Forecasting in Offline Reinforcement Learning for Non-stationary Environments

Suzan Ece Ada[1,2] **Georg Martius**[2] **Emre Ugur**[1] **Erhan Oztop**[3,4]
[1]Bogazici University, Türkiye [2]University of Tübingen, Germany
[3]Ozyegin University, Türkiye [4]Osaka University, Japan
ece.ada@bogazici.edu.tr

Abstract

Offline Reinforcement Learning (RL) provides a promising avenue for training policies from pre-collected datasets when gathering additional interaction data is infeasible. However, existing offline RL methods often assume stationarity or only consider synthetic perturbations at test time, assumptions that often fail in real-world scenarios characterized by abrupt, time-varying offsets. These offsets can lead to partial observability, causing agents to misperceive their true state and degrade performance. To overcome this challenge, we introduce **F**orecasting in Non-stationary **O**ffline **RL** (FORL), a framework that unifies (i) conditional diffusion-based candidate state generation, trained without presupposing any specific pattern of future non-stationarity, and (ii) zero-shot time-series foundation models. FORL targets environments prone to unexpected, potentially non-Markovian offsets, requiring robust agent performance from the onset of each episode. Empirical evaluations on offline RL benchmarks, augmented with real-world time-series data to simulate realistic non-stationarity, demonstrate that FORL consistently improves performance compared to competitive baselines. By integrating zero-shot forecasting with the agent's experience, we aim to bridge the gap between offline RL and the complexities of real-world, non-stationary environments.

1 Introduction

Offline Reinforcement Learning (RL) leverages static datasets to avoid costly or risky online interactions [1, 2]. Yet, agents trained on fully observable states often fail when deployed with noisy or corrupted observations. While robust offline RL methods address test-time perturbations, such as sensor noise or adversarial attacks [3], a critical gap persists in addressing non-stationarity within the observation function—a challenge that fundamentally alters the agent's perception of the environment over time.

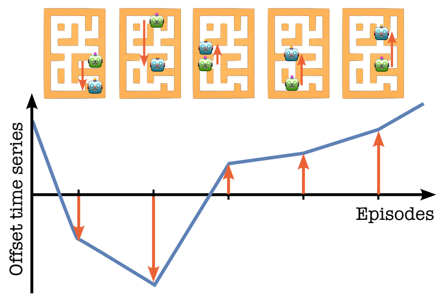

Figure 1: **Setting.** The agent does not know its location in the environment because its perception is offset every episode j by an unknown offset b^j (only vertical offsets are illustrated). FORL leverages historical offset data and offline RL data (from a stationary phase) to forecast and correct for new offsets at test time. Ground-truth offsets (⬇,⬆) are hidden throughout the evaluation episodes.

Prior *online algorithms* that consider the scope of non-stationarity as the observation function focus on learning agent morphologies [4] and generalization in Block MDPs [5]. While this scope of non-stationarity holds significant potential for real-world applications [6], it remains underexplored. We focus on the episodic evolution of the observation function at test-time in offline RL. In our setup, each dimension of an agent's state is influenced by an unknown, time-dependent constant additive offset that remains fixed within a single operational interval (an "episode"). This leads to a stream of evolving observation functions [7], extending across multiple future episodes, **where the offsets remain hidden throughout the prediction window**. For instance, industrial robots might apply a daily calibration offset to each joint, while sensors can exhibit a deviation until the next scheduled recalibration. Similarly, in healthcare or finance, data may be partially aggregated or withheld to comply with regulations, effectively obscuring finer-grained variations and leaving a single offset as the dominant factor per episode. By only storing these representative offsets, we circumvent the challenges of continuous interaction buffers in bandwidth-constrained or privacy-sensitive environments. Because the offset can differ across state dimensions (e.g., different sensor or actuation channels), each state dimension can be affected by a different unknown bias that stays constant for that episode but evolves differently across episodes. Approaches that assume predefined perturbations can struggle with these abrupt, episodic shifts because such offsets violate the typical assumption of smoothly varying observation functions. Frequent retraining, hyperparameter optimization, or extensive online adaptation to new observation function evolution patterns are costly, risky (due to trial-and-error in safety-critical settings), and may be infeasible if these patterns no longer reflect assumptions made during training. By separating offset data (episodic calibration values) from the massive replay buffers, a zero-shot forecasting-based approach can anticipate each new offset from the beginning of the episode without requiring policy updates or making assumptions on task evolution at test time [8]. Modeling these multidimensional additive offsets as stable, per-episode constants presents a robust and efficient way to handle time-varying conditions in non-stationary environments where the evolution of tasks follows a non-Markovian, time-series pattern [9], mitigating the risks of online exploration.

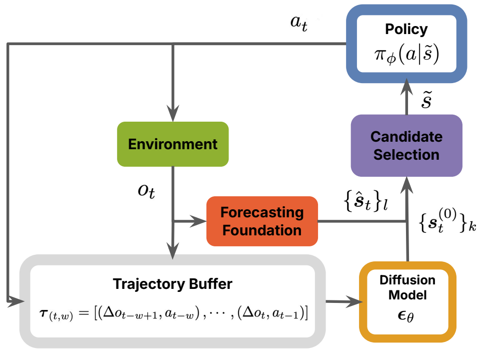
a_t
Policy
$\pi_\phi(a|\tilde{s})$
$\tilde{s}$
Environment
Candidate Selection
o_t
Forecasting Foundation
$\{\hat{\boldsymbol{s}}_t\}_l$
$\{\boldsymbol{s}_t^{(0)}\}_k$
Trajectory Buffer
$\boldsymbol{\tau}_{(t,w)} = [(\Delta o_{t-w+1}, a_{t-w}), \cdots, (\Delta o_t, a_{t-1})]$
Diffusion Model
$\boldsymbol{\epsilon}_\theta$

Figure 2: Overview of the proposed FORL framework at test-time. The observations are processed by both the trajectory buffer and the time-series **forecasting foundation** module [10]. Observation changes and actions sampled from the **policy** $(\Delta o, a)$, are stored in the trajectory buffer. The **diffusion model** generates candidate states $\{\boldsymbol{s}_t^{(0)}\}_k$ conditioned on $\boldsymbol{\tau}_{(t,w)}$. The **candidate selection** module then generates the estimated $\tilde{s}_t$.

We consider an offline RL setting during training where we have only access to a standard offline RL dataset collected from a stationary environment [3] with fully observable states. Initial data may be collected under near-ideal conditions and then gradually affected by wear, tear, or other natural shifts—even as the underlying physical laws (dynamics) remain unchanged. At test time, however, we evaluate in a non-stationary environment where both the observation function and the observation space change due to time-dependent external factors. This setup can be interpreted as environments shifting along observation space dimensions while the initial state of the agent is sampled from a uniform distribution over the state space. A simplified version of this setup for an offset affecting only one dimension of the state is illustrated in Figure 1. Here, the agent "knows" it is in a maze but does not know where it is in the maze. Furthermore, **it will remain uncertain of its location across all episodes at test-time**, as in every episode, a new offset leads to a systematic misalignment between perceived and actual positions. Importantly, these offsets may not conform to Gaussian or Markovian assumptions; instead, they may stem directly from complex, real-world time-series data [9] and remain constant throughout each episode. As a result, standard noise-driven or parametric state-estimation techniques, which typically rely on smoothly varying or randomly perturbed functions, cannot reliably identify these persistent, episode-wide offsets that are not available after the episode terminates. While zero-shot forecasting can adjust observation offsets, its performance depends on the forecaster's accuracy. Similarly, integrating zero-shot forecasting into a model-based offline RL approach [3] can underperform when real-world offsets deviate from predefined assumptions about future observation functions. Our approach uses the insight that the belief of the true states can be refined from a sequence of actions and effects. For instance, in maze navigation, if an agent misjudges its location and hits a wall, analyzing its actions and delta observations leading to the collision can provide evidence for likely locations within the maze.

We propose the **F**orecasting in Non-stationary **O**ffline **RL** (FORL) framework (Figure 2) for test-time adaptation in non-stationary environments where the observation function is perturbed by an arbitrary time-series. Our framework has two main ingredients: forecasting offsets with a zero-shot time-series forecasting model [10] from past episode offsets (ground truth offsets after the episode terminates are not accessible at test-time) and a within-episode update of the state estimation using a conditional diffusion model [11] trained on offline stationary data.

Contributions. We unify the strengths of probabilistic forecasting and decision-making under uncertainty to enable continuous adaptation when the environment diverges from predictions. Consequently, our framework: *(1)* accommodates future offsets *without assuming specific non-stationarity patterns during training*, eliminating the need for retraining and hyperparameter tuning when the agent encounters new, unseen non-stationary patterns at test time, and *(2) targets non-trivial non-stationarities at test time without requiring environment interaction or knowledge of POMDPs during training*. *(3)* FORL introduces a novel, modular framework combining a conditional diffusion model (FORL-DM) for multimodal belief generation with a lightweight Dimension-wise Closest Match (DCM) fusion strategy, validated by extensive experiments on no-access to past offsets, policy-agnostic plug-and-play, offset magnitude-scaling, and inter/intra-episode drifts. *(4)* We propose a novel benchmark that integrates offsets from real-world time-series datasets with standard offline RL benchmarks and demonstrate that FORL consistently outperforms baseline methods.

Background: Diffusion Models Denoising diffusion models [12, 13] aim to model the data distribution with $p_\theta(\boldsymbol{x}_0) := \int p_\theta(\boldsymbol{x}_{0:N})\, d\boldsymbol{x}_{1:N}$ from samples x_0 in the dataset. The joint distribution follows the Markov Chain $p_\theta\left(\boldsymbol{x}_{0:N}\right) := \mathcal{N}(\boldsymbol{x}_N; \mathbf{0}, \mathbf{I}) \prod_{n=1}^{N} p_\theta\left(\boldsymbol{x}_{n-1} \mid \boldsymbol{x}_n\right)$ where $\boldsymbol{x}_n$ is the noisy sample for diffusion timestep n and $p_\theta\left(\boldsymbol{x}_{n-1} \mid \boldsymbol{x}_n\right) := \mathcal{N}\left(\boldsymbol{x}_{n-1}; \boldsymbol{\mu}_\theta\left(\boldsymbol{x}_n, n\right), \boldsymbol{\Sigma}_\theta\left(\boldsymbol{x}_n, n\right)\right)$. During training, we use the samples from the distribution $q\left(\boldsymbol{x}_n \mid \boldsymbol{x}_0\right) = \mathcal{N}\left(\boldsymbol{x}_n; \sqrt{\bar{\alpha}_n}\boldsymbol{x}_0, \left(1 - \bar{\alpha}_n\right)\mathbf{I}\right)$ where $\bar{\alpha}_n = \prod_{i=1}^{n} \alpha_i$ [11]. General information on diffusion models is given in Section B.3.

2 Method

In this section, we formulate our problem statement and describe our FORL diffusion model trained on the offline RL dataset to predict plausible states. Then, we introduce our online state estimation method, Dimension-wise Closest Match that uses plausible states predicted by the multimodal FORL diffusion model (DM) and the states predicted from past episodes prior to evaluation by a probabilistic unimodal zero-shot time-series foundation model.

2.1 Problem Statement

Training (Offline Stationary MDP) We begin with an episodic, stationary Markov Decision Process (MDP) $\mathcal{M}_{\text{train}} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \rho_0)$, where the initial state distribution ρ_0 is a uniform distribution over the state space $\mathcal{S}$. We only have access to an offline RL dataset $\mathcal{D} = \{(\boldsymbol{s}_t^k, \boldsymbol{a}_t^k, \boldsymbol{s}_{t+1}^k, r_t^k)\}$ with k transitions collected from this MDP. Crucially, our FORL diffusion model and a diffusion policy [14] are trained offline using this dataset, such as the standard D4RL benchmark [15], without making any assumptions about how the environment might become non-stationary at test time.

Test Environment (Sequence of POMDPs) At test time, the agent faces an infinite sequence of POMDPs $\{\hat{\mathcal{M}}_j\}_{j=1}^{\infty}$. Each POMDP is described by a 7-tuple [16] $\hat{\mathcal{M}}_j = \left(\mathcal{S}, \mathcal{A}, \mathcal{O}_j, \mathcal{T}, \mathcal{R}, \rho_0, \mathbf{x}_j\right)$, where the state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $\mathcal{T}$, and the reward function $\mathcal{R}$ remain identical to the training MDP. $\mathbf{x}$ is the observation function, where we restrict ourselves to deterministic versions ($\mathbf{x} : \mathcal{S} \to \mathcal{O}$) [6, 17]. Non-stationary environments can be formulated in different ways. In Khetarpal et al. [6], a *general non-stationary RL* formulation is put forward, which allows each component of the underlying MDP or POMDP to evolve over time, i.e. $\left(\mathcal{S}(t), \mathcal{A}(t), \mathcal{T}(t), \mathcal{R}(t), \mathbf{x}(t), \mathcal{O}(t)\right)$. A set κ specifies which of these components vary, and a *driver* determines how they evolve. In particular, *passive* drivers of non-stationarity imply that exogenous factors alone govern the evolution of the environment, independent of the agent's actions. In this work, we consider the scope of non-stationarity [6] (Section B.2) to be the observation function and the observation space, i.e. $\kappa = \{\mathbf{x}, \mathcal{O}\}$.

We consider the case where non-stationarity unfolds over episodes and where the observation function $\mathbf{x}_j$ is different in each episode j. The change in the observation function is assumed to have an additive structure and is independent of the agent's actions (passive non-stationarity [6]). Concretely, the function $\mathbf{x}_j$ offsets states s_t by a fixed offset $b^j \in \mathbb{R}^n$:

$$\mathcal{O}_j \;=\; \{\, s + b^j : s \in \mathcal{S}\}, \quad \mathbf{x}_j(s) \;=\; s + b^j.$$

Importantly, the sequence (b^j) can evolve under arbitrary real-world time-series data, and the agent **does not have access to the ground-truth information throughout the evaluation**—similar to scenarios where observations are only available periodically and shifts occur between these intervals. Thus, the episodes have a temporal order, relating to Non-Stationary Decision Processes (NSDP), defining a sequence of POMDPs [7] (Section B.2).

Partial Observability and Historical Context Since b^j is **never directly observed** for P episodes into the future, each $\hat{\mathcal{M}}_j$ is a POMDP. The agent receives only the offset-shifted observations (o_t), where $o_t = s_t + b^j$ without any noise. Moreover, the agent may have access to a limited historical context of previous offsets $(b^{j-C}, \ldots, b^{j-1})$ at discrete intervals P, but **no direct information** about future offsets $(b^j, b^{j+1}, \ldots b^{j+P-1})$. Hence, the agent must forecast and/or adapt to unknown future offsets without prior non-stationary training.

Partial Identifiability Despite observing $o_t = s_t + b^j$, the agent cannot generally disentangle s_t from b^j. For any single observation, there are infinite possible pairs of state and offset that yield $o_t = s' + b'$. Additionally, the initial state distribution ρ is uniform and does not provide information about b. Thus, we can only form a belief over s_t and refine that belief based on two sources of information: a) the sequence of actions and effects observed within an episode and b) the sequence of past identified offsets. To exploit source a, we utilize a predictive model of commonly expected outcomes based on a diffusion model, which will be explained next. To make use of source b, we use a zero-shot forecasting model (see Section 2.3 for details). Afterwards, both pieces of information are fused (Section 2.4).

2.2 FORL Diffusion Model

In our setting, we assume that the offsets added to the states are unobservable at test time, while the transition dynamics of the evaluation environment remain unchanged. To eventually reduce the uncertainty about the underlying state, we perform filtering or belief updates using the sequence of past interactions. To understand why the history of interactions is indicative of a particular ground truth state, consider the following example in a maze environment illustrated in Fig. 3. When the agent moves north for three steps and then bumps into a wall, the possible ground truth states can only be those three steps south of any wall. The agent cannot observe the hidden green trajectory of ground-truth states; it only has access to the sequence of observation changes (Δo) and action vectors, which narrows down the possible positions to four candidate regions—exactly those identified by our model. Clearly, the distribution of possible states is highly multimodal, such that we propose using a diffusion model as a flexible predictive model of plausible states given the observed actions and outcomes. Diffusion models excel at capturing multimodal distributions [14], making them well-suited for our task. We train the diffusion model on offline data without offsets $(b_j = 0)$ which we detail below.

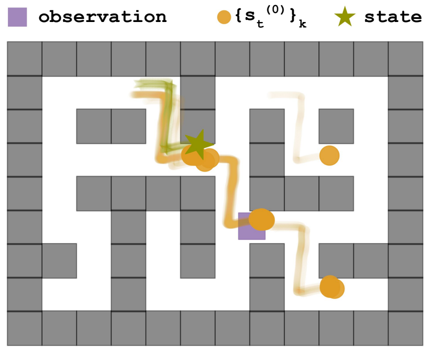

Figure 3: Candidate states generated by FORL Diffusion Model.

To distinguish between the trajectory timesteps in reinforcement learning (RL) and the timesteps in the diffusion process, we use the subscript $t \in \{0, \ldots, T\}$ to refer to RL timesteps and $n \in \{0, \ldots, N\}$ for diffusion timesteps. We first begin by defining a subsequence of a trajectory

$$\boldsymbol{\tau}_{(t,w)} = \left[\left(\Delta o_{t-w+1}, a_{t-w}\right), \cdots, \left(\Delta o_t, a_{t-1}\right)\right]. \tag{1}$$

where delta observations $\Delta o_t = o_t - o_{t-1} = s_t - s_{t-1}$ denote the state changes (effects), w is the window size. Using a conditional diffusion model, we aim to model the distribution $p\left(\boldsymbol{s}_t \mid \boldsymbol{\tau}_{(t,w)}\right)$. For that, we define the reverse process (denoising) as

$$p\big(s_t^{(N)}\big) \prod_{n=1}^{N} p_\theta\left(\boldsymbol{s}_t^{(n-1)} \mid \boldsymbol{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}\right), \quad p\big(s_t^{(N)}\big) = \mathcal{N}(\boldsymbol{s}_t^{(N)}; \mathbf{0}, \mathbf{I}) \tag{2}$$

and p_θ is modeled as the distribution $\mathcal{N}(\boldsymbol{s}_t^{(n-1)}; \mu_\theta(\boldsymbol{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}, n), \Sigma_\theta(\boldsymbol{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}, n))$ with a learnable mean and variance. We could directly supervise the training of μ_θ using the forward (diffusion) process. Following Ho et al. [11], Song et al. [18], we compute a noisy sample $s_t^{(n)}$ based on the true sample $s_t = s_t^{(0)}$:

$$s_t^{(n)} = \sqrt{\bar{\alpha}(n)} s_t + \sqrt{1 - \bar{\alpha}(n)} \boldsymbol{\epsilon} \tag{3}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is the noise, $\bar{\alpha}(n) = \prod_{i=1}^{n} \alpha(i)$ and the weighting factors $\alpha(n) = e^{-\left(\beta_{\text{min}}\left(\frac{1}{N}\right) + (\beta_{\text{max}} - \beta_{\text{min}})\frac{2n-1}{2N^2}\right)}$ where $\beta_{\text{max}} = 10$ and $\beta_{\text{min}} = 0.1$ are parameters introduced for empirical reasons [19].

We can equally learn to predict the true samples by learning a noise model [20]. Hence, we train a noise model $\boldsymbol{\epsilon}_\theta(\boldsymbol{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}, n)$ that learns to predict the noise vector $\boldsymbol{\epsilon}$. By using the conditional version of the simplified surrogate objective from [11], we minimize

$$\mathcal{L}_p(\theta) = \underset{n, \boldsymbol{\tau}, \boldsymbol{s_t}, \boldsymbol{\epsilon}}{\mathbb{E}} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\boldsymbol{s}^{(n)}, \boldsymbol{\tau}_{(t,w)}, n \right) \right\|^2 \right] \tag{4}$$

where $\boldsymbol{s_t}$ is the state sampled from the dataset D for $t \sim U_T(\{w, \ldots, T-1\})$, $\boldsymbol{s}^{(n)}$ is computed according to Eq. (3), $\boldsymbol{\epsilon}$ is the noise, and $n \sim U_D(\{1, \ldots, N\})$ is the uniform distribution used for sampling the diffusion timestep.

We use the true data sample $\boldsymbol{s_t}$ from the offline RL dataset to obtain the noisy sample in Eq. (3). Leveraging our model's capacity to learn multimodal distributions, we generate a set of k samples $\{\boldsymbol{s}_t^{(0)}\}$ as our **predicted state candidates** in parallel from the reverse diffusion chain. We use the noise prediction model [11] with the reverse diffusion chain $\boldsymbol{s}_t^{(n-1)} \mid \boldsymbol{s}_t^{(n)}$ formulated as

$$\frac{\boldsymbol{s}_t^{(n)}}{\sqrt{\alpha_{(n)}}} - \frac{1 - \alpha_{(n)}}{\sqrt{\alpha_{(n)}(1 - \bar{\alpha}_{(n)})}} \boldsymbol{\epsilon}_\theta(\boldsymbol{s}_t^{(n)}, \boldsymbol{\tau}_{(t,w)}, n) + \sqrt{1 - \alpha_{(n)}} \boldsymbol{\epsilon} \tag{5}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ for $n = N, \ldots, 1$, and $\boldsymbol{\epsilon} = 0$ for $n = 1$ [11]. Below, we detail how the state candidates are used during an episode.

2.3 Forecasting using Zero-Shot Foundation Model

Because we assume that the offsets b^j originate from a time series, we propose using a probabilistic zero-shot forecasting foundation model *(Zero-Shot FM)* such as Lag-Llama [10], to forecast future offsets from past ones. We assume that after P episodes, the true offsets are revealed, and we predict the offsets for the following P episodes. Using the probabilistic *Zero-Shot FM* we generate $(\hat{b}_l^j, \ldots, \hat{b}_l^{j+P-1})$, where (l) denotes the number of samples generated for each episode (timestamp). Since Lag-Llama is a probabilistic model, it can generate multiple samples per timestamp, conditioned on C number of past contexts $(b^{j-C}, \ldots, b^{j-1})$. In practice, we forecast every dimension of b independently since the *Zero-Shot FM* (Lag-Llama [10]) is a univariate probabilistic model.

Algorithm 1 Candidate Selection

```
1:  Sample $\{\{\hat{b}\}_l^1, \ldots, \{\hat{b}\}_l^P\} \sim$ *Zero-Shot FM*
2:  **for** each episode $p = 1, \cdots, P$ **do**
3:      $t = 0$
4:      Reset environment $o_0 \sim \mathcal{E}$
5:      $\tilde{s} \leftarrow o_0 - \overline{\{\hat{b}\}_l^p}$
6:      Initialize $\boldsymbol{\tau}_{(t,w)}$
7:      **while** not *done* **do**
8:          Sample $a^{(0)} \sim \pi_\phi(a|\tilde{s})$
9:          Take action $a^{(0)}$ in $\mathcal{E}$, observe $o_{t+1}$
10:         $\{\hat{s}_{t+1}^b\}_l \leftarrow o_{t+1} - \{\hat{b}\}_l^p$
11:         $\boldsymbol{\tau}_{(t+1,w)} = \text{PUSH}\left(\boldsymbol{\tau}_{(t,w)}, \left(\Delta o_{t+1}, a^{(0)}\right)\right)$
12:         $\boldsymbol{\tau}_{(t+1,w)} = \text{POP}\left(\boldsymbol{\tau}_{(t+1,w)}, (\Delta o_{t-w+1}, a_{t-w})\right)$
13:         **if** $t > w$ **then**
14:             Sample $\{\boldsymbol{s}_{t+1}^{(0)}\}_k$ from FORL by Eq. 5
15:             $\tilde{s} \leftarrow$ **DCM**$(\{\boldsymbol{s}_{t+1}^{(0)}\}_k, \{\hat{s}_{t+1}^b\}_l)$
16:         **else**
17:             $\tilde{s} \leftarrow o_{t+1} - \overline{\{\hat{b}\}_l^p}$
18:         **end if**
19:         $t \leftarrow t + 1$
20:     **end while**
21: **end for**
```

2.4 FORL State Estimation

The next step in our method is to fuse the information from the forecaster and the diffusion model into a state estimate used for control at test time.

At the beginning of an episode, no information can be obtained from the diffusion model, so for the first w steps we only rely on the forecaster's mean prediction, i.e. $\tilde{s}_t = o_t - \overline{\hat{b}^j}$ where the mean is taken over the l samples.

As soon as w steps are taken, our FORL *State Estimation* improves on the inferred state as detailed below. Figure 2 offers an overview of the entire system and Algorithm 1 provides a detailed pseudocode.

To recap, the diffusion model generates samples $\{\boldsymbol{s}_t^{(0)}\}_k$ from the in-episode history τ, Eq. (1). These samples represent a multimodal distribution of plausible state regions. The *Zero-Shot FM* generates l samples of offsets $\{\hat{b}\}_l$ from which we compute forecasted states using $\{\hat{\boldsymbol{s}}_t\}_l = o_t - \{\hat{b}\}_l$.

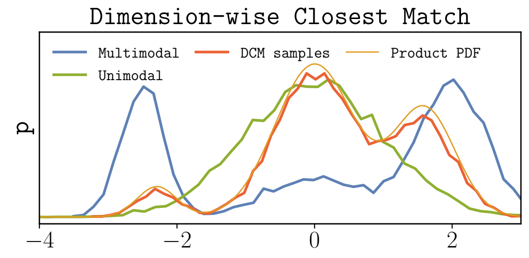

Figure 4: Distribution of samples produced by DCM (histograms for 10k samples for illustration).

FORL: Dimension-wise Closest Match (DCM) We propose a lightweight approach to sample a good estimate based on the samples from the multimodal (*diffusion model* $\{\boldsymbol{s}_t^{(0)}\}_k$) and unimodal (*Zero-Shot FM* $\{\hat{s}_t^b\}_l$) distributions. Let $\mathcal{D}_{\text{diffusion}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, $\mathcal{D}_{\text{timeseries}} = \{\mathbf{y}_1, \ldots, \mathbf{y}_l\}$, where $\mathbf{x}_i, \mathbf{y}_j \in \mathbb{R}^n$. Then DCM constructs $\mathbf{z} \in \mathbb{R}^n$ by

$$z_d = y_{j^*(d),d} \quad \text{where} \quad j^*(d) = \arg\min_j \Big(\min_i \big| x_{i,d} - y_{j,d} \big| \Big),$$

where $d = 1 \ldots n$. In other words, for each dimension d, we choose the sample from $\mathcal{D}_{\text{timeseries}}$ that has the closest sample in $\mathcal{D}_{\text{diffusion}}$. The process is straightforward yet effective, and under ideal sampling conditions for a toy dataset in Fig. 4, DCM approximately samples from the product distribution. DCM uses a non-parametric search to find the forecast sample with the highest score, which corresponds to the minimum dimension-wise distance. DCM's prediction error is governed by the accuracy of the forecast samples in the unimodal $\mathcal{D}_{\text{timeseries}}$ that achieves this best score. As we will demonstrate in the experiments, this approach empirically yields lower maximum errors and is more stable compared to other methods.

FORL Algorithm Algorithm 1 summarizes the entire inference process at test time. We begin the episode by relying on the forecasted states $\tilde{s}_0$. As more transitions $(\Delta o_t, a_{t-1})$ become available, the FORL diffusion model proposes candidate states $\{\boldsymbol{s}_t^{(0)}\}_k$ through *retrospection*—reasoning over the past in-episode experience to adapt state estimation on the fly when they begin to diverge from predictions. We then invoke DCM to blend the diffusion model's candidates with the foundation model's unimodal forecasts and obtain the final state estimate $\tilde{s}_t$. We use an off-the-shelf offline RL policy such as Diffusion-QL (DQL) [14] to select the agent's action a_t.

Summary By combining a powerful *zero-shot* forecasting model with a *conditional diffusion* mechanism, FORL addresses partial observability in continuous state and action space when ground-truth offsets are unavailable. This procedure is performed in the *absence of ground-truth offsets for past, current, and future episodes over the interval $j : j+P$ at test time*. DCM provides a computationally inexpensive yet effective way of using the multimodal diffusion candidates and unimodal time-series forecasts. This robust adaptation approach yields a state estimate $\tilde{s}_t$, aligned with the agent's retrospective experience in the stationary offline RL dataset, incorporating a prospective external offset forecast.

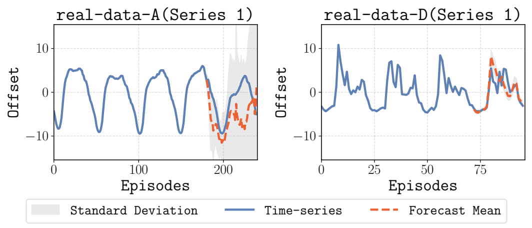

Figure 5: Zero-shot forecasting results of Lag-Llama [10] for the first univariate series (plotted) from the `real-data-A,D` datasets; experiments use the **first two series from each dataset**.

3 Experiments

We evaluate FORL across navigation and manipulation tasks in D4RL [15] and OGBench [21] offline RL environments, each augmented with five real-world non-stationarity domains sourced from [22]. Fig. 5 presents the ground truth, forecast mean, and standard deviation from Lag-Llama [10] for the *first series* of `real-data-A` and `real-data-D`. Our experiments address the following questions: **(1)** Does FORL maintain state-of-the-art performance when confronted with unseen non-stationary offsets? **(2)** How can we use FORL when we have no access to delayed past ground truth offsets? **(3)** How does DCM compare to other fusion approaches? **(4)** Can FORL handle intra-episode non-stationarity? **(5)** How gracefully does performance degrade as offset magnitude α is scaled from 0 (no offset) $\rightarrow$ 1 (our evaluation setup)? **(6)** Can FORL serve as a plug-and-play module for different offline RL algorithms without retraining? Extended results, forecasts for the remaining series, and implementation details are provided in the Appendix. Results average 5 seeds, unless noted.

Baselines We compare our approach with the following baselines: DQL [14], Flow Q-learning (FQL) [23] are diffusion-based and flow-based offline RL policies, respectively, that do not incorporate forecast information. DQL+LAG-$\bar{s}$, FQL+LAG-$\bar{s}$ extend DQL and FQL by using the sample mean of the forecasted states $\{\hat{s}_t\}_l$ at each timestep (using the constant per-episode predicted b^j). DQL+LAG-$\tilde{s}$ similarly extends DQL using the median. DMBP+LAG is a variant of the Diffusion Model–Based Predictor (DMBP)[3] (a robust offline RL algorithm designed to mitigate state-observation perturbations at test time, detailed in Appendix D) that integrates forecasted states from *Zero-Shot FM* [10] into its state prediction module. By using the model learned from the offline data, DMBP+LAG aims to refine the forecasted states to make robust state estimations. The underlying policies throughout our experiments are identical policy checkpoints for both our method and the baselines.

Table 1: **Normalized scores (mean ± std.) for FORL framework and the baselines.** Bold are the best values, and those not significantly different ($p > 0.05$, Welch's t-test).

`maze2d-medium`	DQL	DQL+LAG-$\bar{s}$	DMBP+LAG	FORL (ours)
`real-data-A`	30.2 ± 6.5	30.2 ± 8.6	25.1 ± 9.8	**63.3 ± 6.7**
`real-data-B`	14.1 ± 12.1	**53.4 ± 14.6**	**41.2 ± 21.1**	**66.5 ± 18.2**
`real-data-C`	-2.3 ± 3.3	56.7 ± 18.5	56.9 ± 18.4	**86.3 ± 15.7**
`real-data-D`	4.7 ± 5.0	36.9 ± 16.3	38.5 ± 14.2	**103.4 ± 11.9**
`real-data-E`	3.5 ± 8.8	8.7 ± 6.0	11.4 ± 2.8	**51.2 ± 13.7**
Average	10.0	37.2	34.6	**74.1**
`maze2d-large`				
`real-data-A`	16.2 ± 5.5	2.4 ± 1.1	4.2 ± 5.8	**42.9 ± 4.1**
`real-data-B`	-0.5 ± 2.9	5.5 ± 9.0	15.0 ± 14.6	**34.9 ± 9.2**
`real-data-C`	0.9 ± 1.7	16.6 ± 7.5	26.8 ± 8.4	**45.6 ± 4.1**
`real-data-D`	3.0 ± 6.6	8.6 ± 3.2	13.4 ± 4.1	**58.4 ± 6.5**
`real-data-E`	-2.1 ± 0.4	**2.6 ± 3.4**	**0.9 ± 3.7**	**12.0 ± 9.9**
Average	3.5	7.1	12.1	**38.8**
`antmaze-umaze-diverse`				
`real-data-A`	22.7 ± 3.0	41.0 ± 5.2	45.7 ± 4.8	**65.3 ± 8.7**
`real-data-B`	24.2 ± 3.5	48.3 ± 7.0	**62.5 ± 13.2**	**74.2 ± 10.8**
`real-data-C`	21.7 ± 3.5	50.4 ± 8.3	60.4 ± 3.9	**78.8 ± 8.5**
`real-data-D`	5.8 ± 2.3	26.7 ± 6.3	29.2 ± 5.9	**75.8 ± 8.0**
`real-data-E`	6.0 ± 6.8	58.0 ± 16.6	59.3 ± 7.6	**81.3 ± 6.9**
Average	16.1	44.9	51.4	**75.1**
`antmaze-medium-diverse`				
`real-data-A`	31.0 ± 6.5	**40.0 ± 5.7**	**39.7 ± 4.0**	**44.0 ± 7.9**
`real-data-B`	23.3 ± 4.8	**48.3 ± 4.8**	**43.3 ± 16.0**	**55.8 ± 7.0**
`real-data-C`	10.0 ± 2.3	**48.3 ± 3.4**	**49.6 ± 3.7**	**52.9 ± 9.5**
`real-data-D`	11.7 ± 5.4	46.7 ± 7.5	41.7 ± 6.6	**64.2 ± 8.6**
`real-data-E`	**18.7 ± 4.5**	**27.3 ± 8.6**	**26.0 ± 5.5**	**26.7 ± 4.7**
Average	18.9	42.1	40.1	**48.7**
`antmaze-large-diverse`				
`real-data-A`	11.0 ± 1.9	11.3 ± 4.9	9.0 ± 4.5	**34.3 ± 5.7**
`real-data-B`	5.8 ± 4.8	9.2 ± 4.6	8.3 ± 2.9	**46.7 ± 11.9**
`real-data-C`	5.4 ± 2.4	22.1 ± 5.6	17.9 ± 3.8	**33.8 ± 6.8**
`real-data-D`	2.5 ± 2.3	14.2 ± 3.7	14.2 ± 6.3	**46.7 ± 12.6**
`real-data-E`	**5.3 ± 3.8**	**3.3 ± 2.4**	**3.3 ± 0.0**	**11.3 ± 7.3**
Average	6.0	12.0	10.5	**34.6**
`kitchen-complete`				
`real-data-A`	**16.6 ± 1.4**	7.2 ± 1.9	8.7 ± 1.3	**12.0 ± 3.9**
`real-data-B`	12.9 ± 4.1	**32.7 ± 6.5**	20.0 ± 3.1	**33.1 ± 5.6**
`real-data-C`	13.4 ± 1.7	**23.9 ± 6.6**	**20.5 ± 3.3**	**23.9 ± 6.0**
`real-data-D`	7.5 ± 2.5	**24.0 ± 9.2**	**28.1 ± 8.1**	**27.1 ± 10.1**
`real-data-E`	**18.5 ± 6.0**	2.8 ± 2.1	6.2 ± 1.7	10.3 ± 3.0
Average	13.8	18.1	16.7	**21.3**

`cube-single-play`	FQL	FQL+LAG-$\bar{s}$	FORL-F (ours)
`real-data-A`	0.0 ± 0.0	0.0 ± 0.0	**23.7 ± 3.6**
`real-data-B`	0.0 ± 0.0	15.0 ± 7.0	**60.0 ± 7.0**
`real-data-C`	0.4 ± 0.9	10.0 ± 1.7	**42.1 ± 5.6**
`real-data-D`	0.0 ± 0.0	0.8 ± 1.9	**70.0 ± 13.0**
`real-data-E`	0.0 ± 0.0	0.0 ± 0.0	**32.7 ± 9.5**
Average	0.1	5.2	**45.7**
`antmaze-large-navigate`			
`real-data-A`	**22.7 ± 2.2**	1.3 ± 0.7	**24.3 ± 4.3**
`real-data-B`	21.7 ± 5.4	**29.2 ± 8.8**	**40.0 ± 7.6**
`real-data-C`	5.0 ± 1.1	34.6 ± 6.7	**55.8 ± 3.7**
`real-data-D`	0.8 ± 1.9	37.5 ± 5.1	**75.8 ± 5.4**
`real-data-E`	**10.0 ± 4.1**	3.3 ± 0.0	**15.3 ± 8.7**
Average	12.0	21.2	**42.2**

Illustrative Example Figs. 6 and 16 illustrate an agent navigating the `maze2d-large` environment where the true position is labeled as "state". The agent receives an observation indicating where it *believes* it is located due to unknown time-dependent factors. The candidate states predicted by the FORL diffusion model are shown as circles. Importantly, the agent's $(\Delta o, a)$-trajectory can reveal possible states for the agent. FORL's diffusion model (DM) compo-

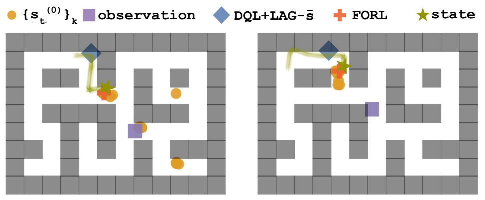

Figure 6: Visualization of states, predicted states as the agent navigates the environment.

nent predicts these candidate states by using observation changes (Δo) and corresponding actions (a). The possible candidate regions where the agent can be are limited, and our model successfully captures these locations. FORL's candidate selection module (DCM) uses the samples from the forecaster and the diffusion model to recover a close estimate for the state. In contrast, the baseline DQL+LAG-$\bar{s}$ relies on the forecaster [10] for state predictions, which are significantly farther from the actual state. Consistent with the results in Fig. 7, FORL reduces prediction errors at test-time, thereby improving performance.

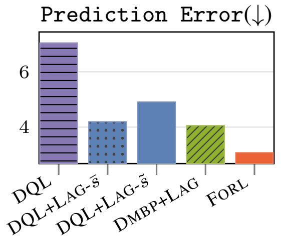

Figure 7: Prediction Error in recovering true agent state.

3.1 Results

FORL outperforms both pure forecasting (DQL+LAG-$\bar{s}$) and the two-stage strategy that first predicts offsets with a time-series model and then applies a noise-robust offline RL algorithm (DMBP+LAG). Its advantage is consistent across previously unseen non-stationary perturbations from five domains, each introducing a distinct univariate series into a separate state dimension at test time. We present the average normalized scores over the prediction length P across multiple episodes run in the D4RL [15] and OGBench [21] for each time-series in Table 1. We conduct pairwise Welch's t-tests across all settings. Figure 7 plots the ℓ_2 norm between the ground-truth states s_t and those predicted by FORL and the baselines in the `antmaze` and `maze2d` environments. Consistent with the average scores, FORL achieves the lowest prediction error on average.

3.1.1 No Access to Past Offsets

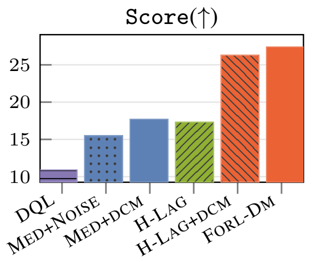

Figure 8: DM Ablations

We evaluate different variants of using DM and *Zero-Shot FM* when we do not have any access to past offsets in Fig. 8. **FORL-DM (DM):** Diffusion Model utilizes the candidate states generated by the FORL's diffusion model component (Section 2.2), which can be a multimodal distribution (Fig. 3). Compared to DM, the full FORL framework yields a 97.8% relative performance improvement. Notably, DM performs on par with our extended baselines that incorporate historical offsets and forecasting—DMBP+LAG, DQL+LAG-$\bar{s}$, and DQL+LAG-$\tilde{s}$. Moreover, without access to historical offset information before evaluation, DM achieves a 151.4% improvement over DQL, demonstrating its efficacy as a standalone module trained solely on a standard, stationary offline RL dataset without offset labels. **H-LAG:** We maintain a history of offsets generated by DM over the most recent C episodes (excluding the evaluation interval P, since offsets are not revealed after episode termination at test-time). We then feed this history into the *Zero-Shot FM* to generate offset samples for the next P evaluation episodes. These samples are applied directly at test time. **H-LAG+DCM:** We initially follow the same procedure in H-LAG to obtain predictions from *Zero-Shot FM*. Then, we apply **DCM** to these predicted offsets and the candidate states generated by FORL's diffusion model. We also compare against MED+DCM and MED+NOISE, simpler median-based heuristics detailed in Section G. Empirically, H-LAG+DCM outperforms H-LAG, demonstrating that DCM with FORL's diffusion model can improve robustness. Overall, scores and prediction errors indicate that just using the samples from DM has better scores on average, while H-LAG+DCM is more stable in Fig. 15.

3.1.2 Dimension-wise Closest Match (DCM) Ablations

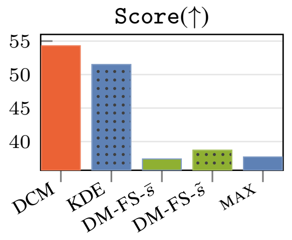

Figure 9: Candidate Selection

We compare FORL (DCM) against four alternative fusion strategies. FORL(KDE): For each dimension, we fit a kernel density estimator (KDE) on $\mathcal{D}_{\text{diffusion}} = \{s_t^{(0)}\}_k$ and then we evaluate that probability density function for each point in $\mathcal{D}_{\text{timeseries}}$. Then, we take the product of these densities in each dimension to obtain the weight for each sample $\hat{s}_t^b$. We obtain a single representative sample by taking the weighted average of samples in $\mathcal{D}_{\text{timeseries}}$. To ensure stability, when the sum of the weights is near zero, we use the mean of the $\mathcal{D}_{\text{timeseries}}$ as the states. We use Scott's rule [24] to compute the bandwidth. DM-FS-$\bar{s}$, DM-FS-$\tilde{s}$ select the closest prediction from DM to the mean and median of the *Zero-Shot FM*'s predictions, respectively. FORL (MAX) constructs a diagonal multivariate distribution from the dimension-wise mean and standard deviation of the

Table 2: **Normalized scores (mean ± std.) for FORL and baselines on `maze2d-large`.** Bolds denote the best scores and those not significantly different (Welch's t-test, $p > 0.05$). Suffixes -T and -R denote the use of TD3+BC [2] and RORL [25], respectively.

	TD3BC Policy				RORL Policy			
`maze2d-large`	TD3BC	TD3BC+LAG-$\bar{s}$	DMBP+LAG-T	FORL (ours)-T	RORL	RORL+LAG-$\bar{s}$	DMBP+LAG-R	FORL (ours)-R
`real-data-A`	**14.7** $\pm$ **5.7**	2.5 $\pm$ 2.7	4.8 $\pm$ 3.9	**20.7** $\pm$ **3.5**	12.2 $\pm$ 2.3	13.0 $\pm$ 2.0	4.3 $\pm$ 5.0	**56.9** $\pm$ **3.0**
`real-data-B`	-0.9 $\pm$ 2.0	4.6 $\pm$ 8.9	11.7 $\pm$ 12.6	**56.8** $\pm$ **14.4**	1.2 $\pm$ 5.5	13.1 $\pm$ 14.7	28.5 $\pm$ 11.7	**98.5** $\pm$ **19.0**
`real-data-C`	0.8 $\pm$ 1.9	21.6 $\pm$ 8.4	29.5 $\pm$ 13.7	**56.9** $\pm$ **14.6**	3.1 $\pm$ 0.9	60.6 $\pm$ 8.5	39.4 $\pm$ 6.1	**139.0** $\pm$ **15.1**
`real-data-D`	2.5 $\pm$ 4.4	14.9 $\pm$ 4.3	14.4 $\pm$ 6.8	**29.5** $\pm$ **10.3**	-1.6 $\pm$ 0.7	17.9 $\pm$ 6.8	17.5 $\pm$ 6.0	**33.1** $\pm$ **2.3**
`real-data-E`	-2.3 $\pm$ 0.2	1.0 $\pm$ 4.2	2.0 $\pm$ 3.9	**8.0** $\pm$ **4.2**	-0.9 $\pm$ 2.0	3.3 $\pm$ 4.4	2.2 $\pm$ 4.5	**32.2** $\pm$ **15.3**
Average	3.0	8.9	12.5	**34.4**	2.8	21.6	18.4	**71.9**

forecasted states, then selects the sample predicted by our diffusion model with the highest likelihood under that distribution. Although all baselines fuse information using the same two sets generated by the diffusion model and *Zero-Shot FM*, DCM has higher performance. In Table 8 we compute the maximum, minimum, and mean prediction error values over the test episodes used in Fig. 6. FORL (DCM) yields significantly stable prediction errors (`Maximum Error` $\downarrow$:**2.40**) for both maximum error and mean error compared to FORL (MAX) (`Maximum Error` $\downarrow$:**9.33**) demonstrating its robustness.

3.1.3 Intra-episode Non-stationarity

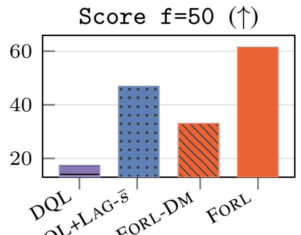

Figure 10: Intra-Episode Performance

Our framework can natively handle intra-episode offsets, where the offset changes every $f = 50$ timesteps. In this setting, the offsets become available after the episode terminates, but the agent is subject to a time-dependent unknown offset within the episode. Zero-shot forecasting foundation module can generate samples before the episode begins. Our diffusion model (FORL-DM) itself does not rely on the forecasts of the foundation module and only tracks observation changes and actions which are invariant to the offsets. The DCM can adaptively fuse information from both models at each timestep without requiring any hyperparameters. Table 10 and Fig. 10 show the average scores for DQL vs. FORL-DM and DQL+LAG-$\bar{s}$ vs. FORL. Among the algorithms that do not use any past ground truth offsets DQL and FORL-DM, only using the diffusion model of FORL significantly increases performance. When we have access to past offsets, FORL obtains a superior performance. This shows that our method covers both cases, when information is available and not available, even when offsets are not constant throughout the episode.

3.1.4 Offset-Scaling

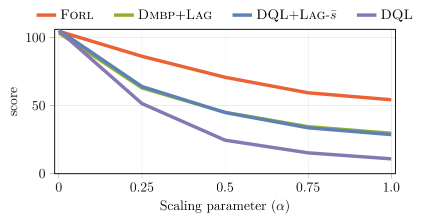

Figure 11: Impact of offset scaling (α) on average normalized scores.

We scale the offsets with α across all maze experiments. We conduct experiments in 5 environments (all antmaze and maze2d used in Table 1) across 5 time-series dataset setups with $\alpha \in \{0, 0.25, 0.5, 0.75, 1.0\}$, where $\alpha = 0$ is the standard offline RL environment used during training and $\alpha = 1.0$ is our evaluation setup. The results show that FORL outperforms the baselines, confirming its robustness. Even a small scaling of 0.25 results in a sudden drop in performance, whereas FORL only experiences a gradual decrease in Figure 11. Detailed results for each environment and α pairs are provided in Appendix Figure 13.

3.1.5 Policy-Agnostic

In the `maze2d-large` experiments (in Table 2, `maze2d-medium` in Appendix Table 3), we use Robust Offline RL (RORL) [25], and TD3BC [2] offline RL algorithms instead of DQL [14], to analyze the effect of offline RL policy choice during evaluation. RORL+LAG-$\bar{s}$ and TD3BC+LAG-$\bar{s}$ extend RORL and TD3BC by using the sample mean of the forecasted states $\{\hat{s}_t\}_l$ at each timestep (using the constant per-episode predicted b^j). Results indicate that using a robust offline RL algorithm during training significantly increases performance (71.9) compared to DQL (38.8) and TD3BC (34.4) at test time when used with FORL, no increase when used alone, and a marginal increase with Lag-Llama and DMBP+LAG. We observe similar performance gains when applying FORL to other policies (Section E), including Implicit Q-Learning (IQL) [26] and FQL [23], as detailed in Table 4 and Table 7.

4 Related Work

Reinforcement Learning in Non-Stationary Environments Existing works in non-stationary reinforcement learning (RL) predominantly focus on adapting to changing transition dynamics and reward functions. Ackermann et al. [27] propose an offline RL framework that incorporates structured non-stationarity in reward and transition functions by learning hidden task representations and predicting them at test time. Although our work also investigates the intersection of non-stationary environments and offline RL, we assume stationarity during training. To learn adaptive policies *online*, meta-learning algorithms have been proposed as a promising approach [28–30]. Al-Shedivat et al. [29] explores a competitive multi-agent environment where transition dynamics change. While these approaches provide valuable insights, they often require samples from the current environment and struggle in non-trivial non-stationarity, highlighting the need for more future-oriented methods [9, 31]. Examples of such future-oriented approaches include Proactively Synchronizing Tempo (ProST) [9] and Prognosticator [31], which address the evolution of transition and reward functions over time. ProST leverages a forecaster, namely, Auto-Regressive Integrated Moving Average (ARIMA), and a model predictor to optimize for future policies in environments to overcome the time-synchronization issue in time-elapsing MDPs. This approach aligns with our focus on time-varying environments and similarly utilizes real-world finance (e.g., stock price) time-series datasets to model non-stationarity. Both ProST and Prognosticator assume that states are fully observable during testing and that online interaction with the environment is possible during training—conditions that are not always feasible in the real world. Instead, our approach assumes that states are not fully observable and that direct interaction with the environment during training is not feasible, necessitating that the policy be learned exclusively from a pre-collected dataset.

Robust offline RL Testing-time robust offline RL methods DMBP [3], RORL [25] examine scenarios where a noise-free, stationary dataset is used for training, but corruption is introduced during testing. This is distinct from [3], training-time robust offline RL [32, 33], which assumes a corrupted training dataset. Both RORL [25] and DMBP [3] assume access only to a clean, uncorrupted offline RL dataset, as FORL, and they are evaluated in a perturbed environment. To the best of our knowledge, FORL is the first work to extend this setting to a non-Markovian, time-evolving, non-stationary deployment environment. We focus on time-dependent exogenous factors from real-data that are aligned with the definition of a non-stationary environment [6].

Diffusion models in offline RL Diffusion models [12] have seen widespread adoption in RL [34, 35] due to their remarkable expressiveness, particularly in representing multimodal distributions, scalability, and stable training properties. In the context of offline RL, diffusion models have been used for representing policies [14, 36–38], planners [39, 40], data synthesis [41, 42], and removing noise [3]. Notably, Diffusion Q-learning [14] leverages conditional diffusion model policies to learn from offline RL datasets, maintaining proximity to behavior policy while utilizing Q-value function guidance. In contrast, our method harnesses diffusion models to learn from a sequence of actions and effect tuples, leveraging the multimodal capabilities of diffusion models to identify diverse candidate locations of the hidden states.

5 Conclusion

We introduce **F**orecasting in Non-stationary **O**ffline **RL** (FORL), a novel framework designed to be robust to passive non-stationarities that arise at test time. This is crucial when an agent trained on an offline RL dataset is deployed in a non-stationary environment or when the environment begins to exhibit partial observability due to unknown, time-varying factors. FORL leverages diffusion probabilistic models and zero-shot time series foundation models to correct unknown offsets in observations, thereby enhancing the adaptability of learned policies. Our empirical results across diverse time-series datasets, OGBench [21] and D4RL [15] benchmarks, demonstrate that FORL not only bridges the gap between forecasting and non-stationary offline RL but also consistently outperforms the baselines. Our approach is currently limited by the assumption of additive perturbations. For future work, we plan to extend our work to more general observation transformations.

Acknowledgments and Disclosure of Funding

Georg Martius is a member of the Machine Learning Cluster of Excellence, EXC number 2064/1 – Project number 390727645. Co-funded by the European Union (ERC, REAL-RL, 101045454). Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. This work was supported by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A. This work was in part supported by the INVERSE project (101136067) funded by the European Union and JSPS KAKENHI Grant Numbers JP23K24926, JP25H01236. The numerical calculations reported in this paper were partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources). The authors would like to thank René Geist, Tomáš Daniš, Ji Shi, and Leonard Franz for their valuable comments on the manuscript.

References

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

[2] Scott Fujimoto and Shixiang Gu. A minimalist approach to offline reinforcement learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[3] YANG Zhihe and Yunjian Xu. Dmbp: Diffusion model-based predictor for robust offline reinforcement learning against state observation perturbations. In *The Twelfth International Conference on Learning Representations*, 2024.

[4] Brandon Trabucco, Mariano Phielipp, and Glen Berseth. Anymorph: Learning transferable polices by inferring agent morphology. In *International Conference on Machine Learning*, pages 21677–21691. PMLR, 2022.

[5] Amy Zhang, Clare Lyle, Shagun Sodhani, Angelos Filos, Marta Kwiatkowska, Joelle Pineau, Yarin Gal, and Doina Precup. Invariant causal prediction for block mdps. In *International Conference on Machine Learning*, pages 11214–11224. PMLR, 2020.

[6] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75: 1401–1476, 2022.

[7] Yash Chandak. *Reinforcement Learning for Non-stationary problems*. PhD thesis, PhD thesis, University of Massachusetts Amherst, 2022.

[8] Annie Xie, James Harrison, and Chelsea Finn. Deep reinforcement learning amidst continual structured non-stationarity. In *International Conference on Machine Learning*, pages 11393–11403. PMLR, 2021.

[9] Hyunin Lee, Yuhao Ding, Jongmin Lee, Ming Jin, Javad Lavaei, and Somayeh Sojoudi. Tempo adaptation in non-stationary reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[10] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-llama: Towards foundation models for time series forecasting. *arXiv preprint arXiv:2310.08278*, 2023.

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[12] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *CoRR*, abs/1503.03585, 2015. URL https://arxiv.org/abs/1503.03585.

[13] Calvin Luo. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022. URL https://arxiv.org/abs/2208.11970.

[14] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

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

[16] 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.

[17] Blai Bonet. Deterministic pomdps revisited. *arXiv preprint arXiv:1205.2659*, 2012. URL https://arxiv.org/abs/1205.2659.

[18] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[19] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021.

[20] Calvin Luo. Understanding diffusion models: A unified perspective, 2022. URL https://arxiv.org/abs/2208.11970.

[21] Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=M992mjgKzI.

[22] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Türkmen, and Yuyang Wang. GluonTS: Probabilistic and Neural Time Series Modeling in Python. *Journal of Machine Learning Research*, 21(116): 1–6, 2020.

[23] Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=KVf2SFL1pi.

[24] David W Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[25] Rui Yang, Chenjia Bai, Xiaoteng Ma, Zhaoran Wang, Chongjie Zhang, and Lei Han. Rorl: Robust offline reinforcement learning via conservative smoothing. *Advances in neural information processing systems*, 35:23851–23866, 2022.

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

[27] Johannes Ackermann, Takayuki Osa, and Masashi Sugiyama. Offline reinforcement learning from datasets with structured non-stationarity. In *Reinforcement Learning Conference*, 2024.

[28] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *International conference on machine learning*, pages 1920–1930. PMLR, 2019.

[29] Maruan Al-Shedivat, Trapit Bansal, Yura Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations*, 2018.

[30] Suzan Ece Ada and Emre Ugur. Unsupervised meta-testing with conditional neural processes for hybrid meta-reinforcement learning. *IEEE Robotics and Automation Letters*, 9(10):8427–8434, 2024. doi: 10.1109/LRA.2024.3443496.

[31] Yash Chandak, Georgios Theocharous, Shiv Shankar, Martha White, Sridhar Mahadevan, and Philip Thomas. Optimizing for the future in non-stationary mdps. In *International Conference on Machine Learning*, pages 1414–1425. PMLR, 2020.

[32] Chenlu Ye, Rui Yang, Quanquan Gu, and Tong Zhang. Corruption-robust offline reinforcement learning with general function approximation. *Advances in Neural Information Processing Systems*, 36:36208–36221, 2023.

[33] Xuezhou Zhang, Yiding Chen, Xiaojin Zhu, and Wen Sun. Corruption-robust offline reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 5757–5773. PMLR, 2022.

[34] Zhengbang Zhu, Hanye Zhao, Haoran He, Yichao Zhong, Shenyu Zhang, Yong Yu, and Weinan Zhang. Diffusion models for reinforcement learning: A survey. *arXiv preprint arXiv:2311.01223*, 2023.

[35] Jiayu Chen, Bhargav Ganguly, Yang Xu, Yongsheng Mei, Tian Lan, and Vaneet Aggarwal. Deep generative models for offline policy learning: Tutorial, survey, and perspectives on future directions. *arXiv preprint arXiv:2402.13777*, 2024.

[36] Longxiang He, Linrui Zhang, Junbo Tan, and Xueqian Wang. Diffcps: Diffusion model based constrained policy search for offline reinforcement learning. *arXiv preprint arXiv:2310.05333*, 2023.

[37] Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023. URL https://arxiv.org/abs/2304.10573.

[38] Suzan Ece Ada, Erhan Oztop, and Emre Ugur. Diffusion policies for out-of-distribution generalization in offline reinforcement learning. *IEEE Robotics and Automation Letters*, 9(4): 3116–3123, 2024. doi: 10.1109/LRA.2024.3363530.

[39] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

[40] Matthew Coleman, Olga Russakovsky, Christine Allen-Blanchette, and Ye Zhu. Discrete diffusion reward guidance methods for offline reinforcement learning. In *ICML 2023 Workshop: Sampling and Optimization in Discrete Space*, 2023.

[41] Haoran He, Chenjia Bai, Kang Xu, Zhuoran Yang, Weinan Zhang, Dong Wang, Bin Zhao, and Xuelong Li. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 64896–64917. Curran Associates, Inc., 2023.

[42] Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. Adaptdiffuser: Diffusion models as adaptive self-evolving planners. *arXiv preprint arXiv:2302.01877*, 2023. URL https://arxiv.org/abs/2302.01877.

[43] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2052–2062. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/fujimoto19a.html.

[44] Noah Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *International Conference on Learning Representations*, 2020.

[45] Yecheng Jason Ma, Dinesh Jayaraman, and Osbert Bastani. Conservative offline distributional reinforcement learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=Z2vksUFuVst`.

[46] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019. URL `https://arxiv.org/abs/1907.00456`.

[47] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019. URL `https://arxiv.org/abs/1911.11361`.

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

[49] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[50] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29, 2016.

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

[52] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. COMBO: conservative offline model-based policy optimization. *CoRR*, abs/2102.08363, 2021. URL `https://arxiv.org/abs/2102.08363`.

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

[54] Bogdan Mazoure, Ilya Kostrikov, Ofir Nachum, and Jonathan J Tompson. Improving zero-shot generalization in offline reinforcement learning using generalized similarity functions. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 25088–25101. Curran Associates, Inc., 2022.

[55] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[56] Zhihe Yang and Yunjian Xu. DMBP: Diffusion model-based predictor for robust offline reinforcement learning against state observation perturbations (official implementation). `https://github.com/zhyang2226/DMBP/tree/main`, 2024.

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

[58] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[59] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[60] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750, 2007.

[61] Qing Wang, Jiechao Xiong, Lei Han, peng sun, Han Liu, and Tong Zhang. Exponentially weighted imitation learning for batched historical data. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/4aec1b3435c52abbdf8334ea0e7141e0-Paper.pdf`.

[62] Denis Tarasov, Alexander Nikulin, Dmitry Akimov, Vladislav Kurenkov, and Sergey Kolesnikov. CORL: Research-oriented deep offline reinforcement learning library. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*, 2022. URL `https://openreview.net/forum?id=SyAS49bBcv`.

[63] Garrett Thomas. Implicit q-learning (iql) in pytorch. `https://github.com/gwthomas/IQL-PyTorch`, 2021.

[64] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[65] Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, and qiang liu. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=1k4yZbbDqX`.

[66] Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024.

[67] Zihan Ding, Chi Jin, Difan Liu, Haitian Zheng, Krishna Kumar Singh, Qiang Zhang, Yan Kang, Zhe Lin, and Yuchen Liu. Dollar: Few-step video generation via distillation and latent reward optimization. *arXiv preprint arXiv:2412.15689*, 2024.

[68] Jiachen Li, Weixi Feng, Wenhu Chen, and William Yang Wang. Reward guided latent consistency distillation. *arXiv preprint arXiv:2403.11027*, 2024.

[69] Elvezio M Ronchetti and Peter J Huber. *Robust statistics*. John Wiley & Sons Hoboken, NJ, USA, 2009.

[70] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6): 381–395, June 1981. ISSN 0001-0782.

[71] B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4:419–420, 1962.

[72] Donald Ervin Knuth. *The art of computer programming*, volume 3. Pearson Education, 1997.

[73] Rakshitha Wathsadini Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[74] Dua Dheeru and Efi Karra Taniskidou. Uci machine learning repository. `http://archive.ics.uci.edu/ml`, 2017.

[75] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. High-dimensional multivariate forecasting with low-rank gaussian copula processes. *Advances in neural information processing systems*, 32, 2019.

[76] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

[77] A. Alexandrov, K. Benidis, M. Bohlke-Schneider, V. Flunkert, J. Gasthaus, T. Januschowski, D. C. Maddix, S. Rangapuram, D. Salinas, J. Schulz, L. Stella, A. C. Türkmen, and Y. Wang. GluonTS: Probabilistic Time Series Modeling in Python. *arXiv preprint arXiv:1906.05264*, 2019.

[78] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[79] Zhendong Wang. Diffusion policies for offline rl — official pytorch implementation. `https://github.com/Zhendong-Wang/Diffusion-Policies-for-Offline-RL`, 2023.

[80] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *ICLR (Poster)*, 2015. URL `http://dblp.uni-trier.de/db/conf/iclr/iclr2015.html#KingmaB14`.

[81] Ilya Kostrikov. Offline reinforcement learning with implicit q-learning (official implementation). `https://github.com/ikostrikov/implicit_q_learning`, 2021.

[82] Seohong Park. Fql: Flow q-learning (official implementation). `https://github.com/seohongpark/fql`, 2025.

[83] Scott Fujimoto. A minimalist approach to offline reinforcement learning pytorch implementation. `https://github.com/sfujim/TD3_BC`, 2018.

[84] Rui Yang. Rorl: Robust offline reinforcement learning via conservative smoothing code repository. `https://github.com/YangRui2015/RORL`, 2022.

[85] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[86] Artur Trindade. ElectricityLoadDiagrams20112014. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C58C86.

[87] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(03):90–95, 2007.

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 claims made in the abstract and introduction are supported by experimental results and compared to the baselines.

   Guidelines:

   - The answer NA 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 NA 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: Limitations are discussed.

   Guidelines:

   - The answer NA 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: [NA]

Justification: Our paper does not include theoretical results.

Guidelines:

- The answer NA 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: We disclose all datasets and environments used (all publicly available), the model architecture hyperparameters, implementation details, and pseudocode of the algorithms.

   Guidelines:

   - The answer NA 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: [No]

Justification: We plan to provide open access to code in the future.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

Answer: [Yes]

Justification:We specify all training and test details.

Guidelines:

- The answer NA 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: We report statistical significance tests in this paper.

Guidelines:

- The answer NA 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: [Yes]

   Justification: We provide computational hardware required to reproduce the experiments.

   Guidelines:

   - The answer NA 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 research conducted fully conforms with the NeurIPS Ethics Guidelines

   Guidelines:

   - The answer NA 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: [NA]

    Justification: To the best of our knowledge there are no direct negative societal impacts of our work.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work does not involve releasing any high risk models.

    Guidelines:

    - The answer NA 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: [Yes]

    Justification: We have cited the original owners and included the licenses.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: Our paper does not introduce a new dataset.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: Our work does not involve crowdsourcing experiments nor research with human subjects.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: Our work does not involve crowdsourcing experiments nor research with human subjects.

    Guidelines:

    - The answer NA 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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: We did not use LLMs for important, original, or non-standard component of the core methods in this research.

Guidelines:

- The answer NA 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 (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.