# From Weak Data to Strong Policy: Q-Targets Enable Provable In-Context Reinforcement Learning

**Yichen Lin**[1,*] **Xuyuan Xiong**[1,*] **Xue Wang**[2] **Xiangfu Meng**[1]
Mike Mingcheng Wei[3] Tao Yao[1,†]
[1]Shanghai Jiao Tong University, Shanghai, China
[2]Alibaba Group
[3]School of Management, University at Buffalo, Buffalo, NY, USA
[1]{linyichen,xxy2021,mengxiangfu,taoyao}@sjtu.edu.cn
[2]wxie91@gmail.com [3]mcwei@buffalo.edu

## Abstract

Existing in-context reinforcement learning methods mainly pretrain Transformers with supervised behavior-prediction objectives. This enables task inference from context, but makes the learned policy strongly depend on the quality of offline actions: when trajectories are weak or suboptimal, imitation itself becomes a biased learning signal. We propose Q-Target Pretrained Transformers (QTPT), which keeps the context-conditioned Transformer architecture but replaces behavior cloning with a Bellman-style Q-target objective. QTPT therefore learns to use rewards and transitions in the context to estimate action values, rather than simply imitating the behavior policy. We theoretically analyze QTPT in stochastic linear bandits and finite-horizon MDPs, showing stronger robustness to data quality than supervised pretraining. Empirically, QTPT improves over supervised behavior prediction on controlled RL benchmarks with random or suboptimal data, and we examine extensions to D4RL Kitchen and AntMaze. Supplementary experiments evaluate backbone robustness, meta-RL comparisons, task-coherent context, and unsupported-action value overestimation. These comparisons distinguish the benefits of Q-target pretraining from the remaining limitations of offline coverage.[1]

## 1 Introduction

Recent advances in large language models have demonstrated their remarkable ability to perform various tasks in a zero-shot or few-shot manner using in-context learning [Brown et al., 2020, Garg et al., 2022]. Building upon this paradigm, transformer-based models have been successfully applied to Reinforcement Learning (RL) settings, showing strong In-Context Reinforcement Learning (ICRL) capabilities [Laskin et al., 2022, Lee et al., 2023, Lin et al., 2024]. These models can implicitly capture temporal dependencies within sequences of state-action-reward tuples, enabling them to generalize and make decisions in unseen environments.

In-context RL requires inferring the current task from interaction history and selecting actions using that information. Supervised approaches include Algorithm Distillation [Laskin et al., 2022], which distills learning trajectories, and Decision Pretrained Transformer [Lee et al., 2023], which uses optimal-action supervision. These supervision sources differ: our matched SPT baseline predicts the behavior actions recorded in the offline data, so when those actions are weak or suboptimal, its targets encourage imitation of weak decisions.

Accepted at NeurIPS 2026. *Equal contribution. †Corresponding author.
[1]Code: https://github.com/lyc51758/qtpt-finalcode.

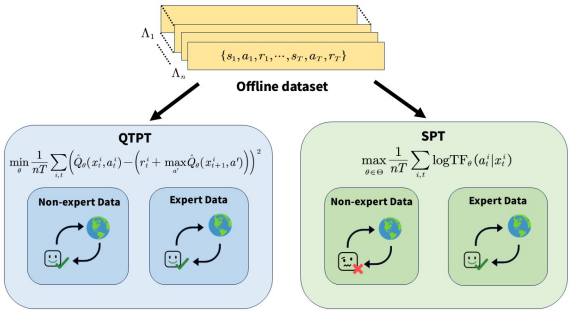



Figure 1: Comparison of Q-target Pretrained Transformers (QTPT) and Supervised Pretrained Transformers (SPT) for in-context RL. While SPT requires expert demonstrations for good performance, QTPT maintains robust performance even when trained on suboptimal data.

To address this objective-level limitation, we propose Q-Target Pretrained Transformers (QTPT), a pretraining framework that replaces supervised behavior cloning with a Bellman-style Q-learning objective. The context still provides the information needed for task adaptation, but the learning signal comes from rewards, transitions, and bootstrapped value estimates rather than behavior actions alone. As a result, QTPT learns a context-conditioned Q-function from offline datasets even when they are generated by suboptimal policies (Figure 1), reducing the reliance on expert trajectories or optimal action labels and improving robustness under weak offline data.

Theoretically, we give a Transformer approximation construction and a suboptimality upper bound with two distinct components: a sample bias induced by the behavior policy, and a model bias governed by the Rademacher complexity and the size of the training dataset. This separation isolates the two effects and sheds light on how QTPT balances leveraging available data against mitigating model bias during pretraining.

We evaluate QTPT on controlled RL benchmarks, D4RL Kitchen and AntMaze, and a language-model reasoning adaptation. Matched comparisons with supervised pretraining test the effect of replacing behavior labels with Q-targets, and ablations examine context and the backup target.

The major contributions of this work are summarized as follows:

1. We propose QTPT, a context-conditioned Q-target pretraining framework for in-context RL. Unlike supervised pretraining, QTPT does not treat behavior actions as ground-truth labels; it trains the Transformer to estimate action values from contextual rewards and transitions. Integrating Q-learning with Transformers enables end-to-end value estimation inside a sequence model; this shift from behavior prediction to Q-target pretraining is our central algorithmic contribution.
2. We derive suboptimality upper bounds for stochastic linear bandits and finite-horizon MDPs, separating sample bias from model bias. The general-MDP analysis contains a worst-case $\widetilde{\mathcal{O}}(\sqrt{L}\,T^2 n^{-1/4})$ statistical term, together with class-complexity and approximation terms. The bandit model-bias component scales as $\widetilde{\mathcal{O}}(\sqrt{T/n})$ under its stated conditions; this rate concerns that component, not the combined bound.
3. We evaluate the roles of supervision, context, and backup construction. The experiments show strong gains over behavior prediction on random data, test the pattern across three decoder families, and directly measure unsupported-action value overestimation. Additional experiments probe meta-RL comparisons, conservative regularization, and task-coherent versus mixed-task context, each under its own protocol rather than pooled into one ranking.

### 1.1 Related Work

**Q-Learning**. Q-learning is a foundational off-policy reinforcement learning algorithm that learns optimal policies through Bellman updates of Q-values [Watkins and Dayan, 1992, Sutton, 2018]. Subsequent advances have improved its efficiency, especially in the exploration–exploitation trade-off: Upper-Confidence Bound (UCB) exploration achieves regret bounds on par with model-based methods [Jin et al., 2018, Zanette and Brunskill, 2019], and optimistic Q-learning [Even-Dar and Mansour, 2001, Yang et al., 2021] further reduces regret under appropriate conditions. Deep Q-Networks (DQN) [Mnih et al., 2015] integrate Q-learning with deep learning to achieve human-level performance in complex tasks, and Kapturowski et al. [2018] leverage LSTM networks to capture temporal dependencies.

Recent advances have adapted Q-learning to large-scale sequence models. Q-SFT [Hong et al., 2025] reframes Q-learning as a supervised fine-tuning problem, optimizing a weighted cross-entropy loss to approximate Q-values and providing theoretical analysis of convergence. Similarly, the Q-learning Decision Transformer [Yamagata et al., 2023] employs a two-step approach: it first relabels reward-to-go (RTG) values using Q-learning, then trains a Decision Transformer [Chen et al., 2021] on these relabeled trajectories. QTPT instead trains a history-conditioned Q-function directly with sampled Bellman targets. The backup is constructed by the training algorithm; it is not an architectural claim that every attention layer performs a Bellman update. The appendix construction concerns representational capacity and does not guarantee that gradient-based training discovers that construction.

**Offline Reinforcement Learning**. Offline Reinforcement Learning (RL) focuses on learning optimal policies from pre-collected datasets, without additional interactions with the environment [Levine et al., 2020, Matsushima et al., 2020]. One of the major challenges in offline RL is the distribution drift between the behavior policy, which generates the dataset, and the learned policy [Levine et al., 2020, Kostrikov et al., 2021, Rashidinejad et al., 2021]. Techniques such as conservative value function estimation, policy constraints, and regularization have been developed to mitigate this challenge [Wu et al., 2019, Kumar et al., 2020, Kidambi et al., 2021, Jin et al., 2022, Dong et al., 2023, Hu et al., 2024, Qu et al., 2024, Setlur et al., 2024, Park et al., 2024]. Unlike Transformer-based ICRL methods such as AD [Laskin et al., 2022] and DPT [Lee et al., 2023], which study whether supervised sequence models can infer tasks from high-quality trajectories, we ask how changing the pretraining objective affects robustness when the offline behavior data are weak or suboptimal. The advantages of offline reinforcement learning over behavior cloning (imitation learning) are studied in Kumar et al. [2022]. Unlike this stream of work, we adopt a *meta-learning framework* to generalize to *unseen* RL tasks after pretraining, while offline RL generally focuses on solving the *same* RL tasks from which the offline datasets were originally collected. QTPT combines Bellman-based value learning with a context-conditioned sequence model evaluated without test-time parameter updates. CQL adds conservative value regularization, whereas IQL uses an in-sample value estimate in its critic backup and advantage-weighted policy extraction (IQL is not Bellman-free); Q-SFT and QDT use value information to supervise sequence-model policies. Appendix C.5 distinguishes these learning mechanisms from the models' input and deployment protocols.

**In-Context Learning and Reinforcement Learning**. Without test-time parameter updates, in-context learning (ICL) adapts its predictions or decisions to a new task by conditioning on context observed at inference time [Brown et al., 2020, Min et al., 2022, Garg et al., 2022]. Mechanistic explanations of ICL include implicit statistical inference and gradient-based learning implemented by attention [Akyürek et al., 2022, Bai et al., 2023, Von Oswald et al., 2023, Zhang et al., 2023]. In-Context Reinforcement Learning (ICRL) addresses sequential decision-making problems by inferring optimal policies from past trajectories. In the realm of ICRL, Wang et al. [2024a] explore how transformers can implement Temporal Difference (TD) learning directly in the forward pass; see Moeini et al. [2025] for a comprehensive survey. While existing work mainly studies ICRL under the supervised pretraining paradigm (e.g., Lin et al. [2024]), our approach uses Q-learning as the pretraining strategy, explicitly minimizing in-context temporal difference errors and thereby aligning the pretraining objective with downstream in-context decision-making, especially under noisy or suboptimal data.

**Meta-RL and Belief-State Control**. QTPT shares the goal of adaptation across tasks with $RL^2$ [Duan et al., 2016] and PEARL [Rakelly et al., 2019]; these are related adaptation problems, not disjoint definitions of ICRL, that differ in data acquisition: $RL^2$ meta-trains recurrent policies online and PEARL combines inferred task context with replay-based actor–critic learning, whereas QTPT uses a fixed offline trajectory corpus. In a belief-MDP or Bayes-adaptive MDP, history determines a belief over latent states or task parameters [Kaelbling et al., 1998, Duff, 2002]; QTPT feeds history to an approximate value function without explicitly computing that posterior. Its contribution is the offline Q-target pretraining objective and its analysis, not the use of history as a belief proxy.

## 2 Model Setting

### 2.1 Preliminaries

We consider a set of decision-making environments $\mathcal{M}$, each operating over $T$ rounds with shared state and action spaces $(\mathcal{S}, \mathcal{A})$. Each environment $M \in \mathcal{M}$ has a unique transition model $\mathbb{P}_M$ :

**Algorithm 1** Q-target Pretrained Transformers (QTPT)

**Input**: an Offline Dataset $D$ collected in $\Lambda_{\text{train}}$, and the horizon $T$.
// Pretraining model on dataset
Randomly initialize $\hat{Q}_\theta$
**while** not converged **do**
  Sample a batch of $\mathcal{B}$ trajectories from $D$
  Compute the Temporal Difference (TD) target $y_t^i = r_t^i + \max_{a' \in \mathcal{A}} \hat{Q}_{\theta,t+1}(s_{t+1}^i, a', D_t^i)$
  Compute the loss: $\mathcal{L}_{\mathcal{B}}(\hat{Q}_\theta) = \frac{1}{|\mathcal{B}|T} \sum_{i=1}^{|\mathcal{B}|} \sum_{t=1}^{T} \left( \hat{Q}_{\theta,t}(s_t^i, a_t^i, D_{t-1}^i) - y_t^i \right)^2$
  Backpropagate to update $\theta$
**end while**
// Online test-time deployment
Initialize an empty dataset $D$ and sample environment $M \sim \Lambda_{\text{test}}$
**for** $t$ in horizon $T$ **do**
  Choose $a_t \in \arg\max_a \hat{Q}_\theta(s_t, a, D)$
  Execute $a_t$ and observe $r_t, s_{t+1}$
  Update $D$ with $(s_t, a_t, r_t)$
**end for**

$\mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, initial state distribution $\eta_M \in \Delta(\mathcal{S})$, and a reward function $r_M : \mathcal{S} \times \mathcal{A} \to \Delta(\mathbb{R})$. The agent's uncertainty is captured by environment priors $\Lambda_{\text{train}} \in \Delta(\mathcal{M})$ and $\Lambda_{\text{test}} \in \Delta(\mathcal{M})$ for training and testing respectively. This framework represents various scenarios, including $T$ rounds of multi-armed bandit problems and $K$ episodes of $H$-step MDPs with $T = KH$. The training stage uses an offline dataset $D$ consisting of $n$ offline trajectories $\{D_T^i = (s_1^i, a_1^i, r_1^i, \ldots, s_T^i, a_T^i, r_T^i)\}_{i=1}^n$, which have been collected by the behavior policy $\pi_\beta$.

We denote a partial interaction trajectory—comprising the sequence of observed states, actions, and rewards—by $D_t = \{s_1, a_1, r_1, \cdots, s_t, a_t, r_t\} \in \mathcal{T}_t = (\mathcal{S} \times \mathcal{A} \times \mathbb{R})^t$. For convenience, we define the *context state* as $x_t = D_{t-1} \cup \{s_t\}$, which lies in the space $\mathcal{X}_t = \mathcal{T}_{t-1} \times \mathcal{S}$. This representation captures the cumulative interaction history up to round $t-1$ and the current state $s_t$. Analogous to standard state transitions in reinforcement learning, the context state evolves as $x_{t+1} = x_t \cup \{a_t, r_t, s_{t+1}\}$ with $r_t \sim r_M(\cdot|s_t, a_t)$ and $s_{t+1} \sim \mathbb{P}_M(\cdot|s_t, a_t)$. We formalize this dynamic with the *context transition model*, defined as $\mathbb{T}_{M,t} : \mathcal{X}_t \times \mathcal{A} \to \Delta(\mathcal{X}_{t+1})$, which maps a given context-action pair to a distribution over the next context state. A *policy* $\pi$ maps a context state $x_t \in \mathcal{X}_t$ to a distribution over the actions $\pi(\cdot|x_t) \in \Delta(\mathcal{A})$.

For the finite-horizon analysis we use the undiscounted setting, as in Lin et al. [2024]; implementation-specific discounting is stated separately. We define the value function $V_{M,t}^\pi : \mathcal{X}_t \to \mathbb{R}$, the Q-function $Q_{M,t}^\pi : \mathcal{X}_t \times \mathcal{A} \to \mathbb{R}$ for policy $\pi$, and the Bellman operator $\Gamma : \mathbb{R}^{\mathcal{X} \times \mathcal{A}} \to \mathbb{R}^{\mathcal{X} \times \mathcal{A}}$ as follows:

$$V_{M,t}^\pi(x_t) = \mathbb{E}\Big[\sum_{h=t}^T r_h \,\Big|\, x_t\Big], \tag{1}$$

$$Q_{M,t}^\pi(x_t, a_t) = \mathbb{E}\left[r_M(s_t, a_t)\right] + \mathbb{E}_{x_{t+1} \sim \mathbb{T}_{M,t}(\cdot|x_t, a_t)}\left[V_{M,t+1}^\pi(x_{t+1})\right], \tag{2}$$

$$(\Gamma Q_{M,t})(x_t, a_t) = \mathbb{E}[r_M(s_t, a_t)] + \mathbb{E}_{x_{t+1} \sim \mathbb{T}_{M,t}(\cdot|x_t, a_t)}\left[\max_{a'} Q_{M,t+1}(x_{t+1}, a')\right]. \tag{3}$$

Finally, the optimal value function and Q-function are given by $V_{M,t}^*(x_t) = \max_\pi V_{M,t}^\pi(x_t)$ and $Q_{M,t}^*(x_t, a_t) = \max_\pi Q_{M,t}^\pi(x_t, a_t)$ for $\forall x_t \in \mathcal{X}_t,\ a_t \in \mathcal{A}$ respectively.

### 2.2 Algorithm

The Q-target Pretrained Transformer (QTPT) is a parameter-shared causal decoder: $(\hat{Q}_1, \ldots, \hat{Q}_T)$ denotes time-indexed outputs, not $T$ independently stored networks. History and current-state inputs are embedded, and an action-value readout produces $\hat{Q}_t(x_t, a; \theta)$. All training tasks update the same parameters; the context supplied to an individual prediction remains task-coherent. To learn from the offline dataset $D$, we utilize a class of sequence functions $\hat{\mathcal{Q}} = (\hat{\mathcal{Q}}_1 \times \cdots \times \hat{\mathcal{Q}}_T)$, where each $\hat{Q}_t \in \hat{\mathcal{Q}}_t$ approximates the optimal Q-value function at time step $t$; a Transformer is one such model of sequential $Q$ functions. Denoting the full Q-function approximator by $\hat{Q} = (\hat{Q}_1, \ldots, \hat{Q}_T) \in \hat{\mathcal{Q}}$, we optimize

$$\min_{\theta\in\Theta} \quad \frac{1}{nT}\sum_{i=1}^{n}\sum_{t=1}^{T}\Big[\hat{Q}_t(x_t^i, a_t^i;\theta) - \big(r_t^i + \max_{a'}\hat{Q}_{t+1}(x_{t+1}^i, a';\theta)\big)\Big]^2. \tag{4}$$

where $n$ is the number of trajectories in $D$ and the approximate Q-value function is parameterized by $\theta$. Algorithm 1 states the full procedure: pretraining minimizes the difference between predicted Q-values and bootstrapped targets, and at test time actions are selected greedily while $\theta$ remains fixed and $x_t = D_{t-1} \cup \{s_t\}$ grows. We set $Q_{T+1} = 0$ at the terminal step. Algorithm 1 and Eq. (4) display the idealized hard-max objective; practical targets are detached from the gradient. Our implementation stabilizes this objective with a periodically synchronized target network and a softmax-weighted backup, specified in Appendix D.1; these are stabilization mechanisms, not an explicit pessimistic objective.

**A batch is not one context.** Sampling trajectories from many tasks trains a shared history-to-value mapping, but does not make every trajectory evidence about a given query's task: two latent tasks can share observations yet have opposite rewarding actions, so their unlabeled pooled histories do not identify the active task, whereas rewards observed within the active trial do. Same-task episodes help when their identity is known and they are also available at test time; the setup permits a task-coherent trial of $K$ episodes ($T = KH$), not adding unrelated tasks to that trial's context (tested in Appendix D.5).

### 2.3 Analysis Framework

Each approximated function $\hat{Q}_t$ induces a time-dependent greedy policy $\pi_{\hat{Q}_t}$, and the overall policy $\pi_{\hat{Q}}$ is the sequence $\{\pi_{\hat{Q}_t}\}_{t=1}^T$. Let $\mathcal{L}_n(Q) = \mathbb{E}_{D_T^n\sim\pi_\beta}[(Q(x,a) - (\Gamma Q)(x,a))^2]$ denote the Bellman-error objective induced by the offline dataset. We use $\hat{Q}_n^* = \arg\min_Q \mathcal{L}_n(Q)$ for the ideal Bellman-error minimizer over the reference value-function class, and $\tilde{Q}_n = \arg\min_{Q\in\hat{\mathcal{Q}}} \mathcal{L}_n(Q)$ for the best solution realizable by the Transformer hypothesis class $\hat{\mathcal{Q}}$, which QTPT approximates in practice by training a parameterized Transformer. We use $\epsilon_{\mathrm{tf}} \geq 0$ to denote the irreducible approximation floor induced by restricting the Bellman-error minimization to $\hat{\mathcal{Q}}$: it is zero under Transformer realizability, but need not vanish for a fixed misspecified Transformer class. The approximation result is a capacity/existence statement, not a guarantee of parameter recovery by the optimizer. Consistency requires the approximation error to vanish, for example under realizability or an appropriate growing-capacity sequence; a fixed nonzero floor cannot be hidden in a vanishing rate. The induced policy $\pi_{\tilde{Q}_n}$ greedily selects actions per $\{\pi_{\tilde{Q}_{n,t}}\}_{t=1}^T$ (omitting $\theta$ for readability). We define the expected cumulative reward of policy $\pi$ as $\mathbb{E}_{M\sim\Lambda,x_1\sim\eta_M}\left[V_{M,1}^{\pi}(x_1)\right]$ and the optimal one as $\mathbb{E}_{M\sim\Lambda,x_1\sim\eta_M}\left[V_{M,1}^{*}(x_1)\right]$. The suboptimality gap between the optimal policy $\pi^*$ and the learned policy $\pi_{\tilde{Q}_n}$ decomposes into sample bias and model bias:

$$\mathsf{Subopt}_\Lambda(\pi^*, \pi_{\tilde{Q}_n}) = \underbrace{\mathsf{Subopt}_\Lambda(\pi^*, \pi_{\hat{Q}_n^*})}_{\text{Sample Bias}} + \underbrace{\mathsf{Subopt}_\Lambda(\pi_{\hat{Q}_n^*}, \pi_{\tilde{Q}_n})}_{\text{Model Bias}}, \tag{5}$$

Sample bias arises from the finite offline dataset and behavior policy, which may not fully represent the true environment dynamics. Model bias stems from the approximation and optimization limits of the Transformer hypothesis class relative to the ideal Bellman-error minimizer.

## 3 Main Analysis and Results

To streamline the analysis of Eq. (5), we denote $d_{M,t}^{\pi^*}(x_t, a_t)$ as the marginal distribution at time $t$ for the optimal policy in the environment $M$, and $d_{M,t}^{\pi_\beta}(x_t, a_t)$ for any behavior policy $\pi_\beta$. Define the marginal occupancy as $d_{M,t}^{\pi}(x) = \sum_a d_{M,t}^{\pi}(x,a)$. The analysis follows Eq. (5): Proposition 3.2 controls sample bias; Propositions 3.3 and 3.4 control model bias; Proposition 3.5 combines them. The coverage parameter $L$, the reciprocal of the positive-occupancy lower bound below, weakens the guarantee through $\sqrt{L}$ independently of whether behavior actions have high reward. We state the required assumptions explicitly:

**Assumption 3.1** (Bounded Rewards and Distributional Coverage)**.** We assume the reward function $r_M$ and the marginal state-action distributions $d_{M,t}^{\pi^*}$ and $d_{M,t}^{\pi_\beta}$ satisfy the following conditions:

(a) **(Bounded Rewards)** The reward function is uniformly bounded: $|r_M(s,a)| \leq 1$ for all $(s,a) \in \mathcal{S} \times \mathcal{A}$.
(b) **(Optimal Policy Concentrability)** The behavior policy $\pi_\beta$ provides sufficient support for the optimal policy $\pi^*$: if $d^{\pi^*}_{M,t}(x_t, a_t) > 0$, then $d^{\pi_\beta}_{M,t}(x_t, a_t) > 0$, for all $t \in [T]$.
(c) **(Lower-Bounded Occupancy)** The occupancy density under $\pi_\beta$ is bounded on its support:

$$L^{-1} := \inf_{(x_t,a_t):\, d^{\pi_\beta}_{M,t}(x_t,a_t)>0} d^{\pi_\beta}_{M,t}(x_t, a_t), \quad \forall t \in [T].$$

The bounded-reward assumption controls the remaining-return range. Optimal Policy Concentrability assumes that $d^{\pi_\beta}$ covers the trajectory of some optimal policy $\pi^*$, and Lower-Bounded Occupancy ensures that the positive occupancy density under $\pi_\beta$ is bounded away from 0 on its support; the infimum excludes zero-occupancy context–action pairs, and neither statement guarantees a small $L$ in a large history space. These two conditions (Assumptions 4.1 and 4.2 in Nguyen-Tang et al. [2023]) ensure that an optimal policy is statistically learnable from offline data without the all-policy uniform coverage considered in batch reinforcement learning (see Assumption 1 in Duan et al. [2021] and Chen and Jiang [2019]).

### 3.1 Bound of Sample Bias

Denote the function bound of $Q \in \hat{\mathcal{Q}}$ as $B_Q = \sup_{x,a} \|Q(x,a) - (\Gamma Q)(x,a)\|_2$. We use $\mathcal{L}_n$ to denote the empirical loss calculated on $n$ samples, which corresponds to $\mathcal{L}_\mathcal{B}$ in Algorithm 1 on a mini-batch $\mathcal{B}$. We can now state the sample bias bound for QTPT.

**Proposition 3.2.** ***(Bound of Sample bias in* QTPT*)*** *Suppose Assumption 3.1 holds and there exists a constant $B > 0$ such that $B_{\hat{Q}^*_n} \leq B$ for all tuples $(D^i_{t-1}, s^i_t, a^i_t, y^i_t)$. Then, with probability at least $1-\delta$, we have*

$$\mathsf{Subopt}_\Lambda(\pi^*, \pi_{\hat{Q}^*_n}) \leq \mathcal{O}\left(\sqrt{L}\,T^2\left[\left(\frac{2\log(2T/\delta)}{n}\right)^{1/4} + \frac{c(\hat{\mathcal{Q}}, n)}{\sqrt{n}}\right] + \sqrt{L}\,TB\right),$$

*where* $c(\hat{\mathcal{Q}}, n) = \sqrt{\max\{\sqrt{n\log|\hat{\mathcal{Q}}|}, \log|\hat{\mathcal{Q}}|\}}$.

The explicit $T^2$ dependence is a conservative worst-case upper bound, not a proven lower bound for sequence-history methods: range-based concentration for squared Bellman error contributes $\widetilde{\mathcal{O}}(T^2 n^{-1/2})$, and taking a square root and accumulating across $T$ stages produces $\widetilde{\mathcal{O}}(\sqrt{L}\,T^2 n^{-1/4})$. The closest full-history ICRL analysis also has a $T^2$ pretraining term [Lin et al., 2024, Theorems 6 and 13]. More favorable offline-RL rates rely on additional structure and different algorithmic machinery, e.g., OPDVR exploits stationary tabular transitions and double variance reduction [Yin et al., 2021, Theorems 3.2 and 4.1]; these bounds are not interchangeable, and improving our exponent under the present assumptions remains open. The term $\sqrt{L}\,TB$ retains Bellman approximation error, which does not vanish merely because the sample size increases; controlling it requires an appropriate value-function class. Similarly, the class-complexity factor is part of the guarantee, so a polynomial explicit horizon factor alone does not establish polynomial complexity for an unrestricted history representation.

### 3.2 Bound of Model Bias

We start with the stochastic linear bandit. At each time step $t \in [T]$, the agent selects an action $a_t \in \mathcal{A}_t \subset \mathbb{R}^d$, where without loss of generality $\|a_t\|_2 \leq 1$, and receives a reward $r_t = \langle a_t, w^* \rangle + \epsilon_t$, where $w^*$ is an unknown parameter vector and $\epsilon_t$ is i.i.d. noise with bounded variance. With the Gram matrix $G_t = \sum_{i=1}^t a_i a_i^\top$, Proposition 3.3 bounds the $TF_\theta$ solved via Eq. (4).

**Proposition 3.3.** ***(Bound of Model Bias in* QTPT *on Stochastic Linear Bandit)*** *If the minimum eigenvalue of the Gram matrix $G_t$ satisfies $\lambda_{min}(G_t) \geq \alpha n t$ for some constant $\alpha > 0$, then for the Transformer $TF_\theta(\cdot)$ at appropriate scale, with probability at least $1-\delta$, it satisfies*

$$\mathsf{Subopt}(\pi_{\hat{Q}^*_n}, \pi_{\bar{Q}_n}) \leq \epsilon_{\mathrm{tf}} + \mathcal{O}\left(\sigma\sqrt{T}\sqrt{\frac{d\log T + \log(1/\delta)}{\alpha n}}\right).$$

Under the stated Gram-matrix condition, the finite-sample term in Proposition 3.3 decreases with $n$ and depends on dimension, horizon, noise, and coverage. For a fixed misspecified Transformer, $\epsilon_{\mathrm{tf}}$ remains, so this is not an unconditional consistency or parameter-recovery claim. For finite-horizon MDPs, we bound the model bias term in Proposition 3.4.

**Proposition 3.4.** *(**Bound of Model Bias in** **QTPT** **on Finite-Horizon MDP**) With probability at least $1-\delta$, we have*

$$\mathsf{Subopt}_{\Lambda}(\pi_{\hat{Q}_n^*}, \pi_{\tilde{Q}_n}) \leq 2T\sqrt{\epsilon_{\mathrm{tf}}} + \mathcal{O}\left(T^2\left(\frac{1}{\sqrt{n}} + \left(\frac{2\log(2T/\delta)}{n}\right)^{1/4}\right)\right).$$

Proposition 3.4 separates an $n^{-1/2}$ complexity term from an $n^{-1/4}$ confidence term and retains the approximation contribution $2T\sqrt{\epsilon_{\mathrm{tf}}}$. The logarithmic confidence dependence is displayed explicitly (statements suppressing it use fixed confidence); a faster excess-risk estimate would improve the statistical rate but is not assumed.

### 3.3 Final Bound

Combining the sample bias bound (Proposition 3.2) and model bias bounds (Propositions 3.3, 3.4), we obtain the following final bound of QTPT.

**Proposition 3.5.** *(**Upper Bound of** **QTPT**) Under Assumption 3.1 and the applicable model-bias proposition, assume $\Lambda_{\mathrm{test}} \ll \Lambda_{\mathrm{train}}$ and $\mathrm{d}\Lambda_{\mathrm{test}}/\mathrm{d}\Lambda_{\mathrm{train}} \leq \mathcal{C}$. Let*

$$S_n(\eta) = \mathcal{O}\left(\sqrt{L}\,T^2\left[\left(\frac{2\log(2T/\eta)}{n}\right)^{1/4} + \frac{c(\hat{\mathcal{Q}}, n)}{\sqrt{n}}\right] + \sqrt{L}\,TB\right),$$

*and let $M_n(\eta)$ denote the applicable model-bias bound,*

$$M_n(\eta) = \begin{cases} \epsilon_{\mathrm{tf}} + \mathcal{O}\left(\sigma\sqrt{T}\sqrt{\frac{d\log T + \log(1/\eta)}{\alpha n}}\right) & \text{for the linear bandit,} \\ 2T\sqrt{\epsilon_{\mathrm{tf}}} + \mathcal{O}\left(T^2\left[n^{-1/2} + \left(\frac{2\log(2T/\eta)}{n}\right)^{1/4}\right]\right) & \text{for the finite-horizon MDP.} \end{cases}$$

*Then, with probability at least $1-\delta$, $\mathsf{Subopt}_{\Lambda_{\mathrm{test}}}(\pi^*, \pi_{\tilde{Q}_n}) \leq \mathcal{C}\,[S_n(\delta/2) + M_n(\delta/2)]$. The sample-bias and model-bias events each receive failure probability $\delta/2$, so a union bound gives overall failure probability at most $\delta$ without requiring independence.*

The constant $\mathcal{C}$ in Proposition 3.5 describes the environment-level distribution shift between the pretraining environments $\Lambda_{\text{train}}$ and the test environments $\Lambda_{\text{test}}$, which can differ in state transitions, reward structures, or the size of the action space; it reflects how well the pretraining data prepares the model for test environments.

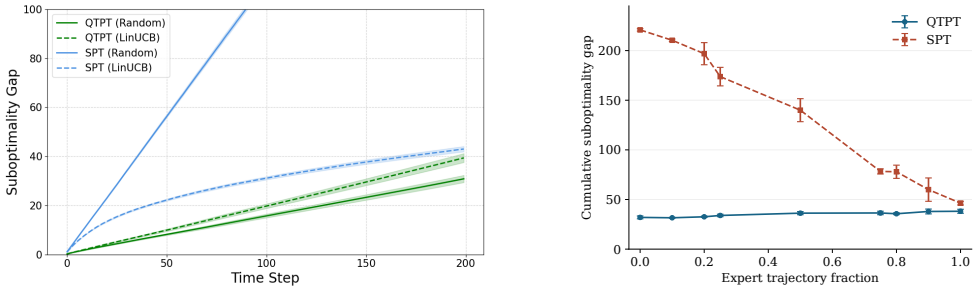
To contextualize the bandit part of Proposition 3.5, we compare it with Lin et al. [2024], who analyze SPT in a related bandit setting and decompose the total regret into statistical estimation bias, approximation bias $\epsilon_{\text{real}}$ (similar to our $\epsilon_{\text{tf}}$; see Section A.4), and intrinsic expert bias $\epsilon_{\text{approx}}$. This comparison clarifies the different behavior of supervised and Bellman-style pretraining under weak random data, rather than claiming a uniform rate dominance across all settings.

**When offline dataset is collected by random policy.** A supervised learner trained on random behavior has no improving behavior-action signal: imitation can retain linear cumulative regret. Bellman pretraining instead uses reward information: our bandit *model-bias component* decreases as $\mathcal{O}(\sqrt{T\log T/n})$ when fixed problem factors are suppressed. This component rate does not replace the full bound in Proposition 3.5, which also contains the sample-bias term of Proposition 3.2.

**When offline dataset is collected by expert policy.** LinUCB trajectories provide stronger action supervision. In that regime, the relevant comparison includes both supervision quality and the finite-data/approximation terms, not a universal rate ordering between methods. The connection to batch fitted-Q analysis [Antos et al., 2007] concerns the role of offline sample size, not a proof that all horizon or coverage factors are identical.

### 3.4 A Dataset Criterion for When QTPT Helps

We now give a sufficient condition on the offline dataset under which QTPT can improve over SPT. Inspired by Kumar et al. [2022], we first define the non-informative context set:




(a) (b)

Figure 2: **(a)** Suboptimality gap over time for QTPT and SPT under random and LinUCB pretraining data, averaged over 1000 runs. **(b)** Mixture evaluation: mean gap and one standard deviation across five training seeds, evaluated on 1,000 task instances, versus the fraction of expert trajectories. Both are finite-horizon results at $T = 200$, not a verification of asymptotic rates.

**Definition 3.6.** (**Non-informative Context**) For an environment $M$ and its optimal Q-function $Q^*_M$, the non-informative context set $G$ contains all $a \in \mathcal{A}$ and $x \in \mathcal{X}$ such that $\nu_M(x, a) \leq O(T^{-1})$, where $\nu_M(x, a) = \max_{a'} Q^*_M(x, a') - Q^*_M(x, a)$.

Here $G$ is a set of *near-indifferent context–action pairs*, not a set of histories carrying no information about the task. It necessarily includes an optimal action at each context because its action gap is zero, so it is not empty; the criterion concerns the gap and visitation of suboptimal actions outside $G$. For any $(x, a) \in G$, no alternative action $a' \in \mathcal{A}$ significantly improves the optimal $Q$ value, so following the behavior policy does not substantially damage the output policy and SPT can behave similarly to QTPT. When some action has a large enough $\nu_M$, however, the Bellman target in Algorithm 1 can move the learned policy toward this better action, while SPT continues to imitate the behavior policy. Proposition 3.7 formalizes this sufficient condition.

**Proposition 3.7.** *Let $G$ be the non-informative context set of an environment $M$ with optimal $Q$ function $Q^*_M$. Assuming there exists a positive constant $c$ such that for $(x, a) \in (\mathcal{X} \times \mathcal{A}) \setminus G$, $\nu_M(x, a) > c$. Then, under the technical coverage and gap conditions stated in the proof, we have*

$$\mathsf{Subopt}_\Lambda(\pi^*, \hat{\pi}_{QTPT}) \lesssim \mathsf{Subopt}_\Lambda(\pi^*, \hat{\pi}_{SPT}).$$

## 4 Numerical Experiments

We evaluate QTPT on theory-aligned controlled benchmarks, contextual MDPs, controlled ablations, and standard D4RL environments, organized around three questions: (i) does replacing supervised behavior prediction with Q-target pretraining improve robustness when offline data are weak or suboptimal? (ii) what roles do context conditioning and Bellman/TD targets play in this improvement? (iii) does the method remain effective beyond the controlled environments used for theoretical calibration?

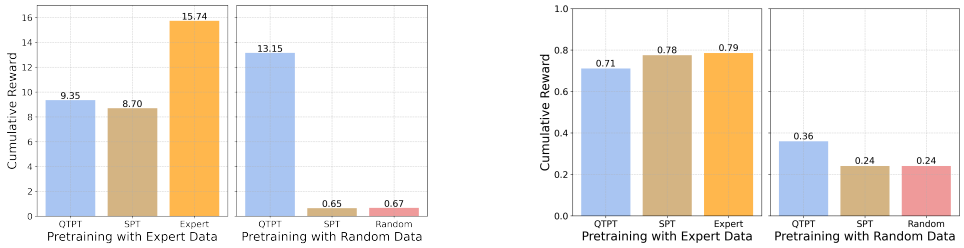
### 4.1 Stochastic Linear Bandit

We consider a stochastic linear bandit with dimension $d = 5$, arms $A = 10$, and horizon $T = 200$. At each time step, the agent selects $a_t$ and receives reward $r_t = \langle a_t, \theta^* \rangle + \epsilon_t$, where $\epsilon_t \sim N(0, 1.5^2)$ and $\theta^*$ is sampled from $[0, 1]^d$. The action set is fixed over time, with actions drawn i.i.d. from the same range. We pretrain on two offline datasets: one collected by random actions and the other by LinUCB, each containing 100,000 trajectories. Transformer backbones follow the GPT-2-style architecture used throughout the paper, with 8 layers, 4 heads, and embedding dimension 256.

Figure 2a shows that QTPT consistently outperforms SPT on both random and LinUCB datasets. Figure 2b further shows that QTPT remains stable as the proportion of expert data decreases, while SPT is highly sensitive to the quality of the offline trajectories. Here weak behavior quality and action coverage are distinct: random actions are poor imitation targets but can supply broad reward/transition evidence. Appendix D.4 repeats the objective comparison with GPT-2-, Llama-, and Qwen2-style decoders; Appendices C.2 and C.5 report non-stationary test distributions and comparisons against CQL, IQL, Q-SFT, and QDT.

Table 1: Controlled ablations on Darkroom under matched architectures and offline datasets: mean reward with 95% confidence-interval half-widths over 300 evaluation episodes (episode-level variation, not training-seed variability); higher is better.

| Variant | Random data | Expert data |
|---|---|---|
| QTPT (full) | $14.46 \pm 3.27$ | $18.29 \pm 4.06$ |
| QTPT-MC | $9.33 \pm 3.01$ (-35%) | $13.77 \pm 3.68$ (-25%) |
| QTPT-NoCtx | $0.38 \pm 0.61$ (-97%) | $0.34 \pm 0.65$ (-98%) |




(a) (b)

Figure 3: Average cumulative reward in **(a)** Darkroom and **(b)** Dark Key-to-door. QTPT is usually stronger under weak data, while extremely sparse expert-only settings remain difficult for TD learning.

### 4.2 Controlled Ablations

To understand *why QTPT helps*, we compare it with two matched variants: *QTPT-MC*, which replaces bootstrapped TD targets with Monte Carlo returns, and *QTPT-NoCtx*, which keeps the TD objective but removes context conditioning. All three models share the architecture, offline dataset, and evaluation protocol.

In Table 1, the point estimates decrease when context is removed and when TD targets are replaced by Monte Carlo returns, motivating both mechanisms; because the intervals are episode-level, they do not alone establish seed-level significance. Appendix D.5 separately examines task-relevant context; the conservative-objective results use their own validation-best protocol and are not pooled with this table.

### 4.3 Markov Decision Processes: Darkroom and Dark Key-to-door

We evaluate QTPT on Darkroom, Dark Key-to-door, and Miniworld, which are standard contextual decision-making benchmarks for in-context RL [Laskin et al., 2022, Lee et al., 2023]. Figure 3 shows Darkroom and Dark Key-to-door; Miniworld follows the same qualitative trend (Appendix B).

Across these MDPs, QTPT outperforms SPT in most settings, especially when the offline data are weak or broadly exploratory. Notably, random-data pretraining can outperform expert-data pretraining in Darkroom and Miniworld, because random trajectories cover wider state-action support and provide richer signals for Q-learning. The main exception is Dark Key-to-door with expert data, where rewards are extremely sparse and bootstrapped training receives less informative supervision.

### 4.4 D4RL Extension

We further extend the study to D4RL Kitchen and AntMaze [Fu et al., 2020], which involve higher-dimensional observations, longer horizons, and much sparser rewards than the synthetic environments. Table 2 reports the results; Kitchen and AntMaze use different metrics, so comparisons are within each task.

QTPT has the largest reported mean under this matched protocol, including 35.94% success on `antmaze-umaze-v2`. Given the substantial between-seed variation, these runs demonstrate an implemented extension, not statistically established superiority or a comparison with published D4RL SOTA scores (Appendix C.4).

Table 2: D4RL with matched Transformer baselines: Kitchen normalized score and AntMaze success rate (%), mean $\pm$ std over 4 seeds.

| Method | Kitchen-partial-v0 | Kitchen-mixed-v0 | AntMaze-umaze-v2 |
|---|---|---|---|
| QTPT (Ours) | $13.23 \pm 4.03$ | $14.62 \pm 3.36$ | $35.94 \pm 13.52$ |
| SPT | $10.84 \pm 4.18$ | $13.25 \pm 2.68$ | $26.96 \pm 11.39$ |
| Q-SFT | $11.37 \pm 1.14$ | $9.44 \pm 4.86$ | $19.05 \pm 9.21$ |
| QDT | $12.58 \pm 4.28$ | $7.28 \pm 5.23$ | $20.00 \pm 8.66$ |

### 4.5 Math Reasoning and Supplementary Tests

**Math reasoning.** To test cross-domain breadth, we adapt QTPT to mathematical reasoning as QTPO (QTPT with policy optimization; setup in Appendix C.3). On GSM8K, QTPO improves over behavior cloning across all mixture ratios by an average of **1.81%**, suggesting that the benefits of value-aware pretraining extend beyond RL.

**Adaptation and value calibration.** Five-seed support-controlled bandit experiments directly expose residual unsupported-action overestimation. QTPT reduces harmful OOD selection relative to naive hard-max QTPT, but adding a CQL penalty produces no statistically reliable incremental improvement in either backup branch (Appendix D.2). Comparisons with $RL^2$ and PEARL test complementary data-acquisition protocols rather than a universal ranking (Appendix D.3). The context experiments separate task-coherent history from unrelated extra examples (Appendix D.5). Each diagnostic states its own configuration, uncertainty unit, and scope.

## 5 Limitations

Our theory requires support/coverage conditions on the offline data, so the guarantees do not apply to arbitrary low-coverage datasets; the intended weak-data regime has suboptimal or noisy behavior with sufficient state-action coverage for Bellman-style learning. Our D4RL experiments are fixed-protocol extensions with matched Transformer baselines and data; they do not claim state-of-the-art performance against specialized D4RL-tuned pipelines. Finally, the double-target and softmax backups mitigate, but do not eliminate, harmful unsupported-action overestimation, and a standard CQL penalty adds no statistically reliable improvement in our support-controlled diagnostic.

## 6 Conclusion and Future Work

We propose Q-target Pretrained Transformers (QTPT), which use reward and Bellman-target supervision to learn a context-conditioned value function from offline trajectories. The analysis separates finite-data error from Transformer approximation, with explicit coverage, confidence, and horizon dependence. Controlled experiments show gains over supervised behavior prediction when behavior actions are weak, and ablations identify context conditioning as necessary for in-context adaptation and Bellman/TD targets as an additional source of robustness. Future directions include larger D4RL suites, newer offline RL benchmarks, and industrial-scale datasets; alternative Bellman backups such as multi-step returns; train–test distribution shift; and broader NLP and reasoning tasks. Complete proofs are given in Appendix A.

## Broader Impact

This work shows how in-context RL agents can learn effective policies from existing, possibly suboptimal, offline data, reducing reliance on expert demonstrations and on costly or risky online exploration. It also carries risks: a policy pretrained on logged data inherits the coverage gaps and biases of the behavior policy, our guarantees hold only under the stated coverage assumptions, and unsupported-action overestimation is reduced but not eliminated. In high-stakes settings such as healthcare or finance, such policies should be validated on held-out environments, monitored for distribution shift, and kept under human oversight.

## Acknowledgments and Disclosure of Funding

Tao Yao's research was supported by the National Science Foundation (NSF) under grant DMS-1820702 and the National Natural Science Foundation of China (NSFC) under grants W2441021, 72371172, 72342023, and 71929101.

## Competing Interests

The authors declare no competing interests.

## References

Yasin Abbasi-yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper_files/paper/2011/file/e1d5be1c7f2f456670de3d53c7b54f4a-Paper.pdf.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

András Antos, Csaba Szepesvári, and Rémi Munos. Fitted q-iteration in continuous action-space mdps. *Advances in neural information processing systems*, 20, 2007.

Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *arXiv preprint arXiv:2306.04637*, 2023.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. In *International Conference on Machine Learning*, pages 1042–1051. PMLR, 2019.

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.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Juncheng Dong, Weibin Mo, Zhengling Qi, Cong Shi, Ethan X. Fang, and Vahid Tarokh. Pasta: Pessimistic assortment optimization, 2023. URL https://arxiv.org/abs/2302.03821.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. $RL^2$: Fast reinforcement learning via slow reinforcement learning, 2016. URL https://arxiv.org/abs/1611.02779.

Yaqi Duan, Chi Jin, and Zhiyuan Li. Risk bounds and rademacher complexity in batch reinforcement learning. In *International Conference on Machine Learning*, pages 2892–2902. PMLR, 2021.

Michael O. Duff. *Optimal Learning: Computational Procedures for Bayes-Adaptive Markov Decision Processes*. PhD thesis, University of Massachusetts Amherst, 2002.

Eyal Even-Dar and Yishay Mansour. Convergence of optimistic and incremental q-learning. *Advances in neural information processing systems*, 14, 2001.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020. URL https://arxiv.org/abs/2004.07219.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Joey Hong, Anca Dragan, and Sergey Levine. Q-sft: Q-learning for language models via supervised fine-tuning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/a9531a3e64e6167c7e1e671157082682-Paper-Conference.pdf.

Shengchao Hu, Ziqing Fan, Chaoqin Huang, Li Shen, Ya Zhang, Yanfeng Wang, and Dacheng Tao. Q-value regularized transformer for offline reinforcement learning, 2024. URL https://arxiv.org/abs/2405.17098.

Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I. Jordan. Is q-learning provably efficient?, 2018. URL https://arxiv.org/abs/1807.03765.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl?, 2022. URL https://arxiv.org/abs/2012.15085.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X. URL https://cs.brown.edu/courses/csci2951-k/papers/kaelbling98.pdf.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the nineteenth international conference on machine learning*, pages 267–274, 2002.

Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel : Model-based offline reinforcement learning, 2021. URL https://arxiv.org/abs/2005.05951.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning, 2021. URL https://arxiv.org/abs/2110.06169.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning, 2020. URL https://arxiv.org/abs/2006.04779.

Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. When should we prefer offline reinforcement learning over behavioral cloning?, 2022. URL https://arxiv.org/abs/2204.05618.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

Jonathan N Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. Supervised pretraining can learn in-context reinforcement learning. *arXiv preprint arXiv:2306.14892*, 2023.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020. URL https://arxiv.org/abs/2005.01643.

Licong Lin, Yu Bai, and Song Mei. Transformers as decision makers: Provable in-context reinforcement learning via supervised pretraining. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/84b5e56a80cbea27de332d456f0d5a4c-Abstract-Conference.html.

Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization, 2020. URL https://arxiv.org/abs/2006.03647.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. MetaICL: Learning to learn in context. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.201. URL https://aclanthology.org/2022.naacl-main.201.

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

Amir Moeini, Jiuqi Wang, Jacob Beck, Ethan Blaser, Shimon Whiteson, Rohan Chandra, and Shangtong Zhang. A survey of in-context reinforcement learning, 2025. URL https://arxiv.org/abs/2502.07978.

Thanh Nguyen-Tang, Ming Yin, Sunil Gupta, Svetha Venkatesh, and Raman Arora. On instance-dependent bounds for offline reinforcement learning with linear function approximation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023. URL https://ojs.aaai.org/index.php/AAAI/article/view/26116.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline rl?, 2024. URL https://arxiv.org/abs/2406.09329.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve, 2024. URL https://arxiv.org/abs/2407.18219.

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 5331–5340, 2019. URL https://proceedings.mlr.press/v97/rakelly19a.html.

Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *Advances in Neural Information Processing Systems*, 34:11702–11716, 2021.

Amrith Setlur, Saurabh Garg, Xinyang Geng, Naman Garg, Virginia Smith, and Aviral Kumar. Rl on incorrect synthetic data scales the efficiency of llm math reasoning by eight-fold, 2024. URL https://arxiv.org/abs/2406.14532.

Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016. URL https://ojs.aaai.org/index.php/AAAI/article/view/10295.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent, 2023. URL https://arxiv.org/abs/2212.07677.

Jiuqi Wang, Ethan Blaser, Hadi Daneshmand, and Shangtong Zhang. Transformers can learn temporal difference methods for in-context reinforcement learning. *International Conference on Learning Representations*, 2024a.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning, 2019. URL https://arxiv.org/abs/1911.11361.

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

Kunhe Yang, Lin F. Yang, and Simon S. Du. $q$-learning with logarithmic regret, 2021. URL https://arxiv.org/abs/2006.09118.

Ming Yin, Yu Bai, and Yu-Xiang Wang. Near-optimal offline reinforcement learning via double variance reduction. In *Advances in Neural Information Processing Systems*, 2021. URL https://papers.nips.cc/paper/2021/file/3f24bb08a5741e4197af64e1f93a5029-Paper.pdf.

Andrea Zanette and Emma Brunskill. Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. In *International Conference on Machine Learning*, pages 7304–7312. PMLR, 2019.

Ruiqi Zhang, Spencer Frei, and Peter L Bartlett. Trained transformers learn linear models in-context. *arXiv preprint arXiv:2306.09927*, 2023.

Zifeng Zhuang, Kun Lei, Jinxin Liu, Donglin Wang, and Yilang Guo. Behavior proximal policy optimization. *ArXiv*, abs/2302.11312, 2023. URL https://api.semanticscholar.org/CorpusID:257079136.

# A Proofs

## A.1 Proofs Related to The Sample Bias

**Definition A.1** (Rademacher complexity)**.** For a generic real-valued function space $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{Y}}$ and $n$ fixed data points $Y = \{y_1, \cdots, y_n\} \in \mathcal{Y}^n$, the empirical Rademacher complexity is defined as follows:

$$\hat{\mathcal{R}}_Y(\mathcal{F}) = \mathbb{E}_\omega \Big[ \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \omega_i f(y_i) | Y \Big],$$

where $\omega_i$ are independent and identically distributed Rademacher random variables, ranging from $-1$ to 1 and, thus, representing randomly assigned labels. Let $\rho$ be the distribution of $Y$, we further define the Rademacher complexity $\mathcal{R}_n^\rho(\mathcal{F}) := \mathbb{E}_\rho[\hat{\mathcal{R}}_Y(\mathcal{F})]$, which measures the complexity of $\mathcal{F}$ by the degree to which the functions in class $\mathcal{F}$ are correlated with the random noise $w_i$.

This definition follows a similar structure in Duan et al. [2021]. The Rademacher complexity provides a measure of the complexity of our function class, which will be crucial for bounding the sample bias.

### A.1.1 Proof of Proposition 3.2

*Proof.* On the one hand, for a fixed $Q \in \hat{\mathcal{Q}}$ , considering the union bound, by Lemma G.1 and Lemma G.6 in Duan et al. [2021],

$$\Pr \Big[ |\mathcal{L}_n(Q) - \mathcal{L}(Q)| \geq \epsilon_n \Big] \leq \delta,$$

where $\epsilon_n = B_Q^2 \sqrt{\frac{2 \ln(2T/\delta)}{n}} + 2B_Q \mathcal{R}_n^{\pi_\beta}(\hat{\mathcal{Q}}) + B_Q^2$. Thus, with probability at least $1 - \delta$, $|\mathcal{L}_n(Q) - \mathcal{L}(Q)| \leq \epsilon_n$, and

$$\mathcal{L}(\hat{Q}^*) \leq \mathcal{L}_n(\hat{Q}^*) + \epsilon_n \leq \mathcal{L}_n(Q^*) + \epsilon_n \leq \mathcal{L}(Q^*) + 2\epsilon_n,$$

since $\mathcal{L}(Q^*) = 0$, $\mathcal{L}(\hat{Q}^*) \leq 2\epsilon_n$.
On the other hand, by Assumption 3.1, it implies that $d_{M,t}^{\pi_\beta}(x_t, a_t) \geq L^{-1}$, when $d_{M,t}^{\pi_\beta}(x_t, a_t) > 0$. For any $(t, x_t, a_t)$,

$$\frac{d_{M,t}^{\pi^*}(x_t, a_t)}{d_{M,t}^{\pi_\beta}(x_t, a_t)} \leq \frac{1}{d_{M,t}^{\pi_\beta}(x_t, a_t)} \leq L.$$

Thus, using Performance Difference Lemma by Lemma 6.1 in Kakade and Langford [2002], we have

$$\begin{aligned}
\mathsf{Subopt}_\Lambda(\pi^*, \pi_{\hat{Q}^*}) &= \mathbb{E}_{M \sim \Lambda} \textstyle\sum_{t=1}^{T} \mathbb{E}_{d^{\pi^*}} \Big[ |(\Gamma \hat{Q}^*)(x_t, a) - \hat{Q}^*(x_t, a)| \Big] \\
&\leq \sqrt{T \textstyle\sum_{t=1}^{T} \mathbb{E}_{d^{\pi^*}} \Big[ |\hat{Q}^*(x_t, a) - (\Gamma \hat{Q}^*)(x_t, a)|^2 \Big]} \\
&\leq T\sqrt{L} \sqrt{\mathcal{L}_n(\hat{Q}^*)} \\
&\leq \sqrt{L} T \sqrt{2\epsilon_n} \\
&\leq \sqrt{L} \Big[ 2T^2 (\tfrac{2\log(2T/\delta)}{n})^{1/4} + \sqrt{2} T^{3/2} \sqrt{\mathcal{R}_n^{\pi_\beta}(\hat{\mathcal{Q}})} + TB \Big],
\end{aligned}$$

using Proposition 6.1 in literature Duan et al. [2021], we get the final results as

$$\mathsf{Subopt}_\Lambda(\pi^*, \pi_{\hat{Q}^*}) \leq \mathcal{O}\Big( \sqrt{L} T^2 \Big[ (\frac{2\log(2T/\delta)}{n})^{1/4} + \sqrt{\max\{\sqrt{\frac{\log|\mathcal{Q}|}{n}}, \frac{\log|\mathcal{Q}|}{n}\}} \Big] + \sqrt{L} TB \Big).$$

□

## A.2 Proofs Related to the Model Bias

### A.2.1 Proof of Proposition 3.3

*Proof.* At iteration $k$, with current parameters $w^{(k)}$, compute $y_t^{i,(k)} = r_t + \max_{a' \in \mathcal{A}_{t+1}} Q_{w^{(k)}}(D_t^i, s_{t+1}^i, a')$, note that $y_t^{i,(k)}$ is now a fixed constant w.r.t. $w$. Thus, QTPT at

round $t-1$ solves the ordinary least-square problem $\hat{w}_{t-1} = \arg\min_w \sum_{i=1}^{n}\sum_{u=1}^{t-1}(\langle a_u^i, w\rangle - r_u^i)^2$. A greedy policy then selects $a_t = \arg\max_{a\in\mathcal{A}_t}\langle a, \hat{w}_{t-1}\rangle$. Define the estimation error term $EE_t = \sup_{a\in\mathcal{A}_t}|\langle a, w^* - \hat{w}\rangle|$. We get the solution $\hat{w}_{t-1} = G_{t-1}^{-1}\sum_{i=1}^{n}\sum_{u=1}^{t-1} a_u^i r_u^i$.
On the one hand, by Theorem 2 in Abbasi-Yadkori et al. (2011)Abbasi-yadkori et al. [2011], for zero-mean, $\sigma$-sub-aussian noise $\epsilon_u$, with probability at least $1-\delta$, we have

$$|\hat{w}_{t-1} - w^*\|_{G_{t-1}} \le \sigma\sqrt{2\log\frac{\det(G_{t-1})^{1/2}}{\det(\alpha nI)^{1/2}\delta}} = \beta_{t-1}$$

On the other hand, for any $a$,

$$|Q_{\hat{w}} - Q^*| = |\langle a, \hat{w} - w^*\rangle| \le \|a\|_{G_{t-1}^{-1}}\|\hat{w} - w^*\|_{G_{t-1}} \le \frac{\beta_{t-1}}{\sqrt{\lambda_{min}(G_{t-1})}} \le \frac{\beta_{t-1}}{\sqrt{\alpha n(t-1)}},$$

since $\det(G_{t-1}) = \mathcal{O}((\alpha n)^d(t-1)^d)$, we have $\beta_{t-1} = \mathcal{O}(\sigma\sqrt{d\ln t + \log(1/\delta)})$.
Thus,

$$EE_t = \sup|Q_{\hat{w}} - Q^*| \le \mathcal{O}(\sigma\sqrt{\frac{d\ln t + \log(1/\delta)}{\alpha nt}}).$$

The cumulative suboptimality over $T$ steps as

$$\mathbb{E}_{M\sim\Lambda}\Big[\sum_{t=1}^{T}(\langle a^*, w^*\rangle - \langle a_t, \hat{w}\rangle)\Big],$$

then we have

$$\langle a^*, w^*\rangle - \langle a_t, \hat{w}\rangle = \Big[\langle a^*, w^*\rangle - \langle a^*, \hat{w}\rangle\Big] + \Big[\langle a^*, \hat{w}\rangle - \langle a_t, \hat{w}\rangle\Big] \le MB_t,$$

Finally, we get that

$$\mathsf{Subopt}(\pi_{\hat{Q}^*}, \pi_{\tilde{Q}}) \le 2\mathbb{E}[\sum_{t=1}^{T} EE_t] + \epsilon_{\text{tf}} \le \epsilon_{\text{tf}} + \mathcal{O}(\sigma\sqrt{T}\sqrt{\frac{d\log(nT) + \log(1/\delta)}{\alpha n}}),$$

where $\epsilon_{\text{tf}}$ is the Transformer approximation error (see Section A.4). □

### A.2.2 Proof of Proposition 3.4

*Proof.* For simplicity, we abbreviate $\hat{Q}_{M,t}^{\pi}$ as $\hat{Q}_t$. Let $\hat{e}_t^{\text{TD}}(x,a) = \hat{Q}_t(x,a) - r - \mathbb{E}_{x_{t+1}}\Big[\max_{a'}\hat{Q}_{t+1}(x_{t+1}, a')\Big]$. Similarly, using the definitions of $\tilde{e}_t^{\text{TD}}$ and $e_t^{\text{TD},*}$, we can express $\hat{Q}$ to $\tilde{Q}$ and $Q^*$ in terms of $\hat{e}_t^{\text{TD}}$. For function $\hat{Q} = (\hat{Q}, \cdots, \hat{Q}_T)$, define

$$\epsilon(\hat{Q}) = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\hat{Q}_t - r - \mathbb{E}_{x_{t+1}}\max_{a'}\hat{Q}_{t+1}(x_{t+1}, a')\right]^2 = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\hat{e}_t^{\text{TD}}(x,a)^2.$$

Similarly, using the definition of $\epsilon(\tilde{Q})$, we can express $\hat{Q}$ to $\tilde{Q}$ in terms of $\epsilon(\hat{Q})$. Since $\pi_{\tilde{Q}}(x) = \arg\max_{a'}\tilde{Q}(x,a')$, we can show that

$$V_M^{\pi_{\hat{Q}^*}}(x) - V_M^{\pi_{\tilde{Q}}}(x) \le V^{\pi_{\hat{Q}^*}}(x) - \tilde{Q}_1(x, \pi_{\hat{Q}^*}(x)) + \tilde{Q}_1(x, \pi_{\tilde{Q}}(x)) - V_M^{\pi_{\tilde{Q}}}(x).$$

For any policy $\pi$, we have

$$\begin{aligned}
&\tilde{Q}_1(x_1, \pi(x_1)) - V_{M,1}^{\pi}(x_1)\\
=&\mathbb{E}\left[\sum_{t=1}^{T}\Big(\tilde{Q}_t(x_t,a_t) - \mathbb{E}_{\pi}[\tilde{Q}_{t+1}(x_{t+1},a_{t+1}) + r_t|x_t,a_t]\Big)\Big|x_1,\pi\right]\\
=&\mathbb{E}\left[\sum_{t=1}^{T}\Big(\tilde{Q}_t(x_t,a_t) - r_t - \tilde{Q}_{t+1}(x_{t+1},a_{t+1})\Big)\Big|x_1,\pi\right].
\end{aligned}$$

We can show that

$$\Big|\mathbb{E}\Big[\sum_{t=1}^{T}\tilde{e}_t^{\mathrm{TD}}(x_t,a_t)|x_1,\pi\Big]\Big| \le \sqrt{T\sum_{t=1}^{T}\mathbb{E}\Big[\tilde{e}_t^{\mathrm{TD}}(x_t,a_t)^2|x_1,\pi\Big]} \le T\sqrt{\epsilon(\tilde{Q})},$$

which implies that

$$\tilde{Q}_1(x_1,\pi_{\tilde{Q}}(x_1)) - V_{M,1}^{\pi_{\tilde{Q}}}(x_1) = \mathbb{E}\Big[\sum_{t=1}^{T}\tilde{e}_t^{\mathrm{TD}}(x_t,a_t)|x_1,\pi_{\tilde{Q}}\Big],$$

$$\tilde{Q}_1(x_1,\pi_{\hat{Q}^*}(x_1)) - V_{M,1}^{\pi_{\hat{Q}^*}}(x_1) \ge \mathbb{E}\Big[\sum_{t=1}^{T}\tilde{e}_t^{\mathrm{TD}}(x_t,a_t)|x_1,\pi_{\hat{Q}^*}\Big].$$

Hence,

$$V_{M,1}^{\pi_{\hat{Q}^*}}(x) - V_{M,1}^{\pi_{\tilde{Q}}}(x) \le \mathbb{E}\Big[\sum_{t=1}^{T}\tilde{e}_t^{\mathrm{TD}}(x_t,a_t)|x_1,\pi_{\tilde{Q}}\Big] - \mathbb{E}\Big[\sum_{t=1}^{T}\tilde{e}_t^{\mathrm{TD}}(x_t,a_t)|x_1,\pi_{\hat{Q}^*}\Big] \le 2T\sqrt{\epsilon(\tilde{Q})}.$$

since $\hat{e}_t^{\mathrm{TD}}(x,a)^2 - \tilde{e}_t^{\mathrm{TD}}(x,a)^2 \in [-2T^2, 2T^2]$, using Massart Finite Class Lemma, with probability at least $1-\delta$, there exist a constant $c$ such that for any $\pi$,

$$\begin{aligned}
& \mathbb{E}\tilde{e}_t^{\mathrm{TD}}(x,a)^2 - \mathbb{E}\hat{e}_t^{\mathrm{TD}}(x,a)^2 \\
\le\ & \tfrac{1}{n}\sum_{(x_t,a_t,r_t,x_{t+1})\sim D_t}(\tilde{e}_t^{\mathrm{TD}}(s,a)^2 - \hat{e}_t^{\mathrm{TD}}(x,a)^2) \\
& \quad +2R_n((\tilde{e}_t^{\mathrm{TD}})^2 - (\hat{e}_t^{\mathrm{TD}})^2|\hat{Q}_t) + 2T^2\sqrt{\tfrac{2\log(2/\delta)}{n}} \\
\le\ & 2T^2\sqrt{\tfrac{2\log(2/\delta)}{n}} + cTR_n(\hat{Q}_t) \\
\le\ & 2T^2\sqrt{\tfrac{2\log(2/\delta)}{n}} + 2cT^2\tfrac{\sqrt{2\log|\mathcal{Q}_t|}}{n}.
\end{aligned}$$

Since

$$\epsilon(\tilde{Q}) - \epsilon(\hat{Q}) = \frac{1}{T}\sum_{t=1}^{T}\Big(\mathbb{E}\tilde{e}_t^{\mathrm{TD}}(x,a)^2 - \mathbb{E}\hat{e}_t^{\mathrm{TD}}(x,a)^2\Big),$$

since $\epsilon(\hat{Q})$ indicates the capacity of Transformers approximation (see Section A.4 for details), we can get that

$$\begin{aligned}
\epsilon(\tilde{Q}) \le\ & \epsilon(\hat{Q}) + 2T^2\sqrt{\tfrac{2\log(2T/\delta)}{n}} + 2cT\sum_{t=1}^{T}\tfrac{\sqrt{2\log|\hat{\mathcal{Q}}_t|}}{n} \\
\le\ & \epsilon_{\mathrm{tf}} + 2T^2\sqrt{\tfrac{2\log(2T/\delta)}{n}} + 2cT\sum_{t=1}^{T}\tfrac{\sqrt{2\log|\hat{\mathcal{Q}}_t|}}{n},
\end{aligned}$$

Thus,

$$\begin{aligned}
& V_{M,1}^{\pi_{\hat{Q}^*}}(x_1) - V_{M,1}^{\pi_{\tilde{Q}}}(x_1) \\
\le\ & 2T\sqrt{\epsilon_{\mathrm{tf}}} + T^2\Big(\sqrt{2}(\tfrac{2\log(2T/\delta)}{n})^{1/4} + \sqrt{2c\tfrac{\sqrt{2\sup\log|\hat{\mathcal{Q}}_t|}}{n}}\Big) \\
=\ & 2T\sqrt{\epsilon_{\mathrm{tf}}} + \mathcal{O}\Big(T^2\Big(n^{-1/2} + (2\log(2T/\delta)/n)^{1/4}\Big)\Big).
\end{aligned}$$

□

### A.2.3 Proof of Proposition 3.5

*Proof.* Since

$$\begin{aligned}
& \int \Lambda_{\mathrm{test}}(M)\,\mathbb{E}_{x_1\sim\eta_M}\Big[V_{M,1}^{*}(x_1) - V_{M,1}^{\pi_{\tilde{Q}}}(x_1)\Big]\,\mathrm{d}M \\
& \qquad \le \mathcal{C}\int \Lambda_{\mathrm{train}}(M)\,\mathbb{E}_{x_1\sim\eta_M}\Big[V_{M,1}^{*}(x_1) - V_{M,1}^{\pi_{\tilde{Q}}}(x_1)\Big]\,\mathrm{d}M,
\end{aligned}$$

thus

$$\mathsf{Subopt}_{\Lambda_{\mathrm{test}}}(\pi^*,\pi_{\tilde{Q}}) \le \mathcal{C}\cdot\mathsf{Subopt}_{\Lambda_{\mathrm{train}}}(\pi^*,\pi_{\tilde{Q}}).$$

For the bandit case apply Proposition 3.2 and Proposition 3.3 with confidence parameter $\delta/2$ each; for the MDP case use Proposition 3.2 and Proposition 3.4 instead. In either case the union bound gives probability at least $1-\delta$ for their simultaneous validity. Add the two bounds using Eq. (5), retaining the approximation terms, and then apply the environment-density ratio $\mathcal{C}$. □

### A.3 Proofs Related to the Comparison of QTPT and SPT

#### A.3.1 Proof of Proposition 3.7

At first, we need to prove a result on non-informative context states, which is inspired by Lemma B.10 in [Kumar et al., 2022]. For simplify, when $G$ be the non-informative context set, define $G_{\mathcal{A}}, G_{\mathcal{X}}$ be the separate sets to represent all unique context states and actions present in $G$, i.e. $G_{\mathcal{A}} = \{a|\exists x, (x,a) \in G\}, G_{\mathcal{X}} = \{x|\exists a, (x,a) \in G\}$.

For an environment $M$ and $\forall a \in \mathcal{A} \setminus G_{\mathcal{A}}$, $\nu_M(x,a) \simeq \Delta_M(x)$. For $(x,a) \in G$, there exists a constant $\epsilon > 0$ that satisfies $\nu_M(x,a) \leq \frac{\epsilon}{T}$.

**Lemma A.2.** *Consider a fixed environment $M$, policy $\hat{\pi}_{QTPT}$ is obtained from the* QTPT *algorithm. There exists a non-informative context set $G$ satisfying $|\mathcal{X} \setminus G_{\mathcal{X}}| \geq n_0$, and let $\forall a \in \mathcal{A}$, $\frac{d_M^{\pi^*}(a|x)}{d_M^{\pi_\beta}(a|x)} \leq \alpha_0 < 2$, where $d_M^{\pi_\beta}(a|x)$ is the conditional action distribution at context state $x$, $\alpha_0$ is a constant. $\tilde{Q}$ is the learned Q-value, since $\tilde{Q}$ can only differ from $Q^*$, there exists $\epsilon_0 > 0$, s.t. $|\tilde{Q}(x,a) - Q^*(x,a)| \leq \epsilon_0$. Then, the probability that the policy $\hat{\pi}_{QTPT}$ doesn't choose the action in $G_{\mathcal{A}}$ at context state $x$ is upper bounded as*

$$\begin{aligned}&\mathbb{P}[\hat{\pi}_{QTPT}(x) \notin G_{\mathcal{A}}] \\ \leq &\exp\Big( - n_0 \frac{\alpha_0^2}{2(\alpha_0 - 1)} \times [\frac{|G_{\mathcal{A}}| n_\Delta(x)}{n_0} - \frac{1}{\alpha_0}]^2\Big)\end{aligned}$$

*where $n(x,a)$ be the expected number of visit $(x,a)$ in dataset $D$, $n_\Delta$ corresponds to the maximum value of $n(x,a)$ s.t.*

$$\frac{\epsilon}{T} \geq \Delta_M(x) - 2\epsilon_0.$$

*Proof.*

$$\begin{aligned}\mathbb{P}[\hat{\pi}_{\text{QTPT}}(x) \notin G_{\mathcal{A}}] = &\quad \mathbb{P}_M[\exists a \notin G_{\mathcal{A}}, \forall a_g \in G_{\mathcal{A}}, \hat{Q}(x,a) \geq \hat{Q}(x,a_g)] \\ \leq &\quad \mathbb{P}_M[\exists a \notin G_{\mathcal{A}}, \forall a_g \in G_{\mathcal{A}}, \hat{Q}(x,a) - Q^*(x,a) \geq \hat{Q}(x,a_g) - Q^*(x,a_g) \\ &\quad + \Delta_M(x) - \epsilon/T] \\ \leq &\quad \mathbb{P}_M[\cap_{a_g \in G_{\mathcal{A}}} \{\epsilon/T \geq \Delta_M(x) - 2\epsilon_0\}] \\ \leq &\quad \mathbb{P}_M[\cap_{a \in G_{\mathcal{A}}} \{n(x,a) \leq n_\Delta\}]\end{aligned}$$

where $n_\Delta$ corresponds to the maximum value of $n(x,a)$ s.t.

$$\frac{\epsilon}{T} \geq \Delta_M(x) - 2\epsilon_0$$

consider the action sampled from $d_M^{\pi_\beta}(a|x)$ belong to the set $G_{\mathcal{A}}$ or not,

$$\begin{aligned}\mathbb{P}_M[\cap_{a_g \in G_{\mathcal{A}}} \{n(x,a) \leq n_\Delta\}] &\leq \mathbb{P}_M[\textstyle\sum_{a_g \in G_{\mathcal{A}}} n(x,a) \leq |G_{\mathcal{A}}| n_\Delta(x)] \\ &\leq \mathbb{P}_M[\frac{\sum_{a_g \in G_{\mathcal{A}}} n(x,a)}{n_0} \leq \frac{|G_{\mathcal{A}}| n_\Delta(x)}{n_0}] \\ &\leq \exp\Big( - n_0 KL\Big(Bern(\tfrac{|G_{\mathcal{A}}| n_\Delta(x)}{n_0}) || Bern(1/\alpha_0)\Big)\Big)\end{aligned}$$

since $KL(p+\epsilon||p) \geq \frac{\epsilon^2}{2p(1-p)}$, if $p \geq 1/2$, $p = 1/\alpha_0$, we have

$$\mathbb{P}_M[\cap_{a \in G_{\mathcal{A}}} \{n(x,a) \leq n_\Delta(x)\}] \leq \exp\Big( - n_0 \frac{\alpha_0^2}{2(\alpha_0 - 1)} \times [\frac{|G_{\mathcal{A}}| n_\Delta(x)}{n_0} - \frac{1}{\alpha_0}]^2\Big)$$

□

Combining the result of Lemma A.2, for an appropriate value of $c$ and under the stated coverage and gap conditions, we obtain Proposition 3.7, which compares QTPT and SPT when a few informative context states appear in the given trajectories.

*Proof.* Suppose there exists $p_M \in [0,1]$ s.t. for any policy $\pi$, the average occupancy of states which is not in non-informative set $G$ is bounded as

$$\sum_{t=1}^{T} \sum_{x \notin G_{\mathcal{X}}} d_{M,t}^{\pi}(x) \leq p_M.$$

Any given environment $M \sim \Lambda$ such that $\sup_{x\in\mathcal{X}}\{d_M^{\pi^*}(x)/d_M^{\pi_\beta}(x)\} = m$ be a generalization parameter which is independent of $T$, and $L_0 \leq 1 + 1/n$. A non-informative context set $G$ is given such that $|\mathcal{X} \setminus G_\mathcal{X}| \geq n_0$. For learning algorithm that return a policy $\hat{\pi}^*$, the suboptimality is given:

$$\begin{aligned}
& \mathsf{Subopt}(\pi^*, \hat{\pi}^*) \\
= \;& \textstyle\sum_x d_M^{\pi^*}(x)(V_{M,1}^{\pi^*}(x) - V_{M,1}^{\hat{\pi}^*}(x)) \\
\leq \;& m\Big[ \underbrace{\sum_{x\notin G_\mathcal{X}} d_M^{\hat{\pi}^*}(x)(Q^*(x;\pi^*) - Q^*(x;\hat{\pi}^*))}_{term\ 1} + \underbrace{\sum_{x\in G_\mathcal{X}} d_M^{\hat{\pi}^*}(x)(Q^*(x;\pi^*) - Q^*(x;\hat{\pi}^*))}_{term\ 2} \Big],
\end{aligned}$$

On the one hand, consider the part of $x \notin G_\mathcal{X}$ (named term 1), we can construct a new MDP where $r(x,a) = r(x,\pi^*(x))$ for all actions and $\forall x \in G_\mathcal{X}$. Denote the suboptimality of this MDP be $\mathsf{Subopt}_{x\notin G_\mathcal{X}}(\pi^*, \hat{\pi}^*)$, then we have

$$\mathsf{Subopt}(\pi^*, \hat{\pi}^*) = \mathsf{Subopt}_{x\notin G_\mathcal{X}}(\pi^*, \hat{\pi}^*)$$

This can be bounded as in Proposition 3.4 , except with the dependence on all context states $\mathcal{X}$ replaced by $\mathcal{X} \setminus G_\mathcal{X}$ . We get that

$$term\ 1 \lesssim p_M T^2 (\frac{1}{\sqrt{n}} + \frac{\sqrt[4]{\log T}}{\sqrt[4]{n}})$$

On the other hand, consider the part of $x \in G_\mathcal{X}$ (named term 2),

$$term\, 2 = \sum_{x\in G_\mathcal{X}} \sum_{a\in G_\mathcal{A}} W(x,a,\pi^*) + \sum_{x\in G_\mathcal{X}} \sum_{a\notin G_\mathcal{A}} W(x,a,\pi^*)$$

where $W(x,a,\pi^*) = d_M^{\hat{\pi}^*}(x,a)(Q^*(x;\pi^*) - Q^*(x,a))$. Since

$$\begin{aligned}
\sum_{a\in G_\mathcal{A}} W(x,a,\pi^*) &\leq d_M^{\hat{\pi}^*}(x)\mathbb{P}[\hat{\pi}^*(x) \in G_\mathcal{A}] \cdot \frac{\epsilon}{T}, \\
\sum_{a\notin G_\mathcal{A}} W(x,a,\pi^*) &\leq d_M^{\hat{\pi}^*}(x)\mathbb{P}[\hat{\pi}^*(x) \notin G_\mathcal{A}] \cdot c,
\end{aligned}$$

using Lemma A.2, and note that in this case, we are interested in the setting where $L_0 \simeq 1 + \mathcal{O}(1/n)$, we can get that

$$\sum_{x\in G_\mathcal{X}} d_M^{\hat{\pi}^*}(x)\mathbb{P}[\hat{\pi}^*(x) \notin G_\mathcal{A}] \cdot c \lesssim (1-p_M) f_{\text{QTPT}}(n,c),$$

where $f = c\cdot\exp(-n_0 \times \frac{(n+1)^2}{2n} \times [\frac{|G_\mathcal{A}|n_\Delta(x)}{n_0} - \frac{n+1}{n}]^2)$ is an exponential function of $-n$. Meanwhile, the corresponding term for SPT is

$$\begin{aligned}
& \sum_{s\in G_\mathcal{X}} d_M^{\hat{\pi}^*}(s)\mathbb{P}[\hat{\pi}^*(s) \notin G_\mathcal{A}] \cdot c \\
\lesssim & (1-p_M)c \cdot \frac{1}{n} \\
= & (1-p_M) g_{\text{SPT}}(n,c).
\end{aligned}$$

By controlling $c$ and $p_M$ for some appropriate values, we have the comparison of QTPT and SPT as follows:

$$\frac{\mathsf{Subopt}_\Lambda(\pi^*, \hat{\pi}_{\text{QTPT}})}{\mathsf{Subopt}_\Lambda(\pi^*, \hat{\pi}_{\text{SPT}})} \simeq \frac{m\Big[p_M T^2 (\frac{1}{\sqrt{n}} + \frac{\sqrt[4]{\log T}}{\sqrt[4]{n}}) + (1-p_M)(f + \epsilon/T)\Big]}{p_M \frac{T}{n} + (1-p_M) g},$$

since $T$ is finite, combining $f \to \mathcal{O}(e^{-n}), g \to \mathcal{O}(1/n)$, setting $p_M = \frac{1}{T\sqrt{T}}$, we get that

$$\begin{aligned}
\frac{\mathsf{Subopt}_\Lambda(\hat{\pi}_{\text{QTPT}})}{\mathsf{Subopt}_\Lambda(\hat{\pi}_{\text{SPT}})} &\simeq \frac{m\Big[p_M T^2 (\mathcal{O}(n^{1/2}) + \mathcal{O}(n^{3/4})) + (1-p_M) fn\Big]}{p_M T + (1-p_M) gn} \\
&\simeq \frac{m\Big[\sqrt{T}(\mathcal{O}(n^{1/2}) + \mathcal{O}(n^{3/4})) + (1-p_M) fn\Big]}{1/\sqrt{T} + (1-p_M) gn} \\
&\simeq \frac{(1-p_M) fn}{(1-p_M) gn} \\
&\simeq \mathcal{O}(e^{-n}/n)
\end{aligned}$$

so we get that

$$\mathsf{Subopt}_\Lambda(\hat{\pi}_{\mathrm{QTPT}}) \lesssim \mathsf{Subopt}_\Lambda(\hat{\pi}_{\mathrm{SPT}}).$$

□

### A.4 Transformer Realization of the QTPT Update

**Proposition A.3.** ***(Transformer Realization of One QTPT Bellman Update)*** *For bounded Q-value predictions and a tokenized transition history, there exists a Transformer block of appropriate scale that constructs the TD target and implements the relaxed QTPT update*

$$Q^+(x_t, a_t) = (1-\alpha)Q(x_t, a_t) + \alpha\left(r_t + \max_{a'} Q(x_{t+1}, a')\right), \quad \forall t \in [T].$$

*This proposition establishes architectural realizability of the update operation. It does not by itself imply that a fixed Transformer class has zero approximation error or that repeated optimization converges to the ideal Bellman minimizer; those effects are represented separately by $\epsilon_{\mathrm{tf}}$ in the model-bias bound.*

*Proof.* **Embedding and Extraction Mappings**

For each $t \in [T]$, we construct two tokens

$$h_{2(t-1)} = \begin{bmatrix} s_t \\ 0_{|\mathcal{A}|+1} \\ pos_{2(t-1)} \end{bmatrix} = \begin{bmatrix} h^a_{2(t-1)} \\ h^b_{2(t-1)} \\ h^c_{2(t-1)} \end{bmatrix}, \quad h_{2t-1} = \begin{bmatrix} 0_{|\mathcal{S}|} \\ a_t \\ r_t \\ pos_{2t-1} \end{bmatrix} = \begin{bmatrix} h^a_{2t-1} \\ h^b_{2t-1} \\ h^c_{2t-1} \end{bmatrix}$$

where $pos$ is the position embedding , $s_t, a_t$ are represented using one-hot embedding, $h^b_{2t-1}$ is used to store the policy at time $t$ given current state $s_t$. we add an empty token $h_{2T} = [0_{|\mathcal{S}|+|\mathcal{A}|+1} \quad pos_{2T}]^T$ to store intermediate calculations. We also include in the tokens the position embedding $pos_i = (i, i^2, 1)^T$ for $i \in [2T]$. We define the token matrix $H_t = [h_0, \cdots, h_{2t-1}] \in \mathbb{R}^{D_{dim} \times 2t}, D_{dim} = |\mathcal{S}| + |\mathcal{A}| + 4$ for all $t \in [T]$.

**Pretraining**

During pretraining the Transformer $TF_\theta$ takes in $H^{pre}_T = [h_0, \cdots, h_{2T-1}]$ as the input matrix, and generates $H^{post}_T = TF_\theta(H^{pre}_T)$ as the output. from each $t$, suppose the linear extraction map $\mathcal{G}$ , $h^{out}_t = H^{post}_T[:, 2t-1]$, $\hat{Q}_\theta(x_t, a_t) = [\mathcal{G}h^{out}_t]_{a_t}$, where $[\mathcal{G}h]_a$ be the logit of line $a$.
Calculating $y_t = r_t + \max_{a'} \hat{Q}_\theta(x_{t+1}, a')$, we then updating the parameter $\theta \in \Theta$ by gradient descent method.

**Rollout**

At online deployment time, we initialize an empty context dataset $D_0$. At each timestep $t \in [T]$, given the accumulated context $D_{t-1}$ and the current state $s_t$, we construct the token matrix $H^{pre}_{roll,t} = [h_0, \cdots, h_{2(t-1)-1}, h_{2(t-1)}]$, and generates $H^{post}_{roll,t} = TF_{\hat{\theta}}(H^{pre}_{roll,t})$, from the last output token corresponding to the current state, $h^{out}_{roll,t} = H^{post}_{roll,t}[:, 2t-1]$, we compute the action logits via the extraction map $\mathcal{G}$ as $\{[\mathcal{G}_a h^{out}_{roll,t}]\}_{a \in \mathcal{A}}$. The agent selects the action $a_t$ according to a greedy policy over these logits $a_t = \arg\max_a [\mathcal{G}_a h^{out}_{roll,t}]$. After selecting, the agent executes the action, observes the reward $r_t$ and the next state $s_{t+1}$, then construct the new token $h_{2(t-1)+1}$ and append it to the context dataset $D_t$.
Given the input token matrix $H^{pre}_{roll,t}$, we construct a Transformer that implements the following steps on the last token. For each token $h^{pre}_{2t-1}$, the Transformer implements the following step-by-step structured transformations:

$$\begin{bmatrix} h^{pre,a}_{2t-1} \\ h^{pre,b}_{2t-1} \\ h^{pre,c}_{2t-1} \end{bmatrix} \xrightarrow{\text{Step 1}} \begin{bmatrix} h^{pre,a}_{2t-1} \\ 0_{\mathcal{A}} \\ r_t \\ pos_{2t-1} \end{bmatrix} \xrightarrow{\text{Step 2}} \begin{bmatrix} h^{pre,a}_{2t-1} \\ \max_{a'} \hat{Q}_\theta(x_{t+1}, a') \\ r_t \\ pos_{2t-1} \end{bmatrix}$$

$$\xrightarrow{\text{Step 3}} \begin{bmatrix} h_{2t-1}^{pre,a} \\ y_t = r_t + \max_{a'} \hat{Q}_\theta(x_{t+1}, a') \\ \star \\ pos_{2t-1} \end{bmatrix} \xrightarrow{\text{Step 4}} \begin{bmatrix} h_{2t-1}^{post,a} \\ h_{2t-1}^{post,b} \\ h_{2t-1}^{post,c} \end{bmatrix}$$

given the current context $x_t = (D_{t-1}, s_t)$, the Transformer $TF_\theta(\cdot)$ outputs a token $h_{2t-1}^{out}$, from which the Q-values $\hat{Q}_\theta(x_t, a)$ are extracted. Each step is defined as below:

**Step 1 (Reward Extraction)**

There exists a attention-only Transformer $\text{TF}_\theta(\cdot)$ that implements Step 1.

**Proof of Step 1** We prove this step by constructing a Transformer that add $r_t$ from $h_{2t-1}^b$ to $h_{2t+1}^b$. We can construct a two-layer attention-only Transformer with $Q_{1,2}^{(1)}, K_{1,2}^{(1)}, V_{1,2}^{(1)}$, such that for all $i \leq 2t-1$,

$$Q_1^{(1)} h_i^{(0)} = \begin{bmatrix} 1 \\ i \end{bmatrix}, \quad K_1^{(1)} h_i^{(0)} = \begin{bmatrix} i+2 \\ -1 \end{bmatrix}, \quad V_1^{(1)} h_{2t-1}^{(0)} = \begin{bmatrix} 0_{|\mathcal{S}|} \\ 0_{|\mathcal{A}|} \\ r_t \end{bmatrix}, \quad V_1^{(1)} h_{2t}^{(0)} = \begin{bmatrix} 0_{|\mathcal{S}|} \\ 0_{|\mathcal{A}|} \\ 0 \end{bmatrix},$$

and we choose $Q_2^{(1)} = Q_1^{(1)}, V_2^{(1)} = -V_1^{(1)}, K_2^{(1)} h_i^{(0)} = \begin{bmatrix} i+1 \\ -1 \end{bmatrix}$, summing up the heads, we obtain the update on a subset of coordinates in $h_{2t+1}^{(0),b}$ as

$$0_{|\mathcal{S}|+|\mathcal{A}|+1} \to 0_{|\mathcal{S}|+|\mathcal{A}|+1} + \sum_{j=1}^{2} \sum_{i=1}^{2t+1} \sigma(\langle Q_j^{(1)} h_{2t+1}^{(0)}, K_j^{(1)} h_i^{(0)} \rangle) V_j h_i^{(0)} = \frac{1}{2t+1} \begin{bmatrix} 0_{|\mathcal{S}|} \\ 0_{|\mathcal{A}|} \\ r_t \end{bmatrix}$$

We then use another attention layer to multiply the updated vectors by a factor of $2t+1$.

Choosing $Q_1^{(2)} h_i^{(1)} = \sqrt{2t+1} \cdot e_i$, $K_1^{(2)} h_j^{(1)} = \sqrt{2t+1} \cdot e_j$, $V_1^{(2)} h_{2t+1}^{(1)} = \frac{1}{2t+1} \begin{bmatrix} 0_S \\ 0_A \\ r_t \end{bmatrix}$, and noting that $\langle Q_1^{(2)} h_i^{(1)}, K_1^{(2)} h_j^{(1)} \rangle = 2t+1$, when $j = i$ and otherwise 0.

**Step 2 (Future Q-Value Lookup)**

Attend to the next state token $h_{2t}^{\text{pre}}$, and extract the maximum predicted $\max_{a'} \hat{Q}_\theta(x_{t+1}, a')$. There exists a Transformer $\text{TF}_\theta(\cdot)$ that implements Step 2.

**Proof of Step 2**

Given $(\hat{Q}_\theta)_T = 0$, we start with constructing an-attention layer, and $\{Q_{jt,s}\}_{s=1}^2$,$\{K_{jt,s}\}_{s=1}^2$,$\{V_{jt,s}\}_{s=1}^2$ such that for all $i \leq 2t-1$ and $j \leq i$,

$$Q_{jt,1}^{(1)} h_i^{(0)} = \begin{bmatrix} x_t \\ -i \\ 3T \end{bmatrix}, \quad K_{jt,1}^{(1)} h_i^{(0)} = \begin{bmatrix} \hat{Q}_t(\cdot, a_j) \\ 3T \\ j \end{bmatrix}, \quad V_{jt,1}^{(1)} h_i^{(0)} = \begin{bmatrix} 0 \\ i e_{jt} \\ 0 \end{bmatrix}, \quad Q_{jt,2}^{(1)} h_i^{(0)} = \begin{bmatrix} -x_t \\ -i \\ 3T \end{bmatrix},$$

$K_{jt,2}^{(1)} = K_{jt,1}^{(1)}, V_{jt,2}^{(1)} = -V_{jt,1}^{(1)}$.
where $e_{jt}$ is a one-hot vector supported on some entry of $h^c$ . Summing up two heads gives the update for $i \leq 2t-1$,

$$0 \to 0 + \left[ \sigma(\langle Q_{jt,1}^{(1)} h_i^{(0)}, K_{jt,1}^{(1)} h_i^{(0)} \rangle) - \sigma(\langle Q_{jt,2}^{(1)} h_i^{(0)}, K_{jt,2}^{(1)} h_i^{(0)} \rangle) \right] e_{jt} = \hat{Q}_t(x_t, a_j) e_{jt}$$

Denote the resulting token vector by $h_i^{(1)}$.

Next, we construct a MLP layer, s.t for any $x \in \mathcal{X}$ on the corresponding coordinates we have

$$W_1^{(2)} h_i^{(1)} = \begin{bmatrix} \vdots \\ \hat{Q}_t(x, a_1) \\ \hat{Q}_t(x, a_2) - \hat{Q}_t(x, a_1) \\ \vdots \\ \hat{Q}_t(x, a_{|\mathcal{A}|}) - \hat{Q}_t(x, a_{|\mathcal{A}|-1}) \\ \vdots \end{bmatrix}$$

where $a_k$ denotes the $k^{th}$ action, and

$$W_2^{(2)} \sigma(W_1^{(2)} h_i^{(1)}) = \sigma(\hat{Q}_t(x, a_1)) + \sum_{k=2}^{A} \sigma(\hat{Q}_t(x, a_k) - \hat{Q}_t(x, a_{k-1})) = \max_{a \in A} \hat{Q}_t(x, a).$$

**Step 3 (TD Target Construction)**

Combine $r_t$ and $\max_{a'} \hat{Q}(x_{t+1}, a')$ to form the TD target:

$$y_t = r_t + \max_{a'} \hat{Q}(x_{t+1}, a').$$

There exists a Transformer $TF_\theta(\cdot)$ that implements Step 3.

**Proof of Step 3**

The proof is similar to that of the literature Wang et al. [2024a]. Let

$$Q_1^{(1)} h_i^{(0)} = K_1^{(1)} h_i^{(0)} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad V_1^{(1)} h_{2t-1}^{(0)} = \begin{bmatrix} 0_{|\mathcal{S}|} \\ r_t \\ 0 \end{bmatrix},$$

$$Q_2^{(1)} h_i^{(0)} = K_2^{(1)} h_i^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad V_2^{(1)} h_{2t}^{(0)} = \begin{bmatrix} 0_{|\mathcal{S}|} \\ \max_{a'} \hat{Q}_\theta(x_{t+1}, a') \\ 0 \end{bmatrix},$$

then we have that

$$h_{2t-1}^{(1)} = \sum_{j=1}^{2} \sum_{i=1}^{2t} \sigma(\langle Q_j^{(1)} h_{2t-1}^{(0)}, K_j^{(1)} h_i^{(0)} \rangle) V_j^{(1)} h_i^{(0)} = \begin{bmatrix} 0_{|\mathcal{S}|} \\ r_t \\ \max_{a'} \hat{Q}_\theta(x_{t+1}, a') \\ 0 \end{bmatrix},$$

define a MLP layer with $W_1^{(2)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $W_2^{(2)} = [0 \quad 1 \quad 1]$. We can get that $W_2^{(2)} \sigma(W_1^{(2)} h_{2t-1}^{(1)}) = r_t + \max_{a'} \hat{Q}_\theta(x_{t+1}, a')$.

**Step 4 (Q-Value Update):**

There exists a small MLP head (two-layer feed-forward network) which updates $h_{2t-1}$ to regress the Q-value prediction $\hat{Q}(x_t, a_t)$ towards the TD target $y_t$.

**Proof of Step 4**

The proof is related to the literature von Oswald et al. [2023]. Let

$$Q_1^{(1)} h_i^{(0)} = K_1^{(1)} h_i^{(0)} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad V_1^{(1)} h_{2t-1}^{(0)} = \begin{bmatrix} 0_{|\mathcal{S}|} \\ \hat{Q}_\theta(x_t, a_t) \\ 0 \\ 0 \end{bmatrix},$$

$$Q_2^{(1)} h_i^{(0)} = K_2^{(1)} h_i^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad V_2^{(1)} h_{2t-1}^{(0)} = \begin{bmatrix} 0_{|\mathcal{S}|} \\ 0 \\ y_t \\ 0 \end{bmatrix}$$

combining two attention heads, we get that $h_{2t-1}^{(1)} = \begin{bmatrix} 0_{|\mathcal{S}|} \\ \hat{Q}_\theta(x_t, a_t) \\ y_t \\ 0 \end{bmatrix}$. define a MLP layer with $W_1^{(2)} = \begin{bmatrix} \cdots & & & \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \cdots & & & \end{bmatrix}$, $W_2^{(2)} = [0 \quad 1-\alpha \quad \alpha \quad 0]$, where $\alpha$ be a coefficient. We have that $W_2^{(2)}\sigma(W_1^{(2)}h_{2t-1}^{(1)}) = (1-\alpha)\hat{Q}_\theta(x_t, a_t) + \alpha y_t$.

□

# B Experiment Details

## B.1 Computing Resources

All experiments are conducted on two NVIDIA A100 GPUs (40GB each).

## B.2 LinUCB Algorithm

Let $T$ denote the time horizon and $\lambda, \alpha > 0$ be input parameters. At each time step $t \in \{1, 2, ..., T\}$, the LinUCB algorithm operates through the following steps:

1. Compute the ridge estimator for the weight vector:
$$\mathbf{w}_{\text{ridge},\lambda}^t = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \left( \frac{1}{2t} \sum_{j=1}^{t-1} (r_j - \langle \mathbf{a}_j, \mathbf{w} \rangle)^2 + \frac{\lambda}{2t} \|\mathbf{w}\|_2^2 \right)$$
2. For each action $i \in [A]$, compute the upper confidence bound:
$$v_{t,i}^* = \langle \mathbf{a}_{t,i}, \mathbf{w}_{\text{ridge},\lambda}^t \rangle + \alpha \sqrt{\mathbf{a}_{t,i}^\top \mathbf{A}_t^{-1} \mathbf{a}_{t,i}},$$
where $\mathbf{A}_t = \lambda \mathbf{I}_d + \sum_{j=1}^{t-1} \mathbf{a}_j \mathbf{a}_j^\top$.
3. Select the action $a_{t,j}$ by:
$$j := \arg\max_{i\in[A]} v_{t,i}^*.$$

## B.3 Darkroom Environment

Darkroom is considered as a complex Markov decision process (MDP) problem and is utilized as a standard benchmark for evaluating in-context learning Laskin et al. [2022], Lee et al. [2023]. In this experiment, the agent must locate an unknown goal within a $10 \times 10$ darkroom. The agent receives a reward of 1 only when it reaches the goal. At each step, the agent can choose from five possible actions: move up, down, left, right, or stay still. If the agent is not at the goal, it receives a reward of 0. The horizon for the Darkroom environment is set to 100 steps. We summarize the details as follows:

**Pretraining Data Collection** Similar to the stochastic linear bandit problem, we consider two types of policies: A random policy, which selects an action randomly at each position, and an expert policy, which chooses legal actions to avoid crashing into walls and will stay still once the agent receives a reward of 1. To test whether the pretrained model can generalize to **unseen** RL problems in context, we collect datasets from 80 out of the total 100 goals, reserving the remaining 20 for testing. For each training goal, we run both the random and expert policies, collecting 1k trajectories from each policy. This leads to a total of 80k trajectories for the random policy and 80k trajectories for the expert policy, resulting in 160k context trajectories in the pretraining dataset.

**Comparison and Implementation** We evaluate QTPT and SPT under two settings: pretraining on purely random data and on purely expert data. The models are evaluated on an **unseen** task, where the target goal is not included in the pretraining dataset. The model architecture is identical to that used in the stochastic linear bandit experiments (see Section B.6).

### B.4 Dark Key-to-Door Environment

The Dark Key-to-Door environment extends the complexity of the Darkroom setting by introducing a hierarchical task structure with sparse rewards. In this environment, the agent must sequentially accomplish two objectives: first locate an invisible key to receive a reward of $r = 1$, then find and open an invisible door to receive an additional reward of $r = 1$. This creates a challenging sparse reward scenario where the agent must learn to complete subtasks in the correct order. The environment consists of a $9 \times 9$ grid, and each episode is limited to 50 steps. At the beginning of each episode, the agent is reset to position $(0, 0)$, while the key and door locations are randomly generated across different episodes. Since door and key can be placed in any positions, there are total $81 \times 81 = 6561$ distinct tasks.

**Pretraining Data Collection** To capture diverse exploration strategies, we collect our pretraining data using two distinct behavioral policies. The expert policy employs a systematic spiral search pattern, ensuring comprehensive coverage of the environment while efficiently locating the key and door. In contrast, the random policy selects actions uniformly at random, providing diverse but generally suboptimal exploration patterns. To ensure a balanced representation of both behaviors, we collect 80k trajectories from each policy type. The pretraining dataset is partitioned into 5249 tasks, which represents approximately 80% ($6561 \times 0.8 \approx 5249$) of the total tasks. The remaining 1312 tasks are held out for testing and are thus considered **unseen**.

**Comparison and Implementation** We evaluate the performance of our QTPT and SPT models under two distinct pretraining configurations: one trained exclusively on random policy data and the other trained exclusively on expert policy data. The evaluation, conducted on **unseen tasks**, specifically assesses the models' ability to generalize the underlying hierarchical task structure and adapt to new key-door configurations during in-context learning. To ensure a fair comparison across environments and configurations, the model architecture is kept consistent with that of previous experiments.

### B.5 Miniworld Environment

To evaluate QTPT's effectiveness in visual domains, we employ the Miniworld [Lee et al., 2023] navigation benchmark, which presents a visually grounded navigation task. In this environment, agents must navigate to the correct colored target box among four boxes positioned at the corners of the environment. The agent receives $25 \times 25$ RGB observations with directional conditioning, providing rich visual information that requires the model to process high-dimensional sensory input. The agent has access to three discrete actions: turn left, turn right, and move forward, which creates a more realistic navigation scenario compared to grid-world abstractions. The agent receives a reward of $r = 1$ only when positioned near the target box, with all other states yielding zero reward. Each episode is constrained to 50 timesteps, requiring efficient navigation strategies.

**Pretraining Data Collection** We construct our pretraining dataset using two complementary data collection strategies to capture both optimal and suboptimal navigation behaviors. The expert policy demonstrates intelligent navigation by selecting legal actions that avoid wall collisions and environmental boundaries, and crucially, the agent remains stationary once it successfully reaches the target and receives the reward signal. This policy represents efficient goal-directed behavior in the visual navigation domain. The random policy, conversely, selects actions uniformly at random, generating diverse but often inefficient exploration trajectories that cover various parts of the state space. We collect 24k trajectories from each policy type for training.

**Comparison and Implementation** The evaluation protocol examines QTPT and SPT performance under both pure random data pretraining and pure expert data pretraining conditions. The model architecture incorporates appropriate visual encoding mechanisms while maintaining consistency with the core transformer structure used in other experimental settings.

### B.6 Implementation Detail

The main linear-bandit decoder is GPT-2-style [Garg et al., 2022], with 8 layers, 4 attention heads, and embedding dimension $D = 256$. Its custom ReLU-attention implementation is not a standard

GPT-2 block: it omits the usual attention output projection and uses a $256 \to 256 \to 256$ MLP rather than $256 \to 1024 \to 256$. The implemented model has 2,858,763 parameters. The original training setup uses batch size 32 and Adam, cosine learning-rate decay with a 2,000-step warmup, peak learning rate $5 \times 10^{-6}$, minimum $10^{-7}$, and weight decay 0.001. Supplementary experiments state any changes to this setup explicitly.

Each model is pretrained for 50 epochs on the collected datasets. After pretraining, the models are evaluated on a test set to assess their effectiveness in the context of the stochastic linear bandit problem.

# C Additional Experiments

## C.1 Ablation Study on the Stochastic Linear Bandit Problem

We also consider using 'softmax' operation when computing the Q-target in experiments. To evaluate the robustness of QTPT, we conducted ablation experiments examining several key factors: The impact of using either 'softmax' or 'hardmax', the effect of incorporating a double DQN framework for network updates, and the differences between ReLU and softmax activation functions in computing attention scores. Note that our original implementation combines 'softmax', ReLU, and a double DQN framework.

Table 3: Comparison of results for different configurations of QTPT.

| **Random Data Results** | Default | Double DQN | Softmax + Hardmax | ReLU-Attn + Softmax-Attn |
|---|---|---|---|---|
| Results | 30.9 | 34.01 | 32.48 | 197.26 |
| **Expert Data Results** | Default | Double DQN | Softmax + Hardmax | ReLU-Attn + Softmax-Attn |
| Results | 39.4 | 45.76 | 53.73 | 203.74 |

In addition, we vary the embedding dimension, number of layers, and number of attention heads to evaluate the model's performance under different configurations. Note that the original configuration consists of 8 layers, 4 attention heads, and an embedding dimension of 256, pretrained on purely random data.

Table 4: Results for varying (a) embedding dimension, (b) number of attention heads, and (c) number of layers. All other settings follow the original configuration.

(a) Embedding dimension

| Dimension | Results |
|---|---|
| 16 | 135.7 |
| 32 | 54.0 |
| 64 | 40.5 |
| 128 | 33.3 |
| 256 | 30.9 |

(b) Attention heads

| Heads | Results |
|---|---|
| 1 | 34.2 |
| 2 | 30.2 |
| 4 | 30.9 |
| 8 | 29.3 |
| 16 | 31.4 |

(c) Layers

| Layers | Results |
|---|---|
| 1 | 122.3 |
| 2 | 33.6 |
| 4 | 31.0 |
| 8 | 30.9 |

## C.2 Experiments on Non-stationary Environment.

We aim to demonstrate the robustness of QTPT by evaluating its performance in non-stationary environments.

**Experiment Setup:** The pretraining phase remains consistent with the process described in Section 4. Yet, in the test phase, we introduce two different settings:

- **Stationary:** The parameter $\theta^*$ is sampled from a uniform distribution $[0, 1]^d$, which matches the conditions of the pretraining stage;

- **Non-stationary:** The parameter $\theta^*$ is sampled from a standard Gaussian distribution $N(0,1)^d$ and rescaled to lie within the range $[0,1]^d$. The rescaling is performed using the transformation:

$$\theta_{\text{scaled}} = \frac{\theta - \theta_{\min}}{\theta_{\max} - \theta_{\min}},$$

where $\theta_{\min}$ and $\theta_{\max}$ represent the minimum and maximum values of the sampled $\theta$. In our experimental setup, we sample 1000 values from the Gaussian distribution and take the minimum and maximum from these samples for rescaling.

Figure 4 demonstrates that QTPT achieves superior performance in the non-stationary environment compared to the stationary setting. While SPT maintains relatively consistent performance across both environments, it significantly underperforms compared to Q-learning, especially under non-stationary conditions. These results highlight QTPT's enhanced adaptability to changing environmental dynamics and robustness to distributional shifts in the underlying parameters. Notably, the performance gap between QTPT and SPT widens considerably in the non-stationary setting, with QTPT maintaining strong performance while SPT's effectiveness diminishes.

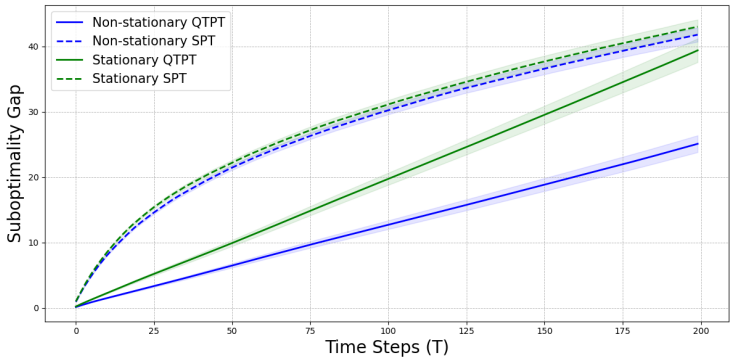



Figure 4: Comparison of Pretrained Transformer Performance Using Q-learning and Supervised Learning in Stationary and Non-stationary Environments. The shaded regions represent the standard deviation of the suboptimality estimates based on 1000 simulation runs.

### C.3 More Complex Task: Math Reasoning

**Algorithm 2** QTPT with Policy Optimization (QTPO)

1: Estimate behavior policy $\hat{\pi}_\beta$ by behavior cloning on offline dataset $D$
2: Training the Value function $\hat{V}_{\pi_\beta}$ and advantage function $\hat{A}_{\pi_\beta}$ together by dueling architecture using QTPT on offline dataset $D$.
3: Initialize $k = 0$ and set $\pi_k \leftarrow \hat{\pi}_\beta$
4: **for** $i = 0, 1, 2, \cdots, I$ **do**
5: Update the policy $\pi$ by maximizing $L_k(\pi)$
6: **if** $J(\pi) > J(\pi_k)$ **then**
7: Set $k = k + 1 \& \pi_k \leftarrow \pi$
8: **end if**
9: **end for**

To demonstrate the broader applicability of QTPT, we extended our approach to mathematical reasoning tasks. Each math problem serves as a distinct environment, where generated tokens represent states, the full sequence forms the context-state, and token selection constitutes actions. The reward signal is derived from binary correctness or step-by-step reasoning accuracy. Our method showed consistent performance improvements over supervised learning baselines in these tasks.

A key challenge emerged during preliminary experiments is that when learning a Q-function from offline data alone, and only use the Q-function to generate token, the model frequently produced degenerate outputs (e.g., generating repetitive tokens). We hypothesize that this is due to semantic

distortion caused by the Bellman update. Specifically, the output in language models consists of token probabilities, and applying cumulative reward-based reasoning to these probabilities disrupts their linguistic coherence, as it is unconventional to treat them as a direct sum of the reward and the probability of the next token.

To address this challenge, we adopt an actor-critic framework in which the Q-function learned by QTPT is not used directly for token generation but to refine the policy. In the offline setting, we modified the Proximal Policy Optimization (PPO) algorithm, drawing inspiration from Behavior Proximal Policy Optimization (BPPO)Zhuang et al. [2023], and incorporated an iterative refinement process, which we name *QTPT with Policy Optimization*(QTPO). The details are provided in Algorithm 2. where

$$L_k(\pi) = \mathbb{E}_{s\sim\rho_{\mathcal{D}}(.),\, a\sim\pi_k(.||s)}\left[\min\left(\frac{\pi(a|s)}{\pi_k(a|s)}\hat{A}(s,a), \text{clip}\left(\frac{\pi(a|s)}{\pi_k(a|s)},\, 1-\epsilon,\, 1+\epsilon\right)\hat{A}(s,a)\right)\right] \tag{6}$$

and the objective $J(\pi)$ is the total accuracy on the offline dataset.

**Data Preparation** We evaluated our method on the GSM8K dataset Cobbe et al. [2021], utilizing reward labels and training data from Math-Shepherd Wang et al. [2024b]. The experimental setup involved two distinct data categories:

- **Expert Data**, consisting of samples with correct final answers
- **Suboptimal Data**, comprising samples with incorrect answers.

Each category contained 50,000 samples.

**Implementation and Comparison** We initialized both the policy and Q-function models using the Qwen-2.5-1.5B-instruct model. These models were then trained using Low-Rank Adaptation (LoRA) with the following hyperparameters: rank=8, lora_alpha=32, and dropout=0.1. We evaluated performance on the **unseen** GSM8k test set. This evaluation compared QTPT against direct behavior cloning, using models trained on various mixtures of expert and imperfect data.

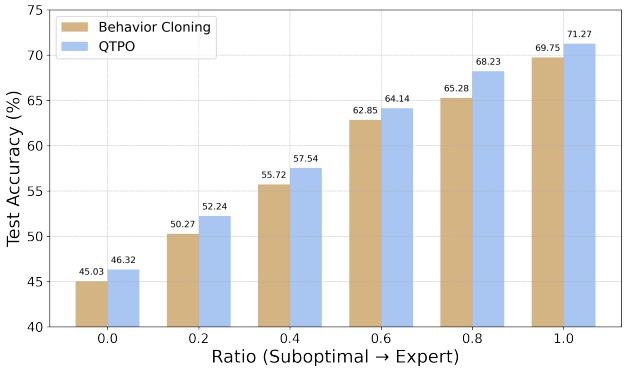



Figure 5: Accuracy (%) on the GSM8K test set across different mixtures of suboptimal and expert data.

**Results** The results of these comparisons are presented in Figure 5. QTPT consistently outperformed behavior cloning across all settings, achieving an average accuracy improvement of **1.81%**. This consistent enhancement is particularly noteworthy as our method relies exclusively on a static offline dataset.

### C.4 D4RL Extension and Controlled Ablations

The D4RL extension uses Franka Kitchen and AntMaze [Fu et al., 2020]. The summaries reported in Table 2 are $13.23 \pm 4.03$ for `kitchen-partial-v0`, $14.62 \pm 3.36$ for `kitchen-mixed-v0`, and $35.94 \pm 13.52$ for `antmaze-umaze-v2`. The first two are normalized scores; the last is success percentage. These are descriptive 4-seed matched-Transformer results. Table 1 compares the TD,

Monte Carlo, and no-context variants. Its intervals are over evaluation episodes, not independent training seeds. The context and OOD diagnostics below address distinct mechanisms under separately stated protocols; they do not turn episode-level uncertainty into seed-level significance.

### C.5 Comparison with Other Baselines

To provide a comprehensive and principled evaluation, we compare QTPT against representative methods from two major paradigms of offline decision learning, distinguished by their learning mechanisms rather than surface algorithmic forms.

**(1) Bellman-based value-function-centric offline RL.** This class is represented by Conservative Q-Learning (CQL, Kumar et al. [2020]), which explicitly relies on Bellman backups and value function optimization with conservative regularization to address distributional shift and out-of-distribution actions. Policy extraction is performed through value maximization over the learned Q-function.

**(2) Value-guided policy learning.** IQL [Kostrikov et al., 2021] trains its critic with an in-sample value backup and extracts a policy by advantage-weighted imitation. Q-SFT [Hong et al., 2025] and QDT [Yamagata et al., 2023] provide value-guided supervision for sequence-model policies. They differ in their critic construction and targets; the presence of a supervised policy loss does not make IQL Bellman-free.

The linear-bandit comparison uses identical offline data and no test-time parameter updates. The Q-guided policy variants share a pretrained critic in this implementation. Consequently, these labels identify matched value-guided variants, not unchanged implementations of every original algorithm; in particular, the shared-critic IQL-style policy is not a full independent IQL reproduction. We retain the figure as that matched comparison, not as a universal ranking of the original offline-RL algorithms.

As shown in Figure 6, QTPT achieves lower cumulative regret than the matched baselines. On the expert dataset, baselines like QDT and CQL achieve moderate performance but still incur higher cumulative regret compared to QTPT. The performance gap becomes more pronounced on the random dataset. In this setting, QTPT maintains a low-regret profile, demonstrating robustness to suboptimal data.

The figure is descriptive evidence about the matched implementations. In the linear bandit, past action–reward observations provide information about the hidden task parameter, so it does test task inference; it does not isolate every aspect of that inference. Appendix D.5 separately tests the role of task-relevant history.

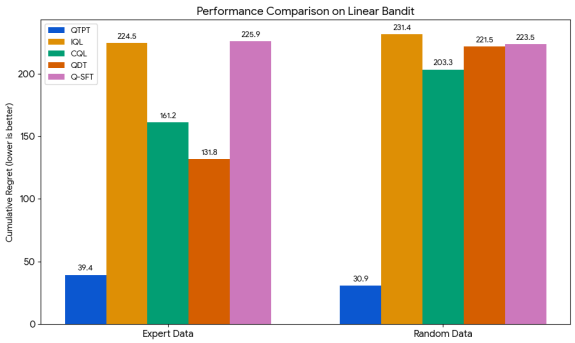



Figure 6: Performance comparison against strong baselines on the Stochastic Linear Bandit task.

## D Supplementary Evaluation and Clarifications

### D.1 Target Construction, Stochasticity, and Cost

**Idealized objective and practical backup.** The formal analysis uses the undiscounted expected Bellman operator. Our implementation uses online values for action weights and a separate target

network for evaluation [van Hasselt et al., 2016]:

$$p_\theta(a \mid x_{t+1}) = \frac{\exp(Q_\theta(x_{t+1}, a)/\tau)}{\sum_b \exp(Q_\theta(x_{t+1}, b)/\tau)}, \qquad y_t = r_t + \gamma \sum_a p_\theta(a \mid x_{t+1}) Q_{\bar{\theta}}(x_{t+1}, a).$$

Here $x_t = D_{t-1} \cup \{s_t\}$, $\tau = 1$, and target parameters are copied from the online model every 100 optimizer steps. The support-controlled diagnostic below uses $\gamma = 0.99$. Gradients do not pass through $y_t$. This Double-target/softmax combination stabilizes target construction but supplies no conservative lower-bound guarantee for unsupported actions.

**Stochastic targets.** For fixed next-step values, write $Y_t = R_t + \max_{a'} Q_{t+1}(X_{t+1}, a')$ and $m_t(x, a) = \mathbb{E}[Y_t \mid X_t = x, A_t = a]$. Then

$$\mathbb{E}[(f(X_t, A_t) - Y_t)^2] = \mathbb{E}[(f(X_t, A_t) - m_t(X_t, A_t))^2] + \mathbb{E}[\mathrm{Var}(Y_t \mid X_t, A_t)].$$

For this fixed-target regression, the last term does not depend on $f$. Stochastic rewards and transitions change target variance, not the population regression minimizer. The identity does not assert that the variance is constant when the target network changes between updates.

**Memory and computation.** Time-indexed Q-functions share one Transformer. Targets are constructed per batch, not stored for every offline trajectory. A model with $P$ parameters requires one additional $O(P)$ target copy, one no-gradient target forward pass per batch, and a hard copy every 100 updates. The target carries neither gradients nor optimizer states and is absent at deployment. For the custom 2,858,763-parameter bandit model, the FP32 copy is approximately 10.91 MiB. Reduced-precision storage and sharding are possible scaling options, not additional experiments reported here.

**Attention versus backup sensitivity.** For fixed $q \in \mathbb{R}^{|\mathcal{A}|}$ and $p = \mathrm{softmax}(q/\tau)$,

$$0 \le \max_a q_a - \sum_a p_a q_a \le \tau \log |\mathcal{A}|.$$

For fixed $p$, replacing online by target-network evaluation changes the weighted value by at most $\|Q_\theta(x, \cdot) - Q_{\bar{\theta}}(x, \cdot)\|_\infty$. These are one-step target comparisons, not guarantees about a complete nonlinear training trajectory. Attention normalization instead changes the history representation. For identical positive attention logits $M$ and value vectors $v$, length-normalized ReLU aggregation gives $Mv$, whereas softmax gives $v$. This illustrates a scale difference not controlled by the backup bound. It offers a plausible representation-level explanation for the original attention ablation, without proving a unique cause of its performance collapse. Attention choices should be validated separately from target-construction hyperparameters.

### D.2 Unsupported-Action Overestimation

**Controlled support and training protocol.** We use a stochastic linear bandit with ten five-dimensional action vectors and $T = 200$. Rewards are $r_t = a_t^\top w^\star + \epsilon_t$, $\epsilon_t \sim \mathcal{N}(0, 1.5^2)$. For each task, seven actions form $S$ and the other three form $U$. Behavior samples uniformly from $S$; every supported action appears at least once, and no action in $U$ is executed in the offline trajectory. All ten vectors remain visible. The split is randomized across tasks, testing behavior-support OOD rather than a fixed action-index rule.

Each paired seed uses 10,000 offline tasks, an 8,000/2,000 train/validation split, and 1,000 disjoint test tasks. The model has 8 layers, width 256, 4 heads, dropout 0.2, batch size 128, learning rate $5 \times 10^{-6}$, and weight decay $10^{-3}$. Training is capped at 100 epochs, with patience 20. The minimum held-out validation objective selects the checkpoint, never test performance.

The $2 \times 2$ design compares QTPT with Double-target/softmax backup against single-network hardmax QTPT, with or without the discrete CQL(H) penalty [Kumar et al., 2020]:

$$L_{\mathrm{TD}} + \alpha L_{\mathrm{CQL}}, \qquad L_{\mathrm{CQL}} = \mathbb{E}\left[\log \sum_a \exp Q_\theta(x_t, a) - Q_\theta(x_t, a_t^{\mathrm{data}})\right], \qquad \alpha = 0.03.$$

QTPT + CQL penalty adds this term to QTPT; it is not an independent MLP CQL baseline. All four cells use paired data and evaluation tasks.

**Exact terminal calibration.** After $T-1$ support-only observations, no future reward remains beyond the final action, so the task-specific oracle value is exactly $\mu(a) = a^\top w^\star$. Define

$$\Delta_{\mathrm{OOD}} = \frac{1}{|U|}\sum_{a\in U}(\hat{Q}_T(x_T,a) - \mu(a)) - \frac{1}{|S|}\sum_{a\in S}(\hat{Q}_T(x_T,a) - \mu(a)),$$

$$G_{\mathrm{OOD}} = \left[\max_{a\in U}\hat{Q}_T(x_T,a) - \max_{a\in S}\hat{Q}_T(x_T,a)\right] - \left[\max_{a\in U}\mu(a) - \max_{a\in S}\mu(a)\right].$$

For greedy action $a_g$, harmful selection means $a_g \in U$ and $\mu(a_g) < \max_{a\in S}\mu(a)$. OOD-attributable regret is this event's indicator times $\max_{a\in S}\mu(a) - \mu(a_g)$, averaged over all tasks. Separately, full-rollout regret is $\sum_{t=1}^{T}[\max_a \mu(a) - \mu(a_t)]$ when all actions are available from step one. Correctly choosing an optimal unsupported action is not a failure.

Table 5: Support-controlled diagnostic. Mean $\pm$ standard deviation across five paired training/data seeds. Lower is better; harmful rate is a fraction.

| Method | $G_{\mathrm{OOD}} \downarrow$ | Harmful rate $\downarrow$ | OOD regret $\downarrow$ | Rollout regret $\downarrow$ |
|---|---|---|---|---|
| QTPT | $0.2569 \pm 0.1021$ | $0.2064 \pm 0.0362$ | $0.1828 \pm 0.0754$ | $153.15 \pm 58.99$ |
| Naive hard-max QTPT | $0.3351 \pm 0.0387$ | $0.2588 \pm 0.0090$ | $0.2856 \pm 0.0157$ | $211.67 \pm 7.81$ |
| QTPT + CQL penalty | $0.2582 \pm 0.1021$ | $0.2144 \pm 0.0390$ | $0.1903 \pm 0.0675$ | $139.86 \pm 55.74$ |
| Naive hard-max + CQL penalty | $0.3330 \pm 0.0386$ | $0.2514 \pm 0.0084$ | $0.2840 \pm 0.0176$ | $205.40 \pm 10.36$ |

**Interpretation.** Task averages are computed within each seed before seed-level means, sample standard deviations, and Student-$t$ intervals. The offset $\Delta_{\mathrm{OOD}}$ is near zero (between $-0.0066$ and $-0.0033$), but QTPT's max-gap, harmful selection, and OOD regret have 95% intervals strictly above zero. This is policy-relevant max-selection error, not uniform inflation of every unsupported value.

Naive hard-max minus QTPT increases harmful selection by 0.0524 and OOD regret by 0.1028; both paired 95% intervals are positive. The combined stabilization reduces harm, without isolating each stabilizer's contribution. In either backup branch, paired intervals for CQL-induced changes in all four metrics contain zero. This does not establish reliable incremental improvement, nor prove equivalence.

Some unsupported actions are genuinely optimal and inferable through shared task structure. Support is context- and task-conditional. CQL is evaluated per context, not as a global blacklist, but its all-action penalty does not explicitly distinguish harmful unsupported actions from beneficial generalization. This is a plausible explanation, not a demonstrated causal mechanism.

Table 6: Darkroom diagnostic: five-seed held-out reward, mean $\pm$ standard deviation, validation-best checkpoints. This protocol is separate from Table 1.

| Expert mix | QTPT reward $\uparrow$ | QTPT + CQL penalty reward $\uparrow$ |
|---|---|---|
| 0.0 | $8.3053 \pm 8.7520$ | $9.3333 \pm 8.7971$ |
| 0.5 | $15.7753 \pm 10.7997$ | $14.0393 \pm 7.8813$ |
| 1.0 | $6.2247 \pm 1.9315$ | $8.9653 \pm 4.2767$ |

These runs use $d = 10$, $T = 100$, five actions, 10,000 training and 100 test environments, and 4-layer, width-128, 4-head models, with 20,000 updates, batch size 128, and learning rate $3 \times 10^{-4}$. The mean difference changes direction across regimes and variability is substantial. These are performance observations, not Q-calibration measurements. Understanding conservatism in context-conditioned generalization remains a promising research direction.

### D.3 Meta-RL Comparisons

$RL^2$ [Duan et al., 2016], PEARL [Rakelly et al., 2019], and QTPT all adapt behavior using task evidence, but differ in pretraining data acquisition and representation. Matching a data budget does not make online collection identical to offline learning.

**RL$^2$: transition-budget comparison.** A five-arm, ten-pull Bernoulli bandit has unknown arm probabilities per task. QTPT receives a fixed random-behavior corpus; RL$^2$ collects fresh on-policy interactions. One pull is one transition, so 1M/5M transitions correspond to 100,000/500,000 trials. Evaluation uses 1,000 common tasks and common outcome tapes. Pseudo-regret is $\sum_{t=1}^{10}(p^\star - p_{a_t})$. Bayes-normalized progress is $100(R_{\mathrm{random}} - R)/(R_{\mathrm{random}} - R_{\mathrm{Bayes}})$, where $R_{\mathrm{random}} = 3.333333$ and $R_{\mathrm{Bayes}} = 1.673345$: random is 0% and the finite-horizon Bayes-optimal reference is 100%.

Table 7: Bernoulli bandit, mean $\pm$ standard deviation over three independent training seeds.

| Method / acquisition | Transitions | Pseudo-regret ↓ | Bayes progress ↑ |
|---|---|---|---|
| QTPT / fixed random corpus | 1M | $1.915 \pm 0.085$ | $85.4\% \pm 5.1\%$ |
| RL$^2$ / on-policy trials | 1M | $2.356 \pm 0.112$ | $58.9\% \pm 6.7\%$ |
| QTPT / fixed random corpus | 5M | $1.784 \pm 0.041$ | $93.3\% \pm 2.5\%$ |
| RL$^2$ / on-policy trials | 5M | $1.780 \pm 0.056$ | $93.6\% \pm 3.4\%$ |

QTPT has lower mean pseudo-regret at 1M transitions; at 5M the means are almost identical. This supports a budget-dependent comparison, not universal superiority of offline acquisition.

**PEARL: common fixed-offline replay.** A separate continuous ten-pull meta-bandit has latent target $M$ and expected reward $1-(a-M)^2$. Both methods receive the same 80,000 uniform-random transitions and 10,000 updates. PEARL retains probabilistic context inference, posterior sampling, and SAC losses, but replaces active collection with this replay corpus. Evaluation uses seven training seeds, 1,000 common tasks, and two common noise tapes. Checkpoints use the fixed training endpoint, not test-set selection.

Table 8: Fixed-offline continuous meta-bandit. Mean $\pm$ standard deviation over seven training seeds; pseudo-regret is $\sum_{t=1}^{10}(a_t - M)^2$.

| Method | Pseudo-regret ↓ | Mean training time (s) |
|---|---|---|
| QTPT | $0.969 \pm 0.140$ | 842.7 |
| PEARL (fixed-offline adaptation) | $2.472 \pm 0.074$ | 1062.7 |

The paired difference is $-1.503 \pm 0.173$; an exploratory bootstrap 95% interval is $[-1.621, -1.383]$, and the exact two-sided sign test gives $p = 0.0156$. This supports an advantage under this fixed weak-corpus protocol, not reproduction of PEARL's original benchmark. The two meta-RL tables have different task/action spaces and must not be numerically pooled.

### D.4 Backbone and Mixture Robustness

This supplementary bandit study uses small randomly initialized GPT-2-, Llama-, and Qwen2-style decoders, comparing QTPT and SPT within each matched backbone rather than fine-tuning pretrained language models. Its independent seeds and training budget differ from the original bandit setting, so the absolute GPT-2 values should not be equated with Figure 2b.

The supplementary configuration uses 3 layers, width 128, 4 heads, dropout 0.1, 10,000 training trajectories, 1,000 validation trajectories, and 1,000 test tasks. Training uses 3,000 updates, batch size 64, AdamW with learning rate $3 \times 10^{-4}$ and weight decay $10^{-4}$. Validation losses are logged, but the table evaluates the fixed 3,000-update endpoint. Its target-network update is Polyak averaging with coefficient 0.01; this is a matched-backbone diagnostic, not an unchanged rerun of the main 8-layer setup.

QTPT has lower mean gap in all nine comparisons, with a large random-data margin and a narrower expert-data margin. Similar random-data SPT means are consistent with imitation of nearly uniform behavior, but numerical proximity alone does not prove exact convergence to a random policy. These tests cover three decoder choices, not all model scales.

Table 9: Cumulative gap at $T = 200$, mean $\pm$ standard deviation over five training seeds. Mix is expert-trajectory fraction. Lower is better.

| Backbone | Expert mix | QTPT gap ↓ | SPT gap ↓ |
|---|---|---|---|
| GPT-2 | 0.0 | $33.52 \pm 3.85$ | $222.20 \pm 2.35$ |
| GPT-2 | 0.5 | $34.42 \pm 2.10$ | $133.41 \pm 13.97$ |
| GPT-2 | 1.0 | $35.53 \pm 3.19$ | $46.92 \pm 1.36$ |
| Llama | 0.0 | $33.18 \pm 1.44$ | $221.93 \pm 2.19$ |
| Llama | 0.5 | $40.59 \pm 2.49$ | $141.29 \pm 11.15$ |
| Llama | 1.0 | $38.11 \pm 2.07$ | $39.88 \pm 0.93$ |
| Qwen2 | 0.0 | $33.88 \pm 1.34$ | $222.09 \pm 1.94$ |
| Qwen2 | 0.5 | $39.35 \pm 0.70$ | $141.08 \pm 11.75$ |
| Qwen2 | 1.0 | $36.54 \pm 2.17$ | $39.98 \pm 1.25$ |

Table 10: Values used for Figure 2b; mean $\pm$ standard deviation over five training seeds per mixture.

| Expert mix | QTPT gap ↓ | SPT gap ↓ |
|---|---|---|
| 0.00 | $31.97 \pm 1.45$ | $220.85 \pm 1.24$ |
| 0.10 | $31.58 \pm 0.41$ | $210.40 \pm 1.24$ |
| 0.20 | $32.57 \pm 0.66$ | $196.89 \pm 11.14$ |
| 0.25 | $33.92 \pm 1.25$ | $173.79 \pm 9.33$ |
| 0.50 | $36.24 \pm 1.57$ | $139.98 \pm 11.59$ |
| 0.75 | $36.43 \pm 1.75$ | $78.30 \pm 2.56$ |
| 0.80 | $35.64 \pm 0.90$ | $77.94 \pm 6.66$ |
| 0.90 | $37.93 \pm 2.44$ | $59.95 \pm 11.76$ |
| 1.00 | $38.14 \pm 2.02$ | $46.32 \pm 1.83$ |

### D.5 Task-Coherent Context and Extra Episodes

These synthetic diagnostics isolate context information rather than rerun the full sequential Bellman benchmark. A hidden goal $g \in [-0.9, 0.9]$ determines reward $r(a, g) = 1 - (a - g)^2 + \epsilon$, with noise standard deviation 0.05 and eleven grid actions in $[-1, 1]$. Behavior selects the nearest-goal action with probability 0.55 and a uniform random action otherwise. Each query has two observed transitions; support-augmented inputs add 24 transitions. The true goal is visible only in the oracle condition.

Both studies use five training seeds, 4,000 updates, batch size 256, learning rate $3 \times 10^{-4}$, and a two-layer, width-96, four-head Transformer. The objective regresses queried-action reward; at this terminal query the value equals expected immediate reward. All context observations precede the query, but the diagnostic encoder uses bidirectional attention over that observed context, not the main causal decoder. Training resamples tasks. Every 500 updates, checkpoints are evaluated on 512 goal-grid points with eight context resamples, selecting minimum diagnostic regret. These exploratory results use the same evaluation for selection and reporting, not a separate final hold-out.

Table 11: Diagnostic A: extra-context source, five-seed mean $\pm$ standard deviation. Regret is true reward loss against the best grid action; success is the fraction choosing it.

| Input protocol | Regret ↓ | Success ↑ | Action flip rate |
|---|---|---|---|
| Current-episode history | $0.0033 \pm 0.0003$ | $0.8808 \pm 0.0097$ | $0.2036 \pm 0.0120$ |
| History + unrelated-task support | $0.0037 \pm 0.0002$ | $0.8694 \pm 0.0096$ | $0.2220 \pm 0.0074$ |
| History + same-task support | $0.0002 \pm 0.0000$ | $0.9638 \pm 0.0069$ | $0.0475 \pm 0.0099$ |

Same-task support improves the diagnostic, whereas unrelated support does not. Removing history or replacing it with equally long unrelated history loses task information. Action flip rate measures sensitivity to context resampling; a constant bad policy also has zero flips, so it is not unconditionally lower-is-better. The results support the task-coherence explanation, not a theorem that arbitrary extra context must hurt or a long-horizon policy advantage over every baseline.

Table 12: Diagnostic B: task relevance versus token count, five-seed mean ± standard deviation. Displayed 0.0000 values are rounded.

| Input protocol | Regret ↓ | Success ↑ | Action flip rate |
|---|---|---|---|
| No history | $0.2386 \pm 0.0000$ | $0.1172 \pm 0.0000$ | $0.0000 \pm 0.0000$ |
| Current-episode history | $0.0033 \pm 0.0003$ | $0.8847 \pm 0.0100$ | $0.1978 \pm 0.0203$ |
| Unrelated history (same length) | $0.2386 \pm 0.0000$ | $0.1172 \pm 0.0000$ | $0.0000 \pm 0.0000$ |
| History + same-task support | $0.0002 \pm 0.0001$ | $0.9536 \pm 0.0112$ | $0.0502 \pm 0.0052$ |
| Oracle goal token | $0.0001 \pm 0.0001$ | $0.9547 \pm 0.0110$ | $0.0000 \pm 0.0000$ |

**Reporting conventions.** Each supplementary table specifies seed count and uncertainty unit. Seed standard deviations, episode confidence intervals, paired seed tests, and conditional checkpoint aggregates are distinct. Different selection rules and information protocols are not pooled.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state the main theoretical and empirical claims, including the scope of the D4RL extension and controlled ablations.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: The conclusion discusses remaining limitations and future work, including broader benchmarks, alternative Bellman backup designs, OOD overestimation, and distribution shift.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes].

Justification: The theoretical assumptions and statements are provided in the analysis section, with additional derivations and proofs included in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

   Justification: The paper describes the benchmark tasks, data-generation protocols, model architecture, optimizer, and major hyperparameters in the main text and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes].

Justification: The code is available at `https://github.com/lyc51758/qtpt-finalcode`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes].

Justification: The appendix specifies the GPT-2-style Transformer configuration, optimizer, learning-rate schedule, batch size, training epochs, and task-specific experimental details.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes].

Justification: Error bars and variability are reported where specified. Darkroom Table 1 reports episode-level confidence intervals. D4RL Table 2 reports mean and standard deviation.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No].

   Justification: The paper provides model and optimization details, but it does not yet fully specify hardware type, memory, runtime, or total compute for each experiment.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The work uses standard reinforcement-learning and reasoning benchmarks and does not involve human subjects, private data, or deceptive data collection.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No].

    Justification: The paper includes a broader-impact statement, but it does not provide a detailed separate discussion of both positive and negative societal impacts because the work is primarily foundational.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not release high-risk pretrained generative models, scraped datasets, or other assets requiring special misuse safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No].

    Justification: The paper cites the existing benchmarks and models used in the experiments, but it does not explicitly list all license terms in the manuscript.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A].

    Justification: The paper proposes a method and does not introduce a new dataset or benchmark asset.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The work does not involve crowdsourcing or human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The work does not involve human-subject research, so IRB approval or equivalent review is not applicable.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes].

    Justification: The appendix describes the GSM8K extension using Qwen-2.5-1.5B-Instruct with LoRA; this is an experimental extension rather than the core theoretical method.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.