Q-value Regularized Transformer for Offline Reinforcement Learning

Shengchao Hu [1 2] **Ziqing Fan** [1 2] **Chaoqin Huang** [1 2] **Li Shen** [3 4] **Ya Zhang** [1 2] **Yanfeng Wang** [1 2] **Dacheng Tao** [5]

Abstract

Recent advancements in offline reinforcement learning (RL) have underscored the capabilities of Conditional Sequence Modeling (CSM), a paradigm that learns the action distribution based on history trajectory and target returns for each state. However, these methods often struggle with stitching together optimal trajectories from sub-optimal ones due to the inconsistency between the sampled returns within individual trajectories and the optimal returns across multiple trajectories. Fortunately, Dynamic Programming (DP) methods offer a solution by leveraging a value function to approximate optimal future returns for each state, while these techniques are prone to unstable learning behaviors, particularly in long-horizon and sparse-reward scenarios. Building upon these insights, we propose the Q-value regularized Transformer (QT), which combines the trajectory modeling ability of the Transformer with the predictability of optimal future returns from DP methods. QT learns an action-value function and integrates a term maximizing action-values into the training loss of CSM, which aims to seek optimal actions that align closely with the behavior policy. Empirical evaluations on D4RL benchmark datasets demonstrate the superiority of QT over traditional DP and CSM methods, highlighting the potential of QT to enhance the state-of-the-art in offline RL.

1. Introduction

Offline reinforcement learning (RL) aims at learning effective policies entirely from previously collected data without interacting with the environment (Fujimoto et al., 2019b). Recent advancements in offline RL have taken a new perspective on the problem, departing from conventional methods for offline RL that concentrate on policy regularization (Kumar et al., 2019a; Fujimoto et al., 2019b) or conservatism for value function approximation (Kostrikov et al., 2021a; Kumar et al., 2020). Instead, the problem is viewed as a generic Conditional Sequence Modeling (CSM) task (Chen et al., 2021; Janner et al., 2021), where past experiences consisting of state-action-reward triplets are input to Transformer (Vaswani et al., 2017a). The model generates a sequence of action predictions using a goal-conditioned policy, effectively converting offline RL to a supervised learning problem. This approach relaxes the MDP assumption by considering multiple historical steps to predict an action, allowing the model to be capable of handling long sequences and avoid stability issues associated with bootstrapping (Srivastava et al., 2019; Kumar et al., 2019b).

However, the CSM approach fails to achieve the stitching property desired in offline RL, which involves synthesizing optimal trajectories from sub-optimal ones (Fu et al., 2020). The primary challenge lies in the inconsistency between sampled target returns and the optimal returns from actions, as high-return trajectories might not reflect superior actions but rather fortunate circumstances (Wang et al., 2023). CSM associates the return-to-go (RTG) token value with individual trajectories, overlooking the stochastic nature of state transitions and optimal future returns that span across different trajectories (Paster et al., 2022). Additionally, the intrinsic uncertainty and approximation errors in behavior policies further exacerbate the inconsistency, leading to inferior performance in stitching tasks, particularly when dealing with sub-optimal data (Wang et al., 2023).

Fortunately, conventional Dynamic Programming methods (Q-learning[1]) provide a robust solution to handle this inconsistency. By treating each timestep individually and backpropagating optimal future returns for each state, these methods enable agents to select actions that maximize long-term returns. However, these techniques are prone to unstable learning behaviors, particularly in long-horizon and sparse-reward scenarios (Yamagata et al., 2023). While the

[1]Shanghai Jiao Tong University, China [2]Shanghai AI Laboratory, China [3]Sun Yat-sen University, China [4]JD Explore Academy, China [5]Nanyang Technological University, Singapore. Correspondence to: Li Shen <mathshenli@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1]In this paper, the terms *Q-learning* and *Dynamic Programming (DP)* will be used interchangeably to refer to any RL algorithm that relies on the Bellman-backup operation.

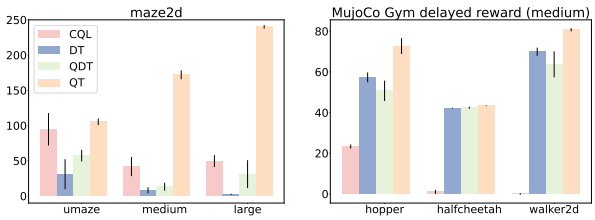
conceptual integration of Q-learning with CSM is straightforward, developing a framework that effectively unites their strengths and overcomes their limitations poses a significant challenge. QDT (Yamagata et al., 2023) takes the first attempt to combine these two methods by learning a conservative value function to relabel the RTG values while remaining other components the same as DT (Chen et al., 2021). This approach seeks to enhance stitching capability by incorporating augmented trajectories into the training dataset. However, empirical evaluations suggest that while it may alleviate some issues, it still struggles with unmatched RTG values during inference arising from trajectory-level modeling (Wang et al., 2023), often achieving results comparable to but not exceeding existing methods (Figure 1).

Building upon these insights, we propose the Q-value regularized Transformer (QT), which combines the trajectory modeling ability with the predictability of optimal future returns from DP methods. Our policy is based on a Transformer structure, with an objective loss comprising two components: 1) a conditional behavior cloning term that aligns the Transformer's action sampling with the training set's distribution, and 2) a policy improvement term for selecting high-reward actions according to the learned Q-value. This hybrid structure offers multiple advantages. First, the trajectory prediction loss serves as an effective distribution-matching technique, functioning as a robust, sample-based policy regularization method, thus eliminating the need for additional behavior cloning. Second, the integration of policy improvement facilitates the identification and prioritization of higher-reward actions as per Q-values, ensuring that the expected returns of sampled actions align with the optimal returns. Third, the amalgamation of these two losses achieves a balance between selecting optimal actions and maintaining fidelity to the behavior policy, which mitigates the risk of preferring out-of-distribution actions with overestimated values, leading to enhanced performance.

In summary, our contributions are three-fold[2]:

- QT, a new offline RL algorithm that leverages Transformer to do precise policy regularization and Q-value regularization to align the expected returns of sampled actions with the optimal returns.
- QT aims to seek optimal actions that align closely with the behavior policy, ensuring robust stitching capability and effective trajectory modeling in scenarios characterized by long horizons and sparse rewards.
- We test QT on the D4RL benchmark tasks and demonstrate the superiority of QT over traditional DP and CSM methods, highlighting the potential of QT to enhance the state-of-the-art in offline RL.

[2]Our code is available at: https://github.com/charleshsc/QT



(a) (b)

Figure 1. Evaluation results for CQL, DT, QDT, and QT in the Maze2D tasks (a) and MuJoCo Gym delayed reward (medium) tasks (b). The results show that DT fails to effectively stitch trajectories and CQL under-performs in sparse reward scenarios (delayed reward). QDT yields consistent yet intermediate results across all environments, while QT consistently secures the top performance across all tested environments, showcasing its superiority.

2. Preliminary

2.1. Offline Reinforcement Learning

The goal of RL is to learn a policy $\pi_\theta(\mathbf{a}|\mathbf{s})$ maximizing the expected cumulative discounted rewards $\mathbb{E}[\sum_{t=0}^{\infty}\gamma^t\mathcal{R}(\mathbf{s}_t,\mathbf{a}_t)]$ in a Markov decision process (MDP), which is a six-tuple $(\mathcal{S},\mathcal{A},\mathcal{T},\mathcal{R},\gamma,d_0)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, environment dynamics $\mathcal{T}(\mathbf{s}'|\mathbf{s},\mathbf{a}) : \mathcal{S}\times\mathcal{S}\times\mathcal{A}\to[0,1]$, reward function $\mathcal{R}:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$, discount factor $\gamma\in[0,1)$, and initial state distribution d_0 (Sutton & Barto, 2018). The action-value or Q-value of a policy π is defined as $Q^\pi(\mathbf{s}_t,\mathbf{a}_t) = \mathbb{E}_{\mathbf{a}_{t+1},\mathbf{a}_{t+2}\cdots\sim\pi}[\sum_{i=0}^{\infty}\gamma^i\mathcal{R}(s_{t+i},a_{t+i})]$. In the offline setting (Levine et al., 2020), instead of the online environment, a static dataset $\mathcal{D}=\{(\mathbf{s},\mathbf{a},\mathbf{s}',r)\}$, collected by a behavior policy π_β, is provided. Offline RL algorithms learn a policy entirely from this static offline dataset $\mathcal{D}$, without online interactions with the environment.

2.2. Rethinking Stitching in CSM

To address the stitching ability of CSM, alternative approaches have been proposed. For example, EDT (Wu et al., 2023) and CGDT (Wang et al., 2023) optimize the trajectory by dynamically filtering the optimal trajectory according to the learned value estimator; ESPER (Paster et al., 2022) clusters trajectories and utilizes the average cluster returns as conditions for the policy; DoC (Yang et al., 2022) conditions the policy on a latent representation of future trajectories, achieved by minimizing mutual information. Incorporating probabilistic statistics from multiple trajectories offers a promising solution for sub-optimal data, which guides policy behaviors with learned estimated returns from the entire distribution of future trajectories. Although these methods exhibit effectiveness in stitching ability, they often necessitate complex objectives for representation learning and additional steps such as statistics, thereby complicating and burdening the training process.

Other approaches exploit the capabilities of Q-learning, which propagates optimal future returns backward for each state, considering each time step individually, thereby effectively stitching the optimal trajectory from sub-optimal data. QDT (Yamagata et al., 2023) takes the first attempt to combine these two methods by learning a conservative value function to relabel the RTG tokens in the dataset, keeping other components aligned with DT (Chen et al., 2021). However, such adaptations essentially constitute simple data augmentation, incorporating "stitched" trajectories into the training set but continuing to encounter unmatched RTG values during inference due to trajectory-level modeling (Wang et al., 2023), thereby failing to consistently exceed existing benchmarks. In contrast, QT employs the n-step Bellman equation to approximate the Q-value function based on sequence history. This Q-value function is then integrated into policy improvement to select high-reward actions while retaining the original DT loss for policy regularization. Such an approach not only empowers the CSM with the stitch ability but also keeps its original trajectory modeling ability important for the sparse-reward scenario. To substantiate this, we compared QT and QDT across various scenarios, including stitching ability scenarios like Maze2D, and sparse reward scenarios like MujoCo Gym with delayed rewards. The results, as illustrated in Figure 1, show that QT consistently achieves superior performance, while QDT's results are intermediate, failing to exceed existing methodologies (more details are presented in Section 4.2).

3. Methodology

We present a method that combines the trajectory modeling ability of Transformer with the predictability of optimal future returns from DP methods, thereby constructing a robust algorithm suitable for offline RL problems. Initially, we detail the application of the Conditional Transformer Policy as an expressive policy framework for behavior cloning. Subsequently, we describe the incorporation of a Q-value module into the training phase of our transformer policy, with the behavior cloning term serving as a policy regularization mechanism. Finally, we illustrate how to do the inference with the learned Q-value functions.

3.1. Conditional Transformer Policy

Transformer (Vaswani et al., 2017b), extensively studied in NLP (Devlin et al., 2018; Fan et al., 2022) and CV (Dosovitskiy et al., 2020; Fan et al., 2024), has also been explored in RL using the CSM pattern (Hu et al., 2022). Unlike the majority of prior RL approaches that estimate value functions or compute policy gradients, DT (Chen et al., 2021) outputs desired future actions from the history sequence, encompassing multiple state $\mathbf{s}_t$, action $\mathbf{a}_t$, and return-to-go $\hat{r}_t$ tuples. The return-to-go token quantifies the cumulative reward from the current time step to the end of the episode. During training with offline collected data, DT processes a trajectory sequence τ_t in an auto-regressive manner which encompasses the most recent K-step historical context:

$$\tau_t = (\hat{r}_{t-K+1}, \mathbf{s}_{t-K+1}, \mathbf{a}_{t-K+1}, \ldots, \hat{r}_t, \mathbf{s}_t, \mathbf{a}_t). \quad (1)$$

The prediction head associated with a state token $\mathbf{s}_t$ is trained to predict the corresponding action $\mathbf{a}_t$. Regarding continuous action spaces, the training objective is to minimize the mean-squared loss:

$$\mathcal{L}_{DT} = \mathbb{E}_{\tau_t \sim \mathcal{D}} \left[\frac{1}{K} \sum_{i=t-K+1}^{t} (\mathbf{a}_i - \pi(\tau_t)_i)^2 \right], \quad (2)$$

where $\pi(\tau_t)_i$ denotes the i-th action output of the Transformer policy in an auto-regressive manner.

Theorem 3.1. *Consider an MDP, behavior policy β, and decision transformer π with condition function f. Assume the ϵ-near determinism of the MDP, where $P(r \neq \mathcal{R}(\mathbf{s}, \mathbf{a})$ or $\mathbf{s}' \neq \mathcal{T}(\mathbf{s}, \mathbf{a}) | \mathbf{s}, \mathbf{a}) \leq \epsilon$ at all $\mathbf{s}, \mathbf{a}$ for some functions $\mathcal{T}$ and $\mathcal{R}$. Let $g(\tau) = \sum_{t=1}^{\mathcal{H}} r_t$, when $P_\beta(g(\tau) = f(\mathbf{s}_1)|\mathbf{s}_1) \geq \alpha_f$ for all initial states $\mathbf{s}_1$, we have:*

$$\mathbb{E}_{\tau \sim \beta}[g(\tau)] - \mathbb{E}_{\tau \sim \pi_f}[g(\tau)] \leq \epsilon(\frac{1}{\alpha_f} + 2)\mathcal{H}^2, \quad (3)$$

where $\mathcal{H}$ is the horizon of the MDP.

Theorem 3.1 demonstrates that training with the DT loss $\mathcal{L}_{DT}$ leads to the gradual convergence of the generated policy towards the behavior policy β. This convergence, however, imposes a constraint that restricts the generated policy from exceeding the performance of the behavior trajectories present in the offline dataset $\mathcal{D}$. Moreover, training exclusively with the DT loss $\mathcal{L}_{DT}$ restricts the stitching ability, resulting in a policy predominantly biased towards actions observed in the training trajectories (Paster et al., 2022). Due to limited space, the proof of this theorem, as well as other results, are provided in the Appendix A.

3.2. Training with Q-value Regularization

To address the stitching challenge and develop a policy capable of aligning the expected returns of sampled actions with the optimal returns, we employ the Q-value module.

The Q-value function is learned conventionally, minimizing the Bellman operator (Fujimoto et al., 2019b) and employing the double Q-learning technique (Hasselt, 2010). We construct two Q-networks, Q_{ϕ_1}, Q_{ϕ_2}, along with their respective target networks, $Q_{\phi'_1}, Q_{\phi'_2}$ and target policy $\pi_{\theta'}$. Given that the input to the transformer policy includes trajectory history, we opt for the n-step Bellman equation to estimate the Q-value function. This choice is premised on

Algorithm 1 QT: Q-value regularized Transformer

Input: Sequence horizon K, offline datasets $\mathcal{D}$, coefficient ρ, a set of candidate return-to-go $\{\hat{r}_0^0, \hat{r}_0^1, \ldots, \hat{r}_0^m\}$.
Initialize policy network π_θ, critic networks Q_{ϕ_1} and Q_{ϕ_2}, and target networks $\pi_{\theta'}, Q_{\phi_1'}$ and $Q_{\phi_2'}$.
// Train the QT
for $t = 1$ **to** T **do**
  Sample sequence transition mini-batch $\mathcal{B} = \{(\hat{r}_j, \mathbf{s}_j, \mathbf{a}_j, r_j)_{j=t}^{t+K}\} \sim \mathcal{D}$.
  // Q-value function learning
  Sample $\hat{\mathbf{a}}_{t+K} \sim \pi_{\theta'}(\hat{\mathbf{a}}_{t+K}|\hat{r}_{t:t+K}, \mathbf{s}_{t:t+K}, \mathbf{a}_{t:t+K-1})$.
  Update Q_{ϕ_1} and Q_{ϕ_2} by Equation 4.
  // Policy learning
  for $i = 1$ **to** K **do**
    Sample $\hat{\mathbf{a}}_{t+i} \sim \pi_\theta(\hat{\mathbf{a}}_{t+i}|\hat{r}_{t:t+i}, \mathbf{s}_{t:t+i}, \mathbf{a}_{t:t+i-1})$ in an auto-regressive way.
  end for.
  Update policy by minimizing Equation 5.
  // Update target networks
  $\theta' = \rho\theta' + (1-\rho)\theta, \phi_i' = \rho\phi_i' + (1-\rho)\phi_i$ for $i = \{1, 2\}$.
end for.
// Inference with QT
Given multiple target return-to-go choice $\hat{r}_0^{1:m}$ and initial state s_0.
repeat
  Sample multiple actions with different return-to-go $\hat{\mathbf{a}}_t^i = \pi_\theta(\hat{\mathbf{a}}_t^i|\hat{r}_{t-K+1:t}^i, \mathbf{s}_{t-K+1:t}, \mathbf{a}_{t-K+1:t-1})$ for $i = 1, \ldots, m$.
  Compute Q value with candidate state-action pair $(\mathbf{s}_t, \hat{\mathbf{a}}_t^i)$ for $i = 1, \ldots, m$.
  Sample the action $\mathbf{a}_t$ from action set $\{\hat{\mathbf{a}}_t^i\}_{i=1}^m$ with the max Q value by Equation 6.
  Execute the action $\mathbf{a}_t$ and collect the reward r_t and next state $\mathbf{s}_{t+1}$.
  Update current return-to-go $\hat{r}_{t+1}^i = \hat{r}_t^i - r_t$ for $i = 1, \ldots, m$.
until $Done$ is $true$.

its demonstrated improvement over the 1-step approximation (Sutton & Barto, 2018). The optimization of ϕ_i for $i = \{1, 2\}$ is carried out by minimizing following equation:

$$\mathbb{E}_{\tau_t \sim \mathcal{D}, \hat{\mathbf{a}}_t \sim \pi_{\theta'}} \sum_{m=t-K+1}^{t-1} \left|\left|\hat{Q}_m - Q_{\phi_i}(\mathbf{s}_m, \mathbf{a}_m)\right|\right|^2, \quad (4)$$

$$\text{where } \hat{Q}_m = \sum_{j=m}^{t-1} \gamma^{j-m} r_j + \gamma^{t-m} \min_{i=1,2} Q_{\phi_i'}(\mathbf{s}_t, \hat{\mathbf{a}}_t),$$

where γ is the discount factor and $\hat{\mathbf{a}}_t$ denotes the predicted action output by the target model $\pi_{\theta'}$.

To enhance the policy, we integrate a Q-value module during the training phase, enabling the preferential sampling of high-value actions. The final policy learning objective emerges as a linear combination of policy regularization and policy improvement elements:

$$\pi = \arg\min_{\pi_\theta} \{\mathcal{L}(\theta) := \mathcal{L}_{DT}(\theta) + \mathcal{L}_Q(\theta)\} \quad (5)$$
$$= \arg\min_{\pi_\theta} \mathcal{L}_{DT}(\theta) - \alpha \cdot \mathbb{E}_{\tau_t \sim \mathcal{D}} \mathbb{E}_{(\mathbf{s}_i, \mathbf{a}_i) \sim \tau_t} Q_\phi(\mathbf{s}_i, \pi(\tau_t)_i).$$

Considering the variation in the scale of the Q-value function across different offline datasets, we adopt a normalization technique from Fujimoto & Gu (2021). We define α as $\alpha = \frac{\eta}{\mathbb{E}_{\tau_t \sim \mathcal{D}} \mathbb{E}_{(\mathbf{s}, \mathbf{a}) \sim \tau_t}[|Q_\phi(\mathbf{s}, \mathbf{a})|]}$, where η is a hyper-parameter that mediates the balance between the two loss terms. Notably, the Q-value in the denominator serves exclusively for normalization and is not subject to differentiation.

Furthermore, we affirm the efficacy of Equation 5 from a theoretical standpoint as delineated in Theorem 3.2, suggesting that the learned final policy is anticipated to consistently outperform the behavior policy in terms of the value function. Specifically, it highlights how the Q-value regularization enhances the policy by enabling preferential sampling of high-value actions, aligning the learning process more closely with optimal returns. This implicitly ensures an improvement over the baseline behavior policy β.

Theorem 3.2. *Let π^* be the optimal policy of Equation 5. For any $\mathbf{s} \in \mathcal{S}$, we have that $V^{\pi^*}(\mathbf{s}) \geq V^\beta(\mathbf{s})$ and $\pi^*(\mathbf{a}|\mathbf{s}) = 0$ given $\beta(\mathbf{a}|\mathbf{s}) = 0$.*

3.3. Inference with Q-value Module

Instead of carefully designing the return-to-go token value in the previous conditional transformer policy, which needs more trials and tuning to find the best value, we sample multiple candidate return-to-go tokens $\{\hat{r}_0^0, \hat{r}_0^1, \ldots, \hat{r}_0^m\}$ and simultaneously output actions in accordance with different return-to-go values. Then we resort to the learned Q-value function to preferentially sample actions with high returns,

Table 1. The performance of QT and SOTA baselines on D4RL Gym, Adroit, Kitchen, Maze2D, and AntMaze tasks. Results for QT correspond to the mean and standard errors of normalized scores over 30 random rollouts (3 independently trained models and 10 trajectories per model) for all tasks, which generally exhibit low variance in performance. Our method outperforms all prior methods by a clear margin in almost all domains, including the conventional Q-learning algorithms and CSM methods.

Gym Tasks	**CQL**	**IQL**	**BCQ**	**BEAR**	**TD3+BC**	**MoRel**	**BC**	**DD**	**DT**	**StAR**	**GDT**	**CGDT**	**QT**
halfcheetah-medium-expert-v2	91.6	86.7	69.6	53.4	90.7	53.3	55.2	90.6	86.8	93.7	93.2	93.6	**96.1** $\pm$ 0.2
hopper-medium-expert-v2	105.4	91.5	109.1	96.3	98.0	108.7	52.5	111.8	107.6	111.1	111.1	107.6	**113.4** $\pm$ 0.4
walker2d-medium-expert-v2	108.8	109.6	67.3	40.1	110.1	95.6	107.5	108.8	108.1	109.0	107.7	109.3	**112.6** $\pm$ 0.6
halfcheetah-medium-v2	49.2	47.4	41.5	41.7	48.4	42.1	42.6	49.1	42.6	42.9	42.9	43.0	**51.4** $\pm$ 0.4
hopper-medium-v2	69.4	66.3	65.1	52.1	59.3	95.4	52.9	79.3	67.6	59.5	77.1	**96.9**	**96.9** $\pm$ 3.1
walker2d-medium-v2	83.0	78.3	52.0	59.1	83.7	77.8	75.3	82.5	74.0	73.8	76.5	79.1	**88.8** $\pm$ 0.5
halfcheetah-medium-replay-v2	45.5	44.2	34.8	38.6	44.6	40.2	36.6	39.3	36.6	36.8	40.5	40.4	**48.9** $\pm$ 0.3
hopper-medium-replay-v2	95.0	94.7	31.1	33.7	60.9	93.6	18.1	100.0	82.7	29.2	85.3	93.4	**102.0** $\pm$ 0.2
walker2d-medium-replay-v2	77.2	73.9	13.7	19.2	81.8	49.8	32.3	75.0	79.4	39.8	77.5	78.1	**98.5** $\pm$ 1.1
Average	80.6	77.0	53.8	48.2	75.3	72.9	52.6	81.8	76.2	66.2	79.1	82.4	**89.8**
Adroit Tasks	**CQL**	**IQL**	**BCQ**	**BEAR**	**O-RL**	**MoRel**	**BC**	**DD**	**D-QL**	**DT**	**StAR**	**GDT**	**QT**
pen-human-v1	37.5	71.5	66.9	-1.0	90.7	-3.2	63.9	66.7	72.8	79.5	77.9	92.5	**129.6** $\pm$ 4.6
hammer-human-v1	4.4	1.4	0.9	0.3	0.2	2.7	1.2	1.9	0.2	3.7	3.7	5.5	**35.6** $\pm$ 7.0
door-human-v1	9.9	4.3	-0.05	-0.3	-0.1	2.2	2.0	2.8	0.0	14.8	1.5	20.6	**28.7** $\pm$ 2.4
pen-cloned-v1	39.2	37.3	50.9	26.5	60	-0.2	37.0	42.8	57.3	75.8	33.1	86.2	**125.0** $\pm$ 2.8
hammer-cloned-v1	2.1	2.1	0.4	0.3	2.0	2.3	0.6	1.7	3.1	3.0	0.3	8.9	**23.0** $\pm$ 2.3
door-cloned-v1	0.4	1.6	0.01	-0.1	0.4	2.3	0.0	1.3	0.0	16.3	0.0	19.8	**20.6** $\pm$ 1.7
Average	15.6	19.7	19.8	4.3	25.5	1.0	17.5	19.5	22.2	32.2	19.4	38.9	**60.4**
Kitchen Tasks	**CQL**	**IQL**	**BCQ**	**BEAR**	**TD3+BC**	**O-RL**	**BC**	**DD**	**D-QL**	**DT**	**StAR**	**GDT**	**QT**
kitchen-complete-v0	43.8	62.5	8.1	0.0	0.0	2.0	65.0	65.0	**84.0**	50.8	40.8	43.8	81.7 $\pm$ 1.2
kitchen-partial-v0	49.8	46.3	18.9	13.1	0.0	35.5	33.8	57.0	60.5	57.9	12.3	73.3	**75.0** $\pm$ 0.1
Average	46.8	54.4	13.5	6.6	0.0	18.8	51.5	61	72.3	54.4	26.6	58.6	**78.4**
Maze2D Tasks	**CQL**	**IQL**	**BCQ**	**BEAR**	**TD3+BC**	**COMBO**	**BC**	**Diffuser**	**DD**	**DT**	**GDT**	**QDT**	**QT**
maze2d-umaze-v1	94.7	42.1	49.1	65.7	14.8	76.4	88.9	113.9	**116.2**	31.0	50.4	57.3	105.4 $\pm$ 4.7
maze2d-medium-v1	41.8	34.9	17.1	25.0	62.1	68.5	38.3	121.5	122.3	8.2	7.8	13.3	**172.0** $\pm$ 6.2
maze2d-large-v1	49.6	61.7	30.8	81.0	88.6	14.1	1.5	123.0	125.9	2.3	0.7	31.0	**240.1** $\pm$ 2.5
Average	62.0	46.2	32.3	57.2	55.2	53.0	42.9	119.5	121.5	13.8	19.6	33.9	**172.5**
AntMaze Tasks	**CQL**	**IQL**	**BCQ**	**BEAR**	**TD3+BC**	**O-RL**	**BC**	**DD**	**D-QL**	**DT**	**StAR**	**GDT**	**QT**
antmaze-umaze-v0	74.0	87.5	78.9	73.0	78.6	64.3	54.6	73.1	93.4	59.2	51.3	76.0	**96.7** $\pm$ 4.7
antmaze-umaze-diverse-v0	84.0	62.2	55.0	61.0	71.4	60.7	45.6	49.2	66.2	53.0	45.6	69.0	**96.7** $\pm$ 4.7
antmaze-medium-diverse-v0	53.7	70.0	0.0	8.0	3.0	0.0	0.0	24.6	**78.6**	0.0	0.0	6.0	59.3 $\pm$ 0.9
antmaze-large-diverse-v0	14.9	47.5	2.2	0.0	0.0	0.0	0.0	7.5	**56.6**	0.0	0.0	0.0	53.3 $\pm$ 4.7
Average	56.7	66.8	34.0	57.2	38.3	31.3	25.1	61.2	73.7	28.1	24.2	37.8	**76.5**

which could be formulated as:

$$\arg\max_{\hat{\mathbf{a}}_t^i} \; Q_{\phi'}(\mathbf{s}_t, \hat{\mathbf{a}}_t^i), \tag{6}$$
$$\text{where} \quad \hat{\mathbf{a}}_t^i = \pi(\hat{r}_{t-K+1:t}^i, \mathbf{s}_{t-K+1:t}, \mathbf{a}_{t-K+1:t-1})).$$

This process is highly parallelizable. By assigning different RTG values to each batch, we can leverage GPU capabilities to concurrently generate multiple action sequences, thereby minimizing additional computational overhead. Corresponding ablation studies are conducted to demonstrate the efficacy of this procedure, as detailed in Section 4.2 and Appendix D. The training and inference procedures are thoroughly outlined in Algorithm 1, providing a comprehensive summary of the processes involved.

4. Experiment

In this section, we present an extensive evaluation of our proposed QT model using the widely recognized D4RL benchmark (Fu et al., 2020). Our main objective is to assess the effectiveness of QT across various domains, setting it against two prevalent algorithms: Q-learning methods and CSM algorithms. Each of these algorithms demonstrates proficiency in specific domains while exhibiting sub-optimal performance in others. Additionally, we execute an empirical ablation study to dissect and understand the individual contributions of the core components of our methodology.

Datasets. We consider five different domains of tasks in D4RL benchmark: Gym, Adroit, Kitchen, Maze2D, and

Table 2. Ablation on the role of different components. Average and standard deviation scores are reported over 3 seeds for the walker2d-medium-replay task. 'CTP' refers to the Conditional Transformer Policy as detailed in Section 3.1, 'none' indicates the absence of the Q-value module in the configuration, and 'Inf.' is short for inference.

Exp	Policy	Q-value Update	Train with Q-value	Inf. with Q-value	Performance
1	BC	none			32.3 ± 9.8
2	BC	n-step	✓	✓	82.2 ± 0.5
3	CTP	none			79.4 ± 2.0
4	CTP	n-step		✓	87.6 ± 1.1
5	CTP	n-step	✓		97.7 ± 0.3
6	CTP	1-step	✓	✓	85.6 ± 1.7
7	CTP	n-step	✓	✓	98.5 ± 1.1

AntMaze. The Gym-MuJoCo locomotion tasks, commonly used as standard benchmarks, are relatively straightforward and characterized by datasets with a significant proportion of near-optimal trajectories and smooth reward functions. In contrast, the Adroit datasets, primarily derived from human behaviors, exhibit a limited state-action space, necessitating robust policy regularization to maintain agent performance within the expected range. The Kitchen environment poses a multi-task challenge, requiring the agent to complete four sequential sub-tasks to achieve a desired state configuration, thereby emphasizing the importance of generalization to unseen states rather than relying purely on trajectories seen during training. Maze2D tasks are designed to assess an offline RL algorithm's capability to effectively stitch together sub-trajectories to identify the shortest path to a set goal. Lastly, AntMaze presents more demanding scenarios with sparse rewards, substituting the simpler 2D ball in Maze2D for a complex 8-DoF "Ant" quadruped robot, thereby elevating the difficulty level.

Baselines. We benchmark against a diverse array of baseline methods, each excelling in specific domain tasks. For policy regularization-based approaches, our selection includes IQL (Kostrikov et al., 2021b), BCQ (Fujimoto et al., 2019a), BEAR (Kumar et al., 2019a), TD3+BC (Fujimoto & Gu, 2021), and O-RL (Brandfonbrener et al., 2021). We also consider the CQL (Kumar et al., 2020) for Q-value constraint methods. In the realm of model-based offline RL, we evaluate against MoRel (Kidambi et al., 2020) and COMBO (Yu et al., 2021). For CSM approaches, our comparisons include DT (Chen et al., 2021), StAR (Shang et al., 2022), QDT (Yamagata et al., 2023), GDT (Hu et al., 2023a), and CGDT (Wang et al., 2023). Additionally, we assess diffusion-based methods such as Diffuser (Janner et al., 2022), DD (Ajay et al., 2022), and Diffusion-QL (Wang et al., 2022). The performance scores for these baseline methods are sourced either from the best results published in their respective papers or from our own runs, ensuring a fair comparison.

4.1. Main Results

We compare our QT with the baselines on five domains of tasks and report the results in Table 1. To ensure fair comparisons, we normalize the scores according to the protocol established in Fu et al. (2020), where a score of 100 corresponds to an expert policy. We give the analysis based on each specific domain.

Results for Gym Domain. We can see while most baseline models demonstrate proficiency on Gym tasks, QT often achieves further enhancements, particularly in 'medium' and 'medium-replay' tasks, surpassing other Transformer-based methods by a large margin. It's noteworthy that these datasets encompass trajectories generated by an online SAC (Haarnoja et al., 2018) agent, trained to reach roughly one-third of an expert's performance. Consequently, other Transformer-based methods typically underperform compared to Q-learning approaches in the absence of an ample quantity of high-quality trajectories (Emmons et al., 2021), as seen in the medium-expert dataset. As elucidated in Section 3, the incorporation of a policy improvement term in QT directs the policy towards optimal actions within the explored action space subset, significantly contributing to QT's commendable empirical performance.

Results for Adroit and Kitchen Domain. In the Adroit domain, where offline RL is particularly challenged by extrapolation error due to the limited scope of human demonstrations (Fu et al., 2020), robust policy regularization is essential. Our Transformer-based policy, employing the DT loss $\mathcal{L}_{DT}$, significantly outperforms diffusion-based baselines. This superiority is attributable to its high expressiveness and more effective policy regularization. Furthermore, the Kitchen tasks, which demand generalization to unseen states and long-term value optimization, also witness notable performance improvements with QT, underscoring its adaptability and effectiveness in this domain.

Results for Maze2D and AntMaze Domain. The Maze2D domain serves as a benchmark to evaluate the capacity of offline RL algorithms to effectively stitch segments of disparate trajectories (Fu et al., 2020). Integrating the Q-value module with the Transformer policy enhances its ability to navigate the shortest path to the goal using pre-collected sub-trajectories. The AntMaze domain, characterized by sparse rewards and an abundance of sub-optimal trajecto-

Table 3. Ablation on the stitching ability. Average and standard deviation scores are reported over 3 seeds for the Maze2D tasks. This encompasses four increasingly complex mazes—open, umaze, medium, and large—each with two reward functions: normal and dense. The highest average scores are highlighted in bold.

	Dataset	CQL	DT	QDT	QT
Sparse Reward	maze2d-open-v0	216.7 ± 80.7	196.4 ± 39.6	190.1 ± 37.8	**497.9** ± 12.3
	maze2d-umaze-v1	94.7 ± 23.1	31.0 ± 21.3	57.3 ± 8.2	**105.4** ± 4.8
	maze2d-medium-v1	41.8 ± 13.6	8.2 ± 4.4	13.3 ± 5.6	**172.0** ± 6.2
	maze2d-large-v1	49.6 ± 8.4	2.3 ± 0.9	31.0 ± 19.8	**240.1** ± 2.5
Dense Reward	maze2d-open-dense-v0	307.6 ± 43.5	346.2 ± 14.3	325.7 ± 61.4	**608.4** ± 1.9
	maze2d-umaze-dense-v1	72.7 ± 10.1	-6.8 ± 10.9	58.6 ± 3.3	**103.1** ± 7.8
	maze2d-medium-dense-v1	70.9 ± 9.2	31.5 ± 3.7	42.3 ± 7.1	**111.9** ± 1.9
	maze2d-large-dense-v1	90.9 ± 19.4	45.3 ± 11.2	62.2 ± 9.9	**177.2** ± 7.8

ries, presents a more difficult challenge. A robust and stable Q-learning approach is essential for achieving notable performance in this setting. Empirically, QT, augmented with our Q-value module and an optimally tuned hyper-parameter η, either matches or exceeds the performance of existing methods, whereas other Transformer-based approaches often struggle in 'medium' and 'large' tasks.

4.2. Ablation Study

This section delves into a quantitative analysis of QT's superior performance over other Transformer-based methods on D4RL tasks. We undertake an ablation study to dissect and quantify the contributions of QT's main components to its overall efficacy. Additionally, further ablations are conducted to assess whether QT successfully integrates the strengths of both CSM and Q-learning methods while overcoming their limitations. We select CQL as the benchmark for evaluating the Q-learning approach, and DT as the benchmark for assessing the CSM approach. We also include QDT as a comparative benchmark in order to showcase the differences between QDT and our approach. Note that further discussion about QT is provided in Appendix D.

Role of Different Components. As delineated in Section 3, our methodology comprises three primary components, alongside the Q-value update method, each warranting individual analysis. We select the walker2d-medium-replay dataset as the benchmark due to its diverse range of agent levels and the substantial performance enhancement QT demonstrates compared to baselines. As indicated in Table 2, integrating our Q-value module significantly boosts performance, as evidenced by the comparative results between experiments 1 vs. 2, and 3 vs. 7. Notably, the Q-value regularization (Equation 5) during the training stage is instrumental, manifesting as the most significant contributor to performance enhancement, with the inference phase also benefiting from the Q-value module (as seen in comparisons among experiments 3 vs. 4, and 5 vs. 7). Furthermore, relying solely on the 1-step Bellman equation for updating the Q-value function results in subpar performance compared to the n-step Bellman equation (as seen in comparisons between experiments 6 and 7), which underscores the criticality of Q-value function accuracy in our methodology.

Stitching Ability. The Maze2D domain, a navigation task with a fixed goal location, serves as a critical test for offline RL algorithms' ability to stitch together different trajectory segments (Fu et al., 2020). This domain comprises four increasingly complex mazes—open, umaze, medium, and large—and utilizes two reward functions: normal and dense. The normal reward is granted solely upon goal achievement, while the dense reward is incrementally distributed at each step, inversely proportional to the distance from the goal. Table 3 summarizes the results. CQL performs notably well, particularly with dense rewards. DT, however, often struggles due to its limited stitching capability. QDT demonstrates a marked improvement over DT but still lags behind CQL. Significantly, QT excels across all tasks, affirming its ability to not only endow the Transformer policy with stitching capacity but also synergistically merge the strengths of both methodologies for enhanced performance.

Sparse Reward Ability. To illustrate the limitations of the Q-learning approach (CQL), we follow Chen et al. (2021) and evaluate the algorithms in a delayed (sparse) reward setting, where rewards are withheld during the trajectory and aggregated at the final timestep. Table 4 presents the results for both delayed (sparse) and dense reward scenarios. As anticipated, CQL exhibits difficulty in formulating an effective policy under sparse conditions, in contrast to DT, which demonstrates commendable performance. QDT, which employs CQL for RTG token value relabeling, registers inferior performance compared to DT, influenced by CQL's inaccurate value function estimations. Conversely, QT, while similarly impacted by these inaccurate estimations in sparse reward scenarios, benefits from our robust policy regularization. This feature effectively mitigates the adverse effects of the Q-value module, enabling QT to outperform these methods across all assessed tasks.

Long Task Horizon Ability. While in a Markovian environment, the state at the previous moment is often sufficient

Table 4. Ablation on the sparse reward ability. Average and standard deviation scores are reported over 3 seeds for the D4RL tasks. The study includes three tasks—halfcheetah, hopper, and walker2d—each evaluated under two reward conditions: sparse and dense. The highest average scores are denoted in bold.

Dataset	Sparse Reward				Dense Reward			
	DT	CQL	QDT	QT	DT	CQL	QDT	QT
halfcheetah-medium-v2	42.2 ± 0.2	1.0 ± 1.0	42.4 ± 0.5	**43.3 ± 0.2**	42.6 ± 0.1	49.2 ± 0.5	42.3 ± 0.4	**51.4 ± 0.4**
hopper-medium-v2	57.3 ± 2.4	23.3 ± 1.0	50.7 ± 5.0	**72.7 ± 3.9**	67.6 ± 1.0	69.4 ± 13.1	66.5 ± 6.3	**96.3 ± 3.1**
walker2d-medium-v2	69.9 ± 2.0	0.0 ± 0.4	63.7 ± 6.4	**80.7 ± 0.8**	74.0 ± 1.4	83.0 ± 0.6	67.1 ± 3.2	**88.8 ± 0.5**
halfcheetah-medium-replay-v2	33.0 ± 4.8	7.8 ± 6.9	32.8 ± 7.3	**42.5 ± 0.2**	36.6 ± 0.8	45.5 ± 0.5	35.6 ± 0.5	**48.9 ± 0.3**
hopper-medium-replay-v2	50.8 ± 14.3	7.7 ± 5.9	38.7 ± 26.7	**94.2 ± 2.2**	82.7 ± 7.0	95.0 ± 2.9	52.1 ± 20.3	**102.0 ± 0.2**
walker2d-medium-replay-v2	51.6 ± 24.6	3.2 ± 1.7	29.6 ± 15.5	**78.5 ± 2.1**	66.6 ± 3.0	77.2 ± 1.1	58.2 ± 5.1	**98.5 ± 1.1**
Average	50.8	7.2	43.0	**68.6**	61.7	69.9	53.6	**81.0**

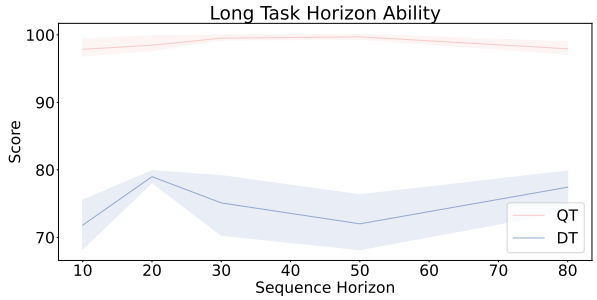


Figure 2. Ablation on the long task horizon ability. This encompasses the performance comparison of different input sequence horizons $K \in [10, 80]$ in the walker2d-medium-replay-v2 task.

to determine the current action, the DT experiment reveals that past information is valuable for the sequence modeling method in some environments, where longer sequences tend to yield better results than those of length 1. We then explore the impact of different sequence lengths on performance and compare the results of DT and QT, where Q-learning methods often perform badly in the long horizon setting (Yamagata et al., 2023; Bhargava et al., 2023). The results are shown in Figure 2. As the sequence horizon K extends, both agents exhibit improved performance. DT initially deteriorates after $K = 20$ but recovers at $K = 80$, whereas QT consistently enhances its performance, demonstrating a superior capability to manage extended task horizons.

5. Related Work

Offline RL algorithms learn a policy entirely from this static offline dataset $\mathcal{D}$, without online interactions with environment (Levine et al., 2020). This paradigm can be precious in case the interaction with environment is expensive or high-risk (e.g., safety-critical applications). However as the learned policy might differ from the behavior policy, the offline algorithms must mitigate the effect of the *distribution shift*, which can result in a significant performance drop, as demonstrated in prior research (Fujimoto et al., 2019b).

Q-learning method is one of the most prominent categories to address the *distribution shift* problem. Especially, previous Q-learning works generally address this problem in one of three ways: 1) constraining the learned policy to the behavior policy (Kumar et al., 2019a; Fujimoto et al., 2019b; Fujimoto & Gu, 2021; Wu et al., 2019; Lyu et al., 2022); 2) constraining the learned policy by making conservative estimates of future rewards (Kumar et al., 2020; Kostrikov et al., 2021a; Chebotar et al., 2023); 3) introducing model-based methods, which learn a model of the environment dynamics to generate more data for policy training and perform pessimistic planning in the learned MDP (Janner et al., 2019; Kidambi et al., 2020; Yu et al., 2021).

Weighted imitation learning addresses the *distribution shift* without restricting the learned policy, which carries out imitation learning by putting higher weights on the good state-action pairs. These methods (Wang et al., 2018; Peng et al., 2019; Wang et al., 2020; Chen et al., 2020; Siegel et al., 2020) usually use an estimated advantage function as the weight. As these approaches imitate the selected parts of the behavior policy, they naturally restrict the learned policy within the behavior policy.

Conditional sequence modeling is the other group of approaches without restricting the learning policy, which predicts subsequent actions from a sequence of past experiences, encompassing state-action-reward triplets. This paradigm lends itself to a supervised learning approach, inherently constraining the learned policy within the boundaries of the behavior policy and focusing on a policy conditioned on specific metrics for future trajectories (Chen et al., 2021; Hu et al., 2023b; Brandfonbrener et al., 2022; Hu et al., 2024b; Meng et al., 2023; Hu et al., 2024a). Moreover, the sequence of trajectories could also be formulated as a conditional generative process and generated by the diffusion model while satisfying conditioned constraints (Janner et al., 2022; Ajay et al., 2022; Wang et al., 2022).

Our approach is distinct from but related to these primary classes of offline RL algorithms. Essentially, our method is a CSM approach as it learns the subsequent actions based on historical sequences and sampled future rewards. Also, the high-level framework of our approach is somewhat

akin to weighted imitation learning, wherein a value function is employed to assign weights to various state-action pairs. However, the practical application of our components markedly differs. Unlike approaches that use the value function merely for training data weighting, our method integrates a learned Q-value module directly into the training phase, which biases action sampling towards higher-return options, a factor that has empirically demonstrated enhanced performance in our experiments.

6. Conclusion

In this study, we introduce QT, which combines the trajectory modeling ability of Transformer with the predictability of optimal future returns from DP methods. QT offers a novel framework for enhancing offline RL algorithms. The Conditional Transformer Policy of QT allows for a highly expressive policy class whose learning itself acts as a strong policy regularization method. Additionally, the integration of a Q-value regularization via a jointly learned Q-value function biases action sampling towards optimal regions within the exploration space. Empirical evaluations on D4RL benchmark datasets demonstrate the superiority of QT over traditional DP and CSM methods, highlighting the potential of QT to enhance the SOTA in offline RL.

Limitation. We introduce a novel Transformer-based policy for offline RL, achieving state-of-the-art performance across various tasks. However, QT's efficacy depends on the availability of explicit reward signals. In scenarios lacking explicit reward signals, such as datasets containing only state-action pairs from human demonstrations, QT's performance may be limited.

Acknowledgements

This work is supported by the National Key R&D Program of China (No. 2022ZD0160702), STCSM (No. 22511106101, No. 22511105700, No. 21DZ1100100), 111 plan (No. BP0719010) and National Natural Science Foundation of China (No. 62306178). Dr Tao's research is partially supported by NTU RSR and Start Up Grants.

Impact Statement

This paper contributes to the advancement of Offline Reinforcement Learning. While there are many potential societal consequences of our work, we believe that none require specific emphasis in this context.

References

Abbasi-Yadkori, Y., Bartlett, P., Bhatia, K., Lazic, N., Szepesvari, C., and Weisz, G. Politex: Regret bounds for policy iteration using expert prediction. In *International Conference on Machine Learning*, pp. 3692–3702. PMLR, 2019.

Ajay, A., Du, Y., Gupta, A., Tenenbaum, J., Jaakkola, T., and Agrawal, P. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

Bhargava, P., Chitnis, R., Geramifard, A., Sodhani, S., and Zhang, A. Sequence modeling is a robust contender for offline reinforcement learning. *arXiv preprint arXiv:2305.14550*, 2023.

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

Brandfonbrener, D., Bietti, A., Buckman, J., Laroche, R., and Bruna, J. When does return-conditioned supervised learning work for offline reinforcement learning? *Advances in Neural Information Processing Systems*, 35: 1542–1553, 2022.

Chebotar, Y., Vuong, Q., Hausman, K., Xia, F., Lu, Y., Irpan, A., Kumar, A., Yu, T., Herzog, A., Pertsch, K., et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Conference on Robot Learning*, pp. 3909–3928. PMLR, 2023.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Chen, X., Zhou, Z., Wang, Z., Wang, C., Wu, Y., and Ross, K. Bail: Best-action imitation learning for batch deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33, 2020.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Emmons, S., Eysenbach, B., Kostrikov, I., and Levine, S. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021.

Fan, Z., Wang, Y., Yao, J., Lyu, L., Zhang, Y., and Tian, Q. Fedskip: Combatting statistical heterogeneity with federated skip aggregation. In *IEEE International Conference on Data Mining*, 2022.

Fan, Z., Yao, J., Han, B., Zhang, Y., Wang, Y., et al. Federated learning with bilateral curation for partially class-disjoint data. *Advances in Neural Information Processing Systems*, 2024.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

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

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019a.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019b.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Hasselt, H. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

Hu, S., Shen, L., Zhang, Y., Chen, Y., and Tao, D. On transforming reinforcement learning by transformer: The development trajectory. *arXiv preprint arXiv:2212.14164*, 2022.

Hu, S., Shen, L., Zhang, Y., and Tao, D. Graph decision transformer. *arXiv preprint arXiv:2303.03747*, 2023a.

Hu, S., Shen, L., Zhang, Y., and Tao, D. Prompt-tuning decision transformer with preference ranking. *arXiv preprint arXiv:2305.09648*, 2023b.

Hu, S., Fan, Z., Shen, L., Zhang, Y., Wang, Y., and Tao, D. Harmodt: Harmony multi-task decision transformer for offline reinforcement learning. In *International Conference on Machine Learning*, 2024a.

Hu, S., Shen, L., Zhang, Y., and Tao, D. Learning multi-agent communication from graph modeling perspective. In *The Twelfth International Conference on Learning Representations*, 2024b.

Hu, X., Ma, Y., Xiao, C., Zheng, Y., and Jianye, H. Iteratively refined behavior regularization for offline reinforcement learning. In *NeurIPS 2023 Workshop on Distribution Shifts: New Frontiers with Foundation Models*, 2023c.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34: 1273–1286, 2021.

Janner, M., Du, Y., Tenenbaum, J. B., and Levine, S. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33: 21810–21823, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kostrikov, I., Fergus, R., Tompson, J., and Nachum, O. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pp. 5774–5783. PMLR, 2021a.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021b.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019a.

Kumar, A., Peng, X. B., and Levine, S. Reward-conditioned policies. *arXiv preprint arXiv:1912.13465*, 2019b.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Lyu, J., Ma, X., Li, X., and Lu, Z. Mildly conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35, 2022.

Meng, L., Wen, M., Le, C., Li, X., Xing, D., Zhang, W., Wen, Y., Zhang, H., Wang, J., Yang, Y., et al. Offline pre-trained multi-agent decision transformer. *Machine Intelligence Research*, 2023.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Bridging the gap between value and policy based reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Paster, K., McIlraith, S., and Ba, J. You can't count on luck: Why decision transformers and rvs fail in stochastic environments. *Advances in Neural Information Processing Systems*, 35, 2022.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Shang, J., Kahatapitiya, K., Li, X., and Ryoo, M. S. Starformer: Transformer with state-action-reward representations for visual reinforcement learning. In *European Conference on Computer Vision*. Springer, 2022.

Siegel, N. Y., Springenberg, J. T., Berkenkamp, F., Abdolmaleki, A., Neunert, M., Lampe, T., Hafner, R., Heess, N., and Riedmiller, M. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

Srivastava, R. K., Shyam, P., Mutz, F., Jaśkowski, W., and Schmidhuber, J. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017a.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017b.

Wang, Q., Xiong, J., Han, L., Liu, H., Zhang, T., et al. Exponentially weighted imitation learning for batched historical data. *Advances in Neural Information Processing Systems*, 31, 2018.

Wang, Y., Yang, C., Wen, Y., Liu, Y., and Qiao, Y. Critic-guided decision transformer for offline reinforcement learning. *arXiv preprint arXiv:2312.13716*, 2023.

Wang, Z., Novikov, A., Zolna, K., Merel, J. S., Springenberg, J. T., Reed, S. E., Shahriari, B., Siegel, N., Gulcehre, C., Heess, N., et al. Critic regularized regression. *Advances in Neural Information Processing Systems*, 33, 2020.

Wang, Z., Hunt, J. J., and Zhou, M. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Wu, Y.-H., Wang, X., and Hamaya, M. Elastic decision transformer. *arXiv preprint arXiv:2307.02484*, 2023.

Yamagata, T., Khalil, A., and Santos-Rodriguez, R. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl. In *International Conference on Machine Learning*, pp. 38989–39007. PMLR, 2023.

Yang, M., Schuurmans, D., Abbeel, P., and Nachum, O. Dichotomy of control: Separating what you can control from what you cannot. *arXiv preprint arXiv:2210.13435*, 2022.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

A. Proofs

A.1. Proof of Theorem 3.1

First we introduce the following Lemma, which is motivated by the work of Brandfonbrener et al. (2022) on return-conditioned supervised learning (RCSL).

Lemma A.1. *(Brandfonbrener et al., 2022) Consider an MDP, behavior β, and conditioning function f. Assume the following:*

1. *Return coverage: $g(\tau) = \sum_{t=1}^{\mathcal{H}} r_t$, $P_\beta(g = f(\mathbf{s}_1)|\mathbf{s}_1) \geq \alpha_f$ for all initial states $\mathbf{s}_1$.*
2. *Near determinism: $P(r \neq \mathcal{R}(\mathbf{s}, \mathbf{a})$ or $\mathbf{s}' \neq \mathcal{T}(\mathbf{s}, \mathbf{a})|\mathbf{s}, \mathbf{a}) \leq \epsilon$ at all $\mathbf{s}, \mathbf{a}$ for some functions $\mathcal{R}$ and $\mathcal{T}$. Note that this does not constrain the stochasticity of the initial state.*
3. *Consistency of f: $f(\mathbf{s}) = f(\mathbf{s}') + r$ for all $\mathbf{s}$.*[3]

Let $J(\pi) = \mathbb{E}_{\tau\sim\pi}[g(\tau)]$, then

$$\mathbb{E}_{\mathbf{s}_1}[f(\mathbf{s}_1)] - J(\pi_f^{RCSL}) \leq \epsilon \left(\frac{1}{\alpha_f} + 2\right) \mathcal{H}^2. \tag{7}$$

Using the above lemma, we can prove the Theorem 3.1.

Proof. Considering the offline dataset collected by the behavior policy β, we choose the condition function f as $f(\mathbf{s}_1) = \sum_{r_{1:\mathcal{H}}\sim\pi_\beta(\mathbf{s}_1)} r$ and plug it into the left part of Equation 7, we can see:

$$\begin{aligned}
\mathbb{E}_{\mathbf{s}_1}[f(\mathbf{s}_1)] - J(\pi_f^{RCSL}) &= \mathbb{E}_{\mathbf{s}_1}[\sum_{r_{1:\mathcal{H}}\sim\pi_\beta(\mathbf{s}_1)} r] - J(\pi_f^{RCSL}) && (8)\\
&= \mathbb{E}_{\tau\sim\pi_\beta}[\sum_{t=1}^{\mathcal{H}} r_t] - J(\pi_f^{RCSL}) && (9)\\
&= \mathbb{E}_{\tau\sim\pi_\beta}[g(\tau)] - J(\pi_f^{RCSL}) && (10)
\end{aligned}$$

Then consider the reward-to-go $\hat{r}_t = \sum_{i=1}^{\mathcal{H}} r_i$ defined in the Equation 1, it is obvious that the condition function $\hat{r}_t$ satisfies the requirement about the consistence of conditioning function, which we could get the following Equation:

$$\mathbb{E}_{\mathbf{s}_1}[f(\mathbf{s}_1)] - J(\pi_f^{RCSL}) = \mathbb{E}_{\tau\sim\pi_\beta}[g(\tau)] - \mathbb{E}_{\tau\sim\pi_f}[g(\tau)] \leq \epsilon(\frac{1}{\alpha_f} + 2)\mathcal{H}^2 \tag{11}$$

Combining this with Lemma A.1 yields the result.

□

A.2. Proof of Theorem 3.2

Motivated by the proof in Hu et al. (2023c), we first give some lemmas to help the proof of Theorem 3.2.

We consider a k-armed one-step decision-making problem. Let Δ be a k-dimensional simplex and $\boldsymbol{q} = (q(1), \ldots, q(k)) \in \mathbb{R}^k$ be the reward vector. The final optimization considers:

$$\max_{\pi\in\Delta} \pi \cdot \boldsymbol{q} + \tau\mathbb{H}(\pi). \tag{12}$$

The next result characterizes the solution of this problem (Lemma 4 of Nachum et al. (2017)).

[3]Note this can be exactly enforced (as in prior work) by augmenting the state space to include the cumulative reward observed so far.

Lemma A.2. *(Nachum et al., 2017) For $\tau > 0$, let*

$$F_\tau(\boldsymbol{q}) = \tau \log \sum_a e^{q(a)/\tau}, \quad f_\tau(\boldsymbol{q}) = \frac{e^{\boldsymbol{q}/\tau}}{\sum_a e^{q(a)/\tau}} = e^{\frac{\boldsymbol{q} - F_\tau(\boldsymbol{q})}{\tau}}. \tag{13}$$

Then there is

$$F_\tau(\boldsymbol{q}) = \max_{\pi \in \Delta} \pi \cdot \boldsymbol{q} + \tau \mathbb{H}(\pi) = f_\tau(\boldsymbol{q}) \cdot \boldsymbol{q} + \tau \mathbb{H}(f_\tau(\boldsymbol{q})). \tag{14}$$

The second result provides the error decomposition when applying the Politex algorithm to compute an optimal policy, as adopted from Abbasi-Yadkori et al. (2019).

Lemma A.3. *(Hu et al., 2023c) Let π_0 be the uniform policy and consider running the following iterative algorithm on a MDP for $t \geq 0$,*

$$\pi_{t+1}(\mathbf{a}|\mathbf{s}) \propto \pi_t(\mathbf{a}|\mathbf{s}) \exp\left(\frac{q^{\pi_t}(\mathbf{a}|\mathbf{s})}{\tau}\right), \tag{15}$$

Then

$$v^*(\mathbf{s}) - v^{\pi_t}(\mathbf{s}) \leq \frac{1}{(1-\gamma)^2} \sqrt{\frac{2 \log |\mathcal{A}|}{t}}. \tag{16}$$

Using the above lemmas, we can prove the Theorem 3.2.

Proof. First recall the in-sample optimality equation

$$q^*_{\pi_\beta}(\mathbf{s}, \mathbf{a}) = \mathcal{R}(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim \mathcal{T}(\cdot|\mathbf{s},\mathbf{a})} \left[\max_{\mathbf{a}': \pi_\beta(\mathbf{a}'|\mathbf{s}') > 0} q^*_{\pi_\beta}(\mathbf{s}', \mathbf{a}') \right], \tag{17}$$

which could be viewed as the optimal value of a MDP $M_\mathcal{D}$ covered by the behavior policy π_β, where $M_\mathcal{D}$ only contains transitions starting with $(\mathbf{s}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}$ such that $\pi_\beta(\mathbf{a}|\mathbf{s}) > 0$. Then the result can be proved by two steps. First, the QT algorithm will never consider actions such that $\pi_\beta(\mathbf{a}|\mathbf{s}) = 0$. This is directly implied by Lemma A.2. Second, we apply Lemma A.3 to show the error bound of using QT on $M_\mathcal{D}$, which implies that $V^{\pi^*}(\mathbf{s}) \geq V^\beta(\mathbf{s})$. This finishes the proof.

□

B. Implementation Details

Conditional Transformer Policy. We build our policy as a Transformer-based model, which is based on minGPT open-source code [4]. The detailed model parameters are in Table 5.

Q networks. We build two Q networks with the same MLP setting as our diffusion policy, which has 3-layer MLPs with Mish activations and 256 hidden units for all networks.

We use the Adam (Kingma & Ba, 2014) optimizer for the training of both Conditional Transformer Policy and Q networks.

C. Hyper-parameters

For QT, we consider two hyper-parameters in total: Q-value regularization weight η and gradient normalization. For the Q-value regularization weight η, we consider values according to the characteristics of different domains, and we also conduct simple ablations to investigate how to choose the value. As indicated in Equation 5, η is a critical hyper-parameter that balances policy regularization and policy improvement losses. The walker2d-medium-replay dataset, in both dense and sparse reward scenarios, is selected for benchmarking. Table 6 displays the outcomes, illustrating QT's sensitivity to η selection, with varying values yielding significantly different performances. A larger η enhances performance when the Q-value is accurately estimated within the dataset. Conversely, in scenarios like sparse rewards where Q-value estimation is challenging, a smaller η proves more efficacious. For the gradient normalization, we consider values in the grid $\{5.0, 9.0, 15.0, 20.0\}$. Based on these considerations, we provide our hyper-parameter setting in Table 7.

[4] https://github.com/karpathy/minGPT

Table 5. Hyper-parameters of QT in our experiments.

Parameter	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	3.0e-4

Table 6. Ablation on the role of the hyper-parameter η. Average and standard deviation scores are reported over 3 seeds for the walker2d-medium-replay task.

η	0.01	0.1	1	2	3
dense	88.0 ± 0.4	89.2 ± 1.0	95.4 ± 0.5	98.5 ± 1.1	98.4 ± 0.4
sparse	78.5 ± 2.1	72.3 ± 0.3	7.0 ± 4.6	8.5 ± 2.5	10.6 ± 6.1

Table 7. Hyper-parameter settings of all selected tasks.

Tasks	η	**grad norm**	**Tasks**	η	**grad norm**
halfcheetah-medium-expert-v2	2.5	15.0	pen-human-v1	0.1	9.0
hopper-medium-expert-v2	1.0	9.0	hammer-human-v1	0.1	5.0
walker2d-medium-expert-v2	2.0	5.0	door-human-v1	0.005	9.0
halfcheetah-medium-v2	5.0	15.0	pen-cloned-v1	0.1	9.0
hopper-medium-v2	1.0	9.0	hammer-cloned-v1	0.01	9.0
walker2d-medium-v2	2.0	5.0	door-cloned-v1	0.001	9.0
halfcheetah-medium-replay-v2	5.0	15.0	kitchen-complete-v0	0.005	9.0
hopper-medium-replay-v2	3.0	9.0	kitchen-partial-v0	0.01	9.0
walker2d-medium-replay-v2	2.0	5.0	-	-	-
maze2d-open-v0	0.01	9.0	maze2d-open-dense-v0	0.01	9.0
maze2d-umaze-v1	5.0	20.0	maze2d-umaze-dense-v1	3.0	5.0
maze2d-medium-v1	5.0	9.0	maze2d-medium-dense-v1	5.0	9.0
maze2d-large-v1	4.0	9.0	maze2d-large-dense-v1	4.0	9.0
antmaze-umaze-v0	0.05	9.0	antmaze-medium-diverse-v0	0.01	9.0
antmaze-umaze-diverse-v0	0.01	9.0	antmaze-large-diverse-v0	0.005	9.0

D. Further Discussions

D.1. Performance of QT in the Atari Environment

Recognizing the importance of discrete action domains in RL, we expand our investigation to include Atari games, a domain characterized by its high-dimensional visual inputs and the delayed reward challenge. We benchmark our QT method against established baselines that are evaluated in the DT method, normalizing scores where 100 represents a professional gamer's score and 0 denotes a random policy. As detailed in Table 8, our findings demonstrate that QT consistently achieves competitive performance, affirming its efficacy in discrete action domains.

Table 8. Results for 1% DQN-replay Atari datasets. We evaluate the performance of QT on four Atari games using three different seeds, and report the mean and variance of the results. The best mean scores are highlighted in bold.

Game	CQL	QR-DQN	REM	BC	DT	**QT**
Breakout	211.1	17.1	8.9	138.9 ± 61.7	267.5 ± 97.5	$\mathbf{423.9 \pm 87.2}$
Qbert	**104.2**	0	0	17.3 ± 14.7	15.4 ± 11.4	46.7 ± 13.3
Pong	**111.9**	18	0.5	85.2 ± 20.0	106.1 ± 8.1	108.3 ± 2.0
Seaquest	1.7	0.4	0.7	2.1 ± 0.3	2.5 ± 0.4	$\mathbf{4.0 \pm 0.3}$
Average	107.2	8.9	2.5	69.9	97.9	**145.7**

Table 9. Ablation on the conditional action generation. Average and standard deviation scores are reported over 3 seeds for the walker2d-medium-replay task. QT* indicates that only Q-value regularization is included in the training stage.

RTG	1000	2000	3000	4000	5000	Infer with Q-value function
QT*	51.0 ± 1.0	68.6 ± 0.7	95.3 ± 1.1	96.3 ± 0.4	97.2 ± 0.2	98.5 ± 1.1
DT	32.4 ± 1.2	58.8 ± 0.5	75.7 ± 0.6	79.4 ± 2.0	77.0 ± 0.6	87.6 ± 1.1

D.2. Conditional Action Generation

In pure DT approaches, the generation of diverse actions is conditioned on varying RTG values due to its trajectory-level modeling. While this method offers diversity, it faces the challenge of unmatched RTG values, requiring significant human effort to identify the optimal RTG for each scenario. Our QT method strategically avoids the manual selection of RTG values, which often relies heavily on prior knowledge and can be labor-intensive, streamlining the learning process and reducing dependency on manual intervention.

Specifically, QT addresses these challenges by integrating a Q-value maximization step within the training phase, guiding the CSM policy toward generating actions aligned with optimal return objectives. Just as Table 9 shows, this adjustment enhances the policy's efficacy and reduces the reliance on precise RTG selection within a certain range, providing a more efficient approach to action generation. However, QT may still encounter difficulties when there is a significant deviation between the selected RTG and the optimal trajectory. Despite this, the QT framework incorporates a Q-value function during the inference stage, offering a dynamic and adaptive strategy to ascertain optimal actions, thus augmenting the method's practicality and reducing the need for extensive manual calibration.

D.3. Differences Between QT and Other Q-Learning Methods

As delineated in Section 2.2, QDT (Yamagata et al., 2023) takes the first attempt to combine the CSM with Q-learning by learning a conservative value function to relable the RTG tokens in the dataset, keeping other components aligned with DT. However, such adaptations essentially constitute simple data augmentation, incorporating "stitched" trajectories into the training dataset but continuing to encounter unmatched RTG values during inference due to trajectory-level modeling.

Conversely, the Q-Transformer (Chebotar et al., 2023) introduces a nuanced utilization of the transformer architecture to refine the learning of the Q-value function. It achieves this through action discretization, coupled with the novel application of a conservative regularizer. This regularizer is specifically designed to constrain out-of-distribution Q-values, ensuring their proximity to the minimal achievable cumulative rewards. However, the Q-Transformer still remains within the purview of traditional Q-learning methodologies, albeit with a significant enhancement in feature representation capabilities through the adoption of transformer architecture.

For a more granular comparison, Table 10 elucidates the key distinctions among these methods.

D.4. Sparse Reward Setting

We explore the performance of the QT in environments with varying reward densities, specifically focusing on the maze2d and MuJoCo Gym tasks. Our findings indicate an inconsistency: in sparse settings of maze2d-medium and maze2d-large environments, QT outperforms compared to denser reward configurations, contrary to the observed trend in MuJoCo tasks.

Table 10. Detailed comparison of QDT, QT, and Q-Transformer.

Aspect	QDT	QT	Q-Transformer
Training dataset	Augmented with relabeled RTG tokens	Utilizes the original dataset	Utilizes the original dataset
Training loss	MSE Loss for continuous actions	MSE Loss for continuous actions, supplemented with Q-value function maximization	TD error coupled with conservative regularization
Hindsight info	Individual Return-to-Go values	A set of candidate Return-to-Go values	Does not utilize hindsight information
Inference	Relies on the transformer's output	Leverages the transformer output with a selection mechanism from the learned Q-value function	Selects from the entire action space through the maximization of the learned Q-value function

Table 11. Comparison of MuJoCo Gym and Maze2D Environments. The table shows the action dimension, state (observation) dimension, and average episode length over the top 5% returns in the dataset.

Environment	Action Dim	State Dim	Good Episode Average Length
hopper	3	11	708.2
halfcheetah	6	17	1000.0
walker2d	6	17	996.7
maze2d-open	2	4	49.8
maze2d-umaze	2	4	128.6
maze2d-medium	2	4	224.1
maze2d-large	2	4	314.6

A potential explanation for this discrepancy lies in the fundamental differences between these environments. Maze2d environments, characterized by their simplicity and shorter episode lengths, contrast with the MuJoCo tasks, which feature higher action/state dimensions and longer episode durations, as detailed in Table 11.

Another potential explanation is the reward structure in the maze2D-dense environments. In these settings, rewards are based on the negative exponentiated distance to the target, potentially inflating the values for 'failure' trajectories that approximate the target yet encounter obstacles. Our method, designed to sample high-value actions while adhering to the behavior policy, may inadvertently prioritize these 'false' high-value actions, leading to suboptimal performance compared to sparse settings where high-value actions are unequivocally associated with reaching the target. Conversely, in environments like open and umaze, where obstacles are absent, QT demonstrates superior performance in dense settings, supporting this hypothesis.

D.5. How QT Improves Stitching Ability

While our theoretical exposition offers a robust motivation, which posits that the Q-value module serves as a pivotal mechanism for policy improvement, the assertion that QT enhances the stitching capability is primarily evidenced through empirical studies. In one word, the integration of Q-value regularization with DT addresses the alignment issues inherent in pure CSM approaches to enhance the model's ability to stitch together optimal actions, thereby improving the overall effectiveness and robustness of the policy learned from offline data.

In pure CSM models, the RTG token significantly influences the learning process by providing a trajectory-level perspective. However, this trajectory-centric approach can lead to potential misalignments between the RTG values and the current state-action pairs during inference, potentially leading to suboptimal decision-making (Wang et al., 2023). To address this concern and enhance the alignment between learning and inference, we integrate the Q-value function, which offers

a granular, state-action specific estimation of future returns. This integration allows for a more dynamic and responsive decision-making process during training and inference, where actions are selected based on their immediate value rather than a predetermined trajectory, and the learning and inference processes are aligned through the learned Q-value function.

During the learning phase, the model is trained to select actions that minimize the combined loss (as outlined in Equation 5), which includes components from both the CSM and Q-value paradigms. This process ensures that the policy is grounded in the distribution of the training dataset while also being attuned to the optimal action values as estimated by the Q-value module. During inference, the model leverages the learned Q-value function to make decisions. Instead of relying on RTG tokens, the model evaluates a set of candidate actions generated based on various RTG values and selects the one with the highest Q-value. This approach ensures that the decision-making process is informed by both the trajectory modeling insights from the CSM component and the optimal action value estimation from the Q-value component.

Our ablation studies, meticulously documented in Table 2, provide empirical substantiation for this methodology. When inference relies on the learned Q-value function (Exp 4), it surpasses the performance of a purely CSM-based method (Exp 3), validating the phenomenon of unmatched RTG value in the trajectory-level modeling. Additionally, we have conducted further ablation studies that vary RTG tokens within the context of pure CSM models in Table 9. These studies are designed to rigorously examine the phenomenon of RTG value misalignment and its impact on the model's performance. Moreover, the integration of the Q-value module throughout both the learning and inference phases aligns the learning objectives with the inference dynamics, which fosters a more robust and effective decision-making framework (Exp. 7 in Table 2).

D.6. How QT Addresses Overfitting of the Q-Value Function

In the training of our Q-value functions, the expected Q-value ($\hat{Q}_m$ in Equation 4) is derived from the n-step Bellman equation. The action $\mathbf{a}_t$ is selected according to the target policy $\pi_{\theta'}$, generated by the CSM models. This design ensures that the actions produced by the CSM models predominantly align with the distribution observed in the training dataset (with small η), thus reducing the risk of overestimating Q-values for out-of-distribution actions. What's more, during the inference, the interplay between the candidate actions generated by the multiple RTG tokens and the Q-value function's guidance facilitates a more nuanced and effective action selection process, avoiding the pitfalls of direct Q-value maximization.

It is imperative to note that our policy derivation is distinct from traditional Q-learning methodologies. Our policy emerges from the CSM models other than the Q-value function, primarily governed by the MSE loss delineated in Equation 2. Here, the Q-value function serves as a component for policy enhancement, with its influence on the final policy modulated by the hyper-parameter η. In scenarios where data is exceptionally sparse or noisy, which complicates accurate Q-value estimation, modulating η can significantly mitigate the adverse effects of overfitting or incorrect Q-value approximation.

To empirically substantiate our claims, we have conducted an ablation study detailed in Table 6 above. We selected the walker2d-medium-replay dataset in both dense and sparse reward settings. The results demonstrate that in environments conducive to accurate Q-value estimation (dense reward scenario, where the performance of CQL is 77.2), a higher η enhances performance. Conversely, in settings where Q-value estimation is challenging (sparse reward scenario, where the performance of CQL is 3.2 ± 1.7), an elevated η exacerbates the training process, leading to diminished performance.