Is Temporal-Difference Learning the Only Path to Stitching in RL?

Anonymous Authors[1]

Abstract

Reinforcement learning (RL) promises to solve long-horizon tasks even when training data contains only short fragments of the behaviors. This *experience stitching* capability is often viewed as the purview of temporal difference (TD) methods. However, outside of small tabular settings, trajectories never intersect, calling into question this conventional wisdom. While it is widely held that Monte Carlo (MC) methods are incapable of recombining experience, might function approximation result in a form of implicit stitching, insofar as it is a simpler model for the data? The goal of this paper is to empirically study whether the conventional wisdom about stitching actually holds in settings where function approximation is used. We empirically demonstrate that Monte Carlo methods do often perform experience stitching. While TD methods do achieve slightly stronger capabilities than MC methods (in line with conventional wisdom), this gap narrows as we use larger neural networks. Furthermore, we find that increasing critic capacity effectively reduces the generalization gap for both the MC and TD methods. These results suggest that the TD learning's inductive bias for stitching may be less necessary in the era of large RL models and, in some cases, may offer diminishing returns. Additionally, our results suggest that stitching, a form of generalization unique to the RL setting, might be achieved not through specialized algorithms (temporal difference learning) but rather through the same recipe that has provided generalization in other machine learning settings: scaling model size.

Project website: https://anonymous.4open.science/r/golden-standard-4C84

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

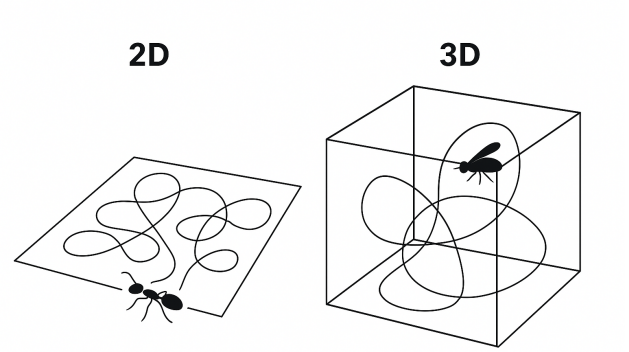


Figure 1. (Left) While TD methods are often conceptualized as piecing together overlapping trajectories, *(Right)* this mental model breaks down in almost all realistic tasks, as trajectories never actually intersect. This paper introduces a new mental model (and formal definitions) for thinking about stitching in such settings, provides a benchmark for rigorously evaluating these stitching capabilities, and performs experiments to understand the degree to which stitching may *actually* achieved through *(i)* temporal difference methods, *(ii)* quasimetric architectures, and *(iii)* simply scaling model architectures.

1. Introduction

In theory, reinforcement learning algorithms should be able to piece together past experiences to find new or better solutions to long-horizon tasks. This ability, sometimes called *experience stitching* (Ghugare et al., 2024; Myers et al., 2025; Wolczyk et al., 2024; Ziebart et al., 2008), is frequently linked to bootstrapping through temporal-difference (TD) updates, wherein the values for one state are updated using predictions from the next state. At least in tabular settings, TD-based methods can boost data efficiency and accelerate convergence (Sutton, 1988; Sutton & Barto, 2018), yet their efficacy in the presence of function approximation remains disputed (Bertsekas, 1995; 2010; Brandfonbrener et al., 2021; Peters et al., 2010).

Outside of tabular or highly-constrained settings, TD methods cannot literally stitch trajectories together: trajectories rarely self-intersect in real-world scenarios. For example, in Figure 1 we compare an ant crawling on a sheet of paper (2D) with a fly flying in an empty room (3D): the ant's 2D path will self-cross far more often than the fly's 3D

For ICML 2026 reviewers: When reviewing this paper, please follow the policy for the use of LLMs as specified on your Reviewer Console. Note that the assigned LLM policy might differ from the policy you indicated during reviewer registration.

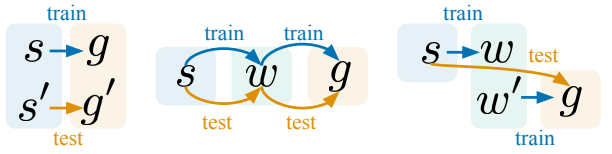
path. Following this example, we observe that stitching is closely tied to generalization. On one hand, stitching requires generalization: the value function must assign similar values to similar states, enabling values to propagate across disconnected trajectories. On the other hand, stitching itself provides generalization: it allows a policy to traverse between states that were never observed together during training.

In this paper, we examine mechanisms that enable stitching from high-dimensional experience. We focus on model scale, learning paradigm (TD vs. MC), and model parametrization, with emphasis on quasimetric architecture. We evaluate three types of regimes of stitching: *no stitching*, *exact stitching* (like Fig. 1 Left)), and *generalized stitching* (like Fig. 1 Right). To probe these regimes, we introduce a minimalist pick-and-place grid benchmark, **`Boxpick`**, that resembles (Schrader, 2018), but with lift/drop actions and without walls. This environment is designed to test composition rather than perception or complex dynamics. To study exact stitching we introduce the Quarters task, wherein agents are trained to transfer boxes between adjacent board quarters and evaluated on diagonal transfer. To study generalized stitching, we introduce the Few-to-Many task, wherein agents are trained to solve easier instances where some boxes are pre-placed, then evaluated on moving all boxes. We also distinguish stitching settings based on how likely it is to encounter states unseen during training; formal definitions appear in Section 4.

The main contribution of this paper is a carefully designed testbed and empirical evaluation of the stitching capabilities of various algorithms and architectures. We train 7 different goal-conditioned agents on these environments. One key observation is that **Monte Carlo methods do stitch**, especially in tasks where they are unlikely to encounter states that are unseen during training. On such tasks, their stitching performance is comparable to TD methods. At the same time, **exact stitching with multi-object coordination is brittle**: performance degrades rapidly as the number of objects grows, and even TD can fail when composition steers rollouts through intermediate states that were never seen during training. In addition, we find that **scale is a powerful lever for stitching**. Increasing the critic network size improves test performance by an average of +42% for MC and enables stitching in methods that failed to stitch with smaller architectures. Taken together, these results suggest a revision to the conventional wisdom: TD is not necessary for stitching, and model scale can significantly improve stitching capabilities.

Our main contributions are the following:

1. We formalize and analyze three stitching regimes: *no stitching*, *exact stitching* (shared waypoint), and *generalized stitching* (waypoint mismatch). We highlight when exact-stitching evaluations can break due to lack of closure under composition.
2. We introduce simple, fast, and configurable environments that isolate stitching phenomena and enable reproducible evaluation across regimes (see Fig. 3).
3. Through controlled experiments across TD and MC algorithms, we find that MC methods can stitch even in settings where trajectories do not exactly overlap. We show, that although TD methods are still superior for stitching capabilities, MC methods can get close to them through scaling of critic networks size, suggesting that MC methods are not fundamentally incapable of performing stitching.



A. No Stitching B. Exact Stitching C. Generalized Stitching

Figure 2. Three types of stitching.

2. Related Work

From tabular prediction to stitching. While build on a foundation of dynamic programming in tabular settings (Bellman & Kalaba, 1957), much of the history of TD learning (Sutton, 1988) has centered around *generalization*. Early work extended TD learning to use function approximation, which enabled predictions on unseen observations (Baird et al., 1995; Bradtke & Barto, 1996; Tsitsiklis & Van Roy, 2002; Bertsekas & Tsitsiklis, 1995). On orthogonal axis is *generalization across state–goal pairs*: solving new *combinations* of familiar states and goals by *stitching* together past experience. While such stitching is possible in the tabular setting (Kaelbling, 1993), recent study of this problem typically employs function approximation (Kaelbling, 1993; Schaul et al., 2015; Andrychowicz et al., 2017; Barreto et al., 2017; 2018).

Prior work has considered different types of stitching, which can be conceptually organized (Figure 2) based on what is present in the replay buffer $\mathcal{D}$ at train time and what is queried at test time:

1. **No stitching (end-to-end only).** *Train:* $\mathcal{D}$ contains end-to-end trajectories $(s' \to g')$. *Test:* evaluate a held-out end-to-end pair $(s \to g)$ (same generator, disjoint pairs).
2. **Exact stitching (shared waypoint).** *Train:* $\mathcal{D}$ contains trajectories $(s \to w')$ and $(w' \to g)$ for the *same* waypoint w'; no $(s \to g)$. *Test:* evaluate

Confidential reviewer copy. This manuscript is under double-blind review by ICML 2026. Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

the end-to-end query $(s \to g)$. This setting aligns with classic dynamic programming and temporal-differenceperspective on stitching (Bellman & Kalaba, 1957; Sutton, 1988), as well as recent discussions of stitching (Ghugare et al., 2024).

3. **Generalized stitching (waypoint mismatch).** *Train:* $\mathcal{D}$ contains $(s \to w')$ and $(w'' \to g)$ with $w' \neq w''$; there is no waypoint $\tilde{w}$ for which both trajectories $(s \to \tilde{w})$ and $(\tilde{w} \to g)$ are present. *Test:* evaluate $(s \to g)$. While we are not aware of work that enforces this train/test split in an online goal-conditioned setting, the notion of *disconnected trajectories* has been investigated in the offline literature: methods operate on separate demonstrations, detect gaps, and learn or synthesize short connecting segments before planning or behavior cloning on the augmented dataset (Char et al., 2022; Hepburn & Montana, 2022), providing a dataset-level analogue of the waypoint-mismatch setting.

Compositional generalization and horizon extension. While generalization in RL is typically studied through the lens of controlled shifts in observations, dynamics, and tasks (Zhang et al., 2018; Packer et al., 2019; Cobbe et al., 2020), our work is more related to the complementary lens of *horizon generalization*, wherein agents train on short-range tasks and test on longer-range ones (typically, by composing subtasks) (Myers et al., 2025; Anil et al., 2022). This type of generalization is similar to the *compositional generalization* studied in supervised learning and generative modeling, which focuses on the systematic recombination of known primitives (Lake & Baroni, 2018; Keysers et al., 2020; Hupkes et al., 2020). While generalization across horizon is typically associated with temporal difference learning, some prior methods propose architectural techniques (such as quasimetrics) that accelerate value propagation to achieve similar generalization (Wang et al., 2023; Piekos et al., 2023; Van Niekerk et al., 2019; Adamczyk et al., 2023). Our experiments will study the extent to which both TD methods and these architectural techniques can solve novel state–goal combinations by recombining familiar parts to solve longer tasks.

Goal-conditioned RL (GCRL). Our study of generalization focuses specifically on goal-reaching tasks. While in tabular settings TD learning seems to be the only tool available for generalization, function approximation offers an alternative way to acquire policies or value functions that can make predictions on state-goal pairs unseen during training. For example, prior work has looked at how policies (Ghosh et al., 2019) and value functions (Schaul et al., 2015; Borsa et al., 2018; Andrychowicz et al., 2017) can simultaneously be trained for multiple goals (though their success at generalization remains disputed (Ghugare et al., 2024)).

Explicit trajectory stitching. Recent offline RL work explicitly recombines sub-trajectories to improve policies from imperfect datasets (Char et al., 2022; Hepburn & Montana, 2022; Li et al., 2024; Ghugare et al., 2024). This stands in contrast to the *implicit* composition often attributed to TD-style value propagation. A natural question, then, is: *which ingredients are actually needed for stitching to emerge in the **online**, goal-conditioned setting?* We investigate this in a controlled online benchmark.

Testbeds for Planning: Sokoban and variants. Our study of stitching will use a benchmark styled after the Sokoban and Boxoban environments (Schrader, 2018), wherein an agent picks or pushes boxes around walls towards a desired configuration. These environments stress-test long-horizon reasoning, and are challenging in a computational complexity sense (Culberson, 1997). The strongest performance on such benchmarks is often achieved by model-based methods (Weber et al., 2017; Guez et al., 2019; Taufeeque et al., 2024) While the state space is combinatorial (k boxes can be in n states), we will carefully design the initial and target configurations used for training to change the likelihood that trajectories exactly overlap. To control for potential confounding effects, our experiments will remove the walls and enable the agent to pick up boxes, thereby eliminating dead ends. We will manipulate the number and placement of boxes across episodes to precisely control the set of seen states, allowing us to test whether agents stitch together familiar subgoals to solve novel state–goal combinations.

3. Preliminaries

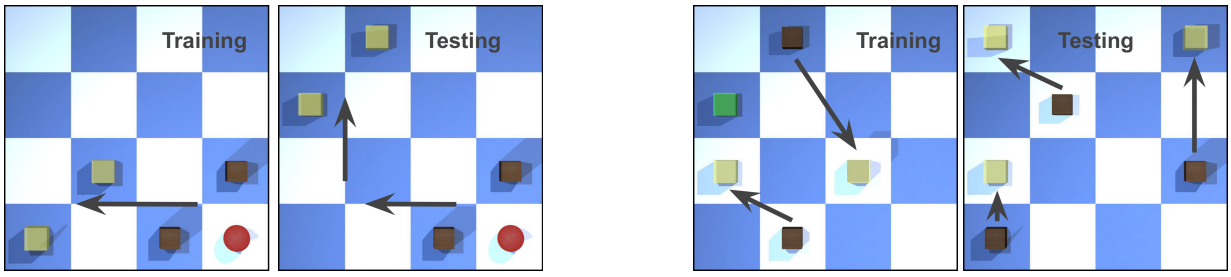
Our paper investigates the generalization properties of on-policy goal-conditioned reinforcement learning, focusing on how Temporal Difference and Monte Carlo methods, as well as network architectures for function approximation, influence stitching capabilities.

We study the problem of goal-conditioned reinforcement learning in a deterministic controlled Markov process with states $s \in \mathcal{S}$, goals $g \in \mathcal{S}$, and actions $a \in \mathcal{A}$. We use an environment with deterministic state transitions and sample the initial states from the distribution $p_0\,(s_0)$. The Q-function, or critic, is defined as

$$Q^\pi(s,a) \;=\; \mathbb{E}_\pi\big[G_t \mid S_t = s, A_t = a, G_t = g,\big],$$

where $G_t \;=\; \sum_{k=0}^{T-t} \gamma^k R_{t+k+1}$ is the empirically observed future $\gamma-$discounted return. Monte Carlo methods learn Q-functions via regressing to the observed returns $(Q(s_t, a_t) \leftarrow \mathrm{G}_t)$ (Sutton & Barto, 2018; Eysenbach

Confidential reviewer copy. This manuscript is under double-blind review by ICML 2026. Unauthorized sharing, redistribution, or disclosure is strictly prohibited.



(a) **Exact stitching (Quarters).** *(b)* **Generalized stitching (Few-to-Many).**

Figure 3. **A benchmark for stitching:** The agent (red ball) must move boxes to the target positions (yellow transparent boxes). *(Left)* During training, boxes are placed in one quarter and must be moved to an *adjacent* quarter (gray arrow indicates the required direction of transfer). During testing, boxes must be moved to the *diagonal* quarter. The gray arrows illustrate one of the valid two-step routes via adjacent quarters (adjacent → adjacent), which were seen separately during training but never as an end-to-end diagonal move. *(Right)* During training, one box is already on a target, and the agent must place the remaining two. During testing, no boxes start on targets. Although both start and goal configurations are individually familiar, training never includes segments that involve moving three boxes.

et al., 2021), while Temporal Difference methods learn Q-functions using bootstrapping ($Q(s_t, a_t) \leftarrow r(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1})$) (Sutton, 1988). Throughout this paper, we sample actions from the Boltzmann (softmax) distribution induced by Q, with learnable temperature τ. The replay buffer stores trajectory sequences, but training uses random i.i.d. pairs sampled from those sequences.

4. A Benchmark for Stitching

To precisely probe the three types of stitching from Fig. 2, we constructed a benchmark called **`Boxpick`**, in which an agent can pick up and place blocks (see Fig. 3). `Boxpick`is fully implemented in JAX (Bradbury et al., 2018), enabling fast, massively parallel data collection across thousands of environments on a single GPU.

Our environment consists of a square grid of fields, with some fields being occupied by *boxes* and *targets*. The task of the agent is to transfer all of the boxes to targets. This setting is thus similar to Sokoban, but differs in *(1)* removing the walls; and *(2)* lifting/dropping boxes instead of pushing boxes, so there is no possibility for the agent to get stuck in an irreversible state. States are discrete, allowing us to determine exactly whether the agent has visited the same state twice. Actions are also discrete, removing policy learning as a potential confounding factor. Nonetheless, the number of states can be made arbitrarily large; for example, Fig. 3b shows 3 blocks in a 4×4 room, so the total number of block configurations is $\binom{16}{3} = 560$. If we increase the number of blocks to 8, and the grid size to 5×5, the number of configurations is more than a million.

Observation and action spaces. The observation consists of `grid_size` × `grid_size` integers, each representing the information about one respective cell of the grid. For instance, 0 represents empty space, while 1 represents a cell with a box. The goal is the same size and shape as the observation. While Fig. 3 overlays yellow target markers on top of the agent's observation, these target markers are added only for human visibility, and are **not** part of the agent's observation. The agent's inputs are the observation and the goal. There are six possible actions that the agent can perform in each state: `go_left`, `go_right`, `go_up`, `go_down`, `pick_up_box`, `put_down_box`.

Within this environment, we constructed three distributions of box and goal placement to test no stitching, exact stitching, and generalized stitching variants:

- **No stitching** (Fig. 2 A) – A fixed number of blocks are placed randomly, and the goal is a different random arrangement of blocks. This setting will be used to check the implementation of algorithm baselines and compare different hyperparameter choices.
- **The Quarters Setting** (Fig. 3a) – The board is split into four equal quarters. During training, the initial state has all blocks randomly placed within one quarter, and the goal state has the blocks randomly placed in an adjacent quarter.

  The agent is then evaluated in two ways. First, it is evaluated on the exact same environment. Second, it is evaluated on the same number of boxes and targets, that are placed in diagonal (i.e., not adjacent) quarters. Ideally, the agent should succeed at moving boxes between opposite corners by sequentially moving boxes between adjacent corners (i.e., via stitching). With a sufficient number of experiences collected, each possible combination of boxes and each possible combination of targets should appear in each of the quarters. Thus, during the evaluation, both the initial states and goal states have each been seen before; they are not out of distribution. However, the relative position of boxes in the initial state

Confidential reviewer copy. This manuscript is under double-blind review by ICML 2026. Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

and goal state has never been observed during training, so the pair (s, g) is out of distribution. Thus, this setting evaluates *exact stitching* (cf. Fig. 2 B).

- **The Few-to-Many Setting** (Fig. 3b) – This setting tests how well the agent can generalize to a task that involves moving a different number of boxes. Let n denote the total number of boxes (equivalently, the number of targets), which are placed uniformly on the board. Let $0 < m < n$ specify how many boxes are initially spawned on their targets. The agent needs to move the remaining $n - m$ boxes to accomplish the task. During training, the environment parameters n and m are fixed (i.e., not randomized). During the evaluation, none of the boxes are spawned on the targets. By construction, the initial state and goal state are both in the distribution of states seen during training, yet their combination is (by construction) never seen during training.[1] This task tests *generalized stitching* (cf. Fig. 2 C).

These settings allow us to test the exact and generalized stitching capabilities. We can also change the difficulty of the task by increasing the size of the grid and the number of boxes.

4.1. Stitching can result in out-of-distribution states

Our discussion above highlights that the evaluation tasks are constructed so that the start and goal states used for evaluation have been seen *individually* during training (though not jointly). Despite this, an agent attempting to navigate from one state to another may visit a state that is truly out-of-distribution. For example, in the Quarter setting with $n = 2$ blocks, if the agent picks up the first block and drops it in the opposite corner, then the state of the world will consist of blocks in opposite corners, a state which was not seen during training (see Fig. 6). This challenge is analogous to the challenge in imitation learning wherein out-of-distribution actions can lead to states unseen during training (Ross et al., 2011).

Below we introduce some terminology for describing this phenomenon, which we will use for analyzing our experiments in Sec. 5. Let $\mathcal{D}$ be the training replay buffer and let $\mathcal{M} := \{\, s \;:\; s \text{ appears in } \mathcal{D} \,\}$ denote its empirical state support. For a test query (s, g), let $\mathsf{Traj}(s, g)$ be the set of feasible trajectories from s that reach g.

We call (s, g) *closed* if all feasible trajectories $\tau \in \mathsf{Traj}(s, g)$ have states all lying in $\mathcal{M}$ (i.e., no out-of-support waypoint is needed).

[1]The start-goal pairs seen during training all consist of at least one boxes already on the goal, so a start-goal pair where no boxes are in the target location is out-of-distribution for the joint distribution over (start, goal) pairs.

We call (s, g) *open* if efficient executions naturally visit states outside $\mathcal{M}$—formally, there exists a near-optimal trajectory $\tilde{\tau} \in \mathsf{Traj}(s, g)$ with some intermediate state $w \notin \mathcal{M}$.

Intuitively, while *open* test query (s, g) does not prevent on-support solutions, it indicates that near-optimal trajectories would traverse states outside $\mathcal{M}$, given how the training trajectories were collected. If, however, training rollouts cover the full relevant state space, the same query (s, g) becomes effectively *closed*. In the `Boxpick` Quarters Setting, most test queries are *open*, since completing the training task does not require moving boxes along the diagonal. In contrast, the Few-to-Many setting primarily uses *closed* test queries, as boxes and targets are randomly spawned across the entire board during training. Accordingly, we refer to the Quarters setting as *open* and the Few-to-Many setting as *closed*.

5. Experiments

The primary goal of our experiments is to understand which types of stitching are performed by TD and MC methods that use a Q-function with function approximation. We also investigate the role of architecture in stitching, focusing on scaling the critic depth Wang et al. (2025) and parametrization using quasimetric networks from Myers et al. (2025); Wang et al. (2023). Our aim is not to propose a new method, but to provide a rigorous evaluation of the stitching capabilities of today's methods. In Section 5.1, we describe the experimental setup, and in the consecutive sections, we aim to answer the following research questions:

- Do any of today's methods do stitching (Section 5.2)?
- Can MC methods perform stitching, or is TD learning necessary (Section 5.3)?
- Does scale improve stitching (Section 5.4)?
- Do quasimetric networks improve stitching (Section 5.5)?

5.1. Experimental setup

We will answer the questions above using four base RL algorithms: Deep Q Networks (DQN) (Mnih et al., 2013), Contrastive Reinforcement Learning (CRL) (Eysenbach et al., 2022), C-learning (Eysenbach et al., 2021), and Implicit Q-Learning (IQL) (Kostrikov et al., 2021). We implement C-learning, IQL, and DQN in two versions: MC and TD. C-learning and CRL are reward-free methods; for DQN and IQL, we use a sparse reward of 1 when all boxes are in the target position and 0 otherwise. We also use hindsight goal relabeling (Andrychowicz et al., 2017) for DQN and IQL (50% of future states 50% of random states), both for the MC and TD versions. CRL and C-learning implicitly

Confidential reviewer copy. This manuscript is under double-blind review by ICML 2026. Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

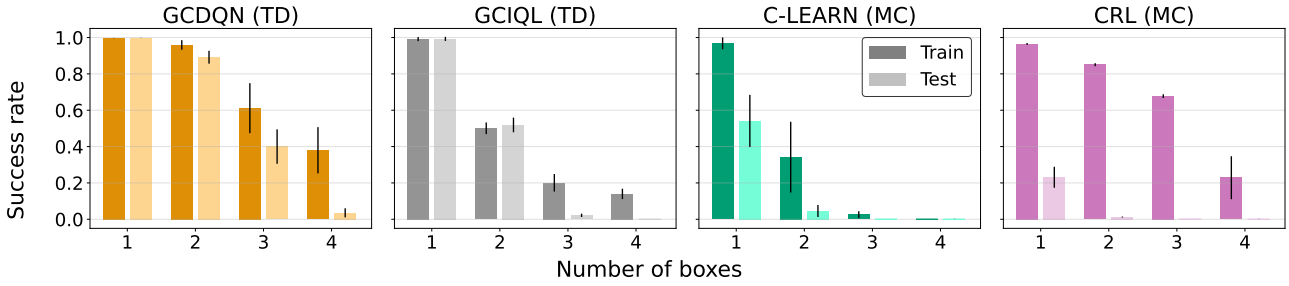


Figure 4. **TD methods can only stitch effectively up to a certain point.** In the Quarters setting (6×6 grid) — which tests exact stitching — increasing the number of boxes widens the generalization gap for both TD and MC methods.

do hindsight relabeling implicitly. In most experiments, all methods use an MLP with two hidden layers, each containing 256 units, followed by post-activation LayerNorm. In Section 5.4, we instead adopt the architecture from Wang et al. (2025), which employs ResNet blocks, Swish activations, and pre-activation LayerNorm. In particular, we use two ResNet blocks, each with 4 hidden layers and 1024 units per layer. Note that CRL uses two networks as encoders in the critic, so its total parameter count is roughly $2\times$ that of the other methods. We use the default hyperparameters from Park et al. (2025) for all methods (see Appendix A), except that we reduce the discount factor for the MC versions of DQN, C-learning, and IQL to stabilize training ($\gamma = 0.99 \rightarrow 0.9$; see Appendix B.4).

We train all methods using the ADAM optimizer for 5 million update steps, collecting a total of 500 million transitions online. Training alternates between full rollouts, data collection, and network updates. For both data collection and evaluation, we sample actions from the Boltzmann (softmax) distribution defined by Q. We do *not* use a separate parameterized policy, as our main focus is on the critic's stitching ability, which could later be distilled into an actor. We tune an additional temperature parameter for all the methods so that the entropy of the Q-induced distribution is close to $\ln(|A|/2) \approx 1.1$. In all experiments, we use the settings introduced in Section 4. As a performance metric, we use *success rate*, i.e., the number of attempts finished with all boxes in the target positions. In the majority of the plots, we report the mean of 10 seeds with standard deviation. We use the term *generalization gap* to refer to the difference between method performance on the training and evaluation tasks, which differ in Quarter and Few-to-Many settings.

5.2. Do any of today's methods do stitching?

Previous works (Ghugare et al., 2024; Myers et al., 2025; Sutton, 1988) argue that temporal difference (TD) methods can compose test-time behavior from sub-behaviors learned during training. However, Monte Carlo (MC) methods might not provide this guarantee.

To test this, we probe exact stitching in a Quarter (6×6) grid: can a method recombine learned sub-behaviors to solve a held-out test task? We evaluate one TD method (DQN) and two MC baselines (CRL and C-learn) and report their final performance on both the training and the test tasks.

In Figure 4, DQN (a TD method) achieves near-perfect training and evaluation performance on the single-box task. By contrast, the Monte-Carlo baselines, CRL, and C-learning all show a large generalization gap even in this simplest setting. As the number of boxes increases and test-time observations become out-of-distribution, only DQN retains any nontrivial performance—highlighting an advantage of TD learning. Still, DQN's generalization steadily worsens with more boxes and falls to 0% test performance on the 4-box test task. **A sudden generalization gap of DQN suggests that as the space of possible states expands, it eventually exceeds the stitching capacity of TD updates.** Visual inspection confirms more failures caused by off-support observations with an increased number of boxes (Figure 6). This pattern argues for methods that regularize agent behavior in online RL so agents remain closer to the training observation distribution, analogous to action regularization in offline RL (Fujimoto & Gu, 2021).

We also examine generalized stitching in the Few-to-many (5x5) grid, which operates in a closed setting, i.e., all observations are seen in training, and no off-support observations can be visited (Section 4). The test task becomes more difficult as the number of boxes spawned at the target position increases, since it requires more stitching. We note that the setting where no boxes are spawned in the target positions corresponds to a no-stitching.

We increase the number of boxes spawned at the target position from left to right in Figure 5. For all three baselines, the generalization gap widens as more boxes appear at the target during training. DQN, using TD updates, consistently sustains the highest performance in these harder settings. **Remarkably, CRL still performs well on the test task even when training required moving only three out of four boxes, indicating that an MC-style method**

Confidential reviewer copy. This manuscript is under double-blind review by ICML 2026. Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

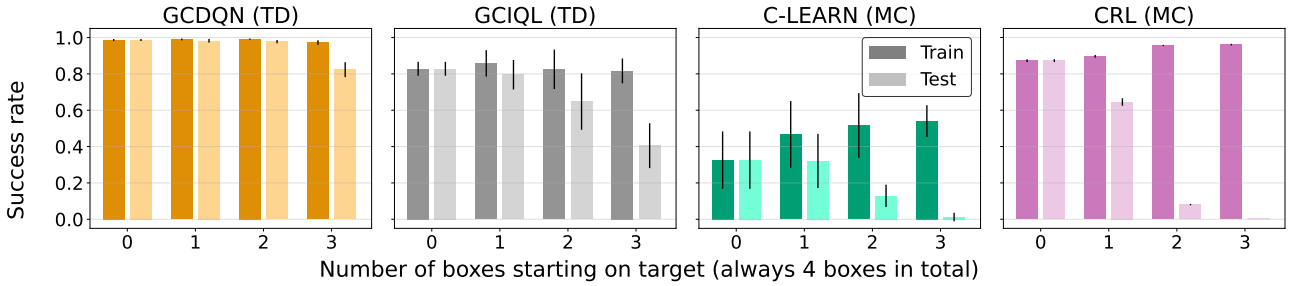


Figure 5. **Stitching is easy on training support, even for MC methods.** In the Few-to-many setting, we probe methods' generalized stitching capabilities. The more difficult the training tasks (fewer boxes starting on target), the smaller the generalization gap for both MC and TD methods.

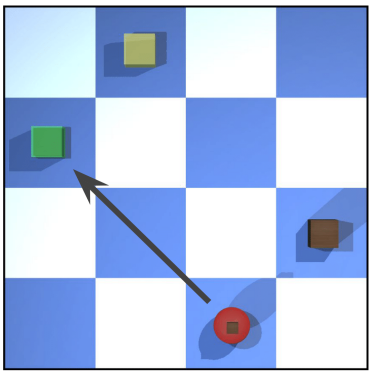


Figure 6. **A subtle failure of stitching.** An agent trained on the quarters task (Fig. 3a) should first move all boxes to an adjacent corner and then to the goal quarter. However, if the agent prematurely moves a box along the diagonal, it will end up in a state that has never been seen before during training.

can stitch subbehaviors. We explore this surprising phenomenon in the next sections, using only the Few-to-many setting from now onwards.

5.3. Are MC methods performing stitching, or is TD learning necessary?

In this section, we compare CRL, TD and MC versions of DQN, and C-learning to investigate their stitching capabilities in a generalized (Few-to-Many) setting. In particular, we use a Few-to-many 5x5 grid, and train the agent to move 2 boxes to target positions, while a third box is always spawned at the target position. During the test time, all 3 boxes are not in the target position. In Figure 7, we observe that all the methods, except DQN MC, exhibit strong stitching as their generalization gap is relatively small for TD and MC methods, with almost no gap for TD versions of DQN. This result might come as a surprise because MC methods do not employ explicit stitching mechanisms, unlike TD methods; however, they still work well in this setting, most likely due to implicit stitching at the representation level enabled by function approximation. The low performance of DQN MC is most probably due to exploration and credit assignment issues.

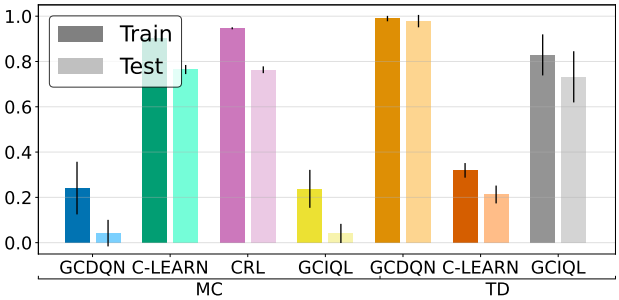


Figure 7. **TD is not necessary for stitching behavior in a generalized setting.** We observe that in the Few-To-Many setting, both TD and MC methods are able to generalize to the test scenario. The exact performance (success rate) varies across different algorithms.

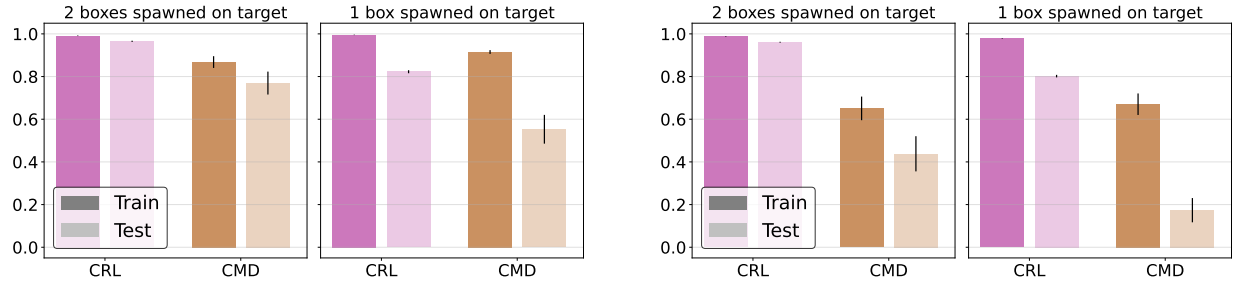
5.4. Does scaling improve stitching?

Previous work (Nauman et al., 2024; Lee et al., 2025; Wang et al., 2025) shows that appropriately scaling critic and actor networks yields substantial performance gains in online RL. In this section, we study whether the scale of the critic similarly benefits the stitching capabilities of MC and TD methods. We use the same setting as in Section 5.3 (Few-to-Many). In Figure 9, we show the performance boost on the test task due to using bigger neural networks (extended results are in Figure 10 in Appendix B). CRL, DQN (MC), and C-learn (TD) benefit the most from the critic scale. **Strikingly, the generalization gap might be reduced by simply increasing the scale of the critic for both TD and MC methods.**

5.5. Do quasimetric networks improve stitching?

Quasimetric networks have been shown to provide benefits such as improved sample efficiency in the goal-conditioned RL by making $Q(s, a, g)$ satisfy the triangle inequality (Myers et al., 2025; Liu et al., 2022). In this section, we compare

Confidential reviewer copy. This manuscript is under double-blind review by ICML 2026. Unauthorized sharing, redistribution, or disclosure is strictly prohibited.



(a) Grid 5x5 *(b)* Grid 6x6

Figure 8. **Quasimetrics do not improve stitching.** In the Few-to-many setting with a 3-box task during testing, CMD results in a worse success rate and a wider generalization gap than CRL.

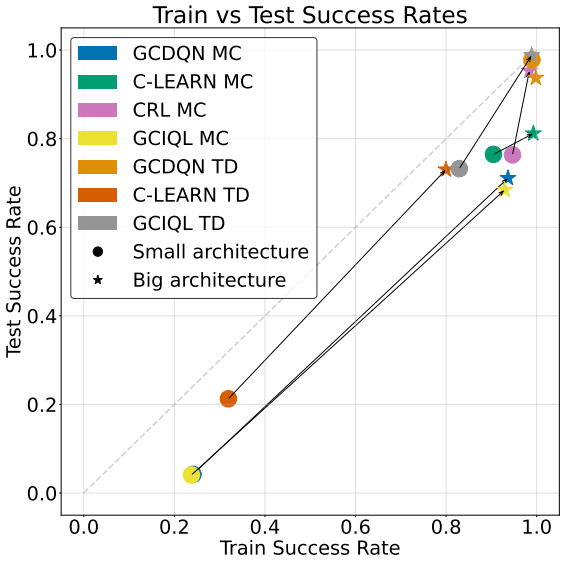


Figure 9. **Scale is a powerful lever for stitching.** Increasing the critic size narrows the generalization gap (reducing distance from the $x = y$ line) and improves test performance for MC methods by an average of $+42\%$.

CRL with Contrastive Metric Distillation (CMD) Myers et al. (2024), which replaces the L2 distance used in CRL with a quasimetric distance between embeddings. We evaluate both methods in the Few-to-many setting on grids 5x5 and 6x6, with 3 boxes. We find that using quasimetric distance only decreases performance and slows down the learning in our benchmark (Figure 8). We believe this is because the environment dynamics is symmetric: for every pair of states A and B, the shortest path from A to B has the same length as from B to A. Thus, splitting embeddings into symmetric and asymmetric components appears to add an unnecessary inductive bias.

6. Conclusion

This work introduces a formal taxonomy and a controllable benchmark to re-evaluate the mechanisms of experience stitching in goal-conditioned RL, yielding key insights that revise conventional wisdom. Our experiments show that, contrary to common belief, Monte Carlo methods can stitch experiences in challenging settings. When test data lies within the training support, they can achieve generalization gaps as small as those of Temporal Difference methods. While TD learning provides an advantage in exact stitching scenarios, its performance degrades as task complexity increases, indicating it is not a universally sufficient solution. Crucially, our results highlight that model scale is a powerful lever for improving stitching. Increasing the critic network's capacity substantially narrows the generalization gap for both MC and TD methods. These findings suggest that the specialized inductive bias of TD learning may be less essential in the era of large models; instead, effective experience stitching can be achieved through the same principle that has proven successful in other machine learning domains: scaling model capacity.

One limitation of our work is the reliance on a relatively simple grid-world with a small action space. We chose this controlled setting to enable a concrete evaluation of stitching, which is difficult to verify in richer domains. Nevertheless, even in this simplified setting, temporal-difference methods fail to exhibit exact stitching as the number of boxes increases. Our experiments focus solely on a sparse-reward regime and a small set of popular baselines that we consider representative of goal-conditioned algorithms. We also did not investigate the stitching or generalization produced by a separately parameterized actor policy. Future work should study actor generalization, the effects of exploration and data collection, and scaling to richer, continuous environments.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

Confidential reviewer copy. This manuscript is under double-blind review by ICML 2026. Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

References

Adamczyk, J., Makarenko, V., Arriojas, A., Tiomkin, S., and Kulkarni, R. V. Bounding the optimal value function in compositional reinforcement learning, 2023. URL https://arxiv.org/abs/2303.02557.

Amit, R., Meir, R., and Ciosek, K. Discount factor as a regularizer in reinforcement learning. In *International conference on machine learning*, pp. 269–278. PMLR, 2020.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight experience replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL https://papers.nips.cc/paper/7090-hindsight-experience-replay.

Anil, C., Wu, Y., Andreassen, A., Lewkowycz, A., Misra, V., Ramasesh, V., Slone, A., Gur-Ari, G., Dyer, E., and Neyshabur, B. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556, 2022.

Baird, L. et al. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the twelfth international conference on machine learning*, pp. 30–37, 1995.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., Silver, D., and van Hasselt, H. Successor features for transfer in reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL https://papers.nips.cc/paper/6994-successor-features-for-transfer-in-reinforcement-learning.

Barreto, A., Borsa, D., Quan, J., Schaul, T., Silver, D., Hessel, M., Mankowitz, D. J., Žídek, A., and Munos, R. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 510–519. PMLR, 2018. URL http://proceedings.mlr.press/v80/barreto18a.html.

Bellman, R. and Kalaba, R. Dynamic programming and statistical communication theory. *Proceedings of the National Academy of Sciences*, 43(8):749–751, 1957.

Bertsekas, D. P. A Counterexample to Temporal Differences Learning. *Neural Computation*, 7:270–279, 1995.

Bertsekas, D. P. Pathologies of Temporal Difference Methods in Approximate Dynamic Programming. *MIT web domain*, 2010.

Bertsekas, D. P. and Tsitsiklis, J. N. Neuro-dynamic programming: an overview. In *Proceedings of 1995 34th IEEE conference on decision and control*, volume 1, pp. 560–564. IEEE, 1995.

Borsa, D., Barreto, A., Quan, J., Mankowitz, D., Munos, R., Van Hasselt, H., Silver, D., and Schaul, T. Universal successor features approximators. *arXiv preprint arXiv:1812.07626*, 2018.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Bradtke, S. J. and Barto, A. G. Linear least-squares algorithms for temporal difference learning. *Machine learning*, 22(1):33–57, 1996.

Brandfonbrener, D., Whitney, W., Ranganath, R., and Bruna, J. Offline rl without off-policy evaluation. *Advances in neural information processing systems*, 34:4933–4946, 2021.

Char, I., Mehta, V., Villaflor, A., Dolan, J. M., and Schneider, J. Bats: Best action trajectory stitching, 2022. URL https://arxiv.org/abs/2204.12026.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. Leveraging procedural generation to benchmark reinforcement learning, 2020. URL https://arxiv.org/abs/1912.01588.

Culberson, J. Sokoban is pspace-complete. 1997.

Eysenbach, B., Salakhutdinov, R., and Levine, S. C-learning: Learning to achieve goals via recursive classification. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=tc5qisoB-C.

Eysenbach, B., Zhang, T., Levine, S., and Salakhutdinov, R. R. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Ghosh, D., Gupta, A., Reddy, A., Fu, J., Devin, C., Eysenbach, B., and Levine, S. Learning to reach goals via iterated supervised learning. *arXiv preprint*, 2019. URL https://arxiv.org/abs/1912.06088.

Confidential reviewer copy. This manuscript is under double-blind review by ICML 2026. Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

Ghugare, R., Geist, M., Berseth, G., and Eysenbach, B. Closing the gap between TD learning and supervised learning – a generalisation point of view. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=qg5JENs0N4.

Guez, A., Mirza, M., Gregor, K., Kabra, R., Racanière, S., Weber, T., Raposo, D., Santoro, A., Orseau, L., Eccles, T., Wayne, G., Silver, D., and Lillicrap, T. An investigation of model-free planning, 2019. URL https://arxiv.org/abs/1901.03559.

Hepburn, C. A. and Montana, G. Model-based trajectory stitching for improved offline reinforcement learning, 2022. URL https://arxiv.org/abs/2211.11603.

Hupkes, D., Dankers, V., Mul, M., and Bruni, E. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67: 757–795, 2020.

Kaelbling, L. P. Learning to achieve goals. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1094–1098, 1993.

Keysers, D., Schärli, N., Scales, N., Buisman, H., Furrer, D., Kashubin, S., Momchev, N., Sinopalnikov, D., Stafiniak, L., Tihon, T., Tsarkov, D., Wang, X., van Zee, M., and Bousquet, O. Measuring compositional generalization: A comprehensive method on realistic data, 2020. URL https://arxiv.org/abs/1912.09713.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. *International Conference On Learning Representations*, 2021.

Lake, B. M. and Baroni, M. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks, 2018. URL https://arxiv.org/abs/1711.00350.

Lee, H., Lee, Y., Seno, T., Kim, D., Stone, P., and Choo, J. Hyperspherical normalization for scalable deep reinforcement learning. *arXiv preprint arXiv: 2502.15280*, 2025.

Li, G., Shan, Y., Zhu, Z., Long, T., and Zhang, W. Diffstitch: Boosting offline reinforcement learning with diffusion-based trajectory stitching. *arXiv preprint arXiv:2402.02439*, 2024.

Liu, B., Feng, Y., Liu, Q., and Stone, P. Metric residual networks for sample efficient goal-conditioned reinforcement learning. *arXiv preprint arXiv: 2208.08133*, 2022.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv: 1312.5602*, 2013.

Myers, V., Zheng, C., Dragan, A., Levine, S., and Eysenbach, B. Learning temporal distances: Contrastive successor features can provide a metric structure for decision-making. *International Conference on Machine Learning*, 2024. doi: 10.48550/arXiv.2406.17098.

Myers, V., Ji, C., and Eysenbach, B. Horizon generalization in reinforcement learning. *arXiv preprint*, 2025. URL https://arxiv.org/abs/2501.02709.

Nauman, M., Ostaszewski, M., Jankowski, K., Milo's, P., and Cygan, M. Bigger, regularized, optimistic: scaling for compute and sample-efficient continuous control. *Neural Information Processing Systems*, 2024. doi: 10.48550/arXiv.2405.16158.

Packer, C., Gao, K., Kos, J., Krähenbühl, P., Koltun, V., and Song, D. Assessing generalization in deep reinforcement learning, 2019. URL https://arxiv.org/abs/1810.12282.

Park, S., Frans, K., Eysenbach, B., and Levine, S. OGBench: Benchmarking offline goal-conditioned RL. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=M992mjgKzI.

Peters, J., Mulling, K., and Altun, Y. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pp. 1607–1612, 2010.

Piekos, P., Ramesh, A., Faccio, F., and Schmidhuber, J. Efficient value propagation with the compositional optimality equation. In *NeurIPS 2023 Workshop on Goal-Conditioned Reinforcement Learning*, 2023.

Rathnam, S., Parbhoo, S., Swaroop, S., Pan, W., Murphy, S. A., and Doshi-Velez, F. Rethinking discount regularization: New interpretations, unintended consequences, and solutions for regularization in reinforcement learning. *Journal of Machine Learning Research*, 25(255):1–48, 2024.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of*

Confidential reviewer copy. This manuscript is under double-blind review by ICML 2026. Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

Machine Learning Research, pp. 1312–1320. PMLR, 2015. URL http://proceedings.mlr.press/v37/schaul15.html.

Schrader, M.-P. B. gym-sokoban. https://github.com/mpSchrader/gym-sokoban, 2018.

Sutton, R. S. Learning to predict by the method of temporal differences. *Machine Learning*, 3(1):9–44, 1988. doi: 10.1023/A:1022633531479.

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

Taufeeque, M., Quirke, P., Li, M., Cundy, C., Tucker, A. D., Gleave, A., and Garriga-Alonso, A. Planning in a recurrent neural network that plays sokoban. *arXiv preprint*, 2024. URL https://arxiv.org/abs/2407.15421.

Tsitsiklis, J. N. and Van Roy, B. On average versus discounted reward temporal-difference learning. *Machine Learning*, 49(2):179–191, 2002.

Van Niekerk, B., James, S., Earle, A., and Rosman, B. Composing value functions in reinforcement learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6401–6409. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/van-niekerk19a.html.

Van Seijen, H., Fatemi, M., and Tavakoli, A. Using a logarithmic mapping to enable lower discount factors in reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Wang, K., Javali, I., Bortkiewicz, M., Trzciński, T., and Eysenbach, B. 1000 layer networks for self-supervised rl: Scaling depth can enable new goal-reaching capabilities. *arXiv preprint arXiv: 2503.14858*, 2025.

Wang, T., Torralba, A., Isola, P., and Zhang, A. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning*, pp. 36411–36430. PMLR, 2023.

Weber, T., Racanière, S., Reichert, D. P., Buesing, L., Guez, A., Rezende, D., Puigdomènech Badia, A., Vinyals, O., Heess, N., Li, Y., Pascanu, R., Battaglia, P., Hassabis, D., Silver, D., and Wierstra, D. Imagination-augmented agents for deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL https://arxiv.org/abs/1707.06203.

Wolczyk, M., Cupial, B., Ostaszewski, M., Bortkiewicz, M., Zajac, M., Pascanu, R., Kucinski, L., and Milos, P. Fine-tuning reinforcement learning models is secretly a forgetting mitigation problem. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=53iSXb1m8w.

Zhang, C., Vinyals, O., Munos, R., and Bengio, S. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, pp. 1433–1438. AAAI Press, 2008. ISBN 9781577353683.

Confidential reviewer copy. This manuscript is under double-blind review by ICML 2026. Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

A. Experimental Setup

All experiments were run on 10 seeds, with the hyperparameters reported in Table 1. In the MC version of DQN and IQL, we use discounted returns for the relabeled goal as targets. To that end, we store experience in a trajectory buffer rather than a standard transition buffer. For each sampled trajectory, we relabel all goals to a future state selected using a geometric distribution. We then compute discounted rewards by propagating them backward through the trajectory. Finally, instead of using a bootstrapped target for the Q-update, we use the discounted cumulative reward computed directly from the replay buffer.

Table 1. Hyperparameters

Hyperparameter	Value
num env steps	500,000,000
num updates	1,000,000
max replay size (per env instance)	10,000
min replay size	1,000
episode length	100
discount	0.99 (0.9 for MC versions of DQN, IQL, and C-learning)
number of parallel envs	1024
batch size	256
learning rate	3e-4
contrastive loss function	sigmoid_binary_cross_entropy
energy function	dot_product
representation dimension	64
target_entropy	1.1

B. Additional Results

B.1. Architecture Scaling

In Figure 9, for readability, we reported the mean, without standard deviation. In Figure 10 we present the full scaling experiment results, with standard deviation, for both grid sizes 4 and 5.

B.2. Hyperparameter tuning

To ensure reproducibility and establish a strong baseline, we adopted the core hyperparameter configurations from OGBench (Park et al., 2025) for the IQL (TD) and CRL (MC) implementations. These configurations also served as the starting point for the algorithms implemented specifically for this project: C-learn (MC and TD), DQN (MC and TD), and IQL (MC). However, to verify the suitability of these hyperparameters for the specific challenges of the proposed stitching benchmark, we conducted a sensitivity analysis on critical hyperparameters, including the batch size, number of parallel environments, number of gradient updates, discount factor γ, and the target entropy used in the Q-induced softmax policy. Selected results from this analysis are shown below.

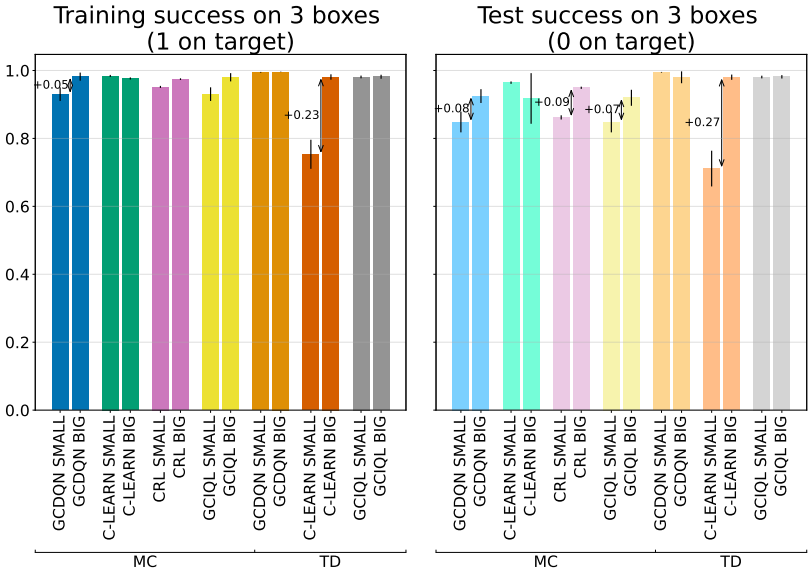
In Figure 11, we report final success rates for different values of discounting and target entropy used in the Q-induced softmax policy during data collection for DQN (TD) and CRL (MC). We use the Quarters Setting with a grid size of 6×6 and 3 boxes, as this setup yielded results that were neither saturated nor trivial for both DQN and CRL.

We note that while our environment can, in principle, be run with many more boxes and larger grid sizes, all implemented RL methods exhibit relatively low performance in these settings. In practice, they struggle and tend to achieve only trivial performance once the grid size exceeds 6 or the number of boxes reaches four or more.

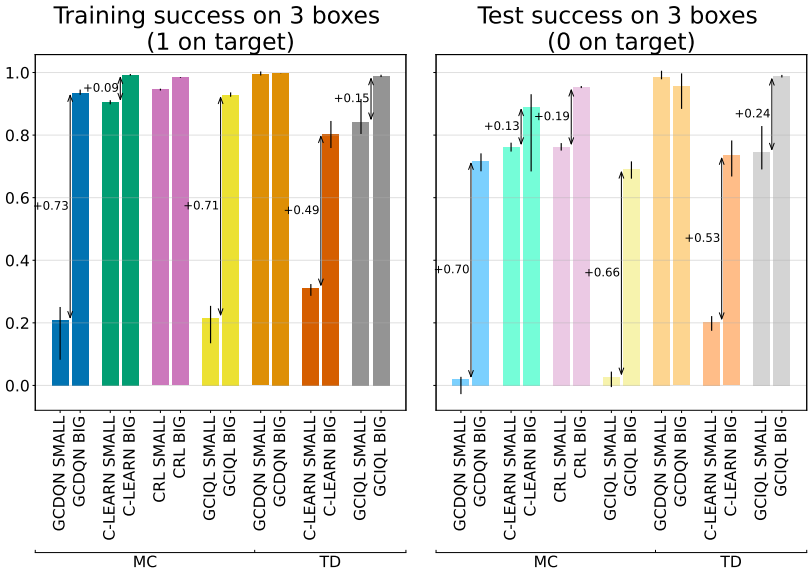
B.3. How exploration affects stitching performance?

To study the relationship between data collection entropy (i.e., exploration) and the policy induced by the learned Q function, we conducted experiments with DQN and CRL using different target entropy values in the Q-induced softmax policy during

Confidential reviewer copy. This manuscript is under double-blind review by ICML 2026. Unauthorized sharing, redistribution, or disclosure is strictly prohibited.



(a) Success rate boost with scaling to bigger networks on a grid size of 4 in a generalized setting



(b) Success rate boost with scaling to bigger networks on a grid size of 5 in a generalized setting

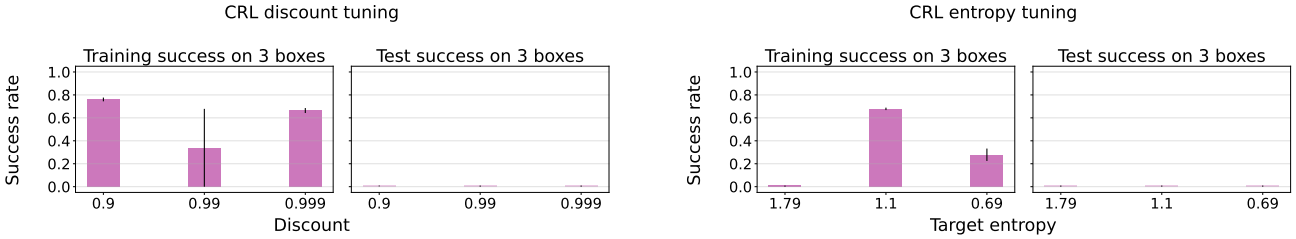
Figure 10. **Full Generalized Stitching setting experiments.**

data collection. We use the Quarters Setting with a grid size of 6×6 and 3 boxes. As shown in Figure 12, the argmax policy for CRL achieves near-zero performance, likely because it gets stuck in states where the Q function is poorly estimated. Policy visualizations suggest that in such cases, the agent either attempts to pick up a box from an empty cell or tries to drop a box it does not have. In contrast, the argmax policy for DQN achieves non-trivial performance, though still lower than that of the softmax policy (see Figure 11(b)), with the best performance occurring at a target entropy of 1.1, the value used in our main experiments.

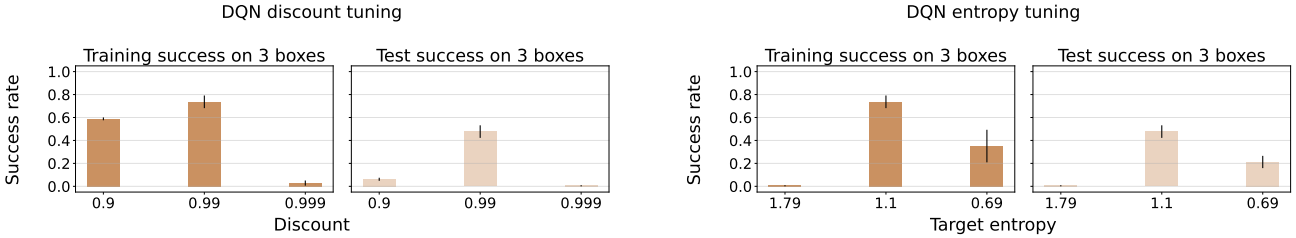
B.4. Effects of Discounting and Network Scaling in the Few-to-Many Setting

In this section, we report the effect of the discount parameter on MC and TD methods in the Few-to-many setup, while scaling the architecture of the critic network. We evaluate all methods with discount factor of $\gamma = 0.9$ and $\gamma = 0.99$. Full results of those experiments are presented in Figure 13 and Figure 14. While TD methods have similar performance for both

Confidential reviewer copy. This manuscript is under double-blind review by ICML 2026. Unauthorized sharing, redistribution, or disclosure is strictly prohibited.



(a) CRL: final train and test success rates for (left) different discounts and (right) different target entropy values used in softmax policy for data collection



(b) DQN: final train and test success rates for (left) different discounts and (right) different target entropy values used in softmax policy for data collection

Figure 11. **Verification of hyperparameters values.** Success rates for different values of discounting and target entropy used in the Q-induced softmax policy during data collection.

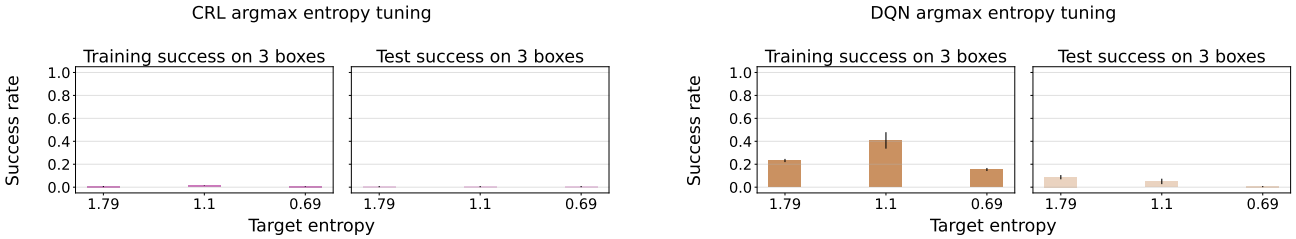


Figure 12. **Relation between data collection policy and argmax policy at test time.** We observe that the argmax(Q) policy yields zero performance for CRL. This suggests that, to prevent CRL from getting stuck in states where the Q-function is misestimated, using a softmax policy at test time is essential. For DQN, the best argmax-policy performance is achieved when the data are collected using a softmax policy with a target entropy of 1.1.

values of γ, all MC methods but CRL benefit from a lower discount factor, $\gamma = 0.9$. This is likely because lower values of γ reduce the variance of relabeled returns (Rathnam et al., 2024; Amit et al., 2020; Van Seijen et al., 2019), which can improve generalization, albeit at the cost of shortening the effective horizon.

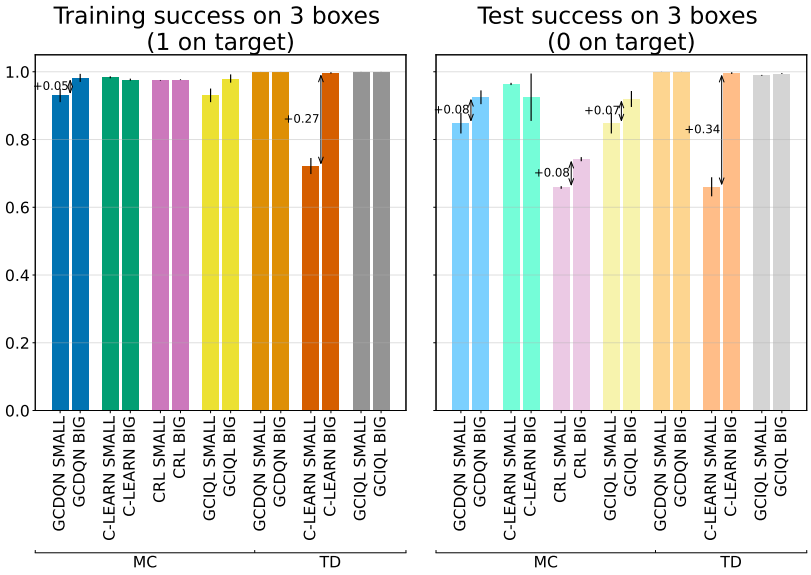
B.5. Monte-Carlo DQN results

Most experiments in the paper use 50 training epochs, corresponding to 5 million gradient updates and 500 million environment steps. However, we found that even this substantial amount of training is insufficient for the MC version of DQN to converge when using a small critic architecture and a discount rate of 0.9. In Figure 15, we present the results for a $10\times$ longer training for this method in the generalized setup (5×5) and 4 boxes in total. Interestingly, even when the agent is trained to move only two boxes (green line), it still learns to stitch, achieving a non-trivial success rate of 20% on the test task, which requires moving all four boxes.

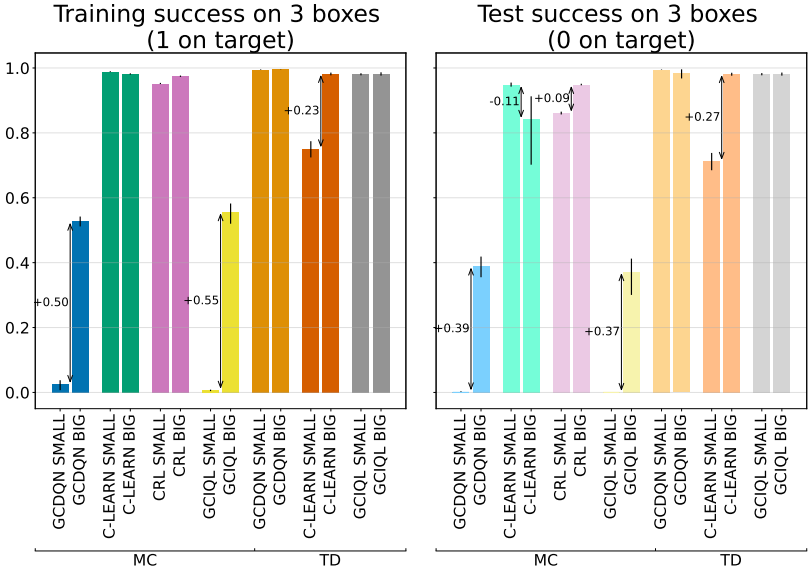
B.6. Wall-clock Time of Training

In Table 2, we report the average wall-clock times for training the agents with 500 million environment transitions and 5 million gradient updates, broken down by their architecture sizes. The times are for 5×5 grid and the Few-To-Many setup. We conduct experiments using an NVIDIA GeForce GTX 200 120GB GPU. On a single GPU card, we could run up to 5 seeds in parallel.

Confidential reviewer copy. This manuscript is under double-blind review by ICML 2026. Unauthorized sharing, redistribution, or disclosure is strictly prohibited.



(a) Success rate boost with scaling to bigger networks on grid size of 4 in generalized setting with discount 0.9



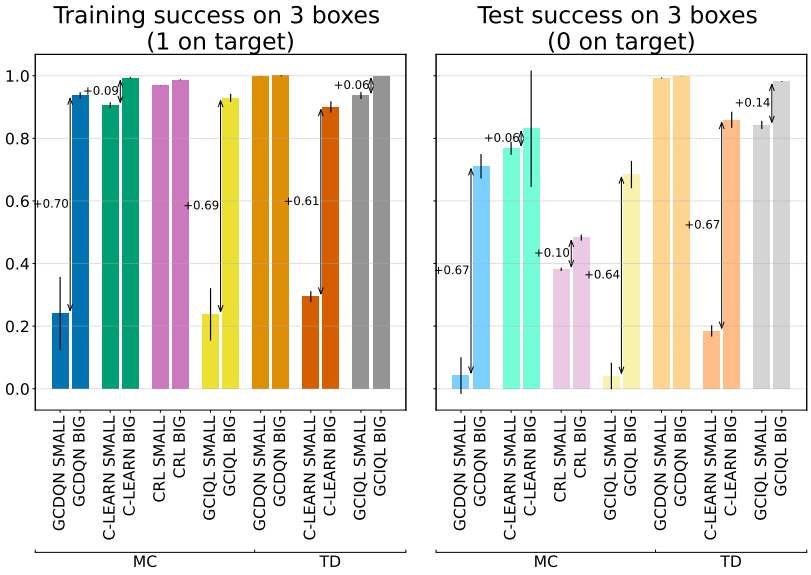
(b) Success rate boost with scaling to bigger networks on grid size of 4 in generalized setting with discount 0.99

Figure 13. **Full discount generalized setting scaling experiments for grid size of 4**

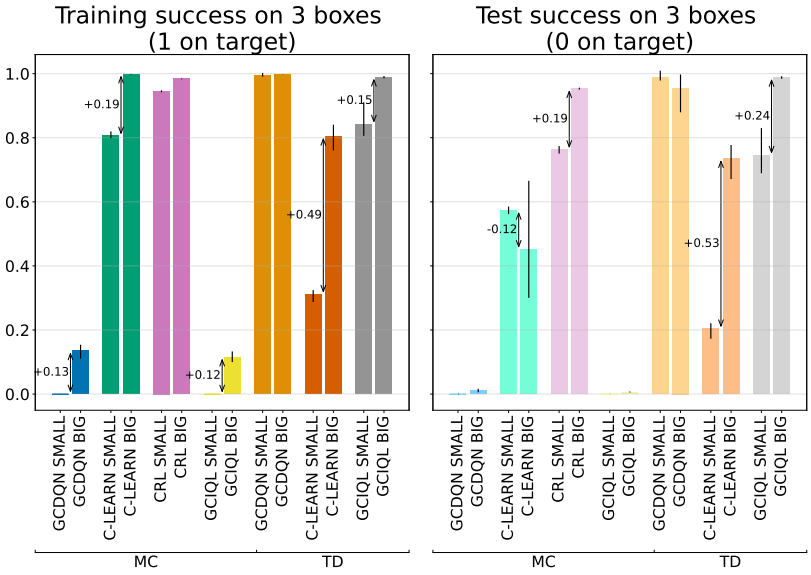
Depth	DQN (TD)	DQN (MC)	CRL (MC)	IQL (MC)	IQL (TD)	C-LEARN (MC)	C-LEARN (TD)
Small	1.03	1.07	1.32	1.19	1.25	0.83	1.27
Large	7.26	4.77	8.77	6.56	8.41	5.49	8.80

Table 2. Average wall-clock training time (in hours) for the methods used in this project.

Confidential reviewer copy. This manuscript is under double-blind review by ICML 2026. Unauthorized sharing, redistribution, or disclosure is strictly prohibited.



(a) Success rate boost with scaling to bigger networks on grid size of 5 in generalized setup with discount 0.9



(b) Success rate boost with scaling to bigger networks on grid size of 5 in generalized setup with discount 0.99

Figure 14. **Full discount generalized setup scaling experiments for grid size of 5**

Confidential reviewer copy. This manuscript is under double-blind review by ICML 2026. Unauthorized sharing, redistribution, or disclosure is strictly prohibited.

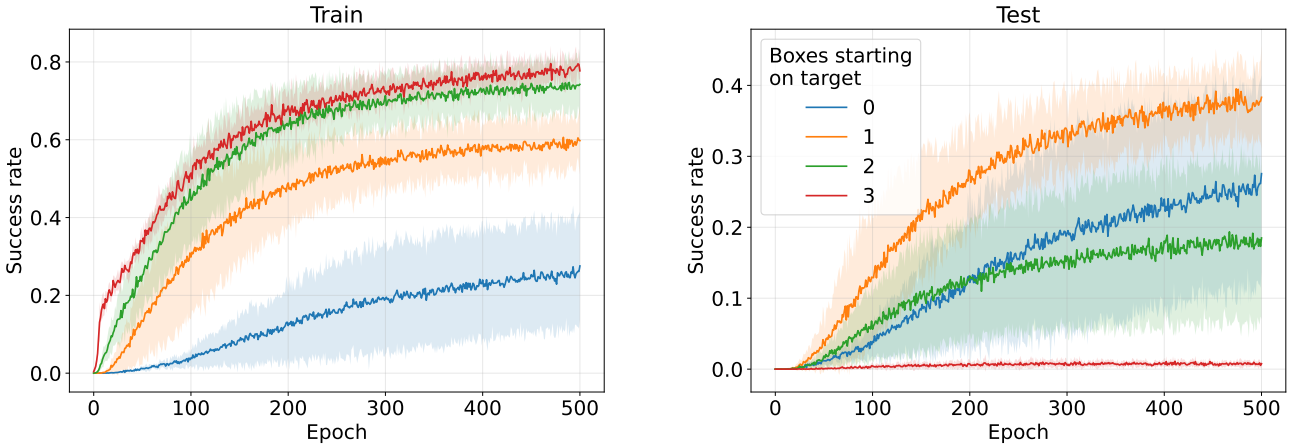


Figure 15. **MC version of Goal-conditioned DQN stitches, but learns slowly.** While the training of MC DQN is slow to converge, the stitching is present for this method even when moving only 3 or 2 boxes out of 4 during the training.

Confidential reviewer copy. This manuscript is under double-blind review by ICML 2026. Unauthorized sharing, redistribution, or disclosure is strictly prohibited.