TRAJECTORY-AWARE VALUE GUIDANCE WITH A CO-TRAINED CRITIC FOR OFFLINE RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline reinforcement learning (RL) faces the significant challenge of deriving optimal policies from static, often suboptimal datasets. Sequence modeling methods like Decision Transformer (DT) have shown promise, but their performance is often constrained by the quality of the best trajectories in the dataset. In contrast, value-based methods have the potential to stitch together suboptimal trajectories to achieve performance beyond that of any single trajectory. Recent work has explored integrating DT with value-based approaches to enhance trajectory stitching capabilities while avoiding distribution shift in offline RL. However, existing methods either rely on static pre-trained Q-functions, which can become misaligned with the evolving policy during training, or apply value guidance uniformly regardless of trajectory quality. We introduce Adaptive Q-Guided Decision Transformer (AQDT), an approach that dynamically co-trains the Q-function alongside DT training, creating a mutually adaptive relationship between policy and critic. AQDT adaptively balances DT learning with value-based guidance based on trajectory quality, while ensuring that the critic remains aligned with the evolving policy. On standard D4RL benchmarks, AQDT achieves the highest average normalized return among DT-based and hybrid methods, with a 4.6-point gain over the strongest baseline on locomotion tasks and strong results on sparse-reward AntMaze tasks, with the largest gains on datasets dominated by suboptimal trajectories.

1 INTRODUCTION

Offline Reinforcement Learning (RL) aims to derive effective policies from static datasets without any interaction with the environment (Levine et al., 2020). Effective offline RL algorithm can transform large static datasets into powerful decision-making tools (Levine et al., 2020). This capability is particularly valuable in safety-critical or high-cost domains such as autonomous driving and healthcare, where online data collection may be impractical, dangerous, or ethically restricted (Kim et al., 2024b; Levine et al., 2020). Despite its promise, offline RL presents significant challenges due to the absence of active exploration and the susceptibility to distributional shift; the agent must learn exclusively from a fixed dataset $\mathcal{D}$ and lacks the ability to collect additional transitions to discover high-return regions of the state-action space (Levine et al., 2020; Fujimoto et al., 2019).

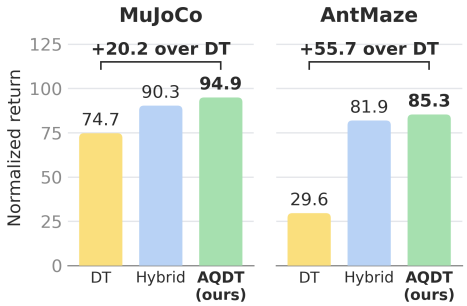


Figure 1: Mean normalized return on MuJoCo and AntMaze environments. The scores for DT, hybrid methods, and AQDT(ours) are reported. Full performance details are provided in Section 6. For hybrid methods, the best-performing method in each domain is shown (QCS for MuJoCo, QCS-G for AntMaze).

Return-Conditioned Supervised Learning (RCSL) has recently gained traction as a promising approach in offline RL (Srivastava et al., 2019; Janner et al., 2021; Chen et al., 2021). It frames policy learning as a conditional prediction task: the agent predicts actions based on its current state (or

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

history) and a target return. Building on this idea, Conditional Sequence Modeling (CSM) methods, most notably the Decision Transformer (DT), have demonstrated competitive performance by reformulating RL as a sequence modeling problem (Chen et al., 2021; Wu et al., 2023; Wang et al., 2024). DT leverages transformer architecture originally developed for natural language processing to model trajectories (Chen et al., 2021). Rather than directly optimizing cumulative reward through value functions or policy gradients, DT learns to imitate trajectories that achieve specific return targets. The DT paradigm relaxes the Markov Decision Process (MDP) assumption by conditioning on extended sequences of prior interactions, rather than solely on the current state. This enables the DT model to capture long-range temporal dependencies while maintaining alignment with the behavior policy, thereby reducing the risk of selecting out-of-distribution actions. Furthermore, DT avoid the stability issues associated with bootstrapping in value-based RL, offering a more robust solution for long-term credit assignment (Chen et al., 2021; Sutton & Barto, 1998). While DT has demonstrated competitive performance in offline RL, it notably struggles with trajectory stitching; the ability to assemble an optimal trajectory from suboptimal trajectory segments (Wu et al., 2023; Kim et al., 2024b; Hu et al., 2024). Consequently, their effectiveness is limited to the optimal trajectories in the dataset. In contrast, value-based methods (Kumar et al., 2020; Kostrikov et al., 2021; Fujimoto & Gu, 2021) learn value functions that guide policy optimization by selecting actions that maximize expected future rewards. These approaches evaluate actions at each timestep independently by estimating Q-values (expected cumulative returns for state or state-action pairs) regardless of the specific trajectories from which they were sampled. This enables value-based methods to "stitch together" high-return behavior by combining segments from different trajectories, including those originating from highly suboptimal data (Wang et al., 2024; Hu et al., 2024). This property is especially advantageous in stochastic environments, where the observed returns may be noisy or misleading; by focusing on expected values rather than raw outcomes, value-based methods can better discern and generalize the underlying quality of actions (Wang et al., 2024). However, value-based methods are prone to unstable learning behaviors, particularly in long-horizon and sparse-reward scenarios (Yamagata et al., 2023; Levine et al., 2020).

Recent methods have attempted to combine the strengths of both paradigms by integrating value-based learning with CSM to enhance stitching capabilities (Hu et al., 2024; Kim et al., 2024b; Wang et al., 2024). While these approaches show promise, they have complementary limitations. EDT and CGDT use value estimates primarily as conditioning signals, without fully exploiting their stitching ability. QCS adapts Q-guidance to trajectory quality but relies on a static, pre-trained Q-function that can become misaligned as the policy evolves. QT co-trains its critic with the policy but applies Q-guidance uniformly, regardless of trajectory quality. As a result, developing a framework that effectively unites the strengths of both paradigms remains a significant challenge. This motivates our work on Adaptive Q-Guided Decision Transformer (AQDT), which introduces a dynamic and trajectory-aware integration of Q-values into the training process of DT. Our approach ensures that Q-values are continually adjusted to reflect the current policy distribution, thereby improving policy consistency and improving trajectory stitching(Figure 1).

- We present AQDT, a novel offline RL algorithm that enhances Decision Transformer with effective trajectory stitching capabilities.
- We combine trajectory-aware Q-guidance with a critic that is pre-trained in-sample and then co-trained with the Decision Transformer, keeping value estimates aligned with the evolving policy while concentrating value-based correction on low-return trajectories.
- We demonstrate that AQDT achieves the highest average performance among DT-based and hybrid methods on D4RL, achieving better performance in both continuous control and long-horizon sparse-reward settings.

2 OFFLINE RL

Markov Decision Process: We formulate the RL problem within the Markov Decision Process (MDP) framework (Puterman, 1990), defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, r, \gamma)$; where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, and $T(s_{t+1} \mid s_t, a_t)$ the transition dynamics, which define the conditional probability of transitioning to state s_{t+1} given the current state s_t and action a_t. The reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ assigns a scalar reward for each state-action pair, and $\gamma \in [0, 1)$ is a discount factor that controls the importance of future rewards. The agent, operating within this

framework, observes the state s_t at each time step t, selects an action a_t, and receives a reward r_t. The RL objective is to find a policy $\pi(a|s)$ that maximizes the expected return (cumulative rewards) $J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$.

In offline RL, the learning algorithm is provided with a fixed dataset of transitions, $\mathcal{D} = \{(s, a, r, s^{'})\}$, collected by an unknown behavior policy π_β. The goal remains to optimize the same objective $J(\pi)$ by learning a policy $\pi(a \mid s)$ from $\mathcal{D}$ that maximizes the expected return when deployed in the environment (Levine et al., 2020). Offline RL has led to the development of numerous approaches designed to address the inherent challenges of learning from fixed datasets (Levine et al., 2020; Wang et al., 2024). These methods can broadly be categorized into *value-based* and *value-free* approaches.

Value-based methods: focus on learning a value function, typically the state-action value function $Q(s, a)$. Once a Q-function is obtained, a policy can be derived by acting greedily with respect to Q. The core learning objective in these methods is to minimize the Bellman error:

$$\mathcal{L}_{\text{Bellman}} = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q(s,a) - \left(r + \gamma\, \mathbb{E}_{a'\sim\pi(\cdot|s')}[Q(s',a')]\right)\right)^2\right]. \tag{1}$$

A critical issue in this setting is the distributional shift between the data distribution induced by π_β and the distribution induced by the learned policy π. Since the Bellman target requires sampling a' from π, many of these actions can be out-of-distribution (OOD) relative to $\mathcal{D}$, leading to unreliable estimates of $Q(s', a')$. This issue, known as extrapolation error (Fujimoto et al., 2019), can destabilize training and degrade policy performance. To mitigate these challenges, various value-based offline RL methods have been proposed. Two widely used strategies are policy constraints and regularization (Wang et al., 2024; Fujimoto et al., 2019; Kumar et al., 2019). Policy Constraints methods enforce closeness between the learned policy and the behavior policy (Fujimoto et al., 2019; Kumar et al., 2019). In contrast, regularization techniques such as Conservative Q-Learning (Kumar et al., 2020) and Implicit Q-Learning (Kostrikov et al., 2021) incorporate penalty terms that penalize large Q-value estimations and shape policy behavior without explicitly modeling π_β.

Value-free methods: these methods directly optimize policies without explicit value estimation, and therefore avoid extrapolation errors entirely. A prominent example is the Decision Transformer (DT) (Chen et al., 2021), which re-frames reinforcement learning as a sequence modeling task using a Transformer architecture (Vaswani et al., 2017). DT encodes sub-trajectories of length H as sequences of the $\langle$RTG, state, action$\rangle$ triplets, and is trained to predict the next action given the preceding context. By using extended historical context rather than relying solely on the current state, as a result, it is better suited for modeling long-horizon dependencies and avoids the instability introduced by bootstrapped value estimates (Srivastava et al., 2019; Chen et al., 2021).

3 Related Works

Trajectory Stitching: DT demonstrates strong planning capabilities, but it is primarily limited to mimicking observed behaviors, and it lacks the ability to stitch suboptimal trajectory segments into improved behavior. As a result, it struggles in scenarios where optimal behavior must be synthesized from fragmented demonstrations. Figure 2 illustrates the concept of trajectory stitching. (a) shows two suboptimal trajectories from the dataset: the blue trajectory reaching the goal with a total return of 9, and the green one with an overall return of 6. Starting from s_1 the optimal path following DT policy will be the blue trajectory with higher overall return.

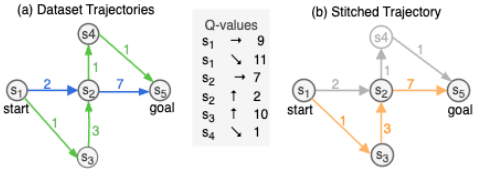


Figure 2: **Trajectory Stitching.** (a) Two suboptimal trajectories from the dataset: blue (return 9) and green (return 6). (b) A stitched trajectory (unseen in the dataset) combines segments from both trajectories to form a new trajectory (orange) with a higher total return.

However, value-based methods operate differently by choosing actions with the highest estimated return (Q-value) at each decision point. As shown in the Q-values table, starting from s_1, the $\searrow$ action leading to state s_3 would be chosen because it has a higher Q-value (11) compared to the $\rightarrow$ action with Q-value 9. Similarly, at state s_2 the value-based policy will choose to move right toward

s_5 (Q-value 7) rather than moving up toward s_4 (Q-value 2), thereby completing a path to the goal with a higher return than either of the original trajectories. Figure 2-(b) demonstrates this stitched trajectory that combines segments from both original trajectories and achieves a trajectory (unseen in the dataset) with higher return of 11.

This example indicates that while Decision Transformers mitigate distributional shift by remaining close to the behavior policy, this conservative approach often restricts their ability to improve beyond the best observed trajectories in the dataset (Kim et al., 2024b). This limitation becomes particularly apparent in datasets containing only suboptimal demonstrations. In contrast, value-based methods can generalize across the dataset by stitching together suboptimal segments into higher-return trajectories. However, this flexibility comes with a greater risk of extrapolation error when the method attempts to estimate Q-values for combinations of state-action pairs that are unlikely under the behavior policy (Kostrikov et al., 2021; Janner et al., 2021). The effective approach would balance the conservative nature of DTs with the generalization capabilities of value-based methods, motivating the hybrid approaches that incorporate both paradigms. Recent studies have explored the stitching limitation in DT (Wu et al., 2023; Kim et al., 2024b; Wang et al., 2024). Elastic Decision Transformer (EDT) dynamically adjusts context lengths during inference, enabling better capture of decision-relevant history according to the learned value estimator. This flexibility allows the model to focus on the most important parts of past experience when making decisions (Wu et al., 2023). Critic-Guided Decision Transformer (CGDT) enhances the DT with a learned critic that aligns target returns with expected returns. This approach improves performance in stochastic environments where standard DTs struggle (Wang et al., 2024). Methods such as EDT and CGDT underutilize Q-functions by treating them merely as conditioning signals rather than leveraging their ability to stitch together high-return trajectories. In contrast, QT integrates Q-learning principles into the DT framework by introducing a Q-value regularization term. This guides the policy toward actions with higher estimated returns while maintaining reasonable proximity to the original behavior distribution (Hu et al., 2024). QCS further refines this approach by adaptively incorporating Q-function guidance based on trajectory quality. It focuses corrections specifically on low-return sequences where standard DTs typically struggle, resulting in more effective learning from suboptimal demonstrations (Kim et al., 2024b). However, QCS relies on a pre-trained Q-function that remains fixed while the DT policy evolves during training. Because this Q-function continues to evaluate actions based on the original data distribution rather than the current policy's behavior, its value estimates become unreliable as the policy moves beyond the behavior distribution, leading to poor action selection and restricting stitching, particularly in complex or stochastic environments where accurate value estimation is crucial (Kumar et al., 2020; Levine et al., 2020). QT avoids this misalignment by training its critic jointly with the policy, but applies the same Q-guidance strength to all trajectories, including near-optimal ones that require little correction. Our method, AQDT, combines the strengths of both approaches: like QCS, it scales Q-guidance according to trajectory quality, and like QT, it keeps the critic aligned with the evolving policy through joint training. Unlike QT, AQDT initializes the critic from in-sample critic pre-training (Kostrikov et al., 2021) before co-training, providing a stable starting point for value estimates. This combination enables more consistent value supervision and more effective stitching of suboptimal trajectories into high-return behavior.

4 Adaptive Q-Guided Decision Transformer

In this section we present the *Adaptive Q-Guided Decision Transformer* (AQDT), which enhances the DT with a dynamically updated Q-function.

4.1 Conditional Transformer-based Policy

The AQDT policy architecture is based on the Decision Transformer framework, modeling reinforcement learning as a sequence prediction task. For a trajectory of length T, at each timestep t, the model processes a context window of length H, which includes return-to-go values, states, and actions:$\tau^{<t} = (\hat{R}_{t-H+1}, s_{t-H+1}, a_{t-H+1}, \ldots, \hat{R}_{t-1}, s_{t-1}, a_{t-1}, \hat{R}_t, s_t)$, where $\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$ denotes the return-to-go, s_t is the state at timestep t, and a_t is the corresponding action. The transformer predicts the actions as $\hat{a}_{t'} = \pi_\theta(\tau_{t'}^{<t})_{t'=t-H+1}^{t}$, where θ denotes the policy parameters. The supervised learning objective minimizes the mean squared error between the predicted and actual

actions:

$$\mathcal{L}_{\mathrm{P}}(\theta) = \mathbb{E}_{\tau^{<t}\sim\mathcal{D}}\big[\frac{1}{H}\sum_{t'=t-H+1}^{t}\big|a_{t'} - \hat{a}_{t'}\big|^2\big] \tag{2}$$

4.2 Dynamic Adaptation of the Q-Function

A key feature of AQDT is that it does not rely on a fixed Q-function. Following QCS (Kim et al., 2024b), both critics are initialized from a Q-function pre-trained with IQL (Kostrikov et al., 2021), which provides in-sample value estimates at the start of training; the critics are then continually updated alongside the policy using the n-step objective below. We implement two critic networks, Q_{ϕ_1} and Q_{ϕ_2}, following the Double Q-learning paradigm (Van Hasselt et al., 2016), each paired with a target network $Q_{\phi_1'}$ and $Q_{\phi_2'}$. We also maintain a target policy network, $\pi_{\theta'}$, to stabilize updates. To better capture temporal dependencies inherent in trajectory data, we use the n-step Bellman equation rather than the standard 1-step approach which has shown to be more effective in sequence-based learning (Sutton & Barto, 1998; Hu et al., 2024). The critic loss for $i \in \{1, 2\}$ is:

$$\mathbb{E}_{\tau^{<t}\sim\mathcal{D}}\left[\sum_{t'=t-H+1}^{t-1}\Big(\hat{Q}_{t'} - Q_{\phi_i}(s_{t'}, a_{t'})\Big)^2\right], \tag{3}$$

where the n-step target $\hat{Q}_{t'}$ is defined as:$\sum_{j=t'}^{t-1}\gamma^{j-t'}r_j + \gamma^{t-t'}\min_{i=1,2}Q_{\phi_i'}(s_t, \pi_{\theta'}(\tau_t^{<t}))$. Here, γ is the discount factor, and the minimum over the two target critics reduces overestimation bias.

4.3 Incorporating Q-Guided Policy Improvement

DT performs well when trained on optimal trajectories, however its effectiveness decreases when the dataset includes suboptimal demonstrations. We augment the policy with a Q-guided policy improvement mechanism. Specifically, we incorporate a *Q-guide* loss term, which encourages the policy to select actions with higher estimated values. For each state s_t, we evaluate the critic (Q_ϕ) on the policy's output action and define the Q-guided loss as:

$$\mathcal{L}_{\mathrm{Q-guide}}(\theta, \phi) = \mathbb{E}_{\tau^{<t}\sim\mathcal{D}}\big[\frac{1}{H}\sum_{t'=t-H+1}^{t} -Q_\phi\big(s_{t'}, \pi_\theta(\tau_{t'}^{<t})\big)\big] \tag{4}$$

To balance imitation learning and value-based improvement, we apply a trajectory-aware weighting scheme similar to (Kim et al., 2024b). High-return trajectories already reflect near-optimal behavior and require minimal Q-guided correction, while low-return trajectories benefit more from value-based guidance. For a trajectory τ with cumulative return $R(\tau)$, and the maximum observed return $R_{\max}$, we define the adaptive weight:

$$\lambda(\tau) = \alpha \cdot \frac{R_{\max} - R(\tau)}{R_{\max}}, \tag{5}$$

where $R(\tau)$ is the return of the full trajectory from which $\tau^{<t}$ is sampled, $R_{\max}$ is the maximum trajectory return in the dataset, and α controls the influence of the Q-guided term. This ensures that for near-optimal trajectories ($R(\tau) \approx R_{\max}$), $\lambda(\tau)$ is small, prioritizing imitation while for suboptimal trajectories ($R(\tau) \ll R_{\max}$), $\lambda(\tau)$ is larger, amplifying Q-guided correction.

The combined policy loss is thus:

$$\mathcal{L}_{\mathrm{AQDT}}(\theta, \phi) = \mathcal{L}_{\mathrm{P}}(\theta) + \lambda(\tau) \cdot \mathcal{L}_{\mathrm{Q-guide}}(\theta, \phi), \tag{6}$$

enabling the policy to dynamically balance imitation and value-based optimization depending on trajectory quality, which is essential for effective stitching of suboptimal data into improved trajectories.

During the policy update, the critic parameters ϕ are held fixed, so gradients of $\mathcal{L}_{\mathrm{AQDT}}$ update only θ.The AQDT training process alternates between updating the critics and the policy where at each timestep the $Q_\phi(.)$ in $\mathcal{L}_{\mathrm{Q-guide}}(\theta, \phi)$ is defined as $Q_\phi(s, a) = \min_{i\in\{1,2\}}Q_{\phi_i}(s, a)$. The pseudocode for the training process is provided in Algorithm 1 and the implementation details are provided in Appendix A.

Algorithm 1 Adaptive Q-Guided Decision Transformer

Input: Sequence context length H, Offline dataset $\mathcal{D}$, Discount factor γ, Coefficient α, Coefficient ρ, Learning rate lr_{policy}, Critic learning rate lr_{critic}
Initialize: Policy network π_θ; critics Q_{ϕ_1}, Q_{ϕ_2} from IQL pre-training; target networks $\pi_{\theta'}, Q_{\phi'_1}, Q_{\phi'_2}$

```
AQDT Training
for each training iteration do
  Sample batch {τ_i} ~ D
  Sample sub-trajectory τ_i^{<t} with random timestep t
  Policy Learning
  Compute R(τ_i) and adaptive weight λ(τ_i)                    ▷ Eq.5
  Predict action: â_{t'} ← π_θ(τ_{t'}^{<t}) for t' = t − H + 1, ..., t
  Compute L_AQDT(θ, φ)                                         ▷ Eq.6
  Update θ ← θ − lr_policy ∇_θ (L_AQDT)
  Critic Tuning
  Update φ_i ← φ_i − lr_critic ∇_{φ_i} L_critic for i ∈ {1, 2}  ▷ Eq.3
  Update Target Networks
  θ' ← ρθ + (1 − ρ)θ',   φ'_i ← ρφ_i + (1 − ρ)φ'_i for i ∈ {1, 2}
end for
```

5 EXPERIMENTS

Our experiments are designed to systematically evaluate the effectiveness of AQDT in offline reinforcement learning tasks. Specifically, we aim to answer the following questions:

- Does AQDT improve performance compared to standard Decision Transformer (DT) and existing hybrid approaches? (section 6.1)
- Does AQDT demonstrate robustness across environments with varying dataset optimality (e.g., medium, medium-replay, and medium-expert datasets)? (section 6.1)
- Does AQDT improve performance compared to baselines on sparse reward and goal oriented environments? (section 6.2)
- Does AQDT improve trajectory stitching compared to baselines? (section 6.3)
- How does the static Q-function guide differ from the dynamic Q-function guidance in AQDT? (section 6.4)

5.1 EXPERIMENTAL SETUP

Environments and Datasets. We evaluate AQDT on the widely used D4RL benchmark suite (Fu et al., 2020), which provides standardized datasets for continuous control tasks. We focus on the Gym (MuJoCo) and AntMaze domains, which collectively test both dense-reward and sparse-reward learning. For the Gym-MuJoCo domain, we conducted experiments in Walker2d , Hopper and HalfCheetah environments, utilizing three datasets for each: medium, medium-replay, and medium-expert. These datasets vary in data quality, from sub-optimal (medium-replay) to near-optimal (medium-expert), and are characterized by smooth, continuous reward functions of locomotion tasks. In the AntMaze domain, which requires long-horizon planning and trajectory stitching, we evaluated performance across four datasets: umaze, umaze-diverse, medium-diverse, and large-diverse. These environments involve sparse-reward goal-reaching tasks, where the agent must navigate complex mazes of varying sizes to reach specified goals.

Baselines. We compare AQDT against three categories of methods: value-based methods, conditional sequence modeling (CSM) methods, and hybrid methods. For each category, we select state-of-the-art approaches. In particular, the hybrid methods are designed to enhance the trajectory stitching capabilities of CSM frameworks by incorporating value-based guidance into sequence modeling. The detail of the baselines are provided in Appendix B.

Evaluation Protocol. Following (Fu et al., 2020), we normalize scores to facilitate cross-environment comparisons. We report the mean normalized score over 10 evaluation episodes, averaged across 3 random seeds.

To ensure fair comparison across tasks with different reward scales, we calculate a normalized score defined as: Normalized Score $= 100 \times \frac{R_{\text{agent}} - R_{\text{random}}}{R_{\text{expert}} - R_{\text{random}}}$, where R_{agent} represents the return achieved by the policy, R_{random} is the average return of a random policy, and R_{expert} is the return of an expert policy. Under this normalization, a score of 100 corresponds to expert-level performance. We analyze the results for each specific domain separately.

6 RESULTS

6.1 MUJOCO CONTINUOUS CONTROL TASKS

As shown in Table 1, our method (AQDT) achieves the highest average score in every dataset category and the best overall average (94.9). It is the best-performing method on all medium and medium-replay tasks, and remains competitive on medium-expert tasks. Most notably, AQDT demonstrates superior performance in scenarios with suboptimal trajectories, where effective stitching capabilities are essential.

On medium datasets, AQDT surpasses the best baseline (QCS) by 5.4 points. The improvement is particularly pronounced in HalfCheetah-medium, where AQDT scores 70.32 $\pm$1.0, significantly outperforming even the strongest baselines (59). For medium-replay datasets, which contain diverse but predominantly suboptimal trajectories, AQDT's advantage becomes even more apparent. Our method averages 90.11 across these environments, compared to 83.1 for QT and 82.9 for QCS. Even in the medium-expert datasets, where demonstrations are already near-optimal and the need for trajectory stitching is minimal, AQDT remains competitive. While some baselines achieve slightly higher scores on medium-expert datasets, AQDT maintains the highest average score across all dataset categories. Notably, when evaluating all MuJoCo datasets collectively, AQDT achieves the highest overall average score, outperforming all baseline methods. This demonstrates that AQDT is not only effective in challenging suboptimal settings but also generalizes across diverse dataset optimality.

Dataset	**Value-Based Method**				**CSM**			**Hybrid Method**							**Ours**
	TD3+BC	IQL	CQL	SQL	DT	DC	RvS-R	QDT	EDT	CGDT	ACT	POR	QCS	QT	AQDT
halfcheetah-m	48.3	47.4	44.0	48.3	42.6	43.0	41.6	42.3	42.5	43.0	49.1	48.8	59.0	51.4	**70.32** $\pm$ 1.0
hopper-m	59.3	66.3	58.5	75.5	67.6	92.5	60.2	65.5	63.5	96.9	67.8	78.6	96.4	96.9	**99.09** $\pm$0.7
walker2d-m	83.7	78.3	72.5	84.2	74.0	79.2	71.7	67.1	72.8	79.1	80.9	81.1	88.2	88.8	**90.45** $\pm$0.1
Average (m)	63.8	64.0	58.3	69.3	61.4	71.6	57.8	58.3	59.6	73.0	65.9	69.5	81.2	79.0	**86.62**
halfcheetah-m-r	44.6	44.2	45.5	44.8	36.6	41.3	38.0	35.6	37.8	40.4	43.0	43.5	54.1	48.9	**60.46** $\pm$ 1.7
hopper-m-r	60.9	94.5	95.0	99.7	82.7	94.2	73.5	52.1	89.0	93.4	98.3	98.9	100.4	102.0	**103.89** $\pm$0.4
walker2d-m-r	81.8	73.9	77.2	81.2	66.6	76.6	60.6	58.2	74.8	73.1	56.1	76.6	94.1	98.5	**105.98** $\pm$3.2
Average (m-r)	62.4	70.9	72.6	75.2	62.0	70.7	57.4	48.6	67.2	69.0	65.8	73.0	82.9	83.1	**90.11**
halfcheetah-m-e	90.7	86.7	91.6	94.0	86.3	93.6	92.2	-	-	-	**96.6**	94.7	93.5	96.1	95.22 $\pm$0.5
hopper-m-e	98.0	91.5	105.4	**114.8**	107.6	110.4	101.7	-	-	-	111.5	90.0	110.2	113.4	112.9 $\pm$0.2
walker2d-m-e	110.1	109.6	108.8	110.0	108.1	109.6	106.0	-	-	-	113.5	109.1	**116.6**	112.6	115.53 $\pm$0.8
Average (m-e)	99.6	95.9	101.9	106.3	100.7	104.5	100.0	-	-	-	107.2	97.9	106.8	107.4	**107.88**
Overall Average	75.3	77.0	77.6	83.6	74.7	82.2	71.7	-	-	82.4	79.6	80.1	90.3	89.8	94.9

Table 1: *Performance comparison on MuJoCo locomotion tasks.* Normalized scores (0-100 scale) are reported, with higher values indicating better performance. Dataset abbreviations: medium (m), medium-replay (m-r), medium-expert (m-e). AQDT consistently outperforms baselines particularly on medium and medium-replay datasets where suboptimal trajectories dominate, demonstrating superior trajectory stitching capabilities. Bold values indicate best performance for each dataset. Results for AQDT are reported as mean $\pm$standard deviation across 3 seeds.

6.2 ANTMAZE NAVIGATION TASKS

The AntMaze domain presents a more challenging test of trajectory stitching capabilities, featuring sparse rewards and requiring effective long-horizon planning. As shown in Table 2, AQDT achieves an average normalized score of 85.3 across all AntMaze environments, outperforming the next best method by 3.4 points. QT exhibits strong performance in the simpler umaze and umaze-diverse

environments. In contrast, AQDT demonstrates clear advantages in the more complex medium-diverse and large-diverse datasets.

In the umaze tasks, the maze is small and a large fraction of dataset trajectories reach the goal, so the data already contains near-optimal behavior. In this regime, uniform Q-guidance as in QT is effective, and AQDT is less competitive. In contrast, in the medium-diverse and large-diverse tasks, successful behavior must be assembled from segments of different trajectories. In these settings, AQDT's trajectory-aware weighting applies stronger Q-guided corrections to unsuccessful trajectories while relying on imitation for successful ones, which is consistent with its gains on the two most challenging mazes. Note that QT results are reported on AntMaze v0 and are not directly comparable.

Table 2: *Performance comparison on AntMaze navigation tasks with sparse rewards.* Dataset abbreviations: umaze (u), umaze-diverse (u-d), medium-diverse (m-d), and large-diverse (l-d). Bold values indicate best performance for each dataset.

Dataset	**Value-Based Method**				**CSM**			**Hybrid Method**					**Ours**
	TD3+BC	IQL	CQL	SQL	DT	DC	RvS-R	RvS-G	POR	QCS-R	QCS-G	QT	AQDT
antmaze-umaze	78.6	87.5	74.0	92.2	65.6	85.0	64.4	65.4	90.6	92.7	92.5	**96.7**	84.4 ± 6.3
antmaze-umaze-diverse	71.4	62.2	84.0	74.0	51.2	78.5	70.1	60.9	71.3	72.3	82.5	**96.7**	83.3 ± 9.4
antmaze-medium-diverse	3.0	70.0	53.7	79.1	1.2	27.5	7.7	67.3	79.2	79.5	75.2	59.3	**87.8** ± 4.5
antmaze-large-diverse	0.0	47.5	14.9	52.3	0.5	12.3	3.7	36.9	73.4	70.6	77.3	53.3	**85.6** ± 11.0
Average	38.3	66.8	56.7	74.4	29.6	50.8	36.5	57.6	78.6	78.8	81.9	76.5	**85.3**

6.3 Trajectory Stitching Performance

To further demonstrate AQDT's trajectory stitching capabilities, we analyze the distribution of trajectory returns in the datasets and compare them against the average return achieved by AQDT. Figure 3 displays histograms of normalized returns for the Hopper, Walker2D, and HalfCheetah medium-replay datasets. It can be seen that Medium-replay datasets exhibit a wide spread of returns with a concentration in the low-return region, indicative of suboptimal behavior. Despite this, AQDT effectively leverages the available data, outperforming the best observed trajectories. This figure affirms AQDT's capability to generalize beyond individual demonstrations and shows its capabilities in scenarios where stitching is essential. We guide the readers to Appendix C.1 for a full analysis of AQDT stitching performance across dataset's return distributions and baselines.

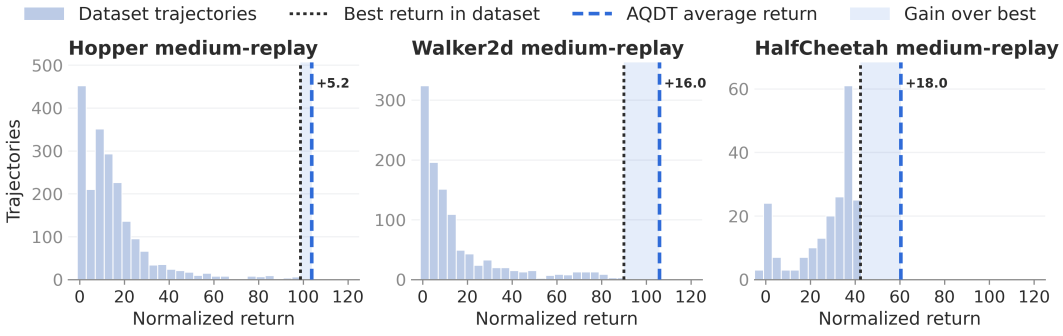


Figure 3: Histograms of normalized returns for the Hopper, Walker2D, and HalfCheetah medium-replay datasets. Each bar represents the number of trajectories corresponding to a given return level. The dotted black line indicates the highest return achieved by any trajectory in the dataset, while the dashed blue line denotes the average return obtained by AQDT. The shaded region highlights the performance improvement of AQDT over the best trajectory available in the dataset. Across all three environments, AQDT attains an average return that surpasses the highest-return trajectory contained in the dataset.

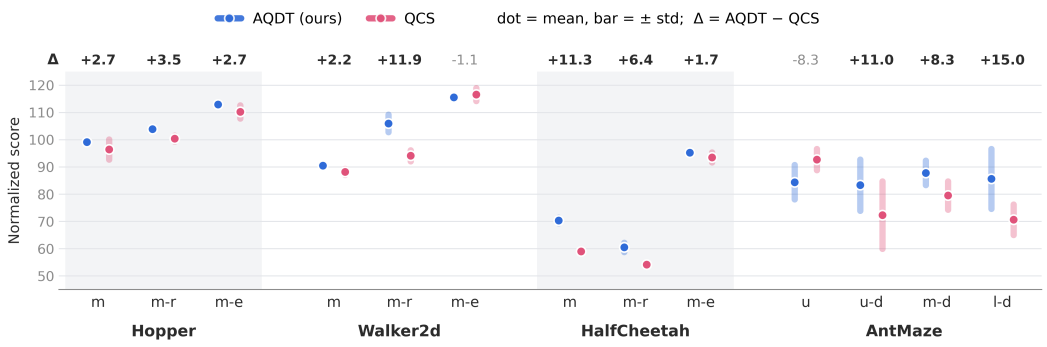


Figure 4: Comparison of AQDT (blue) and QCS (pink) in normalized score across D4RL tasks. For AntMaze, QCS-R is shown. Δ denotes AQDT minus QCS.

6.4 Comparing policies with static Q-guidance vs. dynamic Q-guidance

To evaluate the impact of dynamic critic adaptation, we compare AQDT against QCS, a recent hybrid method that leverages Q-guidance based on trajectory quality but relies on a static, pre-trained Q-function. As shown in Figure 4, while both methods leverage Q-guidance, QCS uses a static, pre-trained Q-function, whereas AQDT dynamically co-trains the Q-function alongside the policy. AQDT achieves higher mean performance in 11 of the 13 tasks. These results highlight the advantage of AQDT's policy alignment for effective trajectory stitching and better generalization.

6.5 Ablation Study

We ablate the adaptive weight $\lambda(\tau)$ by replacing it with a fixed weight matched to its expected value, so both variants apply the same total Q-guidance and differ only in where it is applied. Each variant is trained with a frozen IQL critic and with the co-trained critic on halfcheetah-medium-replay, hopper-medium-replay and hopper-medium-expert (3 seeds; DT re-run under the same protocol).

Weighting protects good data. Adaptive weighting outperforms the fixed weight in five of six settings. On hopper-medium-expert, uniform guidance pulls the policy away from near-optimal demonstrations and falls below DT (72.9 ± 6.9 frozen, 102.6 ± 6.0 co-trained, vs. 112.3 ± 0.1), while $\lambda(\tau)$ recovers it (110.2 ± 2.1 and 112.9 ± 0.2) and cuts variance from 6.0 to 0.2. On hopper-medium-replay it adds $+3.0$ and $+7.9$ points. On halfcheetah-medium-replay, where returns lie in a narrow band and $\lambda(\tau)$ is nearly constant, the variants tie (60.46 ± 1.7 vs. 61.4 ± 4.3), as expected.

Co-training is complementary. With adaptive weighting fixed, co-training the critic improves hopper-medium-expert from 110.2 to 112.9 and reduces variance from 2.1 to 0.2.

7 Conclusion

In this work, we introduced Adaptive Q-Guided Decision Transformer (AQDT), a novel offline reinforcement learning framework that dynamically integrates value-based guidance into the Decision Transformer paradigm. By co-training the Q-function alongside the policy, AQDT maintains alignment between value estimates and the evolving policy, enabling more effective trajectory stitching and robust generalization across datasets of varying quality. Extensive experiments on D4RL benchmarks demonstrate that AQDT achieves the highest average performance among DT-based and hybrid methods, with the largest gains in suboptimal and sparse-reward settings. These results highlight the potential of adaptive value-guided learning in advancing the capabilities of sequence modeling approaches in offline reinforcement learning.

Reproducibility

All source code will be made publicly available upon acceptance.

AI USE STATEMENT

Large language models were used to assist with editing the manuscript for grammar and readability and with formatting references. LLMs were also used to assist with writing code for experiment scripts and plotting. No LLM was used to design the method, design experiments, or generate data.

REFERENCES

Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

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.

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.

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.

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

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

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, pp. 12127–12135, 2024.

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.

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.

Jeonghye Kim, Suyoung Lee, Woojun Kim, and Youngchul Sung. Decision convformer: Local filtering in metaformer is sufficient for decision making. In *International Conference on Learning Representations*, 2024a.

Jeonghye Kim, Suyoung Lee, Woojun Kim, and Youngchul Sung. Adaptive q-aid for conditional supervised learning in offline reinforcement learning. *Advances in Neural Information Processing Systems*, 37:87104–87135, 2024b.

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

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in neural information processing systems*, 32, 2019.

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.

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.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

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.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yuanfu Wang, Chao Yang, Ying Wen, Yu Liu, and Yu Qiao. Critic-guided decision transformer for offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 15706–15714, 2024.

Yueh-Hua Wu, Xiaolong Wang, and Masashi Hamaya. Elastic decision transformer. *Advances in neural information processing systems*, 36:18532–18550, 2023.

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.

Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Wai Kin Chan, and Xianyuan Zhan. Offline rl with no ood actions: In-sample learning via implicit value regularization. *arXiv preprint arXiv:2303.15810*, 2023.

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*, pp. 38989–39007. PMLR, 2023.

Appendix

A Implementation details

A.1 Policy

We built our policy based on Decision Transformer. We utilized the transformer implementation based on GPT-2 with causal attention. Our implementation builds upon the publicly available codebase at (https://github.com/nikhilbarhate99/min-decision-transformer). The detailed model parameters are presented in Table 3. For the AntMaze domain, we used 10^5 gradient steps, and for the MuJoCo domains, we used 5×10^4 gradient steps for training the policy. Following QCS (Kim et al., 2024b) (QCS-R and QCS-G denote its return- and goal-conditioned variants), we set the conditioning target return-to-go (RTG) to double the maximum trajectory return in the dataset during evaluation.

A.2 Q-Function

Both Q-networks, Q_{ϕ_1} and Q_{ϕ_2}, adopt multilayer perceptron (MLP) architectures. AQDT builds on the QCS framework, which pre-trains a Q-function using IQL (Kostrikov et al., 2021). We used the official QCS codebase available at `https://github.com/beanie00/QCS`, along with the default hyperparameters recommended by the original paper. These include an Adam optimizer, a learning rate of 3e-4, batch size of 256, target update rate of 5e-3, a hidden layer size of 256, and ReLU activations (Agarap, 2018). The discount factor is $\gamma = 0.99$ for MuJoCo and $\gamma = 0.995$ in AntMaze.

Table 3: Hyperparameters of Policy in our experiments.

Parameter	Value
Number of layers	4
Number of attention heads	2
Embedding dimension	256
Activation function	ReLU
Batch size	128 (Antmaze:256)
Context length H	20
Dropout	0.1
Learning rate	3.0e-4

In the process of training the policy and refining the Q-function, we employed Polyak (Polyak & Juditsky, 1992) averaging ($\rho = 0.005$) to update the target Q-networks. The environment specific hyperparameter settings are reported in Table 4.

Table 4: Q-function tuning hyperparameters

Environment	α	learning rate
halfcheetah-medium-v2	1	1e-4
halfcheetah-medium-replay-v2	1	5e-4
halfcheetah-medium-expert-v2	1	5e-4
hopper-medium-v2	0.5	1e-4
hopper-medium-replay-v2	1	1e-4
hopper-medium-expert-v2	0.2	1e-4
walker2d-medium-v2	0.2	5e-5
walker2d-medium-replay-v2	1	1e-4
walker2d-medium-expert-v2	1	3e-4
antmaze-umaze-v2	0.05	1e-4
antmaze-umaze-diverse-v2	0.2	1e-4
antmaze-medium-diverse-v2	0.2	1e-4
antmaze-large-diverse-v2	0.2	1e-5

B BASELINES

We compare AQDT against three categories of methods:

Value-Based Methods: we benchmark against four widely recognized top-performing value-based offline RL methods including TD3+BC (Fujimoto & Gu, 2021), IQL (Kostrikov et al., 2021), CQL (Kumar et al., 2020), and SQL (Xu et al., 2023). These methods rely on learning a Q-function to optimize policy performance. We note that, for CQL, the original paper reports results in MuJoCo v0, differing from the v2 benchmark used in our study; therefore, we reference scores reported in (Kostrikov et al., 2021) for consistency.

Conditional Sequence Modeling Methods: we considered three representative CSM approaches including Decision Transformer (DT) (Chen et al., 2021), Decision Convformer (DC) (Kim et al., 2024a), and RvS-R (Emmons et al., 2021). For AntMaze-medium and AntMaze-large, we use the scores reported by Kim et al. (2024) for DT and DC to ensure consistent comparison.

Hybrid Methods: We further evaluated our approach against seven state-of-the-art methods that enhance trajectory stitching in CSM frameworks: QDT (Yamagata et al., 2023), EDT (Wu et al., 2023), CGDT (Wang et al., 2024), ACT (Gao et al., 2024), POR (Xu et al., 2022), QCS (Kim et al., 2024b) and QT (Hu et al., 2024). These methods aim to leverage mechanism for more effective action generation and policy improvement, representing state-of-the-art stitching enhancements. The performance scores for these baselines are taken either from their original publications or replicated under the same experimental setup to ensure fairness.

C Additional Performance Comparison

C.1 Trajectory Stitching performance

To assess trajectory stitching, we measure each method's improvement over the average of the top trajectory returns in the training dataset (Figure 5). This high dataset returns serves as a proxy for the best behaviors present in the offline data. The reported metric captures the percent increase of the policy's return over this reference value. In datasets where stitching is most critical AQDT consistently outperforms all baselines on average, including both pure sequence models (DT) and hybrid approaches (QT, QCS). For instance, in halfcheetah-medium-replay, AQDT exceeds the best dataset return by over 40%, highlighting its superior ability to stitch together suboptimal segments into high-return trajectories. To further demonstrate AQDT's trajectory stitching capabilities, we

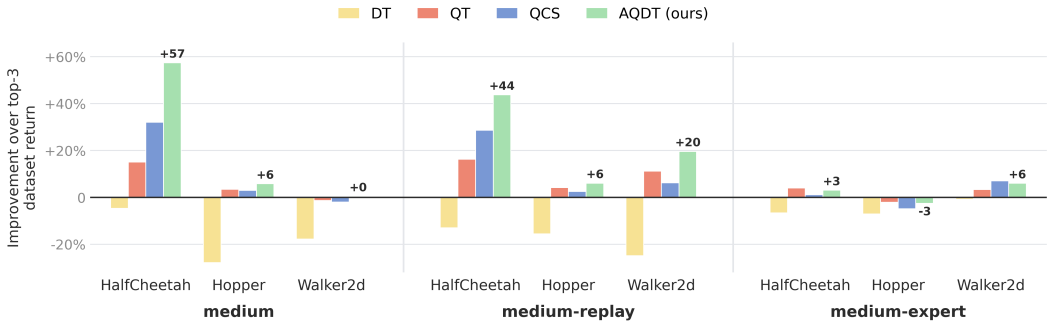


Figure 5: Percentage improvement of DT, QT, QCS, and AQDT over the average return of the top 3 trajectories in the training dataset. Positive values indicate the agent's ability to surpass the best demonstrations via trajectory stitching. AQDT consistently outperforms baselines, particularly in medium and medium-replay datasets, demonstrating superior stitching capability from suboptimal data.

analyze the distribution of trajectory returns in the offline datasets and compare them against the average return achieved by AQDT. Figure 6 presents Kernel Density Estimates (KDEs) of normalized returns for all trajectories in each MuJoCo dataset. This analysis provides insight into the variability and quality of the training data, as well as AQDT's ability to leverage suboptimal demonstrations.

The figure illustrates that each dataset exhibits a distinct distribution pattern of returns, shaped by differences in trajectory quality and behavioral diversity. For example, medium-replay datasets such as halfcheetah-medium-replay and hopper-medium-replay contain a broad spread of return values, often with a heavy concentration in the lower-return region. In contrast, medium-expert datasets like hopper-medium-expert show a bimodal distribution indicating a mixture of near-optimal and suboptimal trajectories.

In each plot, the black dotted line represents the maximum return observed among all trajectories in the dataset, while the purple dashed line indicates the average return attained by AQDT. In medium and medium-replay datasets, AQDT surpasses the best observed return in four of the six medium and medium-replay datasets which demonstrates its ability to stitch together suboptimal segments into novel, high-return behaviors. Meanwhile, in expert-level datasets, AQDT performs close to the top trajectories (within 3.2 points), confirming that its stitching mechanism does not degrade performance when it is unnecessary. These findings underscore AQDT's ability to adaptively leverage value-based guidance, achieving generalization beyond the limitations of observed behavior.

C.2 Adroit Dataset

In addition to the performance comparison in the MuJoCo and AntMaze domains, we evaluate AQDT in the Adroit domain, which presents high-dimensional control challenges involving dexterous robotic manipulation. In this setting, we compare AQDT against value-based methods (IQL, CQL), CSM baselines (DT, DC), and hybrid methods (QCS, QT). For IQL and CQL, we report the

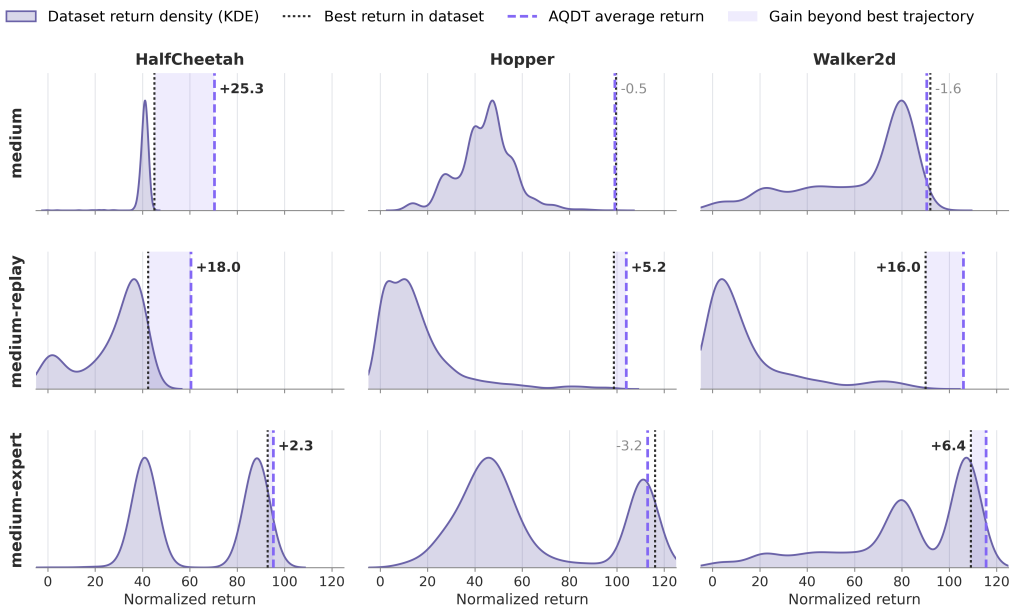


Figure 6: kernel density estimate plots of normalized returns in the training datasets for all MuJoCo tasks. The shaded region shows the density of trajectory returns in the offline dataset. The black dotted line denotes the maximum return among all trajectories in the dataset, while the purple dashed line represents the average return achieved by AQDT. In most cases especially in medium and medium-replay datasets AQDT exceeds the best observed returns, indicating its stitching capability.

scores as presented in their paper. For QT, and QCS we independently evaluate the method using the authors' official implementation and hyperparameters.

Table 5 presents the performance of AQDT alongside the baseline performances. As indicated by the results, AQDT achieves the highest mean normalized return on both Pen tasks, with lower variance than QT and QCS on pen-human, although the margins over these hybrid methods fall within one standard deviation. This demonstrates the model's robustness and ability in high-dimensional tasks. While the cloned dataset contains demonstrations of lower quality, AQDT still performs competitively, suggesting its stitching capabilities. Overall, these results validate AQDT's effectiveness beyond locomotion and navigation tasks.

Dataset	**Value-Based Method**		**CSM**		**Hybrid Methods**		**Ours**
	IQL	CQL	DT	DC	QCS	QT	AQDT
pen-human	71.5	37.5	62.9	74.2	83.9 ± 10.2	92.0 ±15.6	93.0±4.6
pen-cloned	37.3	39.2	28.7	50.0	55.46 ±12.2	55.2±12.6	57.89 ±10.6
average	54.4	38.35	45.8	62.1	69.68	73.6	**75.45**

Table 5: *Performance comparison on Adroit tasks.* Normalized scores (0-100 scale) are reported, with higher values indicating better performance. Bold values indicate best performance for each dataset. Results for AQDT are reported as mean ±standard deviation across 3 seeds.

D Training Resources

All experiments were conducted using NVIDIA RTX 6000 Ada Generation GPUs (50 GB memory). Each training run was performed on a single GPU node with 512 GB system RAM and CUDA version 12.2. No multi-GPU or distributed training was used.