: Robotic learning datasets and RT-X models. In *ICRA*, 2024. 9

Parr, R. and Russell, S. Reinforcement learning with hierarchies of machines. In *NIPS*, pp. 1043–1049, 1998. 5, 13

Pastor, P., Hoffmann, H., Asfour, T., and Schaal, S. Learning and generalization of motor skills by learning from demonstration. In *ICRA*, 2009. 5, 13

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. C. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018. 16

Puterman, M. L. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley and Sons, 1994. 5, 13

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners, 2019. 1, 2

Rajaraman, N., Yang, L., Jiao, J., and Ramchandran, K. Toward the fundamental limits of imitation learning. In *NeurIPS*, 2020. 5, 13

Rosete-Beas, E., Mees, O., Kalweit, G., Boedecker, J., and Burgard, W. Latent plans for task agnostic offline reinforcement learning. 2022. 5, 7, 13, 14

Ross, S., Gordon, G. J., and Bagnell, J. A. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011. 5, 13

Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A., and Bachman, P. Data-efficient reinforcement learning with self-predictive representations. In *ICLR*, 2021. URL https://openreview.net/forum?id=XpSAvlvnMa. 3, 14, 16

Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. In *ACL*, 2016. 2

Shafiullah, N. M., Cui, Z., Altanzaya, A. A., and Pinto, L. Behavior transformers: Cloning k modes with one stone. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *NeurIPS*, 2022. 5

Sun, Y., Ma, S., Madaan, R., Bonatti, R., Huang, F., and Kapoor, A. SMART: Self-supervised multi-task pretraining with control transformers. In *The Eleventh ICLR*, 2023. URL https://openreview.net/forum?id=9piH3Hg8QEf. 14

Sutton, R. and Barto, A. *Reinforcement learning: An introduction*. The MIT Press, 2nd edition, 2018. 5, 13

Sutton, R., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2): 181–211, 1999. 5, 13

Thrun, S. and Schwartz, A. Finding structure in reinforcement learning. In Tesauro, G., Touretzky, D., and Leen, T. (eds.), *Advances in Neural Information Processing Systems*, 1994. URL https://proceedings.neurips.cc/paper_files/paper/1994/file/7ce3284b743aefde80ffd9aec500e085-Paper.pdf. 5

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. Neural discrete representation learning. In *NeurIPS*, 2017. 5, 13

Van den Oord, A., Vinyals, O., and Kavukcuoglu, K. Neural discrete representation learning. In *NeurIPS*, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/7a98af17e63a0ac09ce2e96d03992fbc-Paper.pdf. 2

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017. 2

Wang, X., Zheng, R., Sun, Y., Jia, R., Wongkamjan, W., Xu, H., and Huang, F. COPlanner: Plan to roll out conservatively but to explore optimistically for model-based RL. In *The Twelfth ICLR*, 2024. URL https://openreview.net/forum?id=jnFcKjtUPN. 16

Wei, Y., Sun, Y., Zheng, R., Vemprala, S., Bonatti, R., Chen, S., Madaan, R., Ba, Z., Kapoor, A., and Ma, S. Is imitation all you need? generalized decision-making with dual-phase training. In *ICCV*, 2023. 14

Xie, T., Cheng, C.-A., Jiang, N., Mineiro, P., and Agarwal, A. Bellman-consistent pessimism for offline reinforcement learning. In *NeurIPS*, 2021. 5, 13

Xu, G., Zheng, R., Liang, Y., Wang, X., Yuan, Z., Ji, T., Luo, Y., Liu, X., Yuan, J., Hua, P., Li, S., Ze, Y., III, H. D., Huang, F., and Xu, H. Drm: Mastering visual reinforcement learning through dormant ratio minimization. In *The Twelfth ICLR*, 2024. URL https://openreview.net/forum?id=MSe8YFbhUE. 6

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *ICLR*, 2022. URL https://openreview.net/forum?id=_SJ-_yyes8. 6, 16

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *CoRL*, 2019. URL https://arxiv.org/abs/1910.10897. 6

Zhang, A., McAllister, R. T., Calandra, R., Gal, Y., and Levine, S. Learning invariant representations for reinforcement learning without reconstruction. In *ICLR*, 2021. URL https://openreview.net/forum?id=-2FCwDKRREu. 2

Zhao, T. Z., Kumar, V., Levine, S., and Finn, C. Learning fine-grained bimanual manipulation with low-cost hardware. In Bekris, K. E., Hauser, K., Herbert, S. L., and Yu, J. (eds.), *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023a. doi: 10.15607/RSS.2023.XIX.016. URL https://doi.org/10.15607/RSS.2023.XIX.016. 5, 6, 7, 13

Zhao, T. Z., Kumar, V., Levine, S., and Finn, C. Learning fine-grained bimanual manipulation with low-cost hardware. In *RSS*, 2023b. 5, 6, 13

Zheng, R., Wang, X., Sun, Y., Ma, S., Zhao, J., Xu, H., III, H. D., and Huang, F. TACO: Temporal latent action-driven contrastive loss for visual reinforcement learning. In *NeurIPS*, 2023a. URL https://openreview.net/forum?id=ezCsMOy1w9. 1, 14

Zheng, R., Wang, X., Xu, H., and Huang, F. Is model ensemble necessary? model-based RL via a single model with lipschitz regularized value function. In *The Eleventh ICLR*, 2023b. URL https://openreview.net/forum?id=hNyJBk3CwR. 16

Zheng, R., Liang, Y., Wang, X., Ma, S., III, H. D., Xu, H., Langford, J., Palanisamy, P., Basu, K. S., and Huang, F. Premier-TACO is a few-shot policy learner: Pretraining multitask representation via temporal action-driven contrastive loss. In *arXiv*, 2024. 14

A. Detailed 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 (Puterman, 1994; Sutton & Barto, 2018), 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 (Ross et al., 2011; Jiang et al., 2015; Rajaraman et al., 2020; Xie et al., 2021). This motivates hierarchical approaches (Parr & Russell, 1998; Barto & Mahadevan, 2003; Le et al., 2018; Nachum et al., 2018; Kipf et al., 2019; Kujanpää et al., 2023; Jiang et al., 2022), 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* (Sutton et al., 1999) or *primitives* (Ajay et al., 2021) 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 (Kipf et al., 2019), RPL (Gupta et al., 2019), OPAL (Ajay et al., 2021), TAP (Jiang et al., 2023), and ACT (Zhao et al., 2023a). 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 (Krishnan et al., 2017), 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 we are 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 (Kipf et al., 2019). 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.

We view ACT (Zhao et al., 2023b) 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 (Luo et al., 2023), 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 (van den Oord et al., 2017). Conditional VAE (Kingma & Welling, 2014) is another common choice for this role (see, e.g., Lynch et al. (2019) and Kipf et al. (2019)).

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 (Rosete-Beas et al., 2022) being a notable exception. Namely, approaches such as DMP (Pastor et al., 2009), (Hausman et al., 2018), Play-LMP (Lynch et al., 2019), and MCIL (Lynch & Sermanet, 2021) 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 (Jiang et al., 2023) 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 C-BeT (Cui et al., 2023), Play-LMP (Lynch et al., 2019), RPL (Gupta et al., 2019), MCIL (Lynch & Sermanet, 2021), and TACO-RL (Rosete-Beas et al., 2022) rather than to IL or offline RL. Namely, PRISE needs this data to contain meaningful behavioral patters, 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 (Schwarzer et al., 2021; Sun et al., 2023; Wei et al., 2023; Zheng et al., 2023a; 2024), 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 (Gage, 1994), Unigram (Kudo, 2018), and WordPiece (Devlin et al., 2018) 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 we 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.

B. Additional Experimental Results

In Figure 7, we present the results of mean success rate across 5 tasks in MetaWorld. Here we present the detailed results for each of the downstream unseen task.

MetaWorld Unseen Tasks	Algorithms					
	BC (Scratch)	BC (Pretrained)	ACT (Scratch)	ACT (Pretrained)	PRISE w/o BPE	**PRISE**
Hand Insert	18.8 ± 12.5	46.3 ± 7.6	45.6 ± 6.1	56.3 ± 2.8	54.2 ± 3.9	$\mathbf{60.0 \pm 5.0}$
Box Close	19.4 ± 8.7	53.9 ± 4.2	55.6 ± 6.7	58.1 ± 6.8	54.8 ± 5.1	$\mathbf{60.8 \pm 6.6}$
Disassemble	8.1 ± 17.3	65.6 ± 3.6	65.2 ± 2.9	51.8 ± 8.5	69.6 ± 3.2	$\mathbf{74.1 \pm 7.3}$
Stick Pull	20.6 ± 5.8	56.3 ± 5.6	51.8 ± 4.3	65.0 ± 4.5	59.3 ± 6.2	$\mathbf{67.5 \pm 5.6}$
Pick Place Wall	19.4 ± 6.4	67.5 ± 3.7	56.9 ± 5.9	65.2 ± 6.5	62.5 ± 4.8	$\mathbf{71.7 \pm 5.7}$
Mean	17.3 ± 4.2	57.9 ± 1.4	55.0 ± 1.8	59.3 ± 2.0	60.1 ± 1.5	$\mathbf{66.8 \pm 1.8}$

Table 1. 5-shot imitation learning performance on five unseen tasks in MetaWorld. Results are averaged across 4 random seeds.

For LIBERO, in Table 2, we first provide the task instruction for each of the downstream 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. Language instructions for 8 LIBERO downstream tasks.

The detailed results for each of the LIBERO downstream unseen task are presented in Table 3.

LIBERO Unseen Tasks	Algorithms					
	BC (Scratch)	BC (Pretrained)	ACT (Scratch)	ACT (Pretrained)	PRISE w/o BPE	**PRISE**
Task 0	16.7 ± 6.2	6.7 ± 2.3	21.7 ± 8.4	20.0 ± 6.0	16.7 ± 6.2	$\mathbf{26.7 \pm 6.4}$
Task 1	25.0 ± 4.7	48.3 ± 10.3	26.7 ± 10.5	33.3 ± 13.1	25.0 ± 4.7	$\mathbf{48.3 \pm 9.4}$
Task 2	68.3 ± 7.4	60.0 ± 4.1	61.7 ± 9.2	67.7 ± 6.2	68.3 ± 6.2	$\mathbf{70.0 \pm 0.0}$
Task 3	66.7 ± 7.9	66.7 ± 8.4	73.3 ± 9.3	70.3 ± 6.2	66.7 ± 5.9	$\mathbf{78.3 \pm 8.8}$
Task 4	25.0 ± 10.8	26.7 ± 3.1	45.0 ± 4.3	35.0 ± 4.1	$\mathbf{45.0 \pm 10.8}$	28.3 ± 6.2
Task 5	70.0 ± 0.0	46.7 ± 13.1	58.3 ± 4.3	68.3 ± 6.5	75.0 ± 4.2	$\mathbf{90.0 \pm 4.1}$
Task 6	23.3 ± 4.3	21.7 ± 2.4	$\mathbf{35.0 \pm 4.3}$	15.0 ± 0.0	23.3 ± 6.2	25.0 ± 4.1
Task 7	28.3 ± 4.3	35.0 ± 7.1	10.0 ± 0.0	26.7 ± 7.0	28.3 ± 10.2	$\mathbf{45.0 \pm 8.1}$
Mean	40.4 ± 1.2	39.0 ± 4.1	41.5 ± 2.1	42.0 ± 3.2	43.5 ± 0.6	$\mathbf{52.7 \pm 0.9}$

Table 3. 5-shot imitation learning performance across 8 unseen tasks in LIBERO. Resullts are averaged across 4 random seeds

C. Implementation Details

```
# next_observation: A sequence of K=3 future pixel observations o_{t+1},..., o_{t+K}
# past_observation: A sequence of H historical pixel observations o_{t-H+1},..., o_{t}
# action:           A sequence of $K$ action a_t,..., a_{t+K-1}
# obs_encoder:      Observation Encoder (CNN+Transformer)
# quantizer:        Discrete action code quantizer
# dynamics:         Latent Dynamics Model
# project:          Projection layer
# predictor:        Latent state predictor
# action_decoder:   Quantized action code decoder
# beta:             Weight of decoder loss (1.0 in MetaWorld and 0.01
        in LIBERO for numerical stability)

dynamic_losses, quantization_losses, decoder_losses = 0, 0, 0
z = state_encoder(past_observation)
z_hat = z
for k in range(K):
    z = state_encoder(past_observation)
    past_observation.push(next_observation[k])
    u_quantized, quantization_loss = quantizer(action[k], z.detach())
    quantization_losses += quantization_loss
    decode_action       =  decoder(z, u_quantized)
    decoder_losses      += action_loss(decode_action, action[k])
    z_hat  = dynamics(z_hat, u_quantized)
    z_next = state_encoder(observation[k+1])
    y_hat  = predictor(project(z_hat))
    y_next = project(z_next).detach()
    dynamic_losses += -cosine_similarity(y_hat, y_next)
(dynamic_losses + quantization_losses + beta*decoder_losses).backward()
```

Figure 13. Pseudocode for the pretraining stage I of PRISE

Note: Here our decision to adapt a BYOL-like objective for the forward dynamics losses, rather than directly modeling the forward transition dynamics as in model-based learning approaches (Janner et al., 2019; Zheng et al., 2023b; Hansen et al., 2022; Hafner et al., 2023; Wang et al., 2024), is mainly motivated by the effectiveness of the BYOL (Schwarzer et al., 2021) objective in preventing representation collapse during latent forward dynamics training, as we are jointly learning both the action quantization and state representation here in the pretraining stage I.

Metaworld: We 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. We use the same shallow CNN as in (Yarats et al., 2022) 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 14. Next, we apply a transformer decoder module with 4 layers and number of heads equal to 8 to extract the observation embedding. We set the context length to be 10. The action decoder ψ is a three-layer MLP with hidden size being 1024.

LIBERO: For LIBERO, we 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 state, and a task instruction in natural language. We use the exact same architecture as ResNet-T in (Liu et al., 2023). In particular, we use two ResNet18 encoders (He et al., 2016) to process the image observations, utilizing FiLM (Perez et al., 2018) 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, we take 30,000 gradient steps and conduct evaluations for every 3000 steps. For both MetaWorld and LIBERO, we execute 40 episodes and calculate the success rate of the trained policy. We report the highest success rates across 10 evaluated checkpoints.

```python
class Encoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.repr_dim = 32 * 35 * 35
        self.convnet = nn.Sequential(nn.Conv2d(84, 32, 3, stride=2),
                        nn.ReLU(), nn.Conv2d(32, 32, 3, stride=1),
                        nn.ReLU(), nn.Conv2d(32, 32, 3, stride=1),
                        nn.ReLU(), nn.Conv2d(32, 32, 3, stride=1),
                        nn.ReLU(),
                        nn.Linear(self.repr_dim, feature_dim),
                        nn.LayerNorm(feature_dim), nn.Tanh())
        self.trunk = nn.Sequential(nn.Linear(self.repr_dim, feature_dim),
                        nn.LayerNorm(feature_dim), nn.Tanh())

    def forward(self, obs):
        obs = obs / 255.0 - 0.5
        h = self.convnet(obs).view(h.shape[0], -1)
        return self.trunk(h)
```

Figure 14. Architecture of the Shallow CNN encoder used in MetaWorld.

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

D. Additional Results on Multitask Learning

Below we present the per-task success rate for LIBERO-90. The results of BC (ResNet-RNN), BC (ResNet-T), BC (ViT-T) are taken from (Liu et al., 2023). For the evaluation of ACT and PRISE, we follow the same evaluation protocol. We take the last training checkpoint of them respectively and evaluate the success rate on each task with 20 rollouts.

LIBERO-90	Multitask Algorithms				
Task ID	BC (ResNet-RNN)	BC (ResNet-T)	BC (ViT-T)	ACT	**PRISE**
0	0.45	0.45	0.6	1.0	0.8
1	0.1	0.1	0.0	0.6	0.35
2	0.0	0.25	0.05	0.95	0.7
3	0.15	0.0	0.0	0.4	0.5
4	0.2	0.0	0.0	0.3	0.45
5	0.25	0.0	0.0	0.7	0.65
6	0.05	0.0	0.15	0.4	0.5
7	0.3	0.45	0.25	1.0	0.95
8	0.05	0.0	0.0	1.0	0.6
9	0.35	0.3	0.0	0.95	0.35
10	0.2	0.7	0.15	1.0	0.95
11	0.45	0.4	0.65	1.0	0.95
12	0.2	0.05	0.35	0.35	0.2
13	0.1	0.35	0.0	0.25	0.4
14	0.35	0.1	0.0	0.75	0.35
15	0.5	0.1	0.6	0.95	0.75
16	0.25	0.05	0.25	0.75	0.4
17	0.05	0.05	0.0	0.5	0.15
18	0.4	0.3	0.35	0.25	0.3
19	0.15	0.0	0.4	1.0	0.65
20	0.6	0.7	1.0	1.0	1.0
21	0.15	0.0	0.35	0.7	0.3
22	0.75	0.4	0.55	0.75	0.85
23	0.2	0.0	0.05	0.45	0.05
24	0.25	0.3	0.25	0.1	0.95
25	0.8	0.6	0.85	0.1	0.9
26	0.0	0.0	0.35	0.6	0.55
27	0.05	0.0	0.2	0.35	0.05
28	0.05	0.8	0.9	1.0	1.0
29	0.2	0.1	0.15	0.85	1.0
30	0.05	0.05	0.1	0.4	0.5
31	0.5	0.25	0.2	1.0	0.85
32	0.0	0.0	0.05	0.3	0.2
33	0.1	0.1	0.2	0.5	0.3
34	0.2	0.05	0.1	0.5	0.8
35	0.7	0.1	0.45	1.0	0.75
36	0.05	0.0	0.05	0.05	0.25
37	0.05	0.05	0.15	0.0	0.3
38	0.15	0.0	0.1	0.9	0.2
39	0.2	0.25	0.75	0.4	0.85
40	0.45	0.15	0.05	0.9	0.5
41	0.25	0.4	0.3	0.85	0.55
42	0.15	0.45	0.15	0.7	0.8
43	0.1	0.1	0.15	0.85	0.4
44	0.5	0.4	0.95	0.75	0.85

LIBERO-90	Multitask Algorithms				
Task ID	BC (ResNet-RNN)	BC (ResNet-T)	BC (ViT-T)	ACT	**PRISE**
45	0.0	0.0	0.0	0.8	0.55
46	0.0	0.0	0.0	0.0	0.35
47	0.0	0.0	0.0	0.0	0.25
48	0.0	0.0	0.0	0.0	0.65
49	0.0	0.0	0.0	0.0	0.65
50	0.0	0.0	0.0	0.35	0.4
51	0.0	0.0	0.0	0.1	0.1
52	0.0	0.0	0.0	0.05	0.3
53	0.0	0.0	0.0	0.05	0.6
54	0.0	0.05	0.0	0.15	0.5
55	0.0	0.05	0.0	0.0	0.35
56	0.0	0.3	0.0	0.0	0.8
57	0.0	0.25	0.0	0.0	0.5
58	0.0	0.0	0.0	0.5	0.2
59	0.0	0.25	0.0	0.0	0.65
60	0.0	0.4	0.2	0.45	0.8
61	0.0	0.2	0.25	0.05	0.85
62	0.0	0.0	0.1	0.05	0.4
63	0.0	0.0	0.0	0.0	0.4
64	0.0	0.0	0.05	0.25	0.15
65	0.0	0.0	0.05	0.05	0.15
66	0.15	0.0	0.0	0.45	0.3
67	0.1	0.15	0.05	0.55	0.55
68	0.35	0.05	0.05	0.6	0.85
69	0.1	0.65	0.0	0.85	0.9
70	0.05	0.55	0.0	0.6	0.55
71	0.0	0.2	0.0	0.0	0.35
72	0.0	0.2	0.0	0.30	0.6
73	0.0	0.0	0.05	0.35	0.3
74	0.2	0.05	0.0	0.7	0.45
75	0.0	0.1	0.0	0.35	0.25
76	0.0	0.3	0.35	0.1	0.65
77	0.35	0.3	0.0	0.7	0.8
78	0.25	0.1	0.0	0.1	0.45
79	0.25	0.45	0.05	0.95	0.3
80	0.0	0.2	0.0	0.45	0.3
81	0.2	0.0	0.0	0.5	0.35
82	0.4	0.45	0.15	0.55	0.8
83	0.0	0.05	0.05	0.0	0.55
84	0.0	0.2	0.05	0.15	0.75
85	0.0	0.0	0.05	0.1	0.75
86	0.05	0.2	0.05	0.3	0.95
87	0.2	0.1	0.15	1.0	0.65
88	0.0	0.25	0.05	0.85	0.55
89	0.15	0.1	0.05	0.45	0.55

Table 4. Multitask success rate on LIBERO-90.

Additionally, we also provide the results of multitask performance on 45 MetaWorld pretraining tasks. As shown in 15, all three algorithms perform well on MetaWorld, with an average success rate around 80% across 45 tasks. Thus in this paper, we focus on the more challenging LIBERO-90 multitask benchmark.

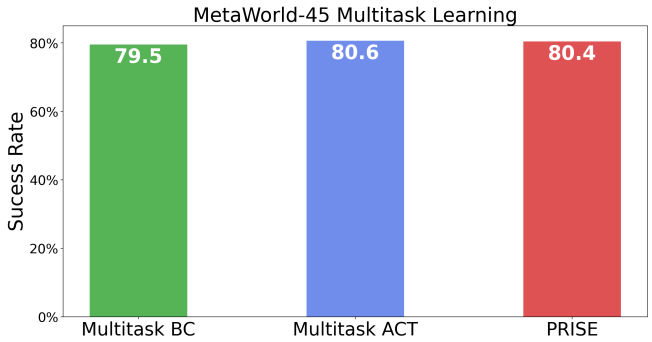


Figure 15. Multitask Learning Results on MetaWorld.