Enhancing Offline Reinforcement Learning with Curriculum Learning-Based Trajectory Valuation

Amir Abolfazli
L3S Research Center
Hannover, Germany
abolfazli@l3s.de

Zekun Song
Technical University of Berlin
Berlin, Germany
zekun.song@tu-berlin.de

Avishek Anand
Delft University of Technology
Delft, Netherlands
avishek.anand@tudelft.nl

Wolfgang Nejdl
L3S Research Center
Hannover, Germany
nejdl@l3s.de

ABSTRACT

The success of deep reinforcement learning (DRL) relies on the availability and quality of training data, often requiring extensive interactions with specific environments. In many real-world scenarios, where data collection is costly and risky, offline reinforcement learning (RL) offers a solution by utilizing data collected by domain experts and searching for a batch-constrained optimal policy. This approach is further augmented by incorporating external data sources, expanding the range and diversity of data collection possibilities. However, existing offline RL methods often struggle with challenges posed by non-matching data from these external sources. In this work, we specifically address the problem of source-target domain mismatch in scenarios involving mixed datasets, characterized by a predominance of source data generated from random or suboptimal policies and a limited amount of target data generated from higher-quality policies. To tackle this problem, we introduce Transition Scoring (TS), a novel method that assigns scores to transitions based on their similarity to the target domain, and propose Curriculum Learning-Based Trajectory Valuation (CLTV), which effectively leverages these transition scores to identify and prioritize high-quality trajectories through a curriculum learning approach. Our extensive experiments across various offline RL methods and MuJoCo environments, complemented by rigorous theoretical analysis, demonstrate that CLTV enhances the overall performance and transferability of policies learned by offline RL algorithms.

KEYWORDS

Offline Reinforcement Learning, Trajectory Valuation

ACM Reference Format:
Amir Abolfazli, Zekun Song, Avishek Anand, and Wolfgang Nejdl. 2025. Enhancing Offline Reinforcement Learning with Curriculum Learning-Based Trajectory Valuation. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

This work is licensed under a Creative Commons Attribution International 4.0 License.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

1 INTRODUCTION

Offline Reinforcement Learning (RL) is a class of RL methods that requires the agent to learn from a dataset of pre-collected experiences without further environment interaction [25]. This learning paradigm decouples exploration from exploitation, rendering it particularly advantageous in scenarios where the process of data collection is costly, time-consuming, or risky [10, 18].

By utilizing pre-collected datasets, offline RL can bypass the technical challenges that are associated with online data collection, and has potential benefits for a number of real environments, such as human-robot collaboration and autonomous systems [2, 39].

However, this task is challenging, as offline RL methods suffer from the *extrapolation error* [10, 23]. This issue arises when offline deep RL methods are trained under one distribution but evaluated on a different one. More specifically, value functions implemented by a function approximator have a tendency to predict unrealistic values for unseen state-action pairs for standard off-policy deep RL algorithms such as BCQ [10], TD3+BC [9], CQL [24] and IQL [22]. This highlights the necessity for approaches that restrict the action space, forcing the agent to learn a behavior that is closely aligned with on-policy with respect to a subset of the source data [10].

In recent years, there have been a number of efforts within the paradigm of supervised learning for overcoming the *source-target domain mismatch problem* valuating data, including *data Shapley* [12] and *data valuation using reinforcement learning* (DVRL) [46]. Such methods have shown promising results on several application scenarios such as robust learning and domain adaptation [46].

Despite the success of such methods in the supervised learning setting, adapting them to the offline reinforcement learning (RL) setting presents several challenges. One major issue is the non-i.i.d. nature of the data. In supervised learning, data samples are typically assumed to be independent and identically distributed (i.i.d.), but this assumption is violated in offline RL since the data is generated by an agent interacting with an environment [26]. The presence of correlated and non-i.i.d. data samples complicates the valuation of these transitions and hinders generalization to the target domain. Additionally, the data distribution in offline RL often changes due to the agent's evolving policy or the environment's dynamics, leading to *distributional shifts* that aggravate the valuation of transitions, as their value may fluctuate with changing dynamics [23, 24].

Furthermore, unlike supervised learning, where the objective is to optimize a loss function given input-output pairs, the objective

in RL is to maximize cumulative rewards by learning a policy that maps states to actions [38]. Therefore, designing a reward function that accounts for distributional similarities between transition items from different domains is essential for effective policy learning.

These challenges highlight the complexities involved in adapting supervised learning methods to the offline RL setting and underscore the need for novel approaches to handle these challenges.

However, a recent work [3] introduced CUORL, a curriculum learning-based approach aimed at enhancing the performance of offline RL methods by strategically selecting valuable transition items. The limitation of CUORL is that only the current policy is considered to valuate trajectories, which lacks consideration of the information in the target dataset, making it challenging for the agent to adapt to different environments.

Another recent work [15] introduced Harness, which addresses a major challenge in offline RL involving mixed datasets, where the prevalence of low-return trajectories can limit the effectiveness of advanced algorithms, preventing them from fully exploiting high-return trajectories. Harness tackles this problem by re-weighting the dataset sampling process to create an artificial dataset that results in a behavior policy with a higher overall return, enabling RL algorithms to better utilize high-performing trajectories. By optimizing the sampling strategy, Harness enhances the performance of offline RL algorithms in environments characterized by mixed-return datasets. However, by re-weighting the dataset sampling process, Harness introduces a bias in favor of high-return trajectories, potentially neglecting important and high-quality transition items from low-return trajectories that could enhance the robustness of the policy.

In this work, we introduce a transition scoring (TS) method, which assigns a score to each transition in a trajectory, and a curriculum learning-based trajectory valuation method (CLTV) that operates based on the scores computed by TS to enable the agent to identify and select the most promising trajectories. Unlike existing methods, our approach leverages high-quality transitions from different trajectories generated by various policies.

Our results, using existing methods (CUORL and Harness) and two base offline RL algorithms (CQL and IQL) in four MuJoCo environments, show that our CLTV method improves the performance and transferability of offline RL policies. We also provide a theoretical analysis to demonstrate the efficacy of our approach.

2 BACKGROUND

Reinforcement Learning. The RL problem is typically modeled by a Markov decision process (MDP), formulated as a tuple $(\mathcal{X}, \mathcal{U}, p, r, \gamma)$, with a state space $\mathcal{X}$, an action space $\mathcal{U}$, and transition dynamics p.

At each discrete time step, the agent performs an action $u \in \mathcal{U}$ in a state $x \in \mathcal{X}$, transitions to a new state $x' \in \mathcal{X}$ based on the transition dynamics $p(x' \mid x, u)$, and receives a reward $r(x, u, x')$. The agent's goal is to maximize the expectation of the sum of discounted rewards, also known as the return $R_t = \sum_{i=t+1}^{\infty} \gamma^i r(x_i, u_i, x_{i+1})$, which weights future rewards with respect to the discount factor $\gamma \in [0, 1)$, determining the effective horizon. The agent makes decisions based on the policy $\pi : \mathcal{X} \to \mathcal{P}(\mathcal{U})$, which maps a given state x to a probability distribution over the action space $\mathcal{U}$. For a given policy π, the value function is defined as the expected return of an agent starting from state x, performing action u, and following the policy $Q^\pi(x, u) = \mathbb{E}_\pi[R_t \mid x, u]$. The state-action value function can be computed through the Bellman equation for the Q function:

$$Q^\pi(x, u) = \mathbb{E}_{s' \sim p}\left[r\left(x, u, x'\right) + \gamma \mathbb{E}_{u' \sim \pi} Q^\pi(x', u')\right]. \quad (1)$$

Given Q^π, the optimal policy $\pi^* = \max_u Q^*(x, u)$, can be obtained by greedy selection over the optimal value function $Q^*(x, u) = \max_\pi Q^\pi(x, u)$. For environments confronting agents with the curse of dimensionality, the value can be estimated with a differentiable function approximator $Q_\theta(x, u)$, with parameters θ. **Offline Reinforcement Learning.** Standard off-policy deep RL algorithms such as deep Q-learning (DQN) [30] and deep deterministic policy gradient (DDPG) [27] are applicable in batch RL as they are based on more fundamental batch RL algorithms [8].

However, they suffer from a phenomenon, known as *extrapolation error*, which occurs when there is a mismatch between the given fixed batch of data and true state-action visitation of the current policy [10]. This is problematic as incorrect values of state-action pairs, not contained in the batch, are propagated through temporal difference updates of most off-policy algorithms [37], resulting in poor performance of the model [40].

In offline RL, the goal is to learn a policy based on a previously collected dataset of transition items, optimizing decision-making without any additional interaction with the environment.

3 PROBLEM DESCRIPTION

We assume the availability of a source dataset $\mathcal{D}_\mathcal{S} = \{(x_i^\mathcal{S}, u_i^\mathcal{S}, x_i'^\mathcal{S}, r_i^\mathcal{S})\}_{i=1}^N \sim \mathcal{P}_\mathcal{S}$ and a target dataset $\mathcal{D}_\mathcal{T} = \{(x_i^\mathcal{T}, u_i^\mathcal{T}, x_i'^\mathcal{T}, r_i^\mathcal{T})\}_{i=1}^M \sim \mathcal{P}_\mathcal{T}$, where $x \in \mathbb{R}^m$ is a state; $u \in \mathbb{R}^n$ is the action that the agent performs at the state x; $r \in \mathbb{R}$ is the reward that the agent gets by performing the action u in the state x; and $x' \in \mathbb{R}^m$ is the state that the agent transitions to (i.e., next state). We also assume that the target dataset $\mathcal{D}_\mathcal{T}$ is much smaller than the the source dataset $\mathcal{D}_\mathcal{S}$, therefore $N \gg M$. Furthermore, the source distribution $\mathcal{P}_\mathcal{S}$ can be different from the target distribution $\mathcal{P}_\mathcal{T}$ (i.e. $\mathcal{P}_\mathcal{S} \neq \mathcal{P}_\mathcal{T}$), confronting the agent with the source-target domain mismatch problem.

Moreover, while intrinsic reward functions may be the same across source and target domains (i.e., $r^\mathcal{S} = r^\mathcal{T}$), modifications to the reward functions in either domain can lead to effective differences (i.e., $r^\mathcal{S} \neq r^\mathcal{T}$). We assume that the reward functions can potentially be different across the domains. These potential differences in reward functions could exacerbate the domain mismatch and introduce additional challenges for effective knowledge transfer. Each trajectory $\tau_k^\mathcal{S}$ in $\mathcal{D}_\mathcal{S}$ is defined as a sequence of transitions $\tau_k^\mathcal{S} = \{(x_i^{k,\mathcal{S}}, u_i^{k,\mathcal{S}}, x_i'^{k,\mathcal{S}}, r_i^{k,\mathcal{S}})\}_{i=1}^{L_k^\mathcal{S}}$, where $L_k^\mathcal{S}$ denotes the length of k-th trajectory. The degree of similarity between these transitions and those in the target dataset $\mathcal{D}_\mathcal{T}$ varies.

The goal is to identify high-quality trajectories by quantifying the similarity of each trajectory $\tau_k^\mathcal{S}$ from the source dataset to the target dataset $\mathcal{D}_\mathcal{T}$, aggregating the similarities of the individual transitions within $\tau_k^\mathcal{S}$ to those in $\mathcal{D}_\mathcal{T}$, and selecting trajectories most relevant for effective knowledge transfer.

4 RELATED WORK

Offline RL. The RL literature contains numerous techniques for dealing with the source-target domain mismatch problem. DARLA [14], a zero-shot transfer learning method that learns disentangled representations that are robust against domain shifts; and SAFER [35], which accelerates policy learning on complex control tasks by considering safety constraints. Meanwhile, the literature on off-policy RL includes principled experience replay memory sampling techniques. Prioritized Experience Replay (PER) [33] (e.g., [16, 17, 19]) attempts to sample transitions that contribute the most toward learning. However, most of the work to date on offline RL is focused on preventing the training policy from being too disjoint with the behavior policy [10, 20, 24].

Batch-Constrained deep Q-learning (BCQ) [10] is an offline RL method for continuous control, restricting the action space, thereby eliminating actions that are unlikely to be selected by the behavior policy and therefore rarely observed in the batch.

Conservative Q-Learning (CQL) [24] prevents the training policy from overestimating the Q-values by utilizing a penalized empirical RL objective. More precisely, CQL optimizes the value function not only to minimize the temporal difference error based on the interactions seen in the dataset but also minimizes the value of actions that the currently trained policy takes, while at the same time maximizing the value of actions taken by the behavior policy during data generation.

To increase the generalization capability of offline RL methods, [22] propose in-sample Q-learning (IQL), approximating the policy improvement step by considering the state value function as a random variable with some randomness determined by the action, and then taking a state-conditional expectile of this random variable to estimate the value of the best actions in that state. [15] introduce Harness, which addresses a challenge in offline RL involving mixed datasets. The prevalence of low-return trajectories in these datasets can limit the effectiveness of algorithms, preventing them from fully exploiting high-return trajectories. Harness tackles this problem by re-weighting the dataset sampling process to create an artificial dataset that results in a behavior policy with a higher return. This enables RL algorithms to better utilize high-performing trajectories.

Curriculum Learning in RL. In reinforcement learning, curriculum learning (CL) sequences tasks in a strategic manner, enhancing agent performance and accelerating training, especially for complex challenges [28, 31, 32, 43]. CL has shown promising performance in real-world applications [5, 29, 45], helping agents solve complex problems and transfer knowledge across different tasks [21, 31].

Among the works on curriculum learning and transition valuation in RL, the most relevant to ours is CUORL [3]. CUORL enhances offline RL methods by training agents on pertinent trajectories but struggles with non-deterministic policy properties. Additionally, CUORL only considers the current policy to evaluate trajectories, neglecting information from the target dataset, which hinders the agent's adaptability to different environments.

Our method addresses these issues by using KL divergence to capture non-deterministic properties and by identifying and selecting high-quality transition items from different trajectories.

5 METHODS

Our proposed method includes two key components: Transition Scoring (TS) and Curriculum Learning-Based Trajectory Valuation (CLTV). TS assigns scores to transitions based on their relevance to the target domain, helping to identify high-quality transitions.

CLTV leverages TS scores to prioritize high-quality trajectories through curriculum learning, guiding the agent's development in a structured manner. These components are discussed below.

Transition Scoring (TS). Inspired by DVRL [46], we adopt the REINFORCE algorithm [44] and use a DNN v_ϕ as the transition scoring method. The goal is to find the parameters ϕ^* of the DNN so that the network returns the optimal probability distribution over the set of all possible transitions.

The TS model $v_\phi : \mathcal{X} \times \mathcal{U} \times \mathcal{X}' \to [0, 1]$ is optimized to output scores corresponding to the similarity of transitions in the source dataset to the target domain. We formulate the corresponding optimization problem as:

$$\max_\phi J\left(\pi_\phi\right) = \mathbb{E}_{\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}\right) \sim P^{\mathcal{S}}}\left[r_\phi\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}, \Delta_\theta\right)\right], \tag{2}$$

where

$$r_\phi\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}, \Delta_\theta\right) = 2 \cdot \frac{r'_\phi\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}, \Delta_\theta\right) - r'_{\min}}{r'_{\max} - r'_{\min}} - 1. \tag{3}$$

Equation 3 represents the normalized reward, keeping the rewards in the range $[-1, 1]$, where $r'_\phi\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}, \Delta_\theta\right)$ is the unnormalized reward and defined as follows:

$$\begin{aligned} r'_\phi\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}, \Delta_\theta\right) = \delta \cdot & \sum_{\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}\right) \in \mathcal{B}} v_\phi\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}\right) \cdot \Delta_\theta\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}\right) \\ & + (1-\delta) \cdot \sum_{\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}\right) \in \mathcal{B}} v_\phi\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}\right). \end{aligned} \tag{4}$$

In Equation 4, the term $\sum_{\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}\right) \in \mathcal{B}} v_\phi\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}\right)$ regularizes the unnormalized reward and prevents assigning low scores to the majority of transitions. Δ_θ represents the dynamics factor, quantified by the classifiers $q_{xu} = p_{\theta_{xu}}(y \mid x, u)$ and $q_{xux'} = p_{\theta_{xux'}}(y \mid x, u, x')$, where $y \in \{\text{source}, \text{target}\}$.

The optimization problem involves adjusting ϕ to maximize the expected reward by aligning transition dynamics with the target domain. The classifiers, parameterized by θ, remain fixed during the optimization of ϕ. These classifiers are implemented as fully connected multi-layer neural networks, similar to those used in DARC [6]. They measure the likelihood of transitions belonging to either the source or target domains. By applying Bayes' rule, the classifier probabilities are related to transition probabilities, enabling the expression of posterior probabilities in terms of likelihoods and prior probabilities.

The classifier q_{xu} is defined as follows:

$$q_{xu} = p_{\theta_{xu}}\left(y \mid x^{\mathcal{S}}, u^{\mathcal{S}}\right) = \frac{p\left(x^{\mathcal{S}}, u^{\mathcal{S}} \mid y\right) p(y)}{p\left(x^{\mathcal{S}}, u^{\mathcal{S}}\right)}. \tag{5}$$

Similarly, the classifier $q_{xux'}$ is defined as follows:

$$q_{xux'} = p_{\theta_{xux'}}\left(y \mid x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}\right) = \frac{p\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}} \mid y\right) p(y)}{p\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}\right)}. \tag{6}$$

We train the classifiers $q_{xux'}$ and q_{xu} to minimize the standard cross-entropy loss. The loss function for the classifier q_{xu} is presented in Equation 7.

$$\begin{aligned}\mathcal{L}_{xu}\left(\theta_{xu}\right) = &-\mathbb{E}_{\mathcal{D}_{\text{target}}}\left[\log q_{\theta_{xu}}\left(\text{target} \mid x^{\mathcal{S}}, u^{\mathcal{S}}\right)\right] \\ &-\mathbb{E}_{\mathcal{D}_{\text{source}}}\left[\log q_{\theta_{xu}}\left(\text{source} \mid x^{\mathcal{S}}, u^{\mathcal{S}}\right)\right].\end{aligned} \tag{7}$$

Similarly, the loss function for the classifier $q_{xux'}$ is presented in Equation 8.

$$\begin{aligned}\mathcal{L}_{xux'}\left(\theta_{xux'}\right) = &-\mathbb{E}_{\mathcal{D}_{\text{target}}}\left[\log q_{\theta_{xux'}}\left(\text{target} \mid x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}\right)\right] \\ &-\mathbb{E}_{\mathcal{D}_{\text{source}}}\left[\log q_{\theta_{xux'}}\left(\text{source} \mid x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}\right)\right]\end{aligned} \tag{8}$$

The dynamics factor Δ (as defined in [6]) measures the discrepancy between the distribution of the source and target datasets and is defined as follows:

$$\begin{aligned}\Delta_\theta\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}\right) = &\log q_{\theta_{xux'}}\left(\text{target} \mid x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}\right) \\ &-\log q_{\theta_{xu}}\left(\text{target} \mid x^{\mathcal{S}}, u^{\mathcal{S}}\right) \\ &-\log q_{\theta_{xux'}}\left(\text{source} \mid x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}\right) \\ &+\log q_{\theta_{xu}}\left(\text{source} \mid x^{\mathcal{S}}, u^{\mathcal{S}}\right).\end{aligned} \tag{9}$$

Optimizing Δ enables the identification of transitions with lower distribution discrepancies. w is the sum of selection probabilities, reflecting the likelihood of each transition to be selected. This factor prevents the TS model from selecting only a limited number of transitions. The parameters δ and $1-\delta$ serve as balancing factors for $v_\phi\Delta$ and v_ϕ, respectively, in the calculation of r'_ϕ.

For training the TS, all transition items in the source buffer $\mathcal{D}_{\mathcal{S}}$ are divided into batches. Each batch $D_B^{\mathcal{S}} = \left(t_j\right)_{j=1}^{B_s} \sim \mathcal{D}^{\mathcal{S}}$ is provided as input to the TS, with shared parameters across the batch. Let $w_j = v_\phi\left(x_j^{\mathcal{S}}, u_j^{\mathcal{S}}, x_j'^{\mathcal{S}}\right)$ denote the probability that transition item j from the source buffer is selected for training the offline RL model. The sampling step $x_j \sim \text{Bernoulli}(w_j)$ is then used to choose transition items based on their importance scores w_j, following the approach in [46].

This method prioritizes high-quality transitions by sampling them with higher probability, effectively focusing on those that contribute most to the performance of the RL agent.

Our adopted version of the REINFORCE algorithm has the following objective function for the policy π_ϕ:

$$\begin{aligned}J\left(\pi_\phi\right) &= \mathbb{E}_{\substack{(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}) \sim P^{\mathcal{S}} \\ w \sim \pi_\phi(\mathcal{D}_{\mathcal{S}}, \cdot)}}\left[r_\phi\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}, \Delta_\theta\right)\right] \\ &= \int P^{\mathcal{S}}\left((x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}})\right) \sum_{w \in [0,1]^N} \pi_\phi(\mathcal{D}_{\mathcal{S}}, w) \\ &\quad \cdot \left[r_\phi\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}, \Delta_\theta\right)\right] d\left((x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}})\right).\end{aligned} \tag{10}$$

In the above equation, $\pi_\phi(\mathcal{D}_{\mathcal{S}}, w)$ represents the probability of the selection probability vector w occurring. The policy uses the scores output by the TS. This contrasts with DVRL [46], which employs a binary selection vector $\mathbf{s} = \left(s_1, \ldots, s_{B_s}\right)$, where s_{B_s} denotes the batch size, $s_i \in \{0, 1\}$, and $P\left(s_i = 1\right) = w_i$. Thus, in our training, the TS does not control exploration, but instead provides scores for the transition items and is tuned accordingly.

We calculate the gradient of the above objective function (Equation 10) as follows:

$$\begin{aligned}\nabla_\phi J\left(\pi_\phi\right) = \mathbb{E}_{\substack{(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}) \sim P^{\mathcal{S}} \\ w \sim \pi_\phi(\mathcal{D}_{\mathcal{S}}, \cdot)}}\Bigg[&r_\phi\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}, \Delta_\theta\right) \cdot \nabla_\phi \log \pi_\phi(\mathcal{D}_{\mathcal{S}}, w) \\ &+ \nabla_\phi r_\phi\left(x^{\mathcal{S}}, u^{\mathcal{S}}, x'^{\mathcal{S}}, \Delta_\theta\right)\Bigg].\end{aligned} \tag{11}$$

Then, the parameters ϕ of $J\left(\pi_\phi\right)$ are updated using the calculated gradient $\nabla_\phi J\left(\pi_\phi\right)$ as follows:

$$\phi \leftarrow \phi + k\left(r_\phi \cdot \nabla_\phi \log \pi_\phi(D_B^{\mathcal{S}}, (x_1, \ldots, x_{B_{\mathcal{S}}})) + \nabla_\phi r_\phi\right), \tag{12}$$

where k is the learning rate and r_ϕ denotes the normalized reward.

The complete derivation of Equation 11 is provided in Appendix A. Our transition scoring approach is presented in Algorithm 1.

Algorithm 1 Transition Scoring (TS)

Input: Source dataset $\mathcal{D}_{\mathcal{S}}$, target dataset $\mathcal{D}_{\mathcal{T}}$;
classifiers q_{xu} and $q_{xux'}$;
ratio of transition similarity to reward δ;
learning rate k
Output: TS model v_ϕ
Train TS $v_\phi(t_j)$ with transition item $t_j = (x_j, u_j, x_j')$
while not converged **do**
  Sample a mini-batch: $D_B^{\mathcal{S}} = (t_j)_{j=1}^{B_{\mathcal{S}}} \sim \mathcal{D}_{\mathcal{S}}$
  Initialize weighted sum $\mathcal{V}_\phi = 0$
  Initialize selection probability sum $\mathcal{W}_\phi = 0$
  for $j = 1, \ldots, B_{\mathcal{S}}$ **do**
    $w_j = v_\phi(t_j)$
    $x_j \sim \text{Bernoulli}(w_j)$
    $\Delta_j = \log \frac{q_{xux'}(\text{target}|x_j,u_j,x_j')}{q_{xux'}(\text{source}|x_j,u_j,x_j')} - \log \frac{q_{xu}(\text{target}|x_j,u_j)}{q_{xu}(\text{source}|x_j,u_j)}$
    $\mathcal{V}_\phi \leftarrow \mathcal{V}_\phi + w_j \cdot \Delta_j$
    $\mathcal{W}_\phi \leftarrow \mathcal{W}_\phi + w_j$
  $r'_\phi = \delta \cdot \mathcal{V}_\phi + (1-\delta) \cdot \mathcal{W}_\phi$
  $r_\phi = 2 \cdot \frac{r'_\phi - r'_{\min}}{r'_{\max} - r'_{\min}} - 1$
  $\phi \leftarrow \phi + k \cdot r_\phi \cdot \nabla_\phi \log \pi_\phi(D_B^{\mathcal{S}}, (x_1, \ldots, x_{B_{\mathcal{S}}}))$

Curriculum Learning-based Trajectory Valuation (CLTV). Building upon the transition scoring (TS) mechanism, CLTV focuses on enhancing policy learning by prioritizing valuable trajectories. CLTV employs curriculum learning to present the agent with the most valuable trajectories at different stages of learning.

CLTV uses two distinct sets of actor-critic networks: one for the source domain $\left(Q_{\omega_1}^{\mathcal{S}}, \pi_{\theta_1}^{\mathcal{S}}\right)$ and another for the target domain $\left(Q_{\omega_2}^{\mathcal{T}}, \pi_{\theta_2}^{\mathcal{T}}\right)$. By maintaining separate networks, CLTV tailors the learning process to the unique dynamics and reward structures of each domain, facilitating effective policy transfer.

CLTV integrates the TS method by training classifiers $q_{xux'}$ and q_{xu} to adjust the rewards, helping to address the distributional shift between the source and target domains in offline RL.

Each trajectory $\tau_k^{\mathcal{S}}$ in the source dataset $\mathcal{D}_{\mathcal{S}}$ is defined as a sequence of transitions $\tau_k^{\mathcal{S}} = \left(x_i^{k,\mathcal{S}}, u_i^{k,\mathcal{S}}, x_i'^{,k,\mathcal{S}}, r_i^{k,\mathcal{S}}\right)_{i=1}^{L_k^{\mathcal{S}}}$, where $L_k^{\mathcal{S}}$ denotes the length of the k-th trajectory.

The degree of similarity between these transitions in $\mathcal{D}_{\mathcal{S}}$ and those in the target dataset $\mathcal{D}_{\mathcal{T}}$ varies. When the source data does not perfectly align with the target environment, relying solely on the original rewards can cause the agent to learn policies that are suboptimal or misaligned with the target setting.

Moreover, not all transitions within a trajectory contribute equally to learning; only a subset of transitions contributes to improving the policy. By assigning a relevance score to each transition based on its similarity to the target domain dynamics, we effectively re-weight the rewards associated with these transitions.

This approach increases the influence of transitions that are more representative of the target environment, guiding the agent to focus on the most valuable experiences. As a result, the agent learns policies that are better aligned with the target domain, enhancing overall performance. This method is especially beneficial in scenarios with sparse or noisy reward signals, as it helps the model identify and leverage valuable experiences.

During the curriculum learning process, the value of the trajectory i in the source dataset is computed as follows:

$$v_i = \exp\left(-D_{\mathrm{KL}}(\pi_{\theta_2}^{\mathcal{T}}(u_t|x_t) \parallel \pi_{\theta_1}^{\mathcal{S}}(u_t|x_t))\right) \cdot \Sigma_{t=1}^{T} \gamma^{t-1} r_t. \tag{13}$$

This valuation formula uses the KL divergence between the target policy $\pi_{\theta_2}^{\mathcal{T}}$ and the policy we need to train $\pi_{\theta_1}^{\mathcal{S}}$, along with discounted rewards, to valuate trajectories. It includes two main components: the *similarity component* $\exp\left(-D_{\mathrm{KL}}(\pi_{\theta_2}^{\mathcal{T}}(u_t|x_t) \parallel \pi_{\theta_1}^{\mathcal{S}}(u_t|x_t))\right)$, which measures the similarity between policies, and the *return component* $\Sigma_{t=1}^{T} \gamma^{t-1} r_t$, representing the discounted sum of rewards.

This approach effectively guides the agent toward states with higher expected value by considering both policy similarity and trajectory returns, ensuring that the trajectories closely resemble those in the target dataset while leveraging information from both the source and target domains. After obtaining the trajectory values, we sort all trajectories in decreasing order based on these values.

We then sample the top m trajectories, where m is a hyperparameter that can be adjusted according to the quality of the dataset. Since these selected trajectories are the most valuable ones in terms of similarity to the target dataset, we merge them with the target dataset and use the combined data to update the critic and actor networks. Our CLTV approach is detailed in Algorithm 2.

To show the effectiveness of our approach, we first present Theorem 1, which provides a formal justification for TS by establishing bounds on the policy's performance in the target domain, and then present Theorem 2, which demonstrates how CLTV ensures value alignment with the target domain through policy improvement. Both theorems are discussed in detail in Section 5.1.

Algorithm 2 Curriculum Learning-Based Trajectory Valuation

Input: Critic networks $Q_{\omega_1}^{\mathcal{S}}$, $Q_{\omega_2}^{\mathcal{T}}$, actor networks $\pi_{\theta_1}^{\mathcal{S}}$, $\pi_{\theta_2}^{\mathcal{T}}$;
offline RL algorithm $\mathcal{A}$;
source dataset $\mathcal{D}^{\mathcal{S}}$, target dataset $\mathcal{D}^{\mathcal{T}}$;
epoch size E;
training steps S;
the ratio of transition score to transition reward λ
Output: Critic network $Q_{\omega_1}^{\mathcal{S}}$, actor network $\pi_{\theta_1}^{\mathcal{S}}$
Initialization:
Train classifiers q_{xu} and $q_{xux'}$
Train TS model $v_\phi(t_j)$
// Modify rewards in the source dataset
for $j = 1, \ldots, |\mathcal{D}^{\mathcal{S}}|$ **do**
  $w_j = v_\phi(t_j)$
  $r_j \leftarrow (1-\lambda) \cdot r_j + \lambda \cdot w_j$
// Curriculum learning loop
for $e = 1$ to E **do**
  $\mathcal{D} \leftarrow \mathcal{D}^{\mathcal{S}}$
  // Compute values for each trajectory in the source dataset
  for each trajectory $\tau_i = \{(x_j^i, u_j^i, r_j^i)\}_{j=1}^{T}$ in $\mathcal{D}$ **do**
    $v_i = \exp\left(-D_{\mathrm{KL}}(\pi_{\theta_2}^{\mathcal{T}}(u_t|x_t) \parallel \pi_{\theta_1}^{\mathcal{S}}(u_t|x_t))\right) \cdot \Sigma_{t=1}^{T} \gamma^{t-1} r_t$
  // Sort trajectories by v_i in decreasing order
  for $s = 1$ to S **do**
  // Sample m trajectories $\{\tau_i\}_{i=1}^{m}$ from $\mathcal{D}$
    $\mathcal{D}^{train} = \{\tau_i\}_{i=1}^{m} \cup \mathcal{D}^{\mathcal{T}}$
    Update ω_1, θ_1 with $\mathcal{A}(\mathcal{D}^{train}, Q_{\omega_1}^{\mathcal{S}}, \pi_{\theta_1}^{\mathcal{S}})$

5.1 Theoretical Analysis

In this section, we present the foundational theoretical aspects of our offline RL approach, focusing on the integration of behavior policies and the critical role of KL divergence in policy optimization.

We begin by applying several lemmas—namely, the *performance difference lemma* [1], the *state value estimation error* [25], the *total variance and KL divergence relation* [4], and the *total variance and L_1 norm relation* [42]—which set the foundation for our analysis.

Lemma 1. *Let π' and $\tilde{\pi}$ denote any two policies, and let $d_{\pi'}$ be the discounted state distribution induced by policy π' over the state space $\mathcal{X}$. The following inequality holds:*

$$\begin{aligned}(1-\gamma)\left(V^{\pi'} - V^{\tilde{\pi}}\right) &= \mathbb{E}_{x\sim d_{\pi'}, u\sim\pi'(\cdot|x)}\left[A_{\tilde{\pi}}(x,u)\right] \\ &\leq \frac{2}{1-\gamma}\mathbb{E}_{x\sim d_{\pi'}}\left\|\pi'(\cdot \mid x) - \tilde{\pi}(\cdot \mid x)\right\|_1 .\end{aligned} \tag{14}$$

Lemma 2. *Let V^{π} be the value function for any policy π, and let $\hat{V}^{\pi}$ be an estimate of V^{π}. Let r and $\hat{r}$ be the true reward and its estimate, respectively, both bounded in $[0,1]$. Similarly, let p and $\hat{p}$ be the true and estimated transition probabilities. Then, for any discount factor $\gamma \in [0,1)$, the following inequality holds:*

$$\left\|V^{\pi} - \hat{V}^{\pi}\right\|_\infty \leq \frac{1}{1-\gamma}\left(\|r-\hat{r}\|_\infty + \frac{\gamma}{1-\gamma}\|p-\hat{p}\|_\infty\right). \tag{15}$$

Lemma 3. *For any two probability distributions P and Q, their total variance and KL divergence are bounded as follows, according*

to Pinsker's inequality:

$$D_{\mathrm{TV}}(P, Q) \leq \sqrt{\frac{1}{2} D_{\mathrm{KL}}(P \parallel Q)}. \tag{16}$$

Lemma 4. *The total variance between any two probability distributions P and Q is equal to half the norm of the difference of P and Q:*

$$\frac{1}{2}\|P - Q\|_1 = D_{\mathrm{TV}}(P, Q). \tag{17}$$

Corollary 1. *Let $p^{\mathcal{S}}$ and $p^{\mathcal{T}}$ represent the transition probability distributions of the source and target domains, respectively, and let π be the policy we aim to train. The following inequality holds:*

$$\left\| V^{\pi}_{p^{\mathcal{S}}} - V^{\pi}_{p^{\mathcal{T}}} \right\|_{\infty} \leq \frac{\gamma}{(1-\gamma)^2} \sqrt{\frac{1}{2} D_{\mathrm{KL}}(p^{\mathcal{S}} \parallel p^{\mathcal{T}})}. \tag{18}$$

Proof. We begin by applying the result from Lemma 2, noting that the source and target domains share the same reward function. Thus, we have:

$$\left\| V^{\pi}_{p^{\mathcal{S}}} - V^{\pi}_{p^{\mathcal{T}}} \right\|_{\infty} \leq \frac{\gamma}{(1-\gamma)^2} \left\| p^{\mathcal{S}} - p^{\mathcal{T}} \right\|_{\infty}. \tag{19}$$

To further refine this bound, we apply Pinsker's inequality (Lemma 3) which provides:

$$\left\| p^{\mathcal{S}} - p^{\mathcal{T}} \right\|_{\infty} \leq \sqrt{\frac{1}{2} D_{\mathrm{KL}}(p^{\mathcal{S}} \parallel p^{\mathcal{T}})}. \tag{20}$$

Substituting the result from Pinsker's inequality into the inequality from Lemma 2, we obtain:

$$\left\| V^{\pi}_{p^{\mathcal{S}}} - V^{\pi}_{p^{\mathcal{T}}} \right\|_{\infty} \leq \frac{\gamma}{(1-\gamma)^2} \sqrt{\frac{1}{2} D_{\mathrm{KL}}(p^{\mathcal{S}} \parallel p^{\mathcal{T}})}. \tag{21}$$

□

Theorem 1. *Let $d(s)$ be a distribution over states. If the condition $\mathbb{E}_{s\sim d(s)}\left[\sqrt{\frac{1}{2} D_{\mathrm{KL}}(p^{\mathcal{S}} \parallel p^{\mathcal{T}})}\right] \leq \epsilon$ is satisfied, the expected value of the policy π in the target domain can be bounded below by:*

$$\mathbb{E}_{s\sim d(s)}\left[V^{\pi}_{p^{\mathcal{T}}}\right] \geq \mathbb{E}_{s\sim d(s)}\left[V^{\pi}_{p^{\mathcal{S}}}\right] - \frac{\gamma\epsilon}{(1-\gamma)^2}. \tag{22}$$

Proof. We begin by applying the expectation operator to Corollary 1, which provides:

$$\mathbb{E}_{s\sim d(s)}\left[\left\| V^{\pi}_{p^{\mathcal{S}}} - V^{\pi}_{p^{\mathcal{T}}} \right\|_{\infty}\right] \leq \tfrac{\gamma}{(1-\gamma)^2} \mathbb{E}_{s\sim d(s)}\left[\sqrt{\tfrac{1}{2} D_{\mathrm{KL}}(p^{\mathcal{S}} \parallel p^{\mathcal{T}})}\right]. \tag{23}$$

By the non-negativity of expectations, we have:

$$\mathbb{E}_{s\sim d(s)}\left[V^{\pi}_{p^{\mathcal{S}}} - V^{\pi}_{p^{\mathcal{T}}}\right] \leq \mathbb{E}_{s\sim d(s)}\left[\left\| V^{\pi}_{p^{\mathcal{S}}} - V^{\pi}_{p^{\mathcal{T}}} \right\|_{\infty}\right]. \tag{24}$$

Substituting the condition into Equation 23, we obtain:

$$\mathbb{E}_{s\sim d(s)}\left[V^{\pi}_{p^{\mathcal{S}}} - V^{\pi}_{p^{\mathcal{T}}}\right] \leq \frac{\gamma\epsilon}{(1-\gamma)^2}. \tag{25}$$

Consequently, this yields the lower bound:

$$\mathbb{E}_{s\sim d(s)}\left[V^{\pi}_{p^{\mathcal{T}}}\right] \geq \mathbb{E}_{s\sim d(s)}\left[V^{\pi}_{p^{\mathcal{S}}}\right] - \frac{\gamma\epsilon}{(1-\gamma)^2}. \tag{26}$$

□

Corollary 2. *Let π_b denote the behavior policy, which induces a distribution d_{π_b} over the state space $\mathcal{X}$. For the current policy π_i, and the policy in the next episode π_{i+1}, the following relationship is valid:*

$$\begin{aligned} &\mathbb{E}_{x\sim d_{\pi_b}, u\sim\pi_b(\cdot|x)}\left[A^{\pi_i}(x,u)\right] \\ &\quad - \frac{2}{1-\gamma}\mathbb{E}_{x\sim d_{\pi_b}} D_{\mathrm{TV}}\left(\pi_b(\cdot \mid x), \pi_{i+1}(\cdot \mid x)\right) \\ &\leq (1-\gamma)\left(V^{\pi_{i+1}} - V^{\pi_i}\right). \end{aligned} \tag{27}$$

Proof. By substituting π^* with π_b and $\tilde{\pi}$ with π_{i+1} in Equation 14, we obtain the first part of the inequality.

$$(1-\gamma)(V^{\pi_b} - V^{\pi_{i+1}}) \leq \tfrac{1}{1-\gamma}\mathbb{E}_{x\sim d_{\pi_b}} \|\pi_b(\cdot \mid x) - \pi_{i+1}(\cdot \mid x)\|_1 \tag{28}$$

Similarly, by substituting π^* with π_b and $\tilde{\pi}$ with π_i in Equation 14, we derive Equation 29, which represents the second part of the inequality in Equation 14.

$$(1-\gamma)(V^{\pi_b} - V^{\pi_i}) = \mathbb{E}_{x\sim d_{\pi_b}, u\sim\pi_b(\cdot|x)}\left[A^{\pi_i}(x,u)\right] \tag{29}$$

By applying Equation 28 and Equation 29, we derive:

$$\begin{aligned} (1-\gamma)(V^{\pi_{i+1}} - V^{\pi_i}) \geq\ &\mathbb{E}_{x\sim d_{\pi_b}, u\sim\pi_b(\cdot|x)}\left[A^{\pi_i}(x,u)\right] \\ &- \frac{1}{1-\gamma}\mathbb{E}_{x\sim d_{\pi_b}} \|\pi_b(\cdot \mid x) - \pi_{i+1}(\cdot \mid x)\|_1 \end{aligned} \tag{30}$$

Utilizing Lemma 4, we derive:

$$\begin{aligned} (1-\gamma)(V^{\pi_{i+1}} - V^{\pi_i}) \geq\ &\mathbb{E}_{x\sim d_{\pi_b}, u\sim\pi_b(\cdot|x)}\left[A^{\pi_i}(x,u)\right] \\ &- \frac{2}{1-\gamma}\mathbb{E}_{x\sim d_{\pi_b}} D_{\mathrm{TV}}\left(\pi_b(\cdot \mid x), \pi_{i+1}(\cdot \mid x)\right) \end{aligned} \tag{31}$$

□

Theorem 2. *The KL divergence between the behavior policy π_b and the policy in the subsequent episode π_{i+1} establishes a lower bound for policy improvement:*

$$\begin{aligned} &(1-\gamma)\left(V^{\pi_{i+1}} - V^{\pi_i}\right) \\ &\geq \mathbb{E}_{x\sim d_{\pi_b}, u\sim\pi_b(\cdot|x)}\left[A^{\pi_i}(x,u)\right] \\ &\quad - \frac{\sqrt{2}}{1-\gamma}\mathbb{E}_{x\sim d_{\pi_b}} \sqrt{D_{\mathrm{KL}}(\pi_b(\cdot \mid x) \parallel \pi_{i+1}(\cdot \mid x))}. \end{aligned} \tag{32}$$

Proof. In Corollary 2, we derived bounds for policy improvement by introducing the total variation distance. To relate total variation to KL divergence, we utilize Lemma 3, which gives us Pinsker's inequality:

$$D_{\mathrm{TV}}(P, Q) \leq \sqrt{\frac{1}{2} D_{\mathrm{KL}}(P \parallel Q)}. \tag{33}$$

Next, we substitute the total variation distance in Equation 31 with the KL divergence bound using Equation 33:

$$
\begin{aligned}
(1-\gamma)(V^{\pi_{i+1}} - V^{\pi_i}) &\geq \mathbb{E}_{x\sim d_{\pi_b}, u\sim\pi_b(\cdot|x)}\left[A^{\pi_i}(x,u)\right] \\
&\quad - \frac{2}{1-\gamma}\mathbb{E}_{x\sim d_{\pi_b}} D_{\mathrm{TV}}\left(\pi_b(\cdot \mid x), \pi_{i+1}(\cdot \mid x)\right) \\
&\geq \mathbb{E}_{x\sim d_{\pi_b}, u\sim\pi_b(\cdot|x)}\left[A^{\pi_i}(x,u)\right] \\
&\quad - \frac{2}{1-\gamma}\mathbb{E}_{x\sim d_{\pi_b}} \sqrt{\frac{1}{2} D_{\mathrm{KL}}(\pi_b(\cdot \mid x) \parallel \pi_{i+1}(\cdot \mid x))} \\
&= \mathbb{E}_{x\sim d_{\pi_b}, u\sim\pi_b(\cdot|x)}\left[A^{\pi_i}(x,u)\right] \\
&\quad - \frac{\sqrt{2}}{1-\gamma}\mathbb{E}_{x\sim d_{\pi_b}} \sqrt{D_{\mathrm{KL}}(\pi_b(\cdot \mid x) \parallel \pi_{i+1}(\cdot \mid x))}.
\end{aligned} \tag{34}
$$

□

6 EXPERIMENTS

Datasets. We used D4RL [7] datasets to create three mixed datasets for each of the considered MuJoCo [41] domains: Ant, HalfCheetah, Hopper, and Walker2d. These datasets were created by mixing 90% of transition items from the source datasets with 10% from the target datasets. In D4RL, the *medium* dataset is generated by first training a policy online using Soft Actor-Critic (SAC) [13], early-stopping the training, and collecting 1M transition items from this partially trained policy. The *random* datasets are generated by unrolling a randomly initialized policy in these domains. Similarly, the *expert* datasets are generated by an expert policy [7].

Baselines. We use CQL [24] and IQL [22] as our base offline RL algorithms as they are the most widely used offline RL algorithms and have demonstrated good results in tasks similar to ours, making them reliable benchmarks [11, 34, 36].

We consider Vanilla (the base algorithm without any additional methods applied), CUORL [3], and Harness [15] as our *baselines*, applied on top of the base offline RL algorithms (CQL and IQL).

Performance Metric. We measure the performance of offline RL algorithms using a normalized score, as introduced in D4RL [7]. The score is calculated as: $100 \times \frac{\text{score}-\text{random score}}{\text{expert score}-\text{random score}}$. This formula is used to normalize the scores for each domain, roughly scaling them to a range between 0 and 100. A normalized score of 0 corresponds to the average returns (over 100 episodes, with each episode containing 5,000 steps) of an agent that takes actions uniformly at random across the action space. A score of 100 corresponds to the average returns of a domain expert. Scores below 0 indicate performance worse than random, while scores above 100 indicate performance surpassing expert level.

Ablation Study. The choice of reward function and the impact of similarity-reward trade-off parameters of our method (λ and δ) are discussed in Appendix B.2 and Appendix B.3, respectively.

Implementation and Parameter Tuning. The implementation details and hyperparameter tuning are presented in Appendix C[1]. Our code is available at https://github.com/amir-abolfazli/CLTV.

[1]The Appendix is available at https://www.arxiv.org/abs/2502.00601.

6.1 Performance of Offline RL Methods

Table 1 provides a comprehensive evaluation of three methods (CUORL, Harness, and CLTV) alongside the Vanilla version, using two popular offline RL algorithms (CQL and IQL). The evaluation reports their performance, measured by normalized score, and standard deviations over 100 episodes and 5 seeds on mixed D4RL datasets. The highest performing scores are highlighted in blue, and the total scores of the second best performing method are marked in bold. Additionally, the percentage increase (PI) in score of CLTV, compared to the second best performing method, is reported. The runtime analysis is presented in Appendix B.1.

The results show that CLTV consistently outperforms the other methods (Vanilla, CUORL and Harness) across all domains and algorithms. For example, in the Ant domain using CQL, CLTV achieves a total score of 238.18, which is 95% higher than the second-best method (Harness) with a score of 120.97. Similarly, in the IQL variant for the Walker2d domain, CLTV surpasses the second-best method by 28%, scoring 268.62 against 208.80. Across the HalfCheetah and Hopper domains, CLTV demonstrates significant improvements, with percentage increases of 43% and 62% respectively, underscoring its superior performance as reflected by the relatively low standard deviations. Moreover, CLTV often highlights the highest individual dataset scores, indicating its robustness across different datasets within each domain.

Table 1: Average normalized score along with standard deviation over 100 episodes and 5 seeds on mixed D4RL datasets.

Domain	RL Algorithm	Method	Dataset random-medium	Dataset random-expert	Dataset medium-expert	Total	PI
Ant	CQL	Vanilla	51.99 ± 22.32	49.81 ± 25.85	-6.25 ± 14.12	95.55	
		CUORL	-8.37 ± 2.59	-7.81 ± 4.61	6.32 ± 23.51	-9.86	
		Harness	67.19 ± 6.47	38.30 ± 38.32	15.48 ± 22.20	**120.97**	
		CLTV	97.48 ± 3.63	115.86 ± 6.82	24.84 ± 13.70	238.18	↑ 95%
	IQL	Vanilla	83.75 ± 7.18	82.96 ± 9.07	117.20 ± 4.39	**283.91**	
		CUORL	4.60 ± 1.18	4.66 ± 0.49	116.23 ± 2.50	125.49	
		Harness	74.45 ± 6.52	82.56 ± 8.26	110.57 ± 2.56	267.58	
		CLTV	78.67 ± 8.26	88.26 ± 4.67	117.00 ± 7.28	283.93	↑ 0%
HalfCheetah	CQL	Vanilla	35.15 ± 2.78	-0.09 ± 0.75	29.17 ± 18.12	64.23	
		CUORL	-3.42 ± 0.46	-3.47 ± 0.30	36.89 ± 16.87	30.00	
		Harness	34.79 ± 4.12	0.53 ± 1.41	44.19 ± 10.86	**79.51**	
		CLTV	44.13 ± 3.47	10.37 ± 2.51	59.88 ± 7.91	114.38	↑ 43%
	IQL	Vanilla	37.78 ± 1.51	6.97 ± 2.33	58.05 ± 3.48	**102.80**	
		CUORL	2.97 ± 1.50	2.55 ± 1.40	60.53 ± 4.25	66.05	
		Harness	36.55 ± 2.78	8.37 ± 2.80	55.90 ± 3.22	100.82	
		CLTV	41.83 ± 0.63	16.28 ± 7.22	77.10 ± 5.16	135.21	↑ 31%
Hopper	CQL	Vanilla	18.93 ± 12.57	22.47 ± 23.90	75.90 ± 11.12	**117.30**	
		CUORL	0.83 ± 0.23	0.73 ± 0.15	71.36 ± 40.45	72.90	
		Harness	12.90 ± 14.26	22.63 ± 7.59	65.33 ± 29.68	100.86	
		CLTV	51.04 ± 3.21	56.23 ± 15.14	83.44 ± 16.95	190.71	↑ 62%
	IQL	Vanilla	53.29 ± 2.71	28.30 ± 6.21	25.63 ± 16.76	107.22	
		CUORL	6.70 ± 3.80	14.91 ± 12.51	12.59 ± 10.06	34.20	
		Harness	55.17 ± 3.55	32.05 ± 9.78	24.79 ± 12.61	**112.01**	
		CLTV	55.79 ± 4.44	39.56 ± 2.78	70.73 ± 4.11	166.08	↑ 48%
Walker2d	CQL	Vanilla	27.55 ± 20.29	83.81 ± 19.68	2.94 ± 3.80	**114.30**	
		CUORL	-0.01 ± 0.09	-0.03 ± 0.07	8.22 ± 9.01	8.18	
		Harness	23.70 ± 15.89	87.04 ± 17.11	1.77 ± 1.35	112.51	
		CLTV	70.45 ± 8.72	97.43 ± 7.74	30.23 ± 41.25	198.11	↑ 73%
	IQL	Vanilla	63.98 ± 5.90	45.67 ± 10.30	96.65 ± 5.87	206.30	
		CUORL	5.05 ± 1.94	3.58 ± 0.72	88.71 ± 5.00	97.34	
		Harness	68.20 ± 3.17	45.12 ± 14.86	95.48 ± 3.81	**208.80**	
		CLTV	68.37 ± 4.12	89.51 ± 9.03	110.74 ± 0.66	268.62	↑ 28%

6.2 Impact of CL on Performance of CLTV

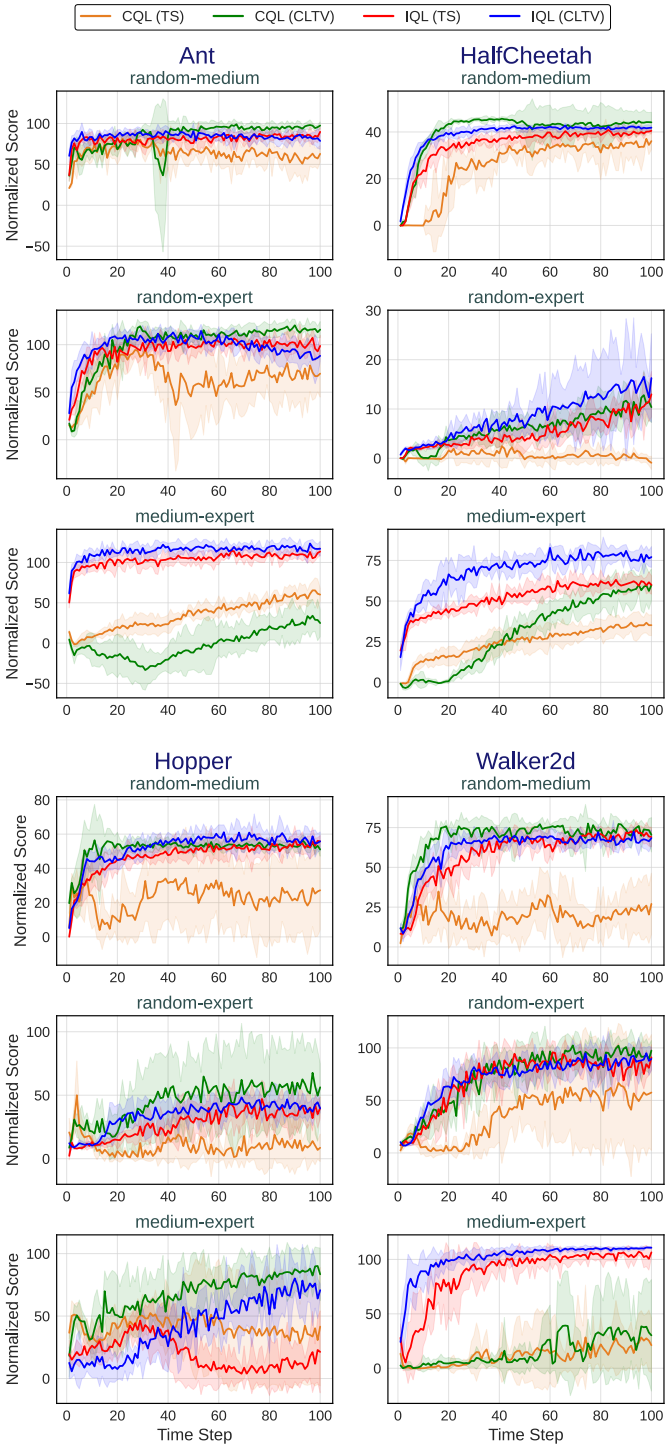

Figure 1: Performance of TS method, compared with CLTV. The curves are averaged over 5 seeds, with the shaded areas showing the confidence interval across seeds.

Our method consists of two key components: Transition Scoring (TS) and Curriculum Learning-Based Trajectory Valuation (CLTV). TS assigns scores to transitions and adjusts rewards, enabling the agent to focus on the transitions most relevant to the target domain. Building upon TS, CLTV integrates curriculum learning (CL) to prioritize high-quality trajectories during training.

Figure 1 illustrates the impact of CL on the performance of CLTV. The results demonstrate that while TS provides a strong foundation for improving agent performance, CLTV consistently achieves better results across various environments, including faster convergence and higher normalized scores. For example, in the Ant domain, CLTV achieves scores approaching or exceeding 100 in the random-medium and random-expert datasets, whereas TS reaches around 50. Similarly, in HalfCheetah, Hopper, and Walker2d, CLTV builds on the framework of TS to deliver competitive results, especially in the random-medium and medium-expert datasets. Additionally, CLTV tends to reduce variance and improve stability in some environments, as evidenced by the smoother learning curves for Walker2d and Ant.

Although performance improvements in HalfCheetah and Hopper are less pronounced, CLTV consistently surpasses TS in most cases, highlighting its robustness across a wide range of conditions.

In general, CLTV improves the learning performance of TS, particularly in settings with diverse data distributions. By prioritizing high-value trajectories, CLTV enables the agent to focus on relevant experiences, resulting in more efficient learning and better adaptability to variations in data quality.

In general, CLTV improves TS by prioritizing high-value trajectories, leading to faster convergence and improved stability. Its adaptability across diverse environments makes it a robust approach for optimizing trajectory valuation in offline RL.

7 CONCLUSION

In this work, we introduced Curriculum Learning-Based Trajectory Valuation (CLTV) to enhance the performance of offline RL algorithms when dealing with mixed datasets, characterized by a predominance of source data generated from random or suboptimal policies and a limited amount of target data from higher-quality policies. Our approach employs Transition Scoring (TS) to evaluate transition items based on their relevance to the target domain. By leveraging these scores, CLTV prioritizes high-quality trajectories through curriculum learning, enabling the agent to utilize valuable transitions from datasets generated by diverse policies. Experimental results across various algorithms and MuJoCo environments demonstrated that CLTV significantly improves performance and accelerates convergence. CLTV is particularly effective in batch-constrained scenarios, operating efficiently with limited data to ensure robust learning even when data is scarce. Furthermore, our theoretical analysis validated the effectiveness of our approach.

ACKNOWLEDGMENTS

This work was supported by the research projects “QuBRA” and “BIFOLD”, funded by the Federal Ministry of Education and Research (BMBF) under grant IDs 13N16052 and BIFOLD24B, respectively.

REFERENCES

[1] Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. 2019. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep* 32 (2019), 96.

[2] Cynthia Breazeal and Andrea L Thomaz. 2008. Learning from human teachers with socially guided exploration. In *2008 IEEE International Conference on Robotics and Automation*. IEEE, 3539–3544.

[3] Yuanying Cai, Chuheng Zhang, Hanye Zhao, Li Zhao, and Jiang Bian. 2023. Curriculum offline reinforcement learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*. 1221–1229.

[4] Imre Csiszár and János Körner. 2011. *Information theory: coding theorems for discrete memoryless systems*. Cambridge University Press.

[5] Rasheed El-Bouri, David Eyre, Peter Watkinson, Tingting Zhu, and David Clifton. 2020. Student-teacher curriculum learning via reinforcement learning: Predicting hospital inpatient admission location. In *International Conference on Machine Learning*. PMLR, 2848–2857.

[6] Benjamin Eysenbach, Swapnil Asawa, Shreyas Chaudhari, Sergey Levine, and Ruslan Salakhutdinov. 2020. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. *arXiv preprint arXiv:2006.13916* (2020).

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

[8] Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. 2019. Benchmarking batch deep reinforcement learning algorithms. *arXiv preprint arXiv:1910.01708* (2019).

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

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

[11] Yasuhiro Fujita, Prabhat Nagarajan, Toshiki Kataoka, and Takahiro Ishikawa. 2021. Chainerrl: A deep reinforcement learning library. *Journal of Machine Learning Research* 22, 77 (2021), 1–14.

[12] Amirata Ghorbani and James Zou. 2019. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*. PMLR, 2242–2251.

[13] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.

[14] Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. 2017. Darla: Improving zero-shot transfer in reinforcement learning. In *International Conference on Machine Learning*. PMLR, 1480–1490.

[15] Zhang-Wei Hong, Pulkit Agrawal, Rémi Tachet des Combes, and Romain Laroche. 2023. Harnessing mixed offline reinforcement learning datasets via trajectory weighting. *arXiv preprint arXiv:2306.13085* (2023).

[16] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado Van Hasselt, and David Silver. 2018. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933* (2018).

[17] Yuenan Hou, Lifeng Liu, Qing Wei, Xudong Xu, and Chunlin Chen. 2017. A novel DDPG method with prioritized experience replay. In *2017 IEEE international conference on systems, man, and cybernetics (SMC)*. IEEE, 316–321.

[18] David Isele, Alireza Nakhaei, and Kikuo Fujimura. 2018. Safe reinforcement learning on autonomous vehicles. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1–6.

[19] Chaohai Kang, Chuiting Rong, Weijian Ren, Fengcai Huo, and Pengyun Liu. 2021. Deep deterministic policy gradient based on double network prioritized experience replay. *IEEE Access* 9 (2021), 60296–60308.

[20] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. 2020. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems* 33 (2020), 21810–21823.

[21] Pascal Klink, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. 2021. Boosted curriculum reinforcement learning. In *International Conference on Learning Representations*.

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

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

[24] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 1179–1191.

[25] Sascha Lange, Thomas Gabel, and Martin Riedmiller. 2012. Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*. Springer, 45–73.

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

[27] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[28] Minghuan Liu, Hanye Zhao, Zhengyu Yang, Jian Shen, Weinan Zhang, Li Zhao, and Tie-Yan Liu. 2021. Curriculum offline imitating learning. *Advances in Neural Information Processing Systems* 34 (2021), 6266–6277.

[29] Amarsagar Reddy Ramapuram Matavalam, Kishan Prudhvi Guddanti, Yang Weng, and Venkataramana Ajjarapu. 2022. Curriculum based reinforcement learning of grid topology controllers to prevent thermal cascading. *IEEE Transactions on Power Systems* (2022).

[30] 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. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[31] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. 2020. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research* 21, 181 (2020), 1–50.

[32] Zhipeng Ren, Daoyi Dong, Huaxiong Li, and Chunlin Chen. 2018. Self-paced prioritized curriculum learning with coverage penalty in deep reinforcement learning. *IEEE transactions on neural networks and learning systems* 29, 6 (2018), 2216–2226.

[33] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952* (2015).

[34] Takuma Seno and Michita Imai. 2022. d3rlpy: An offline deep reinforcement learning library. *Journal of Machine Learning Research* 23, 315 (2022), 1–20.

[35] Dylan Slack, Yinlam Chow, Bo Dai, and Nevan Wichers. 2022. Safer: Data-efficient and safe reinforcement learning via skill acquisition. *arXiv preprint arXiv:2202.04849* (2022).

[36] Yihao Sun. 2023. Offlinerl-kit: An elegant pytorch offline reinforcement learning library.

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

[38] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[39] Shengpu Tang and Jenna Wiens. 2021. Model selection for offline reinforcement learning: Practical considerations for healthcare settings. In *Machine Learning for Healthcare Conference*. PMLR, 2–35.

[40] Sebastian Thrun and Anton Schwartz. 1993. Issues in using function approximation for reinforcement learning. In *Proceedings of the Fourth Connectionist Models Summer School*. Hillsdale, NJ, 255–263.

[41] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 5026–5033.

[42] Alexandre B. Tsybakov. 2008. *Introduction to Nonparametric Estimation* (1st ed.). Springer Publishing Company, Incorporated.

[43] Matteo Turchetta, Andrey Kolobov, Shital Shah, Andreas Krause, and Alekh Agarwal. 2020. Safe reinforcement learning via curriculum induction. *Advances in Neural Information Processing Systems* 33 (2020), 12151–12162.

[44] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8 (1992), 229–256.

[45] Sijia Xu, Hongyu Kuang, Zhuang Zhi, Renjie Hu, Yang Liu, and Huyang Sun. 2019. Macro action selection with deep reinforcement learning in starcraft. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 15. 94–99.

[46] Jinsung Yoon, Sercan Arik, and Tomas Pfister. 2020. Data valuation using reinforcement learning. In *International Conference on Machine Learning*. PMLR, 10842–10851.