Offline Policy Evaluation for Manipulation Policies via Discounted Liveness Formulation

Hao Wang[†,‡] Joshua Bowden[‡] Colton Crosby [‡] Somil Bansal [‡]
[†]University of Southern California [‡]Stanford University
{haowwang, joshuabowden, cjcrosby, somil}@stanford.edu

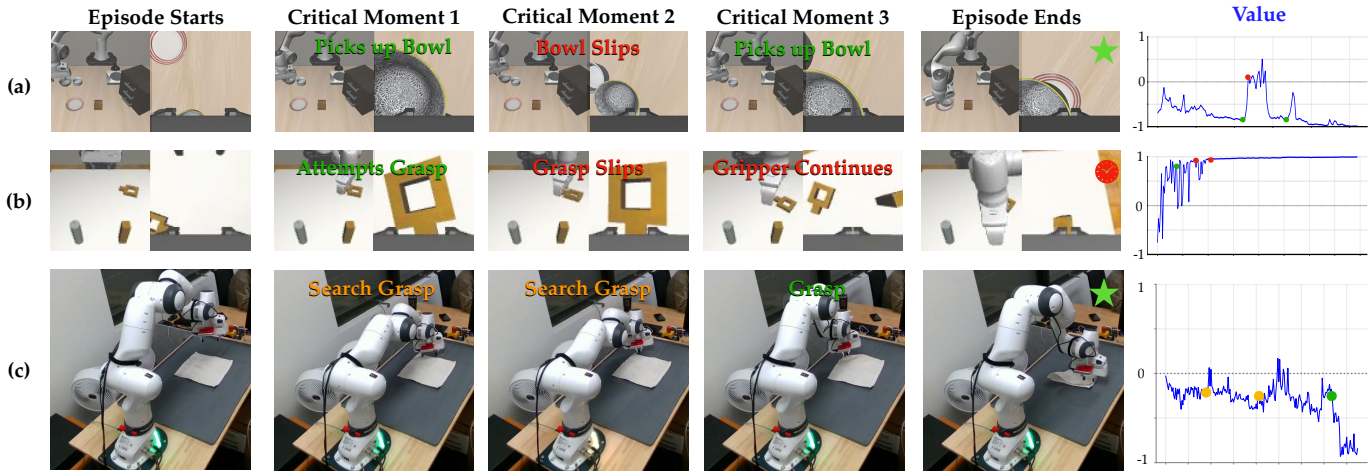

Fig. 1. In this work, we address the problem of offline policy evaluation for manipulation policies under sparse, episode-level rewards, where finite-horizon rollouts and non-monotonic task progression make standard value estimation methods unreliable. Our method formulates policy evaluation as a discounted liveness problem and learns a value function that maps states in the episode to a proxy of task progression. States close to goal completion are assigned low values although the value function is obtained only using sparse, episode-level rewards. We present episodes from three case studies alongside their corresponding value predictions. The base and wrist camera images are displayed side-by-side in the first five columns, while the final column visualizes the value predictions throughout the episode. We label and explain critical moments within the tasks and their associated values. As shown, the value function trends downward to -1 as the task progresses, increases when critical setbacks occur, and decreases again when the policy recovers.

***Abstract*—Policy evaluation is a fundamental component of the development and deployment pipeline for robotic policies. In modern manipulation systems, this problem is particularly challenging: rewards are often sparse, task progression of evaluation rollouts are often non-monotonic as the policies exhibit recovery behaviors, and evaluation rollouts are necessarily of finite length. This finite length introduces truncation bias, breaking the infinite-horizon assumptions underlying standard methods relying on Bellman equations/principle of optimality. In this work, we propose a framework for offline policy evaluation from sparse rewards based on a liveness-based Bellman operator. Our formulation interprets policy evaluation as a task-completion problem and yields a conservative fixed-point value function that is robust to finite-horizon truncation. We analyze the theoretical properties of the proposed operator, including contraction guarantees, and show how it encodes task progression while mitigating truncation bias. We evaluate our method on two simulated manipulation tasks using both a Vision–Language–Action model and a diffusion policy, and a cloth folding task using human demonstrations. Empirical results demonstrate that our approach more accurately reflects task progress and substantially reduces truncation bias, outperforming classical baselines such as TD(0) and Monte Carlo policy evaluation. The full implementation and experiments are available on the project website [1].**

I. Introduction

Recent years have seen rapid progress in autonomous robotic systems, driven in large part by generalist policies powered by high-capacity models and large-scale datasets [6, 31, 12, 25, 4, 28, 13, 29]. As these policies become increasingly capable and are deployed across a growing range of real-world manipulation tasks, reliable policy evaluation has emerged as a critical bottleneck. Policy evaluation plays a central role throughout the development lifecycle: it is used to iteratively improve policies [11], assess readiness for deployment [26], and guide policy steering and refinement [17].

Fundamentally, the problem of policy evaluation centers on the question of *how good is a given policy in completing its task?* While a large number of frameworks and procedures are developed to study this problem, Reinforcement Learning (RL) [24] remains one of the most enduringly popular frameworks

[1]https://github.com/haowwang/offline_policy_eval_manipulation

for the problem, in which a *value function* is computed to evaluate the performance of the target policy. The value function takes in a state and outputs a scalar that quantifies the expected performance of the policy from the starting state. Fittingly, the policy evaluation problem is a cornerstone of the RL theory, and classical methods such as iterative policy evaluation [10], Monte Carlo [20, 22], and temporal-difference learning such as TD(0) [23], have been applied to a wide array of problems.

Despite its importance, offline policy evaluation for manipulation policies remains fundamentally challenging. The difficulty stems from three properties that are ubiquitous in real-world manipulation. First, rewards are typically sparse and task-driven: beyond a binary signal indicating task completion, dense rewards are often unavailable or unreliable. Second, practical policy evaluation is necessarily time-limited—episodes terminate after a finite horizon due to manual time-outs. As a result, policy failures and episode truncations are indistinguishable in offline data, introducing the well-known *truncation bias* in the value function, leading to systematic underestimation of policy performance and poor value estimates. Third, real-world manipulation policies often exhibit recovery behaviors, where they keep trying until task completion or manual time-out, and the resulting rollouts have non-monotonic task progression, complicating value function estimation.

In this work, we address these challenges by reformulating offline policy evaluation as a liveness problem under the formalism of Markov Decision Process (MDP). This problem formulation yields a value function that naturally accommodates sparse rewards and finite-horizon rollouts with non-monotonic task progression. Building on this formulation, we propose a two-stage policy evaluation framework designed to accurately capture task progression and reduce truncation bias. In the first stage, we compute reliable value estimates for states whose outcomes are unambiguous—namely, states that precede observed task completion. In the second stage, we bootstrap the evaluation of the remaining states using the value function from the first stage. Intuitively, this facilitates the backward propagation of value estimates from successful states to their predecessors within truncated episodes. This mechanism effectively overwrites the default pessimistic assumption applied to time-outs, replacing it with value estimates grounded in observed success.

We evaluate our method on two simulated manipulation tasks using visuomotor policies, including a Vision–Language–Action (VLA) model and a diffusion policy, and a hardware task using human demonstrations. Empirical results show that our approach recovers value functions that more accurately reflect task progression and substantially reduce truncation bias, outperforming classical baselines such as TD(0) and Monte Carlo policy evaluation. The contributions of this work are: 1) a liveness-based formulation of offline policy evaluation for sparse-reward manipulation tasks and 2) a two-stage, bootstrapped evaluation algorithm that mitigates truncation bias in finite-horizon offline data.

II. Background

A. Markov Decision Process

Markov Decision Process (MDP) is the formalism that underlies modern Reinforcement Learning (RL) theory [24], and we heavily utilize the framework in this paper. Formally, a MDP $\mathcal{M}$ is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state space and action space, respectively. $P : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the transition probability. $R : \mathcal{S} \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor.

B. Liveness Problem

In this section, we introduce the notion of liveness. Suppose we have a system described by continuous-time dynamics

$$\dot{s} = f(s, a) \tag{1}$$

where $s \in \mathcal{S}$ and $a \in \mathcal{A}$ are the state and control of the system, respectively, and $\dot{s}$ denotes the time derivative of state s.

Intuitively, the problem of liveness concerns with whether the system can ever reach a specific set of states $\mathcal{L}$ [15]. We call this region of interest the *target set*. Correspondingly, we define the target function $l : \mathcal{S} \to \mathbb{R}$ that encodes how far a state s is away from $\mathcal{L}$. $l(\cdot)$ is defined such that for state $s \in \mathcal{L}$, $l(s) \leq 0$. The liveness problem formulation is particularly popular in the optimal control community, with the goal being synthesis of controllers that can get the system to reach as far into the target set as possible [2]. However, in this work, we are primarily interested in the liveness problem formulation for evaluation purposes (i.e., determining how far a given policy can reach into the target set), rather than controller/policy synthesis.

III. Related Work

A. Classical Policy Evaluation

Classical approaches to policy evaluation estimate the value function $V_\pi(s)$ as the expected discounted return, $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ [24]. If the transition dynamics $p(s', r|s, a)$ are known, $V_\pi(s)$ can be computed via Iterative Policy Evaluation – iteratively applying the following Bellman operator:

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)[r + \gamma V_\pi(s')]$$

In offline settings where the transition dynamics is unavailable, Monte Carlo (MC) methods [20] approximate the value function by averaging the observed returns $G_{i,s}$ from the dataset $\mathcal{D}$ using number of visits $N(s)$ to a state:

$$V(s) \approx \frac{1}{N(s)} \sum_{i=1}^{N(s)} G_{i,s}$$

Standard MC treats this as a regression problem. However, recent large-scale robotic policies like $\pi^*_{0.6}$ [11] stabilize this by casting value learning as a multi-class classification problem over discretized return bins, learning a distribution over returns rather than a mean.

Alternatively, Temporal Difference (TD) learning [23, 8] reduces the high variance of MC by updating estimates incrementally via bootstrapping. Instead of waiting for a full episode return, TD updates $V_\pi(s)$ using the immediate reward and the current estimate of the next state's value:

$$V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$$

While these methods are foundational, they rely on summing rewards over time. In sparse-reward offline settings, MC suffers from high variance due to limited samples, while TD methods often struggle to propagate sparse success signals backward over long horizons due to vanishing gradients.

IV. Problem Formulation

In this work, we are interested in evaluating the performance of policies by computing their value functions, or in Reinforcement Learning parlance, the *policy evaluation problem* [24]. Intuitively, our goal is to determine whether the policy can complete the given task. Furthermore, we focus on the *offline* setting where we only have access to episodes generated by the policy but not the policy itself, and we only have access to *sparse rewards* (i.e., the only information provided is whether the task is complete at a given state). Though the problem formulation applies broadly, we are especially interested in manipulation policies.

More formally, given a task $\mathcal{I}$ and a dataset $\mathcal{D} = \{(s^i_{1:N_i}, a^i_{1:N_i-1})\}_{i=1}^M$, which contains M episodes each of length less than or equal to T generated by a stochastic policy π designed to complete $\mathcal{I}$, we aim to determine the value of any state s, under the given sparse rewards.

V. Method

A. Policy Evaluation as Liveness Problem

We consider policy evaluation in sparse-reward tasks where the only reward signal is whether a state is a goal state (i.e., task completion). Manipulation tasks commonly resemble such a task structure. In such settings, value functions computed using cumulative sparse reward formulations struggle to capture the proper progression of the task. A useful measure of task progression is *semantic* closeness to goal completion. Consider our first case study in Fig. 1 (a), a manipulation task where the policy is permitted to attempt actions until task completion, a setting that is common in real life manipulation tasks. The episode does not exhibit a monotonic task progression, a structure that cumulative sparse reward formulation inherently assumes. The states becomes semantically closer to task completion as the manipulator grasps the object, but upon slipping and dropping the object, the states become farther separated from task completion. We therefore reformulate policy evaluation as a *discounted* liveness problem, where the value of a state reflects the semantic distance between itself and task completion; i.e., a distance metric in the state space $\mathcal{S}$.

We first define the *undiscounted* liveness problem using the formalism of Markov Decision Process (MDP). We modify the classical MDP formulation introduced in Sec. II-A by replacing the reward function $R(\cdot)$ with the target function $l(\cdot)$: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, l, \gamma)$. Intuitively, the target function $l(s)$ is a signed distance function of the state to some target set $\mathcal{L}$. We define the value of a state s by

$$V_\pi(s) = \mathbb{E}_{\xi \sim \Xi_s^{\pi,P}} \left[\min_{k \in \{0,1,\ldots,N\}} l\Big(\xi(k)\Big) \right] \tag{2}$$

where ξ is a state trajectory (or equivalently, an episode) rolled out from state s under policy π and transition dynamics P, and $\Xi_s^{\pi,P}$ is the distribution of all such state trajectories. We assume that all episodes are able to achieve the minimum target over N steps (i.e., extending the episodes by continuing to roll out the policy will not lead to a lower target, or equivalently, pushing further into the target set $\mathcal{L}$). The value of a state s is the expectation of the lowest target achieved among all possible trajectories rolled out from s.

We use the principle of dynamic programming to transform the definition of the value function in Eq. 2 into a recursive Bellman equation in Eq. 3.

Proposition 1. *The value function in Eq. 2 satisfies the following inequality:*

$$V_\pi(s) \leq \mathbb{E}_\pi \Big[\mathbb{E}_P \Big[\min \big\{ l(s), V_\pi(s') \big\} \Big] \Big] \tag{3}$$

Furthermore, $|V_\pi(s) - \mathbb{E}_\pi \Big[\mathbb{E}_P \Big[\min \big\{ l(s), V_\pi(s') \big\} \Big] \Big]| \leq \sigma_Y$, *where* σ_Y *is the expectation over action* a_0 *and state* s_1 *of standard deviation of the random variable* $Y = \min_{k \in \{1,\ldots,N\}} l\big(\xi(k)\big)$.

This proposition is proved in Appx. A. The inequality in Eq. 3 implies that any fixed point of the corresponding equality defines a conservative (pessimistic) estimate of the true liveness value. This gap results from applying Jensen's inequality in the derivation of Eq. 3. Intuitively, when future trajectories have low variability (e.g., near-deterministic dynamics or policy), the gap is small, and the approximation is tight. Conversely, high variability in future trajectories widens the gap, making the approximation more pessimistic.

Since over-stating task progress (i.e., predicting a lower value than warranted) is undesirable in practice, we deliberately adopt this conservative (pessimistic) estimate in Eq. 4:

$$\tilde{V}_\pi(s) = \mathbb{E}_\pi \Big[\mathbb{E}_P \Big[\min \big\{ l(s), \tilde{V}_\pi(s') \big\} \Big] \Big] \tag{4}$$

B. Practical Approximation of the Liveness Value Function from Offline Data

In our setting of offline policy evaluation, we approximate the distributions of the policy π and transition dynamics P by the empirical distribution of the dataset $\mathcal{D}$ generated using π. More precisely,

$$\begin{aligned} \tilde{V}_\pi(s) &= \mathbb{E}_\pi \Big[\mathbb{E}_P \Big[\min \big\{ l(s), \tilde{V}_\pi(s') \big\} \Big] \Big] \\ &\approx \mathbb{E}_\mathcal{D} \Big[\min \big\{ l(s), \tilde{V}_\pi(s') \big\} \Big] \end{aligned} \tag{5}$$

Then, following Eq. 5, we arrive at the following Bellman operator

$$\tilde{V}_\pi(s) = \min\left\{l(s), \tilde{V}_\pi(s')\right\} \tag{6}$$

Equation 6 does not lend itself naturally to the notion of task progression, as it only captures the lowest value along the trajectory. Thus, a discount factor γ is introduced, and consequently induces a contraction mapping [9], and the result Prop. 2 is proved in Appx. B.

$$\tilde{V}_\pi(s) = (1-\gamma) + \gamma \min\{l(s), \tilde{V}_\pi(s')\} \tag{7}$$

Proposition 2. *The discounted Bellman operator Eq. 7 induces a contraction mapping under the supremum norm.*

The target function $l(s)$ is defined using the typical sparse reward for manipulation tasks, where the reward signal is only known at task completion (reaching a *goal state*):

$$l(s) = \begin{cases} 1 & \text{if } s \text{ is a not goal state} \\ -1 & \text{if } s \text{ is a goal state} \end{cases} \tag{8}$$

The discounting, along with the sparse reward formulation in Eq. 8, leads naturally to the notion of *steps to task completion*. To see this, let us take a state trajectory $\xi_{s_0}^{\pi,P} = \{s_0, s_1, \ldots s_N, s_{N+1} \ldots\}$. Without loss of generality, suppose $l(s_N) = -1$ and $l(s_k) = 1 \ \forall k \neq N$. Then using the discounted fixed-point Bellman equation Eq. 7, we have the following

$$\begin{aligned}
\tilde{V}_\pi(s_{N-1}) &= (1-\gamma) + \gamma \tilde{V}_\pi(s_N) \\
&= (1-\gamma) + \gamma(-1) \\
&= 1 - 2\gamma \\
\tilde{V}_\pi(s_{N-2}) &= (1-\gamma) + \gamma \tilde{V}_\pi(s_{N-1}) \\
&= (1-\gamma) + \gamma(1-2\gamma) \\
&= 1 - 2\gamma^2 \\
&\vdots \\
\tilde{V}_\pi(s_{N-k}) &= (1-\gamma) + \gamma \tilde{V}_\pi(s_{N-k+1}) \\
&= 1 - 2\gamma^k \ \forall\ 0 \leq k \leq N
\end{aligned} \tag{9}$$

Eq. 9 is the solution to the recurrence relation in Eq. 10

$$\tilde{V}_\pi(s_{N-k}) = (1-\gamma) + \gamma \tilde{V}_\pi(s_{N-k+1}), \ \tilde{V}_\pi(s_N) = -1 \tag{10}$$

The number of steps to task completion, denoted by k, from state s_{N-k} is related to its value $\tilde{V}_\pi(s_{N-k})$ by $k = \log_\gamma \frac{1-\tilde{V}_\pi(s_{N-k})}{2}$. However, with the current sparse reward target function $l(s)$ in Eq. 8, we only capture monotonic task progression.

C. Refining the Target Function via Bootstrapping

In order to fully capture the non-monotonic task progression, the target function $l(s)$ must provide a notion of distance to the target set, which is not provided by the binary sparse reward signal. We identify states where the policy's performance is unambiguous. Let $\mathcal{D}' \subseteq \mathcal{D}$ be the set of success episodes (reaches a goal state), and $\mathcal{D} \backslash \mathcal{D}' \subseteq \mathcal{D}$ be the set of timed-out episodes (never reaches any goal states). We compute the values of all descendant states of all goal states within $\mathcal{D}'$ using the Bellman operator in Eq. 7, and the values of these states are said to be *well-defined*. These values serve as "anchors", as they are computed using observed success from goal states. Let value function $\overline{V_\pi}$ take on the values of the states with well-defined values, and for all states s without well-defined values, $\overline{V_\pi}(s) = 1$.

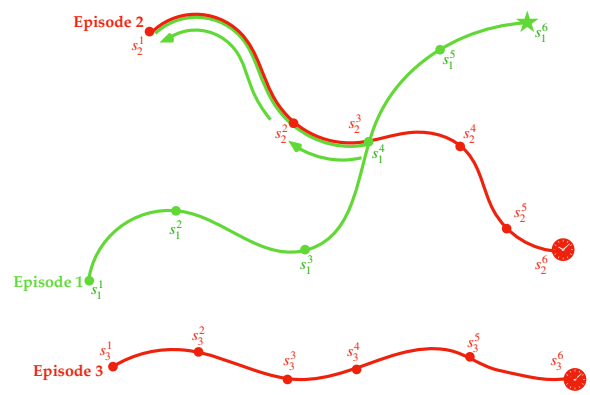

Fig. 2. Visualization of the bootstrap mechanism introduced in Sec. V-C. Episode 1 contains 2 goal states, and all states in this episode have well-defined values (i.e., $\overline{V}_\pi(s_1^k) < 1 \ \forall\ 0 < k \leq 6$). Episode 2 is terminated by time-out, but one of its states s_2^3 intersects with Episode 1. Lastly, Episode 3 is terminated by time-out and does not intersect with any episodes with goal states. In this case, all of its states take on the default sparse reward value of 1.

Next, we bootstrap the target function $l(s)$ with $\overline{V}_\pi(s)$ to provide a more accurate signal for rough distance to the target set. For example, states $\{s_1^1, \ldots s_1^5\}$ were initially pessimistically labeled with $l(s) = 1$, but their updated targets are $\overline{V}_\pi(s) < 1$. We apply the Bellman operator in Eq. 7 to the dataset $\mathcal{D}$. For clarity, the Bellman operator with the bootstrapped $l(s)$ is as follows

$$\tilde{V}_\pi(s) = (1-\gamma) + \gamma \min\{\overline{V}_\pi(s), \tilde{V}_\pi(s')\} \tag{11}$$

Importantly, the $\min$ operator in Eq. 7 acts as a "correction filter". We illustrate the mechanism in Fig. 2. Let us take Episode 2, denoted by ξ_2, an timed-out episode (without any goal states), and we take state $s_2^3 \in \xi_2$. Since ξ_2 is terminated at state s_2^6, using Eq. 7, we compute that $\tilde{V}_\pi(s_2^6) = \tilde{V}_\pi(s_2^5) = \tilde{V}_\pi(s_2^4) = 1$. However, s_2^3 is a state in a successful episode (s_1^4 in Episode 1), via bootstrapping, the known well-defined value $\overline{V}_\pi(s_2^3)$ immediately overrides the pessimistic time-out assumption of $\tilde{V}_\pi(s_2^4) = 1$ as $\min\{\overline{V}_\pi(s_2^3), \tilde{V}_\pi(s_2^4)\} = \overline{V}_\pi(s_2^3)$. Furthermore, for the predecessor state s_2^2 of s_2^3, though s_2^2 is not a state of any successful episodes (i.e., $\overline{V}_\pi(s_2^2) = 1$), the corrected value of $\tilde{V}_\pi(s_2^3)$ will be propagated backward via $\min\{\overline{V}_\pi(s_2^2), \tilde{V}_\pi(s_2^3)\} = \tilde{V}_\pi(s_2^3)$. The corrected value $\tilde{V}_\pi(s_2^2)$ will be similarly propagated backward to the initial state s_2^1 of ξ_2. On the other hand, none of the states in Episode 3 can have

their value corrected so they will take on the given pessimistic sparse reward value of 1.

Effectively, an episode is only penalized as a failure if it neither contains goal states nor overlaps with any episode that does. We summarize our overall method in Algo. 1.

Remark 1. *In practice, the state space is continuous, and as a result we would never encounter two identical states in the dataset. But the intuition holds that if two states are similar enough, the bootstrapping mechanism will trigger as if they are identical. This is further discussed in detail in Sec. VII.*

Algorithm 1: Two-Stage Policy Evaluation Framework

Input: Dataset $\mathcal{D}$
Output: Value function $\tilde{V}_\pi$
1 Compute $\overline{V_\pi}$ using subset $\mathcal{D}'$ of dataset $\mathcal{D}$
2 Bootstrap $\overline{l(s)}$ using $\overline{V_\pi}$
3 Compute $\tilde{V}_\pi$ using $\overline{l(s)}$ and Eq. 7
4 **return** $\tilde{V}_\pi$

D. Reduction of Truncation Bias from Manual Time-outs

We have established a value function that is capable of capturing non-monotonic task progression often seen in manipulation tasks, and a major benefit of the bootstrapping mechanism is the reduction of *truncation bias*. Truncation bias is introduced when episodes that are terminated via time-out are pessimistically assumed to have timed-out in a way that the policy will never reach the goal, even if given more time. Distinguishing between policy failures and episode truncations is a key practical challenge in policy evaluation, and the resulting bias leads to value functions that underestimate policy performance.

To see how our formulation reduces the truncation bias, we take Episode 2 in Fig. 2. Since it is manually timed-out, all of its states would have taken on the pessimistic value of 1 without the bootstrapping mechanism. However, the bootstrapping mechanism updates the values for states s_2^3, s_2^2, and s_2^1 to optimistic values that more accurately reflect the actual task progression, effectively reducing truncation bias.

VI. Experiments

We assess the performance of our method, along with several baselines, in 3 case studies: a pick and place task from LIBERO [14], a square peg and hole insertion task from Robomimic [16] (referred to as the Square task), and a cloth folding task on a Franka Panda hardware platform. Each case study utilizes a dataset generated by the target policy consisting of observations consumed by the policy and the success/timed-out binary label. All methods perform offline policy evaluation by estimating the value function of the target policy.

Baselines. We compare our method against four baselines: (1) **Ours-NB**, an ablation of our method without bootstrapping the target function $l(\cdot)$; (2) **TD**, a standard TD(0) implementation [23]; (3) **MC**, a Monte Carlo baseline [24]; and (4) **MC-D**, the Distributional Monte Carlo implementation from [11]. The baseline Ours-NB uses the sparse reward from Eq. 8, while the remaining baselines use the following equivalent sparse reward from [11]:

$$r_t = \begin{cases} 0 & \text{if } t = T \text{ and success} \\ -C_{\text{fail}} & \text{if } t = T \text{ and failure} \\ -1 & \text{otherwise.} \end{cases} \tag{12}$$

where C_{fail} is a large constant.

Value function learning. All the value functions are parameterized as neural networks. The inputs to the neural networks are the concatenation of SigLIP2 [27] latent embeddings of image observations and proprioception states, matching the input space of the evaluated policy. Additionally, we utilize prioritized experience replay [21] during training. The hyperparameters for the neural networks, training, and prioritized experience replay are provided in Appx. C.

Evaluation metrics. We consider 3 metrics. The first metric is the **success metric**. Given a (sub)episode of length N, where the target policy successfully reaches a goal state, we convert the value at each frame into an estimated number of steps to the goal state. The metric is defined as the percentage of frames where the estimated step count is less than N. A high percentage indicates that the model correctly predicts that the goal can be reached within the actual episode duration N.

The second metric, referred to as the **failure metric**, assesses the value function's ability to recognize frames associated with timed-out episodes (i.e., containing no goal states). Given a failure episode, we compare the estimated steps to success against a standard task horizon N_s, a threshold representing the typical maximum number of steps required to solve the task from any starting state. The metric is defined as the percentage of frames where the inferred step to success exceeds N_s. A high percentage indicates that the model correctly predicts that the goal is effectively unreachable within the task horizon.

The third metric is the **composite metric**, which is defined to be the arithmetic mean of the success and failure metrics. We utilize this metric to capture the trade-off between the success and failure metrics.

Remark 2 (Relation to standard classification metrics)**.** *The success and failure metrics defined above involve thresholding the value to a binary label, which may suggest that standard classification metrics are applicable in our case studies. However, despite the thresholding, standard classification metrics (e.g., precision, recall, and F1) are not directly interpretable in our setting because binary labels are not well-defined for all states due to recovery behaviors and manual time-outs. Specifically, in successful episodes, intermediate states may exhibit recovery behaviors before eventual task completion, making the notion of steps-to-success ambiguous. Therefore, the success metric is restricted to the final N-step subtra-*

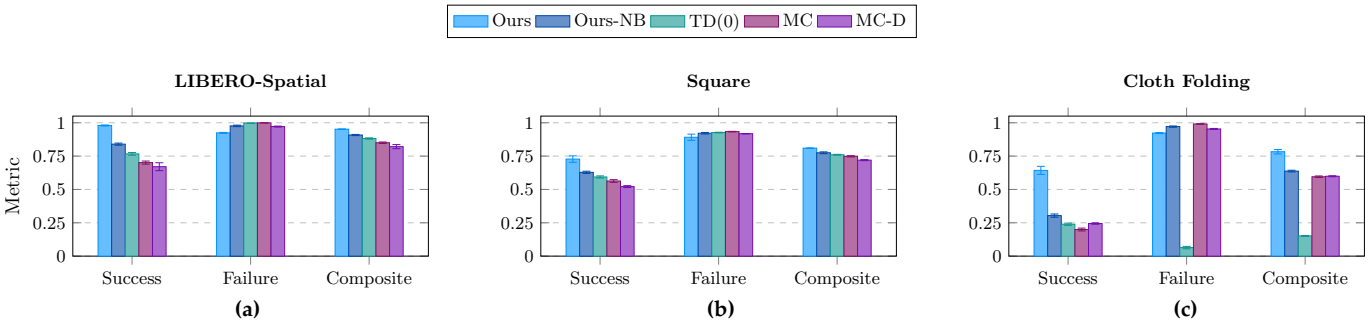

Fig. 3. Success, failure, and composite metrics for each method on the (a) LIBERO-Spatial, (b) Square, and (c) Cloth Folding tasks (dataset size 200 for the simulation experiments and 150 for the hardware experiment, 5 training seeds, higher is better; error bars denote standard deviation).

jectory where progress is monotonic and steps-to-success are well-defined. Earlier states in the same episode are excluded from evaluation. The failure metric is defined as the percentage of states in failure episodes whose predicted steps-to-success exceeds N_s, since these states are empirically unlikely to lead to task completion within N_s steps. Furthermore, a state not classified as successful does not necessarily imply failure — it may still be recoverable. As a result, the two metrics are defined over disjoint subsets of states, and there is no consistent binary labeling over the full dataset, making standard classification metrics unsuitable for this problem setting.

A. Case Study 1: LIBERO Pick and Place

The target policy for this case study is Physical Intelligence's π_0 [5] with a fixed language command of "*pick up the black bowl next to the ramekin and place it on the plate*". The task is part of the LIBERO-Spatial task suite [14], a kitchen table-top simulation environment with various objects where a Franka Panda arm attempts to pick up a bowl and put in on a plate. We collected 200 training and 100 testing episodes, each with 50% success/timed-out split. The manual time-out is set to 250 steps for this case study.

We first provide a qualitative example of our method in action in Fig. 1 (a). Intuitively, a value of -1 corresponds to imminent success, and value of 1 corresponds to requiring infinite steps to complete the task (i.e., a definitive failure). When the episode starts, the gripper is very close to the bowl, and our method assigns a moderate negative value. The manipulator picks up the bowl and the value decreases since the policy is making good progress. Then, the bowl slips out of the gripper, and the value increases sharply, correctly indicating that the state is now significantly farther from the goal, semantically. Finally, the policy recovers by picking up the bowl again and completes the task. Correspondingly, the value tends toward -1 as the policy progresses towards task completion.

We now discuss the quantitative results for the 3 metrics presented in Fig. 3 (a). Our method achieves the best performance in both the success metric and the composite metric, while remaining competitive in the failure metric. Notably, the policy in this experiment is capable of recovering from various setbacks to achieve success if more time is given. Our bootstrapping mechanism is able to correct the pessimistic values of many states in timed-out episodes to reflect a more accurate value for the state given the policy's capabilities. This largely explains the wide margin in the success metric between our method and the baselines.

B. Case Study 2: Square Peg and Hole Insertion

In this case study, we evaluate a diffusion policy [7] trained to perform a peg and hole insertion task simulated in Robomimic [16]. The task involves a Franka Panda arm grasping a square nut and placing it on a square peg. We again collected 200 and 100 episode datasets with 50% success splits each, for training and testing, respectively. The manual time-out is set to 200 steps.

We present the quantitative results in Fig. 3 (b), where our method achieves the best performance in the success metric while remaining competitive in the failure metric. However, both our method and the baselines perform substantially worse in the success metric compared to the LIBERO-Spatial case study. Unlike the VLA-driven LIBERO-Spatial task, the diffusion policy is trained on a single task without expert recovery behavior, meaning unrecoverable failures occur in the episodes at unknown times. Additionally, there are fewer visual cues in the image, which is dominated primarily by the static background. Thus, successful trajectories are often visually identical to timed-out episodes until a setback occurs (e.g., a sudden slip, poor grasp, or bad peg alignment). Hence, the value function is unlikely to benefit from the bootstrapping mechanism when many opposite-label states nearly overlap in the image space, or have similar semantic meaning. This noise in the image-label mapping may result in the decreased success metric score, as the value function is uncertain that the goal can be reached within the task horizon until a strong visual cue (e.g., good grasp, proper alignment) has been observed.

One emergent property of the value function is its predictive capability. We share one example in Fig. 1 (b). In this episode, the gripper does not align well with the square nut leading up to the grasp, and at the time of the grasp being established, the value is already close to 1, strongly indicating that success is unlikely. A few steps later, the square nut slips out of the gripper, and the policy never recovers. The value

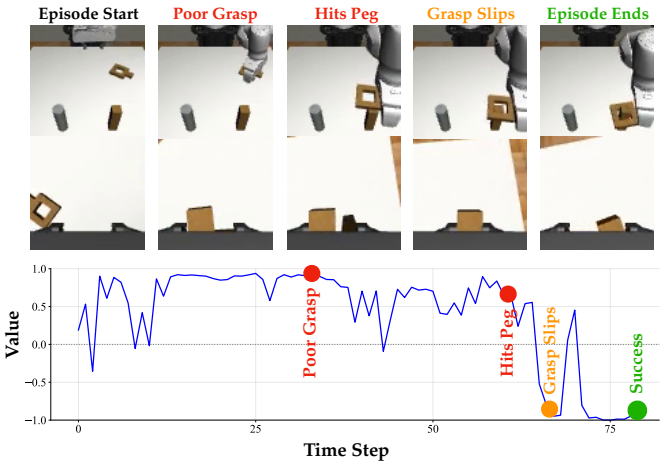

Fig. 4. Our method's predicted values over a successful Square task episode showing a false-positive high value due to a poor grasp that managed to succeed.

function is able to associate the misaligned grasp with the subsequent slippage. This predictability could be due to the strong semantic cue for a poor grasp versus a proper grasp, which is correlated with task success.

On the flip side, the value function is not able to pre-emptively capture failure moments that are "unexpected". For instance, the square nut sometimes slips out of the gripper despite an established good grasp, at least to the human eyes. The value function, at best, can react to the slippage as it happens but not preemptively. Such slippage events may occasionally act in the opposite way, for example, in Fig. 4. The manipulator grasps the square block poorly (an off-center grasp is more often associated with failure), but the grasp happens to slip into a proper grasp after contacting the peg, and the task is completed successfully. This false-positive behavior draws down the performance metrics, but indicates that the value function has learned that an anomalous grasp is likely to lead to failure and adjusts accordingly upon making a proper grasp.

C. Case Study 3: Human Cloth Folding

We conduct a hardware experiment on a cloth folding task using a Franka Panda platform, where the value function is learned from human demonstrations. A human operator teleoperates the robot via a Meta Quest 3 headset and controller, with the goal of folding a rectangular towel into a triangle along one of its diagonals. This setting is substantially more challenging than the simulated tabletop tasks for three reasons: (1) the cloth is deformable and difficult to manipulate, (2) human teleoperation behavior is inconsistent across episodes, and (3) labels are noisy, since human visual inspection sometimes assigns different labels to visually similar end states.

We train all methods on 150 episodes (75 success, 75 timed-out) and evaluate on 20 held-out episodes (10 success, 10 timed-out). The manual time-out is set to 300 steps in this case study. Unlike the simulated experiments, the value function input here consists solely of the encoded latent of the base image, as this is the only observation available to the operator during teleoperation.

We first present a qualitative example in Fig. 1 (c). For most of the episode, the operator struggles to establish a good grasp on the cloth, and the value oscillates near zero, reflecting stagnant task progression. Once a successful grasp is established and the folding begins, the value drops sharply toward -1, indicating clear progress toward task completion.

Quantitative results are shown in Fig. 3 (c). Our method outperforms all baselines on the success metric by a wide margin, more than $2\times$ the next-best baseline (Ours-NB), while trailing slightly on the failure metric. Two factors contribute to this outcome: (1) recovery behaviors are prevalent throughout the dataset because the operator is not adept at teleoperated cloth manipulation, a setting in which our bootstrapping mechanism is particularly effective; and (2) the operator does not exhibit clear failure modes, and as a result observations from success and timed-out episodes are often visually similar, complicating value function learning.

The cloth folding task represents the most extreme instance of the conditions our method targets: frequent recovery behavior, and unpredictable episode truncation caused by fixed-time time-outs that cut off episodes at varying stages of task progression. The wide success metric margin over baselines suggests that the bootstrapping mechanism is most effective precisely in these settings.

D. Ablating the Bootstrapping

To isolate the contribution of the bootstrapping mechanism, we consider an ablation of our method, **Ours-NB**, where the target function is simply the sparse rewards defined in Eq. 8. As we show in our experiments, it underperforms compared to its bootstrapped counterpart in all 3 case studies, performing worse on the success and composite metrics but slightly better on the failure metric (Fig. 3). This observation supports the conclusion that the bootstrapping increases success metric accuracy at a small sacrifice of failure metric accuracy.

E. Ablation On Dataset Size

Since we are trying to evaluate policies with their actual rollouts, sample efficiency is crucial for the practical feasibility of our method. We conduct an ablation study for each case study to assess the performance of all the methods under different dataset sizes. Recall that we collected 200 episodes in the simulation experiments and 150 episodes in the hardware experiment for training, with 50%-50% successful/timed-out episodes split, in each case study. We subsample 50, 100, and 150 episodes, maintaining a 50%-50% success/time-out split, and run all the methods on the smaller datasets.

We visualize the success metric and failure metric results for all 3 case studies in Fig. 5. As the dataset size increases, our method improves on both the success and failure metrics, and the variance on the performance decreases. Furthermore, our method outperforms the baseline in the success metric while being competitive in the failure metric, across all dataset sizes.

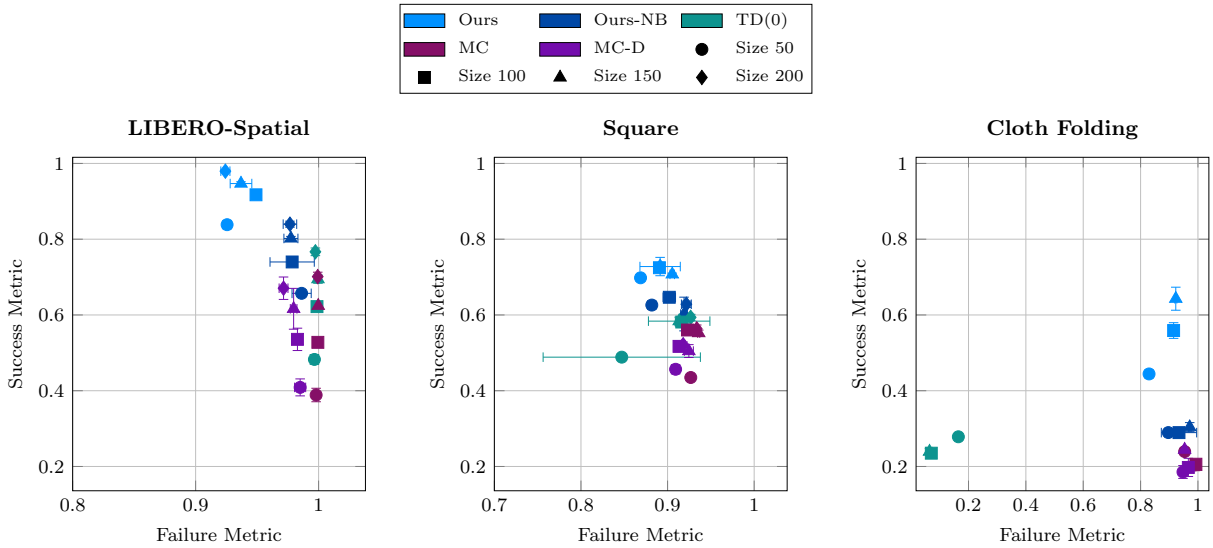

Fig. 5. Dataset size ablation showing success metric vs. failure metric (higher is better for both) for each method on the (a) LIBERO-Spatial, (b) Square, and (c) Cloth Folding tasks across 5 training seeds. Colors distinguish methods while shapes distinguish dataset sizes; error bars denote standard deviation. Cloth Folding is evaluated only up to dataset size 150.

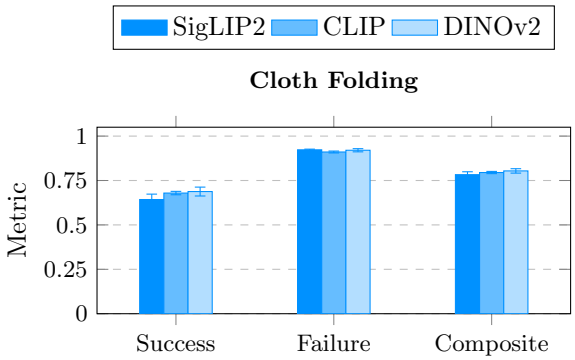

Fig. 6. Encoder ablation on the Cloth Folding task (dataset size 150, 5 training seeds, higher is better; error bars denote standard deviation) comparing SigLIP2, CLIP, and DINOv2 as the image encoder for our method.

Notably, our method with the smallest dataset sizes still maintains its much improved success metric and competitive failure metric when compared to baseline methods with the full dataset. This shows that our method remains useful in low-data regimes.

F. Ablation on Embedding Space

While the default image encoder used in all experiments is SigLIP2 [27], we additionally conduct an ablation on the choice of image encoder in the cloth folding task with the full dataset (150 episodes). We run our method with SigLIP2 [27], DINOv2 [18], and CLIP [19], with quantitative results across all metrics shown in Fig. 6.

Our method performs comparably across the three encoders on all metrics, indicating that it is largely insensitive to the choice of image encoder. This is expected, as large pretrained image encoders are generally competent at retaining task-relevant features. So long as the necessary information is preserved in the latent representation, our method can correlate these features with semantic task progress. Conversely, if the encoder fails to retain such information, we would expect any method, including ours, to degrade.

G. Statistical Significance

To assess statistical significance, we apply the Alexander-Govern test [1] to detect overall differences among methods, followed by pairwise Welch's t-tests [30] with Benjamini-Hochberg correction [3] to control the false discovery rate, both at significance level $\alpha = 0.05$. Across all case studies and dataset sizes, our method achieves statistically significant improvements over all baselines on both the success and composite metrics.

VII. Discussion

In this section, we discuss some observations derived from considering all the experiment results across 3 case studies and different dataset sizes.

A. Qualitative Analysis of Our Method

Our first observation is that our method consistently achieves the best success metric, and this is expected due to the bootstrapping mechanism described in Sec. V-C. In theory, the bootstrapping mechanism ensures that states on a timed-out episode take on values less than 1 as long as they have a descendant state that is previously seen on a success episode, effectively making the values of those states more optimistic than the default pessimistic value of 1. Since the value function is a learned neural approximator with data from the dataset of episodes, the bootstrapping mechanism does not

apply exclusively to the states on timed-out episodes with descendant states seen on other success episodes, but rather states from timed-out episodes that look similar enough to states on success episodes. As a result, a large number of states from timed-out episodes have their values corrected lower (i.e., taking more optimistic values). However, the effectiveness of the bootstrapping mechanism varies depending on the target policy and the task. For policies that exhibit more recovery behaviors, the bootstrapping mechanism is beneficial for capturing the non-monotonic progression of the task, as the timed-out episodes generated from such policies are more severely affected by the truncation bias.

Our second observation is that the `min` operator, that corrects the pessimism induced by failure labels and truncation bias, encourages the network to learn non-monotonic task progression. Fig. 1 (a) shows our method will recognize a recoverable failure (dropping the bowl) and result in an increase in value, indicating it will take more steps to complete the task, until the manipulator adjusts to a position from which it may recover and the value then decreases as it picks up the bowl and completes the task. Fig. 4 shows similar behavior, where an anomalous grasp is flagged as a high-value failure-prone behavior until the grasp is adjusted to a proper grasp and the value accordingly decreases toward the task completion. Conversely, Fig. 1 (b) shows how when an *irrecoverable* failure occurs (the manipulator drops the object and the policy is not trained with recovery behavior from such a state), the value remains as high as possible, i.e., infinite steps to success.

The third observation is that our method consistently achieves the worst failure metric, and this is also expected as a consequence of the bootstrapping mechanism. As we have mentioned in the previous observation, states from timed-out episodes that look similar enough to states on success episodes will take on more optimistic values. In this case, this is to the detriment of our method, as some timed-out episodes should not have their states updated with more optimistic values. But since some of the states look similar enough to some states on success episodes, their values and their predecessors' values get updated to be more optimistic than they should be. This problem is especially severe if the task environment does not contain rich visual cues (i.e., states are not very distinguishable).

B. Practical Considerations

Based on the first and third observations, it seems like our method essentially renders the value function more optimistic. The reality is more nuanced. Our method is able to maintain a competitive failure metric and simultaneously substantially improve upon the success metric. In some sense, our method is paying a small penalty in the failure metric for the larger improvement on the success metric. It is worthwhile to point out that our method does not push the values of all states to be more optimistic, but it only does so for states that can be closely associated with known success states. Such state association depends on the latent/pixel space similarity for the visuomotor policy input. This soft overlap is likely handled by the neural network learning a smooth manifold in the SigLIP2 feature space, such that perceptually similar inputs lead to similar network outputs; i.e., semantic distance. This may implicitly build a graph similar to the tabular case. This mechanism allows the Bellman update to propagate success signals backward through failure trajectories.

This may explain the performance divergence between the LIBERO-Spatial/Cloth Folding tasks and the Square tasks. In the LIBERO-Spatial and Cloth Folding tasks, semantically similar events (e.g., grasping the bowl/cloth) could occur multiple times in both successful and timed-out episodes, allowing the network to learn the association between this behavior and the label without penalizing recovery attempts. However, in the Square task, no recoveries were attempted (dropping the object is irrecoverable under the policy), so the initial $l(s)$ will tend to be overly conservative, reducing the benefit of the bootstrapping mechanism.

VIII. Conclusion

We present a framework for offline evaluation of visuomotor manipulation policies with sparse rewards and truncation bias from finite-length episodes. By reformulating policy evaluation as a liveness problem, our method accurately captures task progression and reduces truncation bias.

This work has several limitations. First, the bootstrapping mechanism can suffer from over-optimism because the overlap of states is learned by the network, leaving no definite boundary for deciding which timed-out episodes are considered to have intersected success episodes (as in Fig. 2). As a result, the effectiveness of the bootstrapping mechanism depends heavily on the image/latent inputs to the network, affecting the initial estimation $\overline{V}_\pi(s)$ in the first stage. Second, the performance of our method depends heavily on the dataset, and it is not expected to be reliable outside its training distribution. Depending on the makeup of the dataset, the value function can skew optimistic or pessimistic, and currently it is unclear to us how to characterize such bias systematically. Third, the performance of our method was shown to decrease with a policy that does not exhibit recovery behaviors, which is an expected result from the bootstrapping.

Future work in this direction could cover the over-optimism from the state overlap problem by implementing some structured boundary defining state similarity. Additionally, improvements in data efficiency could be explored to reduce the burden of policy rollout collection. Evaluating various other visuomotor policy architectures and how their differences manifest in value function performance metrics is another direction to explore.

Acknowledgments

This research is supported in part by the DARPA Assured Neuro Symbolic Learning and Reasoning (ANSR) program and by the NSF CAREER program (2240163). The authors would also like to thank Albert Lin for setting up the Franka Panda hardware platform and assisting with the hardware experiments.

References

[1] Ralph A Alexander and Diane M Govern. A new and simpler approximation for ANOVA under variance heterogeneity. *Journal of Educational Statistics*, 19(2):91–101, 1994. URL https://www.jstor.org/stable/1165140.

[2] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J. Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253, 2017. doi: 10.1109/CDC.2017.8263977. URL https://arxiv.org/abs/1709.07523.

[3] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300, 1995. URL https://www.jstor.org/stable/2346101?seq=1.

[4] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. URL https://arxiv.org/abs/2503.14734.

[5] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

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

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

[8] Peter Dayan. The convergence of $td(\lambda)$ for general λ. *Machine Learning*, 8(3-4):341–362, 1992. URL https://link.springer.com/article/10.1007/BF00992701.

[9] Jaime F. Fisac, Neil F. Lugovoy, Vicenc Rubies-Royo, Shromona Ghosh, and Claire J. Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8550–8556. IEEE. ISBN 978-1-5386-6027-0. doi: 10.1109/ICRA.2019.8794107. URL https://ieeexplore.ieee.org/document/8794107/.

[10] Ronald A. Howard. *Dynamic Programming and Markov Processes*. The Technology Press of MIT and John Wiley & Sons, Cambridge, MA and New York, NY, 1960. ISBN 9780262080095. URL https://gwern.net/doc/statistics/decision/1960-howard-dynamicprogrammingmarkovprocesses.pdf.

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

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

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

[14] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023. URL https://arxiv.org/abs/2306.03310.

[15] John Lygeros. On reachability and minimum cost optimal control. *Automatica*, 40(6):917–927, 2004. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2004.01.012. URL https://www.sciencedirect.com/science/article/pii/S0005109804000263.

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

[17] Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. *arXiv preprint arXiv:2410.13816*, 2024. URL https://arxiv.org/abs/2410.13816.

[18] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. URL https://arxiv.org/abs/2304.07193.

[19] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models

from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. URL https://proceedings.mlr.press/v139/radford21a/radford21a.pdf.

[20] Reuven Y. Rubinstein and Dirk P. Kroese. *Simulation and the Monte Carlo Method*. John Wiley & Sons, Hoboken, NJ, 3rd edition, 2016. ISBN 978-1-118-63216-1. URL https://www.wiley.com/en-us/Simulation+and+the+Monte+Carlo+Method%2C+3rd+Edition-p-9781118632161.

[21] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015. URL https://arxiv.org/abs/1511.05952.

[22] Satinder P. Singh and Richard S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22(1-3):123–158, 1996. doi: 10.1007/BF00114726. URL http://www.incompleteideas.net/papers/singh-sutton-96.ps.

[23] Richard S. Sutton. Learning to predict by the methods of temporal differences. 3(1):9–44. ISSN 1573-0565. doi: 10.1007/BF00115009. URL https://doi.org/10.1007/BF00115009.

[24] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998. URL http://incompleteideas.net/book/the-book-2nd.html.

[25] Gemini Robotics Team, Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025. URL https://arxiv.org/abs/2510.03342.

[26] Philip S. Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High-confidence off-policy evaluation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, volume 29, pages 3000–3006, 2015. URL https://ojs.aaai.org/index.php/AAAI/article/view/9541.

[27] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025. URL https://arxiv.org/abs/2502.14786.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. URL https://arxiv.org/abs/1706.03762.

[29] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023. URL https://arxiv.org/abs/2308.12952.

[30] Bernard L Welch. The generalization of 'Student's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947. URL https://www.jstor.org/stable/2332510?seq=1.

[31] 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*, pages 2165–2183. PMLR, 2023. URL https://arxiv.org/abs/2307.15818.

APPENDIX A
PROOF OF PROPOSITION 1

Proof: We use a slightly different indexing of states in this proof compared to the statement of the proposition for easier indexing in the proof. Let $s = s_0$ and $s' = s_1$.

We start from the definition of value from Eq. 2. Note that in Eq. 13, the states s_k $\forall k \in \{0, 1, \ldots, N\}$ are states, indexed sequentially, in the state trajectory ξ. Throughout the proof, we make use of the law of iterative expectation to break the expectation in Eq. 14 ultimately into 3 nested expectations in Eq. 16. We make use of Jensen's inequality in Eq. 18 to move the inner expectation inside the outer $\min$ operator. With $l(s_0)$ being a constant, $\min\left\{l(s_0), \min_{k\in\{1,\ldots,N\}} l(s_k)\right\}$ is concave with respect to $Y = \min_{k\in\{1,\ldots,N\}} l(s_k)$, and hence with Jensen's Inequality we have $\mathbb{E}\left[f(Y)\right] \leq f(\mathbb{E}\left[Y\right])$. Detailed derivations are as followed.

$$
\begin{aligned}
V_\pi(s_0) &= \mathbb{E}_{\xi\sim\Xi^{\pi,P}_{s_0}} \min_{k\in\{0,1,\ldots,N\}} l(s_k) && (13)\\
&= \mathbb{E}_{a_0,s_1,a_1,\ldots,s_N} \min_{k\in\{0,1,\ldots,N\}} l(s_k) && (14)\\
&= \mathbb{E}_{a_0\sim\pi}\mathbb{E}_{s_1,a_1,\ldots,s_N|a_0} \min_{k\in\{0,1,\ldots,N\}} l(s_k) && (15)\\
&= \mathbb{E}_{a_0\sim\pi}\mathbb{E}_{s_1\sim P}\mathbb{E}_{a_1,s_2,\ldots,s_N|a_0,s_1} \min_{k\in\{0,1,\ldots,N\}} l(s_k) && (16)\\
&= \mathbb{E}_{a_0\sim\pi}\mathbb{E}_{s_1\sim P}\mathbb{E}_{a_1,s_2,\ldots,s_N|a_0,s_1} \min\left\{l(s_0), \min_{k\in\{1,\ldots,N\}} l(s_k)\right\} && (17)\\
&\leq \mathbb{E}_{a_0\sim\pi}\mathbb{E}_{s_1\sim P} \min\left\{l(s_0), \mathbb{E}_{a_1,s_2,\ldots,s_N|a_0,s_1} \min_{k\in\{1,\ldots,N\}} l(s_k)\right\} && (18)\\
&= \mathbb{E}_{a_0\sim\pi}\mathbb{E}_{s_1\sim P} \min\left\{l(s_0), V_\pi(s_1)\right\} && (19)
\end{aligned}
$$

We now show $|V_\pi(s) - \mathbb{E}_{a_0\sim\pi}\mathbb{E}_{s_1\sim P}\min\left\{l(s_0), V_\pi(s_1)\right\}| \leq \sigma_Y$. We first note that $V_\pi(s) \leq \mathbb{E}_{a_0\sim\pi}\mathbb{E}_{s_1\sim P}\min\left\{l(s_0), V_\pi(s_1)\right\}$, and as a result we only need to show $\mathbb{E}_{a_0\sim\pi}\mathbb{E}_{s_1\sim P}\min\left\{l(s_0), V_\pi(s_1)\right\} - V_\pi(s) \leq \sigma_Y$. We first denote $G = \mathbb{E}_{a_0\sim\pi}\mathbb{E}_{s_1\sim P}\min\{l(s_0), V_\pi(s_1)\} - V_\pi(s_0)$. Then,

$$
\begin{aligned}
G &= \mathbb{E}_{a_0,s_1}\min\{l(s_0), V_\pi(s_1)\} - V_\pi(s_0) && (20)\\
&= \mathbb{E}_{a_0,s_1}\min\{l(s_0), V_\pi(s_1)\} - \mathbb{E}_{a_0,s_1,a_1,\ldots}\min\{l(s_0), Y\} && (21)\\
&= \mathbb{E}_{a_0,s_1}\left[l(s_0) - \max\{0, l(s_0) - V_\pi(s_1)\}\right] - \mathbb{E}_{a_0,s_1,a_1,\ldots}\left[l(s_0) - \max\{0, l(s_0) - Y\}\right] && (22)\\
&= \mathbb{E}_{a_0,s_1,a_1,\ldots}\max\{0, l(s_0) - Y\} - \mathbb{E}_{a_0,s_1}\max\{0, l(s_0) - V_\pi(s_1)\} && (23)
\end{aligned}
$$

Note that we use the identity $\min\{a, b\} = a - \max\{0, a - b\}$ in Eq. 22, and we take $l(s_0)$ out of each expectation in Eq 23. We consider 3 cases.

Case 1: $l(s_0) < l(s_k)$ for all possible future states $s_1, \ldots, s_N$. Then $l(s_0) - Y < 0$ for any state trajectories, and as a result $\mathbb{E}_{a_0,s_1,a_1,\ldots}\max\{0, l(s_0) - Y\} = 0$. Similarly, $l(s_0) - V_\pi(s_1) < 0$ for any a_0 and s_1. Hence, $G = \mathbb{E}_{a_0,s_1,a_1,\ldots}\max\{0, l(s_0) - Y\} - \mathbb{E}_{a_0,s_1}\max\{0, l(s_0) - V_\pi(s_1)\} = 0 - 0 = 0$.

Case 2: $l(s_0) > l(s_k)$ for all possible future states $s_1, \ldots, s_N$. Then, we have $\mathbb{E}_{a_0,s_1,a_1,\ldots}\max\{0, l(s_0) - Y\} = l(s_0) = \mathbb{E}_{a_0,s_1}\max\{0, l(s_0) - V_\pi(s_1)\}$. Hence, $G = \mathbb{E}_{a_0,s_1,a_1,\ldots}\max\{0, l(s_0) - Y\} - \mathbb{E}_{a_0,s_1}\max\{0, l(s_0) - V_\pi(s_1)\} = 0$.

Case 3: Otherwise. We continue the derivation from Eq. 23.

$$
\begin{aligned}
G &= \mathbb{E}_{a_0,s_1,a_1,\ldots} \max\{0, l(s_0) - Y\} - \mathbb{E}_{a_0,s_1} \max\{0, l(s_0) - V_\pi(s_1)\} && (24)\\
&= \mathbb{E}_{a_0,s_1}\Big[\mathbb{E}_{a_1,s_2,\ldots|s_1} \max\{0, l(s_0) - Y\} - \max\{0, l(s_0) - V_\pi(s_1)\}\Big] && (25)\\
&= \mathbb{E}_{a_0,s_1}\Big[\mathbb{E}_{a_1,s_2,\ldots|s_1}\big[\max\{0, l(s_0) - Y\}\big] - \max\{0, l(s_0) - \mathbb{E}_{a_1,s_2,\ldots|s_1} Y\}\Big] && (26)\\
&= \mathbb{E}_{a_0,s_1}\mathbb{E}_{a_1,s_2,\ldots|s_1}\Big[\max\{0, l(s_0) - Y\} - \max\{0, l(s_0) - \mathbb{E}_{a_1,s_2,\ldots|s_1} Y\}\Big] && (27)\\
&\leq \mathbb{E}_{a_0,s_1}\mathbb{E}_{a_1,s_2,\ldots|s_1}\Big[|l(s_0) - Y - l(s_0) + \mathbb{E}_{a_1,s_2,\ldots|s_1}[Y]|\Big] && (28)\\
&= \mathbb{E}_{a_0,s_1}\mathbb{E}_{a_1,s_2,\ldots|s_1}\Big[|\mathbb{E}_{a_1,s_2,\ldots|s_1}[Y] - Y|\Big] && (29)\\
&\leq \mathbb{E}_{a_0,s_1}\sqrt{\mathbb{E}_{a_1,s_2,\ldots|s_1}\Big[(\mathbb{E}_{a_1,s_2,\ldots|s_1}[Y] - Y)^2\Big]} && (30)\\
&= \mathbb{E}_{a_0,s_1}\sigma_{Y|s_1} && (31)\\
&= \sigma_Y && (32)
\end{aligned}
$$

Eq. 28 is due to the fact that $\max\{0, a\}$ is 1-Lipschitz, and we use Cauchy-Schwarz inequality in Eq. 30. $\sigma_{Y|s_1}$ is the standard deviation of the random variable $Y = \min_{k\in\{1,\ldots,N\}} l(s_k)$ starting from state s_1, and its expectation over action a_0 and s_1 is denoted by σ_Y. Hence, $G \leq \sigma_Y$.

From the 3 cases, we conclude that $|V_\pi(s) - \mathbb{E}_{a_0\sim\pi}\mathbb{E}_{s_1\sim P} \min\{l(s_0), V_\pi(s_1)\}| \leq \sigma_Y$.

■

APPENDIX B
PROOF OF PROPOSITION 2

Proof: Let B denote the discounted Bellman operator defined in Eq. 7. We would like to show that there exists $\kappa \in [0, 1)$ such that $\left|B[\tilde{V}_\pi](s) - B[\tilde{V}'_\pi](s)\right| \leq \kappa||\tilde{V}_\pi - \tilde{V}'_\pi||_\infty$. Take $s \in \mathcal{S}$, and let us denote the immediate next state of s with s'. Also take $\tilde{V}_\pi$ and $\tilde{V}'_\pi$. In Eq. 34, we make use of the identity $|\min\{a, b\} - \min\{a, c\}| \leq |b - c|$. The full derivation is as follows.

$$
\begin{aligned}
&\left|B[\tilde{V}_\pi](s) - B[\tilde{V}'_\pi](s)\right| \\
&= \Big|\cancel{(1-\gamma)} + \gamma \min\{l(s), \tilde{V}_\pi(s')\} \\
&\qquad - \Big(\cancel{(1-\gamma)} + \gamma \min\{l(s), \tilde{V}'_\pi(s')\}\Big)\Big| && (33)\\
&= \gamma\left|\min\{l(s), \tilde{V}_\pi(s')\} - \min\{l(s), \tilde{V}'_\pi(s')\}\right| && (34)\\
&\leq \gamma\left|\tilde{V}_\pi(s') - \tilde{V}'_\pi(s')\right| && (35)
\end{aligned}
$$

Since $\left|B[\tilde{V}_\pi](s) - B[\tilde{V}'_\pi](s)\right| \leq \gamma\left|\tilde{V}_\pi(s') - \tilde{V}'_\pi(s')\right|$ holds $\forall s \in \mathcal{S}$, we have

$$
\left|B[\tilde{V}_\pi](s) - B[\tilde{V}'_\pi](s)\right| \leq \gamma||\tilde{V}_\pi - \tilde{V}'_\pi||_\infty \qquad (36)
$$

Since the discount factor γ is chosen in the open interval of $(0, 1)$, it serves as the contraction constant κ we look for in this proof. Hence, we have shown that for $\kappa = \gamma \in (0, 1)$, $\left|B[\tilde{V}_\pi](s) - B[\tilde{V}'_\pi](s)\right| \leq \kappa||\tilde{V}_\pi - \tilde{V}'_\pi||_\infty$. ■

APPENDIX C
METHOD HYPERPARAMETERS

The default hyperparameters for our experiments for the LIBERO-Spatial task, Square task, and Cloth Folding task are provided in TABLE I, II, and III, respectively.

TABLE I
LIBERO-SPATIAL TASK HYPERPARAMETERS

Hyperparameter	**Value**
Architecture	
MLP Layers	5
Hidden Units	512
Activation	GELU
Normalization	Layer Norm
Input Dimension	1544 (SigLIP2)
Training	
Learning Rate	1×10^{-5}
Batch Size	512
Training Epochs	5,000
Task & Method Constants	
Standard Task Horizon (T)	200
Our Method Discount factor (γ)	0.993
TD(0) Discount factor (γ)	0.999
TD(0) Reward Normalization	250
MC / MC-D Reward Normalization	500
MC-D Bins	201
Prioritized Experience Replay	
Replay Buffer Capacity	10,000
Gradient Steps per Batch	2
α	0.6
β_{start}	0.4
β_{end}	1

TABLE II
SQUARE TASK HYPERPARAMETERS

Hyperparameter	**Value**
Architecture	
MLP Layers	5
Hidden Units	512
Activation	GELU
Normalization	Layer Norm
Input Dimension	3090 (SigLIP2)
Training	
Learning Rate	1×10^{-5}
Batch Size	512
Training Epochs	5,000
Task & Method Constants	
Standard Task Horizon (T)	100
Our Method Discount factor (γ)	0.993
TD(0) Discount factor (γ)	0.995
TD(0) Reward Normalization	200
MC / MC-D Reward Normalization	400
MC-D Bins	201
Prioritized Experience Replay	
Replay Buffer Capacity	10,000
Gradient Steps per Batch	2
α	0.6
β_{start}	0.4
β_{end}	1

TABLE III
CLOTH FOLDING TASK HYPERPARAMETERS

Hyperparameter	**Value**
Architecture	
MLP Layers	5
Hidden Units	512
Activation	GELU
Normalization	Layer Norm
Input Dimension	768 (SigLIP2)
Training	
Learning Rate	2×10^{-5}
Batch Size	512
Training Epochs	5,000
Task & Method Constants	
Standard Task Horizon (T)	100
Our Method Discount factor (γ)	0.99
TD(0) Discount factor (γ)	0.99
TD(0) Reward Normalization	200
MC / MC-D Reward Normalization	400
MC-D Bins	201
Prioritized Experience Replay	
Replay Buffer Capacity	10,000
Gradient Steps per Batch	2
α	0.6
β_{start}	0.4
β_{end}	1

APPENDIX D
DETAILED EXPERIMENT RESULTS

In this appendix, we present detailed experiment results for the three case studies. Each experiment was run with 5 random seeds and the result is reported in the format `mean±standard deviation`. In TABLE IV, we present results for all 3 metrics for the experiments with full datasets (200 episodes for the simulation tasks and 150 episodes for the hardware cloth folding task). Furthermore, we present the results for the dataset ablation, described in Sec. VI-E, in TABLE V, VI, and VII.

Task	Method	Success Metric (↑)	Failure Metric (↑)	Composite Metric (↑)
LIBERO-Spatial	Ours	$\mathbf{0.9796 \pm 0.0039}$	0.9242 ± 0.0037	$\mathbf{0.9519 \pm 0.0017}$
	Ours-NB	0.8397 ± 0.0079	0.9767 ± 0.0055	0.9082 ± 0.0043
	TD(0)	0.7667 ± 0.0101	0.9974 ± 0.0005	0.8821 ± 0.0048
	MC	0.7012 ± 0.0117	$\mathbf{0.9993 \pm 0.0003}$	0.8502 ± 0.0057
	MC-D	0.6706 ± 0.0296	0.9715 ± 0.0036	0.8210 ± 0.0154
Square	Ours	$\mathbf{0.7278 \pm 0.0242}$	0.8914 ± 0.0233	$\mathbf{0.8096 \pm 0.0017}$
	Ours-NB	0.6283 ± 0.0083	0.9217 ± 0.0057	0.7750 ± 0.0069
	TD(0)	0.5937 ± 0.0081	0.9267 ± 0.0024	0.7602 ± 0.0032
	MC	0.5634 ± 0.0101	$\mathbf{0.9337 \pm 0.0021}$	0.7486 ± 0.0047
	MC-D	0.5214 ± 0.0072	0.9180 ± 0.0015	0.7197 ± 0.0041
Cloth Folding	Ours	$\mathbf{0.6429 \pm 0.0304}$	0.9227 ± 0.0039	$\mathbf{0.7828 \pm 0.0164}$
	Ours-NB	0.3038 ± 0.0120	0.9713 ± 0.0047	0.6376 ± 0.0064
	TD(0)	0.2392 ± 0.0082	0.0641 ± 0.0090	0.1516 ± 0.0036
	MC	0.1995 ± 0.0106	$\mathbf{0.9911 \pm 0.0019}$	0.5953 ± 0.0059
	MC-D	0.2444 ± 0.0052	0.9536 ± 0.0039	0.5990 ± 0.0039

TABLE IV
RESULTS OF ALL 3 METRICS FOR FULL DATASET FOR ALL 3 CASE STUDIES ACROSS 5 TRAINING SEEDS.

		Success Metric (↑)			
Task	Method	50	100	150	200
LIBERO-Spatial	Ours	$\mathbf{0.8383 \pm 0.0096}$	$\mathbf{0.9172 \pm 0.0072}$	$\mathbf{0.9469 \pm 0.0048}$	$\mathbf{0.9796 \pm 0.0039}$
	Ours-NB	0.6571 ± 0.0094	0.7399 ± 0.0024	0.8020 ± 0.0054	0.8397 ± 0.0079
	TD(0)	0.4826 ± 0.0130	0.6220 ± 0.0114	0.6945 ± 0.0108	0.7667 ± 0.0101
	MC	0.3887 ± 0.0178	0.5277 ± 0.0095	0.6243 ± 0.0068	0.7012 ± 0.0117
	MC-D	0.4088 ± 0.0224	0.5356 ± 0.0295	0.6163 ± 0.0539	0.6706 ± 0.0296
Square	Ours	$\mathbf{0.6981 \pm 0.0085}$	$\mathbf{0.7247 \pm 0.0101}$	$\mathbf{0.7072 \pm 0.0033}$	$\mathbf{0.7278 \pm 0.0242}$
	Ours-NB	0.6261 ± 0.0087	0.6467 ± 0.0081	0.6025 ± 0.0445	0.6283 ± 0.0083
	TD(0)	0.4886 ± 0.0093	0.5813 ± 0.0079	0.5837 ± 0.0106	0.5937 ± 0.0081
	MC	0.4349 ± 0.0059	0.5603 ± 0.0081	0.5521 ± 0.0079	0.5634 ± 0.0101
	MC-D	0.4566 ± 0.0121	0.5171 ± 0.0143	0.5047 ± 0.0170	0.5214 ± 0.0072
Cloth Folding	Ours	$\mathbf{0.4443 \pm 0.0110}$	$\mathbf{0.5589 \pm 0.0207}$	$\mathbf{0.6429 \pm 0.0304}$	–
	Ours-NB	0.2895 ± 0.0062	0.2894 ± 0.0081	0.3038 ± 0.0120	–
	TD(0)	0.2784 ± 0.0053	0.2351 ± 0.0079	0.2392 ± 0.0082	–
	MC	0.2394 ± 0.0042	0.2060 ± 0.0083	0.1995 ± 0.0106	–
	MC-D	0.1855 ± 0.0170	0.1975 ± 0.0240	0.2444 ± 0.0052	–

TABLE V
SUCCESS METRIC ACROSS THE DATASET SIZE ABLATION.

Task	Method	Failure Metric (↑)			
		50	100	150	200
LIBERO-Spatial	Ours	0.9256 ± 0.0034	0.9491 ± 0.0029	0.9369 ± 0.0088	0.9242 ± 0.0037
	Ours-NB	0.9862 ± 0.0078	0.9785 ± 0.0180	0.9775 ± 0.0057	0.9767 ± 0.0055
	TD(0)	0.9966 ± 0.0005	0.9987 ± 0.0008	0.9995 ± 0.0008	0.9974 ± 0.0005
	MC	$\mathbf{0.9980 \pm 0.0004}$	$\mathbf{0.9994 \pm 0.0002}$	$\mathbf{0.9997 \pm 0.0004}$	$\mathbf{0.9993 \pm 0.0003}$
	MC-D	0.9850 ± 0.0045	0.9828 ± 0.0035	0.9797 ± 0.0027	0.9715 ± 0.0036
Square	Ours	0.8688 ± 0.0013	0.8906 ± 0.0058	0.9054 ± 0.0026	0.8914 ± 0.0233
	Ours-NB	0.8818 ± 0.0049	0.9020 ± 0.0072	0.9190 ± 0.0039	0.9217 ± 0.0057
	TD(0)	0.8471 ± 0.0908	0.9164 ± 0.0074	0.9134 ± 0.0355	0.9267 ± 0.0024
	MC	$\mathbf{0.9268 \pm 0.0020}$	$\mathbf{0.9230 \pm 0.0030}$	$\mathbf{0.9360 \pm 0.0012}$	$\mathbf{0.9337 \pm 0.0021}$
	MC-D	0.9093 ± 0.0045	0.9132 ± 0.0018	0.9246 ± 0.0052	0.9180 ± 0.0015
Cloth Folding	Ours	0.8295 ± 0.0027	0.9152 ± 0.0048	0.9227 ± 0.0039	–
	Ours-NB	0.8967 ± 0.0117	0.9339 ± 0.0616	0.9713 ± 0.0047	–
	TD(0)	0.1646 ± 0.0125	0.0703 ± 0.0041	0.0641 ± 0.0090	–
	MC	$\mathbf{0.9537 \pm 0.0061}$	$\mathbf{0.9929 \pm 0.0028}$	$\mathbf{0.9911 \pm 0.0019}$	–
	MC-D	0.9478 ± 0.0107	0.9665 ± 0.0085	0.9536 ± 0.0039	–

TABLE VI
FAILURE METRIC ACROSS THE DATASET SIZE ABLATION.

Task	Method	Composite Metric (↑)			
		50	100	150	200
LIBERO-Spatial	Ours	$\mathbf{0.8820 \pm 0.0052}$	$\mathbf{0.9332 \pm 0.0027}$	$\mathbf{0.9419 \pm 0.0045}$	$\mathbf{0.9519 \pm 0.0017}$
	Ours-NB	0.8216 ± 0.0058	0.8592 ± 0.0096	0.8897 ± 0.0034	0.9082 ± 0.0043
	TD(0)	0.7396 ± 0.0065	0.8104 ± 0.0055	0.8470 ± 0.0054	0.8821 ± 0.0048
	MC	0.6933 ± 0.0087	0.7635 ± 0.0047	0.8120 ± 0.0034	0.8502 ± 0.0057
	MC-D	0.6969 ± 0.0099	0.7592 ± 0.0143	0.7980 ± 0.0271	0.8210 ± 0.0154
Square	Ours	$\mathbf{0.7834 \pm 0.0037}$	$\mathbf{0.8076 \pm 0.0045}$	$\mathbf{0.8063 \pm 0.0015}$	$\mathbf{0.8096 \pm 0.0017}$
	Ours-NB	0.7540 ± 0.0049	0.7744 ± 0.0050	0.7608 ± 0.0211	0.7750 ± 0.0069
	TD(0)	0.6678 ± 0.0420	0.7488 ± 0.0052	0.7485 ± 0.0213	0.7602 ± 0.0032
	MC	0.6808 ± 0.0032	0.7417 ± 0.0030	0.7441 ± 0.0037	0.7486 ± 0.0047
	MC-D	0.6829 ± 0.0066	0.7151 ± 0.0072	0.7146 ± 0.0069	0.7197 ± 0.0041
Cloth Folding	Ours	$\mathbf{0.6369 \pm 0.0050}$	$\mathbf{0.7371 \pm 0.0120}$	$\mathbf{0.7828 \pm 0.0164}$	–
	Ours-NB	0.5931 ± 0.0040	0.6117 ± 0.0288	0.6376 ± 0.0064	–
	TD(0)	0.2215 ± 0.0085	0.1527 ± 0.0047	0.1516 ± 0.0036	–
	MC	0.5965 ± 0.0027	0.5995 ± 0.0053	0.5953 ± 0.0059	–
	MC-D	0.5667 ± 0.0090	0.5820 ± 0.0159	0.5990 ± 0.0039	–

TABLE VII
COMPOSITE METRIC ACROSS THE DATASET SIZE ABLATION.

Encoder	Success Metric (↑)	Failure Metric (↑)	Composite Metric (↑)
SigLIP2	0.6429 ± 0.0304	$\mathbf{0.9227 \pm 0.0039}$	0.7828 ± 0.0164
CLIP	0.6793 ± 0.0096	0.9105 ± 0.0053	0.7949 ± 0.0063
DINOv2	$\mathbf{0.6878 \pm 0.0250}$	0.9203 ± 0.0089	$\mathbf{0.8040 \pm 0.0124}$

TABLE VIII
CLOTH FOLDING TASK ENCODER ABLATION AT DATASET SIZE 150 ACROSS TRAINING SEEDS.