How to Provably Improve Return Conditioned Supervised Learning?

Zhishuai Liu[1], **Yu Yang**[1], **Ruhan Wang**[2], **Pan Xu**[1], **Dongruo Zhou**[2]
[1]Duke University
[2]Indiana University Bloomington
{zhishuai.liu,yu.yang,pan.xu}@duke.edu, {ruhwang, dz13}@iu.edu

Abstract

In sequential decision-making problems, Return-Conditioned Supervised Learning (RCSL) has gained increasing recognition for its simplicity and stability in modern decision-making tasks. Unlike traditional offline reinforcement learning (RL) algorithms, RCSL frames policy learning as a supervised learning problem by taking both the state and return as input. This approach eliminates the instability often associated with temporal difference (TD) learning in offline RL. However, RCSL has been criticized for lacking the stitching property, meaning its performance is inherently limited by the quality of the policy used to generate the offline dataset. To address this limitation, we propose a principled and simple framework called Reinforced RCSL. The key innovation of our framework is the introduction of a concept we call the reinforced return. This mechanism leverages our policy to identify the best achievable in-dataset future return based on the current state, avoiding the need for complex return augmentation techniques. Our theoretical analysis demonstrates that Reinforced RCSL can consistently outperform the standard RCSL approach. Empirical results further validate our claims, showing significant performance improvements across a range of benchmarks.

1 Introduction

Reinforcement Learning (RL) has emerged as a powerful paradigm for decision-making and sequential learning [27], achieving remarkable successes across domains such as robotics [13, 25], healthcare [19, 36, 20], games [21, 29], and training large language models [9]. Among the various RL paradigms, offline RL [18, 12, 6] has gained substantial attention due to its ability to learn policies from pre-collected datasets without requiring interaction with the environment, which is particularly appealing in scenarios where exploration is costly, unsafe, or impractical. A key advantage of offline RL lies in its ability to leverage data generated by many existing policies, enabling the discovery of robust and effective behavior patterns.

Within offline RL, return-conditioned supervised learning (RCSL) has recently attracted significant traction [15, 16, 2, 3, 30]. RCSL reframes the policy learning problem as a supervised learning problem: the input consists of the state and the return from the current state, while the output is the optimal action for that state. Compared to classical offline RL algorithms, which primarily rely on dynamic programming (DP) approaches [17, 15], RCSL is easier to train, more straightforward to tune, and often achieves competitive performance across a variety of tasks. Notable RCSL methods include Decision Transformer (DT) [2] and Reinforcement Learning via Supervised Learning (RVS) [3]. However, a critical limitation of RCSL lies in its lack of stitching ability—that is, its inability to derive a policy that exceeds the performance of the policies used to generate the offline dataset. This limitation arises because RCSL tends to follow trajectories from the dataset without effectively combining the best parts of different trajectories, which is essential for achieving superior performance.

2nd Workshop on Aligning Reinforcement Learning Experimentalists and Theorists (ARLET 2025).

This limitation has been rigorously analyzed and demonstrated in [1], suggesting that RCSL's lack of stitching ability may be an inherent property of the approach.

Recently, several works have taken initial steps toward addressing this limitation and exploring potential solutions. For example, [34] proposed the QDT method, which relabels returns using a pre-learned optimal Q-function; [32] introduced the Elastic Decision Transformer, which dynamically adjusts the input sequence length; and [40] developed the Reinformer, which incorporates expectile regression into the Decision Transformer framework. These works have empirically demonstrated some level of stitching ability, indicating that enhancing stitching ability within RCSL methods is indeed possible. However, the theoretical understanding of RCSL remains underdeveloped compared to dynamic programming-based RL methods, where theoretical guarantees are more mature and extensively studied. Bridging this gap between empirical success and theoretical guarantees remains an open challenge. In this context, we pose the following critical question:

Can we improve return-conditioned supervised learning to have provable stitching ability?

Our main contributions are listed as follows.

- We introduce **reinforced RCSL ($\mathbf{R^2}$CSL)**, an advancement over RCSL that strictly improves its performance. The key innovation of R^2CSL is a new concept called the *in-distribution optimal return-to-go (RTG)*, which characterizes the highest accumulated reward a RCSL method can achieve under the offline trajectory distribution. This quantity can be directly learned via supervised learning, thereby avoiding the need for dynamic programming, which is commonly used in classical offline RL methods [17]. We demonstrate that R^2CSL, by incorporating the in-distribution optimal RTG, learns an *in-distribution optimal stitched policy* that surpasses the best policy achievable by traditional RCSL methods. To the best of our knowledge, this is the first work to **provably surpass RCSL without dynamic programming**.
- We provide a **sample complexity analysis** for various environments, including tabular MDPs and MDPs with general function approximation. We show that the sample complexity of R^2CSL to achieve the in-distribution optimal stitched policy is of the same order as classical RCSL methods [1], while R^2CSL converges to a superior policy. We further propose two realizations of R^2CSL using *expectile regression* [22] and *quantile regression* [14] for the in-distribution optimal RTG estimation and analyze their theoretical guarantees.
- We conduct comprehensive experimental studies under a simulated point mass environment, the D4RL gym and Antmaze environments to showcase the effectiveness of the R^2CSL algorithm. Experiment results demonstrate that (1) R^2CSL achieves the stitching ability, and outperforms RCSL-type algorithms like RvS and DT; (2) The R^2CSL framework is also flexible enough to incorporate dynamic programming components, and it achieves performance comparable to the state-of-the-art QT algorithm [10], while maintaining its simplicity.

2 Problem formulation

Reinforcement learning. We consider episodic Markov Decision Processes (MDPs) in this work, where each MDP is represented by a tuple $(\mathcal{S}, \mathcal{A}, P, r, H, \rho)$. Here, $\mathcal{S}$ denotes the state space, $\mathcal{A}$ is the action space, $P(s'|s,a)$ specifies the probability of transitioning to state s' after taking action a at state s, and the reward function $|r_h(s,a)| \leq 1$ assigns a reward to taking action a at state s. The horizon length H indicates the fixed number of steps in each episode, and $\rho(s)$ defines the initial state distribution, giving the probability of starting with state s.

The agent interacts with the environment with a policy $\pi : \mathcal{S} \times [H] \to \Delta(\mathcal{A})$ in a policy class Π. At each timestep h, the agent observes the current state $s_h \in \mathcal{S}$, selects an action $a_h \in \mathcal{A}$ according to its policy π_h, and transitions to the next state s_{h+1}, which is sampled from the transition dynamics $P(\cdot|s_h, a_h)$. Simultaneously, the agent receives a reward $r_h = r_h(s_h, a_h) \in [0,1]$. The objective of the agent is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected cumulative reward over an episode: $\mathbb{E}_\pi^P\big[\sum_{h=1}^H r(s_h, a_h)\big]$, where the expectation is taken over the randomness in the initial state distribution ρ, the policy π, and the transition dynamics P. The value function $V_h^\pi(s) = \mathbb{E}_\pi\big[\sum_{t=h}^H r_t(s_t, a_t) \mid s_h = s\big]$ represents the expected cumulative reward starting from state s at timestep h and following policy π thereafter. The Q-function is given by: $Q_h^\pi(s,a) = r_h(s,a) + \mathbb{E}_{s' \sim P_h(\cdot|s,a)}\big[V_{h+1}^\pi(s')\big]$. The optimal value function $V_h^*(s)$ and the optimal Q-function

$Q_h^*(s,a)$ are defined similarly but correspond to the optimal policy π^*, which maximizes the expected cumulative reward. Finally, we define $J(\pi) = \mathbb{E}_{s_1\sim\rho} V_1^\pi(s_1)$.

The offline setting. We consider an offline dataset $\mathcal{D}$ collected by a behavior policy β, where the dataset size is $|\mathcal{D}| = N$. $\mathcal{D}$ consists of trajectories: $\mathcal{D} = \{\tau^k\}_{k=1}^N$, with each trajectory τ^k represented as $\tau^k = (s_1^k, a_1^k, r_1^k, \cdots, s_H^k, a_H^k, r_H^k)$, where $s_1^k \sim \rho$ is the initial state drawn from the initial state distribution ρ, $a_h^k \sim \beta(\cdot|s_h^k)$ is the action selected by the behavior policy β at step h, and $s_{h+1}^k \sim P(\cdot|s_h^k, a_h^k)$ is the next state sampled from the transition dynamics P. We use $d_h^\beta(s)$ to denote the state distribution of state s under the behavior policy β at step h. Our goal is to leverage this offline dataset $\mathcal{D}$ to learn an effective policy that performs well in the underlying environment.

The RCSL framework The RCSL framework aims to learn a policy by modeling the distribution of actions conditioned on the state, the stage and the return of the trajectory, denoted by $\pi : \mathcal{S} \times [H] \times \mathbb{R} \to \Delta(\mathcal{A})$. Specifically, RCSL algorithms optimize the policy by minimizing the empirical negative log-likelihood loss over the offline dataset $\mathcal{D}$: $\hat{\pi} = \operatorname{argmin}_{\pi\in\Pi} \hat{L}(\pi)$, where

$$\hat{L}(\pi) = -\sum_{\tau\in\mathcal{D}} \sum_{t=1}^{H} \log \pi(a_t \mid s_t, t, g(\tau, t)), \tag{2.1}$$

where $g(\tau, h) = \sum_{t=h}^{H} r_t$ computes the return-to-go (RTG, [2]) along τ, starting from step h. This optimization aligns the learned policy π with the observed behavior in the offline dataset, incorporating both the states and return-based context. At test time, the RCSL algorithms utilize the learned policy $\hat{\pi}$ along with a test-time conditioning function $f(s,h)$, which determines the desired return to condition the policy. The resulting test-time policy π_f is then defined as $\pi_f(a|s,h) := \hat{\pi}(a|s, f(s,h))$, where π_f produces actions conditioned on the current state s and the test-time return determined by $f(s,h)$. This design enables flexible adaptation of the policy to different test-time objectives by modifying the conditioning function $f(s,h)$.

3 The reinforced RCSL

In this section, we introduce our algorithm, R^{2}CSL, discussing its core intuition and the reasons it should be preferred. To provide context, we first revisit existing analyses of the failure modes encountered by classical RCSL, highlighting the limitations that motivate the design of R^{2}CSL.

Why does RCSL fail to stitch? We revisit the findings in [1], which suggest that the policy returned by the default RCSL framework is unable to outperform the behavior policy used to generate the dataset $\mathcal{D}$. Given a pretrained RCSL policy $\hat{\pi}$, classical RCSL approaches such as Decision Transformer (DT) [2] and Return-Conditioned Supervision (RVS) [3] effectively rely on a conditioning return function f that satisfies the following conditions [1]:

- **In-distribution condition:** The initial return $f(s_1, 1)$ must be achievable with non-zero probability under the behavior policy β.
- **Consistency condition:** For any trajectory, the return function must satisfy $f(s_h, h) = f(s_{h+1}, h+1) + r_h(s_h, a_h)$, ensuring consistency across steps.

These conditions are crucial for ensuring that, at each step h, there is no out-of-distribution (OOD) issue with the inputs to the policy π_f. While these conditions guarantee the validity of the learned policy, they significantly restrict the range of feasible conditioning functions f. Specifically, this implies that the value domain of f can only be selected as $g(\tau, h)$, where τ represents any trajectory that could appear in the dataset $\mathcal{D}$ generated under the behavior policy β. Therefore, the return of trajectories generated by π_f will also be upper bounded by $g(\tau, 1)$, which performs no better than the best trajectory in the dataset $\mathcal{D}$. For illustration, we provide a toy example in Figure 1 to show RCSL fails to stitch. In this example, $\mathcal{A} = \{a^1, a^2, a^3\}$, $H = 3$ and $\mathcal{S} = \{s\}$. The state is unique and remains unchanged across stages. Each row represents a trajectory. The number outside (inside) the parentheses are rewards (return-to-go). For RCSL, we can only simply choose f to be 80, 81 or 75 as the conditioning return at the initial stage, and the 'optimal' RCSL policy

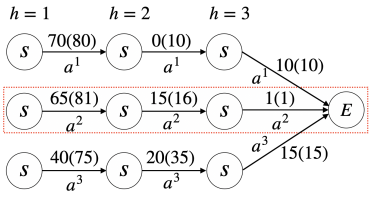


Figure 1: An example when RCSL fails to stitch.

would choose a^2 at each stages, i.e., the trajectory in the red box. However, a better trajectory can be obtained by stitching, e.g. taking the action sequence (a^2, a^3, a^3).

3.1 Algorithm description

To address the non-stitching issue discussed above, we introduce R^{2}CSL. R^{2}CSL discards the consistency condition, thereby allowing greater flexibility in the selection of the conditioning function f. To achieve the stitching ability, at each time step h, R^{2}CSL looks ahead to search for the maximum in-distribution RTG as its conditioning function, instead of following the original return-to-go in trajectory τ. This increased flexibility and 'optimal conditioning' enable R^{2}CSL to achieve superior results by overcoming the limitations of classical RCSL approaches.

Formally, we begin by introducing the *feasible set of trajectories* under the behavior policy β, $T_\beta := \{\tau \mid P_\beta(\tau) > 0\}$, where P_β is the trajectory distribution induced by β under the transition dynamics P. Based on this, we can define the *feasible set of conditioning functions*: $\forall \tau \in T_\beta$ and $(s_h, g_h) \in \tau$, there must exist a conditioning function f such that $f(s_h, h) = g_h$. The feasible conditioning function set $\mathcal{F}_\beta$ is defined as:

$$\mathcal{F}_\beta := \{f : \mathcal{S} \times [H] \to [0, H] \mid \forall (s, h) \in \text{dom}(f), \exists\, \tau \in T_\beta \text{ s.t. } s_h = s \text{ and } f(s, h) = g_h\},$$

where $\text{dom}(f)$ is the domain of f. Notably, the conditioning functions in $\mathcal{F}_\beta$ are not constrained by the consistency assumption. At any stage $h \in [H]$, *the feasible set of states* is defined as $\mathcal{S}_h^\beta := \{s \in \mathcal{S} \mid d_h^\beta(s) > 0\}$. For any feasible state $s \in \mathcal{S}_h^\beta$ at stage h, the *local feasible conditioning function set* is defined as $\mathcal{F}_\beta(s, h) := \{f : \mathcal{S} \times [H] \to [0, H] \mid f \in \mathcal{F}_\beta \text{ and } (s, h) \in \text{dom}(f)\}$.

In-distribution optimal stitched policy. Using $\mathcal{F}_\beta$, we define the in-distribution optimal RTG $f^\star$ as $f^\star(s, h) := \text{argmax}_{f \in \mathcal{F}_\beta(s,h)} f(s, h)$ which represents the maximum in-distribution RTG starting from (s, h) in the offline dataset. Based on this, we define the in-distribution optimal stitched policy $\pi_\beta^\star$ as $\pi_\beta^\star(a \mid s, h) := P_\beta(a \mid s, h, f^\star(s, h))$, where the policy is conditioned on the optimal return-to-go $f^\star(s, h)$. As an extension, we define a broader class of return-conditioned policies $\pi_f(a \mid s, h) := P_\beta(a \mid s, h, f(s, h)), \forall f \in \mathcal{F}_\beta$. We denote the subset of conditioning functions that satisfy the consistency condition as $\mathcal{F}_\beta^{\text{C}} \subset \mathcal{F}_\beta$. Using this, we present our first theorem, which highlights the superiority of the optimal stitched policy. Here, we assume a deterministic transition P and deterministic rewards r_h. The initial state distribution ρ and the behavior policy β remain stochastic.

Theorem 3.1. For any $f \in \mathcal{F}_\beta^{\text{C}}$, we have $J(\pi_\beta^\star) \geq J(\pi_f)$ for all $f \in \mathcal{F}_\beta^{\text{C}}$. In a word, the value achieved by the optimal stitched policy $\pi_\beta^\star$, equipped with $f^\star \in \mathcal{F}_\beta$, is always at least as good as that of policies constrained by the classical RCSL consistency condition.

Theorem 3.1 shows that conditioning function-optimal in-distribution RTG $f^\star$ enables stitching with a better trajectory at each time step. However, in practice, the optimal in-distribution RTG $f^\star$ is unknown, and consequently, so is the in-distribution optimal stitched policy $\pi_\beta^\star$.

We propose our algorithm R^{2}CSL in Algorithm 1 to construct the maximum in-distribution RTG estimation and estimate the in-distribution optimal stitched policy. During training, R^{2}CSL first follows the RCSL framework by minimizing the empirical negative log-likelihood loss defined in (2.1). Then it additionally estimates the maximum in-distribution RTG function (Line 2). Here we do not specify a particular $\hat{f}$ estimation procedure. It should be instantiated under specific settings. During inference, R^{2}CSL uses the maximum in-distribution RTG estimation as the condition to construct the in-distribution optimal stitched policy estimation $\hat{\pi}_\mathcal{D}^\star$.

Algorithm 1 The Reinforced RCSL (R^{2}CSL)

Require: The offline dataset $\mathcal{D}$.

1: Set $\hat{\pi} = \text{argmin}_{\pi \in \Pi} \hat{L}(\pi)$ following (2.1)
2: Obtain the maximum in-distribution RTG function estimation, $\hat{f}^\star(s, h)$.
3: Receive the initial state s_1.
4: **for** $h = 1, \cdots, H$ **do**
5: Establish $\hat{\pi}_\mathcal{D}^\star(\cdot|s_h, h) = \hat{\pi}(\cdot|s_h, h, \hat{f}^\star(s_h, h))$.
6: Implement $a_h \sim \hat{\pi}_\mathcal{D}^\star(\cdot|s_h, h)$ and receive the next state s_{h+1}.
7: **end for**

4 Finite-sample analysis of R^{2}CSL

4.1 Warm-up analysis for deterministic environments

In this section, we study R^{2}CSL realization under different environment setups, and provide finite-sample guarantees for variants of R^{2}CSL. We start with a *deterministic* environment, under which we instantiate and analyze R^{2}CSL to provide clearer insights into its behavior and advantages. We assume finite state and action spaces. For notational simplicity, we redefine a trajectory as $\tau = (s_1, a_1, g_1, s_2, a_2, g_2, \ldots, s_H, a_H, g_H)$, where g_h represents the RTG at stage h.

Given the current state s_h at stage h, we define $T_{\mathcal{D}}(s_h) = \{k \in [N] \mid s_h^k = s_h\}$, representing the empirical feasible index pool at s_h. We set $\hat{f}^\star(s_h, h) = \operatorname{argmax}_{k \in T_{\mathcal{D}}(s_h)} g_h^k$ in Algorithm 1, say, it assigns the *empirical in-distribution optimal RTG* from the dataset to the conditioning function. It subsequently determines the *empirical in-distribution optimal stitched policy*: $\hat{\pi}^\star_{\mathcal{D}}(\cdot|s_h, h) = \hat{\pi}(\cdot|s_h, h, \hat{f}^\star(s_h, h))$. Thus, Algorithm 1 effectively follows the steps outlined in Section 3.1 to learn the in-distribution optimal stitched policy π^*_β using an empirical dataset $\mathcal{D}$ instead of the behavior policy itself.

Theoretical guarantee. The policy learned by Algorithm 1, denoted as $\hat{\pi}^\star_{\mathcal{D}}$, is an estimate of $\pi^\star_\beta$. We now analyze its finite-sample theoretical guarantee. At a high level, achieving a reliable estimation requires: 1) the empirical in-distribution optimal RTG, $\hat{f}^\star$, to be accurate, and 2) sufficient coverage of the offline dataset over the trajectories induced by the in-distribution optimal stitched policy.

We begin by stating a standard assumption on the regularity of the policy class Π, following [1].

Assumption 4.1. For the policy class Π, we assume it is finite, and

- For all $(a, s, g, h, a', s', g', h')$, $\pi \in \Pi$, we have $|\log \pi(a \mid s, h, g) - \log \pi(a' \mid s', h', g')| \le c$.
- The approximation error of MLE is bounded by δ_{approx}, i.e., $\min_{\pi \in \Pi} L(\pi) \le \delta_{\text{approx}}$, where $L(\pi) = \mathbb{E}_{s \sim P_\beta} \mathbb{E}_{g \sim P_\beta(\cdot|s)} \big[D_{\text{KL}}(P_\beta(\cdot \mid s, g) \| \pi(\cdot \mid s, g)) \big]$ is the expected loss.

Next, we introduce an assumption on the data distribution, which characterizes how well the offline dataset covers the target policy.

Assumption 4.2. Given the behavior policy β, we assume:

- There exists a constant $\tilde{c} > 0$ such that for all $(s, h) \in \text{dom}(f^\star)$, $P_\beta(g_h = f^\star(s, h) \mid s_h = s) \ge \tilde{c}$.
- There exists a constant $c^\star_\beta > 0$ such that for all $(h, s) \in [H] \times \mathcal{S}^\beta_h$, we have $d^{\star,\beta}_h(s)/d^\beta_h(s) \le c^\star_\beta$, where $d^{\star,\beta}_h$ is the occupancy measure on states at step h induced by $\pi^\star_\beta$.

Assumption 4.2 imposes a *partial*-type coverage assumption: it only requires the behavior policy (or offline dataset) to cover both the maximum in-distribution return-to-go and the state visitation distribution induced by the optimal stitched policy $\pi^\star_\beta$. Comparing Theorem 4.3 with Corollary 3 of [1], the term $c^\star_\beta$ plays a role analogous to $C_f := \sup_{f \in \mathcal{F}^C_\beta} P_{\pi^{\text{RCSL}}_f}(s)/P_\beta(s)$, which captures the worst-case distribution mismatch. Similarly, our term $\tilde{c}$ corresponds to α_f, the lower bound on return coverage, ensuring that $P_\beta(g = f(s, h) \mid s_h = s) \ge \alpha_f$ for all $f \in \mathcal{F}^C_\beta$.

Let $d^\beta_{\min} := \min_{h,s} \{d^\beta_h(s) \mid d^\beta_h(s) > 0\}$ denote the smallest positive entry of the distribution d^β. Next, we formally state the theoretical guarantee for Algorithm 1.

Theorem 4.3. Under Assumptions 4.1 and 4.2, if we set $\hat{f}^\star(s_h, h) = \operatorname{argmax}_{k \in T_{\mathcal{D}}(s_h)} g_h^k$ in Algorithm 1, then for any $\delta \in (0, 1)$, when $N > \log(SH/\delta)/\log(1 - d^\beta_{\min} \cdot \tilde{c})$, with probability at least $1 - 2\delta$, we have

$$J(\pi^\star_\beta) - J(\hat{\pi}^\star_{\mathcal{D}}) \le O\Big(\frac{c^\star_\beta H^2}{\tilde{c}} \Big(\sqrt{c} \Big(\frac{\log |\Pi|/\delta}{N} \Big)^{1/4} + \sqrt{\delta_{\text{approx}}} \Big) \Big).$$

Theorem 4.3 shows that R^{2}CSL converges to $\pi^\star_\beta$, which outperforms the policies π_f for $f \in \mathcal{F}^{\text{C}}_\beta$ studied by [1]. It also indicates that the sample complexity of Algorithm 1 depends on the approximation error of MLE. This error can be eliminated by selecting a sufficiently expressive function class, such as deep neural networks. Additionally, the convergence rate is $N^{-1/4}$, which is

slower than the standard rate of $N^{-1/2}$ commonly established in the offline RL literature. We believe this discrepancy is due to a limitation in the current analysis, and we aim to refine it in future work.

4.2 Analysis for stochastic environments

Next, we consider a more general setting where the state and action spaces are large, and the underlying environment is *stochastic*. In this case, we can no longer determine the empirical in-distribution optimal RTG by directly selecting the RTG from offline datasets. To address this challenge, we propose to estimate $\hat{f}^\star$ by general function approximation. For now, we do not specify the estimation method for $\hat{f}^\star$, it can be instantiated using expectile regression or quantile regression in later sections. We now outline the assumptions necessary for the theoretical guarantee of Algorithm 1 with general function approximation of the optimal in-distribution RTG.

Assumption 4.4. For the conditioning function $\hat{f}^\star$ and the policy class Π, we assume:

- There exists an error function $\mathrm{Err}(N, \delta, \tilde{c})$ that depends on the sample size N and failure probability δ, such that $\mathbb{E}_{s\sim d_h^\beta}\big[(f^\star(s,h) - \hat{f}^\star(s,h))^2\big] \le \mathrm{Err}(N,\delta,\tilde{c}), \forall h \in [H]$.
- For any $(s,h,\pi) \in \mathcal{S}\times[H]\times\Pi$, given $g_1 \neq g_2$, there exists a constant $\gamma > 0$ such that $\mathrm{TV}(\pi(\cdot \mid s,h,g_1)\|\pi(\cdot \mid s,h,g_2)) \le \gamma|g_1 - g_2|$, where $\mathrm{TV}(\cdot\|\cdot)$ denotes the total variation distance.

The first condition in Assumption 4.4 ensures that the estimation of the conditioning function is sufficiently accurate. It is not meant to introduce additional constraints, but rather to provide a general and abstract formulation—captured via terms like $\mathrm{Err}(N,\delta,\tilde{c})$—that subsumes a wide range of cases. The second condition guarantees that small errors in the conditioning function do not result in significant divergence in the estimated return-conditioned policy. Next we state our theorem.

Theorem 4.5. Assume Assumptions 4.1 and 4.2 hold. Additionally, if the conditioning function $\hat{f}^\star$ and the policy class Π satisfy Assumption 4.4, then for any $\delta \in (0,1)$, with probability at least $1-2\delta$, the policy $\hat{\pi}^\star_{\mathcal{D}}$ learned by Algorithm 1 satisfies

$$J(\pi^\star_\beta) - J(\hat{\pi}^\star_{\mathcal{D}}) \le O\Big(\frac{c^\star_\beta H^2}{\tilde{c}}\Big(\sqrt{c}\Big(\frac{\log|\Pi|/\delta}{N}\Big)^{1/4} + \sqrt{\delta_{\text{approx}}}\Big) + c^\star_\beta H^2 \gamma \sqrt{\mathrm{Err}(N,\delta,\tilde{c})}\Big).$$

Compared to Theorems 4.3 and 4.5 introduces an additional approximation error term, $\mathrm{Err}(N,\delta,\tilde{c})$. While we retain the flexibility to choose our estimation method, we are particularly interested in approaches such as *expectile regression* [32] and *quantile regression* [14], which can potentially achieve error bounds of order $O(1/N)$. We further analyze their properties in the next section.

5 Practical implementation: expectile v.s. quantile regression

In this section, we study how different function approximation procedure-especially those utilized in literature-of the conditioning function $\hat{f}^*$ would affect the learned algorithm, under the simple *tabular* setting. Existing literature [32, 40] leverage the expectile regression [22] due to its simplicity. It returns the empirical conditioning function by $\hat{f}^\star = \mathrm{argmin}_{\tilde{f}\in\mathcal{F}} \sum_{k=1}^K \big[L_2^\alpha(g_h^k - \tilde{f}(s_h^k,h))\big]$, where $L_2^\alpha(u) = |\alpha - \mathbb{1}(u<0)|u^2$ and α is the hyperparameter in order to control how close expectile regression is to the vanilla L_2 regression. However, it is trivial to show that the expectile estimator with $\alpha \neq 1$ could lead to out of distribution RTG. Formally, there exists a tabular MDP and a behavior policy β such that the R^{2}CSL with the expectile regression for $\hat{f}^\star$ estimation can not find the optimal policy $\pi^\star_\beta$ for sure. We postpone the proof to Appendix C.4. The take away message is that the L_2 loss of the expectile regression leads to out-of-distribution returns when $\alpha \neq 1$, due to the fact that L_2 loss is less robust to the noise.

To address this issue, we consider the quantile regression [14], which returns the empirical conditioning function by $\hat{f}^\star = \mathrm{argmin}_{\tilde{f}\in\mathcal{F}} \sum_{k=1}^K L_1^\alpha(g_h^k - \tilde{f}(s_h^k,h))$, where $L_1^\alpha(u) = |\alpha - \mathbb{1}(u<0)|\cdot|u|$ is the L_1 loss. Generally speaking, the L_1 loss is more robust to the noise of the return, which makes Algorithm 1 with quantile regression for $\hat{f}^\star$ estimation better than its expectile regression counterpart. In the following theorem, we state that with a large sample size, the R^{2}CSL with quantile regression can exactly recover the maximum in-distribution RTG, finding the optimal policy. In contrast, R^{2}CSL with expectile regression introduces bias in $f^\star$ estimation, learns out-of-distribution RTGs, and fails to find the optimal policy. We make the following assumption here.

Assumption 5.1. We assume that the environment is deterministic. Besides, we assume there is no tie in the RTG: for any $h \in [H], \forall \tau^1, \tau^2 \in T_\beta$, such that $s_h^1 = s_h^2$ and $a_h^1 \neq a_h^2$, we have $g_h^1 \neq g_h^2$.

Assumption 5.1 essentially suggests a setting where all trajectories generated by the behavior policy β can be 'ranked' based on their RTG. Under Assumption 5.1, the optimal policy $\pi_\beta^\star$ can be represented by a trajectory starting from the initial state, which enables a simplified analysis of how various training methods influence $\hat{f}$.

Theorem 5.2. Assume that for all $(s, h) \in \text{dom}(f^\star)$, there exists an constant $\tilde{c}$, such that $P_\beta(g_h = f^\star(s, h) | s_h = s) \geq \tilde{c}$. We set $\alpha > 1 - \tilde{c}/2$ in the L_2^α loss with quantile regression. Then for any $\delta \in (0, 1)$, when $N \geq \max\{\frac{2}{d_{\min}^{\beta,2}} \log \frac{2SH}{\delta}, \frac{4}{\tilde{c}^2 d_{\min}^{\beta}} \log \frac{2SH}{\delta}\}$ with probability at least $1 - \delta$, we have $J(\pi_\beta^\star) = J(\hat{\pi}_\mathcal{D}^\star)$, where $\hat{\pi}_\mathcal{D}^\star$ is the policy learned by Algorithm 1.

Finally, to further extend the notion of in-distribution optimal RTG, we study the *multi-step in-distribution optimal RTG*. We prove that by increasing the number of steps considered in the in-distribution optimal RTG, R^{2}CSL is capable of finding the *optimal in-distribution policy*, a result that was previously only achievable by dynamic programming-based algorithms [17]. Due to space limit, we postpone related content to Appendix B.

6 Experiments

In this section, we conduct several numerical experiments to answer the following two questions: *How does different choice of hyperparameter α affect R^{2}CSL's stitching ability?* and *How does R^{2}CSL compare to existing extensions of RCSL in literature?*

6.1 An illustration of R^{2}CSL's stitching ability

To answer the first question, we conduct a simulation study in PointMaze to showcase the stitching ability of Algorithm 1 with expectile regression and quantile regression, and study how different choices of α affect stitching. The simulated environment is a point-mass navigation task. Red dots represent starting points and the green dot is the goal state. The agent learns a policy to reach the goal from various starting positions. The offline dataset contains two types of trajectories: **Type I**: starting from the left red point and going directly to the goal; **Type II**: starting from the bottom red point and moving upward without reaching the goal. To succeed when starting from the bottom red point, the agent must **learn to stitch**—i.e., combine information from Type II and Type I trajectories to reach the goal. We also inject action noise at each step to evaluate generalization.

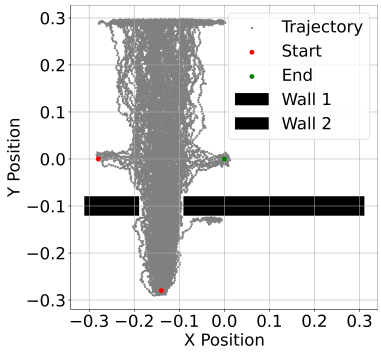
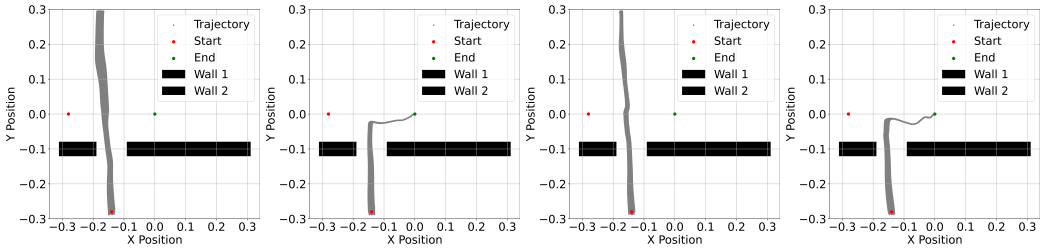


Figure 2: The simulated PointMaze environment

We vary the proportion of Type I trajectories to simulate different levels of coverage of the optimal return-to-go in the offline dataset, which corresponds to $\tilde{c}$ in Assumption 4.2. To analyze how this affects performance, we test Algorithm 1 leveraging expectile regression and quantile regression with various values of the hyperparameter α for $\hat{f}^\star$ estimation. We find that with **10%** Type I trajectories, $\alpha = 0.95$ enables successful stitching (Figures 3(b) and 3(d)), while $\alpha = 0.85$ fails (Figures 3(a) and 3(c)); With **1%** Type I trajectories, a higher $\alpha = 0.99$ is required to achieve stitching (Figures 4(b) and 4(d)), whereas $\alpha = 0.90$ fails (Figures 4(b) and 4(c)). Thus, the results are consistent with our theoretical findings in Theorem 5.2, the hyperparameter α should be chosen according to the underlying coverage factor $\tilde{c}$.

6.2 D4RL benchmark

To answer the second question, we test several R^{2}CSL variants under the D4RL Gym (halfcheetah, hopper and walker2d) and Antmaze environments [5]. Due to space limit, details on experiment setup, implementation and additional results are postponed to Appendix E.



(a) Expectile ($\alpha = 0.85$) (b) Expectile ($\alpha = 0.95$) (c) Quantile ($\alpha = 0.85$) (d) Quantile ($\alpha = 0.95$)

Figure 3: Illustration of the stitching ability of R^{2}CSL - Proportion of type I trajectories in the dataset is set to 0.1. R^{2}CSL with $\alpha = 0.85$ fails to stitch, while with $\alpha = 0.95$ succeeds to stitch.

Table 1: Normalized score on D4RL Gym for RvS and R^{2}CSL with expectile ($\alpha = 0.99$) and quantile ($\alpha = 0.99$) regression respectively. The inference of RvS is conditioned on three target RTGs. We report the mean and standard deviation of the normalized score for five seeds.

Dataset	**RVS**			**R^{2}CSL-Expectile**	**R^{2}CSL-Quantile**
	0.7	0.9	1.1		
halfcheetah-medium	43.10±0.64	36.78±3.04	25.47±4.67	42.09±0.50	42.24±0.35
halfcheetah-medium-replay	15.24±7.60	9.59±6.12	5.17±4.72	38.07±0.91	38.71±0.46
halfcheetah-medium-expert	85.48±1.32	90.97±1.26	91.09±1.11	90.95±1.18	92.25±0.62
hopper-medium	52.05±3.90	47.91±4.72	38.49±13.57	54.49±4.18	53.00±9.62
hopper-medium-replay	52.53±22.07	53.49±29.42	34.30±14.06	46.83±14.55	53.27±20.74
hopper-medium-expert	65.00±6.82	96.18±20.65	106.54±8.25	106.62±8.17	100.75±10.92
walker2d-medium	70.94±4.11	72.25±2.88	68.82±3.79	71.18±4.22	71.66±4.33
walker2d-medium-replay	41.24±12.86	47.25±17.79	26.58±20.09	36.34±10.52	44.85±9.00
walker2d-medium-expert	65.06±1.15	69.23±3.70	106.22±0.58	101.03±6.29	105.18±0.70
Total	490.62	523.66	502.68	587.61	**601.91**

R^{2}CSL with RvS. We implement Algorithm 1 with expectile regression and quantile regression based on the RvS [3], heuristically motivated from the algorithm design and theoretical results developed, that can deal with large state and action spaces leveraging powerful function approximations. Experiment results are shown in Table 1. We can conclude that R^{2}CSL outperforms the RvS framework across all fixed target RTG fraction ratios. This demonstrates the effectiveness and robustness of our approach.

Ablation study The hyperparameter α in both expectile and quantile regression controls how these methods emphasize different regions of the return distribution. Our experiments systematically varied α across $0.9, 0.99, 0.999$ to study its impact on policy performances. As shown in Table 2, the performance of our methods initially increases as α increases, but with $\alpha = 0.999$, the performance of R^{2}CSL-Quantile does not improve further. Predicting higher RTGs generally leads to better performance when the conditioning function is learned within a reasonable range. However, extreme outliers such as those introduced with $\alpha = 0.999$ in R^{2}CSL-Expectile can hurt the performance.

R^{2}CSL with DT We extend our R^{2}CSL framework to DT based methods. In particular, we use the transformer architecture of DT to learn the policy π in Line 1 of Algorithm 1. We use MLP to conduct quantile regression and expectile regression to learn the conditioning function in Line 2 of Algorithm 1. We call this method DT-R2CSL. Experiment results are shown in Table 3. We can conclude that in most cases the best performance belongs to our proposed DT-extensions.

R^{2}CSL incorporating dynamic programming We extend our framework to hybrid methods that incorporate dynamic programming components. In particular, we implement a new method, DP-R2CSL, which integrates the policy learning module from QT [10]. QT remains, to the best of our knowledge, the state-of-the-art on D4RL benchmarks. This module leverages a pre-learned Q-value to regularize the cross-entropy loss used in policy estimation. Experiment results are shown in Table 3. We observe that DP-R^{2}CSL significantly outperforms DT-R^{2}CSL, and performs comparably to QT across all settings except hopper-medium and halfcheetah-medium-expert. This is expected, as QT shares a key feature with our RCSL framework—namely, the use of an optimal conditioning function.

Table 2: Normalized score on D4RL Gym for our methods with different α in expectile and quantile. We report the mean and standard deviation of the normalized score for five seeds.

Dataset	R^2CSL-Expectile			R^2CSL-Quantile		
	0.9	0.99	0.999	0.9	0.99	0.999
halfcheetah-medium	42.04±0.36	42.08±0.49	42.42±0.39	42.28 ± 0.26	42.24±0.35	42.47±0.42
halfcheetah-medium-replay	35.89±0.29	38.07±0.91	38.13±1.81	37.93±0.30	38.71±0.46	38.77±0.52
halfcheetah-medium-expert	89.85±1.58	90.95±1.18	90.90±1.09	91.70±1.09	92.25±0.62	91.81±0.98
hopper-medium	57.01±1.73	54.49±4.18	46.44±4.42	57.24±2.63	53.00±9.62	55.62±8.49
hopper-medium-replay	32.26±10.57	46.83±14.55	53.90±13.73	34.08±4.96	53.27±20.74	72.60±14.08
hopper-medium-expert	102.58±4.93	106.62±8.17	98.57±16.39	104.04±7.67	100.75±10.92	92.24±30.77
walker2d-medium	71.29±3.61	71.18±4.22	71.13±2.31	70.23±5.04	71.66±4.33	71.94±4.75
walker2d-medium-replay	29.87±7.25	36.34±10.52	39.46±19.24	16.60±5.60	44.85±9.00	55.62±5.00
walker2d-medium-expert	60.26±15.55	101.04±6.29	105.08±0.93	104.73±1.79	105.18±0.70	102.21±5.14
Total	521.05	587.61	586.03	558.84	601.91	**623.29**

Table 3: Normalized score on D4RL Gym for DT, QT, DT-R^2CSL and DP-R^2CSL with expectile ($\alpha = 0.99$) and quantile ($\alpha = 0.99$) regression respectively. We report the mean and standard deviation of the normalized score for five seeds.

Dataset	DT-R^2CSL-Expectile	DT-R^2CSL-Quantile	DT	DP-R^2CSL-Expectile	DP-R^2CSL-Quantile	QT
halfcheetah-medium	43.23±0.26	43.21±0.09	42.6±0.1	50.71±0.11	51.11±0.32	51.4±0.4
halfcheetah-medium-replay	38.17±1.10	37.17±1.68	36.6±0.8	48.40±0.48	48.42±0.30	48.9±0.3
halfcheetah-medium-expert	88.03±2.14	88.56±1.99	86.8±1.3	82.86±4.99	83.78±7.11	96.1±0.2
hopper-medium	68.95±11.70	70.24±11.80	67.6±1.0	74.58±13.68	71.01±5.56	96.9±3.1
hopper-medium-replay	83.21±4.53	82.86±5.93	82.7±7.0	98.92±0.43	98.45±0.93	102±0.2
hopper-medium-expert	104.13±3.45	105.50±2.53	107.6±1.8	112.31±0.61	112.73±0.28	113.4±0.4
walker2d-medium	82.88±1.70	81.50±1.37	74±1.4	87.82±0.27	89.50±4.94	88.8±0.5
walker2d-medium-replay	70.03±3.23	69.69±4.15	66.6±3.0	97.37±2.41	98.11±1.26	98.5±1.1
walker2d-medium-expert	109.59±0.66	109.09±0.83	108.1±0.2	110.09±0.51	111.45±1.21	112.6±0.6
Total	688.22	687.83	672.6	763.11	764.55	808.2

Specifically, QT selects the return-to-go (RTG) that maximizes the Q-value as its conditioning input (see Section 3.3 of [10] for details), which aligns with our principle of selecting the optimal in-distribution RTG.

7 Conclusion

We explore methods to provably enhance RCSL for effective trajectory stitching in offline datasets. To this end, we introduce R^2CSL, which leverages an in-distribution optimal RTG quantity. We show that R^2CSL can learn the in-distribution optimal stitched policy, surpassing the best policy achievable by standard RCSL. Furthermore, we provide a theoretical analysis of R^2CSL and its variants. Comprehensive experiment results demonstrate the effectiveness of the R^2CSL framework.

A notable limitation of the RCSL-type algorithms is that they can fail in stochastic environments. Specifically, Theorem 4.5 shows that the R2CSL algorithm can effectively recover the underlying objective policy $\pi^\star_\beta$, which is defined by the stochastic environment and the behavior policy. Prior works [4, 1] suggest that this objective policy $\pi^\star_\beta$ can be arbitrarily suboptimal, and we note that this is a fundamental limitation of RCSL-style algorithms. It remains an open problem to theoretically address this limitation based on the RCSL framework.

References

[1] David Brandfonbrener, Alberto Bietti, Jacob Buckman, Romain Laroche, and Joan Bruna. When does return-conditioned supervised learning work for offline reinforcement learning? *Advances in Neural Information Processing Systems*, 35:1542–1553, 2022. 2, 3, 5, 9, 13, 14, 21

[2] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021. 1, 3, 13, 21

[3] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021. 1, 3, 8, 13, 21

[4] Benjamin Eysenbach, Soumith Udatha, Russ R Salakhutdinov, and Sergey Levine. Imitating past successes can be very suboptimal. *Advances in Neural Information Processing Systems*, 35:6047–6059, 2022. 9

[5] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020. 7, 21, 22

[6] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021. 1

[7] Hiroki Furuta, Yutaka Matsuo, and Shixiang Shane Gu. Generalized decision transformer for offline hindsight information matching. *arXiv preprint arXiv:2111.10364*, 2021. 13

[8] Chen-Xiao Gao, Chenyang Wu, Mingjun Cao, Rui Kong, Zongzhang Zhang, and Yang Yu. Act: empowering decision transformer with dynamic programming via advantage conditioning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 12127–12135, 2024. 13

[9] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. 1

[10] Shengchao Hu, Ziqing Fan, Chaoqin Huang, Li Shen, Ya Zhang, Yanfeng Wang, and Dacheng Tao. Q-value regularized transformer for offline reinforcement learning. *arXiv preprint arXiv:2405.17098*, 2024. 2, 8, 9, 22

[11] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021. 13

[12] Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, pages 5084–5096. PMLR, 2021. 1

[13] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013. 1

[14] Roger Koenker and Kevin F Hallock. Quantile regression. *Journal of economic perspectives*, 15(4):143–156, 2001. 2, 6

[15] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021. 1

[16] Aviral Kumar, Xue Bin Peng, and Sergey Levine. Reward-conditioned policies. *arXiv preprint arXiv:1912.13465*, 2019. 1

[17] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020. 1, 2, 7, 14

[18] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. 1

[19] Ying Liu, Brent Logan, Ning Liu, Zhiyuan Xu, Jian Tang, and Yangzhi Wang. Deep reinforcement learning for dynamic treatment regimes on medical registry data. In *2017 IEEE international conference on healthcare informatics (ICHI)*, pages 380–385. IEEE, 2017. 1

[20] Zhishuai Liu, Jesse Clifton, Eric B Laber, John Drake, and Ethan X Fang. Deep spatial q-learning for infectious disease control. *Journal of Agricultural, Biological and Environmental Statistics*, 28(4):749–773, 2023. 1

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. 1

[22] Whitney K Newey and James L Powell. Asymmetric least squares estimation and testing. *Econometrica: Journal of the Econometric Society*, pages 819–847, 1987. 2, 6

[23] Keiran Paster, Sheila McIlraith, and Jimmy Ba. You can't count on luck: Why decision transformers and rvs fail in stochastic environments. *Advances in neural information processing systems*, 35:38966–38979, 2022. 13

[24] Juergen Schmidhuber. Reinforcement learning upside down: Don't predict rewards–just map them to actions. *arXiv preprint arXiv:1912.02875*, 2019. 13

[25] Bharat Singh, Rajesh Kumar, and Vinay Pratap Singh. Reinforcement learning in robotic applications: a comprehensive survey. *Artificial Intelligence Review*, 55(2):945–990, 2022. 1

[26] Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaśkowski, and Jürgen Schmidhuber. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019. 13

[27] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 1, 13

[28] Gerald Tesauro et al. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995. 13

[29] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019. 1

[30] Ruhan Wang, Yu Yang, Zhishuai Liu, Dongruo Zhou, and Pan Xu. Return augmented decision transformer for off-dynamics reinforcement learning. *arXiv preprint arXiv:2410.23450*, 2024. 1

[31] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992. 13

[32] Yueh-Hua Wu, Xiaolong Wang, and Masashi Hamaya. Elastic decision transformer. *Advances in Neural Information Processing Systems*, 36, 2024. 2, 6, 13

[33] Haoran Xu, Li Jiang, Li Jianxiong, and Xianyuan Zhan. A policy-guided imitation approach for offline reinforcement learning. *Advances in neural information processing systems*, 35:4085–4098, 2022. 13

[34] Taku Yamagata, Ahmed Khalil, and Raul Santos-Rodriguez. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl. In *International Conference on Machine Learning*, pages 38989–39007. PMLR, 2023. 2, 13

[35] Sherry Yang, Dale Schuurmans, Pieter Abbeel, and Ofir Nachum. Dichotomy of control: Separating what you can control from what you cannot. In *The Eleventh International Conference on Learning Representations*. 13

[36] Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1):1–36, 2021. 1

[37] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *international conference on machine learning*, pages 27042–27059. PMLR, 2022. 13

[38] Sirui Zheng, Chenjia Bai, Zhuoran Yang, and Zhaoran Wang. How does goal relabeling improve sample efficiency? In *Proceedings of the 41st International Conference on Machine Learning*, pages 61246–61266. PMLR, 2024. 13

[39] Hanlin Zhu and Amy Zhang. Provably efficient offline goal-conditioned reinforcement learning with general function approximation and single-policy concentrability. *Advances in Neural Information Processing Systems*, 36, 2024. 13

[40] Zifeng Zhuang, Dengyun Peng, Ziqi Zhang, Donglin Wang, et al. Reinformer: Max-return sequence modeling for offline rl. *arXiv preprint arXiv:2405.08740*, 2024. 2, 6, 13, 21

A Related Work

Empirical Studies about RCSL Conditional sequence modeling [26, 11, 24] has emerged as a promising approach to solving offline reinforcement learning through supervised learning. This paradigm learns from the offline dataset a behavior policy with state and return-to-go (RTG) as input, and then predicts subsequent actions by conditioning on the current state and a conditioning function that encodes specific metrics for future trajectories. Existing methods within the RCSL framework have demonstrated significant empirical success. In particular, DT [2, 7, 37] and RvS [3] adopt the vanilla RTG as the conditioning function to predict the optimal action. However, vanilla RCSL methods, such as DT and RvS, lack stitching ability [1], a crucial property of dynamic programming-based methods like Q-learning [31] and TD [27, 28]. To address this limitation, [34] proposed QDT, which enhances RCSL by relabeling RTGs in the dataset using a pre-trained optimal Q-function and then training a DT on the relabeled data. [32] introduced the Elastic Decision Transformer, which enables trajectory stitching during action inference by adjusting the history length used in DT to discard irrelevant or suboptimal past experiences. [40] proposed Reinformer, which incorporates an expectile regression model to estimate the maximum in-distribution RTG as the conditioning function. [35, 23] studied the limitations of RCSL in stochastic environments. [8, 33] propose DT-based and offline RL methods that utilize expectile regression Despite the empirical successes of these methods, none of them provide any theoretical guarantees on the stitching ability.

Theoretical Studies of RCSL Compared to the extensive theoretical studies on dynamic programming (DP)-based algorithms, the theoretical analysis of RCSL methods remains relatively limited. From a theoretical perspective, [1] examined the finite-sample guarantees of RCSL methods, including DT and RvS, and identified the fundamental challenge of sample complexity due to the need for sufficient return and state coverage. [38] investigated the goal-conditioned supervised learning (GCSL) setting and provided a regret analysis for a goal relabeling method. [39] also studied the GCSL setting, offering a finite-sample analysis for GCSL with f-divergence regularization. Our work falls within this line of research on RCSL, but we *rigorously establish* the sample complexity of R^{2}CSL, demonstrating that it achieves a policy superior to that of a classical RCSL method.

B Multi-Step vs. Single-Step Stitching

We have demonstrated that by incorporating a dataset-dependent optimal conditioning function f^*, our algorithm, R^{2}CSL, provably converges to the optimal stitched policy $\pi^\star_\beta$, which outperforms classical RCSL, as verified by our experiments. However, unlike dynamic programming-based algorithms, which can converge to the optimal policy $\pi^\star$ independently of the specific dataset, the relationship between $\pi^\star_\beta$ and $\pi^\star$ remains unclear. In this section, we extend the notion of the optimal stitched policy and explore this relationship further.

Multi-Step In-Distribution Optimal RTG We introduce a multi-step RTG relabeling procedure to enhance the R^{2}CSL framework based on the in-distribution optimal RTG function $f^\star$. This relabeling scheme iteratively predicts the optimal RTG from the current state. For any trajectory $\tau = (s_1, a_1, g_1, s_2, a_2, g_2, \ldots, s_H, a_H, g_H)$ in the feasible set T_β, we relabel the RTGs in a backward fashion for $k \geq 1$ iterations.

For the ease of discussion, we denote $\tilde{\tau}^0 = (s_1, a_1, \tilde{g}^0_1, s_2, a_2, \tilde{g}^0_2, \cdots, s_H, a_H, \tilde{g}^0_H) = \tau$, and $\tilde{T}^0_\beta = T_\beta$. Then for any $k \geq 1$, suppose we have $\tilde{T}^{k-1}_\beta$, we define the feasible conditioning function set after $k-1$ passes relabeling as:

$$\tilde{\mathcal{F}}^{k-1}_\beta := \{f_{k-1} : \mathcal{S} \times [H] \to \mathbb{R} \mid \forall (s,h) \in \text{dom}(f_{k-1}), \\ \exists\, \tilde{\tau}^{k-1} \in \tilde{T}^{k-1}_\beta \text{ s.t. } s_h = s \text{ and } f_{k-1}(s,h) = \tilde{g}^{k-1}_h\},$$

where $\text{dom}(f_{k-1})$ is the domain of f_{k-1}. At any stage $h \in [H]$, recall the feasible set of states is $\mathcal{S}^\beta_h := \{s \in \mathcal{S} \mid d^\beta_h(s) > 0\}$. For any feasible state $s \in \mathcal{S}^\beta_h$, the local feasible conditioning function set after one-pass of relabeling is defined as:

$$\tilde{\mathcal{F}}^{k-1}_\beta(s,h) := \{f_{k-1} : \mathcal{S} \times [H] \to \mathbb{R} \mid f_{k-1} \in \tilde{\mathcal{F}}^{k-1}_\beta \text{and } (s,h) \in \text{dom}(f_{k-1})\}.$$

We then define the optimal conditioning function after $k-1$ passes of relabeling as:

$$f^\star_{k-1}(s,h) := \underset{f \in \mathcal{F}_\beta(s,h)}{\text{argmax}}\ f_{k-1}(s,h),$$

where $f^{\star}_{k-1}(s,h)$ represents the in-distribution optimal RTG from (s,h) in the relabeled feasible set. Given the trajectory set $\tilde{T}^{k-1}_{\beta}$ as well as the correspondingly defined multi-step in-distribution optimal RTG function $f^{\star}_{k-1}(s,a)$, the k-th pass of relabeling proceeds as follows. At the last stage H, set $\tilde{g}^{k}_{H} = \tilde{g}^{k-1}_{H}$. Starting from stage $H-1$, the return-to-go $\tilde{g}^{k-1}_{h}$ is recursively replaced as follows

$$\tilde{g}^{k}_{h} = \max\{r_h + f^{\star}_{k-1}(s_{h+1}, h+1), r_h + \tilde{g}^{k}_{h+1}\}. \tag{B.1}$$

So the trajectory after k passes of relabeling is denoted as $\tilde{\tau}^{k} = (s_1, a_1, \tilde{g}^{k}_{1}, s_2, a_2, \tilde{g}^{k}_{2}, \cdots, s_H, a_H, \tilde{g}^{k}_{H})$, and the accordingly updated feasible set is denoted as $\tilde{T}^{k}_{\beta}$. Finally we define the multi-step R^{2}CSL policy, $\tilde{\pi}^{k,\star}_{\beta}$, corresponding to the k passes relabeling process as

$$\tilde{\pi}^{k,\star}_{\beta} := \tilde{P}^{k}_{\beta}(\cdot|s,h,f^{\star}_{k}(s,h)),$$

where $\tilde{P}^{k}_{\beta}$ is the distribution on $\tilde{T}^{k}_{\beta}$ induced by the behavior policy β and k passes relabeling.

We compare our key relabeling step (B.1) with dynamic programming. At first glance, (B.1) resembles the classical Bellman-type update, where for any state s and action a, the optimal Q-function satisfies:

$$Q^{\star}_{h}(s,a) = r_h(s,a) + \mathbb{E}_{s'} V^{\star}_{h+1}(s'),$$

which involves summing the immediate reward and the expected future return. However, a key distinction is that dynamic programming requires $V^{\star}_{h+1}$ to be the optimal value function, which is not directly obtainable unless we iteratively apply the Bellman equation to $Q^{\star}_{h+1}$ down to the final stage H. In contrast, our approach in (B.1) relies solely on an achievable quantity, $f^{\star}_{k-1}$, which is retained from the $(k-1)$-th relabeling. This distinction makes our relabeling method a natural extension of RCSL towards dynamic programming-based algorithms.

Theoretical Guarantee We have the following theorem that suggests the multi-step relabeling scheme endows the R^{2}CSL with the capability of 'deep stitching': with a sufficient number of relabeling passes, R^{2}CSL utilizing return-to-go relabeling is guaranteed to achieve the optimal policy. To see this, let $\Pi_{\beta} = \{\pi | \forall (s,h) \in \mathcal{S} \times [H], \pi(\cdot|s,h) \ll \beta(\cdot|s,h)\}$[1] be the set of policies that are covered by the behavior policy β, defined the optimal value function covered by β as $V_1^{\star,\beta}(s) = \max_{\pi \in \Pi_{\beta}} V_1^{\pi}(s)$. Then we have the following theorem.

Theorem B.1. Under deterministic environments, we have $J(\tilde{\pi}^{H-1,\star}_{\beta}) = \mathbb{E}_{s\sim\rho} V_1^{\star,\beta}(s)$.

Theorem B.1 establishes that after $k = H-1$ relabeling passes, R^{2}CSL recovers the optimal stitched trajectory, akin to dynamic programming-based methods such as CQL [17]. Moreover, it implies that the relabeling process enhances the worst-case performance of R^{2}CSL. Since R^{2}CSL is guaranteed to recover at least the best k-step stitched trajectory from the initial state, increasing k further strengthens this guarantee.

C Proof of Theorems

In this section, we provide proofs of the theorems in the main text.

C.1 Proof of Theorem 3.1

Proof. By the corollary 2 of [1], we have $J(\pi_f^{\text{RCSL}}) = E_{s\sim\rho}[f(s,1)], \forall f \in \mathcal{F}^{C}_{\beta}$. By the definition of f, we know that $f(s,1)$ is the return-to-go of a trajectory τ starting with $s_1 = s$. Then $E_{s\sim\rho}[f(s,1)]$ is the weighted average of trajectories corresponding to f. In order to show $J(\pi^{\star}) \geq J(\pi_f^{\text{RCSL}})$, we argue in the following that for any $s \in \mathcal{S}$, $V_1^{\pi^{\star}}(s)$ corresponds to the weighted average of return-to-go of a set of stitched trajectories with $s_1 = s$. And the stitched trajectories are no worse than any other trajectory in T_{β} in terms of the cumulative reward (initial return-to-go). Thus we have

$$J(\pi^{\star}) = \mathbb{E}_{s\sim\rho} V_1^{\pi^{\star}}(s) \geq \mathbb{E}_{s\sim\rho} V_1^{\pi_f^{\text{RCSL}}}(s) = J(\pi_f^{\text{RCSL}}), \forall f \in \mathcal{F}^{C}_{\beta}.$$

[1] For two distributions P and Q, $P \ll Q$ means P is absolutely continuous w.r.t. Q.

Our goal is to show starting from any $s \in \mathcal{S}$, any realization of trajectory $\tau^\star = (s, a_1^\star, s_2^\star, a_2^\star, \cdots, s_H^\star, a_H^\star)$ induced by $\pi_\beta^\star$ has cumulative rewards no less than $\tilde{f}(s,1)$. Denote $s_1^\star = s$, then the cumulative reward of $\tau^\star$ is $\sum_{h=1}^H r_h(s_h^\star, a_h^\star)$. We instead prove a more general claim: for any $h \in [H]$, we have $\sum_{t=h}^H r_t(s_t^\star, a_t^\star) \geq f^\star(s_h^\star, h)$, where we recall that

$$f^\star(s_h^\star, h) = \operatorname*{argmax}_{f \in \mathcal{F}_\beta(s_h^\star, h)} f(s_h^\star, h).$$

To facilitate proof, we define a feasible sub-trajectory starting with s_h from step h as $\tau_h(s_h) = (s_h, a_h, g_h, \cdots, s_H, a_H, g_H)$, such that $P_\beta(\tau_h(s_h)) > 0$. Next, we prove the claim by induction.

- At the last step H, we have $r_H(s_H^\star, a_H^\star) = f^\star(s_H^\star, H)$. This is the base case.
- Suppose the claim hold for step $h+1$, i.e., $\sum_{t=h+1}^H r_t(s_t^\star, a_t^\star) \geq f^\star(s_{h+1}^\star, h+1)$, then at step h, we have

$$\begin{aligned}
\sum_{t=h}^H r_t(s_t^\star, a_t^\star) &= r_h(s_h^\star, a_h^\star) + \sum_{t=h+1}^H r_t(s_t^\star, a_t^\star) \\
&\geq r_h(s_h^\star, a_h^\star) + f^\star(s_{h+1}^\star, h+1) \\
&\geq r_h(s_h^\star, a_h^\star) + \{f_h^\star(s_h^\star, h) - r_h(s_h^\star, a_h^\star)\} \\
&= f_h^\star(s_h^\star, a_h^\star),
\end{aligned}$$

  where we use the fact that $f^\star(s_{h+1}^\star, h+1) \geq f_h^\star(s_h^\star, h) - r_h(s_h^\star, a_h^\star)$. This is because $f^\star(s_{h+1}^\star, h+1)$ is the maximum RTG of all feasible sub-trajectories starting with $s_{h+1}^\star$, and $f_h^\star(s_h^\star, h) - r_h(s_h^\star, a_h^\star)$ corresponds the return to go of some particular sub-trajectories starting with $s_{h+1}^\star$. We complete the induction step.

Thus, when $h = 1$, we have $\sum_{h=1}^H r_h(s_h^\star, a_h^\star) \geq f^\star(s,1) = \tilde{f}(s,1)$. We complete the proof.

□

C.2 Proof of Theorem 4.3

Proof. Under the assumption (1) and (2), we known that when the sample size is large enough, the maximum returns will be included in the dataset with high probability, and thus $\hat{f}^\star(s_h, h) = f^\star(s_h, h)$. In particular, we have

$$P\big(s, h, g_h = f^\star(s,h)\big) \geq d_{\min}^\beta \cdot \tilde{c}.$$

Then for any (s_h, h), we want

$$P\big(\forall k \in T_{\mathcal{D}}(s_h), g_h^k \neq f^\star(s_h, h)\big) \leq \big(1 - d_{\min}^\beta \cdot \tilde{c}\big)^N \leq \frac{\delta}{SH},$$

and this leads to

$$N \geq \frac{\log \frac{SH}{\delta}}{\log(1 - d_{\min}^\beta \cdot \tilde{c})}.$$

Then by union bound, when $N > \log(SH/\delta)/\log(1 - d_{\min}^\beta \cdot \tilde{c})$, with probability at least $1-\delta$, the following event holds

$$\mathcal{E} = \{\forall (s_h, h), \exists k \in T_{\mathcal{D}}(s_h),\ s.t.\ g_h^k = f^\star(s_h, h)\}.$$

Under the event $\mathcal{E}$, we have $\hat{f}^\star(s_h, h) = f^\star(s_h, h)$.

Second, we bound the suboptimality the event $\mathcal{E}$. By the definition of the $J(\pi)$, we have

$$J(\pi_\beta^\star) - J(\hat{\pi}_{\mathcal{D}}^\star) = H\Big[\mathbb{E}_P^{\pi_\beta^\star}[r(s,a)] - \mathbb{E}_P^{\hat{\pi}_{\mathcal{D}}^\star}[r(s,a)]\Big] \leq H\big\|d^{\star,\beta} - d^{\star,\mathcal{D}}\big\|_1,$$

where $d^{\star,\beta}$ and $d^{\star,\mathcal{D}}$ are the occupancy measures on state induced by $\pi^\star_\beta$, and $\hat{\pi}^\star_\mathcal{D}$. By Lemma D.1, we have

$$
\begin{aligned}
&J(\pi^\star_\beta) - J(\hat{\pi}^\star_\mathcal{D}) \\
&\leq 2H \sum_{h=1}^{H} \mathbb{E}_{s\sim d^{\star,\beta}_h}\left[\mathrm{TV}\big(\pi^\star_\beta(\cdot|s,h)||\hat{\pi}^\star_\mathcal{D}(\cdot|s,h)\big)\right] \\
&= 2H \sum_{h=1}^{H} \mathbb{E}_{s\sim d^{\star,\beta}_h}\left[\mathrm{TV}\big(P_\beta(\cdot|s,h,f^\star(s,h))||\hat{\pi}(\cdot|s,h,\hat{f}^\star(s,h))\big)\right] \\
&= 2H \sum_{h=1}^{H} \mathbb{E}_{s\sim d^{\star,\beta}_h}\left[\mathrm{TV}\big(P_\beta(\cdot|s,h,f^\star(s,h))||\hat{\pi}(\cdot|s,h,f^\star(s,h))\big)\right] \\
&= 2H \sum_{h=1}^{H} \mathbb{E}_{s\sim d^{\star,\beta}_h}\left[\int_a \left|P_\beta(a|s,h,f^\star(s,h)) - \hat{\pi}(\cdot|s,h,f^\star(s,h))\right|\right] \\
&= 2H \sum_{h=1}^{H} \mathbb{E}_{s\sim d^{\star,\beta}_h}\left[\frac{P_\beta(f^\star(s,h)|s,h)}{P_\beta(f^\star(s,h)|s,h)}\int_a \left|P_\beta(da|s,h,f^\star(s,h)) - \hat{\pi}(da|s,h,f^\star(s,h))\right|\right] \\
&= 2H \sum_{h=1}^{H} \mathbb{E}_{s\sim d^{\star,\beta}_h}\left[\frac{P_\beta(f^\star(s,h)|s,h)}{P_\beta(f^\star(s,h)|s,h)}\int_a \left|P_\beta(da|s,h,f^\star(s,h)) - \hat{\pi}(da|s,h,f^\star(s,h))\right|\right] \\
&\leq \frac{c^\star_\beta H}{\tilde{c}} \sum_{h=1}^{H} \mathbb{E}_{s\sim d^{\beta}_h}\left[P_\beta(f^\star(s,h)|s,h)\int_a \left|P_\beta(da|s,h,f^\star(s,h)) - \hat{\pi}(da|s,h,f^\star(s,h))\right|\right] \\
&\leq \frac{c^\star_\beta H}{\tilde{c}} \sum_{h=1}^{H} \mathbb{E}_{s\sim d^{\beta}_h}\left[\int_g P_\beta(dg|s,h)\int_a \left|P_\beta(da|s,h,g) - \hat{\pi}(da|s,h,g)\right|\right] \\
&= \frac{2c^\star_\beta H}{\tilde{c}} \sum_{h=1}^{H} \mathbb{E}_{s\sim d^{\beta}_h, g\sim P_\beta|s,h}\left[\mathrm{TV}\big(P_\beta(\cdot|s,h,g)||\hat{\pi}(\cdot|s,h,g)\big)\right] \\
&\leq \frac{c^\star_\beta H^2}{\tilde{c}}\sqrt{2L(\hat{\pi})},
\end{aligned}
$$

where the second equation holds under the event $\hat{f}^\star(s_h,h) = f^\star(s_h,h)$, the second inequality holds by assumption (1), and the last step follows from the Pinsker's inequality. Next, for any $(\pi) \in \Pi$, we write $L(\pi) = \bar{L}(\pi) - H_\beta$, where $H_\beta = -\mathbb{E}[\log P_\beta(a|s,h,g)]$ and $\bar{L}(\pi) = -\mathbb{E}[\log \pi(a|s,h,g)]$. Denoting $\pi^\dagger \in \mathrm{argmin}_{\pi\in\Pi} L(\pi)$, we have

$$
L(\hat{\pi}) = L(\hat{\pi}) - L(\pi^\dagger) + L(\pi^\dagger) \leq \bar{L}(\hat{\pi}) - \bar{L}(\pi^\dagger) + \delta_{\text{approx}}.
$$

Denote $\hat{L}$ as the empirical cross-entropy loss that is minimized by $\hat{\pi}$, we have

$$
\begin{aligned}
\bar{L}(\hat{\pi}) - \bar{L}(\pi^\dagger) &= \bar{L}(\hat{\pi}) - \hat{L}(\hat{\pi}) + \hat{L}(\hat{\pi}) - \hat{L}(\pi^\dagger) + \hat{L}(\pi^\dagger) - \bar{L}(\pi^\dagger) \\
&\leq 2\sup_{\pi\in\Pi}|\bar{L}(\pi) - \hat{L}(\pi)|.
\end{aligned}
$$

Under Assumption 4.1, we bound this using McDiarmid's inequality and a union bound. This completes the proof. □

C.3 Proof of Theorem 4.5

Proof. By the definition of the $J(\pi)$, we have

$$
J(\pi^\star_\beta) - J(\hat{\pi}^\star_\mathcal{D}) = H\left[\mathbb{E}_P^{\pi^\star_\beta}[r(s,a)] - \mathbb{E}_P^{\hat{\pi}^\star_\mathcal{D}}[r(s,a)]\right] \leq H\left\|d^{\star,\beta} - d^{\star,\mathcal{D}}\right\|_1,
$$

where $d^{\star,\beta}$ and $d^{\star,\mathcal{D}}$ are the occupancy measures on state induced by $\pi^\star_\beta$, and $\hat{\pi}^\star_\mathcal{D}$. By Lemma D.1, we have

$$
J(\pi^\star_\beta) - J(\hat{\pi}^\star_\mathcal{D})
$$

$$
\begin{aligned}
&\leq 2H \sum_{h=1}^{H} \mathbb{E}_{s\sim d_h^{\star,\beta}} \left[\mathrm{TV}\big(\pi_\beta^\star(\cdot|s,h)||\hat{\pi}_{\mathcal{D}}^\star(\cdot|s,h)\big)\right] \\
&\leq 2c_\beta^\star H \sum_{h=1}^{H} \mathbb{E}_{s\sim d_h^{\beta}} \left[\mathrm{TV}\big(P_\beta(\cdot|s,h,f^\star(s,h))||\hat{\pi}(\cdot|s,h,\hat{f}^\star(s,h))\big)\right] \\
&\leq 2c_\beta^\star H \sum_{h=1}^{H} \mathbb{E}_{s\sim d_h^{\beta}} \Big[\mathrm{TV}\big(P_\beta(\cdot|s,h,f^\star(s,h))||\hat{\pi}(\cdot|s,h,f^\star(s,h))\big) \\
&\qquad + \mathrm{TV}\big(\hat{\pi}(\cdot|s,h,f^\star(s,h))||\hat{\pi}(\cdot|s,h,\hat{f}^\star(s,h))\big)\Big] \\
&\leq 2c_\beta^\star H \sum_{h=1}^{H} \mathbb{E}_{s\sim d_h^{\beta}} \left[\mathrm{TV}\big(P_\beta(\cdot|s,h,f^\star(s,h))||\hat{\pi}(\cdot|s,h,f^\star(s,h))\big) + \gamma\big|f^\star(s,h) - \hat{f}^\star(s,h)\big|\right] \\
&\leq 2c_\beta^\star H \sum_{h=1}^{H} \mathbb{E}_{s\sim d_h^{\beta}} \left[\mathrm{TV}\big(P_\beta(\cdot|s,h,f^\star(s,h))||\hat{\pi}(\cdot|s,h,f^\star(s,h))\big)\right] \\
&\qquad + 2c_\beta^\star H \sum_{h=1}^{H} \gamma \sqrt{\mathbb{E}_{s\sim d_h^{\beta}}\big(f^\star(s,h) - \hat{f}^\star(s,h)\big)^2} \\
&\leq 2c_\beta^\star H \sum_{h=1}^{H} \mathbb{E}_{s\sim d_h^{\beta}} \left[\mathrm{TV}\big(P_\beta(\cdot|s,h,f^\star(s,h))||\hat{\pi}(\cdot|s,h,f^\star(s,h))\big)\right] + 2c_\beta^\star H^2 \gamma \sqrt{\mathrm{Err}(N,\delta)} \\
&\leq 2c_\beta^\star H \sum_{h=1}^{H} \mathbb{E}_{s\sim d_h^{\beta}} \Big[\int_a \big|P_\beta(a|s,h,f^\star(s,h)) - \hat{\pi}(\cdot|s,h,f^\star(s,h))\big|\Big] + 2c_\beta^\star H^2 \gamma \sqrt{\mathrm{Err}(N,\delta)} \\
&= 2c_\beta^\star H \sum_{h=1}^{H} \mathbb{E}_{s\sim d_h^{\beta}} \Big[\frac{P_\beta(f^\star(s,h)|s,h)}{P_\beta(f^\star(s,h)|s,h)} \int_a \big|P_\beta(da|s,h,f^\star(s,h)) - \hat{\pi}(da|s,h,f^\star(s,h))\big|\Big] \\
&\qquad + 2c_\beta^\star H^2 \gamma \sqrt{\mathrm{Err}(N,\delta)} \\
&\leq \frac{c_\beta^\star H}{\tilde{c}} \sum_{h=1}^{H} \mathbb{E}_{s\sim d_h^{\beta}} \Big[P_\beta(f^\star(s,h)|s,h) \int_a \big|P_\beta(da|s,h,f^\star(s,h)) - \hat{\pi}(da|s,h,f^\star(s,h))\big|\Big] \\
&\qquad + 2c_\beta^\star H^2 \gamma \sqrt{\mathrm{Err}(N,\delta)} \\
&\leq \frac{c_\beta^\star H}{\tilde{c}} \sum_{h=1}^{H} \mathbb{E}_{s\sim d_h^{\beta}} \Big[\int_g P_\beta(dg|s,h) \int_a \big|P_\beta(da|s,h,g) - \hat{\pi}(da|s,h,g)\big|\Big] + 2c_\beta^\star H^2 \gamma \sqrt{\mathrm{Err}(N,\delta)} \\
&= \frac{2c_\beta^\star H}{\tilde{c}} \sum_{h=1}^{H} \mathbb{E}_{s\sim d_h^{\beta}, g\sim P_\beta|s,h} \left[\mathrm{TV}\big(P_\beta(\cdot|s,h,g)||\hat{\pi}(\cdot|s,h,g)\big)\right] + 2c_\beta^\star H^2 \gamma \sqrt{\mathrm{Err}(N,\delta)} \\
&\leq \frac{c_\beta^\star H^2}{\tilde{c}} \sqrt{2L(\hat{\pi})} + 2c_\beta^\star H^2 \gamma \sqrt{\mathrm{Err}(N,\delta)},
\end{aligned}
$$

where the last step follows from the Pinsker's inequality. Next, for any $(\pi) \in \Pi$, we write $L(\pi) = \bar{L}(\pi) - H_\beta$, where $H_\beta = -\mathbb{E}[\log P_\beta(a|s,h,g)]$ and $\bar{L}(\pi) = -\mathbb{E}[\log \pi(a|s,h,g)]$. Denoting $\pi^\dagger \in \mathrm{argmin}_{\pi\in\Pi} L(\pi)$, we have

$$
L(\hat{\pi}) = L(\hat{\pi}) - L(\pi^\dagger) + L(\pi^\dagger) \leq \bar{L}(\hat{\pi}) - \bar{L}(\pi^\dagger) + \delta_{\text{approx}}.
$$

Denote $\hat{L}$ as the empirical cross-entropy loss that is minimized by $\hat{\pi}$, we have

$$
\begin{aligned}
\bar{L}(\hat{\pi}) - \bar{L}(\pi^\dagger) &= \bar{L}(\hat{\pi}) - \hat{L}(\hat{\pi}) + \hat{L}(\hat{\pi}) - \hat{L}(\pi^\dagger) + \hat{L}(\pi^\dagger) - \bar{L}(\pi^\dagger) \\
&\leq 2 \sup_{\pi\in\Pi} |\bar{L}(\pi) - \hat{L}(\pi)|.
\end{aligned}
$$

Under Assumption 4.1, we bound this using McDiarmid's inequality and a union bound. This completes the proof. □

C.4 Proof of Hard Instances for R^{2}CSL with Expectile Regression

Proof. We use the following toy example:

	$h=1$	$h=2$	$h=3$
a^1	70(80)	0(10)	10(10)
a^2	65(81)	15(16)	1(1)
a^3	40(75)	20(35)	15(15)

In this toy example, $\mathcal{A} = \{a^1, a^2, a^3\}$, $H = 3$ and $\mathcal{S} = \{s\}$. The state is unique and remains unchanged across stages. At each stage, there are three actions that can be chosen. Each row represents a trajectory induced by implementing the action listed at the beginning. The number outside (inside) the parentheses are rewards (return-to-go). For RCSL, we can only simply choose f to be 80, 81 or 75 as the conditioning return at the initial stage, and the 'optimal' RCSL policy would choose a^2 at each stages. Although there are only three suboptimal trajectories, these suboptimal trajectories collectively cover a better trajectory. Let's see: at the first stage we choose a^2, at the second stage we choose a^3 and at the final stage we choose a^3. This leads to a trajectory

$$a^2 : 65(100) \overset{h=1}{\rightarrow} a^3 : 20(35) \overset{h=2}{\rightarrow} a^3 : 15(15) \overset{h=3}{}.$$

This policy can ideally be inferred by the reinforced RCSL: during the inference process, at each stage, we set the conditioning function as the largest return-to-go corresponding to that stage. Specifically, at the first stage, the largest return-to-go, 81, comes from the second trajectory, thus we set $f_1 = 81$ and $\pi^{\text{RCSL}}(f_1, h = 1) = a^2$; at the second stage, the largest return-to-go, 35, comes from the third trajectory, thus we set $f_2 = 35$ and $\pi^{\text{RCSL}}(f_2, h = 2) = a^3$; at the third stage, the largest return-to-go, 15, comes from the third trajectory, thus we set $f_3 = 15$ and $\pi^{\text{RCSL}}(f_3, h = 3) = a^3$. We highlight that in this toy example, the trajectories implicitly have overlap since the underlying state is unique and remains unchanged, and the dependence of the RCSL policy on the state is also omitted.

Consider policy learning, we set the behavior policy as the uniform distribution on the action space, $\beta = \text{Uniform}(\mathcal{A})$. Note that in the tabular MDP, the expectile regression is conducted at each state-action pair. Next, let's focus on the second stage. The return-to-go candidates are $\{10, 16, 35\}$, and our goal is to find the in-distribution optimal RTG, which is 35 at stage $h = 2$ coming from the third trajectory

$$a^3 : 40(75) \overset{h=1}{\rightarrow} a^3 : 20(35) \overset{h=2}{\rightarrow} a^3 : 15(15) \overset{h=3}{}. \tag{C.1}$$

However, due to the fact that the expectile regression uses L_2 loss, as long as the offline dataset includes a trajectory other than (C.1), the expectile regression with $\alpha < 1$ would return a value less than 35 and may not being among the candidates $\{10, 16, 35\}$ (and thus being out-of-distribution). Then Algorithm 1 with expectile regression fails at the second stage. On the other hand, if all trajectories in the offline dataset is the trajectory (C.1), then the offline dataset does not contain any information about the second trajectory, which is a necessary component of the optimal stitched trajectory. In this case, Algorithm 1 with expectile regression will fail in the first stage. In conclusion, Algorithm 1 with expectile regression will definitely fail in the first or the second stage. This completes the proof. □

C.5 Proof of Theorem 5.2

Proof. We only need to show that with probability at least $1 - 2\delta$, for any (s, h), we have $P(X_h(f^\star(s, h)) \geq N_h^2 \cdot \tilde{c}/2)$. Then setting $\alpha > 1 - \tilde{c}/2$, the α-quantile is exactly $f^\star(s, h)$.

By the Hoeffding inequality and the assumption that $P_\beta(g_h = f^\star(s, h)|s_h = s) \geq \tilde{c}$, we have

$$P(X_h(f^\star(s, h)) \leq N_h^s \cdot \tilde{c} - t) \leq P(X_h(f^\star(s, h)) \leq P_\beta(g_h = g|s_h = s) - t) \leq \exp(-2t^2/N_h^s).$$

Let $t = \tilde{c}N_h^s/2$, we have

$$P(X_h(f^\star(s, h)) \leq N_h^s \cdot \tilde{c}/2) \leq \exp(-\tilde{c}^2 N_h^s/2) \leq \frac{\delta}{2SH}.$$

This leads to $N_h^s \geq 2\log(2SH/\delta)/\tilde{c}^2$. Note that N_h^s is also a random variable, next we study the condition under which $N_h^s \geq 2\log(2SH/\delta)/\tilde{c}^2$ holds with high probability. In particular, we have

$$P(N_h^s \leq N \cdot d_{\min}^{\beta} - t) \leq P(N_h^s \leq N \cdot d_h^{\beta}(s) - t) \leq \exp(-2t^2/N).$$

Let $t = N \cdot d_{\min}^{\beta}/2$ and make the above inequality be less than $\delta/2SH$, we derive that when

$$N \geq \frac{2}{d_{\min}^{\beta,2}} \log \frac{2SH}{\delta},$$

we have

$$P(N_h^s \geq N \cdot d_{\min}^{\beta}/2) \geq 1 - \delta/2.$$

Moreover, we want $N \cdot d_{\min}^{\beta}/2 \geq 2\log(2SH/\delta)/\tilde{c}^2$, which leads to

$$N \geq \frac{4}{\tilde{c}^2 d_{\min}^{\beta}} \log \frac{2SH}{\delta},$$

thus we have

$$P\Big(N_h^s \geq 2\log(2SH/\delta)/\tilde{c}^2\Big) \geq P\Big(N_h^s \geq \frac{N \cdot d_{\min}^{\beta}}{2}\Big) \geq 1 - \frac{\delta}{2}.$$

By union bound, when

$$N \geq \max\Big\{\frac{2}{d_{\min}^{\beta,2}} \log \frac{2SH}{\delta}, \frac{4}{\tilde{c}^2 d_{\min}^{\beta}} \log \frac{2SH}{\delta}\Big\},$$

with probability at least $1 - \delta$, we have $P(X_h(f^\star(s,h)) \geq N_h^2 \cdot \tilde{c}/2)$. This completes the proof. □

C.6 Proof of Theorem B.1

Proof. Define the optimal Q-function covered by the behavior policy β as $Q_h^{\star,\beta}(s,a) = \max_{\pi \in \Pi_\beta} Q_h^\pi(s,a)$. By bellman optimality equation, we know that $V_h^{\star,\beta}(s) = \max_{a \in \mathcal{A}, \beta(a|s)>0} Q_h^{\star,\beta}(s,a)$. We prove Theorem B.1 by showing that the new label $\tilde{g}_h^{H-1}$ is $Q_h^{\star,\beta}(s_h,a_h)$, and the conditioning function $\tilde{f}_k^\star$ serves as the maximum operator in defining the value function.

To show this, let's first answer the following easier question: After one-pass of the relabeling procedure, what does the new RTG in the trajectory mean? Equivalently, what does the return-conditioned policy based on modified data aim for? With the original trajectory $\tau = (s_1, a_1, g_1, \cdots, s_H, a_H, g_H)$, next we delve into the relabeling process. Starting from the last stage H, we have $\tilde{g}_H^1 = g_H$, which is the reward obtained by adopting a_H at s_H. Then the trajectory becomes $\tau = (s_1, a_1, g_1, \cdots, s_H, a_H, \tilde{g}_H^1)$. Moving one stage backward, we perform some real relabeling at stage $H-1$.

$$\tilde{g}_{H-1} = \max\{\underbrace{r_{H-1} + f^\star(s_H, H)}_{\text{I}}, \underbrace{r_{H-1} + \tilde{g}_H^1}_{\text{II}}\}.$$

From now on, at each stage, we answer the following two questions to find common patterns. Q1: what are the term I and term II. Q2: what does the maximization operation mean?

We first focus on Q1. The relabeling consists of two parts. Term I: the current reward + the maximum in-distribution RTG in the original data, and term II: the original RTG g_{H-1}. In both term I and term II, there are two parts: the current reward and the future cumulative reward. In term I, the future cumulative reward is the maximum possible RTG at s_H achieved by β. So term I represents the cumulative reward we can get at s_{H-1} if we take a_{H-1} at the current stage and then take the action associated to $f^\star(s_H, H)$. Term II represents the cumulative reward we can get at s_{H-1} if we take a_{H-1} at the current stage and then take original action in the trajectory τ, which is a_H. This answers Q1. We then focus on Q2. Note that at stage $H-1$, there are two cases can happen: (i) term I = term II, and (ii) term I > term II. For case (i), the tie means that a_H is the action that achieves the maximum RTG, which is the maximum reward at s_H. For case

(ii), term I $>$ term II means that there is a better action, which is associated with $f^\star(s_H, H)$, and we better follow that action in at s_H. Thus, it is clear that at stage $H-1$, if case (ii) happens, we actually would perform *one-step stitching* (because the action associated with $f^\star(s_H, H)$ differs from a_H, in order words it comes from other trajectories instead of τ itself), and the stitching technically happens in the next stage H. This answers Q2. After relabeling g_{H-1}, we have $\tau = \{s_1, a_1, g_1, \cdots, s_{H-2}, a_{H-2}, g_{H-2}, s_{H-1}, a_{H-1}, \tilde{g}^1_{H-1}, s_H, a_H, \tilde{g}^1_H\}$. Before we move on, let's stop for a while to consider a question: What does $\tilde{g}_{H-1}$ represent? Based on the answer to Q1 and Q2 above, we can tell that $\tilde{g}_{H-1}$ represents the return-to-go we can get if we follow the action in the trajectory a_{H-1} at s_{H-1} and then take the optimal action supported by β at s_H, i.e., the optimal RTG we can get after following a_{H-1}.

One stage backward, let's consider the stage $H-2$

$$\tilde{g}_{H-2} = \max\{\underbrace{r_{H-2} + f^\star(s_{H-1}, H-1)}_{\text{I}}, \underbrace{r_{H-2} + \tilde{g}^1_{H-1}}_{\text{II}}\}.$$

Term I is the current reward plus maximum RTG at s_{H-1} achieved by β. Term II is the current reward plus optimal RTG after following a_{H-1} at s_{H-1}. This answers Q1. For Q2, the comparison between term I and term II is to check if there is a better action for stage $H-1$ at s_{H-1}. To see this, we analyze the following three possible cases: (i) term I $=$ term II, (ii) term I $>$ term II, and (iii) term I $<$ term II. For case (i), tie means that following action a_{H-1} at s_{H-1} would be fine; for case (ii), there is a better choice of action than a_{H-1} which leads to a larger cumulative reward (maximum in-distribution RTG at s_{H-1}); for case (iii), we better follow a_{H-1} at s_{H-1} because after checking the RTGs of all possible trajectories with s_{H-1} induced by β, no one is larger than $\tilde{g}_{H-1}$, which is the cumulative reward of a sub-trajectory starting with (s_{H-1}, a_{H-1}) and possibly involving one-time stitching at s_H. To sum up, the maximization operation actually gives us a chance to figure out if the feasible set suggests a better choice of action at s_{H-1}, which could lead to a new trajectory. Note that the term 'better' is in the sense that it leads to a in-distribution RTG that is larger than the cumulative reward of a sub-trajectory starting with (s_{H-1}, a_{H-1}) and possibly involving one-time stitching at s_H. We highlight that the in-distribution RTG is the RTG corresponding to a sequence of actions in some original trajectory involved in T_β, which does not involve stitching. If (ii) happens, choosing term I as the relabeled RTG means that we perform one-step stitching at s_{H-1} by adopting an action other than the original action a_{H-1} in the trajectory, and forget about the sub-trajectory starting with (s_{H-1}, a_{H-1}) and possibly involving one-time stitching at s_H. It will become clear later that the relabeling process is effectively performing a form of shallow stitching/planning, as the new label only remembers an one-time stitching.

Next, we focus on $\tilde{g}^1_{H-3}$, and we will answer the question we raised at the very beginning by providing an exact meaning of $\tilde{g}^1_{H-3}$.

$$\tilde{g}_{H-3} = \max\{\underbrace{r_{H-3} + f^\star(s_{H-2}, a_{H-2})}_{\text{I}}, \underbrace{r_{H-3} + \tilde{g}^1_{H-2}}_{\text{II}}\}.$$

At s_{H-3}, we follow a_{H-3}. Then at the next stage $H-2$ we either follow the action associated to $f^\star(s_{H-2}, H-2)$, or follow the original action a_{H-2}, which could provide a better future stitching. So choosing term I/term II basically means stitching at the next stage or stitching maybe in the further future (after the next stage). Thus, the new label $\tilde{g}_h$ is the cumulative reward achieved by following a_h at the current stage and then follow the best one-time stitching trajectory afterwards. One pass of the relabeling process incorporates information about the future best one-time stitching into the new RTG label. Starting from $\tilde{\tau} = (s_1, a_1, \tilde{g}^1_1, \cdots, s_H, a_H, \tilde{g}^1_H)$, the second pass of the relabeling process endows the label $\tilde{g}^2_h$ information about the future best two-time stitching trajectory. This is exactly the dynamic programming in deterministic environments, and after $H-1$ passes, we have $\tilde{g}^{H-1}_h = Q^{\star,\beta}_h(s_h, a_h)$.

Lastly, the conditioning function $\tilde{f}^\star_k$ is defined based on $\tilde{\tau}^{H-1}$. By definition, $\tilde{f}^\star_k$ is the maximum in-distribution RTG label

$$\tilde{f}^\star_k(s_h, h) = \max_{a \in \mathcal{A}, \beta(a|s_h) > 0} \tilde{g}^{H-1}_h(s_h, a) = \max_{a \in \mathcal{A}, \beta(a|s_h) > 0} Q^{\star,\beta}_h(s_h, a_h),$$

which identifies the best action that achieves the optimal value function at s_h. Thus, the reinforced RCSL policy $\tilde{\pi}^{H-1,\star}_\beta$ recovers the optimal policy.

□

D The Auxiliary Lemmas

Lemma D.1 (Lemma 1 of [1])**.** Let d^π refer to the marginal distribution of P^π over states only. For any two policies π and π', we have

$$\|d^\pi - d^{\pi'}\|_1 \leq 2\sum_{h=1}^{H} \mathbb{E}_{s\sim d_h^\pi}\left[\mathrm{TV}\big(\pi(\cdot|s,h)||\pi'(\cdot|s,h)\big)\right].$$

Proof. The proof follows the proof of Lemma 1 in [1]. Except that

$$\|d^\pi - d^{\pi'}\|_1 \leq \frac{1}{H}\sum_{h=1}^{H}\sigma_h \leq \frac{1}{H}\sum_{h=1}^{H}\sum_{j=1}^{h-1}\sigma_j \leq H\frac{1}{H}\sum_{h=1}^{H}\sigma_h = 2\sum_{h=1}^{H}\mathbb{E}_{s\sim d_h^\pi}\left[\mathrm{TV}\big(\pi(\cdot|s,h)||\pi'(\cdot|s,h)\big)\right].$$

□

E Experiments Details

D4RL [5] is an offline RL benchmark which provide the pre-collected dataset such as Gym-MuJoCo, AntMaze, and Kitchen. In this work, we evaluate our work in the Gym-MuJoCo with the medium, medium-replay, and medium-expert three-level results. For AntMaze, we evaluate our methods on umaze, umaze-diverse, medium-play and medium diverse. All experiments are based on five seeds: 1000, 2000, 3000, 4000 and 5000.

E.1 Implementation and Hyperparameters

In this section, we provide the implementation details and hyperparameters of our R^{2}CSL-Expectile and R^{2}CSL-Quantile.

Implementation of R^{2}CSL with RvS To validate our proposed algorithm, we follow the settings in [3]. RvS leverages the vanilla RTG as the conditioning function to predict the optimal action. We keep the training stage as the same as RvS and modify the inference stage. During the inference, we first pre-train a condition function by expectile regression and quantile regression using MLP. Then we use the pre-trained condition function to predict the max-return given the current state in the inference. We use the RTG prediction to guide the action selection in RvS.

We evaluate our proposed R^{2}CSL method using the RvS framework on the D4RL benchmark [5]. In the RvS framework, achieving optimal performance during inference requires searching for the target RTGs [3]. However, this process is often impractical in real-world scenarios. To address this, we select three appropriate target RTG fraction ratios as $0.7, 0.9, 1.1$ to guide the RvS instead, which means the initial target RTG will be $\mathrm{RTG}_{\mathrm{inital}} = (\mathrm{RTG}_{\max} - \mathrm{RTG}_{\min}) * \mathrm{fraction} + \mathrm{RTG}_{\min}$, where $\mathrm{RTG}_{\max}$ and $\mathrm{RTG}_{\min}$ are the maximum RTG and minimum RTG from the random policy and expert policy in this specific environment respectively. Unlike the RvS approach, our method does not require additional searches for initial target RTGs, as these values are predicted by our pre-trained condition function.

We follow the same implementation as the RvS during the training stage shown in Table 4. We also illustrate our MLP condition function parameters in Table 5.

We run these experiments on the RTX2080Ti for around 8 hours for each setting.

Implementation of DT-R^{2}CSL We also evaluate our proposed R^{2}CSL method using the DT [2] on the D4RL benchmark [5]. DT utilize the transformer model to learn the action given the past trajectory and the target RTG. For the implementation, we follow the recently proposed Reinformer [40] utilizing additional action entropy to regularize to have a more stable DT training. We formulate the loss function as $L_{DT-\mathrm{R^2CSL}} = \mathbb{E}_\tau[-\log \pi_\theta(\cdot|\tau) - \lambda H(\pi_\theta(\cdot|\tau))]$. In the vanilla DT, we also need to give the target RTG during the inference stage. However, similar to the implementation on the RVS framework, our method also does not require additional target RTGs to achieve good performance.

We use the same MLP condition function hyperparameters in Table 5. And we provide the hyperparameters of our DT-R^{2}CSL in Table 6.

Table 4: Hyperparameters in RvS.

Hyperparameter	Value
Hidden layers	2
Layer width	1024
Nonlinearity	ReLU
Learning rate	1×10^{-3}
Epochs	2000
Batch size	16384
Dropout	0
Policy output	Unimodal Gaussian

Table 5: Hyperparameters in Condion Function.

Hyperparameter	Value
Hidden layers	3
Layer width	128
Nonlinearity	ReLU
Learning rate	1×10^{-4}
Epochs	300
Batch size	256
Dropout	0.1

We run these experiments on the A5000 for around 3 hours for each setting.

Table 6: Hyperparameters in DT-R^{2}CSL.

Hyperparameter	Value
Number of layers	3
Number of attention heads	1
Embedding dimension	128
Nonlinearity function	ReLU
Batch size	64
Context length K	20
Dropout	0.1
Learning rate	1×10^{-4}
Grad norm clip	0.25
Weight decay	1×10^{-4}
Action Entropy λ	0.1

Implementation of DP-R^{2}CSL We also extend our empirical study into the RCSL with the dynamic programming component. We evaluate our proposed DP-R^{2}CSL method using the QT framework [10] on the D4RL benchmark [5]. QT incorporates the Q value function learning in the training stage and utilizes this function to guide the action selection in the inference stage. An MLP-based conditioning function is used to model the distribution of RTG values conditioned on the input state. Initial target RTGs are then sampled from this distribution and combined with several given target RTGs during the evaluation process. At each inference step, the action is generated given the predicted RTG and then selected by the critic component which is the Q value function.

The hyperparameters for the MLP-based QT model are provided in Table 7, and the training hyperparameters for DP-R^{2}CSL are summarized in Table 8.

We run these DP-R^{2}CSL experiments on the A5000 for around 15 hours for each setting.

Table 7: Hyperparameters in Condion Function in DP-R^{2}CSL.

Hyperparameter	Value
Hidden layers	3
Layer width	256
Nonlinearity	ReLU
Learning rate	1×10^{-4}
Epochs	300
Batch size	256
Dropout	0.1

Table 8: Hyperparameters in DP-R^{2}CSL.

Hyperparameter	Value
Number of layers	4
Number of attention heads	4
Embedding dimension	256
Nonlinearity function	ReLU
Batch size	256
Context length K	20
Dropout	0.1
Learning rate	3.0×10^{-4}

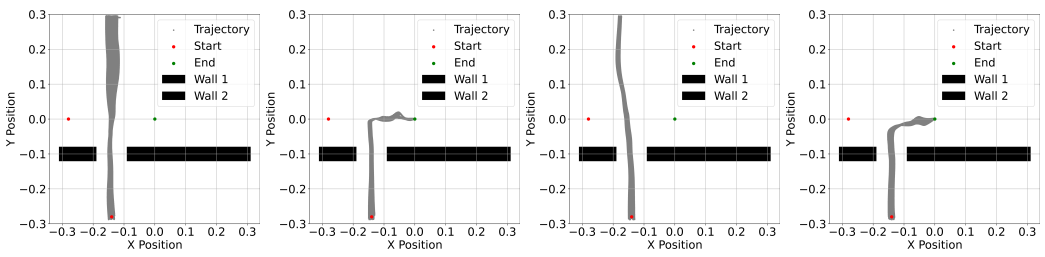
E.2 Additional Experiment results

In this section, we present the experiment results on AntMaze.

R^{2}CSL with RvS. We implement Algorithm 1 with expectile regression and quantile regression based on the RvS. Experiment results are shown in Table 9. We can conclude that R^{2}CSL outperforms the RvS framework across all fixed target RTG fraction ratios.

R^{2}CSL with DT. Experiment results of the DT-extension of our methods, DT-R^{2}CSL, are shown in Table 10. We conclude that DT-R^{2}CSL outperforms vanilla DT across all settings.

R^{2}CSL incorporating dyamic programming. Experiment results of DP-R^{2}CSL, which is a hybrid method incorporating dynamic programming components, are shown in Table 11. We conclude that DP-R^{2}CSL significantly improves DT-R^{2}CSL. DP-R^{2}CSL outperforms QT on the umaze environment, but is slightly worse on umaze-diverse and medium diverse environments. This is well expected as QT shares a key feature with our reinforced RCSL framework—namely, the use of an optimal conditioning function.



(a) Expectile ($\alpha = 0.9$) (b) Expectile ($\alpha = 0.99$) (c) Quantile ($\alpha = 0.9$) (d) Quantile ($\alpha = 0.99$)

Figure 4: Illustration of Stitching - Proportion of type I trajectories in the offline dataset is set to 0.01. R^{2}CSL with $\alpha = 0.85$ fails to stitch, while with $\alpha = 0.95$ succeeds to stitch.

Table 9: Normalized score on D4RL Gym for RvS and R^2CSL with expectile ($\alpha = 0.99$) and quantile ($\alpha = 0.99$) regression respectively. The inference of RvS is conditioned on three target RTGs. We report the mean and standard deviation of the normalized score for five seeds.

Dataset	**RVS**			**R^2CSL-Expectile**	**R^2CSL-Quantile**
	0.7	0.9	1.1		
umaze	54.5±5.83	57.2±8.64	60.2±10.26	61.4±5.97	64.8±3.82
umaze-diverse	54.6±6.71	56.5±4.65	53.7±3.95	57.9±2.99	57.8±3.19
medium-play	0.1±0.27	0.1±0.22	0.1±0.22	0.3±0.27	0.4±0.42
medium-diverse	0.2±0.27	0.2±0.27	0.2±0.27	0.2±0.27	0.5±0.35
Total	109.4	114.0	114.2	119.8	**123.5**

Table 10: Normalized score on D4RL Antmaze for DT, QT, DT-R^2CSL and DP-R^2CSL with expectile (α = 0.99) and quantile (α = 0.99) regression respectively. We report the mean and standard deviation of the normalized score for five seeds.

Dataset	**DT-R^2CSL-Expectile**	**DT-R^2CSL-Quantile**	**DT**
umaze	72.8±3.90	71.4±3.51	59.2
umaze-diverse	63.2±6.06	62.4±4.10	53
medium-play	2.6±1.14	1.4±0.55	0
medium-diverse	2.4±1.52	3.4±1.14	0
Total	**141.0**	138.6	112.2

Table 11: Normalized score on D4RL Antmaze for QT, DP-R^2CSL with expectile (α = 0.99) and quantile (α = 0.99) regression respectively. We report the mean and standard deviation of the normalized score for five seeds.

Dataset	**DP-R^2CSL-Expectile**	**DP-R^2CSL-Quantile**	**QT**
umaze	97.8±3.03	97.4±2.88	96.7±4.7
umaze-diverse	84.6±5.08	88.2±8.14	96.7±4.7
medium-diverse	50.2±7.08	48.4±4.72	59.3±0.9
Total	232.6	234	**252.7**