SEEQ: TRAINING GENERALIST VALUE FUNCTIONS FOR LONG-HORIZON ROBOTIC MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite rapid progress, generalist robot policies remain brittle on complex, long-horizon tasks that comprise multiple stages or require repeated attempts and deliberation on the same underlying stage before success. Q-value functions can improve these policies by ranking candidate actions or guiding policy improvement, but learning from sparse task-level rewards entails long credit-assignment horizons, difficult Bellman backups, and broad data-coverage requirements. We introduce SEEQ (**S**ubtask-**e**licit**e**d **Q**-functions), which instead learns Q-values for the currently active subtask. This shortens the value-prediction horizon and enables effective learning with temporal-difference (TD) objectives. During training, subtask-level annotations present in offline robot data provide the decomposition and enable learning from broad, potentially suboptimal robot datasets. To eliminate the need for human annotations or modular subtask prediction systems at test time, our Q-function architecture is trained to autoregressively predict the active subtask in natural language before estimating its value. We instantiate SEEQ using a base vision-language backbone, pretrain it on diverse open-source robot manipulation data, and finetune it on downstream tasks. Across four real-world manipulation tasks on two bimanual robot platforms, the SEEQ value function substantially improves best-of-N policy steering.

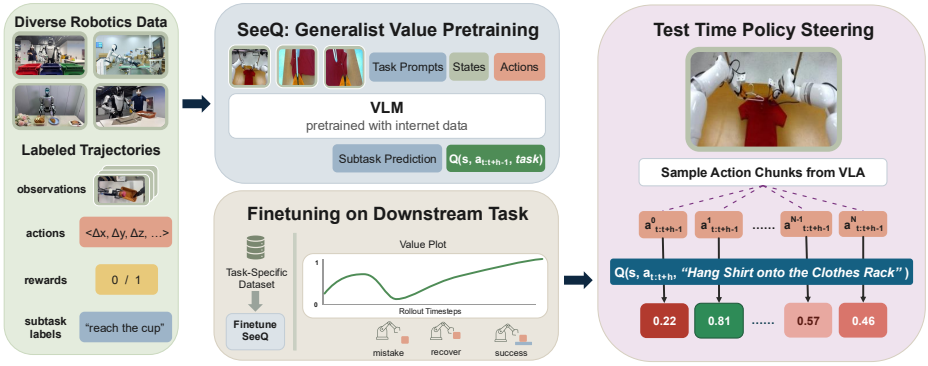


Figure 1: SEEQ overview. We train a generalist Q-value function on diverse robot data using a pretrained vision-language model (VLM) backbone. The value function is then finetuned on a downstream task and used to steer a base policy at test time. To address the challenges of long-horizon value learning, we use temporal-difference (TD) learning to fit the Q-value of the *currently active* subtask rather than that of the entire task. To eliminate the need for humans to specify this subtask at deployment, we modify the Q-function architecture to first predict the active subtask in natural language and then predict its value, encouraging internal representations better aligned with subtask-level value estimation.

1 INTRODUCTION

Generalist robot policies provide a promising foundation for robot control: they acquire useful perceptual and motor priors, follow language instructions, and transfer across objects, scenes, and embodiments (Intelligence et al., 2025b). Yet robust autonomous deployment remains challenging, particularly for tasks that unfold over long horizons or demand high precision. Some long-horizon

Project website with videos: https://anonymous.4open.science/w/seeq-iclr-67A3/.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

tasks may involve multiple stages, while others might simply be shorter precision-heavy tasks that require deliberating among similar actions and retrying after imperfect attempts. Imitation-trained policies struggle in both settings: errors compound over time, recovery behaviors are underrepresented in expert datasets, and behaviors of varying quality are modeled indiscriminately. Value functions offer a natural remedy by distinguishing actions that make progress from those that lead to failure, thereby steering a base policy toward more successful behavior (Nakamoto et al., 2025).

Recent work has shown that learned value functions can substantially improve imitation learning policies in the real world (Mark et al., 2025; Wang et al., 2026; Intelligence et al., 2025a). Ideally, rather than learning a separate value function from scratch for each task, we would train a single *generalist* Q-function that can improve policies across many tasks while benefiting from large-scale pretraining and diverse robot data. However, effective value-learning methods such as temporal-difference (TD) learning become unreliable over long horizons (Park et al., 2025), precisely the regime in which complex tasks present the greatest challenges. Some approaches avoid this issue by discarding TD learning and directly regressing to Monte Carlo returns. Without TD learning, however, the value function cannot improve the policy far beyond the data collection policy, which limits the extent to which value learning can be helpful. This raises the central question of our work: *can we build a generalist value learning approach that utilizes TD learning to learn from diverse data, sidesteps training over long-horizons, but can still produce value functions that are useful for long-horizon tasks?*

Our approach exploits a common structure in manipulation tasks: although a task specifies a distant outcome, the robot's behavior at any moment is typically directed toward an immediate objective. Multi-stage tasks progress through intermediate milestones, such as grasping an object before placing it, while precision- or deliberation-heavy tasks may require repeated attempts to accomplish a given stage, but each attempt may follow a particular short-horizon objective. In both cases, evaluating progress toward the active subtask requires a shorter prediction horizon than evaluating success on the full task. Building on this structure, we train a generalist Q-function on a vision-language model (VLM) backbone and redefine its prediction target. Rather than modeling sparse success over the full task, the Q-function estimates the expected return for the currently active subtask. This yields SEEQ (**S**ubtask-**e**licit**e**d **Q**-functions), which recasts long-horizon value learning as short-horizon, subtask-level value learning. To eliminate the need for subtask annotations at test time, we modify the Q-function to first predict the active subtask in natural language and then estimate its value. In addition, this objective improves robustness near subtask transitions and aligns value estimation with the pretrained VLM's text-generation capabilities.

We instantiate SEEQ using a pretrained 3B PaliGemma vision-language backbone (Beyer et al., 2024) and train it on diverse robot manipulation datasets with subtask annotations, including RoboCOIN (Wu et al., 2026). We evaluate whether the resulting Q-function can infer subtasks for held-out instructions, adapt to new embodiments with limited finetuning, and improve fixed, generalist imitation-learned policies. Empirically, SEEQ decomposes out-of-distribution tasks into meaningful subtask spans and predicts useful values for them, with the subtask-prediction loss proving critical to performance. When used for best-of-N action selection, SEEQ improves generalist robot policies (Intelligence et al., 2025b) on **four** real-world bimanual manipulation tasks across **two** robot platforms, including precision-heavy tasks with deformable objects requiring recovery and planning-heavy tasks comprising many stages. These results suggest that subtask-level values provide a more effective learning target than sparse, long-horizon success when training generalist Q-functions.

2 PRELIMINARIES, DEFINITIONS, AND NOTATION

We formulate our problem in a language-conditioned robotic manipulation setting with access to a base generalist policy, π_{base}, that we wish to improve. In our experiments, we often finetune an off-the-shelf policy on in-domain teleoperation data. Given a natural language task instruction l and a state $\mathbf{s}_t$, specified by visual observations (e.g., over-the-shoulder and wrist cameras) and proprioceptive robot state, the policy produces an action chunk (Zhao et al., 2023), i.e., $\mathbf{a}_{t:t+H-1} \sim \pi_{\text{base}}(\cdot \mid \mathbf{s}_t, l)$. The environment then executes this chunk, or a short prefix of it, before replanning from a new state.

To improve this policy, typically we learn a Q-function that estimates the expected reward-to-go of an action chunk $\mathbf{a}_{t:t+H-1}$ from state $\mathbf{s}_t$, conditioned on the task instruction l. The instruction specifies a sparse reward function r_l, where $r_l(\mathbf{s}_t) = 1$ if task l is complete at state $\mathbf{s}_t$ and 0 otherwise. Formally,

the Q-function for instruction l is $Q_l^\mu(\mathbf{s}_t, \mathbf{a}_{t:t+H-1}) = \mathbb{E}_\mu \left[\sum_{j=0}^{\infty} \gamma^j r_l(\mathbf{s}_{t+j}) | \mathbf{s}_t, \mathbf{a}_{t:t+H-1}\right]$, where μ denotes the policy used after the initial action chunk and $\gamma \in [0, 1)$ is a discount factor.

Q-function training approaches. We train a generalist Q-function $Q_\theta(\mathbf{s}_t, \mathbf{a}_{t:t+H-1}, l)$ from an offline trajectory dataset $\mathcal{D}_{\text{pre}}$. For a recorded trajectory $\tau \sim \mathcal{D}_{\text{pre}}$ ending at timestep T, the return-to-go is $G_t^l = \sum_{j=0}^{T-t} \gamma^j r_l(\mathbf{s}_{t+j})$. **Monte Carlo (MC)** regression fits Q_θ to G_t^l with a squared-error loss. It avoids bootstrapping but can have high-variance targets and evaluates the dataset's behavior policy.

Temporal-difference (TD) learning uses Q-chunking (Li et al., 2025a) to combine discounted rewards over an action chunk with a bootstrapped estimate from a stale target network $Q_{\bar{\theta}}$:

$$y_t^l = \sum_{j=0}^{H-1} \gamma^j r_l(\mathbf{s}_{t+j}) + \gamma^H Q_{\bar{\theta}}\left(\mathbf{s}_{t+H}, \mathbf{a}'_{t+H:t+2H-1}, l\right),$$
$$\mathcal{L}_{\text{TD}}(\theta) = \mathbb{E}_{\tau \sim \mathcal{D}_{\text{pre}}, t}\left[\left(Q_\theta(\mathbf{s}_t, \mathbf{a}_{t:t+H-1}, l) - y_t^l\right)^2\right], \tag{1}$$

where $\mathbf{a}'_{t+H:t+2H-1} \sim \mu(\cdot \mid \mathbf{s}_{t+H})$ is the backup action chunk. **SARSA** uses the next recorded chunk, $\mathbf{a}'_{t+H:t+2H-1} = \mathbf{a}_{t+H:t+2H-1}$. Like MC, it fits a Q-function to the behavior policy, which may be undesirable when the data is collected from a suboptimal policy.

Alternatively, we could select the action chunk that maximizes the target Q-function at the next state. Computing this maximum exactly is intractable because action chunks are high-dimensional and learning a separate policy-improvement operator is both costly and unstable in offline RL (Mark et al., 2025). Instead, we exploit the fact that generalist policies provide a strong prior over the relevant action space: we sample N chunks from the base policy and select the one with the highest target value:

$$\mathbf{a}^{(i)}_{t+H:t+2H-1} \sim \pi_{\text{base}}(\cdot | \mathbf{s}_{t+H}, l), \qquad i = 1, \ldots, N,$$
$$\mathbf{a}'_{t+H:t+2H-1} = \underset{\mathbf{a} \in \{\mathbf{a}^{(i)}_{t+H:t+2H-1}\}_{i=1}^N}{\arg\max} Q_{\bar{\theta}}(\mathbf{s}_{t+H}, \mathbf{a}, l), \tag{2}$$

We refer to this approach as TD learning over a best-of-N policy (**TD-BoN**). By maximizing over actions sampled from the base policy, the backup fits the Q-function to the action distribution of an improved policy rather than the dataset's behavior policy. This supports policy improvement when the data provides sufficient coverage. Both TD variants bootstrap, trading lower target variance for estimation errors that can compound over long horizons, especially with limited state-action coverage (Park et al., 2025).

Language conditioning for generalist Q-functions. Independent of the training objective, we must choose what inputs the Q-function obtains. In addition to the state-action pair, we also need to condition any generalist Q-function on some form of a language instruction. There are several choices for this. The first option conditions only on the task instruction l, as in $Q_\theta(\mathbf{s}_t, \mathbf{a}_{t:t+H-1}, l)$ above, so the Q-function models the reward-to-go of the full task. The second option assumes that each trajectory is additionally segmented into subtasks, and denotes the active subtask at state $\mathbf{s}_t$ by $\tilde{l}_t$. The Q-function then takes both the task and the active subtask as input, $Q_\theta(\mathbf{s}_t, \mathbf{a}_{t:t+H-1}, l, \tilde{l}_t)$, but is trained to model only the value of the active subtask, i.e., the reward-to-go under the subtask reward $r_{\tilde{l}_t}$ rather than the task reward r_l. The task l is included as an input in this option only to make predicting the subtask tractable: the ground-truth subtask $\tilde{l}_t$ is not available at inference time and must be inferred from the observation and the task instruction. Conditioning on the task alone means the Q-function must assign credit over the full long-horizon task, which is precisely what makes value learning struggle in the long-horizon setting.

Evaluation setting: policy steering. We use the learned Q-function to steer a fixed base policy (Nakamoto et al., 2025). At each state, the policy proposes N action chunks, and the robot executes the highest-scoring one, using the selection in Equation 2 with the online network Q_θ in place of the target network $Q_{\bar{\theta}}$. We seek a generalist critic initialization that can adapt to new tasks with limited finetuning.

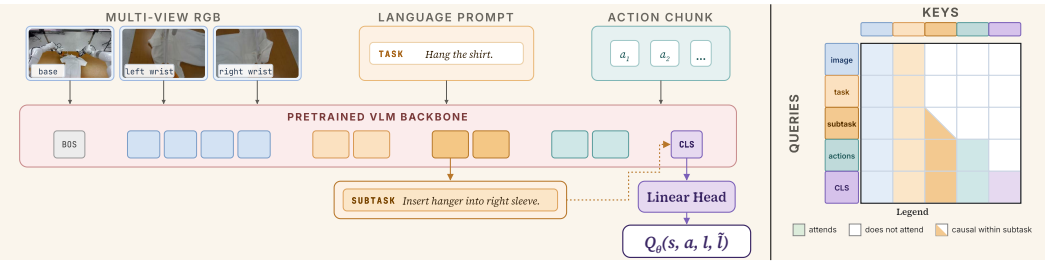


Figure 2: Overview of the SEEQ value-function architecture. The model first rolls out the active subtask at state $\mathbf{s}$ in text using a next-token prediction loss, and then predicts a scalar value $Q_\theta(\mathbf{s}, \mathbf{a}, l, \tilde{l})$ for the input state-action pair conditioned on the task instruction and the subtask. Ground-truth subtask annotations supervise the rollout during training; at inference the model conditions on its own predicted subtask.

3 SEEQ: SUBTASK-ELICITED Q-FUNCTIONS FOR LONG-HORIZON MANIPULATION

Our approach combines a shorter-horizon value-learning target with a generalist VLM backbone. Full-task Q-estimation can accumulate errors over long horizons (Park et al., 2025), especially under limited offline coverage. We instead estimate values for the active subtask and use the VLM to infer that subtask from the observation and task instruction.

Key idea: Learning Q-values at the subtask level. A natural way to shorten the value-learning horizon is to exploit the local structure of manipulation rollouts, especially when VLM backbones are used for value learning. If a task decomposes into subtask spans that each pursue a local objective, such as an intermediate stage in a multi-stage manipulation task or a new attempt within a precision-heavy stage, the value function can evaluate progress toward the immediate objective rather than predict eventual task success. Classical hierarchical RL uses a related decomposition (Stolle & Precup, 2002; Nachum et al., 2018), but typically relies on hand-specified subgoals and a high-level policy that selects among them. In our setting, subtasks are expressed in language, allowing a single language-conditioned Q-function to infer the active subtask and evaluate actions against it.

Following Section 2, we assume that each trajectory $\tau \in \mathcal{D}_{\text{pre}}$ is paired with a task instruction l and segmented into subtask spans[1], where $\tilde{l}_t$ denotes the active subtask at $\mathbf{s}_t$. We define a sparse binary reward for each span, with $r_{\tilde{l}_t}(\mathbf{s}_t) = 1$ when the active subtask terminates and 0 otherwise, following prior work (Kumar et al., 2022b). Rather than learning a Q-function for the full task l, SEEQ predicts the value of the input action chunk for the active subtask $\tilde{l}_t$, assuming policy μ is followed thereafter:

$$Q^\mu(\mathbf{s}_t, \mathbf{a}_{t:t+H-1}, \tilde{l}_t) = \mathbb{E}_\mu \left[\sum_{j=0}^{\infty} \gamma^j \cdot r_{\tilde{l}_t}(\mathbf{s}_{t+j}) \middle| \mathbf{s}_t, \mathbf{a}_{t:t+H-1} \right]. \tag{3}$$

Explicitly decoding the subtask. Since the ground-truth subtask $\tilde{l}_t$ is not available at inference time, the Q-function must infer it from the observation and the task instruction. SEEQ explicitly runs this inference: the model first autoregressively decodes the active subtask in text, $\hat{l}_t \sim p_\theta(\cdot \mid \mathbf{s}_t, l)$, and then predicts the Q-value conditioned on it, $Q_\theta(\mathbf{s}_t, \mathbf{a}_{t:t+H-1}, l, \hat{l}_t)$. The subtask prediction is supervised with a next-token prediction loss on the ground-truth annotation $\tilde{l}_t = (\tilde{w}_{t,1}, \ldots, \tilde{w}_{t,M})$,

$$\mathcal{L}_{\text{subtask}}(\theta) = -\mathbb{E}_{\tau \sim \mathcal{D}_{\text{pre}},\, t} \left[\frac{1}{M} \sum_{m=1}^{M} \log p_\theta \left(\tilde{w}_{t,m} \mid \tilde{w}_{t,<m}, \mathbf{s}_t, l \right) \right], \tag{4}$$

while the value head is conditioned on the ground-truth subtask during training. The TD target for Equation 3 is constructed using a target network $Q_{\bar{\theta}}$:

$$y_t = \sum_{j=0}^{H-1} \gamma^j r_{\tilde{l}_t}(\mathbf{s}_{t+j}) + \gamma^H (1 - I_t) \cdot Q_{\bar{\theta}} \left(\mathbf{s}_{t+H}, \mathbf{a}'_{t+H:t+2H-1}, l, \tilde{l}_{t+H} \right), \tag{5}$$

[1]Several robot datasets include subtask annotations. When unavailable, they can be generated by prompting VLMs.

where $\mathbf{a}'_{t+H:t+2H-1}$ is the backup action chunk and I_t indicates a subtask boundary in $t : t+H-1$. Note that we do not bootstrap across subtask boundaries as it defeats the purpose of subtask-level Q-functions; the backup therefore applies only when $I_t = 0$. The overall objective of SEEQ combines subtask-level TD learning with the subtask-prediction loss:

$$\mathcal{L}_{\text{SEEQ}}(\theta) = \mathcal{L}_{\text{TD}}(\theta) + \lambda_{\text{subtask}}\mathcal{L}_{\text{subtask}}(\theta), \tag{6}$$

where $\mathcal{L}_{\text{TD}}$ regresses Q_θ to y_t as in Equation 1 and λ_{subtask} controls the strength of the next-token prediction loss. In principle, the active subtask need not be explicitly decoded: the Q-function could condition only on the full-task instruction while being trained to match the value of the active subtask, relying on its hidden representations to infer that subtask implicitly. However, explicitly decoding the active subtask at inference time and conditioning value prediction is expected to encourage consistency between the inferred subtask and its value. This is particularly important near subtask boundaries, where similar image observations may correspond to different subtasks and implicit inference can produce noisy values that incorrectly steer the policy. SEEQ therefore decodes the active subtask before predicting its value (Figure 2). We visualize value predictions with and without explicit subtask decoding in Figure 7.

Implementation with a VLM backbone. We parameterize Q_θ with a PaliGemma VLM (Beyer et al., 2024) and pretrain it on broad robot data before finetuning on the target task. The VLM provides vision-language priors for recognizing task progress and interpreting instructions, while regularizing these input spaces during downstream finetuning. Because it lacks a prior over robot actions, robot pretraining additionally exposes the model to diverse action chunks across tasks and embodiments, with the aim of regularizing its action conditioning and reducing memorization. We discuss implementation details in Appendix A.1.4.

Although the architecture supports other value-learning objectives, we use the **TD-BoN** action selection in Equation 2 for the backup in Equation 5, aligning training with test-time policy steering. Appendix A.1.3 examines value overestimation.

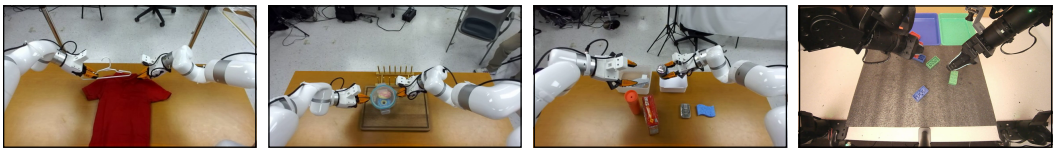
Test-time inference. Given state $\mathbf{s}_t$ and task l, SEEQ decodes the active subtask $\hat{l}_t$ and applies the best-of-N steering procedure in Section 2, scoring candidates with $Q_\theta(\mathbf{s}_t, \mathbf{a}_{t:t+H-1}, l, \hat{l}_t)$.

4 EXPERIMENTS

Implementation setup. We train SEEQ in two stages. We first pretrain a PaliGemma-based Q-function on RoboCOIN (Wu et al., 2026), restricted to its bimanual embodiments, which yields 131 tasks and approximately 40k episodes. We then finetune the resulting generalist Q-function on data from each target task; the amount of downstream data differs across tasks. We use $\pi_{0.5}$ (Intelligence et al., 2025b) as our base policy and finetune it directly on the same target-task data. Across methods, we use $N = 8$ candidate action chunks for both TD-BoN backups during training and best-of-N policy steering at inference. We average the subtask prediction loss over subtask tokens and assign it a weight of $\lambda_{\text{subtask}} = 0.1$.

Tasks. We evaluate on four real-world, long-horizon bimanual manipulation tasks at 60 Hz (Figure 3) that test the ability of the Q-function trained via SEEQ to steer a robot policy through long, multistage tasks that often demand precise low-level behavior. We often observe that a Q-function is needed to select the best action in states requiring precision and to prevent errors from compounding over longer rollouts.

1. **shirt-hang** (Hu et al., 2025) (1–2 min; 386 episodes): remove a hanger from a rod, insert it into both sleeves of a t-shirt, and hang the shirt back on the rod. The task demands bimanual coordination and precise positioning to guide the hanger into each sleeve.
2. **lid-sealing** (Hu et al., 2025) (1–2 min; 447 episodes): pick up the lid of a food-storage container, place it on the container, and close all four latching flaps. The task demands precise alignment of the lid and rim for proper sealing.
3. **grocery-packing** (Anonymous, 2026) (1–3 min; 473 episodes): pack deformable grocery items into boxes. The task demands grasping diverse objects, following episode-specific language instructions, and bimanual handoffs for objects lying opposite their target box.
4. **LEGO-disassembly** (1–2 min; 97 training episodes): separate assembled LEGO blocks of different colors and place each block into the tray of the matching color. Similar to **grocery-packing**, the instruction varies across episodes, since the pairing of block colors and tray colors changes.



(a) shirt-hang (b) lid-sealing (c) grocery-packing (d) LEGO-disassembly

Figure 3: Our bimanual real-robot evaluation tasks. Top-camera snapshots from dataset demonstrations of the four real-world tasks. Three tasks are set up on a bimanual xArm-7 platform, and one uses a bimanual YAM platform. See Appendix A.1.5 for task definitions and evaluation protocols.

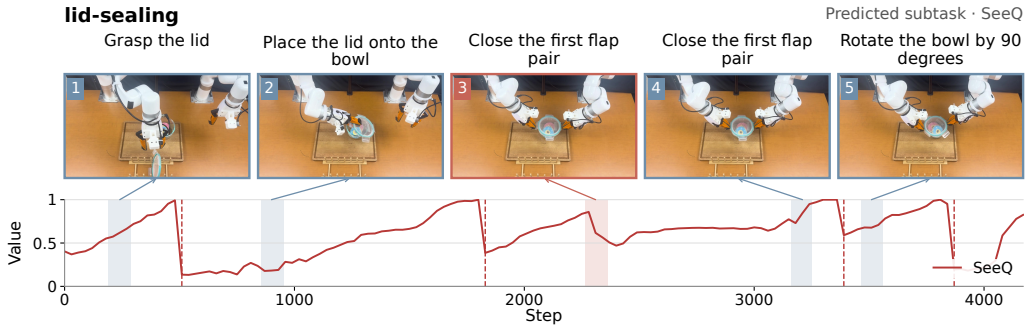


Figure 4: SeeQ value predictions during a lid-sealing evaluation. The rollout uses the same checkpoint as in Table 1. The curve shows the value of the highest-valued of $N = 8$ candidate action chunks. The value drops at snapshot 3 after a failed flap closure, followed by recovery at snapshot 4. Snapshot labels give predicted subtasks; red vertical dashed lines mark changes in these predictions.

The task demands reasoning about the orientation in which to hold the blocks and how to pull them apart, as well as bimanual coordination for handoffs.

Research questions. Our experiments are designed to answer the following questions:

Q1. Can SeeQ steer the base policy to improve its robustness and performance?
Q2. What kinds of training data improve the quality of the Q-function learned via SeeQ?
Q3. How does SeeQ compare to standard Monte Carlo/TD-trained value functions at the task level and SARSA-trained value functions at the subtask level?
Q4. How critical is generalist pretraining to the success of SeeQ?
Q5. How much does subtask instruction supervision contribute to the efficacy of SeeQ, and how important is decoding the predicted subtask at test time?

Figure 4 shows value predictions during a **lid-sealing** rollout. We also provide several diagnostic visualizations to understand the behavior of SeeQ in the appendix, including comparisons across baselines on downstream and RoboCOIN trajectories in Appendix A.2.2.

4.1 Main Results: Policy Steering on Real-World Long-Horizon Tasks

We report success rates for the base policy and the same policy steered by a SeeQ Q-function in Table 1, selecting among $N = 8$ action chunks sampled at each step. We study policy steering under two groups of settings, with datasets that vary in coverage and suboptimality.

Task	Data	Base	SeeQ
shirt-hang	RaC	10/24	**22/24**
lid-sealing	RaC	10/24	**15/24**
grocery-packing	Teleop	9/24	**17/24**
LEGO-disassembly	Teleop	5/24	**10/24**

Table 1: **Policy steering results.** Success rates over 24 trials for the base policy and for the same policy steered by SeeQ with best-of-N action selection ($N = 8$).

RaC (Hu et al., 2025) data. For **shirt-hang** and **lid-sealing**, data is collected with RaC (Hu et al., 2025): the policy runs autonomously until it begins to make a mistake, at which point a human teleoperator takes control to demonstrate corrections and recoveries. These trajectories therefore combine suboptimal policy actions with expert human interventions. When training both the policy and the Q-function on this data, SeeQ raises the success rate of **shirt-hang** from 10/24 to 22/24 and of **lid-sealing** from 10/24 to 15/24.

Teleoperation data. For **grocery-packing** and **LEGO-disassembly**, downstream data consists solely of expert demonstrations, collected via human teleoperation. On **grocery-packing**, SEEQ improves the success rate of the base policy from 9/24 to 17/24, indicating that steering remains beneficial with expert demonstrations alone. The **LEGO-disassembly** dataset is particularly limited: it contains only 97 episodes, all starting from the same block configuration (shape and orientation), resulting in narrow state-space coverage. Even in this setting, SEEQ improves the policy success rate from 5/24 to 10/24.

4.2 COMPARING SEEQ TO OTHER VALUE LEARNING OBJECTIVES

Next, we compare SEEQ against baseline and prior approaches for fitting value functions: task-level Monte Carlo return fitting, subtask-level SARSA, and task-level TD learning. We evaluate these methods on **shirt-hang** and **grocery-packing**. All methods use the same PaliGemma backbone, are pretrained on RoboCOIN, and are finetuned on identical task-specific data. They differ only in their return target, backup action, and use of subtask prediction and conditioning. The task-level ablations condition only on the full-task instruction and use neither a next-token prediction (NTP) loss nor subtask conditioning; subtask-level SARSA and SEEQ use both.

Task-level MC. This baseline tests whether direct return regression, the simplest value-learning approach, suffices for policy steering. Following prior work on state-based return regression (Fei et al., 2026; Intelligence et al., 2025a), we fit full-task action values to G_t^l, the return-to-go defined in Section 2, with a squared-error loss and no bootstrapping.

Objective	shirt-hang	grocery-packing
Base policy	10/24	9/24
Task-level MC	9/24	10/24
Task-level TD	12/24	9/24
Subtask-level SARSA	13/24	10/24
SEEQ (Ours)	**22/24**	**17/24**

Table 2: **Comparison of value-learning objectives.** Success rates on **shirt-hang** and **grocery-packing**. Fitting values of the data collection policy underperforms SEEQ, while task-level MC and TD offer little improvement over the base policy.

Subtask-level SARSA. This baseline tests whether maximization in the TD-BoN backup improves steering beyond evaluation of the behavior policy. It retains SEEQ's subtask rewards and boundary termination, but replaces the backup action in Equation 5 with the next recorded action chunk, as defined for SARSA in Section 2.

Task-level TD learning. This baseline tests the benefit of the subtask formulation by applying TD-BoN at the full-task level. It uses the full-task TD loss and best-of-N backup from Equations 1 and 2. Backups cross subtask boundaries and terminate only at task completion; all TD variants accumulate discounted rewards over the backup interval.

Results. Table 2 shows that SEEQ outperforms all evaluated baselines on both tasks. Task-level MC and TD learning offer little improvement over the base policy, while subtask-level SARSA provides a modest gain on **shirt-hang** and a one-trial gain on **grocery-packing**. These results suggest that both the subtask formulation and the best-of-N backup contribute to effective policy steering, with gains on both mixed policy and human intervention data and purely expert demonstrations.

4.3 IMPORTANCE OF PRETRAINING AND VLM INITIALIZATION FOR SEEQ

To quantify the benefits of initializing SEEQ from a general VLM backbone and pretraining on broad robot data, we evaluate two baselines. First, we compare against a task-specific value function trained on target-task data alone. For a fair comparison, this baseline uses the same TD-learning objective and best-of-N policy extraction with the same base policy. It also predicts the active subtask as a categorical token and conditions its value prediction on this token. Its architecture consists of a pretrained ResNet-50 image encoder followed by an MLP, following the design in Kumar et al. (2022b). Next, we evaluate SEEQ without robot data pretraining. This ablation retains SEEQ's architecture and training objective but skips the robot data pretraining stage, finetuning the pretrained PaliGemma backbone directly on the target task, **shirt-hang**. We observe in Table 3 that with the same VLM backbone and SEEQ objective, general robot data pretraining raises success from 8/24 to 22/24. In addition, SEEQ outperforms a task-specific value function, indicating that both pretraining and VLM initialization are critical.

Value function	Robot pretraining	VLM initialization	Success rate
Base policy	—	—	10/24
Task-specific (ResNet-50 + MLP, TD-BoN)	✗	✗	16/24
SEEQ minus pretraining	✗	✓	8/24
SEEQ (Ours)	✓	✓	**22/24**

Table 3: **Importance of pretraining and VLM initializations for SEEQ, evaluated on shirt-hang.** Columns indicate whether the value function uses robot data pretraining and VLM initialization. Observe that both robot data pretraining and base VLM initialization are important for the success of SEEQ.

4.4 Ablation Study: Effect of Predicting Subtasks and Conditioning on Them

Next, we assess the importance of predicting the active subtask and conditioning value estimation on it for **shirt-hang**. In the first variant, we remove subtask prediction and conditioning, training the model with the subtask-level TD loss alone. In the second variant, we retain the subtask prediction loss but do not condition the value head on the subtask. The subtask is therefore not decoded at inference time and is represented only implicitly in the model's activations.

Variant	Success rate
Base policy	10/24
Subtask-level TD *No subtask prediction, no subtask conditioning*	8/24
Subtask-level TD *Subtask prediction, no subtask conditioning*	15/24
SEEQ (Ours)	**22/24**

Table 4: **Ablation of subtask elicitation on shirt-hang.** Subtask prediction supervision improves success, with further gains from explicitly conditioning value estimation on the predicted subtask.

As shown in Table 4, removing the subtask prediction loss causes the steered policy to underperform the base policy (8/24 vs. 10/24). Adding the prediction loss improves success to 15/24, even without conditioning value predictions on the predicted subtask. Explicitly decoding the subtask and conditioning value estimation on it further improves success to 22/24. Figure 7 illustrates how subtask conditioning aligns value resets with predicted subtask changes.

4.5 Qualitative Analysis of Value-Function Landscapes

VLM pretraining provides regularization in the vision-language space. We hypothesize that SEEQ without robot pretraining performs poorly in Section 4.3 because robot actions come from a distribution unseen during pretraining of the VLM backbone, so direct target-task finetuning may overfit to image features. Pretraining on diverse robot data can alleviate this issue by helping the critic learn action conditioning. To examine this hypothesis, we compare SEEQ, SEEQ without robot pretraining, and the task-specific ResNet-50 critic on six complete held-out **shirt-hang** trajectories. All three critics use subtask-level returns and condition on their own predicted subtask, decoded at every frame. At each frame, they score the same eight cached $\pi_{0.5}$ action chunks, and both gradients are evaluated at that critic's highest-valued candidate. The selected action can differ across critics.

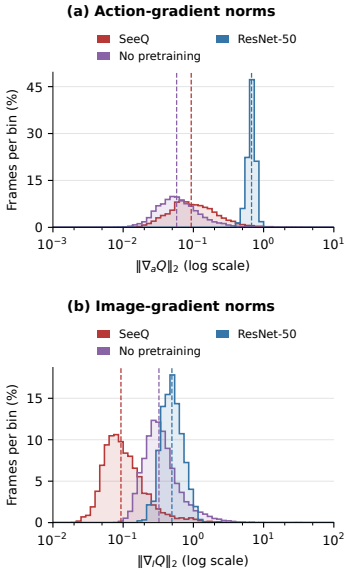


Figure 5: Action and image sensitivity. (a) Action-gradient norms. (b) Joint image-gradient norms across three cameras. Dashed lines denote medians.

Figure 5 compares local sensitivity to actions and images. No pretraining denotes SEEQ finetuned directly from PaliGemma. Panel (a) shows $\|\nabla_{\mathbf{a}} Q\|_2$ in normalized 60×14 action coordinates. Panel (b) shows $\|\nabla_I Q\|_2$, with I concatenating all three 224×224 RGB camera inputs in the network's $[-1, 1]$ coordinates. Both histograms pool all 12,582 frames.

Results. We observe that SEEQ attains a median image-gradient norm of 0.093, compared with 0.323 without robot pretraining and 0.494 for ResNet-50. The $3.5\times$ reduction rel-

ative to direct VLM finetuning is consistent with our hypothesis that robot pretraining reduces sensitivity to image features. Its median action-gradient norm is 0.093, compared with 0.673 for ResNet-50 ($7.2\times$ lower). However, the no-pretraining critic has an even smaller action-gradient median of 0.058 despite worse steering performance (8/24), suggesting that it may be insufficiently discriminative between candidate actions and may not use the action input meaningfully.

5 Related Work

Recent robot learning has increasingly followed the recipe of scaling data and model capacity, with substantial success (Zitkovich et al., 2023; Kim et al., 2024; O'Neill et al., 2024; Intelligence et al., 2025b). However, pure imitation learning struggles to learn from diverse, potentially suboptimal data (Kumar et al., 2022a) and remains brittle on long-horizon tasks, where errors compound over time (Ross & Bagnell, 2010; Belkhale et al., 2023). To address these limitations, prior work has explored training value functions with RL (Kumar et al., 2022b; Bhateja et al., 2023) and using them to guide (Nakamoto et al., 2025) or improve policies (Mark et al., 2025). However, these methods typically instantiate value functions with small networks and use little pretraining data (Bhateja et al., 2023; Kumar et al., 2022b; Yang et al., 2024; Nakamoto et al., 2025), if any (Mark et al., 2025). Thus, the value functions do not inherit the broad pretraining that makes generalist policies useful, and hence, require task-specific demonstrations before they can be deployed.

A recent line of work has sought to scale value-function learning in robotics in the same spirit as generalist imitation learning (Springenberg et al., 2024; Chebotar et al., 2023; Ma et al., 2023; Lee et al., 2026; Kim et al., 2026). For example, Springenberg et al. (2024) and Chebotar et al. (2023) train large transformer backbones from scratch with temporal-difference (TD) learning, but do not leverage modern large-scale vision-language pretraining. In contrast, Lee et al. (2026) finetune a Qwen-3-VL model (Bai et al., 2025) on broad robot data to obtain a VLM-scale reward model. Related efforts learn purely state-based value or progress estimates: WCM (Fei et al., 2026) regresses Monte Carlo (MC) returns, RynnValue (Huang et al., 2026) predicts observed time-to-completion, and Robometer (Liang et al., 2026) combines frame-level progress and success supervision with trajectory preferences. These predictions are trained using trajectory-derived targets rather than TD backups. Such approaches address the challenge of fitting value or reward models at large parametric scale, but long-horizon value learning remains difficult. Direct supervision with MC returns (Kim et al., 2026; Fei et al., 2026) or observed progress avoids TD credit assignment, but ties the targets to the behaviors and coverage of the training data. Conversely, methods that rely on TD learning (Wang et al., 2026) must propagate sparse success signals over long horizons. Our work addresses this tradeoff by training an action-value function with TD over shorter-horizon subtasks, supporting test-time steering against the learned critic. We further show that explicitly predicting the subtask in text improves value estimation, as language can supervise intermediate features for assessing the active subtask at a given state.

Prior work has also explored similar styles of automatic subtask decomposition, but primarily for policy learning rather than value learning (Intelligence et al., 2025b; Zhang et al., 2024; Ahn et al., 2022). For example, Intelligence et al. (2025b) train a high-level VLM to predict per-step subtask instructions for a low-level controller, while Zhang et al. (2024) discover visual subgoals from phase shifts in a pretrained representation and use them to condition imitation policies and shape rewards.

6 Conclusion

We presented SEEQ, an approach for training generalist Q-functions that steer robot policies on long-horizon tasks. By predicting values for the active subtask, SEEQ shortens the credit-assignment horizon while retaining TD learning and best-of-N backups for policy improvement. Explicitly predicting the subtask in language allows the value function to use this structure without requiring subtask annotations at deployment. Across four real-world bimanual manipulation tasks, SEEQ raises the average success rate from 35.4% to 66.7% ($1.88\times$), with improvements on various data compositions. Our ablations highlight the importance of robot data pretraining, the best-of-N backup, and explicit subtask conditioning. These findings suggest that choosing an appropriate prediction horizon and using language to structure value estimation are useful ingredients for generalist value learning. We discuss limitations and future work in Appendix A.4.

AI USE STATEMENT

We used LLM coding assistants to help implement our ideas in code. All code was verified before being committed and pushed to the code release, including the anonymized version referenced in this paper. We also used LLMs to help format tables and figures and check the manuscript for writing errors. No AI was used for idea generation. Tasks involving the formulation of mathematical claims and the development or writing of proofs are not applicable to this work. We take responsibility for the final content of this work, including text, claims, and artifacts produced with AI assistance.

REPRODUCIBILITY STATEMENT

We open-source our code in an anonymous repository. The base SEEQ checkpoint, pretrained on RoboCOIN, is available on Hugging Face.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Anonymous. Building exploratory vision-language-action models via midtraining, 2026. Manuscript in preparation.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Hydra: Hybrid robot actions for imitation learning. In *Conference on Robot Learning*, pp. 2113–2133. PMLR, 2023.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, Thomas Unterthiner, Daniel Keysers, Skanda Koppula, Fangyu Liu, Adam Grycner, Alexey Gritsenko, Neil Houlsby, Manoj Kumar, Keran Rong, Julian Eisenschlos, Rishabh Kabra, Matthias Bauer, Matko Bošnjak, Xi Chen, Matthias Minderer, Paul Voigtlaender, Ioana Bica, Ivana Balazevic, Joan Puigcerver, Pinelopi Papalampidi, Olivier Henaff, Xi Xiong, Radu Soricut, Jeremiah Harmsen, and Xiaohua Zhai. Paligemma: A versatile 3b vlm for transfer, 2024. URL https://arxiv.org/abs/2407.07726.

Chethan Bhateja, Derek Guo, Dibya Ghosh, Anikait Singh, Manan Tomar, Quan Vuong, Yevgen Chebotar, Sergey Levine, and Aviral Kumar. Robotic offline RL from internet videos via value-function pre-training. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023. URL https://openreview.net/forum?id=Rfc9zK6PNO.

Yevgen Chebotar, Quan Vuong, Karol Hausman, Fei Xia, Yao Lu, Alex Irpan, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Conference on Robot Learning*, pp. 3909–3928. PMLR, 2023.

Senyu Fei, Xiaopeng Yu, Siyin Wang, Xianzhong Zhao, Jingjing Gong, and Xipeng Qiu. WCM: A world critic model for vision-language-action reinforcement learning, 2026. URL https://arxiv.org/abs/2607.29613.

Yuchun Feng, Jinliang Zheng, Zhihao Wang, Dongxiu Liu, Jianxiong Li, Jiangmiao Pang, Tai Wang, and Xianyuan Zhan. Demystifying action space design for robotic manipulation policies, 2026. URL https://arxiv.org/abs/2602.23408.

Zheyuan Hu, Robyn Wu, Naveen Enock, Jasmine Li, Riya Kadakia, Zackory Erickson, and Aviral Kumar. Rac: Robot learning for long-horizon tasks by scaling recovery and correction, 2025. URL https://arxiv.org/abs/2509.07953.

Dongchi Huang, Hongyin Zhang, Bohan Hou, Siteng Huang, Zhian Su, Hang Guo, Tong Lu, Zhaofeng Xu, Jiahao Tang, Jianfei Yang, Donglin Wang, Peixi Peng, Mingxiu Chen, Deli Zhao, and Xin Li. RynnValue: Scaling robotic value foundation models with temporal distance, 2026. URL https://arxiv.org/abs/2608.09853.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025a. URL https://arxiv.org/abs/2511.14759.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025b. URL https://arxiv.org/abs/2504.16054.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, et al. Openvla: An open-source vision-language-action model. In *8th Annual Conference on Robot Learning*, 2024.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=wPEIStHxYH.

Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. Should i run offline reinforcement learning or behavioral cloning? In *International conference on learning representations*, 2022a.

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

Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. Roboreward: General-purpose vision-language reward models for robotics, 2026. URL https://arxiv.org/abs/2601.00675.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *Advances in Neural Information Processing Systems*, 2025a. URL https://arxiv.org/abs/2507.07969.

Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model, 2025b. URL https://arxiv.org/abs/2503.00200.

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S. Huang, Luke Zettlemoyer, Dieter Fox, Yu Xiang, Anqi Li, Andreea Bobu, Abhishek Gupta, Stephen Tu, Erdem Biyik, and Jesse Zhang. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. In *Robotics: Science and Systems*, 2026. URL https://arxiv.org/abs/2603.02115.

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. In *The Eleventh International Conference on Learning Representations*, 2023.

Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. Policy-agnostic rl: Offline rl and online rl fine-tuning of any class and backbone. In *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. In *Conference on Robot Learning*, pp. 4996–5013. PMLR, 2025.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable. *Advances in Neural Information Processing Systems*, 38:8350–8389, 2025.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 661–668. JMLR Workshop and Conference Proceedings, 2010.

Jost Tobias Springenberg, Abbas Abdolmaleki, Jingwei Zhang, Oliver Groth, Michael Bloesch, Thomas Lampe, Philemon Brakel, Sarah Maria Elisabeth Bechtle, Steven Kapturowski, Roland Hafner, et al. Offline actor-critic reinforcement learning scales to large models. In *International Conference on Machine Learning*, pp. 46323–46350. PMLR, 2024.

Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, pp. 212–223. Springer, 2002.

Yi Wang, Xinchen Li, Pengwei Xie, Pu Yang, Buqing Nie, Yunuo Cai, Qinglin Zhang, Chendi Qu, Jeffrey Wu, Jianheng Song, Xinlin Ren, Jingshun Huang, Mingjie Pan, Siyuan Feng, Zhi Chen, and Jianlan Luo. Learning while deploying: Fleet-scale reinforcement learning for generalist robot policies, 2026. URL https://arxiv.org/abs/2605.00416.

Shihan Wu, Xuecheng Liu, Shaoxuan Xie, Pengwei Wang, Xinghang Li, Bowen Yang, Zhe Li, Kai Zhu, Hongyu Wu, Yiheng Liu, Zhaoye Long, Runtian Xu, Yue Wang, Chong Liu, Dihan Wang, Ziqiang Ni, Xiang Yang, You Liu, Ruoxuan Feng, Lei Zhang, Denghang Huang, Chenghao Jin, Anlan Yin, Xinlong Wang, Zhenguo Sun, Junkai Zhao, Mengfei Du, Mingyu Cao, Xiansheng Chen, Hongyang Cheng, Xiaojie Zhang, Yankai Fu, Ning Chen, Cheng Chi, Sixiang Chen, Huaihai Lyu, Xiaoshuai Hao, Yequan Wang, Bo Lei, Dong Liu, Xi Yang, Yance Jiao, Tengfei Pan, Yunyan Zhang, Songjing Wang, Ziqian Zhang, Xu Liu, Ji Zhang, Caowei Meng, Zhizheng Zhang, Jiyang Gao, Song Wang, Xiaokun Leng, Zhiqiang Xie, Zhenzhen Zhou, Peng Huang, Wu Yang, Yandong Guo, Yichao Zhu, Suibing Zheng, Hao Cheng, Xinmin Ding, Yang Yue, Huanqian Wang, Chi Chen, Jingrui Pang, YuXi Qian, Haoran Geng, Lianli Gao, Haiyuan Li, Bin Fang, Gao Huang, Yaodong Yang, Hao Dong, He Wang, Hang Zhao, Yadong Mu, Di Hu, Hao Zhao, Tiejun Huang, Shanghang Zhang, Yonghua Lin, Zhongyuan Wang, and Guocai Yao. Robocoin: An open-sourced bimanual robotic data collection for integrated manipulation, 2026. URL https://arxiv.org/abs/2511.17441.

Jingyun Yang, Max Sobol Mark, Brandon Vu, Archit Sharma, Jeannette Bohg, and Chelsea Finn. Robot fine-tuning made easy: Pre-training rewards and policies for autonomous real-world reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4804–4811. IEEE, 2024.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination?, 2026. URL https://arxiv.org/abs/2603.16666.

Zichen Zhang, Yunshuang Li, Osbert Bastani, Abhishek Gupta, Dinesh Jayaraman, Yecheng Jason Ma, and Luca Weihs. Universal visual decomposer: Long-horizon manipulation made easy. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6973–6980. IEEE, 2024.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

A Appendices

A.1 Additional Implementation Details

A.1.1 Action Space and Normalization

Value-function pretraining and finetuning use a shared 14-dimensional bimanual end-effector (EEF) action space. The base policies use the same representation except on **LEGO-disassembly** (YAM platform), where the policy operates in joint space. For each of the two arms, an EEF action contains translational deltas $(\Delta x, \Delta y, \Delta z)$, roll–pitch–yaw orientation deltas, and a gripper command, giving the layout

$$\big[\underbrace{\Delta x, \Delta y, \Delta z}_{\text{left pos}},\ \underbrace{\Delta\phi, \Delta\theta, \Delta\psi}_{\text{left rot}},\ g_{\text{left}},\ \underbrace{\Delta x, \Delta y, \Delta z}_{\text{right pos}},\ \underbrace{\Delta\phi, \Delta\theta, \Delta\psi}_{\text{right rot}},\ g_{\text{right}}\big].$$

We use the chunk-wise delta parameterization of Feng et al. (2026): every action in a chunk is expressed relative to the current state rather than recursively relative to the previous predicted action, with orientation deltas composed as relative rotations on the rpy slots. Deltas are applied to all non-gripper dimensions; the two gripper channels are absolute. The proprioceptive state $\mathbf{s}_t^p$ uses the same 14-D EEF layout.

For **LEGO-disassembly**, demonstrations are collected using a bimanual YAM station and recorded as six joint-angle targets and a gripper command per arm, matching the joint-position interface used by our teleoperation stack. The base policy is trained to predict joint-angle deltas relative to the current joint configuration, with absolute gripper commands. To retain the EEF representation used during SEEQ pretraining, we apply a computationally inexpensive forward-kinematics (FK) transform to the recorded joint targets and current joint configuration, then express the resulting target poses as EEF deltas relative to the current pose for critic training. At inference, the same conversion maps each joint-space policy candidate into EEF space for critic scoring; the robot executes the selected candidate in joint space.

Both state and action chunks are normalized with quantile normalization: each dimension is mapped via its $1^{\text{st}}/99^{\text{th}}$ percentiles, $\tilde{x} = 2\,\frac{x - q_{01}}{q_{99} - q_{01}} - 1$, and then clipped to $[-1.25,\ 1.25]$. The same normalize-then-clip scheme is applied to states, action chunks and the action candidates used in the TD backup. Quantile normalization and clipping are common to value-function pretraining, finetuning, and base-policy training, with statistics computed in the representation used by each model. For **LEGO-disassembly**, we undo the policy's normalization and recover absolute joint targets before applying FK; the resulting EEF deltas are then normalized and clipped using the critic's statistics before scoring.

A.1.2 Handling Mixed Control Frequencies in the RoboCOIN Dataset

Our pretraining dataset, RoboCOIN, aggregates demonstrations collected at two different control rates: part of the data is recorded at 30 fps and part at 50 fps. Rather than resampling the trajectories, the dataloader places both rates on a common wall-clock timeline, so that a single discount factor, a single TD horizon, and a single action-chunk length all correspond to the same real-time duration regardless of the source rate. In effect, the per-frame discount is applied per unit of time rather than per frame, and the H-step lookahead used to form the bootstrap target is expressed as a fixed time window (e.g. 1 s) that spans proportionally more frames in the 50 fps data than in the 30 fps data.

This rate normalization is what keeps the value targets consistent across the two sources. The MC return to the end of the active subtask, the H-step reward, and the termination flag are all computed on this normalized timeline: the subtask-completion reward fires, and the transition is marked terminal, when the active subtask ends within the TD window, with the reward discounted to the subtask boundary; otherwise the reward is zero and the value bootstraps from the next state using a discount matched to the same time window. Because a fixed-length action chunk covers more wall-clock time at a lower frame rate, the 30 fps trajectories additionally use only the leading portion of each chunk (the trailing slots are masked), so the effective action horizon again matches across the two rates.

Algorithm 1 Training SEEQ

Require: subtask-annotated dataset $\mathcal{D}$; base policy π_{base}; discount γ; learning rate η; target EMA rate τ; chunk duration T (wall-clock time spanned by one action chunk); action horizon $H = \text{round}(f_{\text{fps}}\, T)$ (chunk length at the recording frame rate); reference control rate $c = 150\,\text{Hz}$ (Appendix A.1.2); subtask-loss weight $\lambda_{\text{subtask}} = 0.1$; backup width $N = 8$

Initialize θ from PaliGemma for pretraining, or from pretrained SEEQ for finetuning
Initialize target parameters $\bar{\theta} \leftarrow \theta$
for each training step **do**
  Sample a minibatch $\mathcal{B}$ of annotated transitions $(\mathbf{s}_t, \mathbf{a}_{t:t+H-1}, r_{\tilde{l}_t}, \mathbf{s}_{t+H}, l, \tilde{l}_t, \tilde{l}_{t+H})$
  For each transition, retrieve N cached chunks $\mathbf{a}^{(i)}_{t+H:t+2H-1} \sim \pi_{\text{base}}(\cdot \mid \mathbf{s}_{t+H}, l)$, $i = 1, \ldots, N$
  $\mathbf{a}'_{t+H:t+2H-1} \leftarrow \arg\max_{\mathbf{a} \in \{\mathbf{a}^{(i)}\}_{i=1}^{N}} Q_{\bar{\theta}}(\mathbf{s}_{t+H}, \mathbf{a}, l, \tilde{l}_{t+H})$
  $y_t \leftarrow r_{\tilde{l}_t}(\mathbf{s}_t) + \gamma^{c\,T}\,(1 - I_t)\, Q_{\bar{\theta}}(\mathbf{s}_{t+H}, \mathbf{a}'_{t+H:t+2H-1}, l, \tilde{l}_{t+H})$ {I_t: subtask boundary in $t : t + H - 1$}
  $\mathcal{L}_{\text{TD}} \leftarrow \mathbb{E}_{\mathcal{B}}\big[\big(Q_\theta(\mathbf{s}_t, \mathbf{a}_{t:t+H-1}, l, \tilde{l}_t) - \text{sg}(y_t)\big)^2\big]$ {sg: stop-gradient}
  Compute $\mathcal{L}_{\text{subtask}}$ by teacher forcing $\tilde{l}_t$ (Eq. 4), averaging over tokens within each example and then over $\mathcal{B}$
  $\mathcal{L} \leftarrow \mathcal{L}_{\text{TD}} + \lambda_{\text{subtask}} \mathcal{L}_{\text{subtask}}$
  Update θ with AdamW using $\nabla_\theta \mathcal{L}$ and learning rate η
  $\bar{\theta} \leftarrow (1 - \tau)\,\bar{\theta} + \tau\,\theta$
end for

A.1.3 TRAINING AND INFERENCE ALGORITHM DETAILS

Algorithm 1 summarizes the SEEQ training procedure used for both pretraining and finetuning, following Section 3. The algorithm's γ is defined at the reference rate $c = 150$ Hz. At recording rate f, the per-frame discount therefore corresponds to $\gamma^{c/f}$, and a T-second backup uses γ^{cT}. The task-level methods use a discount factor of 0.9995, compared with 0.999 for the subtask-level methods, to accommodate the longer return horizon. The model predicts the active subtask autoregressively from the visual observation and task instruction, then evaluates the action chunk conditioned on that subtask. During training, ground-truth subtask tokens provide teacher-forced supervision for the next-token loss and remain visible to the action tokens and the value token. Causal attention within the subtask sequence ensures that each token is predicted from the observation, task instruction, and preceding subtask tokens.

The rewards and discounts in the TD targets follow the control-rate normalization in Appendix A.1.2. In the algorithm, $r_{\tilde{l}_t}(\mathbf{s}_t)$ denotes the discounted subtask-completion reward within the TD window, and bootstrapping stops when the active subtask terminates within that window. Both current and target Q-values are conditioned on the corresponding ground-truth subtask during training. Base-policy candidates for the TD backups are sampled and cached offline.

At inference, given $(\mathbf{s}_t, l)$, SEEQ first autoregressively decodes the active subtask $\hat{l}_t$. The base policy proposes $N = 8$ action chunks, and the critic scores each candidate as $Q_\theta(\mathbf{s}_t, \mathbf{a}^{(i)}, l, \hat{l}_t)$. The robot executes the highest-scoring candidate, using the predicted subtask in place of the ground-truth annotation supplied during training. No external subtask annotations or manual subtask switching are required at inference.

Value overestimation. We monitor the difference between predicted Q-values for dataset action chunks and their recorded MC returns during RoboCOIN pretraining, using subtask returns for SEEQ and task returns for task-level TD-BoN. The expected $Q - \text{MC}$ gap upper-bounds mean signed overestimation relative to the optimal value for the corresponding reward: an optimal continuation achieves at least the expected return of the recorded behavior. A positive gap can therefore reflect both estimation error and improvement over the recorded continuation. This interpretation applies in expectation on dataset actions. Figure 6(a) shows this gap for SEEQ over the full pretraining run.

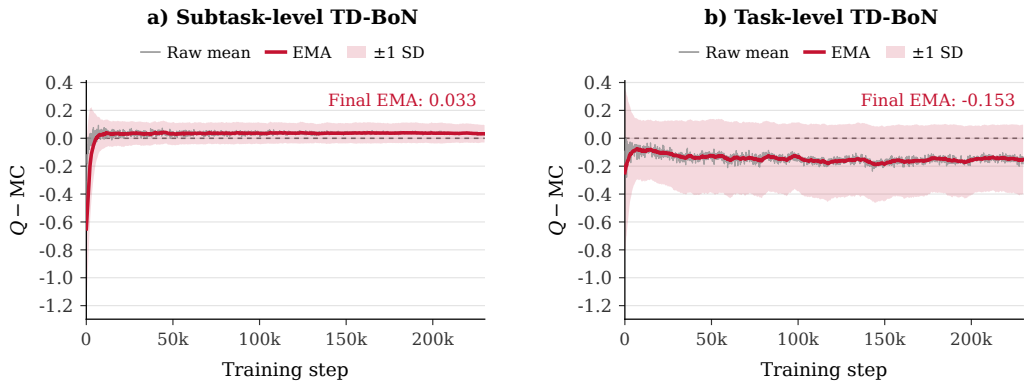


Figure 6: **Q-values relative to recorded returns during RoboCOIN pretraining.** Gray: batch-mean $Q - \mathrm{MC}$; red: EMA with a 1,000-step half-life; shading: ± 1 standard deviation of the exponentially weighted prediction-minus-return differences. (a) SEEQ with subtask returns shows a near-zero gap after initial underestimation. (b) Task-level TD-BoN persistently underestimates task returns.

A.1.4 SEEQ ARCHITECTURE

Figure 2 summarizes the PaliGemma-based architecture of SEEQ. SigLIP encodes the camera images into patch tokens; Gemma processes these with task and subtask text, linearly projected action tokens, and a learned value token at the end of the sequence. Image and task tokens attend bidirectionally. Subtask tokens use a causal mask: they attend to the images, task instruction, and preceding subtask text, but not to later actions or the value token. Action tokens attend bidirectionally within the candidate chunk and can attend to all preceding image and language tokens, including the subtask.

The final value token attends to all valid preceding tokens, combining the observed state, language context, and candidate actions; a linear readout maps its final representation to a scalar Q-value. Earlier tokens cannot attend to the value token. Padding and unused action positions are masked. We do not use proprioceptive state as an input to any of the value functions trained. During training, the model receives ground-truth subtask tokens; at inference, it first decodes the subtask and then scores candidate action chunks.

A.1.5 EVALUATION PROTOCOL

For each of the four tasks, we evaluate the models being compared over 24 trials. Randomization is consistent across models: trial i uses the same physical setup and task instruction for every model evaluated on that task. Figure 3 illustrates the four tasks using top-camera snapshots from dataset demonstrations. In all evaluations, each action chunk spans 1 s, and actions are replanned every 0.5 s. For evaluations that infer the active subtask and condition value predictions on it, the subtask is decoded every four replanning calls (2 s).

1. **shirt-hang**. We randomize the shirt's position and tilt across trials. The hanger's position also varies, and the robot must insert the hanger into both sleeves of the shirt and hang it on the rod.
2. **lid-sealing**. We randomize the positions of the container and the lid on the dish rack. The robot must pick up the lid and align it with the container. Successful sealing requires closing all four latching flaps.
3. **grocery-packing**. Each trial uses two of three box types, with eight trials for each of the three possible pairs. Each trial has a different language instruction specifying the set of objects to pack. The exact object sets here are absent from the training distribution, testing generalization to new packing requests.
4. **LEGO-disassembly**. The assembled blocks have the same initial shape and orientation as in training. We randomize the language instruction specifying the mapping from block colors to tray colors. The robot must disassemble the blocks and place them in the trays according to this mapping.

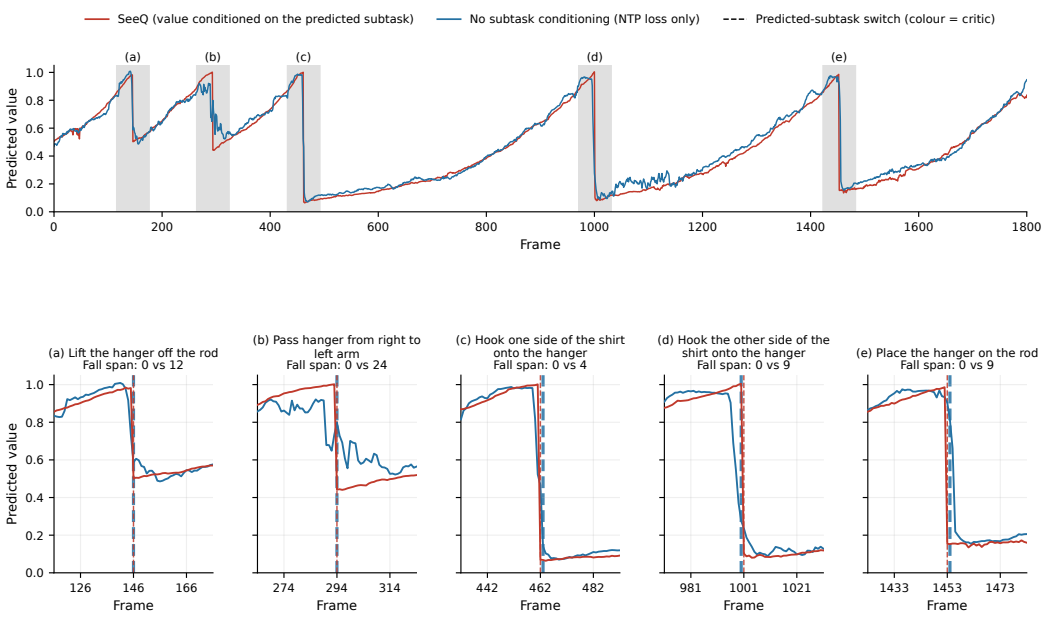


Figure 7: **Subtask conditioning aligns value resets with predicted subtask changes.** Both critics evaluate recorded dataset action chunks. Top: predicted values along a held-out **shirt-hang** trajectory. Bottom: enlarged views of the five shaded regions; colored dashed lines mark each critic's predicted switch nearest the panel center. *Fall span* counts frames between the first 5% and 95% of the local peak-to-trough drop, using the maximum value in the 20 frames before that critic's switch and the minimum from the switch through 20 frames after it. A drop crossing both thresholds in one frame has span 0. Panel titles report SEEQ versus no conditioning.

A.2 Additional Experimental Results

A.2.1 Subtask Prediction and Value Conditioning

The ablation in Section 4.4 separates the benefit of the auxiliary next-token prediction (NTP) loss from that of explicitly conditioning values on the decoded subtask. Skipping subtask decoding is an appealing way to reduce inference latency: joint training might allow latent activations to carry the relevant subtask information implicitly. A similar motivation appears in video–action models such as UVA and Fast-WAM, which retain video supervision while bypassing explicit video generation at action-inference time (Li et al., 2025b; Yuan et al., 2026).

We compare SEEQ with the variant that retains the auxiliary NTP loss but does not condition its value head on subtask tokens. Both use subtask-level TD-BoN and the same NTP weight of 0.1, and are finetuned on **shirt-hang** with the same learning-rate schedule, starting from their corresponding RoboCOIN-pretrained checkpoints.

Figure 7 evaluates both critics on the same held-out **shirt-hang** trajectory, using the same checkpoints as in our evaluations. We also decode the ablation's subtask prediction for this diagnostic, without supplying it to the value head. Although the overall value curves are similar, their behavior near subtask changes differs. For SEEQ, the largest one-frame drop coincides with the predicted subtask switch in all five displayed transitions, and both fall thresholds are crossed in that frame. Without conditioning, the drops span 4–24 frames and can begin before or finish after the switch. These results suggest that, without explicit subtask conditioning, the model learns two highly accurate but largely independent predictions. The lags between subtask switches and value resets indicate that these predictions lack the desired coupling. In SEEQ, explicit conditioning ties the value reset to a change in the model's inferred subtask.

A.2.2 Value Predictions across Baselines

The objectives in Section 4.2 differ both in their backup and in the horizon over which reward must propagate. Figure 6(b) shows persistent underestimation and a broad distribution of $Q - \mathrm{MC}$

differences for task-level TD-BoN during RoboCOIN pretraining. One plausible source of this dispersion is variation in task horizon: with sparse terminal rewards, TD estimates can collapse toward zero on very long tasks while remaining useful on shorter tasks.

Figure 8 compares snapshot-and-value strips on a held-out RoboCOIN **pour-tea** trajectory (panel (a)) and held-out **grocery-packing** and **shirt-hang** trajectories (panels (b) and (c)). All critics evaluate recorded dataset action chunks. The downstream examples qualitatively assess the generalization of the learned value functions. On the RoboCOIN trajectory, task-level TD provides almost no signal for most of the trajectory, while MC predictions are noisy.

Figure 9 compares SEEQ with the no-pretraining and task-specific ResNet-50 baselines from Section 4.3 on a held-out **shirt-hang** trajectory.

A.3 HYPERPARAMETERS

Shared critic settings. Except for the ResNet-50 ablation, all critics use a PaliGemma backbone with 224×224 images from three cameras and `float32` precision. Critics use AdamW with weight decay 10^{-6}, $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, and gradient-norm clipping at 1. TD critics use a target-network EMA rate of $\tau = 0.005$. We use the action representation and quantile normalization described in Appendix A.1.1, with action chunks of length 50 for RoboCOIN pretraining and 60 for downstream training. RoboCOIN critic pretraining uses batch size 256; all downstream critic training uses batch size 128.

Base policies. For each downstream task, we finetune $\pi_{0.5}$ from its released base checkpoint. The same fixed task-specific policy supplies candidates for SEEQ and all critic ablations, and is evaluated directly as the BC baseline. Policies use Adam and the same optimizer coefficients and clipping threshold as the critics. Their cosine learning-rate schedules have peak 5×10^{-5}, 1000 warmup steps, and endpoint 5×10^{-6}. Table 5 distinguishes the number of training steps used for the selected policy from the full LR schedule length. The grocery-packing and LEGO-disassembly policies condition on manually advanced subtask prompts; this is a property of the base policy, separate from the critic's own subtask prediction.

Table 5: **Downstream base-policy hyperparameters.** Training steps refer to the policy checkpoint used in evaluation; LR decay length refers to the full configured cosine schedule.

Task	Training steps	LR decay steps	Batch size
shirt-hang	60k	200k	128
lid-sealing	40k	60k	256
grocery-packing	70k	70k	256
LEGO-disassembly	20k	20k	256

The separate $\pi_{0.5}$ policy used to generate candidate actions for RoboCOIN critic pretraining is trained for 230k steps with batch size 256. Its learning rate warms up for 1000 steps to 10^{-5} and then remains constant.

Critic pretraining. All PaliGemma critics in Table 6 are pretrained on the same 131 RoboCOIN task datasets for 230k steps, using a cosine learning-rate schedule with peak 10^{-5}, 1000 warmup steps, and endpoint 10^{-6}. The auxiliary next-token prediction (NTP) loss has weight 0.1 whenever subtask prediction is enabled, and 0 otherwise. TD-BoN uses 8 policy candidates for its pretraining backup.

Downstream critic finetuning. Each pretrained critic starts from its 230k checkpoint. Finetuning uses a cosine LR from 5×10^{-6} to 5×10^{-7} over 20k steps, without warmup. Table 7 lists the selected checkpoints: 250k corresponds to 20k finetuning steps, while 240k corresponds to 10k finetuning steps within the same 20k LR schedule. For the **shirt-hang** variant with NTP but no subtask conditioning, we select the 240k checkpoint because it performed better than the 250k checkpoint. This differs from the evaluation protocol in the remaining experiments, where we do not perform checkpoint selection. The resulting success rate can therefore be viewed as a favorable upper bound for this ablation; the conclusion that explicit subtask conditioning improves performance still holds. This value function is used for policy-steering comparisons only in Section 4.4. For SEEQ on **LEGO-disassembly**, we use the 240k checkpoint because this task has a much smaller dataset.

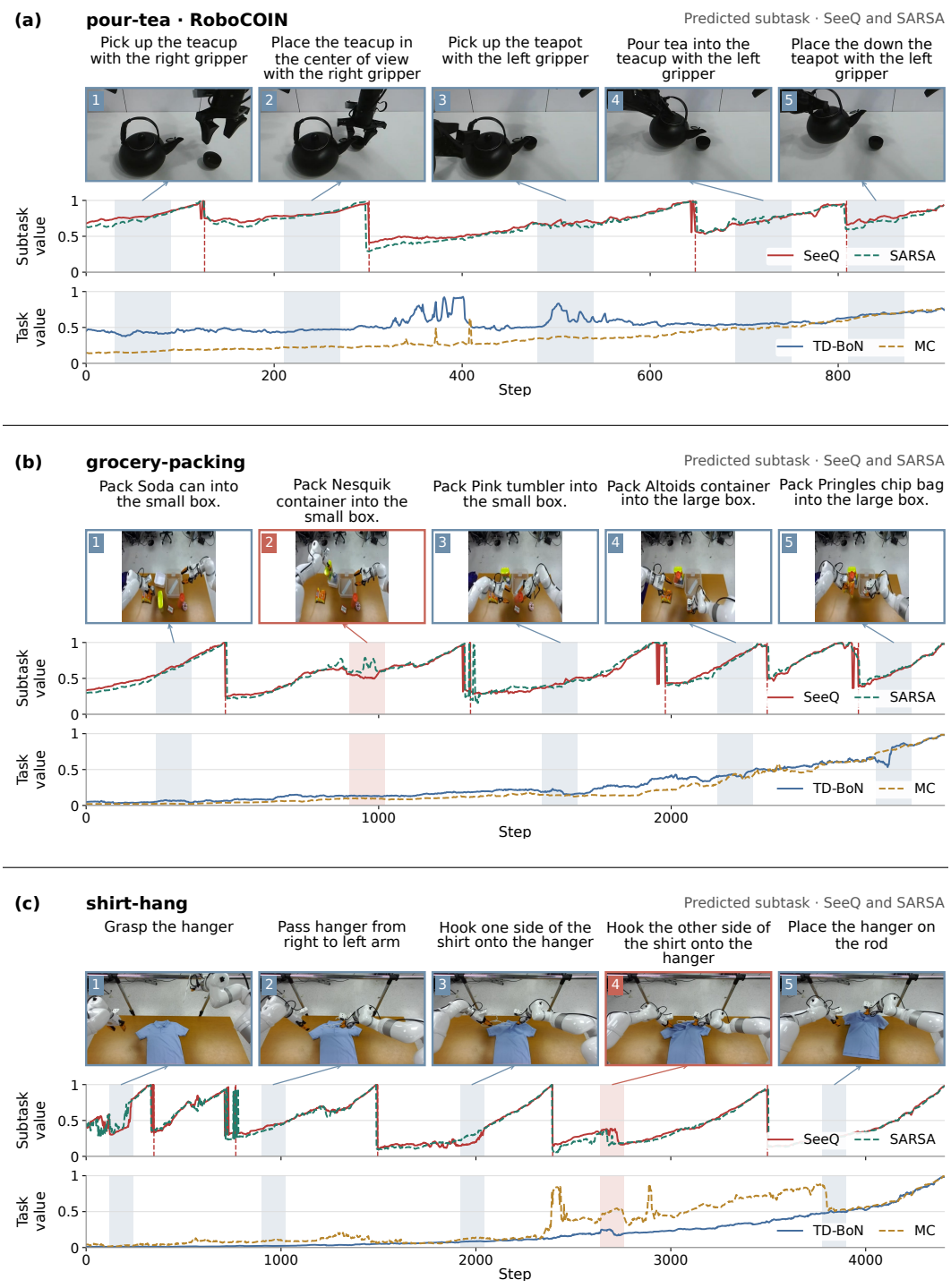


Figure 8: **Comparison of the value functions in Section 4.2.** All critics evaluate recorded dataset action chunks. **(a)** On a held-out RoboCOIN trajectory, task-level TD provides almost no signal for most of the trajectory, consistent with compounding bootstrapping errors, while MC predictions are noisy, potentially reflecting spurious correlations. **(b)** In this held-out episode, the Nesquik container must be reoriented in frame 2 because the soda can obstructs it, causing SEEQ's value to drop. In the final frame, MC predictions are noisy despite smooth progress. **(c)** In this held-out episode, the value jumps in frame 1 as the gripper reorients to grasp the hanger correctly. In frame 4, the left gripper unintentionally releases the collar, causing the value to drop. Frame 5 again shows noisy MC predictions. Solid curves denote SEEQ and TD-BoN; dashed curves denote SARSA and MC. Red vertical dashed lines mark SEEQ-predicted subtask switches and are omitted from the task-level value plots.

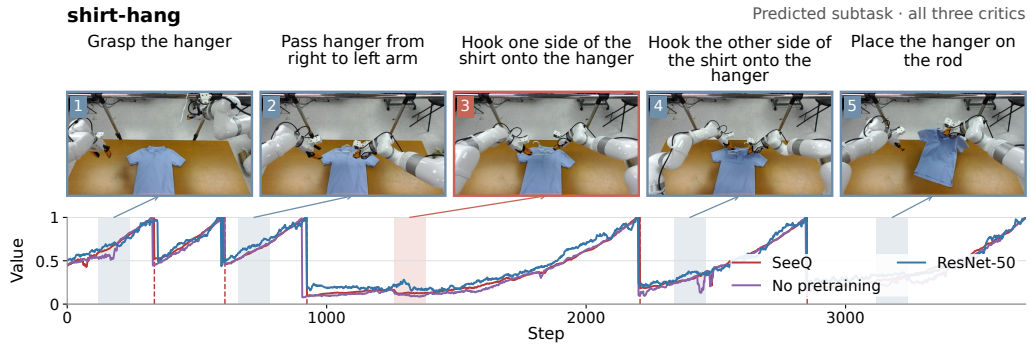


Figure 9: Comparison of the value functions in Section 4.3 on a held-out episode. All critics evaluate recorded dataset action chunks. In frame 3 (red), the right gripper releases the collar, causing the value to drop. The ResNet-50 value peaks about 0.7 s before this release, so the evaluated action chunk includes the failure; this indicates an incorrect value estimate, unlike the VLM critics in this case. SEEQ generally produces smoother predictions than SEEQ without robot pretraining and the task-specific ResNet-50 critic, indicating better generalization.

Table 6: **Critic pretraining variants.** All rows use 230k pretraining steps and the shared schedule above. Conditioning refers to the critic's value prediction, independently of the base-policy prompt.

Critic	Return scope	γ	NTP weight	Subtask conditioning
SEEQ (TD-BoN)	Subtask	0.999	0.1	Yes
SARSA	Subtask	0.999	0.1	Yes
MC	Task	0.9995	0	No
TD-BoN	Task	0.9995	0	No
NTP, no conditioning	Subtask	0.999	0.1	No
Neither NTP nor conditioning	Subtask	0.999	0	No

Critics without robot pretraining. The two task-specific **shirt-hang** ablations are trained directly for 20k steps and evaluated at checkpoint 20k. SEEQ without robot pretraining initializes from PaliGemma and uses a cosine LR from 5×10^{-6} to 5×10^{-7} over 20k steps with 1000 warmup steps. The ResNet-50 + MLP critic initializes its image encoder from ImageNet weights and uses a cosine LR from 10^{-5} to 10^{-6} over 20k steps with 1000 warmup steps. Both use the subtask-level TD-BoN objective with $\gamma = 0.999$, 8 backup candidates, and subtask-prediction loss weight 0.1; the ResNet critic predicts the subtask as a categorical label.

A.4 LIMITATIONS AND FUTURE WORK

SEEQ relies on subtask annotations during training, but decomposing manipulation trajectories into discrete spans can be ambiguous, particularly during recoveries or near subtask boundaries. Incorporating past frames and using finer-grained annotations may partly mitigate this ambiguity, but the appropriate subtask granularity remains task-dependent. At deployment, errors in subtask prediction can cause the critic to evaluate actions against an incorrect objective. Learning decompositions that are useful for value estimation and accounting for uncertainty over the active subtask are promising directions for addressing this limitation. Moreover, optimizing the value of the current subtask favors local progress, which need not align with the overall task. For example, a robot may knock over another object while packing the current one, making subsequent subtasks harder without reducing progress on the active subtask. Combining subtask-level values with estimates of downstream consequences could address this tradeoff while preserving the benefits of shorter-horizon learning.

Finally, our evaluation uses the Q-function to select among action chunks proposed by a fixed base policy, so steering is limited by the quality and diversity of these candidates. Producing more exploratory base policies and using the learned Q-function to directly improve the policy is a natural next step. Our experiments also finetune the generalist value function on each target task; evaluating

Table 7: **Downstream critic checkpoints.** All rows initialize from the corresponding 230k pretrained critic and use a 20k LR decay schedule.

Critic	Downstream task	Finetuning steps	Checkpoint step
SEEQ	**shirt-hang**, **lid-sealing**, **grocery-packing**	20k	250k
SEEQ	**LEGO-disassembly**	10k	240k
Task-level MC	**shirt-hang**, **grocery-packing**	20k	250k
Task-level TD-BoN	**shirt-hang**, **grocery-packing**	20k	250k
Subtask-level SARSA	**shirt-hang**, **grocery-packing**	20k	250k
NTP, no conditioning	**shirt-hang**	10k	240k
Neither NTP nor conditioning	**shirt-hang**	20k	250k

transfer with less downstream data, or without task-specific finetuning, would further clarify the scope of its generalization.

A.5 ROBOCOIN PRETRAINING DATASET

We construct our diverse pretraining dataset from the bimanual-embodiment subset of RoboCOIN (Wu et al., 2026), including humanoid robot data. The combined training and validation splits contain 131 task datasets, 40,054 episodes, and 31,337,006 steps across three embodiments: Agilex Cobot Magic, Agilex Split ALOHA, and Galaxea R1 Lite. This is the filtered set available when we constructed the dataset; subsequent RoboCOIN releases have added more data. Table 8 lists the task datasets included in this snapshot. Figure 10 shows the distribution of subtask durations in RoboCOIN, in seconds. To match the temporal scale of pretraining, we define subtasks in the target tasks with average durations of $\approx$ 5–25 seconds.

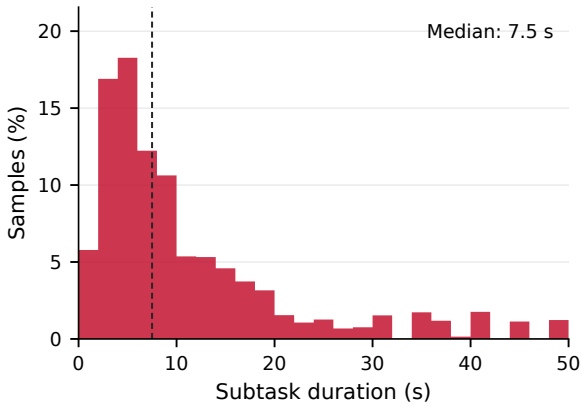


Figure 10: **Histogram of RoboCOIN subtask durations in seconds.**

Table 8: **The** 131 **RoboCOIN task datasets used for pretraining.** Task identifiers are grouped by embodiment, with the embodiment prefix omitted and underscores rendered as hyphens. Released spellings and variant suffixes are retained.

Agilex Cobot Magic (63 tasks)		
box-storage-chopsticks	**cap-the-pen-a**	**catch-the-ball**
classification-of-fruits-and-vegetables	**classification-of-fruits-and-vegetables-a**	**classification-of-tableware**
clean-blackboard	**clean-up-the-tableware**	**clear-the-desktop**
close-book	**close-button**	**cube-reset**
cut-banana	**desktop-organization**	**drawer-storage-mineral-water**
fold-clothes	**fold-the-towel**	**fold-towel-a**
food-packaging	**make-fruit-salad**	**make-hamburger**
mobile-cube	**mobile-cube-blackboard**	**move-beverage**

Table 8 (continued)

move-plate	**move-the-ball**	**move-the-ball-and-the-cube-block**
move-the-ball-interference	**move-the-bread**	**move-the-cup**
move-the-plate	**move-the-small-ball**	**movethe-position-of-the-bluetooth**
open-the-shoebox	**place-square-pyramid**	**place-the-cube-block**
place-the-test-tube	**plate-storage-apple**	**plate-storage-bread**
plate-storaje-baozi	**pot-storage-steamer**	**pour-drink**
pour-water-a	**pour-water-bottle**	**prepare-breakfast**
pull-zipper	**pushing-magnet**	**put-in-the-pear**
put-the-building-block-on-the-table	**steamer-storage-dumpling**	**storage-plate**
take-out-a-pen-from-the-pen-holder	**take-out-the-bread**	**take-the-shoes-off-the-shelf**
the-box-stores-table-tennis-balls	**the-plate-holds-the-fruit**	**the-plate-holds-the-vegetables**
turn-off-the-desk-lamp	**turn-on-the-bulb**	**turn-on-the-desk-lamp**
twist-bottle-cap	**vase-storage-flower**	**water-bottle-storage**
Agilex Split ALOHA (16 tasks)		
basket-storage-banana	**basket-storage-bread**	**basket-storage-egg-yolk-pastry**
basket-storage-long-bread	**basket-storage-orange**	**basket-storage-peach**
fold-the-pants	**plate-storage**	**pour-rice**
pour-tea	**scoop-coffee-beans**	**stack-baskets**
stir-coffee	**wipe-table**	**wipe-the-table**
zip-up-the-document-bag		
Galaxea R1 Lite (52 tasks)		
boil-water-in-a-kettle	**catch-the-water**	**clean-the-floor**
clean-the-sink	**clean-toilet**	**connect-the-router-cable**
cook-a-meal	**cover-the-pot-lid**	**dispose-of-leftover-food**
drawer-storage-hair-dryer	**fold-clothes**	**garbage-disposal**
hang-clothes	**make-a-landline-call**	**make-breakfast**
make-tea	**make-the-bed**	**open-and-close-curtains**
open-and-close-microwave-oven	**open-and-close-nightstand-door**	**open-and-close-nightstand-drawer**
open-and-close-the-freezer-door	**open-the-food-pan**	**opening-and-closing-aalcony-sliding-doors**
pick-up-and-store-items	**place-the-dress-shirt-on-the-hanger**	**plug-the-socket**
pour-water	**put-on-a-garbage-bag**	**put-slippers-into-floor-standing-shoe-cabinet**
put-the-pillow-on-the-bed	**put-the-shoes-into-the-shoe-box**	**put-the-tableware-into-the-cupboard**
sliding-chair	**storage-of-toiletries**	**switch-labels**
switch-on-and-off-the-central-air-conditioning	**tableware-arrangement**	**tableware-cleaning**
take-and-place-the-portable-power-bank	**take-and-put-away-garden-stuff**	**take-and-put-away-garden-stuff-a**
take-and-put-away-items	**take-and-put-the-bowl**	**take-clothes-out-of-the-washing-machine**
take-or-store-plates	**tea-service-table-setting**	**throw-out-the-trash**
tidy-up-toiletries	**wash-the-tableware**	**washing-board**
wipe-the-table		