STATE-VALUE TRANSFORMERS WITH IN-SAMPLE BELLMAN BACKUPS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in Transformer-based critics have shown the benefits of scaling value functions in reinforcement learning (RL). However, these methods rely on specialized architectures or attention entropy control to scale effectively. In this paper, we identify the value learning objective as a key factor in the effectiveness of Transformer-based critics and demonstrate that in-sample learning improves performance without attention entropy control. Building on these findings, we propose State-Value Transformer Implicit Q-Learning (VTIQL), a method that combines a Transformer-based V-function with an MLP-based Q-function under in-sample learning. Extensive numerical experiments on OGBench demonstrate that VTIQL consistently outperforms prior offline RL methods, achieving the best performance on **7** out of 9 environments. Notably, VTIQL exhibits high performance on robotic manipulation tasks and maze navigation tasks with high-dimensional states, whereas prior methods struggle to achieve high performance in both task categories. VTIQL also performs strongly with a compact Transformer-based V-function and remains robust across a wide range of model sizes.

1 INTRODUCTION

Transformers (Vaswani et al., 2017) have become a central architecture in deep learning, supporting major advances in natural language processing (Devlin et al., 2019; Brown et al., 2020), computer vision (Dosovitskiy et al., 2021; Liu et al., 2021), and robotics (Brohan et al., 2022; Zitkovich et al., 2023). These advances have motivated the use of Transformers in reinforcement learning (RL) for supervised trajectory modeling (Chen et al., 2021; Janner et al., 2021; Yamagata et al., 2023; Wu et al., 2023) and model-based data augmentation (Wang et al., 2022). However, learning reliable Transformer-based critics remains challenging because these critics can be difficult to optimize under RL objectives (Parisotto et al., 2020).

Recent studies have explored Transformer-based state-action value functions (Q-functions) (Chebotar et al., 2023; Springenberg et al., 2024; Dong et al., 2026a). In particular, Dong et al. (2026a) identify attention collapse (i.e., attention distributions over input tokens becoming highly concentrated, as reflected in low attention entropy) as a failure mode when scaling Transformer Q-functions under standard Q-learning. They propose attention entropy control with a prescribed target to mitigate attention collapse and stabilize Q-values. In Q-learning, however, bootstrapped temporal difference (TD) targets depend on value estimates for next-state actions generated by the learned policy. These actions may be out-of-distribution (OOD) relative to the training data, allowing errors in their estimated values to propagate through subsequent Bellman updates (Levine et al., 2020). Hence, the following question arises:

What is a suitable value learning objective for Transformer-based critics?

In this work, we investigate this question by comparing standard Q-learning and in-sample learning using Transformer-based Q-functions. Following Kostrikov et al. (2022), we construct in-sample learning by utilizing a separate state-value function (V-function) alongside the Q-function, without querying the values of unseen or OOD actions. In contrast to the findings of Dong et al. (2026a), we observe that lower attention entropy is not necessarily accompanied by lower performance under in-sample learning with a Transformer-based Q-function and a multilayer perceptron (MLP)-based V-function (Section 4.1). Since in-sample learning uses both Q- and V-functions, we further ex-

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.

amine how Transformer parameterization of each function affects performance and Bellman consistency. We observe two main findings (Section 4.2): (i) replacing an MLP-based Q-function with a Transformer does not consistently lower held-out TD mean squared error (MSE) across the two V-function architectures, and (ii) replacing an MLP-based V-function with a Transformer while retaining an MLP-based Q-function improves performance and reduces the average held-out TD MSE across tasks. These findings motivate a new Transformer-based value learning method designed to achieve strong performance across diverse environments without additional regularization, such as attention entropy control.

In response, we propose State-Value Transformer Implicit Q-Learning (VTIQL), a method that combines a Transformer-based V-function with an MLP-based Q-function under in-sample learning. We evaluate VTIQL in an offline RL setting on challenging maze navigation and robotic manipulation tasks from the OGBench benchmark (Park et al., 2025a) and observe strong performance compared with prior offline RL algorithms. Additionally, VTIQL achieves high performance even with a compact Transformer-based V-function and remains robust across a wide range of Transformer model sizes. Our main contributions are summarized as follows:

- We provide empirical evidence that the relationship between attention entropy and performance in Transformer-based critics depends on the value learning objective (Section 4.1). Specifically, we show that lower attention entropy is not necessarily accompanied by lower performance under in-sample learning, whereas both decline under standard Q-learning.
- We propose VTIQL, which uses a Transformer-based V-function to construct in-sample Bellman targets and an MLP-based Q-function for policy improvement, without requiring attention entropy targets. An ablation study comparing all MLP/Transformer combinations for the Q- and V-functions validates this design, showing that VTIQL achieves the highest performance and the lowest held-out TD MSE (Section 4.2).
- Extensive numerical experiments on OGBench demonstrate that VTIQL consistently outperforms prior offline RL methods, achieving the best performance on **7** out of 9 environments. Notably, VTIQL is particularly effective on challenging robotic manipulation tasks and maze navigation tasks with high-dimensional states. Moreover, VTIQL maintains high performance across a wide range of Transformer critic sizes (Section 5.2).
- We demonstrate that VTIQL remains computationally efficient with larger Transformer-based V-functions while retaining fixed-size MLPs for the trainable and target Q-functions. For Transformer critics of the same size, VTIQL achieves faster updates and lower peak memory than the evaluated methods with Transformer-based Q-functions (Section 5.4).

2 Related Work

Offline RL. Offline RL aims to learn policies from a fixed dataset without additional environment interaction. A major challenge in offline RL is distributional shift (Levine et al., 2020), which arises when the state-action distribution induced by the learned policy differs from that induced by the behavior policy. To mitigate this issue and improve value estimation, prior methods have explored strategies such as penalizing value estimates for OOD actions (Kumar et al., 2020; Yu et al., 2020; 2021; Lyu et al., 2022; Kim & Oh, 2023; Sun et al., 2023; Kim & Oh, 2026), constraining the learned policy to remain close to the behavior policy (Fujimoto et al., 2019; Kumar et al., 2019; Wu et al., 2019; Fujimoto & Gu, 2021; Hansen-Estruch et al., 2023; Tarasov et al., 2023; Park et al., 2025b), learning without querying OOD actions (Kostrikov et al., 2022), and applying generative modeling techniques such as flow matching to value learning (Dong et al., 2026b; Agrawalla et al., 2026). Recent work (Dong et al., 2026a) has investigated value function modeling with Transformers, enabling stable and effective scaling by controlling the entropy of attention scores. In contrast to this approach, we focus on value learning objectives for Transformer-based critics, investigating the role of Bellman backup design in critic learning and scalability without explicit attention entropy control.

Transformer-based Critics. Transformer architectures have increasingly been studied as backbones for Q-function approximation in RL. Existing approaches can be categorized by whether the Transformer-based critic evaluates action chunks (Li et al., 2025; Tian et al., 2026; Huang et al., 2025; Nagy et al., 2026; Chen et al., 2026) or individual actions (Chebotar et al., 2023; Springenberg et al., 2024; Dong et al., 2026a). Using action chunks for multi-step value evaluation, the former

approaches apply causal Transformer critics to episodic RL (Li et al., 2025), adapt them to step-based RL with single-step actors (Tian et al., 2026), incorporate multi-horizon targets (Nagy et al., 2026), and use prefix-conditioned critic outputs to select state-dependent chunk sizes (Chen et al., 2026). For individual action evaluation, existing approaches discretize action dimensions for autoregressive Q-value prediction (Chebotar et al., 2023), use Perceiver-based architectures for scalable offline actor-critic learning (Springenberg et al., 2024), and control layer-wise attention entropy to prevent attention collapse when scaling Q-networks (Dong et al., 2026a). Unlike these approaches, we address Transformer-based critic design by combining a Transformer-based V-function with an MLP-based Q-function under in-sample learning.

3 PRELIMINARIES

3.1 MARKOV DECISION PROCESS

We consider a Markov Decision Process (MDP) defined by a tuple $\mathcal{M} := (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, d_0, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta_{\mathcal{S}}$ represents the state transition probability kernel, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $d_0 \in \Delta_{\mathcal{S}}$ is the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$ maps each state to a probability distribution over actions. Given any policy π, an agent starts from an initial state s_0 and interacts with $\mathcal{M}$, repeatedly taking actions, receiving rewards, and observing subsequent states. This process generates a trajectory $\tau = \{(s_t, a_t, r(s_t, a_t))\}_{t\geq 0}$, where $a_t \sim \pi(\cdot|s_t)$ and $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$. The state-value function (V-function) and action-value function (Q-function) for the policy π are defined as $V^\pi(s) := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s\right]$ and $Q^\pi(s, a) := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a\right]$, respectively. The goal of RL is to obtain an optimal policy π^* that maximizes the expected discounted return under d_0, represented as $\max_\pi J_{\mathcal{M}}(\pi) := \mathbb{E}_{s\sim d_0}[V^\pi(s)]$. In this work, we focus on the offline RL setting, where the offline dataset $\mathcal{D} = \{\tau_i\}_{i=1}^n$ denotes a static dataset collected by unknown behavior policies π_β. Offline RL aims to learn a policy that maximizes $J_{\mathcal{M}}(\pi)$ using only the trajectories in $\mathcal{D}$, without further interaction with $\mathcal{M}$.

3.2 TRANSFORMER-BASED CRITICS

Following the critic architecture of Dong et al. (2026a), we use a standard Transformer encoder with full self-attention to estimate critic values. Each input token corresponds to one dimension of the state-action pair $(s, a) \in \mathbb{R}^{n_s+n_a}$ for the Q-function or the state $s \in \mathbb{R}^{n_s}$ for the V-function, where n_s and n_a denote the state and action dimensions, respectively. We denote the token embedding dimension by D, the number of attention heads by H, and the dimension of each attention head by $d_h = D/H$. A learnable `[VALUE]` token is prepended to the input sequence, resulting in $1+n_s+n_a$ tokens for the Q-function and $1+n_s$ tokens for the V-function. The token embeddings are combined with positional encodings before being processed by a stack of Transformer layers. The final representation of the `[VALUE]` token is projected to a scalar Q-value or state value, depending on the critic.

4 STATE-VALUE TRANSFORMER IMPLICIT Q-LEARNING

In this section, we present State-Value Transformer Implicit Q-Learning (VTIQL), a method that combines a Transformer-based V-function with an MLP-based Q-function under in-sample learning. We begin by revisiting the attention collapse identified by Dong et al. (2026a) as a failure mode of Transformer critics trained with standard Q-learning backups. Specifically, we examine whether decreasing attention entropy aligns with declining performance as Transformer-based Q-functions are scaled up. We observe that the relationship between attention entropy and performance depends on the value learning objective. Under in-sample learning, lower attention entropy is not necessarily accompanied by lower performance as model size increases (Section 4.1). Focusing on the in-sample learning framework of IQL (Kostrikov et al., 2022), we show that a Transformer-based V-function with an MLP-based Q-function achieves the best performance and the lowest held-out TD MSE among the four MLP/Transformer combinations (Section 4.2). Finally, we describe the policy extraction procedure used in VTIQL (Section 4.3).

4.1 Comparison of Bellman Backups

Attention collapse has been associated with training instability in Transformers (Zhai et al., 2023). In offline RL, Dong et al. (2026a) identify attention collapse as a failure mode when scaling Transformer-based Q-functions and show that controlling attention entropy improves training stability. In token prediction tasks, Transformer models are trained against targets specified by the training data (Vaswani et al., 2017; Devlin et al., 2019). In both standard Q-learning and in-sample learning, however, the Q-function learns from bootstrapped TD targets computed from evolving value estimates (Sutton, 1988). These targets change during training even when the dataset is fixed. In offline RL, the reliability of these targets also depends on whether their construction requires evaluating actions outside the dataset support. These properties motivate us to investigate *whether the relationship between attention entropy in Transformer critics and performance depends on Bellman target construction*.

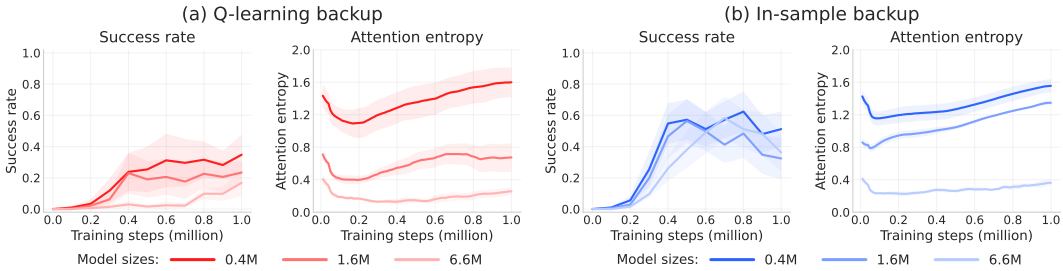


Figure 1: **Effect of Bellman target construction on Transformer critics.** Success rates and attention entropy of Transformer-based Q-functions on `cube-double` across Q-function sizes (0.4M, 1.6M, and 6.6M parameters) under (a) standard Q-learning and (b) in-sample learning.

In an off-policy actor-critic framework, the critic minimizes the Bellman error $\mathcal{L}_Q(\theta)$, while policy improvement includes maximizing the expected Q-value term $J_\pi(\omega)$:

$$\mathcal{L}_Q(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left(r(s,a) + \gamma \mathbb{E}_{a'\sim\pi_\omega(\cdot|s')}\left[Q_{\hat{\theta}}(s',a')\right] - Q_\theta(s,a)\right)^2\right],$$

$$J_\pi(\omega) = \mathbb{E}_{s\sim\mathcal{D},\, a\sim\pi_\omega(\cdot|s)}\left[Q_\theta(s,a)\right],$$

where θ and ω denote the critic and policy parameters, respectively, and $\hat{\theta}$ denotes the target critic parameters. However, even when the next state s' comes from the fixed dataset, the learned policy may select an out-of-distribution (OOD) action a', requiring the critic to extrapolate when evaluating $Q_{\hat{\theta}}(s',a')$. Errors in these estimates enter the bootstrap TD target and propagate to Q-values learned for dataset state-action pairs. Policy improvement can then favor actions with overestimated values, potentially reinforcing these errors through subsequent Bellman targets (Levine et al., 2020). For Transformer critics, this raises the possibility that limited performance reflects errors propagated through Bellman targets as well as concentrated attention.

Motivated by this mechanism, we hypothesize that *avoiding OOD action queries during bootstrapping improves performance with Transformer critics, even without attention entropy control*. To investigate our hypothesis, we compare the Q-learning formulation above with an in-sample formulation based on expectile regression (Kostrikov et al., 2022). The in-sample formulation uses the following objectives for the V- and Q-functions:

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[L_2^\tau\left(Q_{\hat{\theta}}(s,a) - V_\psi(s)\right)\right],$$

$$\mathcal{L}_Q(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left(r(s,a) + \gamma V_\psi(s') - Q_\theta(s,a)\right)^2\right],$$

where ψ denotes the parameters of the V-function, and $L_2^\tau(u) = |\tau - \mathbf{1}\{u < 0\}|u^2$ is the expectile regression loss with parameter $\tau \in (0,1)$. At each model size, both variants use the same Transformer-based Q-function architecture and parameter count. The in-sample variant additionally uses an MLP-based V-function. Neither variant uses attention entropy control. Following Dong et al. (2026a), both variants use the policy extraction procedure from FQL (Park et al., 2025b).

As shown in Figure 1, increasing Transformer critic size under the Q-learning backup is accompanied by lower success rates and lower attention entropy toward the end of training. Under the in-sample backup, attention entropy also decreases with critic size toward the end of training. However,

its success rates do not decrease monotonically with model size and exceed those of the Q-learning backup at every tested size. The in-sample results contrast with the pattern reported by Dong et al. (2026a), suggesting that lower attention entropy is not necessarily accompanied by lower performance. These results are consistent with our hypothesis that avoiding OOD action queries during bootstrapping can improve performance with Transformer critics without attention entropy control.

4.2 Comparing Q- and V-function Architectures

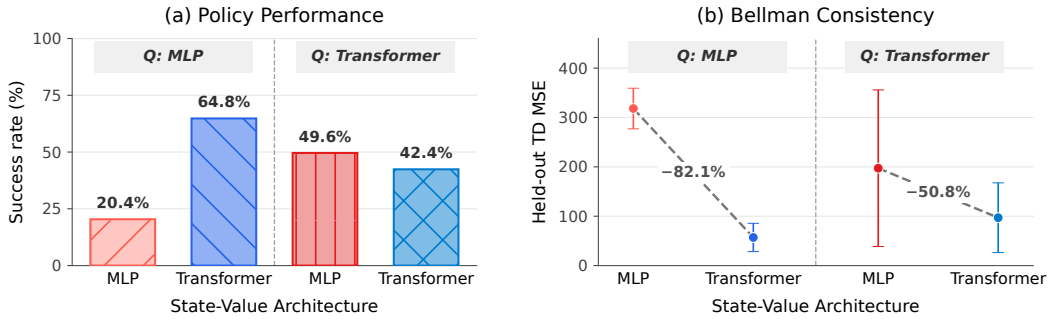


Figure 2: **Effect of Q- and V-function architectures on performance and Bellman consistency.** We compare all four MLP/Transformer combinations on the `cube-double` dataset under fixed IQL critic objectives and FQL policy extraction. (a) Mean success rates across 5 tasks. (b) Held-out TD MSE averaged over the same 5 tasks, computed using dataset actions and each model's own bootstrap targets.

Within the in-sample learning framework, we investigate how the architectures of the Q- and V-functions affect performance and Bellman consistency. We compare all four MLP/Transformer combinations for Q- and V-functions, keeping the IQL critic objectives, FQL policy extraction procedure, and other training settings fixed. A Transformer-based V-function with an MLP-based Q-function achieves the highest mean success rate of 64.8% in Figure 2-(a). Compared with this configuration, using a Transformer-based Q-function yields lower success rates with either an MLP or a Transformer-based V-function. Using MLPs for both functions yields the lowest success rate. These results suggest that replacing an MLP-based V-function with a Transformer improves performance when the Q-function is an MLP, but not when it is a Transformer.

To complement the performance comparison, we examine the Bellman consistency of the learned critics. For each task, we evaluate all four configurations on the same set of $8,192$ transitions from a held-out validation split. Using dataset actions, we compute the mean squared TD residual over these transitions and both Q heads, with bootstrap targets constructed from each model's own V-function. Figure 2-(b) reports the mean TD MSE across the five tasks, with error bars indicating one standard deviation across tasks. Replacing an MLP-based V-function with a Transformer reduces the held-out TD MSE for both Q-function architectures, indicating closer agreement with their respective bootstrap targets. In contrast, replacing an MLP-based Q-function with a Transformer does not consistently reduce held-out TD MSE across the two V-function architectures. Among the four assignments, a Transformer-based V-function with an MLP-based Q-function achieves both the highest mean success rate and the lowest mean held-out TD MSE. VTIQL adopts this architecture and trains both functions using the IQL objectives above.

4.3 Policy Extraction

VTIQL can be combined with different value-based policy extraction schemes. Common choices include value-weighted behavioral cloning objectives such as AWR (Peng et al., 2019) and behavior-constrained policy gradient objectives such as DDPG+BC (Fujimoto & Gu, 2021) and FQL (Park et al., 2025b). Behavior-constrained policy gradient objectives often use learned value functions more effectively than value-weighted behavioral cloning objectives (Park et al., 2024). Thus, we adopt the policy extraction objective of FQL (Park et al., 2025b), training a one-step policy to maximize Q-values while regularizing it through distillation from a behavioral cloning (BC) flow policy:

$$\mathcal{L}_{\mathrm{BC}}(\phi) = \mathbb{E}_{\substack{s,a=x^1\sim\mathcal{D},\ x^0\sim\mathcal{N}(0,I_{n_a}),\\ t\sim\mathrm{Unif}([0,1])}} \left[\left\| v_\phi(t,s,x^t) - (x^1 - x^0) \right\|_2^2 \right],$$

$$\mathcal{L}_{\mathrm{Distill}}(\omega) = \mathbb{E}_{s\sim\mathcal{D}, z\sim\mathcal{N}(0,I_{n_a})} \left[\left\| \mu_\omega(s,z) - \mu_\phi(s,z) \right\|_2^2 \right],$$

$$\mathcal{L}_\pi(\omega) = \mathbb{E}_{s\sim\mathcal{D},\, a^\pi\sim\pi_\omega(\cdot|s)} \left[-Q_\theta(s,a^\pi) \right] + \alpha \mathcal{L}_{\mathrm{Distill}}(\omega)$$

where $x^t = (1-t)x^0 + tx^1$ and α is the distillation weight. The velocity field v_ϕ of the BC flow policy generates the action $\mu_\phi(s,z)$ by integration from initial noise z over $t \in [0,1]$. Additionally, in Section B.3, we present experimental comparisons using AWR for policy extraction.

5 EXPERIMENTS

In this section, we describe our detailed experimental procedures and report the corresponding results to address the following questions:

(i) How does the performance of VTIQL compare to that of prior offline RL baselines using MLP or Transformer critics across various environments?

(ii) Can VTIQL maintain high performance as model size increases, while consistently outperforming other baselines with Transformer critics across different model sizes?

(iii) What impact do architectural choices within the Transformer-based V-function, such as the number of attention heads, have on the performance of VTIQL?

(iv) Does VTIQL offer computational advantages over other methods with Transformer critics and its variants with different Q and V architectures?

For a fair comparison, we evaluate all algorithms with a training budget of 1M gradient steps. We average success rates at 0.8M, 0.9M, and 1M gradient steps for each seed and report the mean $\pm$ standard deviation across 5 random seeds. Each Transformer critic module contains approximately 26M parameters, excluding target networks and any accompanying MLP critics.

Datasets. We evaluate our proposed algorithm and baselines on reward-based single-task environments (referred to as "singletask") from OGBench (Park et al., 2025a), a comprehensive benchmark for offline RL. We use 9 environments with 5 tasks each, yielding a total of 45 singletask settings, encompassing maze navigation (`antmaze` and `humanoidmaze`) and robotic manipulation (`cube`, `scene`, and `puzzle`). The maze navigation tasks evaluate the agent's ability to make decisions over long horizons, while the robotic manipulation tasks require precise object manipulation using a 6-DoF robot arm. More details on the datasets are included in Appendix A.1.

Baselines. We compare VTIQL with BC and prior offline RL methods using MLP or Transformer critics. For the baselines with MLP critics, we consider the following methods: (i) BC (Pomerleau, 1989) learns a policy by directly imitating actions in the offline dataset, (ii) IQL (Kostrikov et al., 2022) estimates the V-function with expectile regression and updates the Q-function through Bellman regression, (iii) ReBRAC (Tarasov et al., 2023) penalizes deviations from dataset actions using separate regularization coefficients for the actor and critic, (iv) FBRAC (Park et al., 2025b) combines behavior-regularized actor-critic (Wu et al., 2019) with a flow policy, (v) IFQL (Park et al., 2025b) adapts implicit diffusion Q-learning (Hansen-Estruch et al., 2023) by replacing the diffusion policy with a flow policy, and (vi) FQL (Park et al., 2025b) trains a one-step policy to maximize Q-values while regularizing it through distillation from a BC flow policy. For the Transformer baselines, we consider the following methods: (vii) PAC (Springenberg et al., 2024) uses a Perceiver critic in which action queries attend to latent state representations for Q-value estimation, and (viii) TQL (Dong et al., 2026a) parameterizes the Q-function with a Transformer and controls attention entropy to stabilize training as model size increases. Additionally, our ablation study includes QTIQL as a variant of VTIQL with a Transformer-based Q-function and an MLP-based V-function.

Table 1: **Full benchmark table.** We report the (binary) success rate (%) across the five tasks in each environment, averaged over 5 random seeds. † The results of PAC and TQL in maze navigation tasks (`antmaze` and `humanoidmaze`) are reproduced using the official implementation.

	MLP Value						Transformer Value		
Environment (5 tasks each)	BC	IQL	ReBRAC	FBRAC	IFQL	FQL	PAC†	TQL†	VTIQL (**ours**)
`antmaze-large`	11 ± 3	53 ± 11	$\mathbf{81} \pm 8$	60 ± 15	28 ± 13	79 ± 6	65 ± 10	61 ± 26	71 ± 4
`antmaze-giant`	0 ± 0	4 ± 1	$\mathbf{26} \pm 8$	4 ± 4	3 ± 2	9 ± 6	4 ± 4	3 ± 1	15 ± 2
`humanoidmaze-medium`	2 ± 1	33 ± 5	22 ± 15	38 ± 8	60 ± 16	57 ± 8	29 ± 6	14 ± 11	$\mathbf{73} \pm 5$
`humanoidmaze-large`	0 ± 0	2 ± 1	3 ± 2	2 ± 0	11 ± 2	4 ± 2	2 ± 1	2 ± 1	$\mathbf{27} \pm 4$
`cube-double`	2 ± 1	6 ± 2	12 ± 3	15 ± 6	14 ± 5	29 ± 6	30 ± 14	67 ± 10	$\mathbf{70} \pm 8$
`cube-triple`	0 ± 0	1 ± 1	0 ± 0	0 ± 0	0 ± 0	4 ± 2	0 ± 0	$\mathbf{7} \pm 4$	$\mathbf{7} \pm 3$
`puzzle-3x3`	2 ± 1	9 ± 3	22 ± 2	14 ± 5	19 ± 1	30 ± 4	28 ± 5	32 ± 7	$\mathbf{93} \pm 5$
`puzzle-4x4`	0 ± 0	7 ± 2	14 ± 3	13 ± 5	25 ± 8	17 ± 5	9 ± 5	31 ± 6	$\mathbf{40} \pm 5$
`scene`	5 ± 2	28 ± 3	41 ± 7	45 ± 5	30 ± 4	56 ± 2	51 ± 4	61 ± 7	$\mathbf{63} \pm 4$
Average Score	3 ± 1	16 ± 3	24 ± 5	21 ± 5	21 ± 6	32 ± 5	24 ± 5	31 ± 8	$\mathbf{51} \pm 5$

5.1 Benchmark Results

We summarize the experimental results on the full benchmark in Table 1. VTIQL achieves the best performance in **7** out of 9 environments. Its average success rate is 51%, exceeding those of FQL (32%) and TQL (31%) by 19 and 20 percentage points, respectively.

Maze Navigation. For maze navigation tasks such as `antmaze` and `humanoidmaze`, we observe marked differences in high performance across baselines. Although ReBRAC achieves the highest mean success rates on both `antmaze-large` and `antmaze-giant`, it is less competitive on both `humanoidmaze-medium` and `humanoidmaze-large`, which have higher state dimensionality. TQL underperforms FQL on all four maze navigation environments. In contrast, VTIQL consistently outperforms PAC and TQL, both of which use Transformer critics, across maze navigation tasks. Although VTIQL achieves lower performance than ReBRAC and FQL on `antmaze-large`, it surpasses both methods on `humanoidmaze` at both the `medium` and `large` sizes. These results highlight VTIQL's strong performance on the high-dimensional `humanoidmaze` tasks.

Robotic Manipulation. For robotic manipulation tasks such as `cube`, `puzzle`, and `scene`, VTIQL exhibits remarkable performance, matching or outperforming all baselines. Methods with MLP-based critics struggle particularly on the `cube` and `puzzle` tasks. Although TQL is less competitive on maze navigation tasks, it outperforms all MLP-based value baselines on manipulation tasks. However, VTIQL further improves upon TQL, with substantial gains on `puzzle-3x3` and `puzzle-4x4`, where the success rates increase from 32% to 93% and from 31% to 40%, respectively. It also obtains slightly higher performance on `cube-double` and `scene`. These results highlight the effectiveness of VTIQL for challenging robotic manipulation tasks.

5.2 Robustness across Model Sizes

While Table 1 demonstrates the strong performance of VTIQL with approximately 26M parameters, it is important to investigate how its performance varies with model size in comparison with other methods using the same policy extraction procedure. To this end, we compare VTIQL with FQL, TQL, and QTIQL on `humanoidmaze-medium`, `cube-double`, and `puzzle-4x4`, using critic sizes of approximately 0.4M, 1.6M, 6.6M, and 26M parameters. For VTIQL, we vary the embedding dimension D while keeping the number of attention heads H fixed. As shown in Figure 3, VTIQL exhibits stable performance across a wide range of model sizes, from 0.4M to 26M parameters. Its robustness and consistently superior performance across this range provide a substantial additional advantage beyond the strong results obtained with the 26M model. In contrast, FQL's average success rate across the three environments decreases beyond 1.6M parameters. TQL improves at model sizes above 1.6M parameters on the robotic manipulation tasks, `cube-double` and `puzzle-4x4`, but its performance remains below that of VTIQL even at 26M parameters. On `humanoidmaze-medium`, TQL's success rate increases up to 6.6M parameters but drops sharply

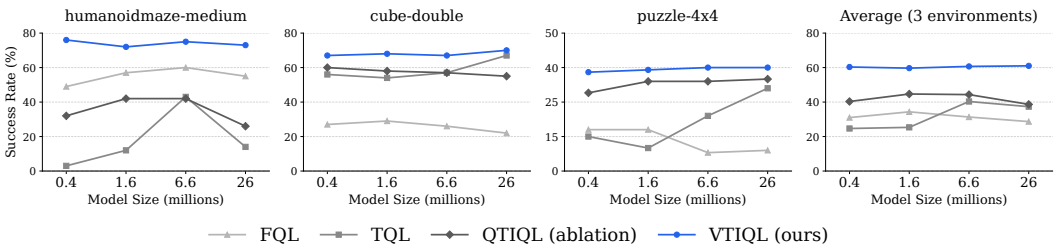


Figure 3: Success rates across model sizes on maze navigation (`humanoidmaze-medium`) and robotic manipulation (`cube-double` and `puzzle-4x4`) tasks. The last panel shows the average across the three environments.

at 26M, remaining below that of VTIQL at all evaluated sizes. The average success rate of QTIQL declines at 26M parameters and remains below that of VTIQL at all evaluated sizes. Overall, these results indicate that VTIQL combines robustness across model sizes with consistently higher performance than competing methods with Transformer critics.

5.3 Effects of Attention Head Configuration

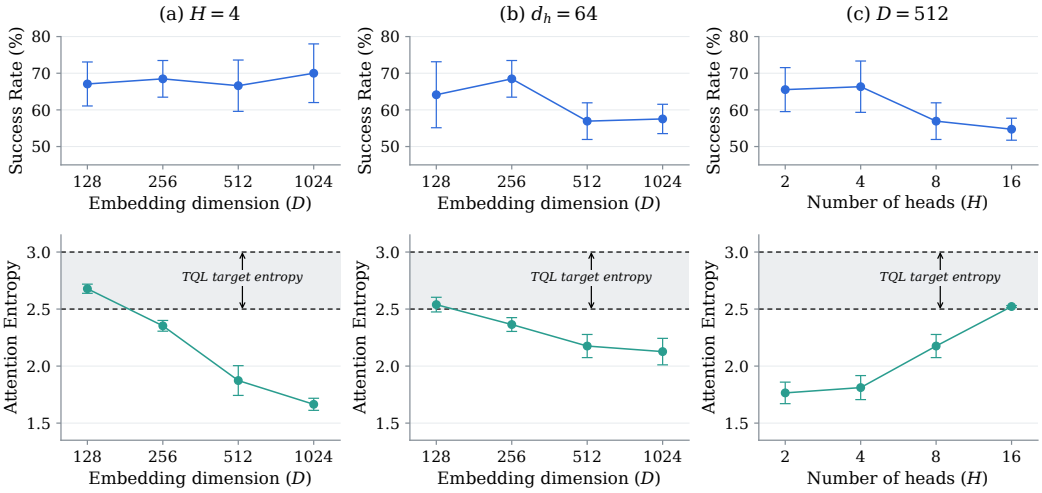


Figure 4: **Attention head ablations in VTIQL on** `cube-double`. Columns show configurations with (a) a fixed number of heads ($H = 4$), (b) a fixed per-head dimension ($d_h = 64$), and (c) a fixed embedding dimension ($D = 512$). The top row shows success rates averaged over five tasks, while the bottom row shows attention entropy. Dashed lines mark TQL's entropy targets of 2.5 and 3.0.

We investigate how architectural choices within the Transformer-based V-function affect task performance and attention entropy in VTIQL on `cube-double`, using the same training and evaluation protocol. Prior work in natural language processing (Michel et al., 2019; Bhojanapalli et al., 2020) suggests that additional attention heads do not necessarily improve performance and that the dimension of each head can constrain its expressive power. Thus, we consider three configurations involving the number of heads H, embedding dimension D, and head dimension $d_h = D/H$.

As shown in the upper plot of Figure 4-(a), we fix $H = 4$ and vary D from 128 to 1024, thereby increasing d_h from 32 to 256. VTIQL maintains similar success rates across these head dimensions, with overlapping error bars. In Figure 4-(b), we fix $d_h = 64$ and increase H from 2 to 16, increasing D from 128 to 1024. Configurations with $H = 8$ or 16 yield lower mean success rates than those with $H = 2$ or 4, despite their larger model sizes. This performance drop occurs even though the per-head dimension remains fixed. Finally, we fix $D = 512$ and compare the same head counts in the upper plot of Figure 4-(c), keeping the model size approximately constant. The two configurations with fewer heads perform similarly and outperform those with more heads. Within the tested ranges on `cube-double`, VTIQL maintains similar success rates as D increases with $H = 4$, whereas

configurations with $H = 8$ or 16 perform worse than those with $H = 2$ or 4 under both fixed d_h and fixed D. Increasing the embedding dimension should therefore not automatically be accompanied by a proportional increase in head count.

The attention entropy measurements further show that higher entropy does not consistently correspond to better performance. TQL (Dong et al., 2026a) controls attention entropy toward prescribed targets of 2.5 and 3.0 on `cube-double` to stabilize Transformer-based Q-function learning. However, Figure 4-(a) shows that VTIQL maintains strong performance as attention entropy decreases from 2.68 to 1.66, reaching a value below both targets. Conversely, in the lower plot of Figure 4-(c), increasing H from 4 to 16 raises entropy from 1.81 to 2.52 while reducing the success rate. These results show that VTIQL maintains strong performance on `cube-double` under in-sample learning without attention entropy control, even at entropy levels below TQL's prescribed targets.

5.4 Computational Efficiency

Table 2: **Computational efficiency on** `cube-double`. VTIQL achieves faster training updates and lower peak memory usage than QTIQL and TQL at matched Transformer sizes. Relative time is normalized to VTIQL within each size group. Total includes target networks and excludes policies. † denotes the number of parameters in the trainable Transformer-based critic network.

		Critic parameters (M)				Computational cost		
Transformer size†	Method	V	Q	Target Q	Total	Update time (ms) ↓	Relative time ↓	Memory (GiB) ↓
1.6M	VTIQL	1.6	0.8	0.8	3.2	**24.03**	1.00×	**0.63**
	QTIQL	0.8	1.6	1.6	4.0	44.79	1.86×	1.15
	TQL	–	1.6	1.6	3.2	39.92	1.66×	0.94
6.6M	VTIQL	6.6	0.8	0.8	8.2	**43.28**	1.00×	**1.26**
	QTIQL	0.8	6.6	6.6	14.0	75.00	1.73×	2.18
	TQL	–	6.6	6.6	13.2	69.51	1.61×	2.01
26M	VTIQL	26	0.8	0.8	27.6	**96.19**	1.00×	**2.50**
	QTIQL	0.8	26	26	52.8	172.93	1.80×	4.98
	TQL	–	26	26	52.0	164.75	1.71×	4.69

We have compared task performance at the same Transformer sizes, but total critic parameter counts can differ across methods once target networks are included. Although TQL does not use a separate V-function, it maintains both a trainable Transformer-based Q-function and a target copy updated from the trainable critic. VTIQL scales the Transformer-based V-function while retaining MLPs of fixed size for the trainable and target Q-functions. In contrast, QTIQL keeps its MLP-based V-function fixed in size while scaling both the trainable and target Transformer-based Q-functions.

Each configuration is evaluated in a separate process on a single NVIDIA RTX 3090 GPU using a batch size of 256. After JAX compilation, we perform 10 warm-up updates followed by 100 synchronized training updates. We report the median wall-clock time per update. Peak GPU memory usage is obtained from JAX memory statistics. As shown in Table 2, VTIQL achieves lower update time and peak memory usage than both QTIQL and TQL at every tested Transformer size. TQL and QTIQL require 1.61–1.71 and 1.73–1.86 times the runtime of VTIQL, respectively. VTIQL also reduces peak memory usage by approximately 33–47% relative to TQL and 42–50% relative to QTIQL. Together with the preceding scaling results, these measurements demonstrate that VTIQL combines higher success rates with faster training updates and lower peak memory usage.

6 Conclusion

We propose VTIQL, an in-sample learning method that combines a Transformer-based V-function with an MLP-based Q-function. Our analysis highlights the importance of the value learning objective, showing that lower attention entropy does not necessarily correspond to lower performance under in-sample learning. VTIQL maintains strong task performance across model sizes without attention entropy control, while achieving faster training updates and lower peak memory usage at matched Transformer sizes. These results demonstrate the effectiveness of Transformer-based state-value functions with in-sample Bellman backups for offline RL.

AI USE STATEMENT

In this work, we used generative AI tools for code assistance and running experiments, generating the plots and for grammar correction. We have not used generative AI tools for idea generation, designing the experiments and the general writing of the paper. We developed the base code. We have manually vetted and reviewed all the code generated by AI as well as made sure that the plots faithfully and accurately describe our results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work presents an in-sample learning method that uses a Transformer-based state-value function for offline RL. All experiments are conducted in simulated environments. Detailed experimental settings and implementation details are provided in Appendix A. This study does not involve human subjects, personal data, or deployment in real-world decision-making systems. We are not aware of any specific ethical or societal risks arising directly from the proposed method beyond the general potential for more efficient machine learning systems to be used in applications with societal impacts.

REPRODUCIBILITY STATEMENT

In Appendix A, we provide detailed descriptions of the datasets, experimental setup, and baseline methods, along with hardware specifications and hyperparameter settings. All datasets used in this work are publicly available and cited in the paper. Our source code is also included in the supplementary material.

REFERENCES

Bhavya Kumar Agrawalla, Michal Nauman, Khush Agrawal, and Aviral Kumar. floq: Training critics via flow-matching for scaling compute in value-based RL. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=m14YNdmPAh.

Srinadh Bhojanapalli, Chulhee Yun, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Low-rank bottleneck in multi-head attention models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 864–873. PMLR, 2020. URL https://proceedings.mlr.press/v119/bhojanapalli20a.html.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2212.06817*, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Yevgen Chebotar, Quan Vuong, Karol Hausman, Fei Xia, Yao Lu, Alex Irpan, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, Keerthana Gopalakrishnan, Julian Ibarz, Ofir Nachum, Sumedh Anand Sontakke, Grecia Salazar, Huong T. Tran, Jodilyn Peralta, Clayton Tan, Deeksha Manjunath, Jaspiar Singh, Brianna Zitkovich, Tomas Jackson, Kanishka Rao, Chelsea Finn,

and Sergey Levine. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 3909–3928. PMLR, 2023.

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.

Qian Chen, Junqiao Zhao, Hongtu Zhou, Hang Yu, Yanping Zhao, Chen Ye, and Guang Chen. Acsac: Adaptive chunk size actor-critic with causal transformer q-network. *arXiv preprint arXiv:2605.11009*, 2026.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Perry Dong, Kuo-Han Hung, Alexander Swerdlow, Dorsa Sadigh, and Chelsea Finn. TQL: Scaling q-functions with transformers by preventing attention collapse. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=lwHSE1xYHH.

Perry Dong, Chongyi Zheng, Chelsea Finn, Dorsa Sadigh, and Benjamin Eysenbach. Value flows. In *International Conference on Learning Representations*, 2026b.

JaeHyeok Doo, Byeongguk Jeon, Seonghyeon Ye, Kimin Lee, and Minjoon Seo. Q-Flow: Stable and expressive reinforcement learning with flow-based policy. In *International Conference on Machine Learning*, 2026.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 35:20132–20145, 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.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4246–4253. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.379.

Anas Houssaini, Mohamad H. Danesh, Amin Abyaneh, Scott Fujimoto, Hsiu-Chin Lin, and David Meger. Drift Q-learning. *arXiv preprint arXiv:2606.00350*, 2026.

Dongchi Huang, Zhirui Fang, Tianle Zhang, Yihang Li, Lin Zhao, and Chunhe Xia. Co-rft: Efficient fine-tuning of vision-language-action models through chunked offline reinforcement learning. *arXiv preprint arXiv:2508.02219*, 2025.

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.

Byeongchan Kim and Min-hwan Oh. Model-based offline reinforcement learning with count-based conservatism. In *International conference on machine learning*, pp. 16728–16746. PMLR, 2023.

Byeongchan Kim and Min-hwan Oh. Peng's q(λ) for conservative value estimation in offline reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Ml4AtrrfQT.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

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*, 33, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34:1179–1191, 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.

Ge Li, Dong Tian, Hongyi Zhou, Xinkai Jiang, Rudolf Lioutikov, and Gerhard Neumann. Top-erl: Transformer-based off-policy episodic reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 44534–44563, 2025.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 9992–10002. Ieee, 2021.

Jiafei Lyu, Xiaoteng Ma, Xiu Li, and Zongqing Lu. Mildly conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:1711–1724, 2022.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32, 2019.

CF Nagy, Onur Celik, Emiliyan Gospodinov, Florian Seligmann, Weiran Liao, Aryan Kaushik, and Gerhard Neumann. Sear: Sample efficient action chunking reinforcement learning. *arXiv preprint arXiv:2603.01891*, 2026.

Franki Nguimatsia Tiofack, Théotime Le Hellard, Fabian Schramm, Nicolas Perrin-Gilbert, and Justin Carpentier. Guided flow policy: Learning from high-value actions in offline reinforcement learning. In *International Conference on Learning Representations*, 2026.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pp. 7487–7498. PMLR, 2020.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline rl? *Advances in Neural Information Processing Systems*, 37:79029–79056, 2024.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, 2025a.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 48104–48127. PMLR, 2025b.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, pp. 305–313, 1989.

Jost Tobias Springenberg, Abbas Abdolmaleki, Jingwei Zhang, Oliver Groth, Michael Bloesch, Thomas Lampe, Philemon Brakel, Sarah Maria Elisabeth Bechtle, Steven Kapturowski, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Offline actor-critic reinforcement learning scales to large models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 46323–46350. PMLR, 2024.

Yihao Sun, Jiaji Zhang, Chengxing Jia, Haoxin Lin, Junyin Ye, and Yang Yu. Model-bellman inconsistency for model-based offline reinforcement learning. In *International Conference on Machine Learning*, pp. 33177–33194. PMLR, 2023.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3 (1):9–44, 1988.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Dong Tian, Onur Celik, and Gerhard Neumann. Chunking the critic: A transformer-based soft actor-critic with n-step returns. In *International Conference on Learning Representations*, volume 2026, pp. 120056–120094, 2026.

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.

Kerong Wang, Hanye Zhao, Xufang Luo, Kan Ren, Weinan Zhang, and Dongsheng Li. Bootstrapped transformer for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:34748–34761, 2022.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

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

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in neural information processing systems*, 33:14129–14142, 2020.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

Shuangfei Zhai, Tatiana Likhomanenko, Etai Littwin, Dan Busbridge, Jason Ramapuram, Yizhe Zhang, Jiatao Gu, and Joshua M. Susskind. Stabilizing transformer training by preventing attention entropy collapse. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 40770–40803. PMLR, 2023.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot*

Learning, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A Experiment Details

A.1 Dataset and Environment Details

We evaluate VTIQL on 9 state-based single-task environment configurations from OGBench (Park et al., 2025a), covering maze navigation and robotic manipulation. Each environment configuration provides 5 predefined tasks, resulting in 45 tasks in total. We use the `navigate` datasets for the maze environments and the `play` datasets for the manipulation environments.

Maze Datasets. For `antmaze` and `humanoidmaze`, we use the `navigate` datasets collected by noisy expert policies that repeatedly navigate toward randomly sampled goals. The trajectories contain transitions generated during locomotion and navigation throughout the maze. For each task, the policy is trained solely on the corresponding offline dataset and evaluated on a fixed target.

AntMaze. The `antmaze` environments use an 8-DOF quadrupedal robot with a 29-dimensional state space comprising joint configurations and velocities. These environments require learning both low-level locomotion and high-level maze navigation from offline trajectories. We evaluate `antmaze-large` and `antmaze-giant`, which differ in maze size. The larger layouts require continuous control over longer navigation horizons.

HumanoidMaze. The `humanoidmaze` environments use a 21-DOF full-body humanoid with a 69-dimensional state space. The state includes joint configurations, body positions, orientations, and velocity information. We evaluate `humanoidmaze-medium` and `humanoidmaze-large`. Compared with `antmaze`, these environments have higher-dimensional state and action spaces and require coordinated whole-body locomotion and balance while navigating toward target locations.

Manipulation Datasets. For `cube`, `scene`, and `puzzle`, we use the `play` datasets collected by scripted policies. In `cube`, the data-collection policy repeatedly selects and relocates cubes. In `scene`, the policy interacts with the available objects, while in `puzzle`, it generates sequences of button presses. These datasets are not collected specifically to demonstrate complete solutions to the evaluation tasks. For each task, the policy is trained on the corresponding offline dataset using task-specific reward labels.

Cube. The `cube` environments use a 6-DOF robot arm to perform pick-and-place operations on multiple colored cubes. The policy operates in a 5-dimensional action space controlling end-effector translation, yaw rotation, and gripper opening. We evaluate `cube-double` and `cube-triple`, which have 37- and 46-dimensional state spaces, respectively. The five tasks in each configuration require different combinations of moving, swapping, rearranging, and stacking cubes. Each task requires a sequence of cube manipulations to reach the specified target configuration.

Scene. The `scene` environment uses the same 6-DOF robot arm and 5-dimensional action space as in `cube`, with a 40-dimensional state representation describing the robot, a cube, two buttons, a drawer, and a window. The buttons control the lock states of the drawer and window, introducing constraints on the order of object manipulations. The five tasks specify different target configurations of these objects. Tasks involving the drawer or window require interaction sequences that satisfy the corresponding lock constraints.

Puzzle. The `puzzle` environments implement the Lights Out problem as a continuous-control robotic manipulation task. The robot uses a 5-dimensional action space to press buttons on a grid. Each press toggles the state of the selected button and its horizontally and vertically adjacent neighbors. We evaluate `puzzle-3x3` and `puzzle-4x4`, which have 55- and 83-dimensional state spaces, respectively. Each of the five tasks specifies a target button configuration. The policy must execute a sequence of button presses that transforms the initial configuration into the target configuration.

A.2 Evaluation Details

Each task is trained independently with five random seeds. For each seed, we evaluate the policy over 50 episodes at 800K, 900K, and 1M training steps and average the success rates across the three checkpoints. We then report the mean across seeds, with standard deviation where applicable. Success rate is the fraction of episodes satisfying the task specific completion criterion defined by OGBench. Environment-level performance is averaged over its five tasks, and overall performance is averaged equally across environment configurations. For the full benchmark, this corresponds to averaging over all 45 tasks. Unless otherwise specified, all results follow this evaluation protocol. The sources and implementation details of the baseline methods are described in Section A.4.

A.3 VTIQL Implementation Details

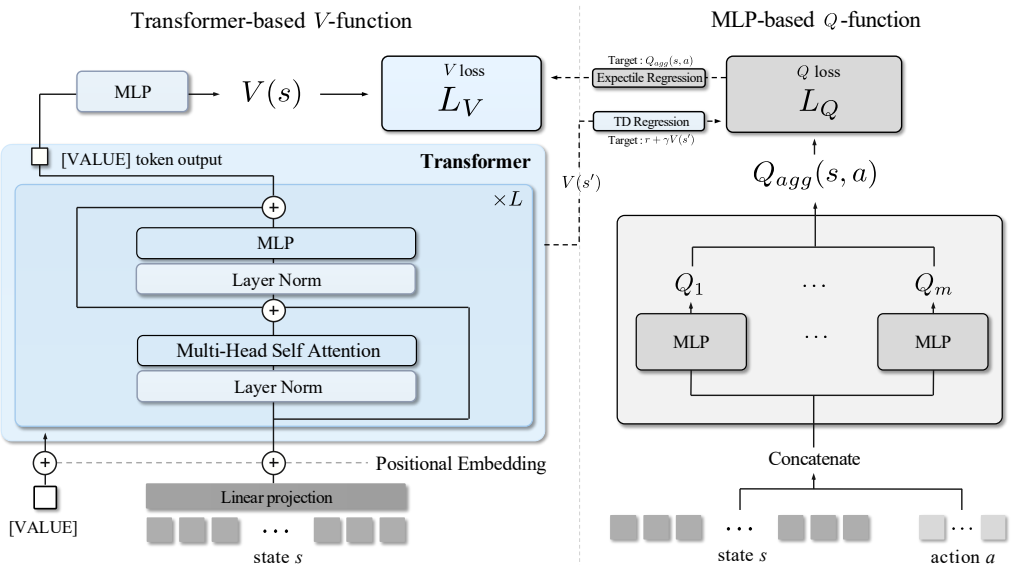


Figure 5: **VTIQL architecture.**

Figure 5 illustrates the overall architecture of VTIQL. We parameterize the state-value function with a Transformer and the action-value function with an MLP. For a state $s \in \mathbb{R}^{n_s}$, we represent each scalar state dimension as a separate token. A shared linear projection maps each scalar state value to a D-dimensional embedding. We add a learnable state-type embedding shared across all state tokens together with dimension-specific positional embeddings that identify individual state dimensions. A learnable `[VALUE]` token is prepended to the sequence, resulting in an input sequence of $n_s + 1$ tokens.

The sequence is processed by L Transformer blocks with pre-layer normalization. Each block consists of multi-head self-attention followed by a two-layer feedforward network with GELU activations, with residual connections applied around both sublayers. With H attention heads, each attention head has dimension $d_h = D/H$. Following query-key normalization (Henry et al., 2020), we apply an RMSNorm-based variant to the projected queries and keys within each attention head before computing the attention logits. Queries and keys are normalized using separate RMSNorm modules with learnable scale parameters and $\epsilon = 10^{-6}$, while retaining the standard $1/\sqrt{d_h}$ scaling of the attention logits. This normalization is applied within the attention mechanism and is distinct from the pre-layer normalization applied before the attention and feedforward sublayers. After the final Transformer block, we apply a final layer normalization and pass the representation corresponding to the `[VALUE]` token through an output head to produce the scalar state-value estimate $V(s)$.

The MLP-based Q-function takes the state and action as inputs and is trained using the Bellman objective, with targets constructed from the Transformer-based V-function. When aggregating multiple Q-value estimates, we denote the resulting value by Q_{agg}, with either minimum or mean aggregation. The Transformer-based V-function is trained using the expectile regression objective. For policy extraction, we follow FQL (Park et al., 2025b), using a flow-matching behavioral policy and a one-step policy optimized with the learned Q-function. For the robustness experiments across model sizes, we vary the Transformer embedding dimension D over $\{128, 256, 512, 1024\}$. The default training hyperparameters are provided in Table 3.

A.4 Baseline Implementation Details

We summarize the implementation details and sources of the baseline results used in our comparisons. We run FQL, PAC, and TQL, together with the QTIQL variant, using the implementations and configurations described below. For BC, IQL, ReBRAC, FBRAC, and IFQL, we use results reported in prior OGBench evaluations (Park et al., 2025b; Dong et al., 2026a) without rerunning these methods. For these previously reported baselines, we briefly describe the source configurations and define the method-specific hyperparameters listed in our tables. The corresponding hyperparameter values are provided for reference and reflect the settings selected in the source evaluations, rather than additional tuning performed in this work.

Behavioral Cloning (BC). The reported BC baseline uses a Gaussian policy with unit standard deviation, parameterized by a four-layer MLP with a hidden dimension of 512 (Park et al., 2025b).

Implicit Q-Learning (IQL) (Kostrikov et al., 2022). The reported IQL configuration uses an expectile parameter of 0.9 and advantage-weighted regression (AWR) for policy extraction (Park et al., 2025b). Here, α denotes the AWR inverse temperature, with environment-specific values selected in the source evaluation. For state-based manipulation tasks, the source evaluation uses four-layer MLPs with a hidden dimension of 256 to mitigate overfitting.

ReBRAC (Tarasov et al., 2023). For ReBRAC, α_1 and α_2 denote the behavior-cloning regularization coefficients for the actor and critic, respectively. Specifically, α_1 weights the behavior-cloning penalty in the actor objective, while α_2 weights the behavior regularization applied to the Bellman target. The source implementation applies Q-value normalization only in the actor loss (Park et al., 2025b). The environment-specific coefficient values listed in our tables are those selected in the source evaluation.

FBRAC and IFQL (Park et al., 2025b). The reported FBRAC and IFQL baselines are implemented on the same codebase as FQL and share its flow-matching implementation. FBRAC follows the default FQL configuration, with α denoting the behavior-cloning coefficient used for policy regularization. IFQL follows the default IQL configuration for value learning and uses sampling-based policy extraction. At test time, IFQL samples N candidate actions from the learned flow policy and selects the candidate with the highest Q-value. Thus, N denotes the number of test-time action samples. The environment-specific values of α and N listed in our tables are taken from the source evaluation.

Flow Q-Learning (FQL) (Park et al., 2025b). We follow the official FQL implementation, which consists of an MLP-based Q-function, a flow-matching behavioral policy, and a one-step policy for policy extraction. Unless otherwise specified, we use the default architectural and optimization settings from the original implementation. For the model-capacity experiments, we keep the Q-function depth fixed at four layers and vary the hidden dimension over $\{320, 512, 1536, 2944\}$.

Perceiver Actor-Critic (PAC) (Springenberg et al., 2024). We follow the state-based OGBench adaptation of PAC described by Dong et al. (2026a). The critic uses 32 latent tokens, dense encoders for state and action representations, and discretized value prediction. For each task, the value range extends from the minimum possible return to 0, with a bin width of 1. Following this adaptation, we disable the discrete action prediction component and use the FQL policy extraction procedure (Park et al., 2025b) instead. Thus, the PAC baseline preserves the Perceiver-based value architecture

while using the same policy extraction scheme as VTIQL. This comparison concerns the adapted PAC critic rather than the complete policy and value learning system of the original method.

TQL (Dong et al., 2026a). We use TQL as a Transformer-based Q-function baseline trained with standard Q-learning backups and attention entropy regularization. For the model-capacity experiments, we vary the Transformer hidden dimension over $\{128, 256, 512, 1024\}$ while keeping the remaining architectural settings fixed. For the five manipulation environments evaluated in the original TQL study, namely `cube-double`, `cube-triple`, `scene`, `puzzle-3x3`, and `puzzle-4x4`, we use the environment-specific target attention entropy values reported in the original paper. For the additional locomotion environments, including `antmaze` and `humanoidmaze`, which were not included in the original TQL evaluation, we determine the target entropy from the input dimensionality following the initialization rule proposed by TQL. Specifically, we compute the maximum attention entropy as $H_{\max} = \ln(1 + n_s + n_a)$, where n_s and n_a denote the state and action dimensions, respectively. We then set the target attention entropy to $\bar{H} = 0.8H_{\max}$. This procedure extends the entropy-target selection rule of TQL to the additional locomotion environments while preserving the original environment-specific settings for the manipulation tasks.

QTIQL. QTIQL is an architectural variant of VTIQL that parameterizes the Q-function with a Transformer and the state-value function with an MLP. For the model-capacity experiments, we vary the hidden dimension of the Transformer-based Q-function over $\{128, 256, 512, 1024\}$, following the same scaling configuration used for TQL. The value-learning and policy-extraction objectives are otherwise identical to those of VTIQL.

A.5 HYPERPARAMETERS

We include the default hyperparameters for VTIQL and the baseline methods in Table 3 and their hyperparameters in Table 4.

Table 3: Default training hyperparameters for VTIQL and baselines.

Hyperparameter	**Value**
Optimizer	AdamW (weight decay 0.01)
Training steps	1M
Batch size	256
Discount factor (γ)	0.99 (`antmaze-large`) 0.995 (default) 0.999 (`humanoidmaze-large`, `scene`, `puzzle-3x3`)
Expectile (τ_{exp})	0.7 (default), 0.9 (`cube-double, puzzle-4x4`)
Policy network dimensions	(512, 512, 512, 512)
Target smoothing coefficient (τ)	0.005
Actor learning rate	5×10^{-4} with cosine schedule
Critic learning rate	1×10^{-4} with cosine schedule
BC coefficient (α)	Environment-specific; see Table 4
Q aggregation (Q_{agg})	min (default), mean (`cube-double, puzzle-3x3, scene`)
Number of attention heads (H)	4 (default), 8 (`scene`)
Actor flow steps	10
Number of action samples	16

Table 4: **Hyperparameters for VTIQL and baselines.** The definitions of the method-specific hyperparameters are provided in Section A.4. For TQL, $\bar{H}$ denotes the target attention entropy.

	IQL	ReBRAC		FBRAC	IFQL	FQL	PAC	TQL		VTIQL
Domain or task	α	α_1	α_2	α	N	α	α	α	$\bar{H}$	α
antmaze-large	10	0.003	0.01	3	32	10	10	10	$((3.0, 3.0), (2.5, 2.5))$	10
antmaze-giant	10	0.003	0.01	10	32	10	10	10	$((3.0, 3.0), (2.5, 2.5))$	10
humanoidmaze-medium	10	0.01	0.01	30	32	30	30	30	$((3.5, 3.5), (3.0, 3.0))$	30
humanoidmaze-large	10	0.01	0.01	30	32	30	30	30	$((3.5, 3.5), (3.0, 3.0))$	30
cube-double-play	0.3	0.1	0	100	32	300	300	300	$((3.0, 3.0), (2.5, 2.5))$	300
cube-triple-play	10	0.03	0	100	32	300	300	300	$((3.5, 3.5), (3.0, 3.0))$	300
puzzle-3x3-play	10	0.3	0.001	100	32	1000	1000	1000	$((3.5, 3.5), (3.0, 3.0))$	1000
puzzle-4x4-play	3	0.3	0.01	300	32	1000	1000	1000	$((3.5, 3.5), (3.0, 3.0))$	1000
scene-play	10	0.1	0.001	100	32	300	300	300	$((3.5, 3.5), (3.0, 3.0))$	1000

B Additional Results and Experiments

B.1 Full Single Task Results

Table 5 reports the complete results on all 45 single task problems. We compare VTIQL with BC, IQL, ReBRAC, FBRAC, IFQL, FQL, PAC, and TQL. VTIQL achieves the highest environment level performance on **7** out of 9 environment configurations, including both locomotion and manipulation environments. The task level results provide a detailed comparison across the five tasks within each environment configuration.

B.2 Full Training Curves

Figures 6 and 7 report the training curves of TQL and VTIQL across the nine environment configurations. We report both the performance averaged over the five tasks within each environment and the individual training curves for all 45 tasks.

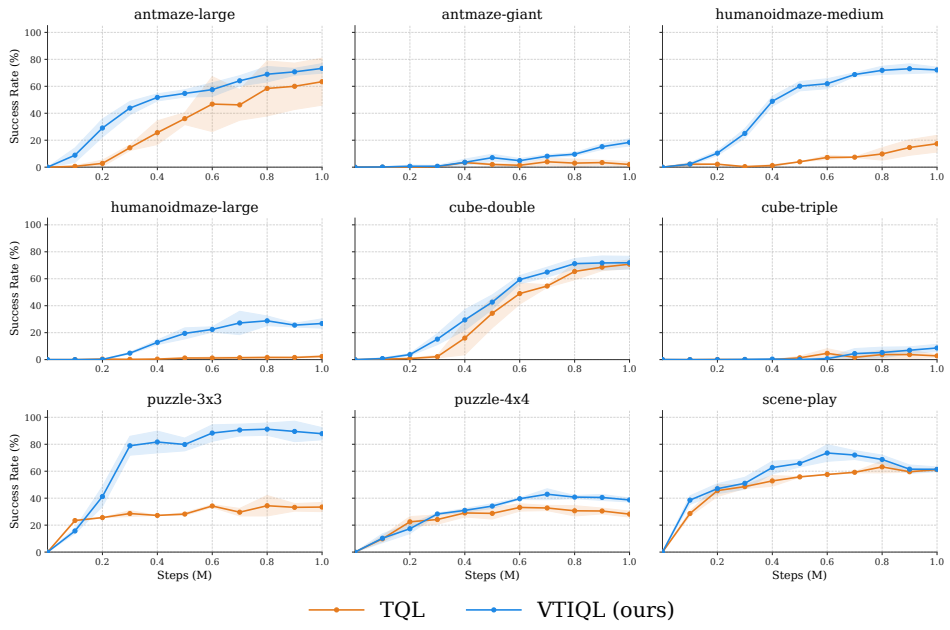


Figure 6: Training curves of TQL and VTIQL across environments.

Table 5: Performance comparison across all single-task environments. In average rows, task means and reported uncertainty values are averaged separately and rounded to the nearest integer; the latter do not represent the uncertainty of the aggregate mean.

	MLP Value						Transformer Value		
	BC	IQL	ReBRAC	FBRAC	IFQL	FQL	PAC	TQL	VTIQL (**ours**)
antmaze-large-navigate-task1	0 ± 0	48 ± 9	91 ± 10	70 ± 20	24 ± 17	80 ± 8	65 ± 5	82 ± 11	73 ± 5
antmaze-large-navigate-task2	6 ± 3	42 ± 6	88 ± 4	35 ± 12	8 ± 3	57 ± 10	25 ± 21	43 ± 19	41 ± 5
antmaze-large-navigate-task3	29 ± 5	72 ± 7	51 ± 18	83 ± 15	52 ± 17	93 ± 3	85 ± 5	91 ± 0	87 ± 5
antmaze-large-navigate-task4	8 ± 3	51 ± 9	84 ± 7	37 ± 18	18 ± 8	80 ± 4	64 ± 12	46 ± 43	77 ± 4
antmaze-large-navigate-task5	10 ± 3	54 ± 22	90 ± 2	76 ± 8	38 ± 18	83 ± 4	87 ± 5	41 ± 57	77 ± 3
antmaze-large-navigate (average)	11 ± 3	53 ± 11	81 ± 8	60 ± 15	28 ± 13	79 ± 6	65 ± 10	61 ± 26	71 ± 4
antmaze-giant-navigate-task1	0 ± 0	0 ± 0	27 ± 22	0 ± 1	0 ± 0	4 ± 5	0 ± 0	0 ± 0	3 ± 3
antmaze-giant-navigate-task2	0 ± 0	1 ± 1	16 ± 17	4 ± 7	0 ± 0	9 ± 7	9 ± 8	0 ± 0	28 ± 1
antmaze-giant-navigate-task3	0 ± 0	0 ± 0	34 ± 22	0 ± 0	0 ± 0	0 ± 1	0 ± 0	0 ± 0	0 ± 0
antmaze-giant-navigate-task4	0 ± 0	0 ± 0	5 ± 12	9 ± 4	0 ± 0	14 ± 23	3 ± 2	14 ± 7	26 ± 2
antmaze-giant-navigate-task5	1 ± 1	19 ± 7	49 ± 22	6 ± 10	13 ± 9	16 ± 28	8 ± 10	0 ± 0	16 ± 2
antmaze-giant-navigate (average)	0 ± 0	4 ± 2	26 ± 19	4 ± 4	3 ± 2	9 ± 13	4 ± 4	3 ± 1	15 ± 2
humanoidmaze-medium-navigate-task1	1 ± 0	32 ± 7	16 ± 9	25 ± 8	69 ± 19	19 ± 12	1 ± 0	1 ± 2	71 ± 3
humanoidmaze-medium-navigate-task2	1 ± 0	41 ± 9	18 ± 16	76 ± 10	85 ± 11	94 ± 3	86 ± 1	40 ± 25	68 ± 8
humanoidmaze-medium-navigate-task3	6 ± 2	25 ± 5	36 ± 13	27 ± 11	49 ± 49	74 ± 18	23 ± 17	17 ± 19	90 ± 5
humanoidmaze-medium-navigate-task4	0 ± 0	0 ± 1	15 ± 16	1 ± 2	1 ± 1	3 ± 4	16 ± 0	7 ± 6	39 ± 5
humanoidmaze-medium-navigate-task5	2 ± 1	66 ± 4	24 ± 20	63 ± 9	98 ± 2	97 ± 2	16 ± 9	5 ± 2	95 ± 2
humanoidmaze-medium-navigate (average)	2 ± 1	33 ± 5	22 ± 15	38 ± 8	60 ± 16	57 ± 8	29 ± 6	14 ± 11	73 ± 5
humanoidmaze-large-navigate-task1	0 ± 0	3 ± 1	2 ± 1	0 ± 1	6 ± 2	7 ± 6	0 ± 0	0 ± 0	49 ± 5
humanoidmaze-large-navigate-task2	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	2 ± 2
humanoidmaze-large-navigate-task3	1 ± 1	7 ± 3	8 ± 4	10 ± 2	48 ± 10	11 ± 7	7 ± 0	8 ± 3	16 ± 2
humanoidmaze-large-navigate-task4	1 ± 0	1 ± 0	1 ± 1	0 ± 0	1 ± 1	2 ± 3	1 ± 1	1 ± 0	35 ± 4
humanoidmaze-large-navigate-task5	0 ± 1	1 ± 1	2 ± 2	1 ± 1	0 ± 0	1 ± 3	2 ± 2	0 ± 0	35 ± 5
humanoidmaze-large-navigate (average)	0 ± 0	2 ± 1	3 ± 2	2 ± 1	11 ± 3	4 ± 4	2 ± 1	2 ± 1	27 ± 4
cube-double-play-task1	8 ± 3	27 ± 5	45 ± 6	47 ± 11	35 ± 9	61 ± 9	28 ± 16	95 ± 2	91 ± 8
cube-double-play-task2	0 ± 0	1 ± 1	7 ± 3	22 ± 12	9 ± 5	36 ± 6	36 ± 16	84 ± 8	83 ± 6
cube-double-play-task3	0 ± 0	0 ± 0	4 ± 1	4 ± 2	8 ± 5	22 ± 5	50 ± 11	60 ± 27	63 ± 18
cube-double-play-task4	0 ± 0	0 ± 0	1 ± 1	0 ± 1	1 ± 1	5 ± 2	10 ± 6	14 ± 6	21 ± 6
cube-double-play-task5	0 ± 0	4 ± 3	4 ± 2	2 ± 2	17 ± 6	19 ± 10	25 ± 21	84 ± 9	93 ± 3
cube-double-play (average)	2 ± 1	6 ± 2	12 ± 3	15 ± 6	14 ± 5	29 ± 6	30 ± 14	67 ± 10	70 ± 8
cube-triple-play-task1	1 ± 1	4 ± 4	1 ± 2	0 ± 0	2 ± 2	20 ± 6	1 ± 1	29 ± 16	30 ± 12
cube-triple-play-task2	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	1 ± 2	0 ± 0	0 ± 1	2 ± 1
cube-triple-play-task3	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	5 ± 4	4 ± 3
cube-triple-play-task4	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0
cube-triple-play-task5	0 ± 0	1 ± 1	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0
cube-triple-play (average)	0 ± 0	1 ± 1	0 ± 0	0 ± 0	0 ± 0	4 ± 2	0 ± 0	7 ± 4	7 ± 3
puzzle-3x3-play-task1	5 ± 2	33 ± 6	97 ± 4	63 ± 19	94 ± 3	90 ± 4	92 ± 4	96 ± 4	94 ± 3
puzzle-3x3-play-task2	1 ± 1	4 ± 3	1 ± 1	2 ± 2	1 ± 2	16 ± 5	17 ± 12	19 ± 12	93 ± 7
puzzle-3x3-play-task3	1 ± 1	3 ± 2	3 ± 1	1 ± 1	0 ± 0	10 ± 3	8 ± 4	7 ± 5	90 ± 9
puzzle-3x3-play-task4	1 ± 1	2 ± 1	2 ± 1	2 ± 2	0 ± 0	16 ± 5	14 ± 3	10 ± 4	98 ± 2
puzzle-3x3-play-task5	1 ± 0	3 ± 2	5 ± 3	2 ± 2	0 ± 0	16 ± 3	9 ± 3	30 ± 10	90 ± 3
puzzle-3x3-play (average)	2 ± 1	9 ± 3	22 ± 2	14 ± 5	19 ± 1	30 ± 4	28 ± 5	32 ± 7	93 ± 5
puzzle-4x4-play-task1	1 ± 1	12 ± 2	26 ± 4	32 ± 9	49 ± 9	34 ± 8	14 ± 7	59 ± 11	55 ± 5
puzzle-4x4-play-task2	0 ± 0	7 ± 4	12 ± 4	5 ± 3	4 ± 4	16 ± 5	13 ± 5	19 ± 6	22 ± 6
puzzle-4x4-play-task3	0 ± 0	9 ± 3	15 ± 3	20 ± 10	50 ± 14	18 ± 5	8 ± 7	43 ± 5	68 ± 3
puzzle-4x4-play-task4	0 ± 0	5 ± 2	10 ± 3	5 ± 1	21 ± 11	11 ± 3	3 ± 2	20 ± 3	37 ± 9
puzzle-4x4-play-task5	0 ± 0	4 ± 1	7 ± 3	4 ± 3	2 ± 2	7 ± 3	5 ± 3	16 ± 5	16 ± 4
puzzle-4x4-play (average)	0 ± 0	7 ± 2	14 ± 3	13 ± 5	25 ± 8	17 ± 5	9 ± 5	31 ± 6	40 ± 5
scene-play-task1	19 ± 6	94 ± 3	95 ± 2	96 ± 8	98 ± 3	100 ± 0	100 ± 0	100 ± 0	100 ± 0
scene-play-task2	1 ± 1	12 ± 3	50 ± 13	46 ± 10	0 ± 6	76 ± 9	54 ± 16	81 ± 14	95 ± 2
scene-play-task3	1 ± 1	32 ± 7	55 ± 16	78 ± 14	54 ± 19	98 ± 1	98 ± 2	98 ± 2	34 ± 8
scene-play-task4	2 ± 2	0 ± 1	3 ± 3	4 ± 4	0 ± 0	5 ± 1	4 ± 3	25 ± 17	79 ± 5
scene-play-task5	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	0 ± 0	6 ± 3
scene-play (average)	5 ± 2	28 ± 3	41 ± 7	45 ± 7	30 ± 6	56 ± 2	51 ± 4	61 ± 7	63 ± 4

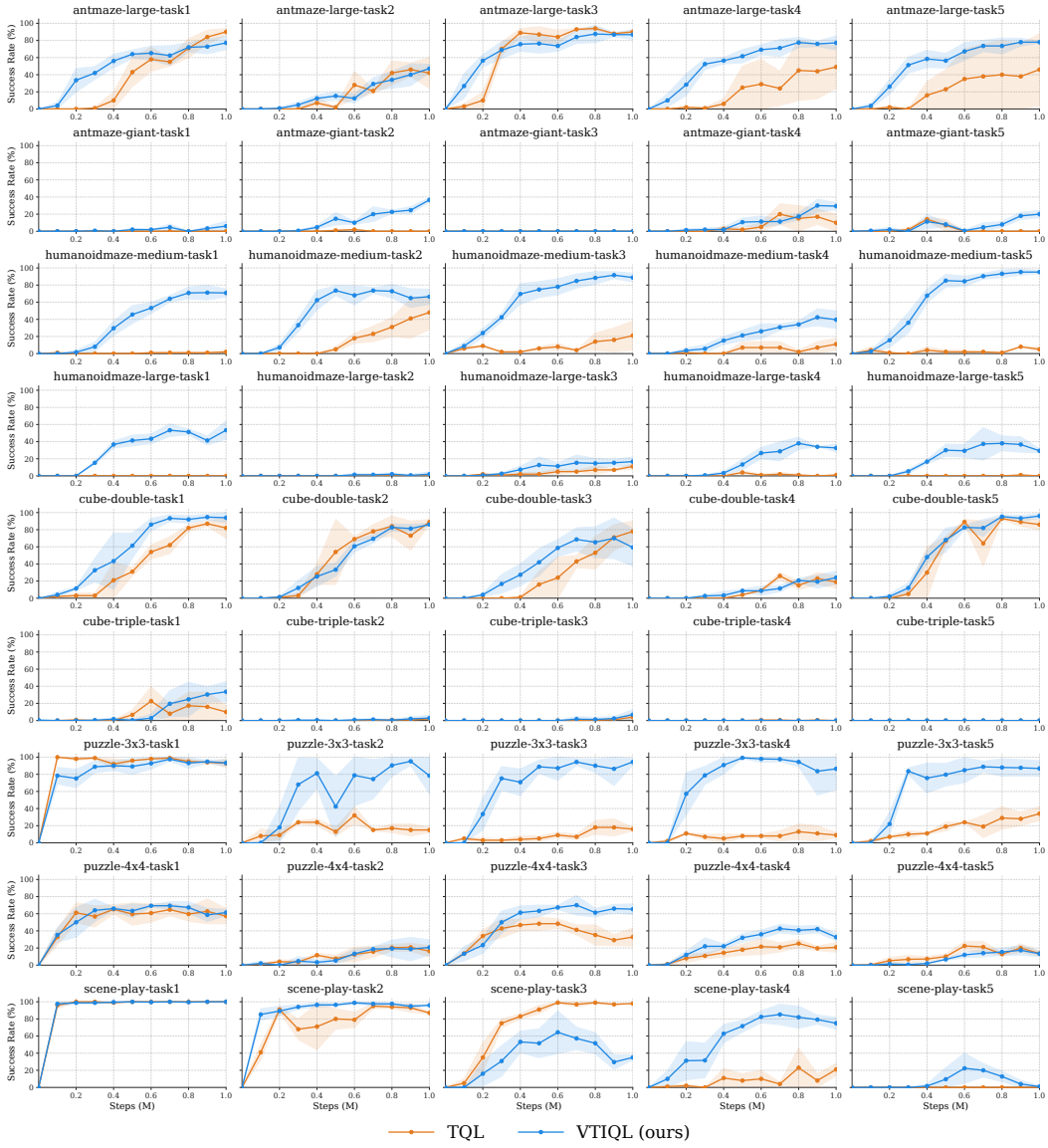


Figure 7: Per-task training curves of TQL and VTIQL across all environments.

B.3 Evaluation with AWR Policy Extraction

To evaluate VTIQL under a policy extraction method different from the DDPG+BC-style objective used in our main experiments, we compare IQL and VTIQL using advantage-weighted regression (AWR) with the same policy extraction procedure. We conduct the comparison on five single tasks from `cube-double`, representing manipulation, and five single tasks from `humanoidmaze-medium`, representing high-dimensional navigation.

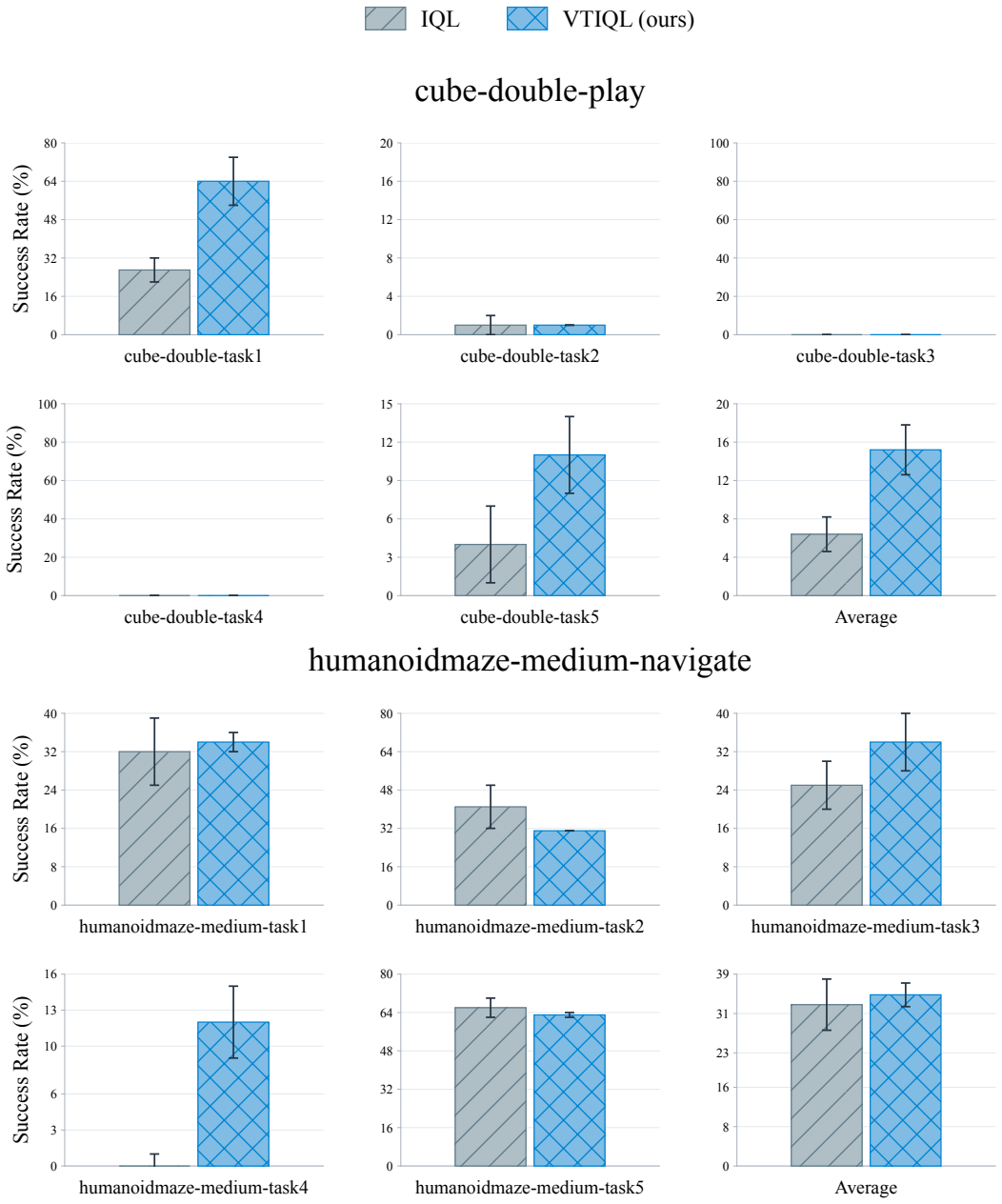


Figure 8: Success rates of IQL and VTIQL with AWR policy extraction on five single tasks each in `cube-double` and `humanoidmaze-medium`.

As shown in Figure 8, VTIQL achieves more than twice the success rate of IQL on tasks 1 and 5 in `cube-double`. On `humanoidmaze-medium`, VTIQL achieves a higher average success rate than IQL and obtains a non-zero success rate on task 4, for which IQL achieves zero success.

These results show that the performance improvement of VTIQL over IQL is preserved when both methods use AWR for policy extraction, indicating that the observed improvement is not specific to the DDPG+BC-style policy extraction objective used in our main experiments.

B.4 Comparison with Recent RL Methods

Recent RL methods modify value estimation or policy optimization through different learning objectives and model parameterizations. GFP (Nguimatsia Tiofack et al., 2026) incorporates value-based supervision into the training objective of a flow policy. Q-Flow (Doo et al., 2026) estimates value functions at intermediate states of the flow trajectory and uses these estimates to guide policy optimization. DriftQL (Houssaini et al., 2026) regularizes policy optimization with a drift-based behavior constraint while using critic estimates for policy improvement. Value Flows (Dong et al., 2026b) parameterizes return distributions with flow matching and uses the resulting return distribution for value estimation. Although their methodological scope differs from that of VTIQL, we additionally compare against these recent methods on the state-based single-task OGBench tasks to assess VTIQL alongside recent advances in offline RL.

Table 6: **Comparison with recent RL methods on OGBench.** Success rates (%) are averaged over the five tasks in each environment.

Environment (5 tasks each)	GFP	Q-Flow	DriftQL	Value Flows	VTIQL (**ours**)
`antmaze-large`	**94** $\pm$ 2	89 $\pm$ 5	92 $\pm$ 4	-	71 $\pm$ 4
`antmaze-giant`	28 $\pm$ 9	40 $\pm$ 4	**60** $\pm$ 2	-	15 $\pm$ 2
`humanoidmaze-medium`	72 $\pm$ 3	**83** $\pm$ 4	62 $\pm$ 2	-	73 $\pm$ 5
`humanoidmaze-large`	18 $\pm$ 10	8 $\pm$ 2	5 $\pm$ 8	-	**27** $\pm$ 4
`cube-double`	47 $\pm$ 2	36 $\pm$ 3	25 $\pm$ 2	69 $\pm$ 4	**70** $\pm$ 8
`puzzle-3x3`	23 $\pm$ 2	49 $\pm$ 3	35 $\pm$ 3	87 $\pm$ 13	**93** $\pm$ 5
`puzzle-4x4`	26 $\pm$ 2	29 $\pm$ 2	27 $\pm$ 3	27 $\pm$ 4	**40** $\pm$ 5
`scene`	54 $\pm$ 3	60 $\pm$ 2	**74** $\pm$ 4	59 $\pm$ 4	63 $\pm$ 4

Table 6 reports average success rates for each environment using results reported in the corresponding papers. We use the evaluation protocol adopted by each baseline rather than re-evaluating all methods under a unified protocol. For GFP, we report the primary one-step actor π_θ rather than the auxiliary value-aware behavior cloning flow policy π_ω. For GFP, Q-Flow, DriftQL, and Value Flows, we use the reported performance averaged over evaluations at 800K, 900K, and 1M training steps. For Q-Flow, we use the standard configuration reported in the paper. We include only environments with directly corresponding state-based singletask OGBench evaluations.

Across these environments, VTIQL achieves competitive performance relative to the recent methods considered here. The compared methods introduce additional algorithmic mechanisms, including value-weighted policy supervision, intermediate-flow value optimization, drift-based behavior regularization, and flow-based distributional return modeling. VTIQL does not introduce these mechanisms. It retains the in-sample value-learning and FQL policy-extraction objectives, while parameterizing the V-function with a Transformer and the Q-function with an MLP. The results indicate that this value function parameterization can achieve competitive performance without modifying the underlying value-learning or policy-extraction objectives.