Robotic Offline RL from Internet Videos via Value-Function Pre-Training

Chethan Bhateja[*,1], **Derek Guo**[*,1], **Dibya Ghosh**[*,1], **Anikait Singh**[1,2], **Manan Tomar**[1], **Quan Vuong**[2], **Yevgen Chebotar**[2], **Sergey Levine**[1,2], **and Aviral Kumar**[1,2]

[*]Equal contributions, [1]UC Berkeley, [2]Google DeepMind

Abstract

The full paper is available here. Pre-training on Internet data has proven to be a key ingredient for broad generalization in many modern ML systems. Offline RL methods, which learn from datasets of robot experience, offer one way to leverage prior data into the robotic learning pipeline. However, these methods have a "type mismatch" with video data (such as Ego4D), the largest prior datasets available for robotics, since video offers observation-only experience without the action or reward annotations needed for RL methods. In this paper, we develop a system for leveraging large-scale human video datasets in robotic offline RL, based entirely on learning value functions via temporal-difference learning. We show that value learning on video datasets learns representations that are more conducive to downstream robotic offline RL than other approaches for learning from video data. Our system, called V-PTR, combines the benefits of pre-training on video data with robotic offline RL approaches that train on diverse robot data, resulting in value functions and policies for manipulation tasks that perform better, act robustly, and generalize broadly. On several manipulation tasks on a real WidowX robot, our framework produces policies that greatly improve over prior methods. Our video and additional details can be found on our project website.

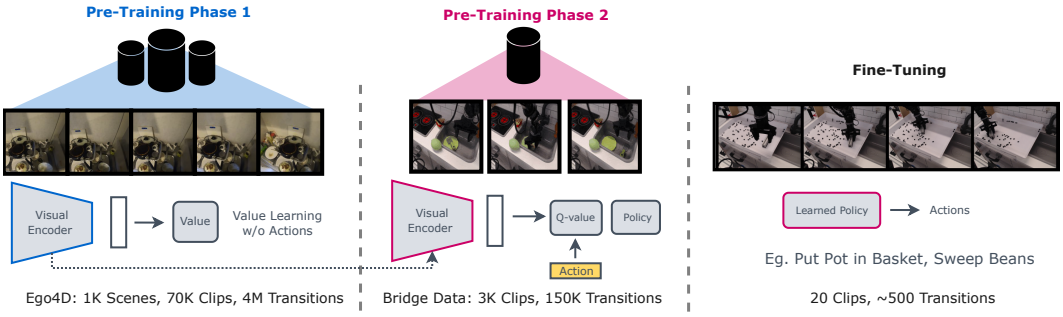

Figure 1: Video pre-training for robots (V-PTR) pre-trains by learning value functions on a large-scale video dataset like Ego4D, and continues refining learned visual features by performing offline RL on multi-task robot data like the Bridge dataset. This pre-training provides a useful initialization for offline RL fine-tuning on downstream robot tasks with improved generalization and robustness compared to other approaches.

1 Introduction

The full paper is available here. Developing methods capable of acquiring robotic skills that generalize widely to new scenarios is an important problem in robotic learning. In other areas of machine learning, broad generalization has been fueled primarily by pre-training on large datasets with a

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

diversity of behavior. It seems compelling that the same formula may be applied to robotic learning, but in practice, even our largest robotic datasets contain data for relatively few tasks, and from a limited number of scenarios. In principle, robotic reinforcement learning (RL) should be able to learn from more general sources of data like human video, which are far more abundant and capture a broader set of skills, situations, and interactions. However, these datasets are difficult to incorporate into RL methods, since internet-scale video data does not come with action or reward annotations present in typical robot data.

Existing works [21; 35; 22] include video data in the robot learning pipeline by performing self-supervised visual representation learning on video data [9], followed by downstream policy learning via behavioral cloning using the learned representations. While such an approach can extract visual features from video, it is limited in its ability to extract a deeper "functional" understanding of the world from video data: in principle, despite differences in embodiment, human videos can still be used to understand intents and affordances that can be executed in the real world, the dynamics, and the eventual outcomes that can be attained by acting. Motivated by the above desiderata for video pre-training, we aim to develop an approach that pre-trains on Internet-scale human video to produce representations for downstream offline RL. Our main contribution is a system, which we call **V**ideo **P**re-**T**raining for **R**obots (V-PTR), that fits value functions to model long-term outcomes achieved when solving tasks on action-free video data.

Concretely, V-PTR pre-trains on human videos by learning an intent-conditioned value function [7] via temporal-difference learning (TD-learning). This approach eschews self-supervised representation learning objectives utilized in prior works [21; 17; 22] in favor of a TD value learning objective, which reflects downstream finetuning objectives . Next, we fine-tune on a multi-task robotic dataset, which is annotated with actions, tasks, and rewards, using value-based offline RL [16]. Downstream, when a target task is specified, V-PTR fine-tunes the multi-task policy on this task. Each phase of our system gradually incorporates the knowledge of "what future outcomes can be achieved" (video pre-training), "what robot actions lead to these outcomes" (robot pre-training), and "how can the desired task be solved" (fine-tuning). Our experiments on several manipulation tasks on a real WidowX robot show that by pre-training on human video data (Ego4D [8]) and multi-task robot data (Bridge data [6]), V-PTR endows downstream offline RL methods with significantly improved zero-shot generalization and robustness to different target objects, distractors, and other variations in the workspace compared to prior methods that learn from videos, significantly outperforming prior methods including VIP [17]. To our knowledge, our work presents the first large-scale demonstration showing that TD-learning alone can effective for pre-training from video for robotic RL.

2 Related Work

A number of prior approaches learn representations from video by applying image-level representation objectives on individual frames like reconstruction [19; 25; 34; 12] or contrastive learning on images [29]. While these objectives are widely used in computer vision, resulting representations do not capture any information about environment dynamics. Other approaches model long-term dynamics from video by predicting the next frame [26], learning value functions [17; 7], running time-contrastive learning [27; 20], or learning language-video alignment [12].

In the context of robot learning, recent works learn representations from internet video datasets like Ego4D and train downstream policies on frozen features using behavioral cloning. In our experiments, we compare to several of these methods, and find that V-PTR attains a higher performance on real-world tasks, especially when evaluated with high initial state variability and in the presence of distractors.

The most closely related work is value-implicit pre-training (VIP) [17], which pre-trains a value function using time-contrastive prediction for downstream reward shaping. Both learn value functions during pre-training, albeit with entirely different algorithms (contrastive learning vs. TD learning), for different policies (dataset policy vs. intent-conditioned policy), exhibiting different generalization properties [2; 5]. Furthermore, the system desiderata differ for VIP and V-PTR: VIP focuses on learning good visual reward functions for weighted behavioral cloning, while we seek good value function initializations for downstream offline RL. In our experiments, we find that when both pre-training approaches are evaluated on the quality of downstream offline RL, V-PTR generalizes better than VIP, even with the VIP reward shaping term as proposed in [17].

Finally, a class of methods attempt to train on video and robot data together in lieu of a video pre-training stage. For instance, [31; 23; 3; 4] train an inverse dynamics model to label video transitions with action pseudo-labels; [30; 32] use inverse RL to imitate the state distribution present in the video. These methods only succeed when a small domain gap exists between video data and robot data, so that observations from video data can plausibly be interpreted as robot data. These methods fail in our setup, as the Ego4D [8] and Bridge [6] datasets differ significantly from each other, including a difference in viewpoint (e.g., egocentric view in video data [8] & shoulder view in robot data [6]). To our knowledge, no method of this type has been successfully applied to this setting.

3 Problem Statement and Background

We aim to leverage Internet-scale video data and multi-task robotic data to boost the robustness and generalization of robotic offline RL. We formulate the robot skill learning problem as the problem of maximizing infinite-horizon discounted reward in a Markov decision process (MDP).

Formal problem statement. We assume access to two pre-training datasets: an Internet-scale video dataset $\mathcal{D}_{\text{video}}$ (e.g., the Ego4D dataset [8]) and a target dataset, $\mathcal{D}_{\text{target}}$ of a limited number of demonstrations for a given target task on the robot. Additionally we are also provided a dataset of multi-task robot behaviors, $\mathcal{D}_{\text{robot}}$, which may not contain any data relevant to the target task. The video dataset $\mathcal{D}_{\text{video}}$ consists of sequences of frames (i.e., observations in the MDP), with no action or rewards. Denoting a frame as $\mathbf{s}_{i,j}$, we define $\mathcal{D}_{\text{video}} := \left\{ \left(\mathbf{s}_{i,0}, \mathbf{s}_{i,1}, \cdots\right) \right\}_{i=1}^{n_{\text{video}}}$. The target dataset, $\mathcal{D}_{\text{target}}$, comprises of a few demonstrations of the target task on the robot $\mathcal{D}_{\text{target}} := \left\{ \left(\mathbf{s}_{i,0}, \mathbf{a}_{i,0}, r_{i,0}, \mathbf{s}_{i,1}, \cdots\right) \right\}_{i=1}^{n_{\text{target}}}$, where the reward, $r_{i,j}$ is annotated to be +1 only on the final three timesteps of the demonstration (following [16]). The multi-task robot dataset $\mathcal{D}_{\text{robot}}$ is organized identically to the target robot dataset, but with an additional task annotation on each trajectory t_i, which is specified either as a one-hot identifier or by natural language. Our goal is train policy π which maximizes the γ-discounted cumulative reward, $\mathbb{E}_{\mathbf{s}_0 \sim \rho_0, \mathbf{a}_{0:\infty}, \mathbf{s}_{1:\infty} \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right]$, starting from a more diverse set of initial states indicated by the distribution ρ_0 than what was observed in the target dataset (e.g., more variation in distractor objects).

Background. Our system utilizes a generalized formulation of goal-conditioned RL and temporal-difference learning for pre-training value functions. In a nutshell, the goal-conditioned RL problem trains the agent to achieve arbitrary goal frames $\mathbf{g}$, where rewards are specified by the sparse signal of $\mathbb{I}\left(\mathbf{s} = \mathbf{g}\right)$ when the frame is identical to the goal frame. Although the reward signal is sparse, goals and rewards can be defined by hindsight relabelling [1]. To learn a policy for the downstream task, we use value-based offline RL methods, which optimize π against a learned Q-function $Q^\pi(\mathbf{s}, \mathbf{a})$. The Q-value function measures the expected long-term reward attained when executing action $\mathbf{a}$ at state $\mathbf{s}$, then following policy π thereafter, and satisfies the Bellman equation $Q^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}', \mathbf{a}'}[Q^\pi(\mathbf{s}', \mathbf{a}')]$.

4 Video Pre-Training for Robotic Offline RL

In order to integrate video data, which contains rich functional and dynamic data useful for downstream learning but is not annotated with actions or rewards, we develop V-PTR, our system that pre-trains general value functions on Internet-scale video data to extract visual representations useful for downstream RL. We first present an overview of our system next, and then discuss each of the components in detail subsequently.

System overview. Our system, V-PTR, pre-trains in three phases: on video data, on multi-task robot data, and on single-task target data. In the first phase, we train an intent-conditioned value function [7] on action-less video data using a value-learning objective to model the outcomes associated with solving the parametric family of goal-achieving tasks. Next, we refine this representation with multi-task robot data with actions and rewards, by training a state-action Q-function on this representation using offline RL. We expect the video and multi-task robot data to capture dynamic features in the environment, with the robot data bridging the domain gap between human video and the robot, and aligning the learned representation with the agent's action space. In our final phase, to adapt the system to a new task, our system fine-tunes the Q-function and the policy on the target dataset.

4.1 Phase 1: Video Pre-Training via TD-Learning

Since the goal of video pre-training is to improve the performance of downstream value-based offline RL, we turn to learning value functions on the video data as a natural pre-training procedure. We

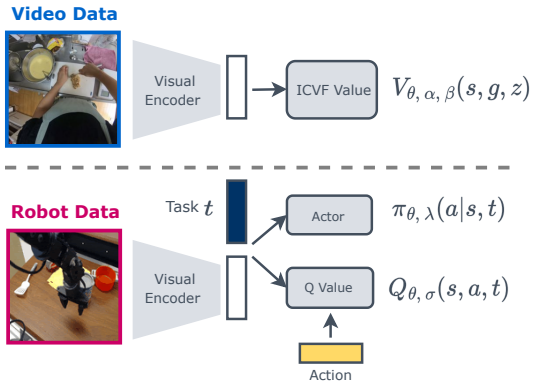

Figure 2: Network architecture. V-PTR first pre-trains image representations by training a general value function from video and then refines this representation via multi-task pre-training on robot data.

choose to pre-train by learning an intent-conditioned value function (ICVF), a recently-proposed general value function that can be efficiently trained on passive data without action labels [7]. An ICVF, annotated $V(\mathbf{s}_{\text{video}}, \mathbf{g}_{\text{video}}, \mathbf{z})$ computes the value obtained towards reaching a goal $\mathbf{g}_{\text{video}}$, assuming the policy *intended* to reach a different intended goal $\mathbf{z}$, and is formally defined as

$$V(\mathbf{s}_{\text{video}}, \mathbf{g}_{\text{video}}, \mathbf{z}) = \mathbb{E}_{a_t \sim \pi_{\mathbf{z}}^*(\cdot|\mathbf{s}_t)}\Big[\sum_t \gamma^t \mathbb{I}\left(\mathbf{s}_{\text{video}} = \mathbf{g}_{\text{video}}\right)\Big].$$

As with a standard value function, the ICVF can be learned by temporal-difference (TD) learning on its corresponding Bellman equation, using a target network to bootstrap the predictions of the learned value function. Specifically, for a given tuple $(\mathbf{s}, \mathbf{g}, \mathbf{z}) \sim \mathcal{D}_{\text{video}}$, this TD objective function is given by:

$$\min \Big[(\alpha - \mathbb{I}(A \le 0)) \cdot \left(\mathbb{I}\left(\mathbf{s}_{\text{video}} = \mathbf{g}_{\text{video}}\right) + \gamma \bar{V}(\mathbf{s}'_{\text{video}}, \mathbf{g}_{\text{video}}, \mathbf{z}) - V(\mathbf{s}_{\text{video}}, \mathbf{g}_{\text{video}}, \mathbf{z})\right)^2\Big],$$

where $A = \mathbb{I}\left(\mathbf{s}_{\text{video}} = \mathbf{g}_{\text{video}}\right) + \gamma \bar{V}(\mathbf{s}'_{\text{video}}, \mathbf{z}, \mathbf{z}) - V(\mathbf{s}_{\text{video}}, \mathbf{z}, \mathbf{z})$ is the implied advantage of $\mathbf{s} \to \mathbf{s}'$ when acting to reach the observation $\mathbf{z}$ and $\bar{V}$ is a delayed copy of the value function (i.e., a target network). We follow the goal-sampling strategy from ICVF: after sampling a start frame $\mathbf{s}$ from the video dataset, we choose the goal $\mathbf{g}$ to be either the next observation, a future observation in the video, or a random observation from a different video. The intent $\mathbf{z}$ is also an observation appearing in the video, and is chosen in a similar fashion as the goal state. Additionally, following Ghosh et al. [7], with some probability we set $\mathbf{z} = \mathbf{g}$.

We follow Ghosh et al. [7] and parameterize our estimated value function as

$$V(\mathbf{s}_{\text{video}}, \mathbf{g}_{\text{video}}, \mathbf{z}) := \phi(\mathbf{s}_{\text{video}})^\top T(\mathbf{z}) \psi(\mathbf{g}_{\text{video}}),$$

where ϕ_θ and ψ_α denote models that transform the observation and the goal observation respectively into low-dimensional representations, and T_β, a learned mapping aligning the two representations. At convergence, the ICVF provides a measure of temporal spatiality, and the learned representation $\phi_\theta(\mathbf{s})$ offers a useful feature basis for downstream value functions.

4.2 Phase 2: Multi-Task Robot Pre-Training via Offline RL

In the next phase, we refine the learned representation on a multi-task robot dataset, $\mathcal{D}_{\text{robot}}$, to narrow the domain gap between robot image observations and human video, and to provide information about the target robot embodiment (i.e., the actions affordable by the robot): note that the tasks and workspaces in this robot dataset are explicitly disjoint from the target tasks used in the downstream evaluation.

V-PTR uses multi-task conservative Q-learning (CQL) [16] to pre-train a Q-function and a policy on multi-task robot data. The Q-function and policy are conditioned on the pre-trained representation of the robot observation $\phi_\theta(\mathbf{s}_{\text{robot}})$ alongside a task vector t (either a one-hot task identifier or a language embedding from a sentence transformer). At the onset of this phase, we initialize the representation encoder ϕ_θ for both the Q-function and the policy to the encoder $\phi_{\theta^*_{\text{video}}}$ obtained at the end of phase 1.

The value function is trained to satisfy the Bellman equation,

$$\min_{\theta} \alpha \cdot \mathcal{L}_{\text{CQL}}(\theta) + \mathbb{E}_{\mathcal{D}_{\text{robot}}} \left[\left(Q_\theta(\mathbf{s}, \mathbf{a}; t) - r - \gamma \bar{Q}(\mathbf{s}', \mathbf{a}', t) \right)^2 \right],$$

with target Q-network $\bar{Q}$, and CQL regularizer $\mathcal{L}_{\text{CQL}}(\theta)$ [14], and the policy trained to maximize value,

$$\max_{\lambda} \quad \mathbb{E}_{\mathcal{D}_{\text{robot}}} \left[\mathbb{E}_{\mathbf{a} \sim \pi_\lambda(\cdot|\mathbf{s};t)} [Q(\mathbf{s}, \mathbf{a}; t)] \right] + \beta \mathcal{H}(\pi_\lambda).$$

After pre-training, we have a multi-task Q-function and policy that can be fine-tuned to the desired downstream task using a small target dataset.

4.3 Phase 3: Fine-Tuning to a Target Task

Finally, we fine-tune the value function and policy from the pre-training stage to the target task on the target dataset $\mathcal{D}_{\text{target}}$. We follow Kumar et al. [16] and treat the target data simply as a new task; fine-tuning involves assigning a new task identifier to the target data (either a new one-hot vector or a new language command), and continuing to run multi-task CQL on $\mathcal{D}_{\text{robot}}$ and $\mathcal{D}_{\text{target}}$ jointly. To emphasize the target task during fine-tuning, we perform stratified sampling where $1 - \tau$ proportion of the training batch comes from $\mathcal{D}_{\text{robot}}$ and τ from $\mathcal{D}_{\text{target}}$, where τ is small (we use $\tau = 0.1$).

4.4 Implementation Details

Video pre-training: We use video frames at a 224×224 resolution. The three components parameterizing the ICVF are implemented as separate 3-layer MLP heads on a shared visual backbone encoder. We use a Resnetv2-50 [10] as our backbone since smaller convolutional networks led to worse pre-training performance. To avoid overfitting to spurious correlations between consecutive frames in a video clip, we use image augmentation (random crops and color jitter) [13], weight decay, and dropout. We train the model for 2×10^6 gradient steps with a batch size of 64, using Adam, and learning rate 10^{-4} cosine annealed through training. All remaining designs we take from the open-source code [7].

Multi-task robot pre-training: We fine-tune on the multi-task robot dataset primarily following design decisions from Kumar et al. [16]. The CQL policy takes the RGB image, proprioceptive state, and task identifier as input. The RGB image is passed through a visual encoder, then concatenated with the remaining information, and passed through a 2-layer MLP. We additionally concatenate task and action information into each hidden layer of the MLP. The Q-function is parameterized similarly, but it also takes the action as input, which is also concatenated at each hidden layer, and does not receive the proprioceptive state. Encoder parameters for both the value and policy are initialized from the video pre-trained ICVF, but there is no further weight tying between the networks. We train multi-task CQL for 4×10^5 gradient steps with batch size of 64, and Adam with a constant learning rate of 10^{-4}. Finetuning on the target task begins at 2×10^5 gradient steps.

5 Experimental Results

The goal of our experiments is to validate the effectiveness of V-PTR in boosting the generalization and robustness of robotic offline RL. We evaluate V-PTR in several scenarios requiring generalization to new scenes, compare to other approaches for incorporating video data, and perform additional diagnostic experiments to understand how value pre-training can provide useful representations for downstream robotic RL. Our settings include several challenging robotic manipulation tasks that require zero-shot generalization to new objects and distractors. A video of our evaluations and diagnostic experiments can be found on our **project website**.

Real-world setup. We conduct our experiments on a WidowX robot platform. We perform video pre-training on Ego4D [8], an egocentric video dataset consisting of 4M transitions of humans attempting diverse tasks in the real world, using the same pre-processed subset from prior works [21; 17]. Then, for multi-task robot pre-training, we utilize the subset of the Bridge dataset [6; 33] used by prior work [16], a dataset with 150K transitions of various manipulation task demonstrations on a WidowX robot in toy kitchens. We fine-tune on several tasks performed by a WidowX robot in a *previously unseen* toy-kitchen workspace. For each target task, we collect expert demonstrations using teleoperation, with a range of robot positions, object positions, and distractor objects. Solving these tasks requires skills such as picking and placing a variety of objects, using tools to accomplish tasks (e.g., sweeping), and two-phase door opening (Figures 3 and 4).

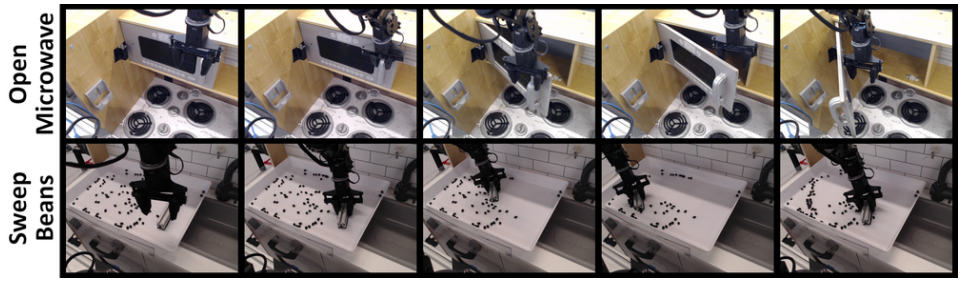

Figure 3: Examples of setup and successful rollouts for complex tasks. We utilize the robot setup from the Bridge dataset [6] for our tasks. **Top:** Two-phase open microwave; **Bottom:** Sweep beans into pile with tool.

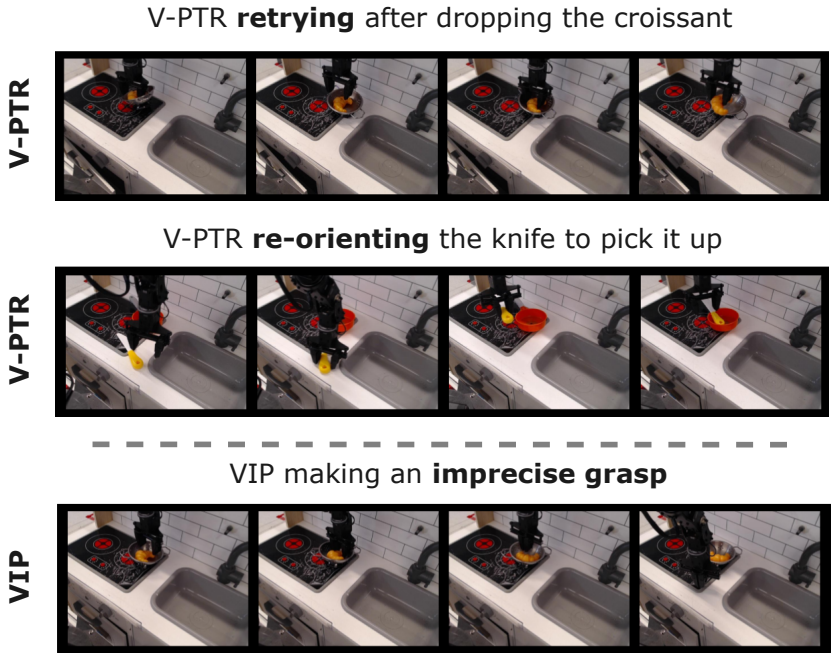

Figure 4: Visualizing qualitative performance of V-PTR and VIP. Here we show rollouts for V-PTR (top) and VIP (bottom) on the real robot manipulation tasks. V-PTR carefully executes the task by orienting the gripper to match the object and retrying on failure whereas VIP grasp objects without this re-orientation.

Comparisons to prior methods. We compare V-PTR to approaches that do not utilize video data (PTR [16], BC [6]), as well as other methods for video pre-training (R3M [21], MVP [35; 22], and VIP [17]). Following the protocols in these prior works, we fine-tune the R3M and MVP representations with imitation learning on multi-task and target robot data (phases 2 and 3). We also evaluate three versions of VIP. To provide an apples-to-apples comparison with V-PTR we use CQL with **(i)** "$\text{VIP}_{\text{frozen}}$", which freezes the representation encoder pre-trained on video data, and **(ii)** "VIP", which does not. Additionally, following [17], we evaluate **(iii)** "$\text{VIP}_{\text{reward}}$", which not only utilizes the pre-trained representation encoder but also uses distances between representations of current and future frames as a reward for reward-weighted regression. When using language task specification, we compare to language conditioned variants of BC and PTR.

5.1 Real-World Results

We evaluate V-PTR and comparisons in three testing scenarios. In **Scenario 1**, we evaluate the performance of the policy as we vary the robot's initial pose and the position of objects in scene. In **Scenario 2**, we repeat these experiments after adding novel distractor objects. Finally, in **Scenario 3**, we test the ability of the learned policy to manipulate novel target objects that were never seen during training. We evaluate different methods on four pick-and-place tasks in Table 1. Observe that V-PTR outperforms all other prior approaches, and in some tasks (e.g., "place knife in pot") is the only

Table 1: Task success rates of V-PTR and prior methods

		Video pre-training					No videos	No robot data
	Task	**V-PTR (Ours)**	**R3M+BC**	**MVP+BC**	**VIP+CQL**	**$\text{VIP}_{\text{frozen}}$+CQL**	**PTR**	**V-PTR w/o phase 2**
Scenario 1	Croissant from bowl	7 / 12	0 / 12	4 / 12	2 / 12	0 / 12	3 / 12	5 / 12
	Sweet potato on plate	6 / 12	0 / 12	1 / 12	0 / 12	0 / 12	1 / 12	1 / 12
	Knife in pot	6 / 12	0 / 12	0 / 12	0 / 12	0 / 12	0 / 12	0 / 12
	Cucumber in pot	5 / 12	0 / 12	1 / 12	0 / 12	0 / 12	1 / 12	1 / 12
	Total	24 / 48	0 / 48	6 / 48	2 / 48	0 / 48	5 / 48	7 / 48
Scenario 2 with distractor objects	Croissant from bowl	8 / 12	0 / 12	3 / 12	2 / 12	0 / 12	0 / 12	3 / 12
	Sweet potato on plate	4 / 12	0 / 12	2 / 12	0 / 12	0 / 12	1 / 12	2 / 12
	Knife in pot	4 / 12	0 / 12	0 / 12	1 / 12	0 / 12	0 / 12	0 / 12
	Cucumber in pot	4 / 12	0 / 12	0 / 12	1 / 12	0 / 12	0 / 12	1 / 12
	Total	20 / 48	0 / 48	5 / 48	4 / 48	0 / 48	1 / 48	6 / 48
Scenario 3 novel target objects	Carrot	2 / 3	0 / 3	0 / 3	1 / 3	0 / 3	0 / 3	2 / 3
	Cucumber	1 / 3	0 / 3	0 / 3	1 / 3	0 / 3	0 / 3	1 / 3
	Ice-cream	0 / 3	0 / 3	0 / 3	1 / 3	1 / 3	1 / 3	0 / 3
	Total	3 / 9	0 / 9	0 / 9	3 / 9	1 / 9	1 / 9	3 / 9

Table 2: Performance of V-PTR, VIP, and PTR on more complex tasks

				No CQL
Task	**V-PTR**	**VIP [17]+CQL**	**PTR [16]**	**$\text{VIP}_{\text{reward}}$ [17]**
Open Microwave	5 / 12	2 / 12	0 / 12	0 / 12
Sweep Beans	6 / 12	5 / 12	2 / 12	2 / 12

method that produces any successful trials. We observed that all of these methods do learn behavior that *attempt* solve the task, for example by moving toward relevant objects in the scene, but do not eventually succeed due to either imprecise localization of target objects or a prematurely executed grasp attempt. Next, we evaluate V-PTR with distractor objects in Table 1 (Scenario 2). Adding distractors reduces performance for every method, but V-PTR still exhibits the smallest degradation compared to the next best approach. To study generalization to novel target objects (Scenario 3), we also consider a "take [object] from bowl" task, and replace the target object with unseen target objects at evaluation. Performance is low for all comparisons in Table 1, but V-PTR performs as well as the best prior methods.

Comparisons to VIP on more complex tasks. We compare V-PTR in more detail to VIP (which also uses similar video data) on manipulation tasks that require learning more complex skills: "open microwave" and "sweep beans" in Table 2. We find that V-PTR achieves a higher success rate than all variants of VIP discussed above. Qualitatively in Figure 5, we observe that on the "sweep beans" task, V-PTR sweeps a larger area than the VIP policy, which is too slow to execute the sweep motion a second time. These differences in performance corroborate our analysis (Figure 6 and 7) that value functions trained on the V-PTR representation tend to have lower error than those utilizing VIP representations.

Language-based task specification. We next study how V-PTR works when the robot pre-training data is labeled with natural language descriptions (a more general format) instead of task identifiers. To handle language, we first encode the task description into an embedding vector using the pre-

Figure 5: Examples of areas swept by VIP [17] (**top**) and V-PTR (**bottom**) methods. V-PTR sweeps a much larger area (blue), and begins a second sweep, whereas VIP [17] is too slow to sweep a second time.

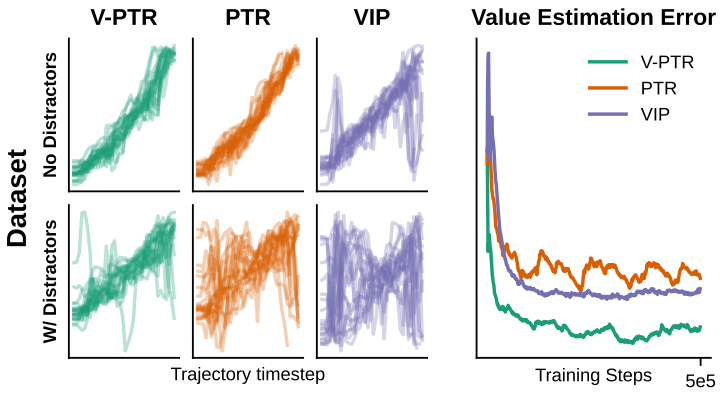

Figure 6: Visualizing the learned values $V(\mathbf{s}_t)$ w.r.t. time-step t on rollouts with no distractor objects (**top**) and rollouts with distractor objects (**bottom**) obtained after multi-task pre-training in phase 2 ($V(\mathbf{s}_t)$ is computed using the average of the multi-task Q-value under actions sampled from the learned policy). Note that values trained by PTR and VIP tend to be highly non-smooth, especially on held-out rollouts with novel distractors, whereas V-PTR produces smooth value functions.

trained language encoder from GRIF [18], which has been shown to be effective at learning BC policies. The Q-function and the policy then utilize these embedding vectors in lieu of the one-hot task identifiers, processing them identically as before. In Table 3, we compare V-PTR to imitation learning and PTR with language embeddings, and find that V-PTR improves over both of these methods by around 50%, indicating that V-PTR can leverage downstream language.

Table 3: Performance of language-conditioned V-PTR compared to language-conditioned BC and PTR

				No language
Task	**V-PTR (language)**	**BC**	**PTR**	**V-PTR (one-hot)**
Croissant	7 / 12	3 / 12	5 / 12	7 / 12
Sweet potato	6 / 12	2 / 12	6 / 12	6 / 12
Knife in Pan	5 / 12	0 / 12	3 / 12	6 / 12
Cucumber in Pot	9 / 12	0 / 12	3 / 12	5 / 12
Open Microwave	9 / 12	1 / 12	0 / 12	5 / 12
Sweep Beans	8 / 12	2 / 12	4 / 12	6 / 12
Total	**44 / 72**	8 / 72	21 / 72	35 / 72

5.2 Visualizations and Diagnostic Experiments

We now analyze V-PTR more carefully by visualizing the learned features from video-pretraining, probing the generalization of the system, and assessing the quality of value estimation.

Video pre-training via V-PTR improves target value estimation. We visualize the learned value function on frames from different rollouts in Figure 6 (left), where the true value function should monotonically increase throughout a successful rollout. We visualize the downstream value $V(\mathbf{s}_t)$ for rollouts with and without distractor objects. In the presence of novel distractor objects, V-PTR representations induce smooth and monotonic value functions while the other methods do not.

To more precisely measure the ability of V-PTR to fit downstream value functions, we train a SARSA value function (for which we may compute a closed-form optimal solution) on top of frozen pre-trained representations from PTR (no video data), VIP [17], and V-PTR, and report the error between the predicted value and the ground-truth value on a held-out dataset in Figure 6 (right). V-PTR attains the smallest fitting error compared to both PTR and VIP.

What kind of visual cues do representations trained by V-PTR capture? We probe which parts of the image influence the output of the learned policy for V-PTR and other baselines, by utilizing Grad-CAM [24] to mark patches of the frame that influence the output of the learned policy in green in Figure 7. We observe that V-PTR policies discard the scene background and focus on cues relevant

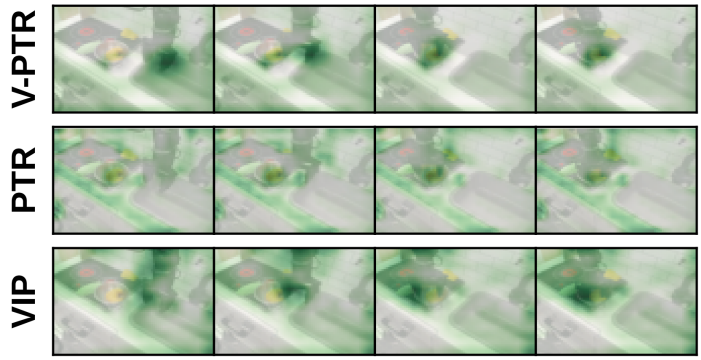

Figure 7: Grad-CAM visuals superimposed on frames from robot data. Regions highlighted in green denote patches of the observation with the most significant influence on the learned policy. Without PTR and VIP, background areas in the image exert a significant influence on the output of the learned policy. In contrast, the policy initialized with V-PTR focuses more on gripper and object positions, crucial for solving the task.

for robot control (e.g., object, gripper positions), while PTR and VIP place higher focuses on the scene background. This evidence provides an explanation for why V-PTR robustly attains better performance in our experiments.

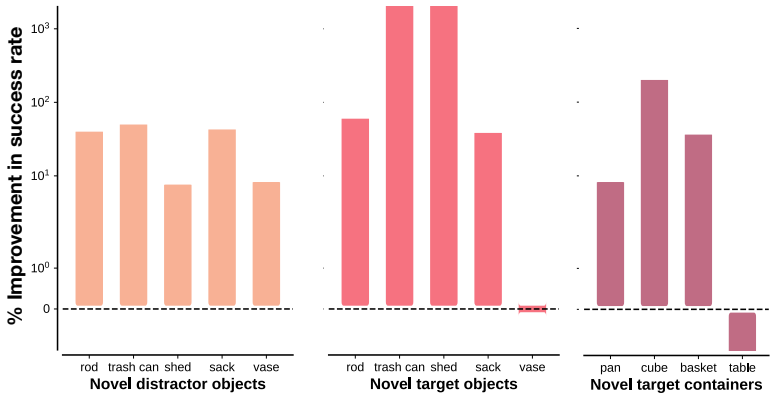

Figure 8: Simulation results. Percentage improvement in success rates of V-PTR over PTR [16], that does not utilize any video data for pre-training. y-axis is in log scale.

Simulation results. We also evaluate V-PTR in simulated pick-and-place tasks [15; 16] over a variety of target, container, and distractor objects to precisely measure generalization to new objects. Like our real-world experiments, we pre-train on video data from Ego4D, and use simulated robot demonstrations for fine-tuning. In Figure 8, we present the percentage improvement in success rates obtained by V-PTR over those of PTR [16] (with no video data). In all but one scenario, V-PTR improves over PTR, demonstrating that value-based video pre-training on Ego4D boosts the generalization ability of robotic RL.

6 Discussion and Conclusion

We designed a robotic system, V-PTR, that uses value function pre-training on the large-scale Ego4D video dataset [8] and the Bridge dataset [6] to improve the performance of policies learned downstream. While V-PTR outperforms prior methods for learning from video data, we found that all the current methods remain sensitive to deviations in workspace height, camera angle, and robot configurations. There also exist many opportunities to scale, whether incorporating multi-robot datasets, larger human video datasets with language, or larger models. Nevertheless, our evaluations and diagnostic experiments indicate the promise of using RL-like value pre-training on videos to improve robot learning algorithms.

References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017. 3

[2] Leemon Baird. Residual Algorithms : Reinforcement Learning with Function Approximation. In *International Conference on Machine Learning (ICML)*, 1995. 2

[3] Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *ArXiv*, abs/2206.11795, 2022. 3

[4] Matthew Chang, Arjun Gupta, and Saurabh Gupta. Learning value functions from undirected state-only experience. *ArXiv*, abs/2204.12458, 2022. 3

[5] Christoph Dann, Gerhard Neumann, Jan Peters, et al. Policy evaluation with temporal differences: A survey and comparison. *Journal of Machine Learning Research*, 15:809–883, 2014. 2

[6] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021. 2, 3, 5, 6, 9, 12, 13

[7] Dibya Ghosh, Chethan Bhateja, and Sergey Levine. Reinforcement learning from passive data via latent intentions. *arXiv preprint arXiv:2304.04782*, 2023. 2, 3, 4, 5, 12, 13

[8] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022. 2, 3, 5, 9, 13

[9] K He, X Chen, S Xie, Y Li, P Dollár, and RB Girshick. Masked autoencoders are scalable vision learners. arxiv. 2021 doi: 10.48550. *arXiv preprint arXiv.2111.06377*, 2021. 2

[10] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 2016. 5

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 15

[12] Siddharth Karamcheti, Suraj Nair, Annie S Chen, Thomas Kollar, Chelsea Finn, Dorsa Sadigh, and Percy Liang. Language-driven representation learning for robotics. *arXiv preprint arXiv:2302.12766*, 2023. 2

[13] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020. 5

[14] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020. 5, 12, 14

[15] Aviral Kumar, Anikait Singh, Stephen Tian, Chelsea Finn, and Sergey Levine. A workflow for offline model-free robotic reinforcement learning. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=fy4ZBWxYbIo. 9

[16] Aviral Kumar, Anikait Singh, Frederik Ebert, Yanlai Yang, Chelsea Finn, and Sergey Levine. Pre-training for robots: Offline rl enables learning new tasks from a handful of trials. *RSS 2023; arXiv:2210.05178*, 2023. 2, 3, 4, 5, 6, 7, 9, 12, 13, 14, 15

[17] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022. 2, 5, 6, 7, 8, 13, 14, 15

[18] Vivek Myers, Andre He, Kuan Fang, Homer Walke, Philippe Hansen-Estruch, Ching-An Cheng, Mihai Jalobeanu, Andrey Kolobov, Anca Dragan, and Sergey Levine. Goal representations for instruction following: A semi-supervised language interface to control. *arXiv preprint arXiv:2307.00117*, 2023. 8

[19] A. Nair, S. Bahl, A. Khazatsky, V. Pong, G. Berseth, and S. Levine. Contextual imagined goals for self-supervised robotic learning. In *Conference on Robot Learning (CoRL)*, 2019. 2

[20] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhi Gupta. R3m: A universal visual representation for robot manipulation. *ArXiv*, abs/2203.12601, 2022. 2, 13, 14

[21] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022. 2, 5, 6

[22] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. *arXiv preprint arXiv:2210.03109*, 2022. 2, 6, 14

[23] Karl Schmeckpeper, Oleh Rybkin, Kostas Daniilidis, Sergey Levine, and Chelsea Finn. Reinforcement learning with videos: Combining offline observations with interaction. In *Conference on Robot Learning*, 2020. 3

[24] RR Selvaraju, M Cogswell, A Das, R Vedantam, D Parikh, and D Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. arxiv 2016. *arXiv preprint arXiv:1610.02391*. 8

[25] Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and P. Abbeel. Masked world models for visual control. *ArXiv*, abs/2206.14244, 2022. 2

[26] Younggyo Seo, Kimin Lee, Stephen James, and P. Abbeel. Reinforcement learning with action-free pre-training from videos. *ArXiv*, abs/2203.13880, 2022. 2

[27] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141, 2017. 2

[28] Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. Cog: Connecting new skills to past experience with offline reinforcement learning. *arXiv preprint arXiv:2010.14500*, 2020. 12

[29] A. Srinivas, Michael Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, 2020. 2

[30] Bradly C. Stadie, P. Abbeel, and Ilya Sutskever. Third-person imitation learning. *ArXiv*, abs/1703.01703, 2017. 3

[31] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *ArXiv*, abs/1805.01954, 2018. 3

[32] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *ArXiv*, abs/1807.06158, 2018. 3

[33] Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, et al. Bridgedata v2: A dataset for robot learning at scale. *arXiv preprint arXiv:2308.12952*, 2023. 5

[34] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *ArXiv*, abs/2203.06173, 2022. 2, 14

[35] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022. 2, 6

Appendices

A Full System Description: V-PTR

In this section, we will provide further details regarding the 3 phases of V-PTRİn particular, we will describe the video pre-training with ICVF [7] as well as Multi-robot pretraining with PTR [16], describing both the networks present during training as well as the objective function.

A.1 Video Pre-Training with ICVF

Network. We define the visual backbone $f_\theta(\mathbf{s})$ to be a ResNet50-v2 model, which outputs a 2048-dimensional vector. The ICVF heads ϕ, ψ, T are 2-layer MLPs with hidden dimensions of 256 and a final dimension of 256. The ICVF is then defined as $V(\mathbf{s}, \mathbf{g}, \mathbf{z}) = \phi(f(\mathbf{s}))^\top T(f(\mathbf{z}))\psi(f(\mathbf{g}))$ for video frames $\mathbf{s}, \mathbf{g}, \mathbf{z}$.

A.2 Multi-robot pretraining with PTR

Networks. We mirror the experimental setup of [16], with no modifications except for a different visual backbone. PTR uses CQL [14] as the base offline RL method, meaning it trains two Q functions, a separate policy, and delayed target copies of the Q-functions. Each network is represented by a 3-layer MLP head with width of 256, after the visual representation. To make comparisons between PTR and V-PTR exact, we also use separate encoders for the actor and critic, although both are initialized at the beginning of Phase 2 using the same visual representation as learned during Phase 1. We refer to [16] for a more complete description.

B Environment Setup and Dataset Details

In this section, we will describe the setup for our experimental results with respect to both the real-world experiments and the sim diagnostic experiments. There will be a description of the task setup as well as the corresponding datasets associated with the tasks.

B.1 Description of State and Action Space

Our state and action description follows that of PTR [16]. The state is a 128×128 RGB image captured from an over-the-shoulder camera, a one-hot vector identifying which task is to be performed, the position and rotation of the end-effector, and how much the gripper is closed. The action is the translational and angular velocity of the robot end-effector, as well as how much the gripper is closed. Rotations and angular velocities are expressed using Euler angles.

B.2 Description of Real Robotic Setup

We mirror our real-world experimental setup from Bridge [6] and PTR [16]. In particular, we designed and evaluated the performance of our method under several distinct conditions in 2 different toy kitchen domains. The robot that is used is a 6-DoF WidowX 250 robot with a fixed side camera. The scene in which the robot was placed in was a toy kitchen with elements such as stove tops, sinks, microwaves, and food objects found in the scene. A picture of our setup is found in Figure 9.

We evaluate several pick place tasks, in which the agent has to place an object into a container amongst distractors found in the scene. Distractor objects in this scene can include other objects and containers that may have even been shown to be corresponding to different tasks.

B.3 Description of our Simulation Task

Our manipulation tasks in simulation are modified versions of the pick-and-place tasks from Singh et al. [28], which were then extended by Kumar et al. [16]. This environment requires controlling a simulated WidowX 250 arm which is placed in front of a single bin with an object to place in the bin and a distractor. These objects are selected from a pool of 10 objects, with no overlap between the objects to be placed in the bin and the distractors. The goal of the task is to pick up the object and place it in the bin successfully. The setup is designed in the PyBullet simulation framework which models the physics of the robotic arm, the objects in the scene, as well as the interactions between them. A picture of our setup is found in Figure 10. The dataset is collected with a scripted policy that reaches for the object of interest, grasps it with some likelihood, and places it in the bin. This collected dataset had an overall success rate of 30%.

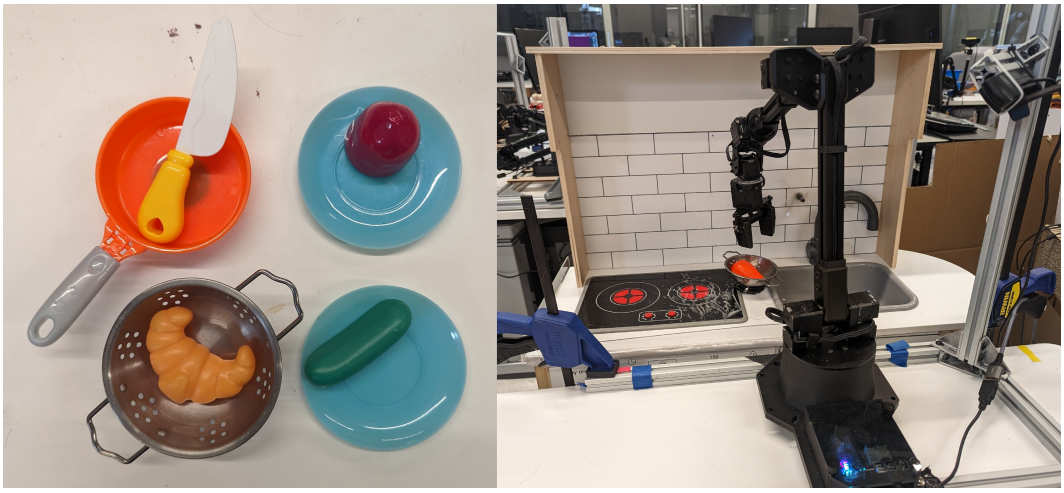

Figure 9: Real-robot experiments. We utilize the setup from the Bridge dataset [6]. The bridge dataset is collected on a 6-DoF WidowX 250 robot, with a fixed side camera placed in diverse toy-kitchens. Observations for the tasks consist of one 128×128 RGB image from a side camera, as well as robot proprioception. **Left:** task objects and containers for the tasks that we study. **Right:** Evaluation setup pick place environment.

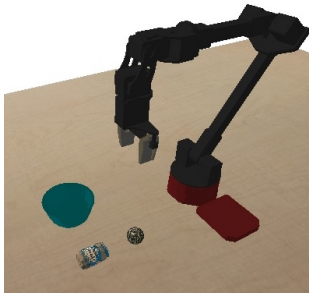

Figure 10: Observations for the simulated pick-and-place tasks consist of one 128×128 RGB image from a top-down camera, as well as a proprioceptive robot state.

B.4 Description of Video Pre-Training Dataset

The Video Pre-training Dataset that we utilize in Stage 1 of our method, PTR, is Ego4D [8]. We used the same pre-processed Ego4D dataset as in R3M [20] and VIP [17]. In particular, long videos that are raw in nature are clipped to be shorter consisting of 10-150 frames. From here, the clipped video is decomposed into individual frames that are pre-processed with a random crop at the video level. The frame is then resized and center-cropped to be of size $224 \times 224 \times 3$. These processed video frames are then fed into the replay buffer to be individual transitions for the ICVF objective [7].

B.5 Description of Real-World Multi-Task Robot Datasets

For the pre-training and target datasets in Stage 2 of V-PTR for the real-world multi-task training, we utilize subsets of the Bridge Dataset [6]. Mirroring the setup of Scenario 3 in PTR [16], the pre-training data comprises of all pick-and-place data found in the bridge dataset except for any demonstration data collected in the toy kitchen that was evaluated on. For the target dataset, we collect 44 new demonstrations for each of the 4 tasks: Removing Croissant from Colander, Placing Knife in Pot, Placing Sweet Potato on Plate, and Placing Cucumber in Bowl. A total of 218 successful demonstrations were collected with a human demonstrator using an Oculus Quest Headset for Phase 3 of V-PTR.

C Evaluation Details

In this section, we will provide how real and simulated environments were evaluated fairly across our method and baselines. This protocol is similar in nature to the one presented in PTR [16].

C.1 Evaluation Protocol for Real-World Experiments

To evaluate a policy in the real world, we loop through 4 starting gripper transformations that move the gripper to the left front, right front, left rear, and right rear of the kitchen environment. For each of these starting transformations, we evaluate a total of 3 times: once with the object to pick up directly underneath the gripper and twice with it shifted slightly so that the policy cannot simply grip from the starting location. For each of these evaluations, we let the policy run for 60 seconds.

For our experiments testing generalization to novel distractor objects, we place two novel distractor objects into the scene. We shift the locations of these objects between each evaluation and switch out the objects themselves when we change start transforms so that each combination of distractors is evaluated 3 times.

For new target object experiments, which we only conducted for the croissant out of colander task, we replace the croissant with a new object never seen at training time. We switch out the target object for each start transform so that each new target object is evaluated 3 times.

When taking objects out of containers, we do not count knocking over the container so that the object falls out as a success - the gripper must lift up the object in the air and set it outside the container.

C.2 Evaluation Protocol for Simulation Experiments

The multi-task robot dataset in these experiments consists of 1500 pick-place trajectories for 6 target objects. We created three small target datasets to test generalization along different axes: **(a)** diverse target objects to pick and place, **(b)** diverse containers holding the target objects, and **(c)** diverse distractor objects in the scene. Each of these datasets contains 10 successful demonstrations of the target task, spread across 5 different objects. We assign a unique one-hot task id for each of the source target objects and a single task id for the target dataset. Then, we train our system, V-PTR, and a baseline PTR policy using multi-task CQL with hyperparameters $\alpha = 5$ and target mixing ratio $\tau = 0.7$.

When evaluating zero-shot generalization, we test each of these policies along the axis of generalization present in their target datasets. In particular, for the policies trained with diverse targets, diverse distractors, and diverse containers, we test generalization to 5 new targets, distractors, and containers respectively, which are unseen in the source and target datasets. Each evaluation is performed with 20 rollouts and an environment time limit of 40.

D Experimental Details for V-PTR and Baseline Methods

D.1 CQL Finetuning [16]

For our second pre-training phase on multi-task robot data and fine-tuning on target data, we utilized CQL [14] as the downstream offline RL algorithm. Following the design decisions in the official implementation of PTR [16], we utilized a variant of Bellman backup that computes the target value by performing a maximization over target values computed for $n = 4$ actions sampled from the policy at the next state (max_q_backup). In each domain, we swept over the alpha values of $\alpha = 0.1, 1, 5, 10$. We built our code upon the CQL implementation from https://github.com/Asap7772/PTR [16].

D.2 MVP [34; 22]

For masked autoencoding (MAE)-based methods such as MVP [22], we loaded in the pre-trained PyTorch Checkpoints for the ViT-Base model. For this method, we kept the pre-trained representation frozen and finetuned a Multi-Layer Perceptron (MLP) that takes in the class token of the ViT as input to the model and outputs an action dimensional vector. The output of the MAE is normalized with layer normalization. Other hyperparameters follow the CQL fine-tuning section.

D.3 VIP [17] and R3M [20]

Prior methods have shown that TD-based updates with networks with batch normalization have instabilities during training. Given this, methods such as VIP and R3M that utilize a standard ResNet-

50 [11] backbone with batch normalization do not finetune stably for methods such as CQL. For this reason, any comparisons using standard ResNet 50 either use frozen batch statistics, or are trained with a behavioral cloning objective.

All hyperparameters described above are summarized in Table D.

E Reducing the Number of Fine-Tuning Demonstrations

We examine how reducing the number of demonstrations from 44 to 10 degrades performance in Table 5. V-PTR continues to significantly outperform baselines.

Table 5: Task success rates of V-PTR and prior methods with only 10 demonstrations.

		Video pre-training	**No videos**
Task	**V-PTR (Ours)**	**VIP [17]+CQL**	**PTR [16]**
Croissant out of Colander	4 / 12	1 / 12	0 / 12
Sweet Potato on Plate	5 / 12	0 / 12	0 / 12
Knife in Pan	2 / 12	0 / 12	0 / 12
Cucumber in Pot	4 / 12	0 / 12	0 / 12
Open Microwave	8 / 12	2 / 12	4 / 12
Sweep Beans	2 / 12	0 / 12	5 / 12