MULTISTEP QUASIMETRIC LEARNING FOR SCALABLE GOAL-CONDITIONED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

The problem of learning how to reach goals in an environment has been a long-standing challenge in for AI researchers. Effective goal-conditioned reinforcement learning (GCRL) methods promise to enable reaching distant goals without task-specific rewards by stitching together past experiences of different complexity. Mathematically, there is a duality between the notion of optimal goal-reaching value functions (the likelihood of success at reaching a goal) and temporal distances (transit times states). Recent works have exploited this property by learning quasimetric distance representations that stitch long-horizon behaviors using the inductive bias of their architecture. These methods have shown promise in simulated benchmarks, reducing value learning to a shortest-path problem. But quasimetric, and more generally, goal-conditioned RL methods still struggle in complex environments with stochasticity and high-dimensional (visual) observations. There is a fundamental tension between the local dynamic programming (TD backups, temporal distances) that enables optimal shortest-path reasoning in theory and the statistical global MC updates (multistep returns, suboptimal in theory). We show how these approaches can be integrated into a practical GCRL method that fits a quasimetric distance using a multistep Monte-Carlo return. We show our method outperforms existing GCRL methods on long-horizon simulated tasks with up to 4000 steps, even with visual observations. We also demonstrate that our method can enable stitching in the real-world robotic manipulation domain (Bridge setup). Our approach is the first end-to-end GCRL method that enables multistep stitching in this real-world manipulation domain from an unlabeled offline dataset of visual observations.[1]

1 INTRODUCTION

It is natural for humans to use inherent ideas of distances to represent task progress: a GPS will tell you how far you are from the destination, and a cookbook will tell you how long a recipe will take. Humans can generalize tasks by taking the shortest route possible (stitching) and combining several learned tasks together in sequence in a new environment (combinatorial generalization). All of these problems can be thought of as reaching goals. The AI problem of reaching goals presents rich structure (formally, an optimal substructure property) that can be exploited to decompose hard problems into easy problems, and reinforcement learning (RL) has been used to address such problems. However, many past attempts at leveraging this property in modern high-dimensional, stochastic settings are incoherent or inconsistent. Current approaches tend to separate TD learning (local updates) (Mnih et al., 2013; Kostrikov et al., 2022; Kumar et al., 2020) and MC learning (global value propagation) (Eysenbach et al., 2022; Myers et al., 2024) into separate categories, with each method basing its approach based on only one, but not the other.

However, the effectiveness of both these methods degrade from a combination of increasing horizon length for TD methods (Park et al., 2025b) or difficulties in finding the optimal temporal distance (Park et al., 2024a). In conjunction, separate challenges exist if one were to train a policy using offline RL in real world. Under controlled datasets in simulated benchmarks, these algorithms may return a good goal-reaching policy, recent works have shown that additional work are needed for it to successfully train and deploy a policy in real life (Zheng et al., 2024).

[1]Website and code: `https://anonymous.4open.science/w/mqe-paper-686D/`

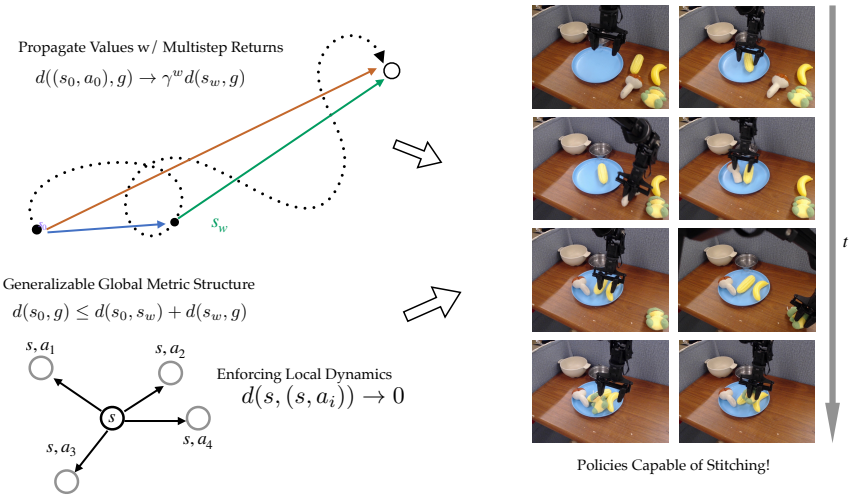

Figure 1: By combining one-step and multistep returns under a quasimetric architecture, we are able to obtain a Q function that allows the agent to "stitch" together tasks.

Our main contribution lies in **M**ultistep **Q**uasimetric **E**stimation (MQE), an offline GCRL method that incorporates both multistep value learning and quasimetric architectures without needing explicit hierarchy. We make the crucial observation that under the constraint of a quasimetric architecture, the regression objective for multistep returns are compatible with objectives for one-step consistency and action invariance, which are key insights for finding the optimal Q function. To our knowledge, MQE is the first method that is capable of leveraging multistep TD returns with global value propagation via quasimetric architectures.

By leveraging such a unique combinations of the benefits of TD and MC methods, MQE allows the learned policy to (*i*): display a much stronger level of horizon generalization compared to previous methods, which allow us for demonstrating the desired "stitching" behavior, (*ii*): provide a stable method for optimizing towards an optimal Q function, and (*iii*): such stability in training allows it to be applied in real-world robot learning problems without additional work in engineering. MQE can demonstrate SoTA performance on tasks that require complex control and long-horizon reasoning (up to 21 DoF and 4000 timesteps respectively), and in real world robotic settings, MQE displays compositional generalization behaviors that are not seen in previous RL algorithms.

2 Related Works

Our work build upon previous works in offline RL and temporal distance learning.

Goal-conditioned Reinforcement Learning Our findings particularly concern the discussions seen in recent work regarding the stitching and horizon generalization capabilities of GCRL. Recent findings (Park et al., 2025b) have shown that while it is easy to scale offline RL on short-horizon tasks, it is difficult to learn long-horizon tasks that require more complex reasoning within agent, which can easily deviate from the true optimal distance due to compounding TD errors.

Offline RL for Robotics In the same vein, we also focused on how we can apply these distance learning techniques to robotics. While reinforcement learning has been used to obtain highly capable specialist policies across various embodiments (Luo et al., 2025; Ball et al., 2023; Seo et al., 2025; Smith et al., 2023), behavior cloning still remains the most capable method for training generalist policies (O'Neill et al., 2024). Efforts have been made in allowing self-supervised and offline RL in robotics (Zheng et al., 2024), however, directly training a policy with offline RL remains difficult, as many researchers have instead used other ways to incorporate RL, such as rejection sampling (Nakamoto et al., 2025; Wagenmaker et al., 2025) or dataset curation (Mark et al., 2024; Xu et al., 2024). The focus of this work lies more on designing and implementing an offline RL method that can work in real world instead of only focusing on techniques for such training.

Temporal Distance Learning Our work is partially inspired by successor representations (Dayan, 1993; **?**). Previous works have shown that by using contrastive learning, (Myers et al., 2024) Other works have also shown that using contrastive learning as a way to parameterize distance learning may also recover behavior Q function.

Multistep RL RL using multistep returns have been widely used in online RL and offline-to-online RL settings (**?**Tian et al., 2025). This is desirable because in on-policy settings, performing RL with multistep return is correct (Munos et al., 2016). However, in an offline setting, the theory behind such correctness breaks down (watkins1989learning, 1989), as the learning becomes inherently off-policy. However, methods that use the entire trajectory and attempt to recover the behavior Q function can also be seen as a multistep RL algorithm (Eysenbach et al., 2022), however they also suffer the same problem of unable to learn Q^*.

3 PRELIMINARIES

In this section, we concretely define temporal distances and learning objective.

Notation. We consider a controlled Markov process (CMP) $\mathcal{M}$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition dynamics $\mathrm{P}(s'|s,a)$, and discount factor as γ. We consider goal-reaching policies $\pi(a|s,g) : \mathcal{S} \times \mathcal{S} \to \mathcal{A} \in \Pi$. We denote the behavior policy that is used to populate the offline dataset as π_β. In lieu of rewards, we optimize the maximum discounted likelihood of a policy reaching the goal, in which we can represent the Q-function and value function as:

$$Q_g^\pi(s,a) \triangleq \mathbb{E}_{\{\mathfrak{s}_t,\mathfrak{a}_t\}\sim\pi}\Big[\sum_{t=0}^{\infty}\gamma^t\,\mathrm{P}(\mathfrak{s}_t = g \mid \mathfrak{s}_0 = s, \mathfrak{a}_0 = a)\Big]. \tag{1}$$

$$V_g(s) \triangleq \mathbb{E}_{\{\mathfrak{s}_t\}\sim\pi}\Big[\sum_{t=0}^{\infty}\gamma^t\,\mathrm{P}(\mathfrak{s}_t = g \mid \mathfrak{s}_0 = s)\Big]. \tag{2}$$

Equivalently, we can define the optimal Q-function as $Q_g^*(s,a) \triangleq \max_{\pi\in\Pi} Q_g^\pi(s,a)$. Previous work have shown that using contrastive leaning (Eysenbach et al., 2022; van den Oord et al., 2019) can recover the behavior distance, but not the optimal distance.

Prior work on MC learning (Myers et al., 2024; Eysenbach et al., 2022; 2021) has demonstrated that instead of only using the end of trajectories as the goal state, we sample future states via a geometric distribution as described in Eq. (3). This allows us to classify any future state in trajectory as goals, providing a robust way of learning goal-reaching policies.

$$\mathfrak{s}_t^+ \triangleq \mathfrak{s}_{t+K}, K \sim \mathrm{Geom}(1-\gamma). \tag{3}$$

Quasimetric distance representations. Traditionally, offline RL algorithms use neural networks to represent the critic and value function $Q_g(s,a)$ and $V_g(s)$ (Kumar et al., 2020; Kostrikov et al., 2022). Separately, other works have been using dot products $Q(s,a,g) \triangleq \varphi(s,a)^\intercal\psi(g)$ (Eysenbach et al., 2022; Zheng et al., 2024) or geometric norms $||\varphi(s) - \psi(g)||_k$ for suitable values of k (Eysenbach et al., 2024; Park et al., 2024c). We use quasimetric architectures (Wang et al., 2023; Pitis et al., 2020) to parameterize our value-functions, which by construction (*i*): positivity ($d(x,y) > 0$, (*ii*): triangle inequality, and (*iii*): identity.

To enforce the quasimetric properties of successor distances, we use Metric Residual Networks (MRN) (Liu et al., 2023) that directly operate on vector-based representations, which is defined by Eq. (4). MRN splits the representation into N equally sized ensembles, and in each part, takes the sum of the asymmetric component (maximum of ReLU) and a symmetric component (l_2 norm) of the difference between the two embeddings x and y. For simplicity, we denote $d(x,y)$ as $d_{\mathrm{MRN}}(x,y)$ in the following sections.

$$d_{\text{MRN}}(x, y) \triangleq \frac{1}{N} \sum_{k=1}^{N} \max_{m=1\ldots M} \text{ReLU}(x_{kM+m} - y_{kM+m}) + ||x_{kM+m} - y_{kM+m}||_2 \tag{4}$$

4 MULTISTEP QUASIMETRIC ESTIMATION

In this section, we develop the framework of MQE based on quasimetric architectures, which can be broken down into 3 parts: (1) propagating the learned temporal distance locally using a multistep return with geometrically sampled future states as waypoints, (2) imposing additional objectives that need to be satisfied to learn Q^*, and (3) practical implementation details for extracting the goal-reaching policy. To the best of our knowledge, MQE is the **first** method to effectively use multistep returns under quasimetric distance representations, and achieve superior results to previous methods that use TD returns, contrastive learning for value estimation, or hierarchical methods.

Definitions. Argued by (Myers et al., 2024), define the relations between the distance metric and the underlying Q/value function of the environment:

$$Q_g(s, a) \triangleq \exp(-d(\varphi(s, a), \psi(g))), \qquad V_g(s) \triangleq \exp(-d(\psi(s), \psi(g))). \tag{5}$$

We also adopt the following shorthand :

$$d((s, a), g) \triangleq d(\varphi(s, a), \psi(g)), \qquad d(s, g) \triangleq d(\psi(s), \psi(g)). \tag{6}$$

4.1 MULTISTEP RETURNS WITH QUASIMETRIC ARCHITECTURE

The first design principle of MQE is to use multistep returns under a quasimetric architecture. To that end, we start with the fitted one-step Q iteration that operates on quasimetric distance representations, in which $\rightarrow$ denotes regressing from the LHS to the RHS:

$$e^{-d((s,a),g)} \rightarrow \mathbb{E}_{\{(s,a,s')\}\sim\mathcal{D}}[\gamma \cdot e^{-d(s',g)}]. \tag{7}$$

This is similar to optimizing the critic in IQL (Kostrikov et al., 2022), which recovers V^* in goal-conditioned settings, the main difference being that we use a quasimetric architecture to represent the Q and value function. We may omit the reward signal, as the value function V is defined to have a value of 1 when $s = g$, abstracting away the need of a reward. We denote this regression objective as $\mathcal{T}$.

Our insight for the $\mathcal{T}$ objective described in Eq. (7) is that, instead of applying this invariance to only the future state $s' \sim \text{P}(s'|s, a)$, we can extend this principle to any state between the current state and goal. This transforms a one-step optimization into a on-policy multistep optimization procedure. To do so, we first define the shorthand $\mathfrak{s}_t^w$, which refers to a "**waypoint**" between the current state and goal state. Empirically, we find that using a geometric distribution capped at the index of the future state work the best.

$$\mathfrak{s}_t^w \triangleq \mathfrak{s}_{t+k'}, k' \sim \min(\text{Geom}(1 - \lambda), K) \tag{8}$$

We now optimize the same objective as in Eq. (7), but across any such waypoint we sample. To account for the multistep nature of this objective, we modify Eq. (7) below to accommodate such changes.

$$e^{-d((s,a),g)} \rightarrow \mathbb{E}_{\{(s_t,a_t),s_t^w\}\sim\mathcal{D}}[\gamma^{k'} \cdot e^{-d(s_t^w,g)}]. \tag{9}$$

We denote this new objective as $\mathcal{T}_\beta$, as sampled future state is constrained by π_β. This is indeed similar to the n-step returns we see in RL (Munos et al., 2016; Li et al., 2025), although we do not sample future states with a fixed number of steps. In practice, we can use any loss function to make the LHS equal to the RHS (in expectation) concerning Eq. (7) and Eq. (9). We use a form of Bregman divergence, as it does not incur vanishing gradients when the two distance have become close in value, and regresses d to d' in expectation (Banerjee et al., 2004).

$$D_T(d, d') \triangleq \exp(d - d') - d' \tag{10}$$

By using this loss, we can concretely define both $\mathcal{L}_{\mathcal{T}_\beta}$ and $\mathcal{L}_{\mathcal{T}}$ that can optimize $\mathcal{T}$ and $\mathcal{T}_\beta$ objectives:

$$\mathcal{L}_{\mathcal{T}_\beta}\left(\phi, \psi; \{s_i, a_i, s_i^w, g_i\}_{i=1}^N\right) = \sum_{i=1}^N \sum_{j=1}^N D_T\big(d((s_i, a_i), g_j), d(s_i^w, g_j)) - k' \log \gamma\big). \tag{11}$$

$$\mathcal{L}_{\mathcal{T}}\left(\phi, \psi; \{s_i, a_i, s_i', g_i\}_{i=1}^N\right) = \sum_{i=1}^N \sum_{j=1}^N D_T\big(d((s_i, a_i), g_j), d(s_i', g_j)) - \log \gamma\big). \tag{12}$$

These two losses, when applied to our quasimetric network, will allow us to propagate the value in either an one-step or multistep manner.

4.2 Enforcing Optimality Constraints in Distance Representation

Remark: $\mathcal{L}_{\mathcal{T}_\beta}$ (weakly) enforces $\mathcal{L}_{\mathcal{T}}$. While optimizing $\mathcal{L}_{\mathcal{T}_\beta}$ allows us to learn a multistep backup under the quasimetric architecture, it is *biased* towards the behavior policy when $k' \neq 1$, as the state on the waypoint $\mathfrak{s}_t^w$ must be conditioned on the trajectories made by the behavior policy. This is in contrast to the one-step objective described in Eq. (7), which allows us to learn the optimal value function in theory (Myers et al., 2024; Kostrikov et al., 2022), but is hard to use only $\mathcal{L}_{\mathcal{T}}$ in practice due to the extremely local nature of the one-step value propagation (Myers et al., 2025b).

The geometric distribution provides some remedy, as $k' = 1$ is the most likely outcome of sampling. As a result, we can slightly enforce $\mathcal{L}_{\mathcal{T}}$, as some of the sampled waypoints will be equal to s_t'. We discuss ways to manually increase the weighting of $\mathcal{L}_{\mathcal{T}}$ in Section 4.4.

Enforcing the action invariance property. We also note that the optimal Q and value function observe the below property (Sutton & Barto, 2018):

$$V_g^*(s) \propto \max_{a \in \mathcal{A}} Q_g^*(s, a). \tag{13}$$

This can be satisfied, under our construction, if $\psi(s) = \varphi(s, a), \forall a \in \mathcal{A}$ (Myers et al., 2025b) (action invariance). This can be done with the same MRN distance formulation due to the identity property of the quasimetric network, as described in Eq. (14).

$$d(\psi(s), \varphi(s, a)) \to 0 \tag{14}$$

While this objective can be optimized by taking the gradient with just the MRN distance, we modify the action invariance such that we use the sum of squares of the MRN distance between s and (s, a). The insight is that now the loss will scale with the magnitude of such deviation, which removes the need of hyperparameter tuning for an appropriate multiplier as well as stabilizing training dynamics. We can now formally define $\mathcal{L}_{\mathcal{I}}$ objective as:

$$\mathcal{L}_{\mathcal{I}}\left(\varphi, \psi; \{s_i, a_i\}_{i=1}^N\right) = \sum_{i,j=1}^N \left(d\big(\psi(s_i), \varphi(s_i, a_j)\big)\right)^2. \tag{15}$$

4.3 Policy Extraction

We extract the goal-conditioned policy $\pi(s, g) : \mathcal{S}^2 \to \mathcal{A}$ using learned distance by using Behavior-Regularized Deep Deterministic Policy Gradient (DDPG+BC) (Fujimoto & Gu, 2021):

$$\mathcal{L}_\mu(\pi; \{s_i, a_i, g_i\}_{i=1}^N) = \mathbb{E}\Big[\big[\sum_{i,j=1}^N d\big((s_i, \pi(s_i, g_j)), g_j\big)\big] - \alpha \log \pi(a_i | s_i, g_i)\Big]. \tag{16}$$

Given that a smaller d measure correspond to a higher Q value, we can maximize the Q values by minimizing the distance produced by our quasimetric network. We tune the BC coefficient α per environment. We provide more hyperparameter details in Appendix B.

4.4 Implementation Details & Algorithm

Although $\mathcal{L}_{\mathcal{T}_\beta}$ becomes $\mathcal{L}_{\mathcal{T}}$ in the case where $k' = 1$, we can adjust the scale of the two losses by explicitly up-weighting the probability of sampling the next state as waypoint via a Bernoulli distribution. In practice, we designate $\mathfrak{s}_t^w = s'_t$ with probability p, and with probability $1 - p$, we designate $\mathfrak{s}_t^w$ as is. In this instance, if we designate $p = 1$, $\mathcal{L}_{\mathcal{T}_\beta}$ becomes $\mathcal{L}_{\mathcal{T}}$. We find using $p = 0.2$ across all environments allows the network to both propagate local distances fast and maintain the ability to optimize towards Q^* using one-step consistency.

We concisely define our final learning objective in Algorithm 1. Unlike previous works in hierarchical RL (Nachum et al., 2018; Park et al., 2024b), we randomly sample these waypoints, and we learn a single critic $\mathcal{Q}$ that operates on $\mathcal{S}$ and $\mathcal{A}$ and a single goal-reaching policy π_μ. As a result, our method does not contain any hierarchical components and is simpler to implement than other horizon reduction methods.

Algorithm 1: Multistep Quasimetric Reinforcement Learning

Require: Dataset $\mathcal{D}$, Batch size B, training iteration T, Probability p

1: Initialize Quasimetric network $\mathcal{Q}$ with parameters $(\varphi(s, a), \psi(s))$, goal-reaching policy π_μ
2: **for** $t = 1 ... T$ **do**
3: Sample $\{s_i, a_i, s'_i, \tilde{s}_i^w, g_i\}_{i=1}^B \sim \mathcal{D}$ (Eq. (8))
4: For each element in batch, choose $s_i^w \sim \begin{cases} s'_i & \text{with probability } p, \\ \tilde{s}_i^w & \text{with probability } 1 - p. \end{cases}$
5: Update $\mathcal{Q}$ with multistep backup by minimizing $\mathcal{L}_{\mathcal{T}_\beta}(\varphi, \psi; \{s_i, a_i, s_i^w\}_{i=1}^B)$ (Eq. (9))
6: Update $\mathcal{Q}$ with optimality constraints by minimizing $\mathcal{L}_{\mathcal{I}}(\varphi, \psi; \{s_i, a_i\}_{i=1}^B)$ (Eq. (15))
7: Update policy π_μ with DDPG+BC by minimizing $\mathcal{L}_\mu(\pi_\mu; \{s_i, a_i, g_i\}_{i=1}^B)$ (Eq. (16))
8: **return** π_μ

5 Experiments

Our goal of experiments is to understand the benefits MQE brings when it comes to enabling a policy to generalize compositionally (execute multiple tasks seen in training separately together) and in terms of horizon (generalize over a longer task when a similar but shorter task was seen in training set). To that end, we ask the following research questions:

1. How much does MQE improve the horizon generalization abilities of agents compared to prior works?
2. What qualitative improvements does MQE bring in terms of compositional generalization?
3. What are the important hyperparameters to ensure the success of MQE?

Experiment setup Our experiments use challenging, long-horizon problems in offline RL benchmarks as well as real-world settings. We use OGBench (Park et al., 2025a) for our experiments on simulated benchmarks and the BridgeData setup (Walke et al., 2024) for our real-world evaluation.

5.1 Evaluating MQE on Offline GCRL Benchmark

We evaluate MQE in both locomotion and manipulation in OGBench (Park et al., 2025a). For locomotion tasks, in addition to the three standard sized mazes, we also designed a "colossal"-sized maze. This maze is 50% larger than that of the "giant"-sized mazes currently available on OGBench, and it requires as many as 4000 steps for an agent to traverse through the entire maze (see Fig 2). We employ 13 state-based environments and 5 pixel-based environments (each pixel-based environment takes in a $64 \times 64 \times 3$ observation) with 5 tasks each, bringing a total of **90** tasks to evaluate in our OGBench setup.

We compare against the following baselines: GCIQL (Kostrikov et al., 2022), CRL (Eysenbach et al., 2022), QRL (Wang et al., 2023), HIQL (Park et al., 2024b), nSAC+BC (Park et al., 2025b; Haarnoja

scene-play visual-antmaze-giant antmaze-colossal puzzle-4x4

Figure 2: We provide visualizations of tasks from various state and pixel-based environments for OGBench. Note that `Antmaze-colossal` is 50% larger than any other mazes available on OGBench, and in `stitch` datasets, test the agent's ability to generalize over a horizon that is up to **1000%** larger.

Table 1: OGBench Evaluation

Dataset	Methods							
	MQE	TMD	CMD	CRL	QRL	GCIQL	HIQL	n-SAC+BC
`pointmaze_giant_navigate`	**72.8**$^{(\pm 2.5)}$	39.9$^{(\pm 5.2)}$	45.3$^{(\pm 3.7)}$	27.4$^{(\pm 3.4)}$	**68.5**$^{(\pm 2.8)}$	0.0$^{(\pm 0.0)}$	45.9$^{(\pm 3.0)}$	0.0$^{(\pm 0.0)}$
`pointmaze_giant_stitch`	28.3$^{(\pm 1.1)}$	9.1$^{(\pm 1.0)}$	8.1$^{(\pm 0.6)}$	0.0$^{(\pm 0.0)}$	**49.7**$^{(\pm 2.3)}$	0.0$^{(\pm 0.0)}$	0.0$^{(\pm 0.0)}$	0.4$^{(\pm 0.1)}$
`antmaze_large_explore`	**44.1**$^{(\pm 4.2)}$	0.9$^{(\pm 0.2)}$	0.8$^{(\pm 0.3)}$	0.3$^{(\pm 0.1)}$	0.0$^{(\pm 0.0)}$	0.4$^{(\pm 0.1)}$	3.9$^{(\pm 1.8)}$	0.2$^{(\pm 0.1)}$
`antmaze_giant_stitch`	**35.1**$^{(\pm 1.7)}$	2.7$^{(\pm 0.6)}$	2.0$^{(\pm 0.5)}$	0.0$^{(\pm 0.0)}$	0.4$^{(\pm 0.2)}$	0.0$^{(\pm 0.0)}$	1.8$^{(\pm 0.6)}$	9.2$^{(\pm 2.1)}$
`antmaze_colossal_navigate`	**48.6**$^{(\pm 2.4)}$	22.3$^{(\pm 1.1)}$	22.5$^{(\pm 3.1)}$	14.6$^{(\pm 1.8)}$	0.0$^{(\pm 0.0)}$	0.0$^{(\pm 0.0)}$	0.0$^{(\pm 0.0)}$	0.3$^{(\pm 0.1)}$
`antmaze_colossal_stitch`	**27.6**$^{(\pm 2.9)}$	0.0$^{(\pm 0.0)}$	0.2$^{(\pm 0.1)}$	0.0$^{(\pm 0.0)}$	0.0$^{(\pm 0.0)}$	0.0$^{(\pm 0.0)}$	0.0$^{(\pm 0.0)}$	0.5$^{(\pm 0.3)}$
`humanoidmaze_giant_navigate`	**46.5**$^{(\pm 2.5)}$	9.2$^{(\pm 1.1)}$	5.0$^{(\pm 0.8)}$	0.7$^{(\pm 0.1)}$	0.4$^{(\pm 0.1)}$	0.5$^{(\pm 0.1)}$	12.5$^{(\pm 1.5)}$	3.2$^{(\pm 0.5)}$
`humanoidmaze_giant_stitch`	**26.5**$^{(\pm 1.3)}$	6.3$^{(\pm 0.6)}$	0.2$^{(\pm 0.1)}$	1.5$^{(\pm 0.5)}$	0.4$^{(\pm 0.1)}$	1.5$^{(\pm 0.1)}$	3.3$^{(\pm 0.7)}$	1.7$^{(\pm 0.1)}$
`cube_double_play`	35.6$^{(\pm 1.9)}$	13.1$^{(\pm 2.3)}$	0.2$^{(\pm 0.1)}$	1.5$^{(\pm 0.5)}$	0.4$^{(\pm 0.1)}$	**40.2**$^{(\pm 1.7)}$	6.4$^{(\pm 0.7)}$	19.1$^{(\pm 0.3)}$
`cube_triple_noisy`	**3.9**$^{(\pm 0.8)}$	2.1$^{(\pm 0.6)}$	1.5$^{(\pm 0.5)}$	2.7$^{(\pm 0.5)}$	**3.4**$^{(\pm 0.4)}$	1.8$^{(\pm 0.2)}$	2.6$^{(\pm 0.4)}$	1.4$^{(\pm 0.3)}$
`puzzle_4x4_play`	18.7$^{(\pm 2.3)}$	10.0$^{(\pm 1.4)}$	0.2$^{(\pm 0.1)}$	1.5$^{(\pm 0.5)}$	0.4$^{(\pm 0.1)}$	**25.7**$^{(\pm 1.1)}$	7.4$^{(\pm 0.7)}$	11.4$^{(\pm 0.9)}$
`scene_play`	**69.1**$^{(\pm 2.1)}$	21.9$^{(\pm 1.9)}$	1.2$^{(\pm 0.4)}$	18.6$^{(\pm 0.8)}$	5.4$^{(\pm 0.3)}$	51.3$^{(\pm 1.5)}$	38.2$^{(\pm 0.9)}$	17.6$^{(\pm 1.4)}$
`scene_noisy`	**30.8**$^{(\pm 1.6)}$	19.6$^{(\pm 1.7)}$	4.0$^{(\pm 0.7)}$	1.2$^{(\pm 0.3)}$	9.1$^{(\pm 0.7)}$	25.9$^{(\pm 0.8)}$	25.2$^{(\pm 1.3)}$	19.1$^{(\pm 2.2)}$
`visual_scene_play`	27.2$^{(\pm 3.1)}$	20.7$^{(\pm 2.5)}$	16.1$^{(\pm 2.2)}$	9.6$^{(\pm 0.6)}$	5.4$^{(\pm 0.3)}$	12.2$^{(\pm 0.8)}$	**49.9**$^{(\pm 0.6)}$	7.1$^{(\pm 1.2)}$
`visual_cube_triple_play`	**19.8**$^{(\pm 0.9)}$	**17.9**$^{(\pm 1.3)}$	**18.9**$^{(\pm 1.1)}$	16.9$^{(\pm 1.1)}$	16.3$^{(\pm 0.3)}$	15.2$^{(\pm 0.6)}$	**21.0**$^{(\pm 0.2)}$	**21.1**$^{(\pm 2.4)}$
`visual_cube_double_noisy`	25.9$^{(\pm 1.6)}$	14.2$^{(\pm 1.3)}$	0.3$^{(\pm 0.3)}$	6.0$^{(\pm 1.4)}$	6.1$^{(\pm 1.2)}$	21.6$^{(\pm 0.9)}$	**59.4**$^{(\pm 1.6)}$	22.7$^{(\pm 1.1)}$
`visual_puzzle_4x4_play`	11.3$^{(\pm 1.6)}$	9.8$^{(\pm 3.6)}$	7.2$^{(\pm 0.4)}$	9.6$^{(\pm 3.2)}$	0.0$^{(\pm 0.0)}$	16.2$^{(\pm 2.2)}$	**60.1**$^{(\pm 20.4)}$	10.3$^{(\pm 2.6)}$
`visual_antmaze_giant_stitch`	**26.9**$^{(\pm 3.1)}$	14.5$^{(\pm 2.5)}$	**22.3**$^{(\pm 1.9)}$	0.1$^{(\pm 0.1)}$	0.0$^{(\pm 0.0)}$	0.0$^{(\pm 0.0)}$	0.2$^{(\pm 0.1)}$	7.6$^{(\pm 1.1)}$
`Overall`	**32.3**$^{(\pm 0.5)}$	13.0$^{(\pm 0.5)}$	8.7$^{(\pm 0.3)}$	6.2$^{(\pm 0.3)}$	9.8$^{(\pm 0.3)}$	11.8$^{(\pm 0.2)}$	18.7$^{(\pm 1.2)}$	8.5$^{(\pm 0.3)}$

We **bold** the best performance. Success rate (%) is presented with the standard error across eight seeds for state-based environments and four seeds for pixel-based environments. All datasets contain 5 separate tasks each. We record the aggregate across all 5 tasks.

et al., 2018), CMD (Myers et al., 2024), and TMD (Myers et al., 2025b). These methods use either only TD learning (GCIQL, QRL, nSAC+BC), MC value estimation via contrastive learning (CRL, CMD, TMD), use horizon reduction techniques (nSAC+BC with value horizon, HIQL with policy horizon), or use a quasimetric architecture for distance learning (QRL, CMD, TMD). We detail how these methods are implemented in Appendix B. By comparing against these methods, we can gain a better outlook on *what* advantage MQE has over other works that learn distances only locally, globally, or in a hierarchical manner.

Table 1 shows the performance of MQE across state- and pixel-based environments on OGBench. Overall, MQE performed at a much better level compared to all baselines. We note the significant gains over previous methods on many extremely challenging locomotion environments, and that in manipulation environments, where GCIQL dominates the performance, MQE still closes the gap considerably, and provided the best performance over in the `scene` environment. We also demonstrate that MQE is able to extract better goal-reaching policies from both `explore` and `noisy` datasets, which for methods that only obtained Q_β, was difficult for solve, showing that MQE is capable of extracting optimal policies from suboptimal demonstrations.

In visual environments, MQE maintains its strong empirical performance, only being bested by HIQL, which does explicit policy horizon reduction (Park et al., 2025b) and was outperformed by other methods in state-based environments due to HIQL learning an additional subgoal representation that helps to stabilize training. Nevertheless, MQE is still able to outperform all other methods, including n-SAC+BC, which also conducts horizon reduction.

5.2 EVALUATING MQE IN REAL WORLD

While the OGBench environments focus on learning long-horizon tasks using mixed quality data in a single environment, we can use the real-world BridgeData tasks evaluate a more sophisticated kind of compositionality: the BridgeData tasks consist of individual object manipulation primitives (e.g., picking up a banana and placing it on a plate), and our evaluation tasks are significantly longer, requiring the composition of multiple tasks in the dataset (e.g., placing four different objects on the plate) (Walke et al., 2024). Critically, the dataset *does not contain any example trajectories* that compose multiple tasks in this way. Accomplishing this sort of temporal composition is an important goal in offline RL, because it allows "stitching" long behaviors out of shorter chunks. However, most current methods struggle with this challenge, especially in real-world domains with many objects, scenes, and scenarios, where this sort of stitching necessarily requires both temporal compositionality and generalization.

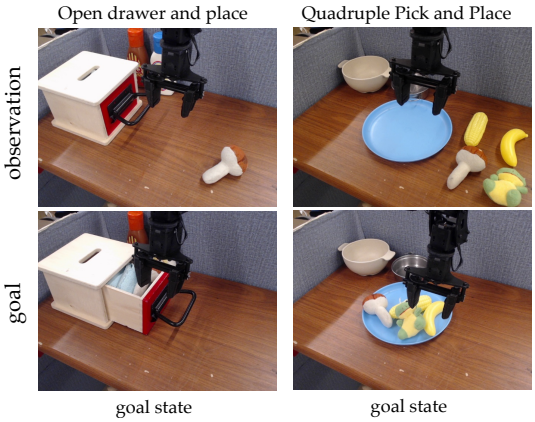

Figure 3: We evaluate MQE on multi-stage manipulation tasks on BridgeData. Below are examples of the starting observations and goal image being passed in.

As a result, a policy that is truly able to perform compositional generalization should be able to compose multiple tasks at once without external guidance. We designed the tasks on BridgeData with this specific intention. Instead of tasking the policy to complete a single pick and place (abbreviated as PnP), we evaluate a policy's performance with PnP of up to 4 objects, which, to our knowledge, **has never been completed without the use of hierarchical policies or high level planners**. We also evaluate the policy on tasks requiring dependencies, with the policy being tasked with opening a drawer and then placing the item within the drawer all conditioned by one image of an opened drawer with an item inside. Such a task is also challenging, as **only one previous work** (Myers et al., 2025a) has shown a non-zero success rate when using an end-to-end policy to the best of our knowledge.

We use a 6DoF, 5Hz, WidowX250 manipulator for our robot learning tasks and we train and deploy a policy $\pi(a|s, g)$ conditioned on observations and goal images. We compare against the following methods: ding2019goal, GCIQL, and TRA (Myers et al., 2025a). GCIQL follows the same structure as OGBench training, and both ding2019goal and TRA is a successor feature learning method that aligns present and future representations for compositional generalization. TRA is designed for following both goal images and language instructions, but since we only use goal images as the modality, we denote TRA-g as the specific modality we're testing. We provide more details on policy training in Appendix C.1.

As in (Black et al., 2024), we use task progress to measure the effectiveness of these policies due to the long-horizon nature of these tasks. We detail more on the experimental setup and how we assign these points in Appendix C.1.

What qualitative improvements does MQE provide in terms of compositional generalization? Fig. 4 reports the *overall task progress* on 2 single-stage tasks and 4 tasks requiring compositionality. We provide both the binary success rate and further analysis of policy rollout in Appendix C. Here, we observe that while MQE helps with single-stage tasks (single PnP, open drawer) against ding2019goal, both TRA and GCIQL can still perform competitively. However, as the number of tasks needed to be performed in sequence increases, we see that MQE is able to retain a relatively high task progress, while both GCIQL and TRA-g's performance regressed.

Taking a look at the two most difficult tasks, we have `quadruple PnP` and `drawer open and place`. These tasks are the most challenging since quadruple PnP required the agent to reason 4 consecutive primitives together, and drawer open and place required the agent to complete the first task (open the drawer) before completing the second task (putting the mushroom in the drawer).

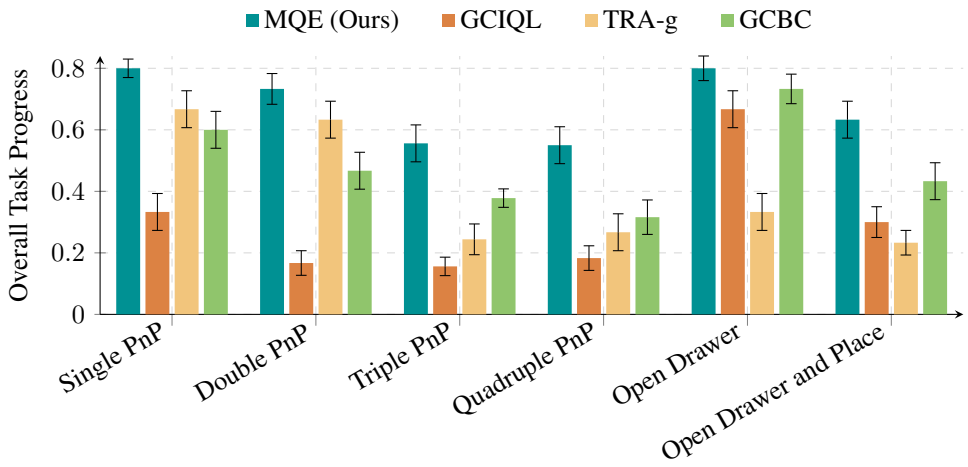

Figure 4: Task progress on BridgeData tasks; plotted with both the mean and the standard error bars.

Among all methods that we have tested, only MQE and TRA-g displayed positive success rate, as demonstrated in Table 5. We provide more details on policy rollouts in Appendix C.

5.3 Ablation Studies

In this section, we explore design choices involved for MQE. Given the two components needed for critic loss and an additional sampling of waypoints, it is natural for one to ask whether they are conflicting objectives and *how* they contribute to the strong empirical observation. To that end, we investigate design decisions for MQE, and ask the following questions:

- What is the best empirical distribution for sampling the waypoint $\mathfrak{s}_t^w$?
- What role does the invariance objective $\mathcal{I}$ play in MQE?
- What is the relationship between λ and p, the two hyperparameters that enforce $\mathcal{L}_{\mathcal{T}_\beta}$ and $\mathcal{L}_\mathcal{T}$ separately?

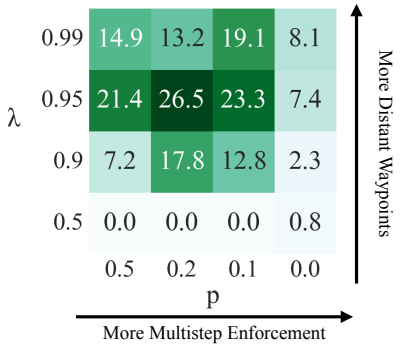

Figure 5: Sucess rate of MQE on `humanoidmaze_giant_stitch` using $\alpha = 0.01$, averaged over 4 seeds.

Fig. 5 provides an illustration of success rate over pairs of (p, λ). The figure suggests that both hyperparameters need to be relatively high in value, which indicates that MQE needs: (1) a far enough waypoint for value to quickly propagate, and (2) a high enough p to ensure that local consistencies are being respected. This is promising, as we do not need to decide to decide a trade-off between the two hyperparameters, and these two hyperparameters are in fact not mutually interfering.

We provide additional, more minor ablations in Appendix E.

6 Conclusion

We introduced Multistep Quasimetric Estimation (MQE), a novel method that combines the benefits of fast value propagation via multistep backup and the global constraint of quasimetric distances. MQE is able to solve extremely challenging and long-horizon tasks in simulated benchmarks as well as in real life.

Limitations and Future Work. While MQE achieves strong performance, we sample the waypoints based on empirical analyses. This could incur more computational costs when finding the

optimal way of sampling the waypoint for environments that are outside of our evaluation range. Future work can investigate the theoretical connection between sampling waypoints and successor distances, investigate the effect of such policy learning on different policy classes such as autoregressive (i.e. transformer) or flow policies, and apply the same method across methods beyond offline RL in scenarios such as offline-to-online RL.

Ethics Statement In this work, we studied ways in which we enable a policy to better perform compositional and horizon generalization in reinforcement learning contexts. While our implementations in this paper do not constitute real-world effects due to the smaller networks we have chosen, future works using larger and more expressive policies should pay attention to the potential failure mode of our algorithm and examine the interpretability of the learned policy.

In this work, we ran the text through LLMs to fix grammar and spelling mistakes. We did not use LLMs to generate any content in the paper that were not already ideated and fully drafted by the authors.

Reproducibility Statement To ensure reproducibility, we have uploaded our implementation on OGBench as supplementary material, and have stated the relevant hyperparameters needed to reproduce the results in the experiment section.

REFERENCES

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient Online Reinforcement Learning With Offline Data. arXiv:2302.02948, 2023. URL https://arxiv.org/abs/2302.02948.

Arindam Banerjee, Srujana Merugu, Inderjit Dhillon, and Joydeep Ghosh. Clustering With Bregman Divergences. In *SIAM International Conference on Data Mining*, pp. 234–245. Society for Industrial and Applied Mathematics, April 2004. ISBN 978-0-89871-568-2 978-1-61197-274-0. doi: 10.1137/1.9781611972740.22. URL https://epubs.siam.org/doi/10.1137/1.9781611972740.22.

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. arXiv:2410.24164, 2024. URL https://arxiv.org/abs/2410.24164.

Peter Dayan. Improving Generalisation for Temporal Difference Learning: The Successor Representation. *Neural Computation*, (4):613–624, 1993. doi: 10.1162/neco.1993.5.4.613.

Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowledge Insulating Vision-Language-Action Models: Train Fast, Run Fast, Generalize Better. arXiv:2505.23705, 2025. URL https://arxiv.org/abs/2505.23705.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-Learning: Learning to Achieve Goals via Recursive Classification. In *International Conference on Learning Representations*. arXiv, 2021. URL https://arxiv.org/abs/2011.08909.

Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. Contrastive Learning as Goal-Conditioned Reinforcement Learning. In *Neural Information Processing Systems*, volume 35, pp. 35603–35620. arXiv, 2022. URL https://arxiv.org/abs/2206.07568.

Benjamin Eysenbach, Vivek Myers, Ruslan Salakhutdinov, and Sergey Levine. Inference via Interpolation: Contrastive Representations Provably Enable Planning and Inference. In *Neural Information Processing Systems*, December 2024. URL https://arxiv.org/abs/2403.04082.

Scott Fujimoto and Shixiang Shane Gu. A Minimalist Approach to Offline Reinforcement Learning. *Neural Information Processing Systems*, 34:20132–20145, 2021. URL https://arxiv.org/abs/2106.06860.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning With a Stochastic Actor. arXiv:1801.01290, 2018. URL https://arxiv.org/abs/1801.01290.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline Reinforcement Learning With Implicit Q-Learning. In *International Conference on Learning Representations*. arXiv, 2022. URL http://arxiv.org/abs/2110.06169.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-Learning for Offline Reinforcement Learning. arXiv:2006.04779, 2020. URL https://arxiv.org/abs/2006.04779.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement Learning With Action Chunking. arXiv:2507.07969, 2025. URL https://arxiv.org/abs/2507.07969.

Bo Liu, Yihao Feng, Qiang Liu, and Peter Stone. Metric Residual Networks for Sample Efficient Goal-Conditioned Reinforcement Learning. arXiv:2208.08133, January 2023. URL https://arxiv.org/abs/2208.08133.

Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. SERL: A Software Suite for Sample-Efficient Robotic Reinforcement Learning. arXiv:2401.16013, 2025. URL https://arxiv.org/abs/2401.16013.

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. arXiv:2412.06685, December 2024. URL https://arxiv.org/abs/2412.06685.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari With Deep Reinforcement Learning. arXiv:1312.5602, 2013. URL https://arxiv.org/abs/1312.5602.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and Efficient Off-Policy Reinforcement Learning. arXiv:1606.02647, 2016. URL https://arxiv.org/abs/1606.02647.

Vivek Myers, Chongyi Zheng, Anca Dragan, Sergey Levine, and Benjamin Eysenbach. Learning Temporal Distances: Contrastive Successor Features Can Provide a Metric Structure for Decision-Making. In *International Conference on Machine Learning*, 2024. URL https://arxiv.org/pdf/2406.17098.

Vivek Myers, Bill Chunyuan Zheng, Anca Dragan, Kuan Fang, and Sergey Levine. Temporal Representation Alignment: Successor Features Enable Emergent Compositionality in Robot Instruction Following. arXiv:2502.05454, 2025a. URL https://arxiv.org/abs/2502.05454.

Vivek Myers, Bill Chunyuan Zheng, Benjamin Eysenbach, and Sergey Levine. Offline Goal Conditioned Reinforcement Learning With Temporal Distance Representations. In *Neural Information Processing Systems*, 2025b. URL https://tmd-website.github.io/static/pdf/myers2025offline.pdf.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-Efficient Hierarchical Reinforcement Learning. arXiv:1805.08296, 2018. URL https://arxiv.org/abs/1805.08296.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering Your Generalists: Improving Robotic Foundation Models via Value Guidance. arXiv:2410.13816, 2025. URL https://arxiv.org/abs/2410.13816.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, et al. Open X-Embodiment: Robotic Learning Datasets and RT-X Models. In *International Conference on Robotics and Automation*. arXiv, May 2024. URL http://arxiv.org/abs/2310.08864.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is Value Learning Really the Main Bottleneck in Offline RL? arXiv:2406.09329, June 2024a. URL https://arxiv.org/abs/2406.09329.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. HIQL: Offline Goal-Conditioned RL With Latent States as Actions. arXiv:2307.11949, 2024b. URL https://arxiv.org/abs/2307.11949.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation Policies With Hilbert Representations. arXiv:2402.15567, 2024c. URL https://arxiv.org/abs/2402.15567.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking Offline Goal-Conditioned RL. In *International Conference on Learning Representations*. arXiv, 2025a. URL https://arxiv.org/abs/2410.20092.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon Reduction Makes RL Scalable. arXiv:2506.04168, 2025b. URL https://arxiv.org/abs/2506.04168.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual Reasoning With a General Conditioning Layer. arXiv:1709.07871, 2017. URL https://arxiv.org/abs/1709.07871.

Silviu Pitis, Harris Chan, Kiarash Jamali, and Jimmy Ba. An Inductive Bias for Distances: Neural Nets That Respect the Triangle Inequality. arXiv:2002.05825, 2020. URL https://arxiv.org/abs/2002.05825.

Younggyo Seo, Carmelo Sferrazza, Haoran Geng, Michal Nauman, Zhao-Heng Yin, and Pieter Abbeel. FastTD3: Simple, Fast, and Capable Reinforcement Learning for Humanoid Control. arXiv:2505.22642, 2025. URL https://arxiv.org/abs/2505.22642.

Laura Smith, Yunhao Cao, and Sergey Levine. Grow Your Limits: Continuous Improvement With Real-World RL for Robotic Locomotion. arXiv:2310.17634, 2023. URL https://arxiv.org/abs/2310.17634.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Dong Tian, Ge Li, Hongyi Zhou, Onur Celik, and Gerhard Neumann. Chunking the Critic: A Transformer-Based Soft Actor-Critic With N-Step Returns. arXiv:2503.03660, 2025. URL https://arxiv.org/abs/2503.03660.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning With Contrastive Predictive Coding. arXiv:1807.03748, 2019. URL https://arxiv.org/abs/1807.03748.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering Your Diffusion Policy With Latent Space Reinforcement Learning. arXiv:2506.15799, 2025. URL https://arxiv.org/abs/2506.15799.

Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. BridgeData V2: A Dataset for Robot Learning at Scale. arXiv:2308.12952, 2024. URL https://arxiv.org/abs/2308.12952.

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal Goal-Reaching Reinforcement Learning via Quasimetric Learning. arXiv:2304.01203, 2023. URL https://arxiv.org/abs/2304.01203.

C. J. C. H. watkins1989learning. Learning From Delayed Rewards, 1989.

Charles Xu, Qiyang Li, Jianlan Luo, and Sergey Levine. RLDG: Robotic Generalist Policy Distillation via Reinforcement Learning. arXiv:2412.09858, 2024. URL https://arxiv.org/abs/2412.09858.

Chongyi Zheng, Benjamin Eysenbach, Homer Walke, Patrick Yin, Kuan Fang, Ruslan Salakhutdinov, and Sergey Levine. Stabilizing Contrastive RL: Techniques for Robotic Goal Reaching From Offline Data. In *International Conference on Learning Representations*. arXiv, 2024. URL https://arxiv.org/abs/2306.03346.

A Website and Code

We provide the full implementation of MQE, TMD, CMD, and the new antmaze-colossal mazes on OGBench https://anonymous.4open.science/r/ogbench-D4D9/. We provide website for MQE at https://anonymous.4open.science/w/mqe-paper-686D/.

B OGBench Experiment Details

B.1 Baseline Details

Here, we briefly describe the inner workings of each baseline on OGBench.

GCIQL Goal-conditioned Implicit Q-learning (GCIQL) uses expectile regression to learn a value function. **QRL** Quasimetric RL learns a quasimetric distance using bootstrapping under a quasimetric architecture, and constrains the distance with one-step cost in *deterministic* settings. **CRL** Contrastive RL (CRL) uses binary cross entropy to regress the critic (defined as a dot product) towards goals that are future states and repel those that are not. **CMD** Contrastive Metric Distillation (CMD) uses InfoNCE loss (van den Oord et al., 2019) to recover Q_β. **TMD** Temporal Metric Distillation (TMD) (Myers et al., 2025b) use contrastive learning to learn the behavior Q-function, and then tightens the bound to optimize towards Q^*. CMD and TMD enforce the quasimetric property of successor distance architecturally.

Additionally, we compare MQE against two horizon reduction methods, n-step Goal-Conditioned Soft Actor-Critic with Behavior Cloning (n-SAC+BC) and Hierarchical Implicit Q learning (HIQL), which explicitly reduce the policy and value horizon separately. **n-SAC+BC** n-SAC+BC is equivalent to SAC+BC, but with n-step updates. **HIQL** trains the same value function as GCIQL, but the agent extracts a hierarchial policy. All policies trained on OGBench are designed to return a multimodal Gaussian distribution $\mathcal{N}(\mu; \Sigma)$, and during inference time, the neural network produces the distribution, and the policy samples from that as action.

B.2 Task visualization

We provide the tasks used for `antmaze-colossal` environment on Fig. 6. The maze itself is 24 blocks in height and 18 blocks in width, which is 50% larger than the `giant` mazes in each dimension.

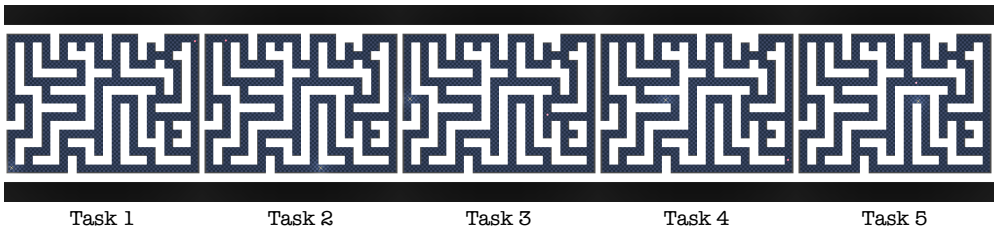

Figure 6: Task visualizations from `antmaze-colossal` environment. The ant occupies the starting position, and it must reach the red dot to complete the task.

B.3 Implementation Details & Hyperparameters

Table 2 details the common hyperparameters for all methods on OGBench. Table 3

For visual environments on OGBench, we define the ResNet encoder for state-action representation and state representation $f_\varphi(s, a)$, $f_\psi(s)$. Empirically, we find that when calculating the $\mathcal{L}_\mathcal{I}$ loss, we freeze gradients for the ResNet encoder, in a manner similar to (Driess et al., 2025). The action invariance loss objective now becomes:

$$\mathcal{L}_\mathcal{I}\left(\varphi, \psi; \{s_i, a_i\}_{i=1}^N\right) = \sum_{i,j=1}^{N} \left(d\big(\psi(\bar{f}_\psi(s_i)), \varphi(\bar{f}_\varphi(s_i), a_j)\big)\right)^2. \tag{17}$$

where $\bar{f}$ denotes freezing the gradient. We find this to be helpful to not let the learned representations collapse during the beginning of training, and that it maintained a good level of generalization ability during training.

Table 2: Network configuration for MQE on OGBench.

Configuration	Value
batch size	256
latent dimension size	512
encoder MLP dimensions	$(512, 512, 512)$
policy MLP dimensions	$(512, 512, 512)$
layer norm in encoder MLPs	`True`
visual encoder (`visual-` envs)	`impala-small`
MRN components	8
Discount (γ)	0.995
Waypoint discount (λ)	0.95
Probability p for directly imposing $\mathcal{L}_\mathcal{T}$	0.2

Table 3: BC coefficient α for each environment

Environment	α
`antmaze-navigate`	0.1
`antmaze-stitch`	0.03
`antmaze-explore`	0.003
`humanoidmaze`	0.01
`pointmaze`	0.03
`{cube, puzzle, scene (play)}`	1.0
`{visual - {cube, puzzle, scene (play)}}`	3.0
`*-noisy`	0.1
`visual-antmaze`	0.3

We use a batch size of 256 for MQE, which, when being trained on an A6000 GPU, takes around 2.5 hours to finish both training and evaluation.

B.4 Policy evaluation

We maintain an evaluation procedure similar to that of OGBench's. We evaluate each policy 50 times the last 3 training epochs (800k, 900k, 1M step for state-based environments, 300k, 400k, 500k in pixel-based environments). For each of these evaluation epochs, use 8 seeds for every state-based environment and 4 seeds for every pixel-based environment.

B.5 Full Result Table

We record the full success rate of all tasks across OGBench in Table 1.

C BridgeData Experiment Details

We evaluate each policy with a total of 15 trials each for every task. For each of the pick and place tasks, we assign 1 point for each successful pick and place (i.e. move a desired object to the desired location). Therefore, a policy can earn a maximum of i points for each PnP task that manipulates i objects.

For open drawer and place, we assign one point if the policy opens the drawer to the extent where the mushroom can be placed, but does not pull the drawer off the base completely, and assign another point if the policy is able to put the mushroom in the drawer.

Table 4: Network configuration for MQE on BridgeData.

Configuration	Value
latent dimension size	256
encoder MLP dimensions	$(256, 256, 256)$
policy MLP dimensions	$(256, 256, 256)$
layer norm in encoder MLPs	`True`
MRN components	8
Discount (γ)	0.98
Waypoint discount (λ)	0.95
Probability p for directly imposing $\mathcal{L}_\mathcal{T}$	0.2

C.1 Training Configurations

We use a pretrained ResNet34 as the backbone of the policy and encoders. We do not share the actor and critic encoder for both GCIQL and MQE. We co-train the actor and the critic for a total of 500,000 total steps with a batch size of 128, which takes around 60 hours when the model is trained on 4 TPU pods. For TRA-g, we produce the embedding of two separate encoders ϕ, ψ, and align each other using symmetric InfoNCE loss. During inference, we use FiLM (Perez et al., 2017) to embed the learned goal representation into the actor, staying consistent with the implementation from (Myers et al., 2025a). In addition, we also describe the common hyperparameters used for BridgeData setup at Table 4.

C.2 Defining Task Successes

Table 5 records the success rate of each task across all six tasks that we evaluate. Unlike Section 5.2, we only measure whether each task has been completed to its fullest. While it does give out a stronger signal on whether a task displays nonzero success rate, it does not provide as much information on how the task was progressing overall.

Table 5: Binary success counts for each task.

Task	MQE (Ours)	GCBC	GCIQL	TRA
Single PnP	12/15	5/15	10/15	9/15
Double PnP	10/15	1/15	4/15	6/15
Triple PnP	4/15	0/15	0/15	1/15
Quadruple PnP	2/15	0/15	0/15	0/15
Open Drawer	12/15	10/15	5/15	11/15
Open Drawer & Place	5/15	0/15	0/15	1/15

D Policy Rollout in BridgeData

We provide the rollout of `triple PnP`. We especially consider TRA-g because it also exhibits compositional generalization, yet there is no explicit policy improvement as compared to offline RL methods such as MQE.

E Additional Ablations

What is the best distribution to sample $\mathfrak{s}_t^w$? We consider three different ways of sampling waypoint $\mathfrak{s}_t^w$ in our approach: (1) uniformly sampling between state and goal. (2) uniformly sampling from Unif$[1, ..., 50]$, and (3) we set $k' = 50$. We note that method (3) directly transforms our method into performing n-step return within the quasimetric architecture, and that it also reduces the value horizon due to sampling a fixed number of states ahead.

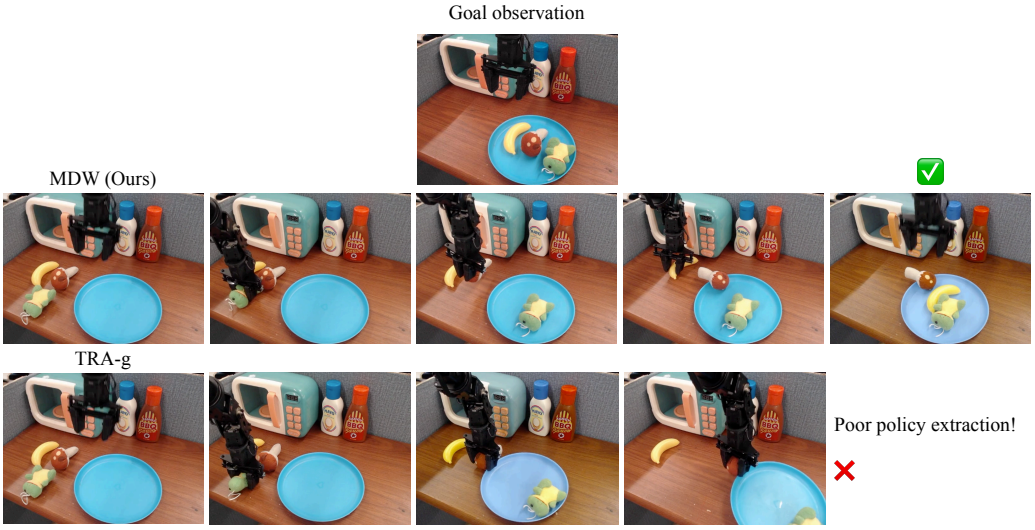

Figure 7: Inference from `triple pnp` task. We note that due to poor policy extraction and generalization, TRA-g is not able to complete the task.

Table 6 demonstrates that using a geometric distribution is the optimal way to sample such waypoints, and explicit horizon reduction techniques actually *hurt* the performance of MQE, as it learns a much less robust policy compared to random sampling.

Table 6: $\mathfrak{s}_t^w$ Sampling Ablation

Configuration	Success Rate (%)
$k' \sim \text{Geom}(1-\lambda)$	**26.5**
$k' \sim \text{Unif}[1, K]$	18.9
$k' \sim \text{Unif}[1, 50]$	17.8
$k' = 50$	1.7

What role does $\mathcal{L}_\mathcal{I}$ play? We now ablate away $\mathcal{L}_\mathcal{I}$ by removing line 6 of Algorithm 1, we examine whether the multistep value propagation is enough to learn a good policy.

Table 7: $\mathcal{L}_\mathcal{I}$ Ablation

Configuration	Success Rate (%)
With $\mathcal{L}_\mathcal{I}$	**26.5**
Without $\mathcal{L}_\mathcal{I}$	7.9

Table 7 shows that $\mathcal{L}_\mathcal{I}$ is in fact necessary, as without this invariance loss, the performance of MQE degrades by a large margin.